Guia do Custom Labels

Rekognition

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Rekognition: Guia do Custom Labels

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é o Amazon Rekognition Custom Labels? ... 1
Benefícios principais ... 2
Escolha de usar o Amazon Rekognition Custom Labels ... 2
Detecção de rótulos do Amazon Rekognition Image ... 3
Amazon Rekognition Custom Labels ... 3
Você é um usuário iniciante do Amazon Rekognition Custom Labels? ... 4
Como configurar o Amazon Rekognition Custom Labels ... 5
Etapa 1: criar uma AWS conta ... 5
Inscreva-se para um Conta da AWS ... 6
Criar um usuário com acesso administrativo ... 6
Acesso programático ... 7
Etapa 2: configure as permissões do console ... 9
Como permitir o acesso ao console ... 9
Como acessar os buckets externos do Amazon S3 ... 11
Como atribuir permissões ... 11
Etapa 3: crie um bucket do console ... 12
Etapa 4: configurar o AWS CLI e AWS SDKs ... 13
Instale os AWS SDKS ... 13
Conceder acesso programático ... 7
Configurar permissões do SDK ... 17
Chamar uma operação ... 19
Etapa 5: (opcional) criptografe os arquivos de treinamento ... 23
Descriptografando arquivos criptografados com AWS Key Management Service ... 23
Como criptografar imagens copiadas de treinamento e teste ... 24
Etapa 6: (opcional) associe conjuntos de dados anteriores ... 24
Como usar um conjunto de dados anterior como um conjunto de dados de teste ... 25
Noções básicas do Amazon Rekognition Custom Labels ... 27
Decida o tipo do seu modelo ... 27
Encontre objetos, cenas e conceitos ... 28
Encontre localizações de objetos ... 29
Encontre a localização das marcas ... 29
Criar um modelo ... 30
Criar um projeto ... 30
Crie conjuntos de dados de treinamento e teste ... 31

Treinar seu modelo ... 32
Melhore seu modelo ... 33
Avalie seu modelo ... 33
Melhore seu modelo ... 34
Executar seu modelo ... 34
Executar seu modelo (console) ... 35
Executar seu modelo ... 35
Analisar uma imagem ... 35
Interrompa seu modelo ... 37
Interrompa seu modelo (console) ... 37
Interrompa seu modelo (SDK) ... 37
Conceitos básicos ... 38
Tutoriais em vídeo ... 38
Projetos de exemplo ... 39
Classificação de imagens ... 39
Classificação de imagens com vários rótulos ... 39
Detecção de marca ... 40
Localização de objetos ... 40
Como usar os projetos de exemplo ... 41
Como criar o projeto de exemplo ... 41
Como treinar o modelo ... 42
Como usar o modelo ... 42
Próximas etapas ... 42
Etapa 1: escolha um projeto de exemplo ... 42
Etapa 2: treine seu modelo ... 45
Etapa 3: inicie seu modelo ... 50
Etapa 4: analise uma imagem com seu modelo ... 51
Como obter uma imagem de exemplo ... 56
Etapa 5: interrompa seu modelo ... 58
Etapa 6: próximas etapas ... 60
Classificar imagens ... 62
Etapa 1: colete suas imagens ... 62
Etapa 2: decida suas classes ... 64
Etapa 3: crie um projeto ... 64
Etapa 4: crie conjuntos de dados de treinamento e teste ... 65
Etapa 5: adicione rótulos ao projeto ... 71

Etapa 6: atribuia rótulos em nível de imagem aos conjuntos de dados de treinamento e teste 72
Etapa 7: treine seu modelo 73
Etapa 8: inicie seu modelo 78
Etapa 9: analise uma imagem com seu modelo 80
Etapa 10: interrompa seu modelo 83
Como criar um modelo 86
Como criar um projeto 86
Criar um projeto (console) 87
Como criar um projeto (SDK) 87
Formato da solicitação de criação do projeto 92
Criar conjuntos de dados 92
Como definir os conjuntos de dados 94
Como preparar imagens 100
Como criar conjuntos de dados com imagens 101
Rotulagem de imagens 164
Como depurar conjuntos de dados 174
Como treinar um modelo 181
Como treinar um modelo (console) 183
Treinando um modelo (SDK) 188
Como depurar o treinamento de modelos 198
Erros terminais 198
Lista de erros não terminais de validação de linha JSON 201
Noções básicas sobre o resumo do manifesto 202
Noções básicas sobre treinar e testar manifestos de resultados de validação 206
Como obter os resultados de validação 211
Como corrigir erros de treinamento 214
Erros terminais do arquivo de manifesto 216
Erros terminais de conteúdo do manifesto 218
Erros não terminais de validação de linha JSON 228
Como melhorar um modelo treinado 253
Métricas para avaliar seu modelo 253
Como avaliar o desempenho do modelo 254
Limite assumido 255
Precisão 255
Recall 256
F1 256

Uso de métricas do ... 257
Como acessar as métricas de avaliação (console) ... 257
Como acessar as métricas de avaliação (SDK) ... 260
Acessar o arquivo de resumo do modelo ... 261
Interpretar o snapshot do manifesto de avaliação ... 263
Como acessar o arquivo de resumo e o snapshot do manifesto de avaliação (SDK) ... 267
Como visualizar a matriz de confusão para um modelo ... 268
Referência: arquivo de resumo ... 275
Como melhorar um modelo ... 277
Dados ... 277
Como reduzir falsos positivos (melhor precisão) ... 278
Como reduzir falsos negativos (melhor recall) ... 278
Como executar um modelo treinado ... 279
Unidades de inferência ... 279
Como gerenciar o throughput com unidades de inferência ... 280
Zonas de disponibilidade ... 282
Como iniciar um modelo ... 283
Como iniciar ou interromper um modelo (console) ... 283
Como iniciar um modelo (SDK) ... 284
Como interromper um modelo ... 294
Como interromper um modelo (console) ... 294
Como interromper um modelo (SDK) ... 295
Duração do relatório e unidades de inferência ... 304
Como analisar uma imagem com um modelo treinado ... 307
DetectCustomLabels solicitação de operação ... 334
DetectCustomLabels resposta da operação ... 334
Gerenciar recursos ... 335
Como gerenciar um projeto ... 335
Excluir um projeto ... 336
Como descrever um projeto (SDK) ... 346
Criando um projeto com AWS CloudFormation ... 353
Como gerenciar conjuntos de dados ... 354
Como adicionar um conjunto de dados ... 354
Como adicionar mais imagens ... 364
Como criar um conjunto de dados usando um conjunto de dados existente (SDK) ... 374
Como descrever um conjunto de dados (SDK) ... 383

Como listar entradas do conjunto de dados (SDK) ... 388
Como distribuir um conjunto de dados de treinamento (SDK) ... 394
Como excluir um conjunto de dados ... 404
Como gerenciar um modelo ... 411
Excluir um modelo ... 412
Como marcar um modelo ... 421
Como descrever um modelo (SDK) ... 428
Como copiar um modelo (SDK) ... 436
Exemplos de rótulos personalizados ... 474
Melhorar um modelo com Model Feedback ... 474
Demonstração do Amazon Rekognition Custom Labels ... 475
Detectar rótulos personalizados em vídeos ... 475
Analisando imagens com uma AWS Lambda função ... 478
Etapa 1: criar uma AWS Lambda função (console) ... 478
Etapa 2: (opcional) crie uma camada (console) ... 481
Etapa 3: adicione o código em Python (console) ... 482
Etapa 4: teste sua função do Lambda ... 485
Segurança ... 490
Como proteger projetos do Amazon Rekognition Custom Labels ... 490
Protegendo DetectCustomLabels ... 491
Políticas gerenciadas pela AWS ... 492
Diretrizes e cotas ... 493
Regiões compatíveis ... 493
Cotas ... 493
Treinamento ... 493
Teste ... 494
Detecção ... 495
Cópia do modelo ... 495
Referência da API ... 496
Como treinar seu modelo ... 507
Projetos ... 507
Políticas do projeto ... 507
Conjuntos de dados ... 507
Modelos da ... 508
Tags ... 507
Como usar seu modelo ... 508

Histórico de documentos 509
..... dxvi

O que é o Amazon Rekognition Custom Labels?

Com o Amazon Rekognition Custom Labels, é possível identificar os objetos, os logotipos e as cenas nas imagens que são específicos das necessidades dos seus negócios. Por exemplo, é possível encontrar seu logotipo em postagens de mídias sociais, identificar seus produtos nas prateleiras das lojas, diferenciar plantas saudáveis ou infectadas, classificar as peças de máquina em uma linha de montagem ou detectar personagens animados em imagens.

Desenvolver um modelo personalizado para analisar imagens é uma tarefa importante que exige tempo, experiência e recursos. Geralmente, leva meses para ser concluído. Além disso, pode exigir milhares ou dezenas de milhares de rótulos etiquetados à mão para fornecer ao modelo dados suficientes para tomar decisões com precisão. A geração desses dados pode levar meses para ser reunida e pode exigir que grandes equipes de rotuladores os preparem para uso em machine learning.

O Amazon Rekognition Custom Labels amplia os recursos existentes do Amazon Rekognition, que já são treinados em dezenas de milhões de imagens em várias categorias. Em vez de milhares de imagens, é possível fazer upload de um pequeno conjunto de imagens de treinamento (normalmente algumas centenas de imagens ou menos) que são específicas para seu caso de uso. Você pode fazer isso usando o easy-to-use console. Se suas imagens já estiverem rotuladas, o Amazon Rekognition Custom Labels pode começar a treinar um modelo em pouco tempo. Caso contrário, você pode rotular as imagens diretamente na interface de rotulagem ou usar o Amazon SageMaker AI Ground Truth para rotulá-las para você.

Depois que o Amazon Rekognition Custom Labels começar a treinar com seu conjunto de imagens, ele poderá produzir um modelo de análise de imagem personalizado para você em apenas algumas horas. Nos bastidores, o Amazon Rekognition Custom Labels carrega e inspeciona automaticamente os dados de treinamento, seleciona os algoritmos de machine learning corretos, treina um modelo e fornece métricas de desempenho do modelo. Em seguida, é possível usar seu modelo personalizado por meio da API Amazon Rekognition Custom Labels e integrá-lo às suas aplicações.

Tópicos

- Benefícios principais
- Escolha de usar o Amazon Rekognition Custom Labels
- Você é um usuário iniciante do Amazon Rekognition Custom Labels?

Benefícios principais

Rotulagem de dados simplificada

O console do Amazon Rekognition Custom Labels fornece uma interface visual para tornar a rotulagem de suas imagens rápida e simples. A interface permite que você aplique um rótulo à imagem inteira. Você também pode identificar e rotular objetos específicos em imagens usando caixas delimitadoras com uma click-and-drag interface. Como alternativa, se você tiver um grande conjunto de dados, poderá usar o Amazon SageMakerGround Truth para rotular com eficiência suas imagens em grande escala.

Machine learning automatizado

Não é necessário ter experiência em machine learning para criar seu modelo personalizado. O Amazon Rekognition Custom Labels inclui recursos de machine learning automatizado (AutoML) que cuidam do machine learning para você. Quando as imagens de treinamento são fornecidas, o Amazon Rekognition Custom Labels carrega e inspeciona automaticamente os dados de treinamento, seleciona os algoritmos de machine learning corretos, treina um modelo e fornece métricas de desempenho do modelo.

Avaliação, inferência e feedback simplificados do modelo

O desempenho do seu modelo personalizado é avaliado em seu conjunto de testes. Para cada imagem no conjunto de teste, você pode ver a side-by-side comparação da previsão do modelo com a etiqueta atribuída. Você também pode revisar métricas de desempenho detalhadas, como precisão, recall, pontuações F1 e pontuações de confiança. É possível começar a usar seu modelo imediatamente para análise de imagens ou pode iterar e retreinar novas versões com mais imagens para melhorar o desempenho. Depois de começar a usar seu modelo, você rastreia suas previsões, corrige quaisquer erros e usa os dados de feedback para treinar novas versões do modelo e melhorar o desempenho.

Escolha de usar o Amazon Rekognition Custom Labels

O Amazon Rekognition fornece dois atributos que podem ser usados para encontrar rótulos (objetos, cenas e conceitos) em imagens: Amazon Rekognition Custom Labels e Amazon Rekognition Image Labels Detection. Use as informações a seguir para determinar qual atributo você deve usar.

Detecção de rótulos do Amazon Rekognition Image

É possível usar o atributo de detecção de rótulos no Amazon Rekognition Image para identificar, classificar e pesquisar rótulos comuns em imagens e vídeos, em grande escala e sem precisar criar um modelo de machine learning. Por exemplo, é possível detectar facilmente milhares de objetos comuns, como carros e caminhões, tomates, bolas de basquete e bolas de futebol.

Se sua aplicação precisar encontrar rótulos comuns, recomendamos usar a detecção de rótulos do Amazon Rekognition Image, pois você não precisa treinar um modelo. Para obter uma lista dos rótulos encontrados pela detecção de rótulos do Amazon Rekognition Image, consulte Como detectar rótulos.

Se seu aplicativo precisar encontrar rótulos não encontrados pela detecção de rótulos do Amazon Rekognition Image, como peças personalizadas de máquinas em uma linha de montagem, recomendamos que você use o Amazon Rekognition Custom Labels.

Amazon Rekognition Custom Labels

É possível usar o Amazon Rekognition Custom Labels para treinar facilmente um modelo de machine learning que encontre rótulos (objetos, logotipos, cenas e conceitos) em imagens exclusivas para suas necessidades comerciais.

O Amazon Rekognition Custom Labels pode classificar imagens (previsões em nível de imagem) ou detectar localizações de objetos em uma imagem (previsões em nível de objeto/caixa delimitadora).

O Amazon Rekognition Custom Labels oferecem uma maior flexibilidade nos tipos de objetos e cenas que podem ser detectadas. Por exemplo, é possível usar a detecção de rótulos do Amazon Rekognition Image para encontrar plantas e folhas. Para distinguir entre plantas saudáveis, danificadas e infectadas, você precisa usar o Amazon Rekognition Custom Labels.

A seguir, veja exemplos de como utilizar o Amazon Rekognition Custom Labels.

- Identificar os logotipos da equipe nas camisas e capacetes dos jogadores
- Distinguir entre peças ou produtos específicos de máquinas em uma linha de montagem
- Identificar personagens de desenhos animados em uma biblioteca de mídia
- Localize produtos de uma marca específica nas prateleiras do varejo
- Classifique a qualidade dos produtos agrícolas (como podres, maduros ou crus)

> **Note**
>
> O Amazon Rekognition Custom Labels não foi projetado para analisar rostos, detectar texto ou descobrir conteúdo de imagem não seguro em imagens. Para realizar essas tarefas, é possível usar o Amazon Rekognition Image. Para obter mais informações, consulte O que é o Amazon Rekognition.

Você é um usuário iniciante do Amazon Rekognition Custom Labels?

Se estiver usando o Amazon Rekognition Custom Labels pela primeira vez, recomendamos que leia as seguintes seções em ordem:

1. Como configurar o Amazon Rekognition Custom Labels: nesta seção, você define os detalhes da sua conta.
2. Noções básicas do Amazon Rekognition Custom Labels: nesta seção, você aprende sobre o fluxo de trabalho para criar um modelo.
3. Conceitos básicos do Amazon Rekognition Custom Labels: nesta seção, você treina um modelo usando exemplos de projetos criados pelo Amazon Rekognition Custom Labels.
4. Classificar imagens: nesta seção, você aprende a treinar um modelo que classifica imagens com conjuntos de dados criados por você.

Como configurar o Amazon Rekognition Custom Labels

As instruções a seguir mostram como configurar o console e o SDK do Amazon Rekognition Custom Labels.

Observe que é possível usar o console do Amazon Rekognition Custom Labels com os seguintes navegadores:

- Chrome: versão 21 ou posterior
- Firefox: versão 27 ou posterior
- Microsoft Edge: versão 88 ou posterior
- Safari: versão 7 ou posterior. Além disso, você não pode usar o Safari para desenhar caixas delimitadoras com o console do Amazon Rekognition Custom Labels. Para obter mais informações, consulte Como rotular objetos com caixas delimitadoras.

Antes de usar o Amazon Rekognition Custom Labels pela primeira vez, conclua as seguintes tarefas:

Tópicos

- Etapa 1: criar uma AWS conta
- Etapa 2: configure as permissões do console do Amazon Rekognition Custom Labels
- Etapa 3: crie um bucket do console
- Etapa 4: configurar o AWS CLI e AWS SDKs
- Etapa 5: (opcional) criptografe os arquivos de treinamento
- Etapa 6: (opcional) associe conjuntos de dados anteriores com novos projetos

Etapa 1: criar uma AWS conta

Nesta etapa, você cria uma AWS conta, cria um usuário administrativo e aprende a conceder acesso programático ao AWS SDK.

Tópicos

- Inscreva-se para um Conta da AWS
- Criar um usuário com acesso administrativo
- Acesso programático

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma ligação ou mensagem de texto e inserir um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira a senha.

   Para obter ajuda ao fazer login usando o usuário-raiz, consulte Fazer login como usuário-raiz no Guia do usuário do Início de Sessão da AWS .
2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

   Para obter instruções, consulte Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilita o Centro de Identidade do IAM.

   Para obter instruções, consulte Habilitar o AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .

2. No Centro de Identidade do IAM, conceda o acesso administrativo a um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com o seu usuário do Centro de Identidade do IAM, use o URL de login enviado ao seu endereço de e-mail quando o usuário do Centro de Identidade do IAM foi criado.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Criar um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Adicionar grupos no Guia do usuário do AWS IAM Identity Center .

Acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identidade da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de ferramentas AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para ferramentas AWS SDKs e ferramentas,

Qual usuário precisa de acesso programático?	Para	Por
		consulte Autenticar usando credenciais de longo prazo no Guia de referência de ferramentas AWS SDKs e ferramentas. • Para isso AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Etapa 2: configure as permissões do console do Amazon Rekognition Custom Labels

Para usar o console do Amazon Rekognition, é preciso adicionar para ter as permissões apropriadas. Se quiser armazenar seus arquivos de treinamento em um bucket diferente do bucket do console, precisará de permissões adicionais.

Tópicos

- Como permitir o acesso ao console
- Como acessar os buckets externos do Amazon S3
- Como atribuir permissões

Como permitir o acesso ao console

Para usar o console Amazon Rekognition Custom Labels, você precisa da seguinte política do IAM que abrange Amazon S3, AI SageMaker Ground Truth e Amazon Rekognition Custom Labels. Para obter informações sobre como atribuir permissões, consulte Como atribuir permissões.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "s3Policies",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:CreateBucket",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:GetBucketVersioning",
                "s3:GetObjectVersionTagging",
                "s3:PutBucketCORS",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:PutObject",
                "s3:PutObjectTagging",
                "s3:PutBucketVersioning",
                "s3:PutObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::custom-labels-console-*"

            ]
        },
        {
            "Sid": "rekognitionPolicies",
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "groundTruthPolicies",
```

```
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Como acessar os buckets externos do Amazon S3

Quando você abre pela primeira vez o console do Amazon Rekognition Custom Labels em uma nova AWS região, o Amazon Rekognition Custom Labels cria um bucket (bucket do console) que é usado para armazenar arquivos do projeto. Como alternativa, você pode usar seu próprio bucket do Amazon S3 (bucket externo) para carregar as imagens ou o arquivo de manifesto no console. Para usar um bucket externo, adicione o bloco de políticas a seguir à política anterior. Substitua `amzn-s3-demo-bucket` pelo nome do bucket.

```
        {
            "Sid": "s3ExternalBucketPolicies",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket*"
            ]
        }
```

Como atribuir permissões

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .

- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.

- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Etapa 3: crie um bucket do console

Um projeto do Amazon Rekognition Custom Labels é usado para criar e gerenciar seus modelos. Quando você abre o console Amazon Rekognition Custom Labels pela primeira vez em uma nova AWS região, o Amazon Rekognition Custom Labels cria um bucket do Amazon S3 (bucket do console) para armazenar seus projetos. Você deve anotar o nome do bucket do console em algum lugar onde possa consultá-lo posteriormente, pois talvez seja necessário usar o nome do bucket nas operações do AWS SDK ou nas tarefas do console, como criar um conjunto de dados.

O formato do nome do bucket é `custom-labels-console-<region>-<random value>`. O valor aleatório garante que não haja uma colisão entre os nomes dos buckets.

Para criar o bucket do console

1. Certifique-se de que o usuário tenha as permissões corretas. Para obter mais informações, consulte Como permitir o acesso ao console.
2. Faça login no AWS Management Console e abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
3. Escolha Começar.
4. Se for a primeira vez que abrir o console na região da AWS atual, faça o seguinte na caixa de diálogo Primeira configuração:

a. Copie o nome do bucket do Amazon S3 exibido. Você precisará dessas informações posteriormente.

b. Escolha Criar bucket do S3 para permitir que os Amazon Rekognition Custom Labels criem um bucket do Amazon S3 (bucket de console) em seu nome.

5. Feche a janela do navegador.

Etapa 4: configurar o AWS CLI e AWS SDKs

Você pode usar etiquetas personalizadas do Amazon Rekognition com () e. AWS Command Line Interface AWS CLI AWS SDKs Se você precisar executar operações do Amazon Rekognition Custom Labels do terminal, instale a AWS CLI. Se você estiver criando um aplicativo, baixe o AWS SDK para a linguagem de programação que você está usando.

Tópicos

- Instale os AWS SDKS
- Conceder acesso programático
- Configurar permissões do SDK
- Chame uma operação do Amazon Rekognition Custom Labels

Instale os AWS SDKS

Siga as etapas para fazer download e configurar a AWS SDKs.

Para configurar o AWS CLI e o AWS SDKs

- Baixe e instale o AWS CLIe o AWS SDKs que você deseja usar. Este guia fornece exemplos para o AWS CLI, Java e Python. Para obter informações sobre a instalação AWS SDKs, consulte Tools for Amazon Web Services.

Conceder acesso programático

Você pode executar os exemplos de código AWS CLI e os exemplos deste guia em seu computador local ou em outros AWS ambientes, como uma instância do Amazon Elastic Compute Cloud. Para executar os exemplos, você precisa conceder acesso às operações do AWS SDK que os exemplos usam.

Tópicos

- Executando código em seu computador local
- Executando código em AWS ambientes

Executando código em seu computador local

Para executar código em um computador local, recomendamos que você use credenciais de curto prazo para conceder ao usuário acesso às operações do AWS SDK. Para obter informações específicas sobre como executar o AWS CLI e exemplos de código em um computador local, consulteUsando um perfil em seu computador local.

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identidade da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de ferramentas AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações	Siga as instruções em Como usar credenciais temporárias

Qual usuário precisa de acesso programático?	Para	Por
	programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para o AWS CLI, AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para ferramentas AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de ferramentas AWS SDKs e ferramentas. • Para isso AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Usando um perfil em seu computador local

Você pode executar os exemplos de código AWS CLI e de código neste guia com as credenciais de curto prazo que você criou. Executando código em seu computador local Para obter as credenciais e outras informações de configurações, os exemplos usam um perfil chamado `custom-labels-access`, por exemplo:

```
session = boto3.Session(profile_name='custom-labels-access')
```

```
rekognition_client = session.client("rekognition")
```

O usuário que o perfil representa deve ter permissões para chamar as operações do SDK do Amazon Rekognition Custom Labels e outras operações AWS do SDK exigidas pelos exemplos. Para obter mais informações, consulte Configurar permissões do SDK. Para atribuir permissões, consulte Configurar permissões do SDK.

Para criar um perfil que funcione com os exemplos de código AWS CLI e, escolha uma das opções a seguir. Verifique se o nome do perfil que você criou é `custom-labels-access`.

- Usuários gerenciados pelo IAM — Siga as instruções em Mudar para um perfil do IAM (AWS CLI).
- Identidade da força de trabalho (usuários gerenciados por AWS IAM Identity Center) — Siga as instruções em Configuração da AWS CLI para uso. AWS IAM Identity Center Para os exemplos de código, recomendamos o uso de um ambiente de desenvolvimento integrado (IDE), que oferece suporte ao AWS Toolkit, permitindo a autenticação por meio do IAM Identity Center. Para ver os exemplos de Java, consulte Começar a criar com Java. Para ver os exemplos de Python, consulte Começar a criar com Python. Para obter mais informações, consulte Credenciais do IAM Identity Center.

> **Note**
>
> Você pode usar o código para obter credenciais de curto prazo. Para obter mais informações, consulte Mudar para um perfil do IAM (API da AWS). Para o IAM Identity Center, obtenha as credenciais de curto prazo para uma função seguindo as instruções em Obter credenciais de perfil do IAM para acesso à CLI.

Executando código em AWS ambientes

Você não deve usar as credenciais do usuário para assinar chamadas do AWS SDK em AWS ambientes, como código de produção executado em uma AWS Lambda função. Em vez disso, você configura uma função que define as permissões de que seu código precisa. Em seguida, você atribui a função ao ambiente em que seu código é executado. A forma como você atribui a função e disponibiliza credenciais temporárias varia de acordo com o ambiente em que seu código é executado:

- AWS Lambda função — Use as credenciais temporárias que o Lambda fornece automaticamente à sua função quando assume a função de execução da função Lambda. As credenciais estão

disponíveis nas variáveis de ambiente do Lambda. Você não precisa especificar um perfil. Para obter mais informações, consulte Função de execução do Lambda.

- Amazon EC2 — Use o provedor de credenciais de endpoint de metadados da EC2 instância Amazon. O provedor gera e atualiza automaticamente as credenciais para você usando o perfil de EC2 instância da Amazon que você anexa à instância da Amazon EC2 . Para obter mais informações, consulte Como usar uma função do IAM para conceder permissões a aplicativos executados em EC2 instâncias da Amazon
- Amazon Elastic Container Service — Use o provedor de credenciais do Container. O Amazon ECS envia e atualiza as credenciais para um endpoint de metadados. Um perfil do IAM de tarefa que você especifica fornece uma estratégia para gerenciar as credenciais que seu aplicativo usa. Para obter mais informações, consulte Interagir com os serviços da AWS.

Para obter mais informações sobre provedores de credenciais, consulte Provedores padronizados de credenciais.

Configurar permissões do SDK

Para usar as operações do Amazon Rekognition Custom Labels SDK, você precisa de permissões de acesso à API do Amazon Rekognition Custom Labels e ao bucket do Amazon S3 usado para treinamento de modelos.

Tópicos

- Conceder permissões de operação do SDK
- Atualizações de políticas para usar o AWS SDK
- Como atribuir permissões

Conceder permissões de operação do SDK

É recomendável conceder apenas as permissões necessárias para executar uma tarefa (permissões de privilégio mínimo). Por exemplo, para ligar DetectCustomLabels, você precisa de permissão para executar`rekognition:DetectCustomLabels`. Para encontrar as permissões para uma operação, verifique a referência da API.

Quando estiver apenas começando a usar um aplicativo, talvez não saiba as permissões específicas de que precisa, então é possível começar com permissões mais amplas. As políticas gerenciadas pela AWS fornecem permissões para ajudá-lo a começar. Você pode usar a política

`AmazonRekognitionCustomLabelsFullAccess` AWS gerenciada para obter acesso completo à API Amazon Rekognition Custom Labels. Para obter mais informações, consulte a política gerenciada da AWS: AmazonRekognitionCustomLabelsFullAccess. Quando você conhece as permissões de que sua aplicação precisa, reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pelo cliente.

Para atribuir permissões, consulte Como atribuir permissões.

Atualizações de políticas para usar o AWS SDK

Para usar o AWS SDK com a versão mais recente do Amazon Rekognition Custom Labels, você não precisa mais conceder permissões aos Amazon Rekognition Custom Labels para acessar o bucket do Amazon S3 que contém suas imagens de treinamento e teste. Se adicionou permissões anteriormente, não é necessário removê-las. Se optar por isso, remova qualquer política do bucket onde o serviço para a entidade principal é `rekognition.amazonaws.com`. Por exemplo:

```
"Principal": {
    "Service": "rekognition.amazonaws.com"
}
```

Para obter mais informações, consulte Como usar políticas de bucket.

Como atribuir permissões

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .
- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.
- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.

- (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Chame uma operação do Amazon Rekognition Custom Labels

Execute o código a seguir para confirmar que é possível fazer chamadas para a API Amazon Rekognition Custom Labels. O código lista os projetos em sua AWS conta, na AWS região atual. Se ainda não tiver criado um projeto, a resposta está vazia, mas confirma que é possível chamar a operação `DescribeProjects`.

Em geral, chamar uma função de exemplo requer um cliente do AWS SDK Rekognition e quaisquer outros parâmetros necessários. O cliente do AWS SDK é declarado na função principal.

Se o código falhar, verifique se o usuário que você usa tem as permissões corretas. Verifique também se a AWS região que você está usando como etiquetas personalizadas do Amazon Rekognition não está disponível em todas as regiões. AWS

Para chamar uma operação do Amazon Rekognition Custom Labels

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir para visualizar seus projetos.

   CLI

   Use o comando `describe-projects` para listar os projetos em sua conta.

   ```
   aws rekognition describe-projects \
   --profile custom-labels-access
   ```

   Python

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   ```

```
This example shows how to describe your Amazon Rekognition Custom Labels
 projects.
If you haven't previously created a project in the current AWS Region,
the response is an empty list, but does confirm that you can call an
Amazon Rekognition Custom Labels operation.
"""
from botocore.exceptions import ClientError
import boto3

def describe_projects(rekognition_client):
    """
    Lists information about the projects that are in in your AWS account
    and in the current AWS Region.

    : param rekognition_client: A Boto3 Rekognition client.
    """
    try:
        response = rekognition_client.describe_projects()
        for project in response["ProjectDescriptions"]:
            print("Status: " + project["Status"])
            print("ARN: " + project["ProjectArn"])
            print()
        print("Done!")
    except ClientError as err:
        print(f"Couldn't describe projects. \n{err}")
        raise


def main():
    """
    Entrypoint for script.
    """

    session = boto3.Session(profile_name='custom-labels-access')
    rekognition_client = session.client("rekognition")

    describe_projects(rekognition_client)


if __name__ == "__main__":
    main()
```

Java V2

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class Hello {

    public static final Logger logger = Logger.getLogger(Hello.class.getName());

    public static void describeMyProjects(RekognitionClient rekClient) {

        DescribeProjectsRequest descProjects = null;

        // If a single project name is supplied, build projectNames argument

        List<String> projectNames = new ArrayList<String>();


        descProjects = DescribeProjectsRequest.builder().build();

        // Display useful information for each project.
```

```
        DescribeProjectsResponse resp =
 rekClient.describeProjects(descProjects);

        for (ProjectDescription projectDescription : resp.projectDescriptions())
 {

            System.out.println("ARN: " + projectDescription.projectArn());
            System.out.println("Status: " +
 projectDescription.statusAsString());
            if (projectDescription.hasDatasets()) {
                for (DatasetMetadata datasetDescription :
 projectDescription.datasets()) {
                    System.out.println("\tdataset Type: " +
 datasetDescription.datasetTypeAsString());
                    System.out.println("\tdataset ARN: " +
 datasetDescription.datasetArn());
                    System.out.println("\tdataset Status: " +
 datasetDescription.statusAsString());
                }
            }
            System.out.println();
        }

    }

    public static void main(String[] args) {

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Describe projects

            describeMyProjects(rekClient);

            rekClient.close();

        } catch (RekognitionException rekError) {
```

```
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Etapa 5: (opcional) criptografe os arquivos de treinamento

É possível escolher uma das seguintes opções para criptografar os arquivos de manifesto e os arquivos de imagem do Amazon Rekognition Custom Labels que estão em um bucket de console ou em um bucket externo do Amazon S3.

- Use uma chave do Amazon S3 (SSE-S3).
- Use seu AWS KMS key.

> **Note**
>
> A entidade principal do IAM chamada precisa de permissões para descriptografar os arquivos. Para obter mais informações, consulte Descriptografando arquivos criptografados com AWS Key Management Service.

Para obter informações sobre como criptografar um bucket do Amazon S3, consulte Definir o comportamento da criptografia padrão do lado do servidor para os buckets do Amazon S3.

Descriptografando arquivos criptografados com AWS Key Management Service

Se você usa AWS Key Management Service (KMS) para criptografar seus arquivos de manifesto e arquivos de imagem do Amazon Rekognition Custom Labels, adicione o principal do IAM que chama Amazon Rekognition Custom Labels à política de chaves da chave KMS. Isso permite que o Amazon Rekognition Custom Labels descriptografe seus arquivos de manifesto e de imagem antes do treinamento. Para obter mais informações, consulte Meu bucket do Amazon S3 tem criptografia padrão usando uma chave do AWS KMS personalizada. Como permitir que os usuários baixem e façam upload para o bucket?

A entidade principal do IAM precisa das permissões a seguir na chave do KMS.

- kms:GenerateDataKey
- kms:Decrypt

Para obter mais informações, consulte Como proteger dados usando criptografia do lado do servidor com chaves KMS armazenadas no AWS Key Management Service (SSE-KMS).

Como criptografar imagens copiadas de treinamento e teste

Para treinar seu modelo, o Amazon Rekognition Custom Labels faz uma cópia das imagens originais de treinamento e teste. Por padrão, as imagens copiadas são criptografadas em repouso com uma chave que a AWS possui e gerencia. Também é possível optar por usar a sua própria AWS KMS key. Se usa sua própria chave do KMS, precisará das permissões a seguir na chave do KMS.

- kms:CreateGrant
- kms:DescribeKey

Opcionalmente, especifique a chave KMS ao treinar o modelo com o console ou ao chamar a operação `CreateProjectVersion`. A chave KMS que você usa não precisa ser a mesma chave KMS que você usa para criptografar arquivos de manifesto e imagem em seu bucket do Amazon S3. Para obter mais informações, consulte Etapa 5: (opcional) criptografe os arquivos de treinamento.

Para obter mais informações, consulte Conceitos do AWS Key Management Service. Suas imagens de origem não são afetadas.

Para obter informações sobre como treinar um modelo, consulte Como treinar um modelo do Amazon Rekognition Custom Labels.

Etapa 6: (opcional) associe conjuntos de dados anteriores com novos projetos

O Amazon Rekognition Custom Labels agora gerencia conjuntos de dados com projetos. Os conjuntos de dados anteriores (anteriores) que você criou são somente para leitura e devem ser associados a um projeto antes que você possa usá-los. Ao abrir a página de detalhes de um projeto com o console, associamos automaticamente os conjuntos de dados que treinaram a versão mais

recente do modelo do projeto ao projeto. A associação automática de um conjunto de dados a um projeto não acontece se você estiver usando o AWS SDK.

Os conjuntos de dados anteriores não associados nunca foram usados para treinar um modelo ou foram usados para treinar uma versão anterior de um modelo. A página Conjuntos de dados anteriores mostra todos os seus conjuntos de dados associados e não associados.

Para usar um conjunto de dados anterior não associado, você cria um novo projeto na página Conjuntos de dados anteriores. O conjunto de dados se torna o conjunto de dados de treinamento para o novo projeto. Também é possível criar um projeto para um conjunto de dados já associado, pois os conjuntos de dados anteriores podem ter várias associações.

Para associar um conjunto de dados anterior a um novo projeto

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Usar rótulos personalizados. A página inicial do Amazon Rekognition Custom Labels é exibida.
3. No painel de navegação esquerdo, selecione Conjuntos de dados anteriores.
4. Na visualização de conjuntos de dados, escolha o conjunto de dados anterior que você deseja associar a um projeto.
5. Escolha Criar projeto com conjunto de dados.
6. Na janela Criar um projeto, insira um nome para seu novo projeto em Nome do projeto.
7. Escolha Criar projeto para criar o projeto. O projeto pode demorar um pouco para ser criado.
8. Use o projeto. Para obter mais informações, consulte Noções básicas do Amazon Rekognition Custom Labels.

Como usar um conjunto de dados anterior como um conjunto de dados de teste

É possível usar um conjunto de dados anterior como conjunto de dados de teste para um projeto existente associando primeiro o conjunto de dados anterior a um novo projeto. O conjunto de dados de teste do novo projeto é copiado ao conjunto de dados de treinamento do projeto existente.

Para usar um conjunto de dados anterior como um conjunto de dados de teste

1. Siga as instruções em Etapa 6: (opcional) associe conjuntos de dados anteriores com novos projetos para associar o conjunto de dados anterior a um novo projeto.

2. Crie o conjunto de dados de teste no projeto existente usando a cópia do conjunto de dados de treinamento do novo projeto. Para obter mais informações, consulte Copiar conteúdo de um conjunto de dados existente.
3. Siga as instruções em Como excluir um projeto do Amazon Rekognition Custom Labels (console) para excluir o novo projeto.

Como alternativa, é possível criar o conjunto de dados de teste usando o arquivo de manifesto do conjunto de dados anterior. Para obter mais informações, consulte Criar um arquivo de manifesto.

Noções básicas do Amazon Rekognition Custom Labels

Esta seção fornece uma visão geral do fluxo de trabalho para treinar e usar um modelo de etiquetas personalizadas do Amazon Rekognition com o console e o SDK. AWS

> **Note**
>
> O Amazon Rekognition Custom Labels agora gerencia conjuntos de dados dentro de um projeto. Você pode criar conjuntos de dados para seus projetos com o console e com o AWS SDK. Se você já usou o Amazon Rekognition Custom Labels, talvez seja necessário associar seus conjuntos de dados antigos a um novo projeto. Para ter mais informações, consulte Etapa 6: (opcional) associe conjuntos de dados anteriores com novos projetos

Tópicos

- Decida o tipo do seu modelo
- Criar um modelo
- Melhore seu modelo
- Executar seu modelo
- Analisar uma imagem
- Interrompa seu modelo

Decida o tipo do seu modelo

Primeiro, decida qual tipo de modelo deseja treinar, o que depende de suas metas comerciais. Por exemplo, é possível treinar um modelo para encontrar seu logotipo em publicações nas redes sociais, identificar seus produtos nas prateleiras das lojas ou classificar peças de máquinas em uma linha de montagem.

O Amazon Rekognition Custom Labels pode treinar os seguintes tipos de modelo:

- Encontre objetos, cenas e conceitos
- Encontre localizações de objetos
- Encontre a localização das marcas

Para ajudar a decidir qual tipo de modelo treinar, o Amazon Rekognition Custom Labels fornece exemplos de projetos que podem ser usados. Para obter mais informações, consulte Conceitos básicos do Amazon Rekognition Custom Labels.

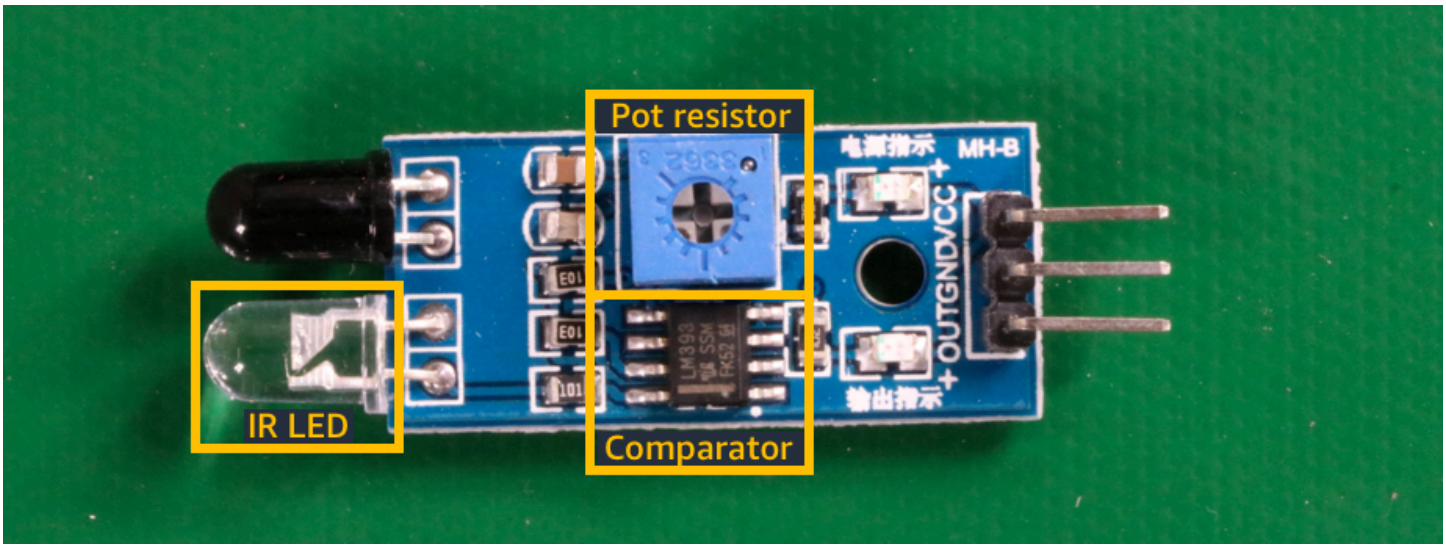
Encontre objetos, cenas e conceitos

O modelo prevê classificações para os objetos, cenas e conceitos associados a uma imagem inteira. Por exemplo, é possível treinar um modelo que determine se uma imagem contém uma atração turística ou não. Para obter um objeto de exemplo, consulte Classificação de imagens. A imagem a seguir de um lago é um exemplo do tipo de imagem em que você pode reconhecer objetos, cenas e conceitos.

Como alternativa, é possível treinar um modelo que categorize as imagens em várias categorias. Por exemplo, a imagem anterior pode ter categorias como cor do céu, reflexo ou lago. Para obter um objeto de exemplo, consulte Classificação de imagens com vários rótulos.

Encontre localizações de objetos

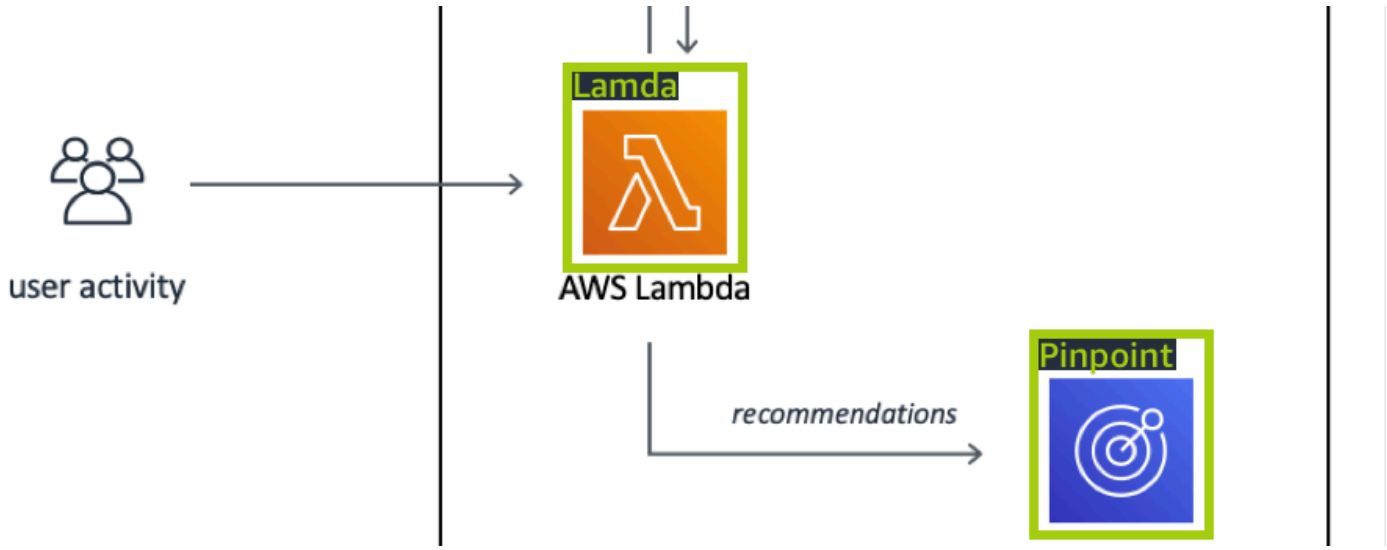
O modelo prevê a localização de um objeto em uma imagem. A previsão inclui informações da caixa delimitadora para a localização do objeto e um rótulo que identifica o objeto dentro da caixa delimitadora. Por exemplo, a imagem a seguir mostra as caixas delimitadoras em torno de várias partes de uma placa de circuito, como um comparador ou potenciômetro.



O projeto de exemplo Localização de objetos mostra como o Amazon Rekognition Custom Labels usa caixas delimitadoras rotuladas para treinar um modelo que encontra a localização dos objetos.

Encontre a localização das marcas

O Amazon Rekognition Custom Labels pode treinar um modelo que encontra a localização de marcas, como logotipos, em uma imagem. A previsão inclui informações da caixa delimitadora para a localização da marca e um rótulo que identifica o objeto dentro da caixa delimitadora. Para obter um objeto de exemplo, consulte Detecção de marca. A imagem a seguir é um exemplo de algumas das marcas que o modelo pode detectar.



Criar um modelo

As etapas para criar um modelo são as seguintes: criar um projeto, criar conjuntos de dados de treinamento e teste, e treinar o modelo.

Criar um projeto

Um projeto é um grupo de recursos necessários para criar e gerenciar versões de um modelo do Amazon Rekognition Custom Labels. Um projeto gerencia o seguinte:

- Conjuntos de dados: as imagens e os rótulos de imagem usados para treinar um modelo. Um projeto tem um conjunto de dados de treinamento e um conjunto de dados de teste.
- Modelos: o software que você treina para encontrar conceitos, cenas e objetos exclusivos da sua empresa. É possível ter várias versões de um modelo em um projeto.

É recomendável usar um projeto para um único caso de uso, como descobrir peças da placa de circuito em uma placa de circuito.

Você pode criar um projeto com o console Amazon Rekognition Custom Labels e com a API. CreateProject Para obter mais informações, consulte Como criar um projeto.

Crie conjuntos de dados de treinamento e teste

Um conjunto de dados é um conjunto de imagens e rótulos que descrevem essas imagens. No projeto, é criado um conjunto de dados de treinamento e um conjunto de dados de teste que o Amazon Rekognition Custom Labels usa para treinar e testar seu modelo.

Um rótulo identifica um objeto, cena, conceito ou caixa delimitadora ao redor de um objeto em uma imagem. Os rótulos são atribuídos a uma imagem inteira (nível de imagem) ou são atribuídos a uma caixa delimitadora que circunda um objeto em uma imagem.

> **Important**
>
> A forma como você rotula as imagens em seus conjuntos de dados determina o tipo de modelo que o Amazon Rekognition Custom Labels cria. Por exemplo, para treinar um modelo que encontre objetos, cenas e conceitos, você atribui rótulos de nível de imagem às imagens em seus conjuntos de dados de treinamento e teste. Para obter mais informações, consulte Como definir os conjuntos de dados.

As imagens devem estar nos formatos PNG e JPEG, e você deve seguir as recomendações das imagens de entrada. Para obter mais informações, consulte Como preparar imagens.

Crie conjuntos de dados de treinamento e teste (console)

É possível iniciar um projeto com um único conjunto de dados ou com conjuntos de dados de treinamento e teste separados. Se você começar com um único conjunto de dados, o Amazon Rekognition Custom Labels divide seu conjunto de dados durante o treinamento para criar um conjunto de dados de treinamento (80%) e um conjunto de dados de teste (20%) para seu projeto. Comece com um único conjunto de dados se quiser que o Amazon Rekognition Custom Labels decida quais imagens serão usadas para treinamento e teste. Para ter controle total sobre o treinamento, teste e ajuste de desempenho, recomendamos que você inicie seu projeto com os conjuntos de dados de treinamento e teste separados.

Para criar os conjuntos de dados para um projeto, importe as imagens das seguintes maneiras:

- Importe imagens do seu computador local.
- Importe imagens de um bucket do S3. O Amazon Rekognition Custom Labels podem rotular as imagens usando os nomes das pastas que contêm as imagens.

- Importe um arquivo de manifesto do Amazon SageMaker AI Ground Truth.
- Copie um conjunto de dados existente do Amazon Rekognition Custom Labels.

Para obter mais informações, consulte Como criar conjuntos de dados de treinamento e teste com imagens.

Dependendo de onde você importa suas imagens, elas podem não estar rotuladas. Por exemplo, imagens importadas de um computador local não estão rotuladas. As imagens importadas de um arquivo de manifesto do Amazon SageMaker AI Ground Truth são rotuladas. É possível usar o console do Amazon Rekognition Custom Labels para adicionar, alterar e atribuir rótulos. Para obter mais informações, consulte Rotulagem de imagens.

Para criar seus conjuntos de dados de treinamento e teste com o console, consulte Como criar conjuntos de dados de treinamento e teste com imagens. Para ver um tutorial que inclui a criação de conjuntos de dados de treinamento e teste, consulte Classificar imagens.

Crie conjuntos de dados de treinamento e teste (SDK)

Para criar seus conjuntos de dados de treinamento e teste, use a API `CreateDataset`. É possível criar um conjunto de dados usando um arquivo de manifesto no formato Amazon Sagemaker ou copiando um conjunto de dados existente do Amazon Rekognition Custom Labels. Para obter mais informações, consulte Crie conjuntos de dados de treinamento e teste (SDK) . Se necessário, é possível criar o seu próprio arquivo de manifesto. Para obter mais informações, consulte the section called “Criar um arquivo de manifesto”.

Treinar seu modelo

Treine seu modelo com o conjunto de dados de treinamento. Uma nova versão de um modelo é criada toda vez que ele é treinado. Durante o treinamento, o Amazon Rekognition Custom Labels testa o desempenho do seu modelo treinado. É possível usar os resultados para avaliar e melhorar seu modelo. O treinamento demora para ser concluído. Só há uma cobrança por um treinamento de modelo com êxito. Para obter mais informações, consulte Como treinar um modelo do Amazon Rekognition Custom Labels. Se o treinamento do modelo falhar, o Amazon Rekognition Custom Labels fornecerá informações de depuração que podem ser usadas. Para obter mais informações, consulte Como depurar um treinamento de modelo em falha.

Treinar seu modelo (console)

Para treinar seu modelo com o console, consulte Como treinar um modelo (console).

Treinando um modelo (SDK)

Você treina um modelo de etiquetas personalizadas do Amazon Rekognition ligando para. CreateProjectVersion Para obter mais informações, consulte Treinando um modelo (SDK).

Melhore seu modelo

Durante o teste, o Amazon Rekognition Custom Labels cria métricas de avaliação que podem ser usadas para melhorar seu modelo treinado.

Avalie seu modelo

Avalie o desempenho do seu modelo usando as métricas de desempenho criadas durante o teste. As métricas de desempenho, como F1, precisão e recall, permitem que você entenda o desempenho do seu modelo treinado e decida se está pronto para usá-lo na produção. Para obter mais informações, consulte Métricas para avaliar seu modelo.

Avaliar um modelo (console)

Para visualizar as métricas de desempenho, consulte Como acessar as métricas de avaliação (console).

Avaliar um modelo (SDK)

Para obter métricas de desempenho, você liga DescribeProjectVersionspara obter os resultados do teste. Para obter mais informações, consulte Como acessar as métricas de avaliação (SDK) do Amazon Rekognition Custom Labels. Os resultados do teste incluem métricas não disponíveis no console, como uma matriz de confusão para resultados de classificação. Os resultados do teste são retornados nos seguintes formatos:

- Pontuação F1: um valor único que representa o desempenho geral de precisão e recall do modelo. Para obter mais informações, consulte F1.
- Localização do arquivo de resumo: i resumo do teste inclui métricas de avaliação agregadas para todo o conjunto de dados de teste e métricas para cada rótulo individual. `DescribeProjectVersions` retorna o bucket do S3 e a localização da pasta do arquivo de resumo. Para obter mais informações, consulte Acessar o arquivo de resumo do modelo.
- Localização do snapshot do manifesto de avaliação: o snapshot contém detalhes sobre os resultados do teste, incluindo as classificações de confiança e os resultados dos testes de

classificação binária, como falsos positivos. `DescribeProjectVersions` retorna o bucket do S3 e a localização da pasta dos arquivos de snapshot. Para obter mais informações, consulte Interpretar o snapshot do manifesto de avaliação.

Melhore seu modelo

Se forem necessárias melhorias, é possível adicionar mais imagens de treinamento ou melhorar a rotulagem do conjunto de dados. Para obter mais informações, consulte Como melhorar um modelo do Amazon Rekognition Custom Labels. Também é possível dar feedback sobre as previsões que seu modelo faz e usá-lo para fazer melhorias em seu modelo. Para obter mais informações, consulte Melhorar um modelo com Model Feedback.

Melhore seu modelo (console)

Para adicionar imagens a um conjunto de dados, consulte Como adicionar mais imagens a um conjunto de dados. Para adicionar ou alterar rótulos, consulte the section called “Rotulagem de imagens”.

Para treinar seu modelo novamente, consulte Como treinar um modelo (console).

Melhore seu modelo (SDK)

Para adicionar imagens a um conjunto de dados ou alterar a rotulagem de uma imagem, use a API `UpdateDatasetEntries`. `UpdateDatasetEntries` atualiza ou adiciona linhas JSON a um arquivo de manifesto. Cada linha JSON contém informações para uma única imagem, como rótulos atribuídos ou informações da caixa delimitadora. Para obter mais informações, consulte Como adicionar mais imagens (SDK). Para visualizar as entradas em um conjunto de dados, use a API `ListDatasetEntries`.

Para treinar seu modelo novamente, consulte Como treinar um modelo (SDK).

Executar seu modelo

Antes de usar seu modelo, você inicia o modelo usando o console do Amazon Rekognition Custom Labels ou a API `StartProjectVersion`. Há uma cobrança pela quantidade de tempo que o modelo é executado. Para obter mais informações, consulte Como executar um modelo treinado.

Executar seu modelo (console)

Para iniciar o seu modelo usando o console, consulte Como iniciar um modelo do Amazon Rekognition Custom Labels (console).

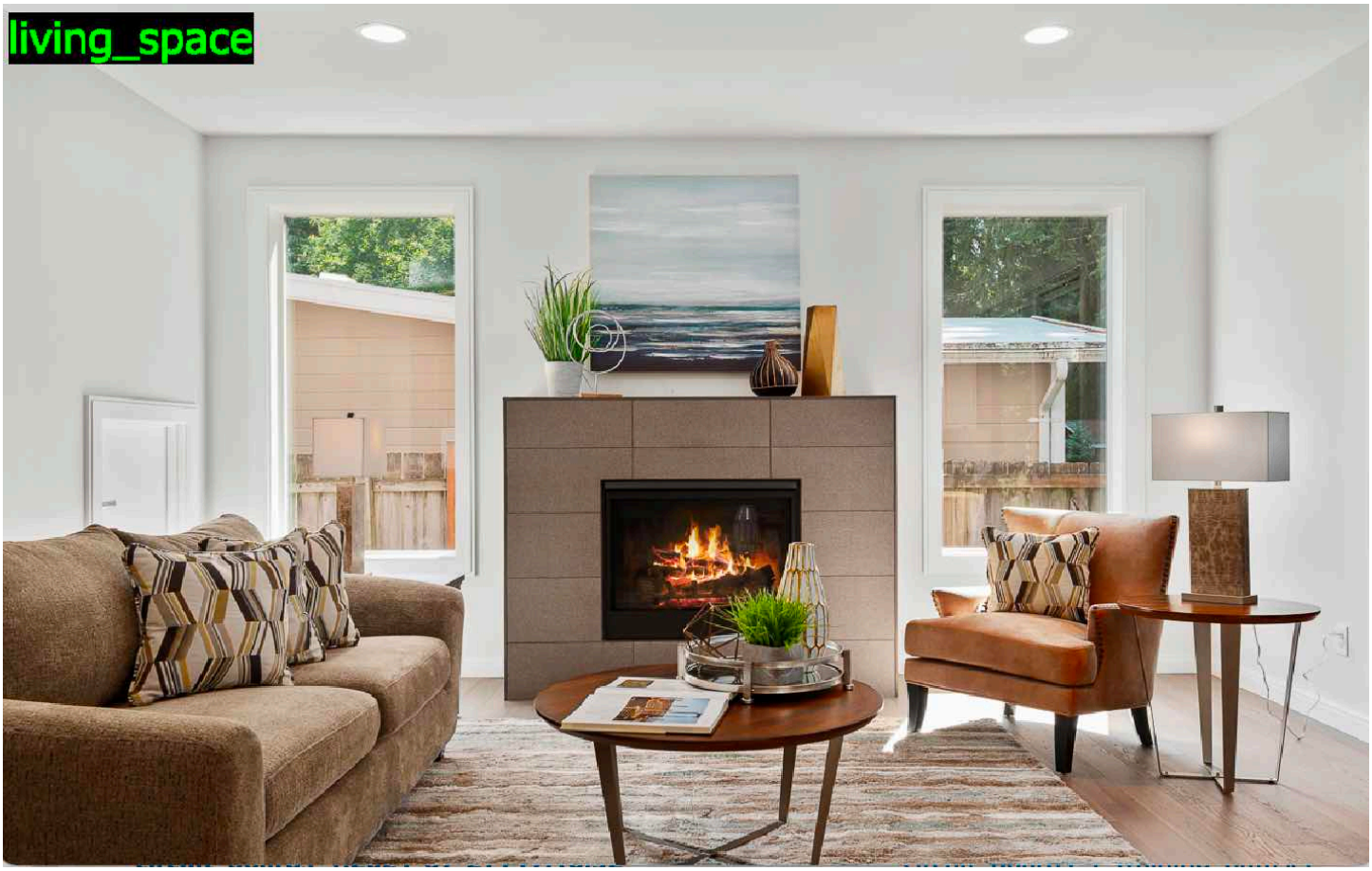
Executar seu modelo

Você começa a ligar para sua modelo StartProjectVersion. Para obter mais informações, consulte Como iniciar um modelo do Amazon Rekognition Custom Labels (SDK).

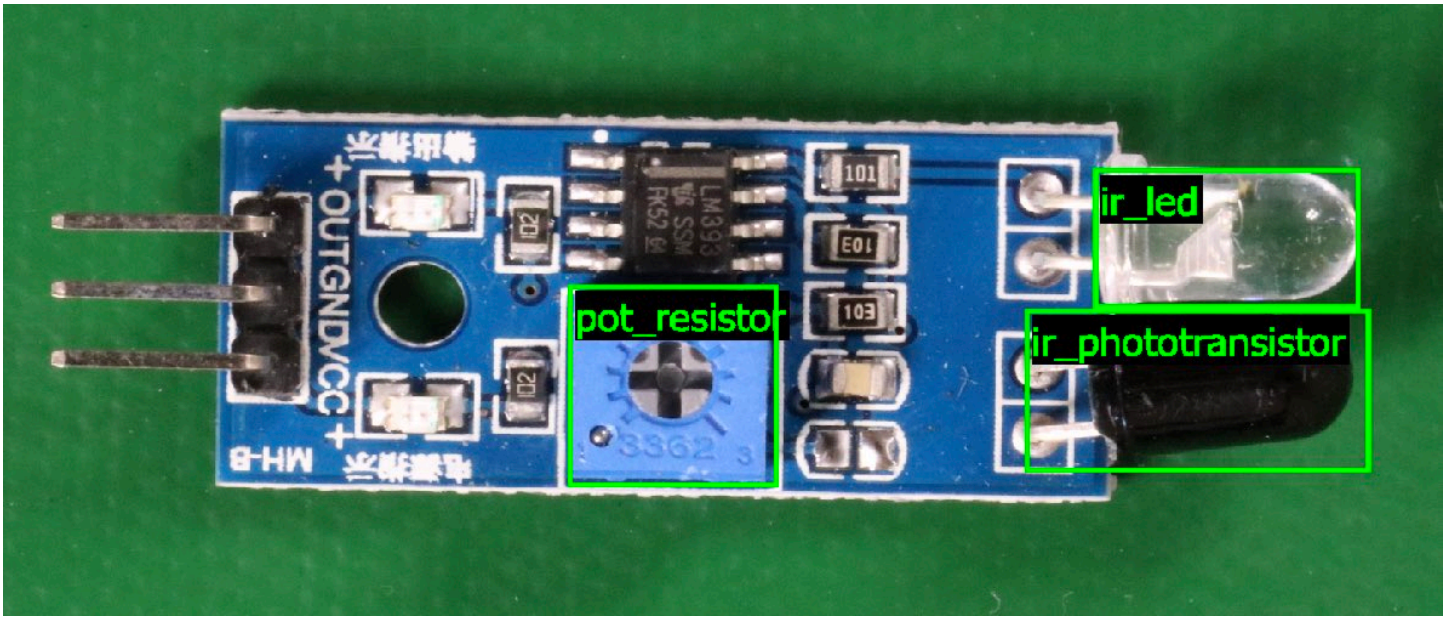
Analisar uma imagem

Para analisar uma imagem com seu modelo, você usa a API `DetectCustomLabels`. É possível especificar uma imagem local ou uma imagem armazenada em um bucket do S3. A operação também requer o nome do recurso da Amazon (ARN) do modelo que deseja utilizar.

Se seu modelo encontrar objetos, cenas e conceitos, a resposta incluirá uma lista de rótulos em nível de imagem encontrados na imagem. Por exemplo, a imagem a seguir mostra os rótulos no nível da imagem encontrados usando o projeto de exemplo Cômodos.



Se o modelo encontrar a localização dos objetos, a resposta incluirá uma lista de caixas delimitadoras rotuladas encontradas na imagem. Uma caixa delimitadora representa a localização de um objeto em uma imagem. É possível usar as informações da caixa delimitadora para desenhar uma caixa delimitadora ao redor de um objeto. Por exemplo, a imagem a seguir mostra caixas delimitadoras ao redor das partes da placa de circuito encontradas usando o projeto de exemplo de Placas de circuito.



Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.

Interrompa seu modelo

Há uma cobrança pelo tempo que o modelo está em execução. Se não estiver mais usando seu modelo, interrompa o modelo usando o console do Amazon Rekognition Custom Labels ou usando a API `StopProjectVersion`. Para obter mais informações, consulte Como interromper um modelo do Amazon Rekognition Custom Labels.

Interrompa seu modelo (console)

Para interromper a execução de um modelo com o console, consulte Como interromper um modelo do Amazon Rekognition Custom Labels (console).

Interrompa seu modelo (SDK)

Para interromper a execução de um modelo, ligue StopProjectVersion. Para obter mais informações, consulte Como interromper um modelo do Amazon Rekognition Custom Labels (SDK).

Conceitos básicos do Amazon Rekognition Custom Labels

Antes de iniciar estas instruções de Conceitos básicos, recomendamos que você leia Noções básicas do Amazon Rekognition Custom Labels.

São usados rótulios personalizados do Amazon Rekognition Custom Labels para treinar um modelo de machine learning. O modelo treinado analiza imagens para encontrar objetos, cenas e conceitos exclusivos da sua empresa. Por exemplo, é possível treinar um modelo para classificar imagens de casas ou encontrar a localização de peças eletrônicas em uma placa de circuito impresso.

Para ajudar você a começar, o Amazon Rekognition Custom Labels inclui tutorais em vídeo e projetos de exemplo.

> **Note**
>
> Para obter informações sobre as AWS regiões e endpoints compatíveis com o Amazon Rekognition Custom Labels, consulte Endpoints e cotas do Rekognition.

Tutoriais em vídeo

Os vídeos mostram como usar o Amazon Rekognition Custom Labels para treinar e usar um modelo.

Para visualizar os tutoriais em vídeo

1. Faça login no AWS Management Console e abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Usar rótulos personalizados. A página inicial do Amazon Rekognition Custom Labels é exibida. Se você não vê a opção Usar etiquetas personalizadas, verifique se a região da AWS que você está usando é compatível com o Amazon Rekognition Custom Labels.
3. No painel de navegação, escolha Conceitos básicos.
4. Em O que é o Amazon Rekognition Custom Labels?, escolha o vídeo para assistir ao vídeo de visão geral.
5. No painel de navegação, escolha Tutoriais.
6. Na página Tutoriais, escolha os tutoriais em vídeo que você deseja assistir.

Projetos de exemplo

O Amazon Rekognition Custom Labels fornece os seguintes exemplos de projetos.

Classificação de imagens

O projeto de classificação de imagens (Cômodos) treina um modelo que encontra uma ou mais localizações domésticas em uma imagem, como quintal, cozinha e pátio. As imagens de treinamento e teste representam um único local. Cada imagem é rotulada com um único rótulo em nível de imagem, como cozinha, pátio ou sala_de_estar. Para uma imagem analisada, o modelo treinado retorna um ou mais rótulos correspondentes do conjunto de rótulos em nível de imagem usado para treinamento. Por exemplo, o modelo pode encontrar o rótulo sala_de_estar na imagem a seguir. Para obter mais informações, consulte Encontre objetos, cenas e conceitos.

Classificação de imagens com vários rótulos

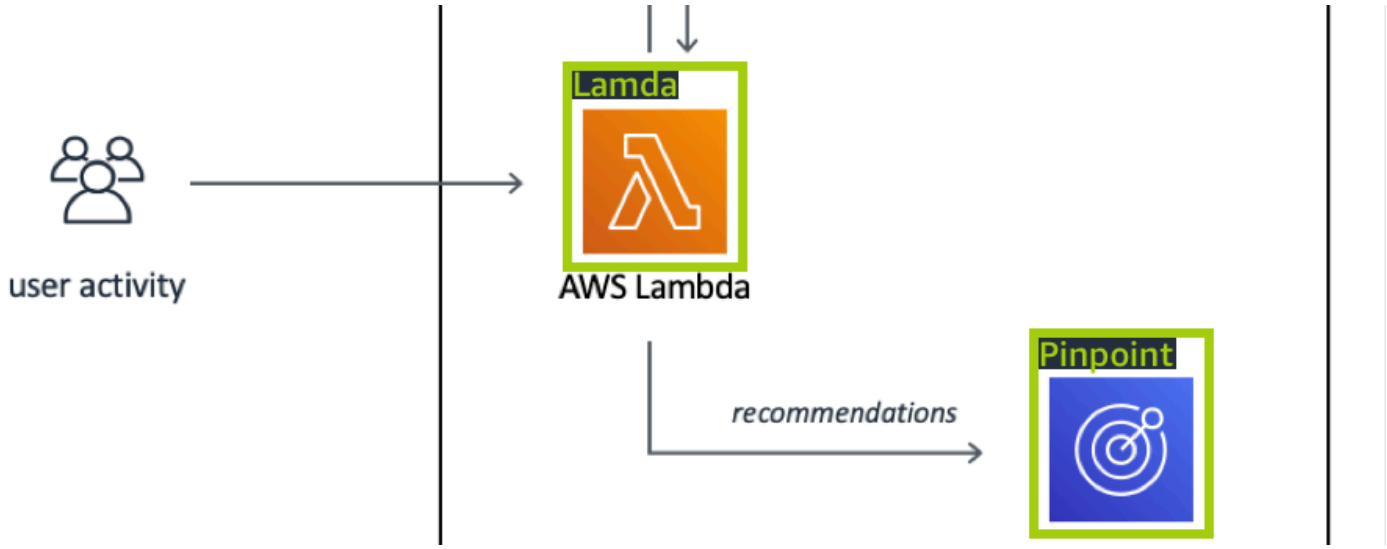
O projeto de classificação de imagens com vários rótulos (Flores) treina um modelo que categoriza imagens de flores em três conceitos (tipo de flor, presença de folhas e estágio de crescimento).

As imagens de treinamento e teste têm rótulos em nível de imagem para cada conceito, como camélia para um tipo de flor, with_leaves para uma flor com folhas e fully_grown para uma flor que está totalmente crescida.

Para uma imagem analisada, o modelo treinado retorna rótulos correspondentes do conjunto de rótulos em nível de imagem usado para treinamento. Por exemplo, o modelo retorna os rótulos herbacia_do_mediterraneo e com_folhas para a imagem a seguir. Para obter mais informações, consulte Encontre objetos, cenas e conceitos.

Detecção de marca

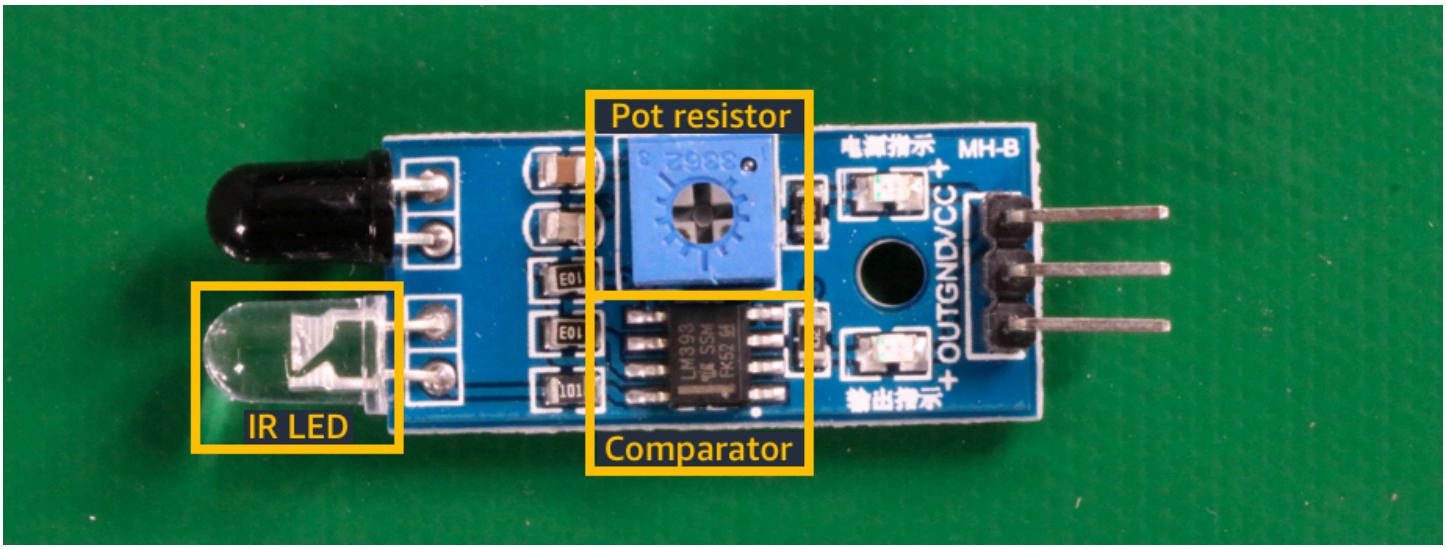
O projeto de detecção de marca (Logos) treina um modelo que encontra a localização de determinados AWS logotipos, como Amazon Textract e AWS lambda. As imagens de treinamento são apenas do logotipo e têm um único rótulo de nível de imagem, como lambda ou textract. Também é possível treinar um modelo de detecção de marca com imagens de treinamento que tenham caixas delimitadoras para a localização da marca. As imagens de teste têm caixas delimitadoras rotuladas que representam a localização dos logotipos em locais naturais, como um diagrama arquitetônico. O modelo treinado encontra os logotipos e retorna uma caixa delimitadora rotulada para cada logotipo encontrado. Para obter mais informações, consulte Encontre localizações de marcas.



Localização de objetos

O projeto de localização de objetos (placas de circuito) treina um modelo que encontra a localização das peças em uma placa de circuito impresso, como um comparador ou um diodo emissor de luz infravermelha. As imagens de treinamento e teste incluem caixas delimitadoras que cercam as partes da placa de circuito e um rótulo que identifica a peça dentro da caixa delimitadora. Na imagem de exemplo a seguir, os nomes dos rótulos são ir_phototransistor, ir_led, pot_resistor e comparator. O

modelo treinado encontra as peças da placa de circuito e retorna um limite rotulado para cada parte do circuito encontrada. Para obter mais informações, consulte Encontre localizações de objetos.



Como usar os projetos de exemplo

Estas instruções de introdução mostram como treinar um modelo usando exemplos de projetos que o Amazon Rekognition Custom Labels cria para você. Também mostra como iniciar o modelo e usá-lo para analisar uma imagem.

Como criar o projeto de exemplo

Para começar, decida qual projeto usar. Para obter mais informações, consulte Etapa 1: escolha um projeto de exemplo.

O Amazon Rekognition Custom Labels usa conjuntos de dados para treinar e avaliar (testar) um modelo. Um conjunto de dados gerencia imagens e os rótulos que identificam o conteúdo das imagens. Os projetos de exemplo incluem um conjunto de dados de treinamento e um conjunto de dados de teste no qual todas as imagens são rotuladas. Não é preciso fazer nenhuma alteração antes de treinar seu modelo. Os exemplos de projetos mostram as duas maneiras pelas quais o Amazon Rekognition Custom Labels usa rótulos para treinar diferentes tipos de modelos.

- image-level: o rótulo identifica um objeto, cena ou conceito que representa a imagem inteira.
- caixa delimitadora: o rótulo identifica o conteúdo de uma caixa delimitadora. Uma caixa delimitadora é um conjunto de coordenadas de imagem que circunda um objeto em uma imagem.

Posteriormente, ao criar um projeto com suas próprias imagens, você deve criar conjuntos de dados de treinamento e teste, além de rotular suas imagens. Para obter mais informações, consulte Decida o tipo do seu modelo.

Como treinar o modelo

Depois que o Amazon Rekognition Custom Labels criar o projeto de exemplo, é possível treinar o modelo. Para obter mais informações, consulte Etapa 2: treine seu modelo. Após o término do treinamento, você normalmente avalia o desempenho do modelo. As imagens no conjunto de dados de exemplo já criam um modelo de alto desempenho e você não precisa avaliar o modelo antes de executá-lo. Para obter mais informações, consulte Como melhorar um modelo treinado do Amazon Rekognition Custom Labels.

Como usar o modelo

Em seguida, inicie o modelo. Para obter mais informações, consulte Etapa 3: inicie seu modelo.

Depois de começar a executar seu modelo, é possível usá-lo para analisar novas imagens. Para obter mais informações, consulte Etapa 4: analise uma imagem com seu modelo.

Há uma cobrança pela quantidade de tempo que o modelo é executado. Ao terminar de usar o modelo de exemplo, deve interromper o modelo. Para obter mais informações, consulte Etapa 5: interrompa seu modelo.

Próximas etapas

Quando tudo estiver pronto, poderá criar e implantar seus próprios projetos. Para obter mais informações, consulte Etapa 6: próximas etapas.

Etapa 1: escolha um projeto de exemplo

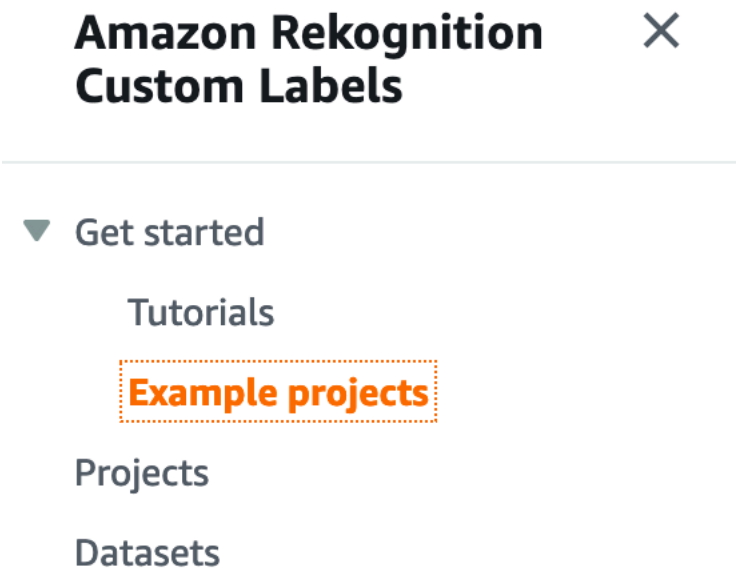
Nesta etapa, use Escolher um projeto de exemplo. Em seguida, o Amazon Rekognition Custom Labels cria um projeto e um conjunto de dados para você. Um projeto gerencia os arquivos usados para treinar seu modelo. Para obter mais informações, consulte Como gerenciar um projeto do Amazon Rekognition Custom Labels. Conjunto de dados contêm as imagens, os rótulos atribuídos e as caixas delimitadoras que você usa para treinar ou testar um modelo. Para obter mais informações, consulte the section called “Como gerenciar conjuntos de dados”.

Para obter informações sobre os projetos de exemplo, consulte Projetos de exemplo.

Escolha um projeto de exemplo

1. Faça login no AWS Management Console e abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Usar rótulos personalizados. A página inicial do Amazon Rekognition Custom Labels é exibida. Se você não vê a opção Usar etiquetas personalizadas, verifique se a região da AWS que você está usando é compatível com o Amazon Rekognition Custom Labels.
3. Escolha Começar.

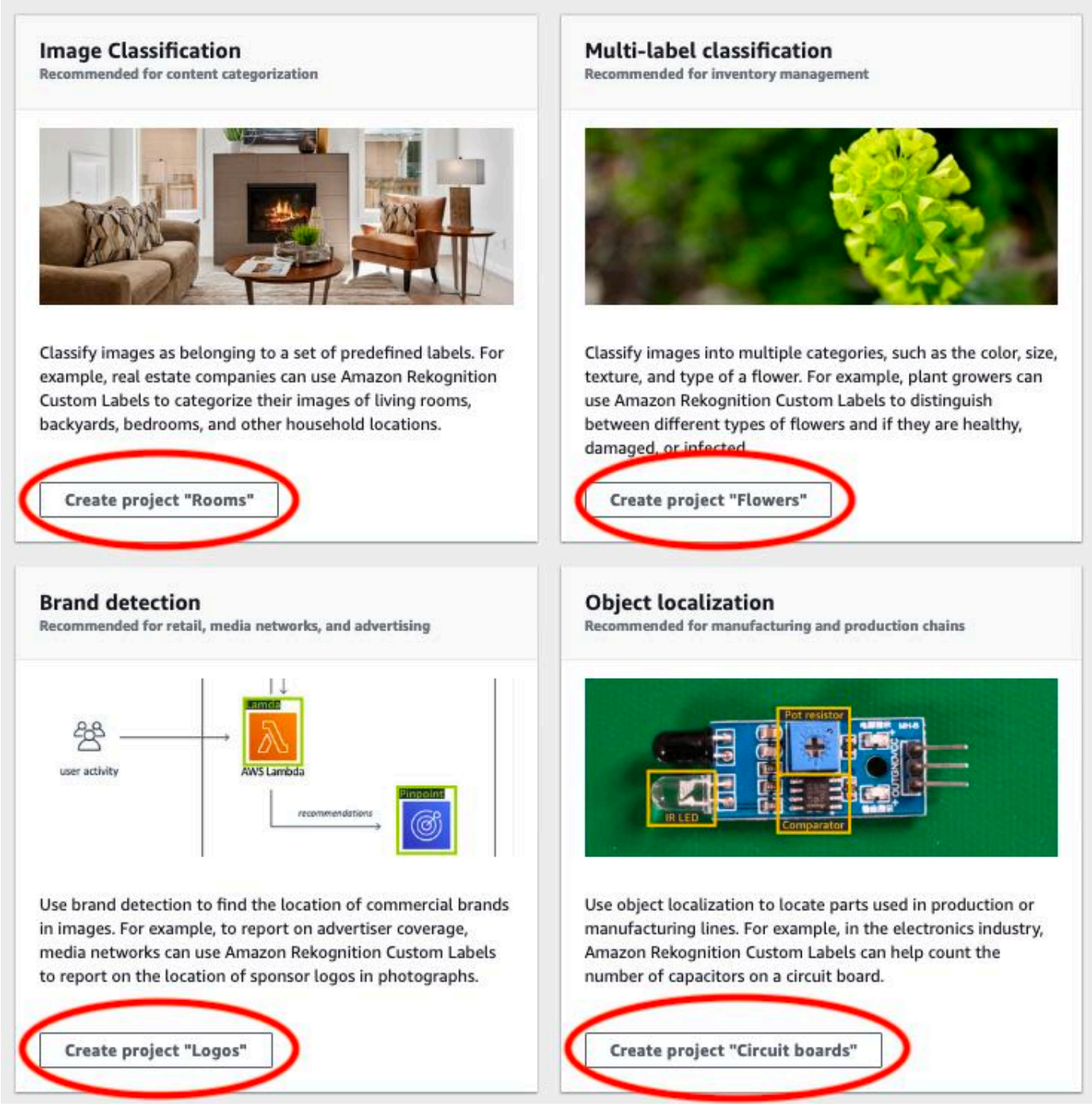
   Cabeçalho do Amazon Rekognition Custom Labels mostrando as seções Introdução, Tutoriais, Exemplos de projetos (em destaque), Projetos e Conjuntos de dados.



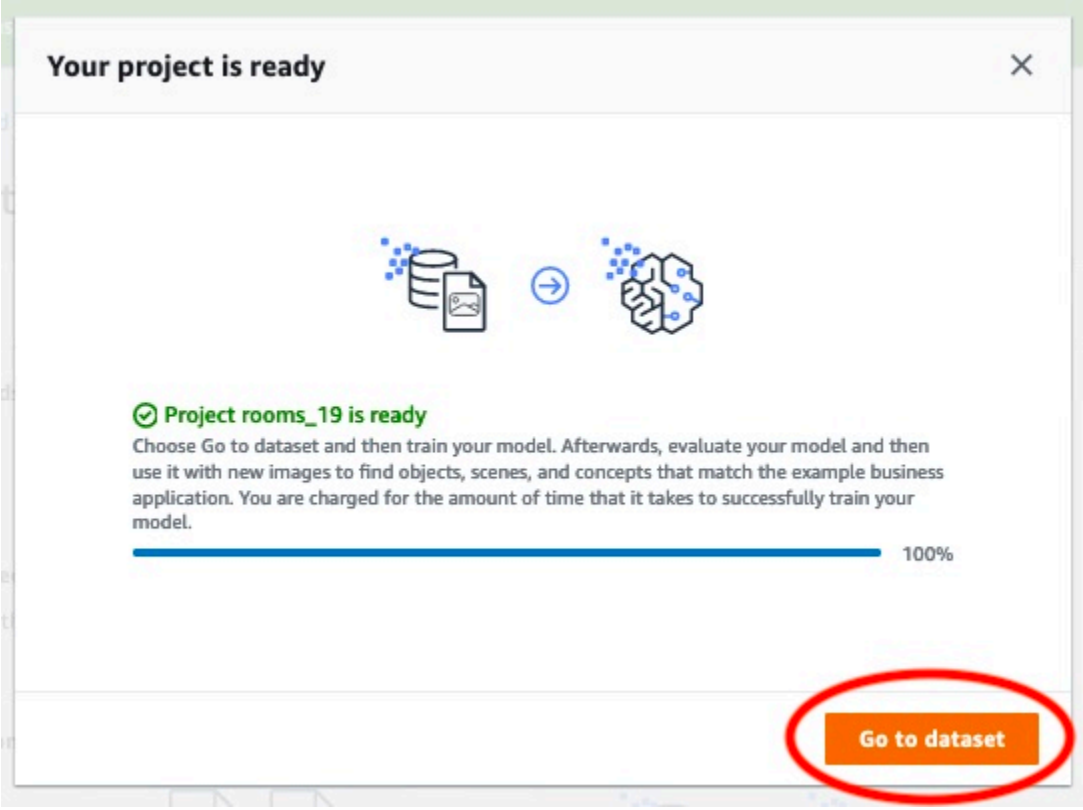
4. Em Explorar projetos de exemplo, escolha Experimentar projetos de exemplo.
5. Decida qual projeto você deseja usar e escolha Criar projeto "***project name***" na seção de exemplo. Em seguida, o Amazon Rekognition Custom Labels cria um projeto de exemplo para você.

> **Note**
>
> Se for a primeira vez que você abre o console na AWS região atual, a caixa de diálogo First Time Set Up será exibida. Faça o seguinte:
>
> 1. Observe o nome do bucket do Amazon S3 exibido.

2. Escolha Continuar para permitir que o Amazon Rekognition Custom Labels crie um bucket do Amazon S3 (bucket de console) em seu nome. A imagem do console abaixo mostra exemplos com os botões “Criar projeto” para Classificação de imagens (salas), Classificação de vários rótulos (flores), Detecção de marca (logotipos) e Localização de objetos (placas de circuito).



6. Depois que seu projeto estiver pronto, escolha Ir para o conjunto de dados. A imagem a seguir mostra qual é a aparência do painel quando o projeto está pronto.

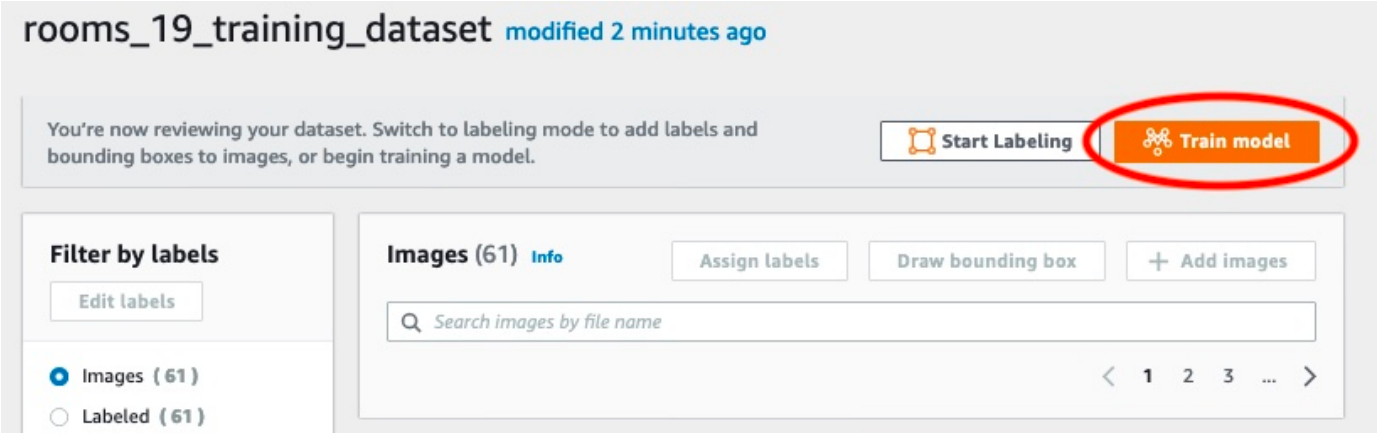


Etapa 2: treine seu modelo

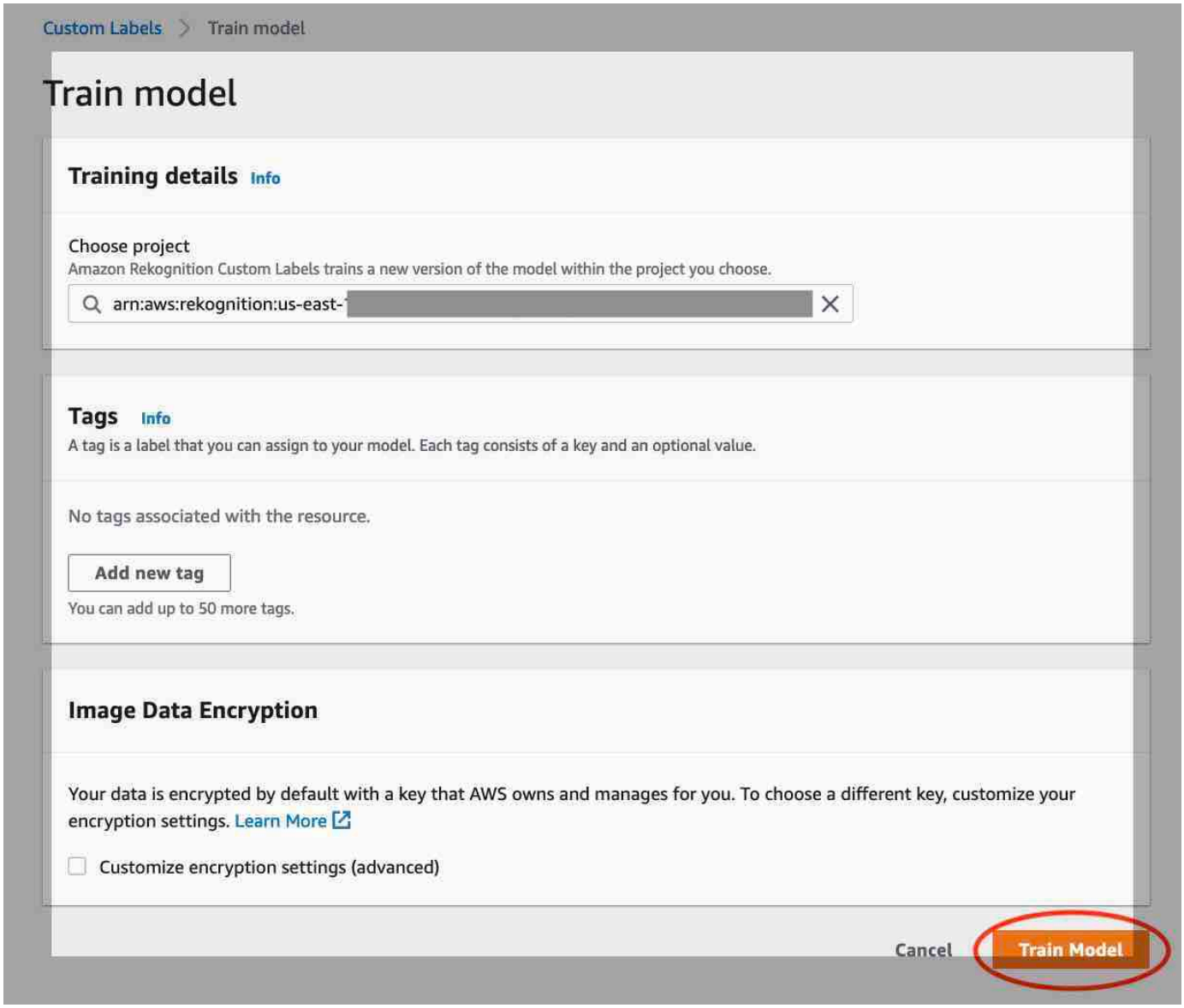
Nesta etapa, seu modelo é treinado. Os conjuntos de dados de treinamento e teste são configurados automaticamente para você. Depois que o treinamento for concluído com sucesso, é possível ver os resultados gerais da avaliação e os resultados da avaliação de imagens de teste individuais. Para obter mais informações, consulte Como treinar um modelo do Amazon Rekognition Custom Labels.

Como treinar seu modelo do

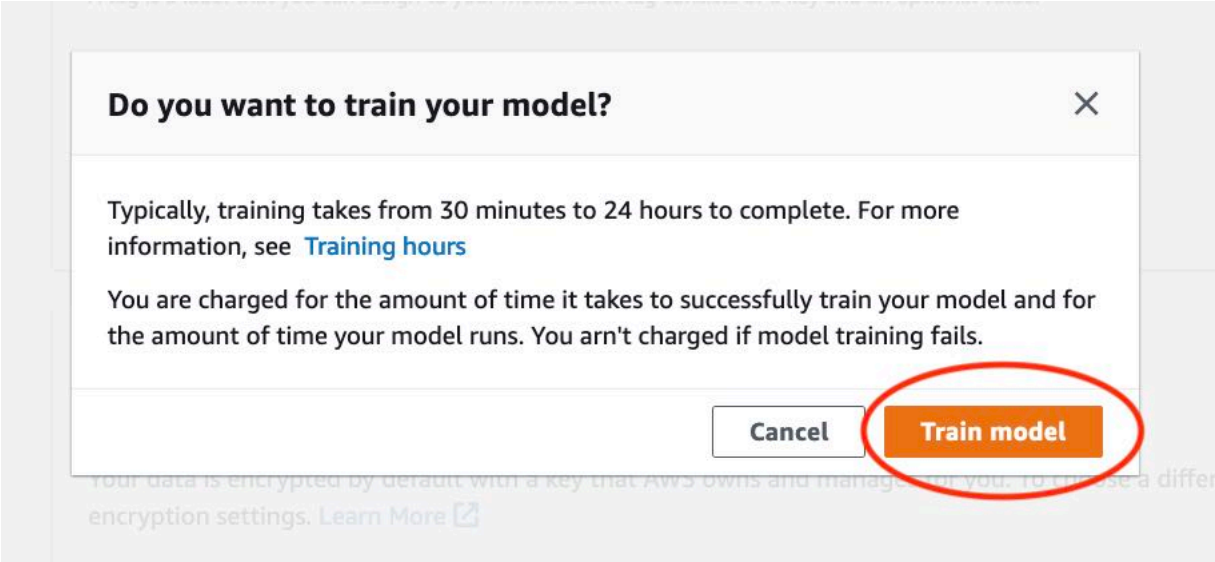
1. Na página do conjunto de dados, escolha Treinar modelo. A imagem a seguir mostra o console com o botão Treinar modelo.



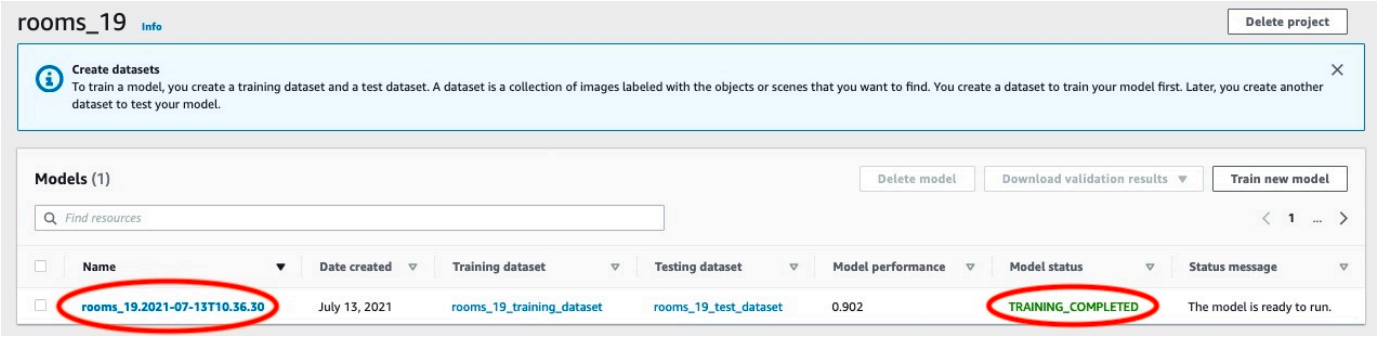
2. Na página Treinar modelo, escolha Treinar modelo. A imagem abaixo mostra o botão Treinar modelo. Observe que o nome do recurso da Amazon (ARN) do projeto está na caixa de edição Escolher projeto.



3. Na caixa de diálogo Quer treinar seu modelo? mostrada na imagem a seguir, escolha Treinar modelo.

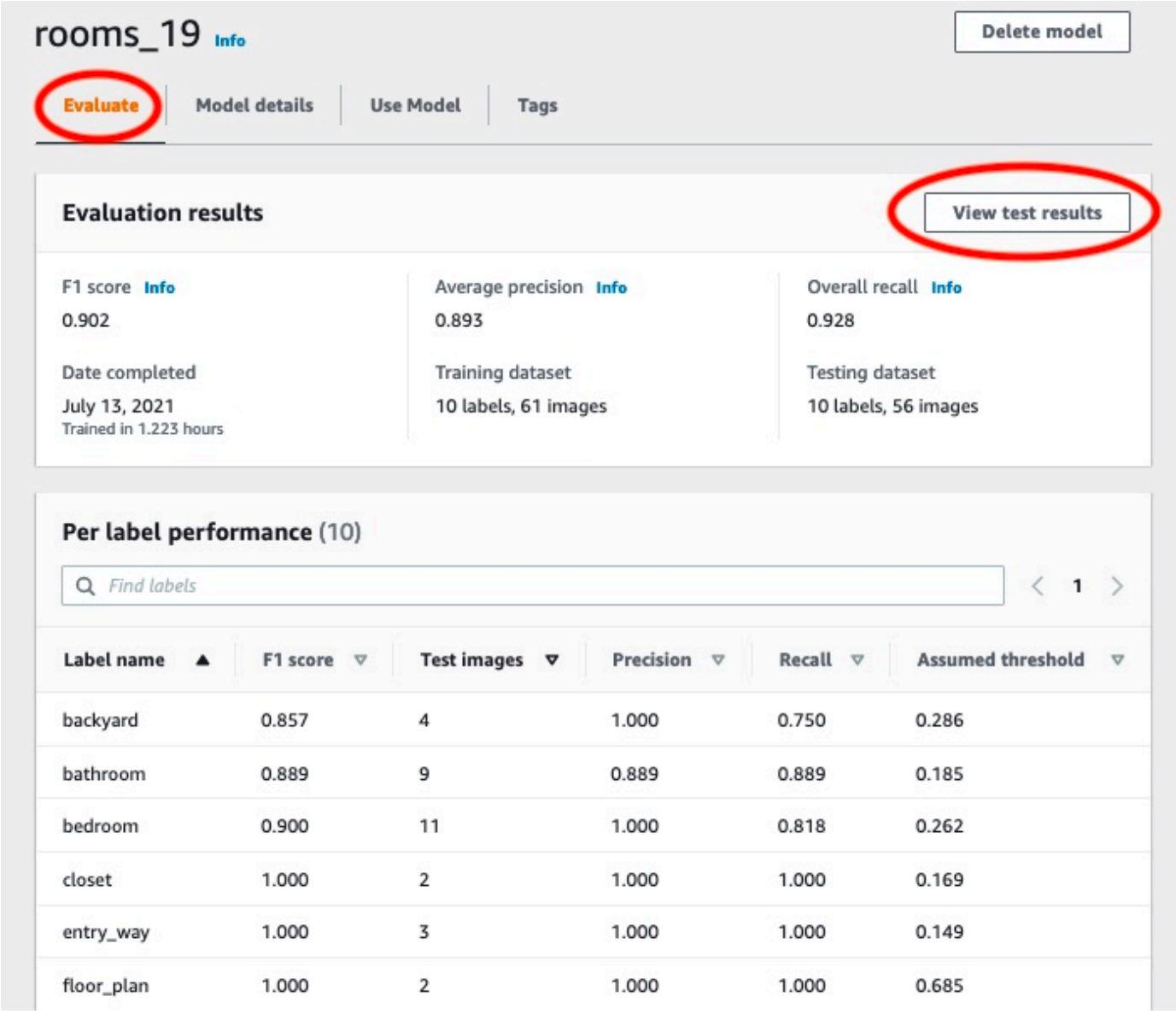


4. Após a conclusão do treinamento, escolha o nome do modelo. O treinamento estará concluído quando o status do modelo for TRAINING_COMPLETED, conforme demonstrado na captura de tela do console a seguir.

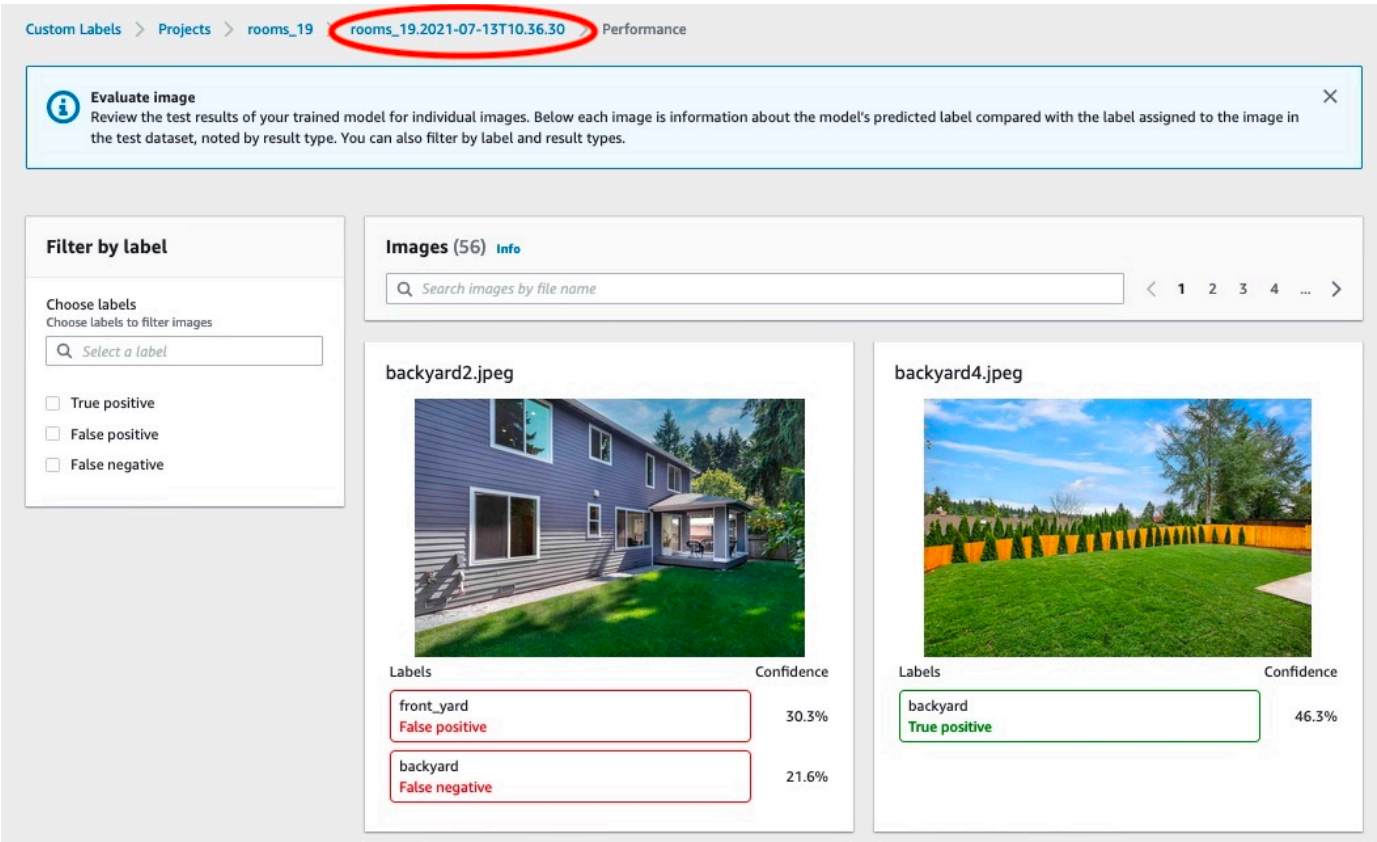


5. Escolha o botão Avaliar para ver os resultados da avaliação. Para obter informações sobre como avaliar um modelo, consulte Como melhorar um modelo treinado do Amazon Rekognition Custom Labels.

6. Escolha Exibir resultados do teste para ver os resultados de imagens de teste individuais. Conforme visto na captura de tela a seguir, o painel de avaliação mostra algumas métricas, como pontuação F1, precisão, recall e número de imagens de teste para cada rótulo. Métricas gerais, como média, precisão e recall, também são exibidas.



7. Depois de visualizar os resultados do teste, escolha o nome do modelo para retornar à página do modelo. A captura de tela a seguir do painel de desempenho destaca onde você pode clicar para retornar à página do modelo.



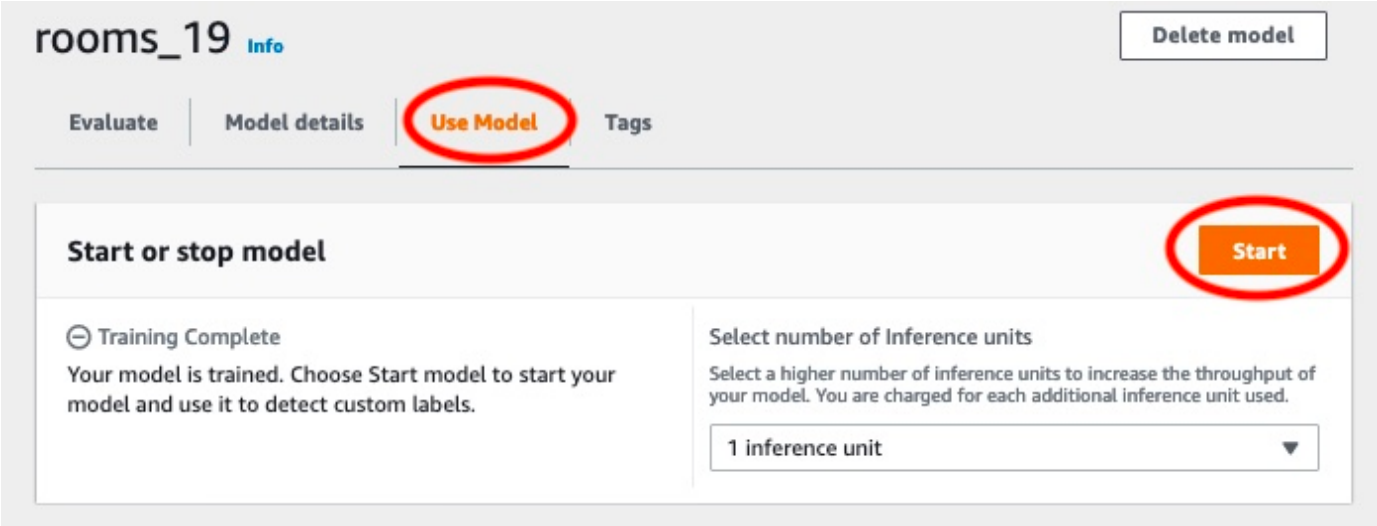
Etapa 3: inicie seu modelo

Nesta etapa, seu modelo é iniciado. Depois que o modelo começar, é possível usá-lo para analisar novas imagens.

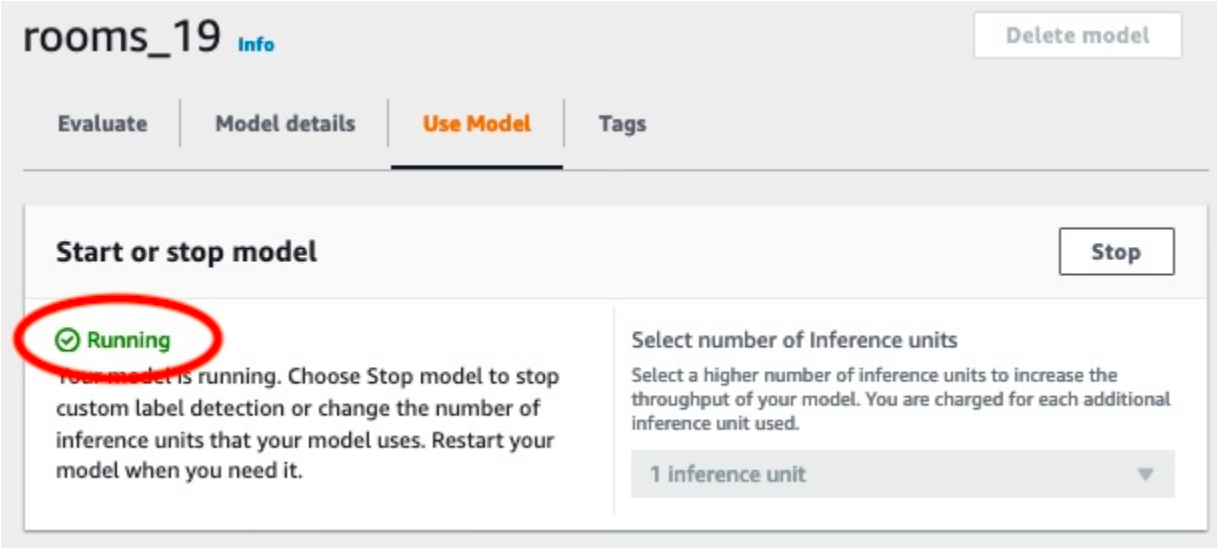
Há uma cobrança pela quantidade de tempo que o modelo é executado. Interrompa seu modelo se você não precisar analisar as imagens. Será possível reiniciar o modelo mais tarde. Para obter mais informações, consulte Como executar um modelo do Amazon Rekognition Custom Labels.

Para iniciar o seu modelo

1. Escolha a guia Usar modelo na página do modelo.
2. Na seção Iniciar ou interromper o modelo, faça o seguinte:
   a. Escolha Iniciar.
   b. Na caixa de diálogo Iniciar modelo, escolha Iniciar. A imagem a seguir mostra o botão Iniciar no painel de controle do modelo.



3. Espere até que o modelo esteja em execução. A captura de tela a seguir mostra o console enquanto o modelo está em execução, onde o status na seção Iniciar ou interromper o modelo é Em execução.



4. Use seu modelo para classificar imagens. Para obter mais informações, consulte Etapa 4: analise uma imagem com seu modelo.

Etapa 4: analise uma imagem com seu modelo

Você analisa uma imagem chamando a DetectCustomLabelsAPI. Nesta etapa, você usa o comando `detect-custom-labels` AWS Command Line Interface (AWS CLI) para analisar uma imagem de exemplo. Você recebe o AWS CLI comando no console Amazon Rekognition Custom Labels. O

console configura o AWS CLI comando para usar seu modelo. Só é preciso fornecer uma imagem que esteja armazenada em um bucket do Amazon S3. Este tópico fornece uma imagem que é possível usar para cada projeto de exemplo.

> **Note**
>
> O console também fornece um código de exemplo em Python.

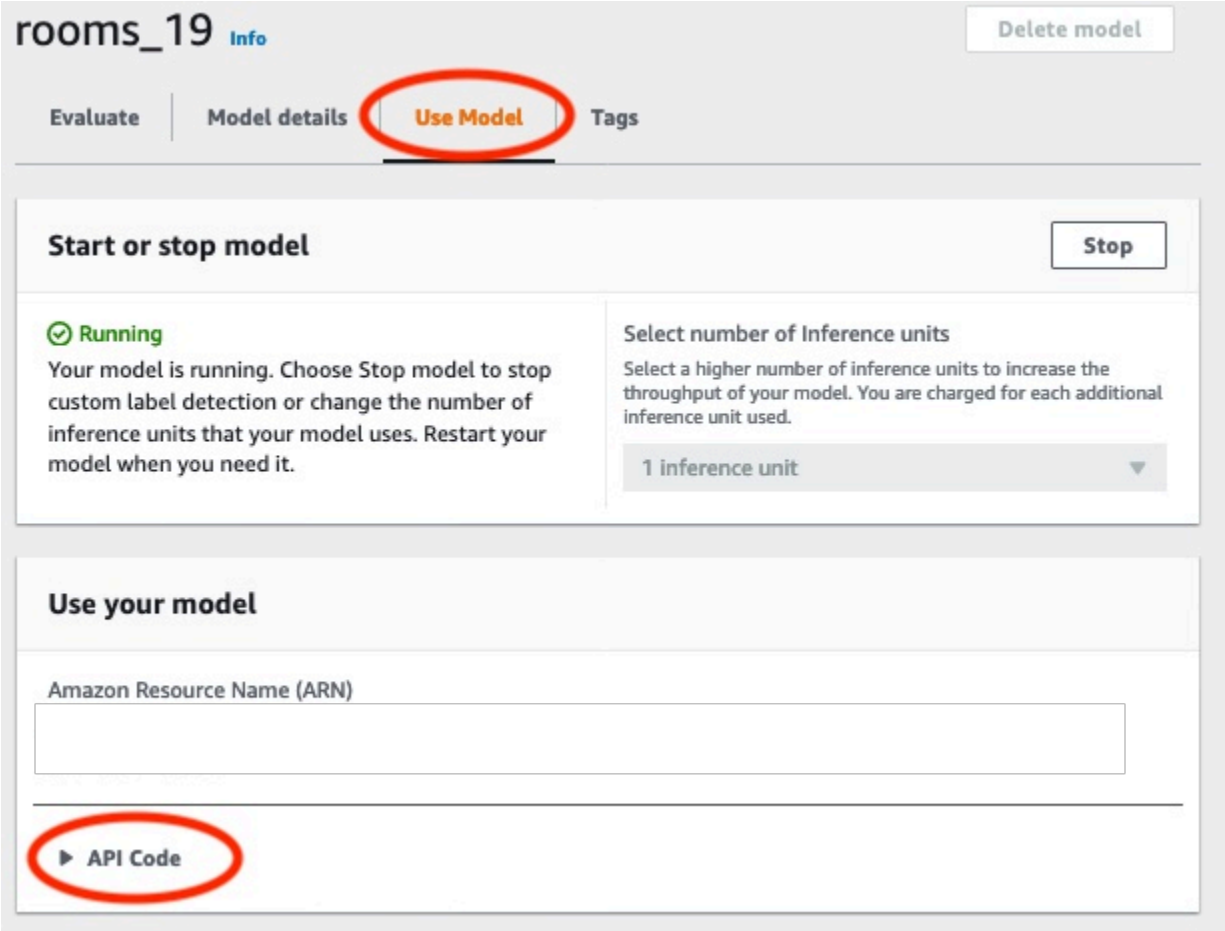
A saída de `detect-custom-labels` inclui uma lista de rótulos encontrados na imagem, caixas delimitadoras (se o modelo encontrar a localização dos objetos) e a confiança que o modelo tem na precisão das previsões.

Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.

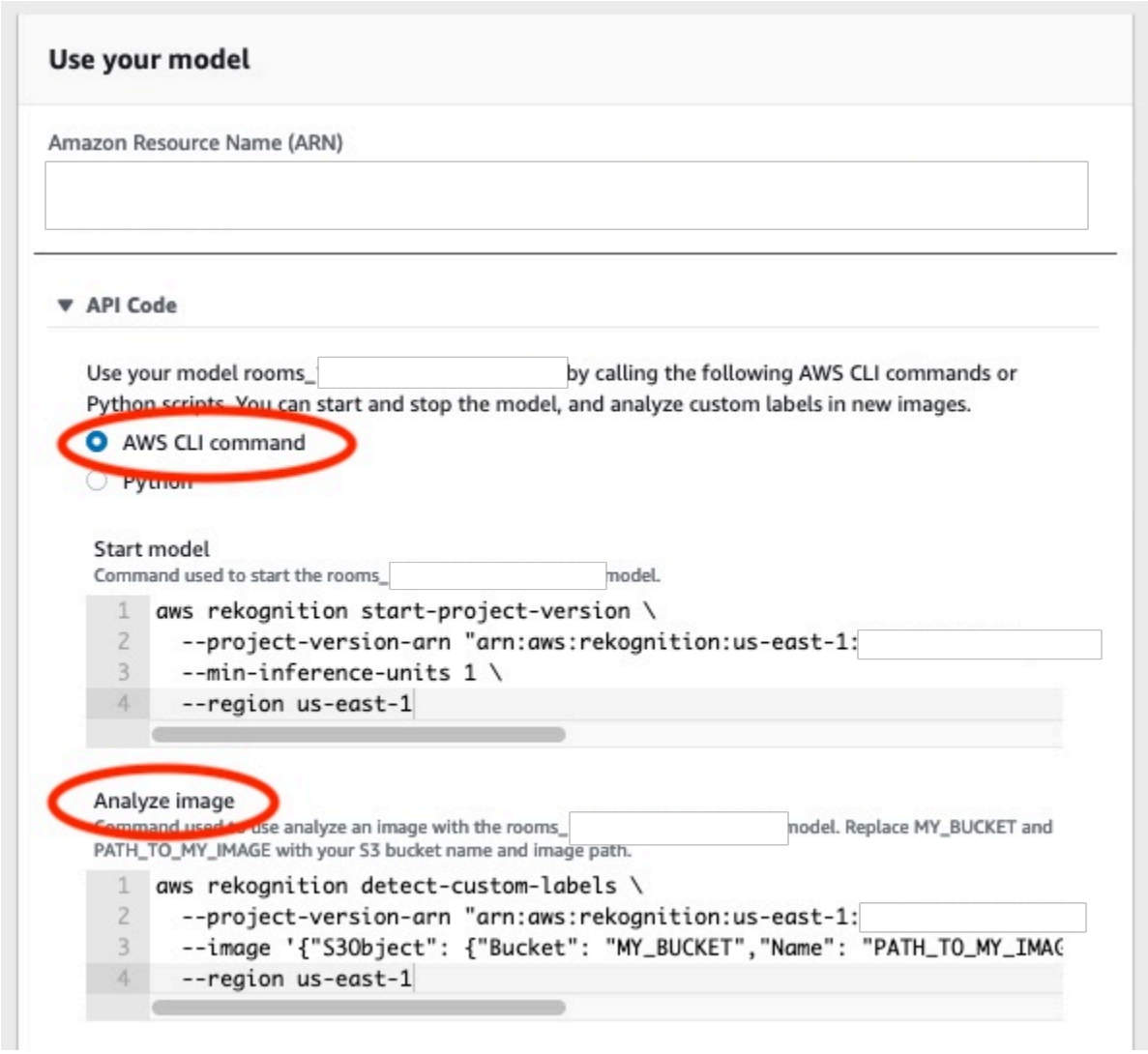
Para analisar uma imagem (console)

1. <textobject><phrase>O status do modelo é exibido como Em execução, com o botão Interromper para parar o modelo em execução.</phrase></textobject>

   Se você ainda não o fez, configure AWS CLI o. Para instruções, consulte the section called “Etapa 4: configurar o AWS CLI e AWS SDKs”.
2. Se ainda não tiver iniciado, comece a executar seu modelo. Para obter mais informações, consulte Etapa 3: inicie seu modelo.
3. Escolha a guia Usar modelo e escolha o código da API. O painel de status do modelo mostrado abaixo mostra o modelo como Em execução, com um botão Interromper para parar a execução do modelo e uma opção para exibir a API.



4. Escolha o Comando da AWS CLI.
5. Na seção Analisar imagem, copie o AWS CLI comando que chama`detect-custom-labels`. A imagem a seguir do console do Rekognition mostra a seção “Analisar imagem” com o comando da AWS CLI para detectar rótulos personalizados em uma imagem usando um modelo de machine learning e instruções para iniciar o modelo e fornecer detalhes da imagem.



6. Faça upload de uma imagem de exemplo para um bucket do Amazon S3. Para instruções, consulte Como obter uma imagem de exemplo.
7. No prompt de comando, digite o AWS CLI comando que você copiou na etapa anterior. Ele se parece com o exemplo a seguir.

   O valor de `--project-version-arn` deve ser o nome do recurso da Amazon (ARN) do seu modelo. O valor de `--region` deve ser a região da AWS na qual você criou o modelo.

   Altere `MY_BUCKET` e `PATH_TO_MY_IMAGE` use o bucket e a imagem do Amazon S3 que você usou na etapa anterior.

   Se você estiver usando o perfil custom-labels-access para obter credenciais, adicione o parâmetro `--profile custom-labels-access`.

```
aws rekognition detect-custom-labels \
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET","Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

Se o modelo encontrar objetos, cenas e conceitos, a saída JSON do comando AWS CLI deverá ser semelhante à seguinte. O `Name` é o nome do rótulo em nível de imagem que o modelo encontrou. A `Confidence` (0-100) é a confiança do modelo na precisão da previsão.

```
{
    "CustomLabels": [
        {
            "Name": "living_space",
            "Confidence": 83.41299819946289
        }
    ]
}
```

Se o modelo encontrar a localização dos objetos ou encontrar a marca, as caixas delimitadoras rotuladas serão retornadas. A `BoundingBox`contém a localização de uma caixa que circunda o objeto. O `Name` é o objeto que o modelo encontrou na caixa delimitadora. A `Confidence` é a confiança do modelo de que a caixa delimitadora contém o objeto.

```
{
    "CustomLabels": [
        {
            "Name": "textract",
            "Confidence": 87.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

8. Continue usando o modelo para analisar outras imagens. Interrompa o modelo se não estiver mais usando. Para obter mais informações, consulte Etapa 5: interrompa seu modelo.

Como obter uma imagem de exemplo

É possível usar as imagens a seguir com a operação `DetectCustomLabels`. Há uma imagem para cada projeto. Para usar as imagens, faça upload delas em um bucket do S3.

Para usar uma imagem de exemplo

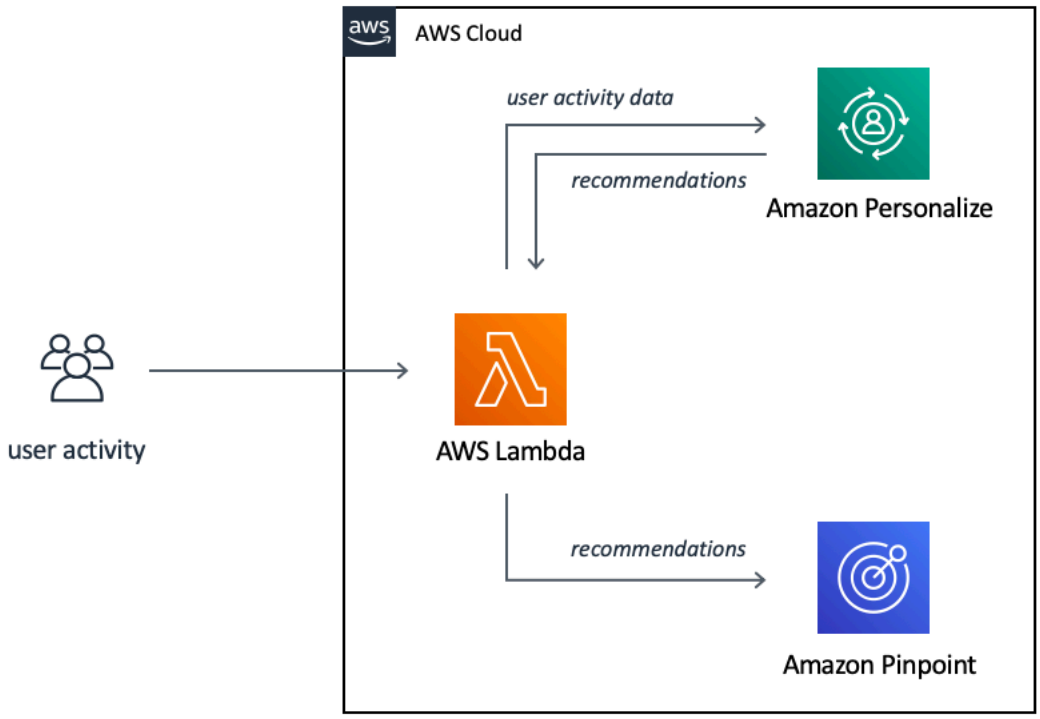
1. Clique com o botão direito do mouse na imagem a seguir que corresponde ao projeto de exemplo que você está usando. Em seguida, escolha Salvar imagem para salvar a imagem no seu computador. A opção do menu pode ser diferente, dependendo do navegador que você está usando.
2. Faça o upload da imagem para um bucket do Amazon S3 que pertence à sua AWS conta e está na mesma AWS região em que você está usando etiquetas personalizadas do Amazon Rekognition.

   Para obter mais informações, consulte Fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

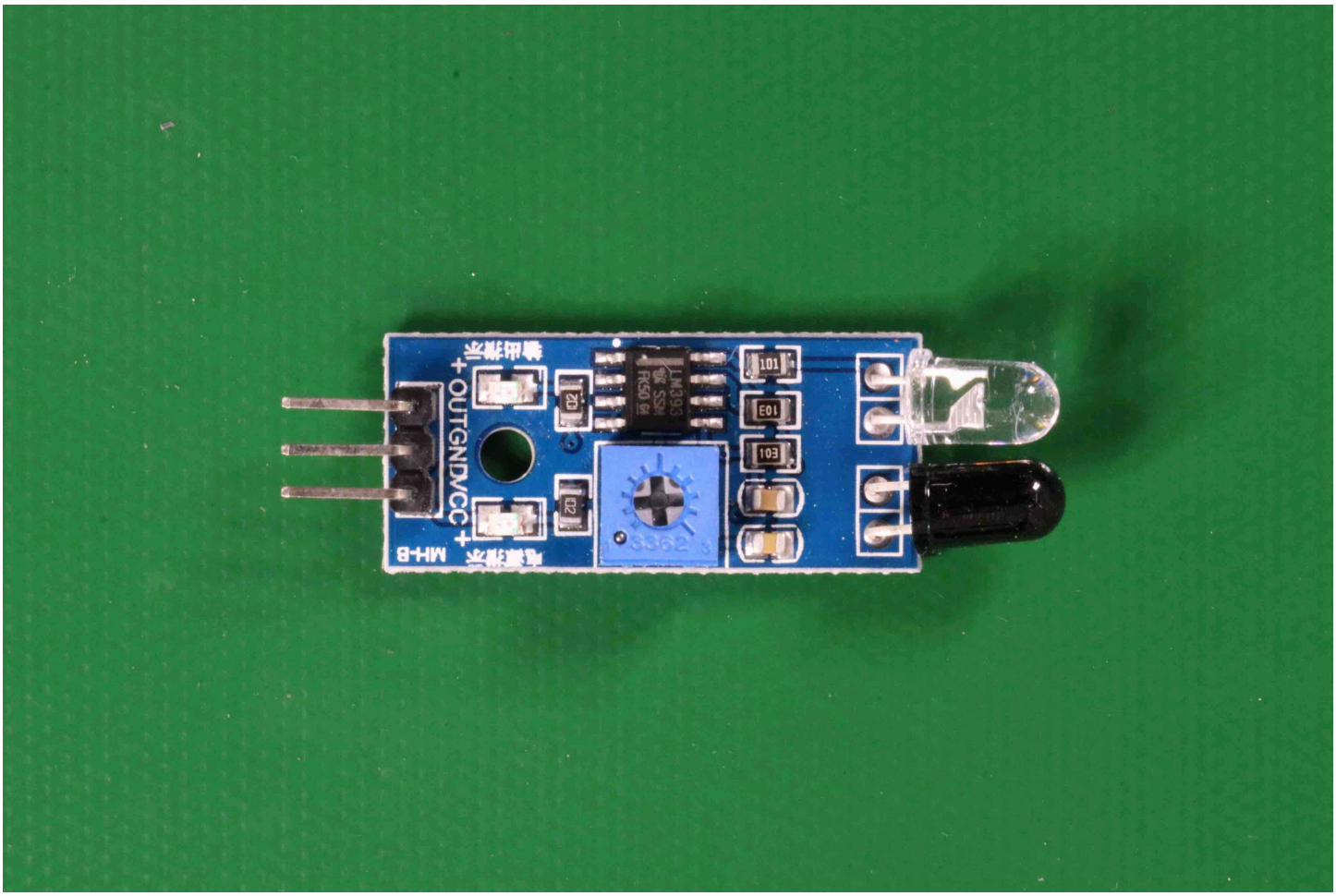
Classificação de imagens

Classificação com vários rótulos

Detecção de marca



Localização de objetos

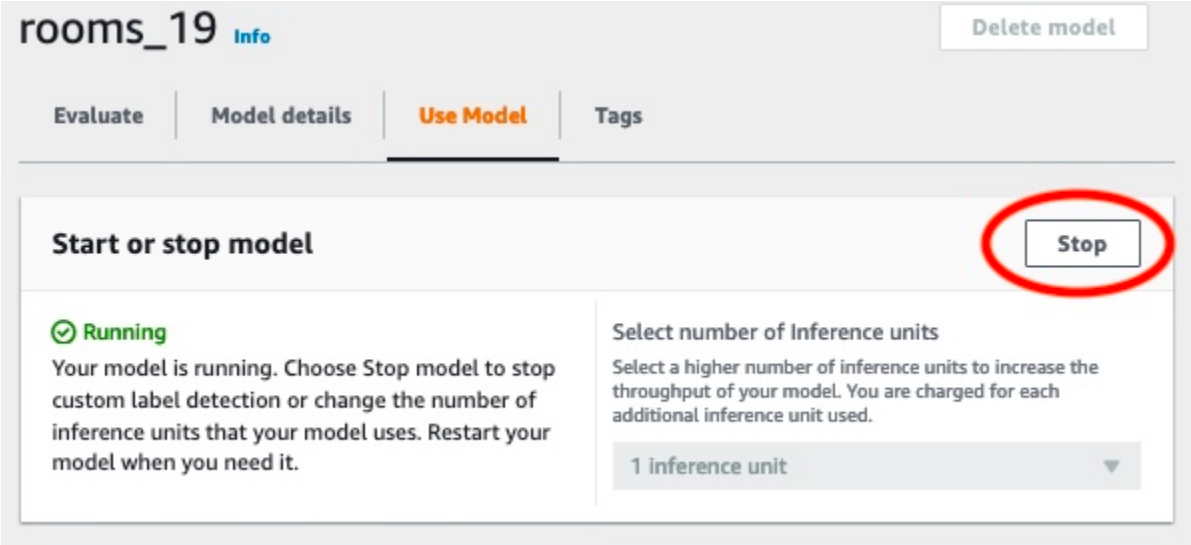


Etapa 5: interrompa seu modelo

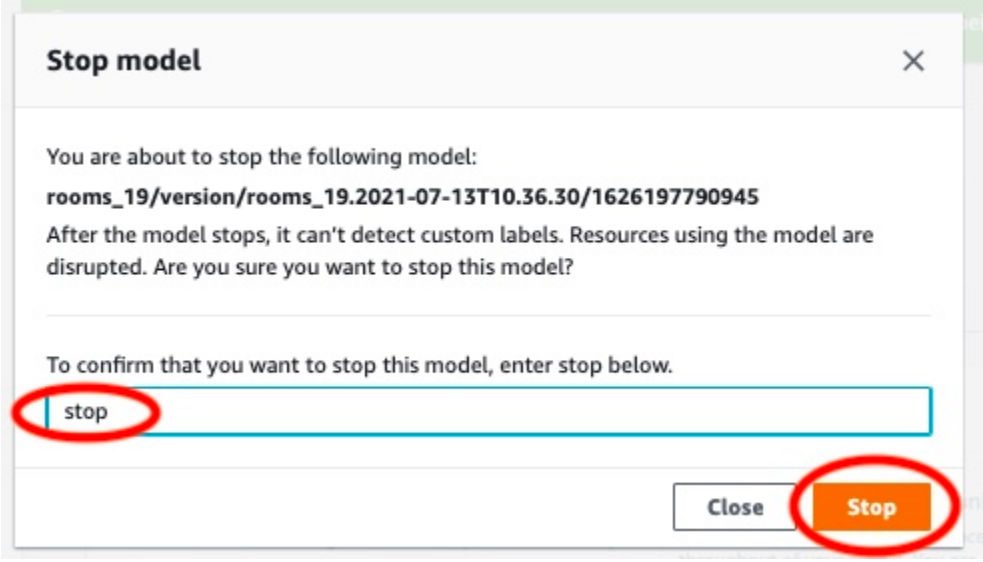
Nesta etapa, seu modelo para de ser executado. Há uma cobrança pela quantidade de tempo que o modelo está em execução. Se terminou de usar o modelo, deve interrompê-lo.

Para interromper seu modelo

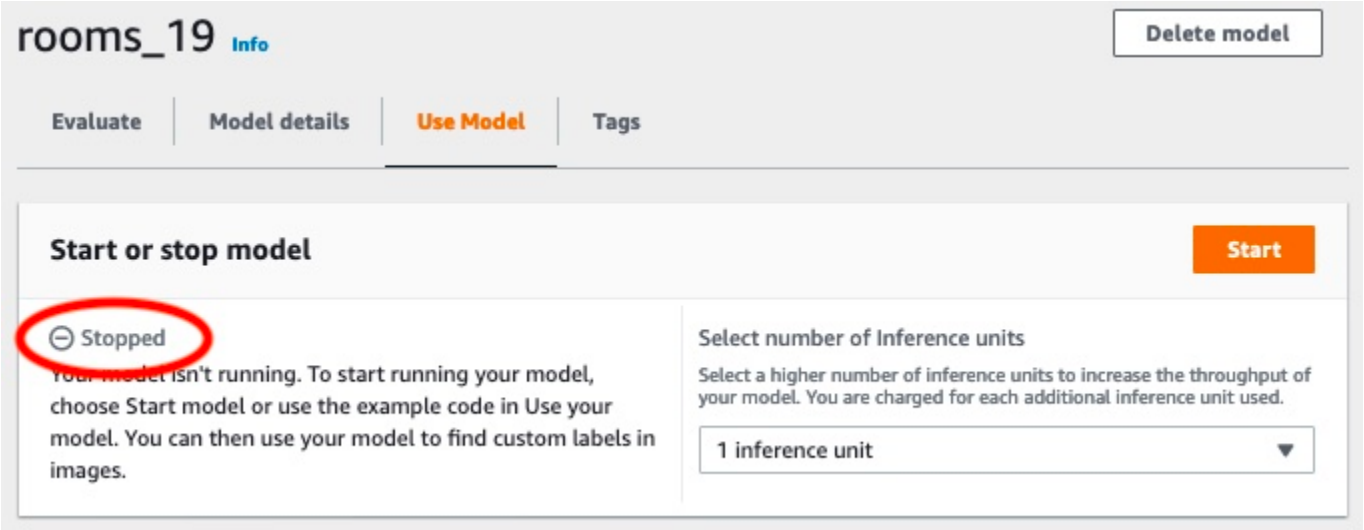
1. Na seção Iniciar ou interromper modelo, escolha Interromper.



2. Na caixa de diálogo Interromper modelo, insira interromper para confirmar que deseja interromper o modelo.



3. Escolha Parar para interromper seu modelo. O modelo foi interrompido quando o status na seção Iniciar ou interromper modelo é Interrompido. Na captura de tela a seguir, a seção da interface do usuário tem a opção de iniciar ou interromper um modelo de machine learning. O status do modelo é exibido como “Interrompido” com um botão “Iniciar” para iniciar o modelo e uma lista suspensa para selecionar o número de unidades de inferência.



Etapa 6: próximas etapas

Depois de testar os projetos de exemplos, é possível usar suas próprias imagens e conjuntos de dados para criar seu próprio modelo. Para obter mais informações, consulte Noções básicas do Amazon Rekognition Custom Labels.

Use as informações de rotulagem na tabela a seguir para treinar modelos semelhantes aos projetos de exemplo.

Exemplo	Imagens de treinamento	Imagens de teste
Classificação de imagens (Cômodos)	1 Rótulo em nível de imagem por imagem	1 Rótulo em nível de imagem por imagem
Classificação com vários rótulos (flores)	Vários rótulos em nível de imagem por imagem	Vários rótulos em nível de imagem por imagem
Detecção de marca (logotipos)	rótulos de nível de imagem (também é possível usar caixas delimitadoras rotuladas)	Caixas delimitadoras rotuladas
Localização de imagens (placas de circuito)	Caixas delimitadoras rotuladas	Caixas delimitadoras rotuladas

O Classificar imagens mostra como criar um projeto, conjuntos de dados e modelos para um modelo de classificação de imagens.

Para obter informações detalhadas sobre a criação de conjuntos de dados e modelos de treinamento, consulte Como criar um modelo do Amazon Rekognition Custom Labels.

Classificar imagens

Este tutorial mostra como criar o projeto e os conjuntos de dados para um modelo que classifica objetos, cenas e conceitos encontrados em uma imagem. O modelo classifica a imagem inteira. Por exemplo, seguindo este tutorial, é possível treinar um modelo para reconhecer locais domésticos, como uma sala de estar ou cozinha. O tutorial também mostra como usar o modelo para analisar imagens.

Antes de iniciar o tutorial, recomendamos que você leia Noções básicas do Amazon Rekognition Custom Labels.

Neste tutorial, conjuntos de dados de treinamento de teste são criados ao fazer upload de imagens do seu computador local. Posteriormente, você atribui rótulos em nível de imagem às imagens em seus conjuntos de dados de treinamento e teste.

O modelo que você cria classifica as imagens como pertencentes ao conjunto de rótulos em nível de imagem que você atribui às imagens do conjunto de dados de treinamento. Por exemplo, se o conjunto de rótulos em nível de imagem em seu conjunto de dados de treinamento for `kitchen`, `living_room`, `patio` e `backyard`, o modelo possivelmente poderá encontrar todos esses rótulos em nível de imagem em uma única imagem.

> **Note**
>
> É possível criar modelos para diferentes propósitos, como descobrir a localização dos objetos em uma imagem. Para obter mais informações, consulte Decida o tipo do seu modelo.

Etapa 1: colete suas imagens

São necessários dois conjuntos de imagens. Um conjunto para adicionar ao seu conjunto de dados de treinamento. Outro conjunto para adicionar ao seu conjunto de dados de teste. As imagens devem representar os objetos, cenas e conceitos que você deseja que seu modelo classifique. As imagens devem estar no formato PNG ou JPEG. Para obter mais informações, consulte Como preparar imagens.

Devem haver pelo menos 10 imagens para seu conjunto de dados de treinamento e 10 imagens para seu conjunto de dados de teste.

Se ainda não tiver imagens, use as imagens do projeto de classificação de exemplo Cômodos. Depois de criar o projeto, as imagens de treinamento e teste estão nos seguintes locais de bucket do Amazon S3:

- Imagens de treinamento: `s3://custom-labels-console-region-numbers/assets/rooms_version number_test_dataset/`
- Imagens de teste: `s3://custom-labels-console-region-numbers/assets/rooms_version number_test_dataset/`

`region`é a AWS região na qual você está usando o console Amazon Rekognition Custom Labels. `number`sé um valor que o console atribui ao nome do bucket. `Version number`é o número da versão do projeto de exemplo, começando em 1.

O procedimento a seguir armazena imagens do projeto Cômodos em pastas locais em seu computador chamadas `training` e `test`.

Para baixar os arquivos de imagem do projeto de exemplo Cômodos

1. Crie o projeto Cômodos. Para obter mais informações, consulte Etapa 1: escolha um projeto de exemplo.
2. No prompt de comando, insira o comando a seguir para baixar as imagens de treinamento.

```
aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version
 number_training_dataset/ training --recursive
```

3. No prompt de recomendação, insira o comando a seguir para baixar as imagens de teste.

```
aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version
 number_test_dataset/ test --recursive
```

4. Mova duas das imagens da pasta de treinamento para uma pasta separada de sua escolha. As imagens serão usadas para testar seu modelo treinado em Etapa 9: analise uma imagem com seu modelo.

Etapa 2: decida suas classes

Faça uma lista das classes que deseja que o modelo encontre. Por exemplo, se você estiver treinando um modelo para reconhecer cômodos em uma casa, é possível classificar a imagem a seguir como `living_room`.

Cada classe é mapeada para um rótulo em nível de imagem. Posteriormente, você atribui rótulos em nível de imagem às imagens em seus conjuntos de dados de treinamento e teste.

Se estiver usando as imagens do projeto de exemplo Cômodos, os rótulos em nível de imagem são quintal, banheiro, quarto, armário, caminho_de_entrada, planta_baixa, jardim_da_frente, cozinha, sala_de_estar e pátio.

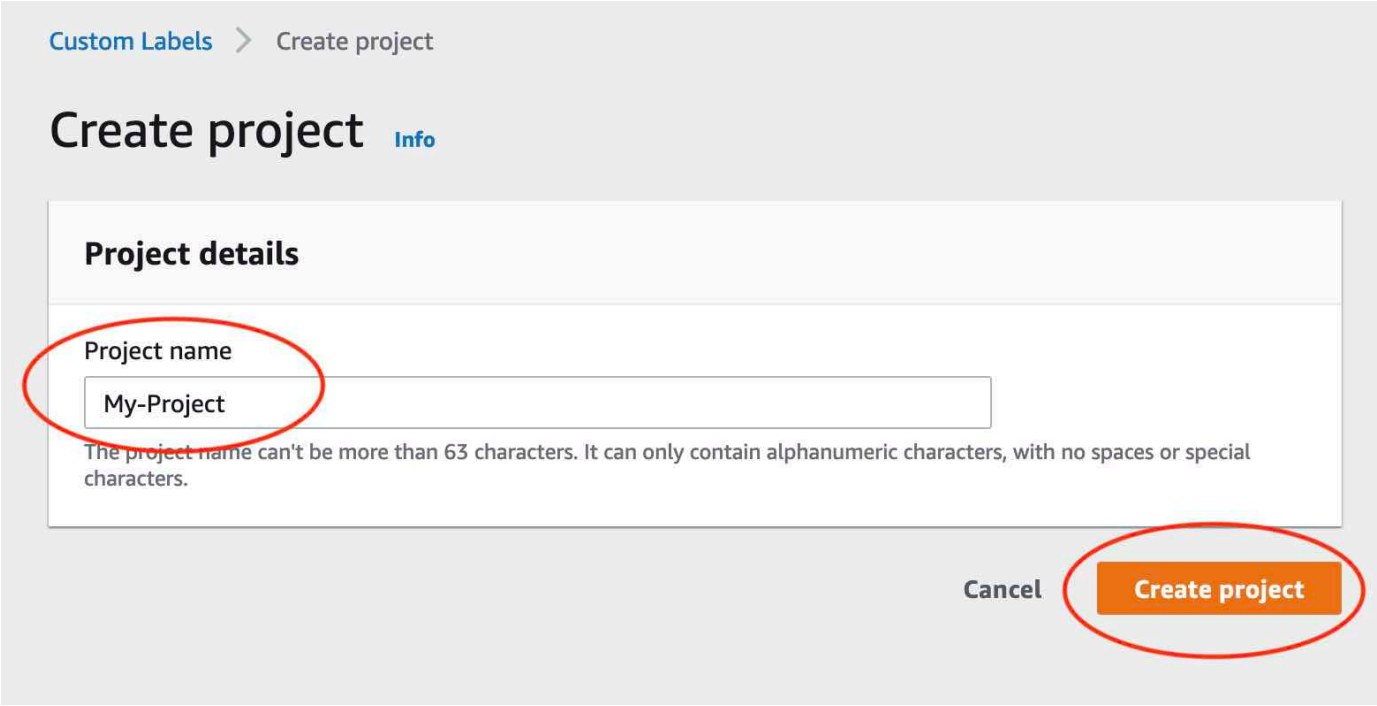
Etapa 3: crie um projeto

Para gerenciar seus conjuntos de dados e modelos, você cria um projeto. Cada projeto deve abordar um único caso de uso, como reconhecer cômodos em uma casa.

Para criar um projeto (console)

1. Se ainda não tiver feito isso, configure o console do Amazon Rekognition Custom Labels. Para obter mais informações, consulte Como configurar o Amazon Rekognition Custom Labels.
2. Faça login no AWS Management Console e abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
3. No painel esquerdo, escolha Usar rótulos personalizados. A página inicial do Amazon Rekognition Custom Labels é exibida.

4. Na página inicial do Amazon Rekognition Custom Labels, escolha Conceitos básicos
5. No painel de navegação esquerdo, selecione Projetos.
6. Na página de projetos, escolha Criar projeto.
7. Em Nome do projeto, digite um nome para o seu projeto.
8. Escolha Criar projeto para criar seu projeto.



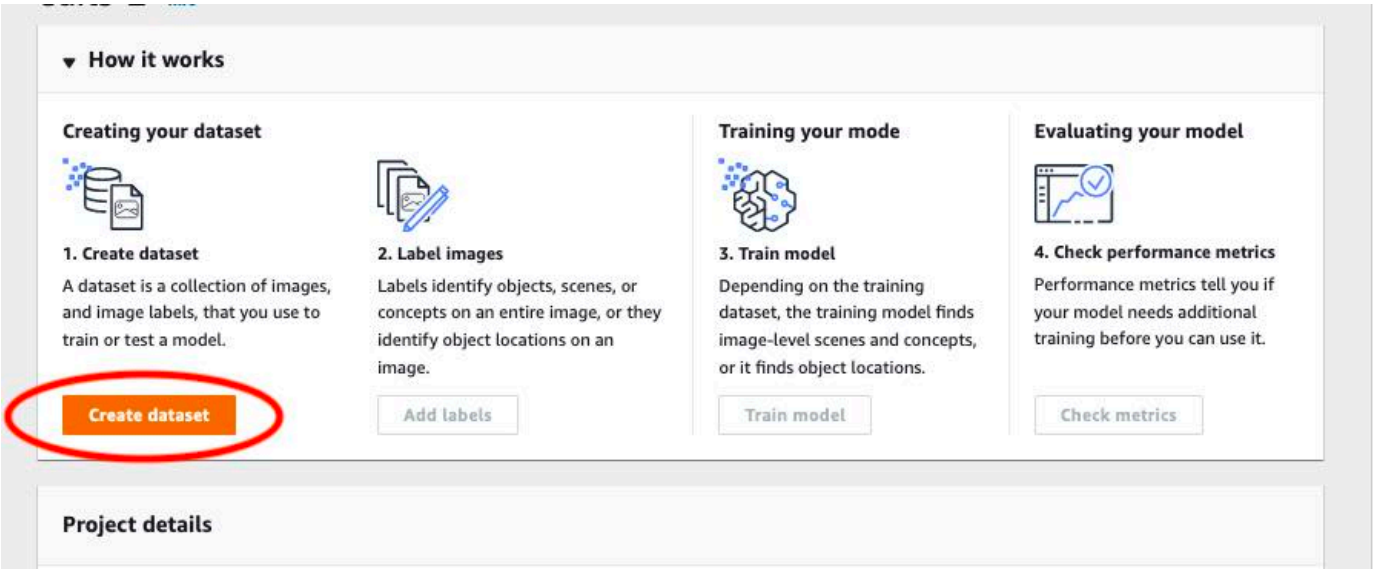
Etapa 4: crie conjuntos de dados de treinamento e teste

Nesta etapa, um conjunto de dados de treinamento e um conjunto de dados de teste são criados ao fazer upload de imagens do seu computador local. É possível fazer upload de até 30 imagens por vez. Se tiver muitas imagens para carregar, considere criar os conjuntos de dados importando as imagens de um bucket do Amazon S3. Para obter mais informações, consulte Importar imagens de um bucket do Amazon S3.

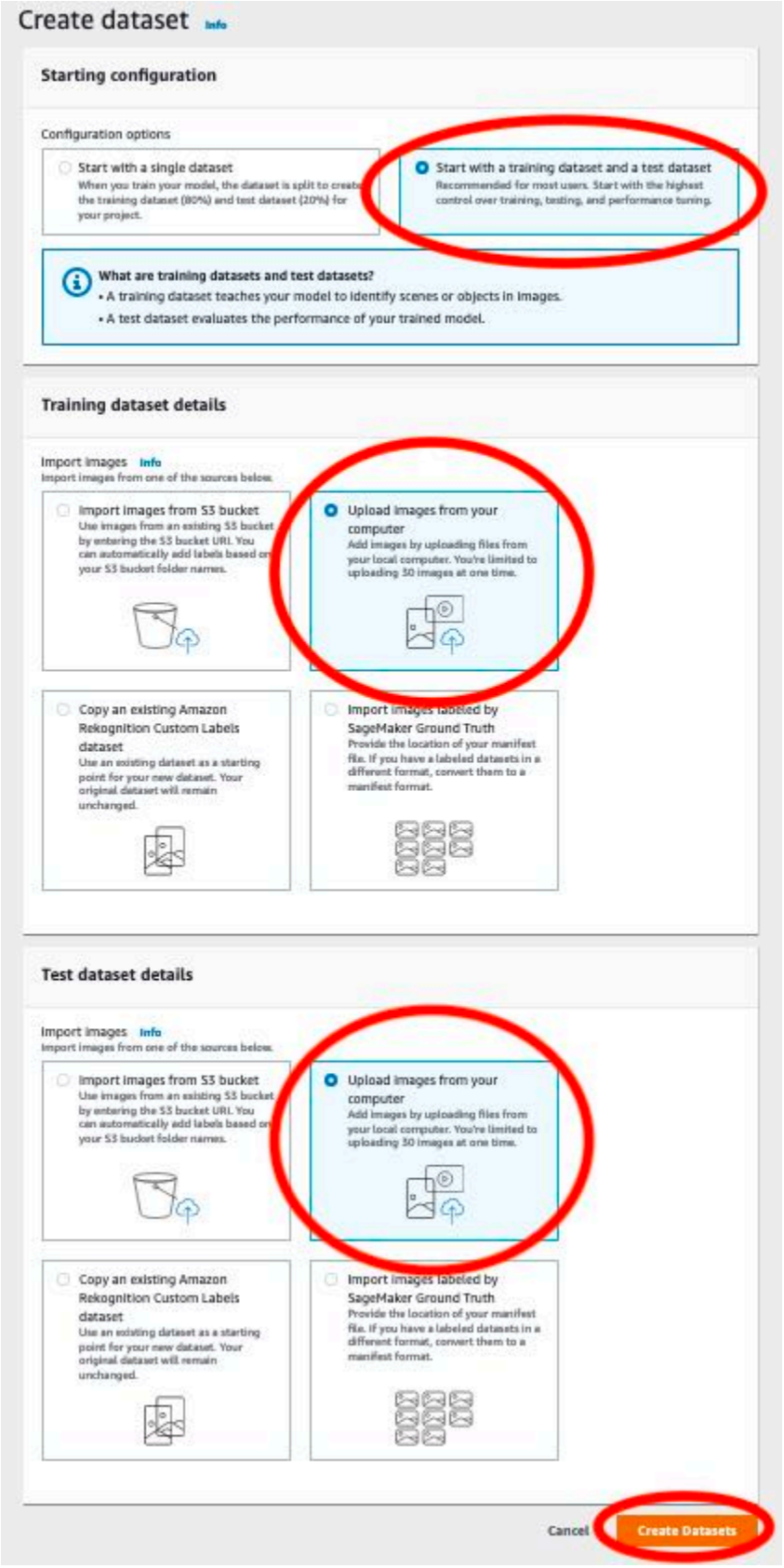
Para obter mais informações sobre conjuntos de dados, consulte Como gerenciar conjuntos de dados.

Para criar um conjunto de dados usando imagens em um computador local (console)

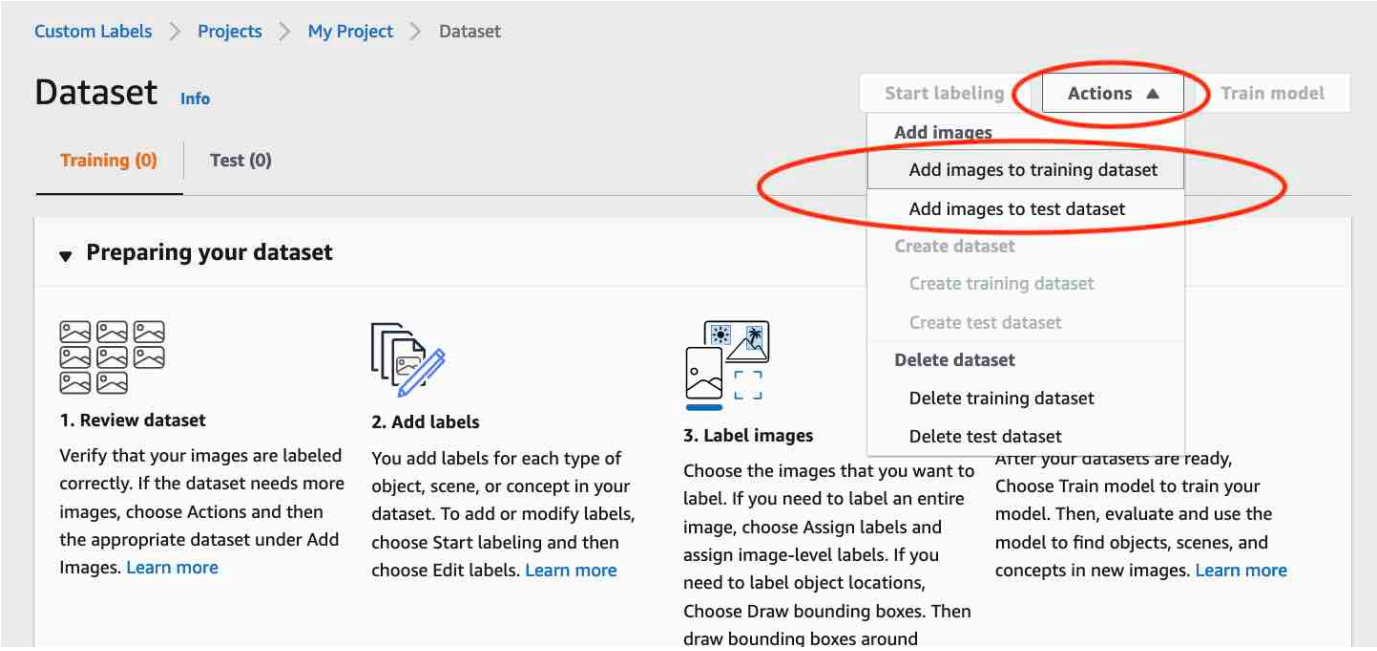
1. Na página de detalhes do projeto, escolha Criar conjunto de dados.



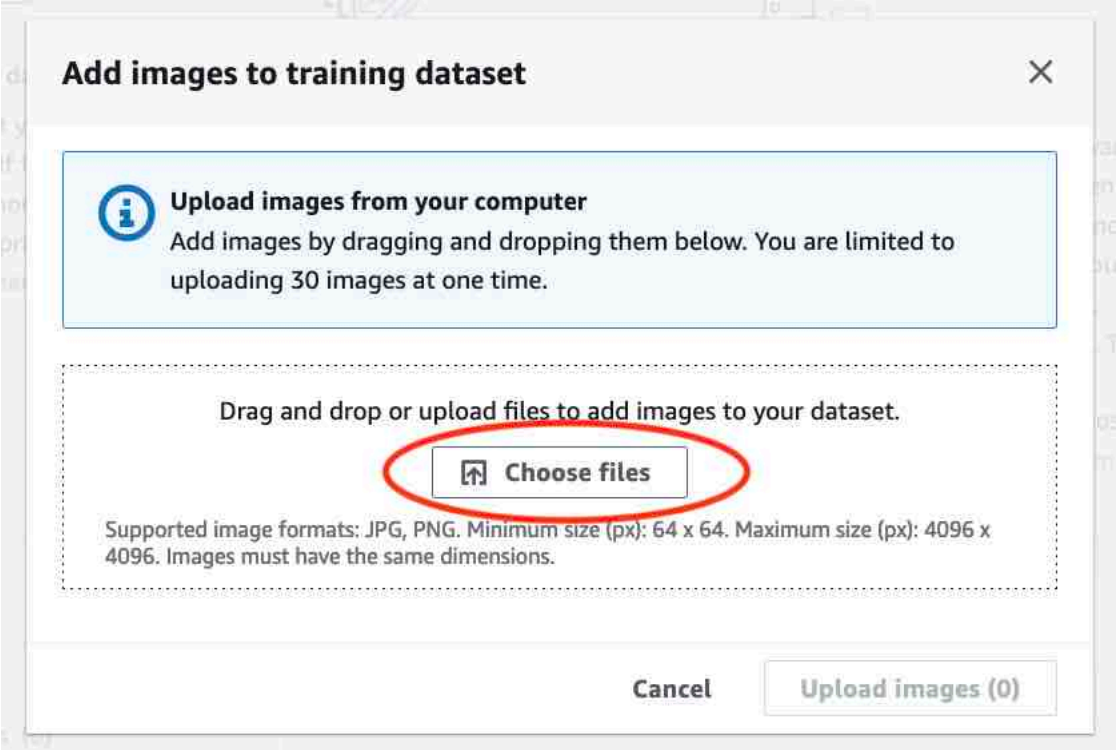
2. Na seção Configuração inicial, escolha Iniciar com um conjunto de dados de treinamento e um conjunto de dados de teste.
3. Na seção Detalhes do conjunto de dados de treinamento, escolha Fazer upload de imagens do seu computador.
4. Na seção Detalhes do conjunto de dados de teste, escolha Fazer upload de imagens do seu computador.
5. Escolha Criar conjuntos de dados.


Create dataset Info
Starting configuration
Configuration options
Start with a single dataset
When you train your model, the dataset is split to create the training dataset (80%) and test dataset (20%) for your project.
Start with a training dataset and a test dataset
Recommended for most users. Start with the highest control over training, testing, and performance tuning.
What are training datasets and test datasets?
• A training dataset teaches your model to identify scenes or objects in images.
• A test dataset evaluates the performance of your trained model.
Training dataset details
Import images Info
Import images from one of the sources below.
Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URL. You can automatically add labels based on your S3 bucket folder names.
Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 30 images at one time.
Copy an existing Amazon Rekognition Custom Labels dataset
Use an existing dataset as a starting point for your new dataset. Your original dataset will remain unchanged.
Import images labeled by SageMaker Ground Truth
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.
Test dataset details
Import images Info
Import images from one of the sources below.
Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URL. You can automatically add labels based on your S3 bucket folder names.
Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 30 images at one time.
Copy an existing Amazon Rekognition Custom Labels dataset
Use an existing dataset as a starting point for your new dataset. Your original dataset will remain unchanged.
Import images labeled by SageMaker Ground Truth
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.
Cancel
Create Datasets

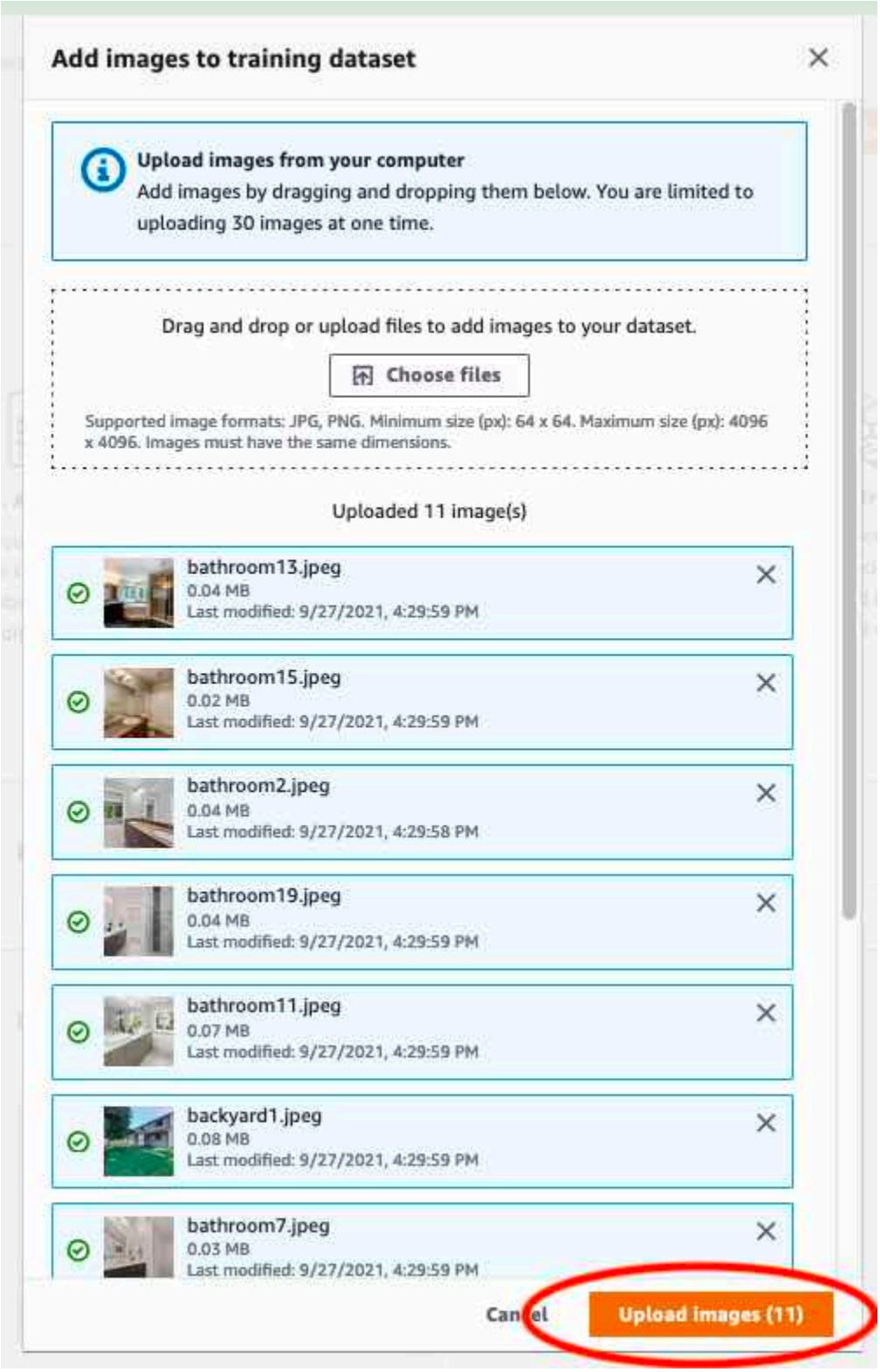
6. Uma página do conjunto de dados aparece com uma guia Treinamento e uma guia Teste para os respectivos conjuntos de dados.
7. Na página conjuntos de dados, escolha a guia Treinamento.
8. Escolha Ações e escolha Adicionar imagens ao conjunto de dados de treinamento.



9. Na caixa de diálogo Adicionar imagens ao conjunto de dados de treinamento, escolha Escolher arquivos.



10. Escolha as imagens que você deseja fazer upload no conjunto de dados. É possível fazer upload de até 30 imagens por vez.
11. Escolha Fazer upload de imagens. Pode levar alguns segundos para que o Amazon Rekognition Custom Labels adicione as imagens ao conjunto de dados.



12. Se tiver mais imagens para adicionar ao conjunto de dados de treinamento, repita as etapas 9 a 12.
13. Selecione a guia Testar.

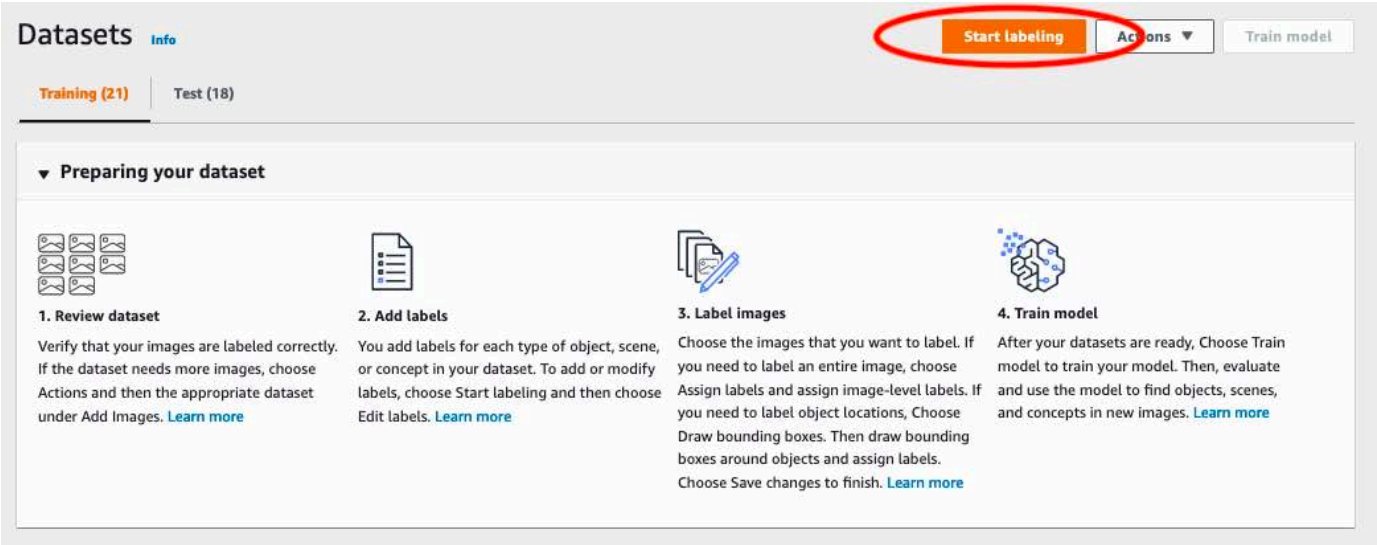
14. Repita as etapas de 8 a 12 para adicionar imagens ao conjunto de dados de teste. Para a etapa 8, escolha Ações e escolha Adicionar imagens ao conjunto de dados de teste.

Etapa 5: adicione rótulos ao projeto

Nesta etapa, é adicionado um rótulo ao projeto para cada uma das classes identificadas na etapa Etapa 2: decida suas classes.

Para adicionar um novo rótulo (console)

1. Na página da galeria do conjunto de dados, escolha Iniciar rotulagem para entrar no modo de rotulagem.



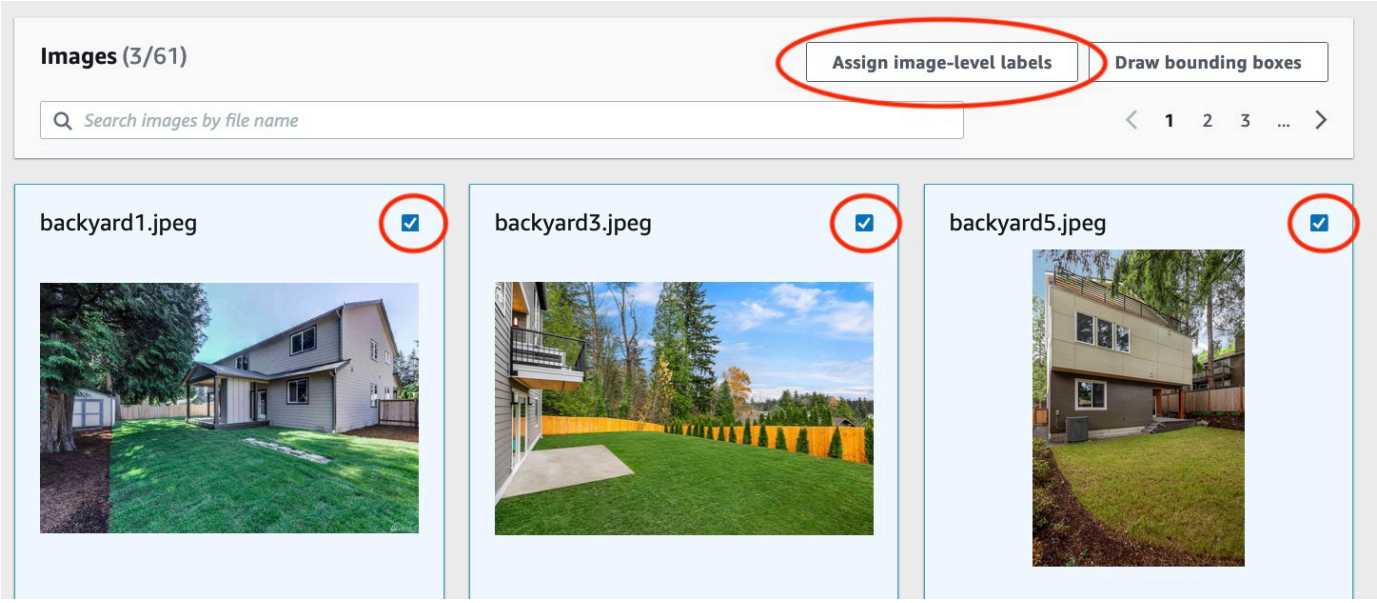
2. Na seção Rótulos da galeria do conjunto de dados, escolha Editar rótulos para abrir a caixa de diálogo Gerenciar rótulos.
3. Na caixa de edição, insira um novo nome de rótulo.
4. Selecione Adicionar novo rótulo.
5. Repita as etapas 3 e 4 até criar todos os rótulos necessários.
6. Escolha Salvar para salvar os rótulos que você adicionou.

Etapa 6: atribuia rótulos em nível de imagem aos conjuntos de dados de treinamento e teste

Nesta etapa, é atribuído um único nível de imagem a cada imagem em seus conjuntos de dados de treinamento e teste. O rótulo no nível da imagem é a classe que cada imagem representa.

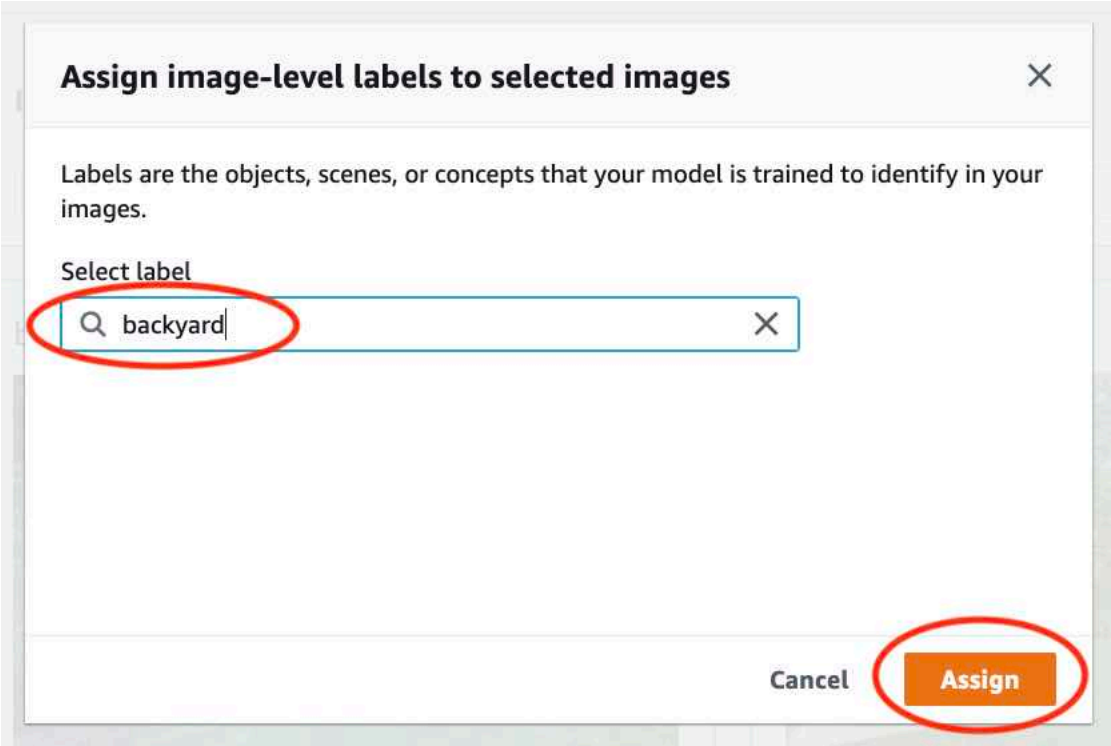
Para atribuir rótulos em nível de imagem em uma imagem (console)

1. Na página Conjuntos de dados, escolha a guia Treinamento.
2. Escolha Iniciar rotulagem para entrar no modo de rotulagem.
3. Selecione uma ou mais imagens às quais você deseja adicionar rótulos. Só é possível selecionar imagens em uma única página de cada vez. Para selecionar uma faixa contígua de imagens em uma página:
   a. Selecione a primeira imagem.
   b. Pressione e segure a tecla shift.
   c. Selecione a segunda imagem. As imagens entre a primeira e a segunda imagem também são selecionadas.
   d. Solte a tecla shift.
4. Escolha Atribuir rótulos em nível de imagem.



5. Na caixa de diálogo Atribuir rótulos em nível de imagem às imagens selecionadas, selecione um rótulo que você deseja atribuir à imagem ou imagens.

6. Escolha Atribuir para atribuir um rótulo à imagem.



7. Repita a rotulagem até que cada imagem seja anotada com os rótulos necessários.
8. Selecione a guia Testar.
9. Repita as etapas para atribuir rótulos de nível de imagem às imagens do conjunto de dados de teste.

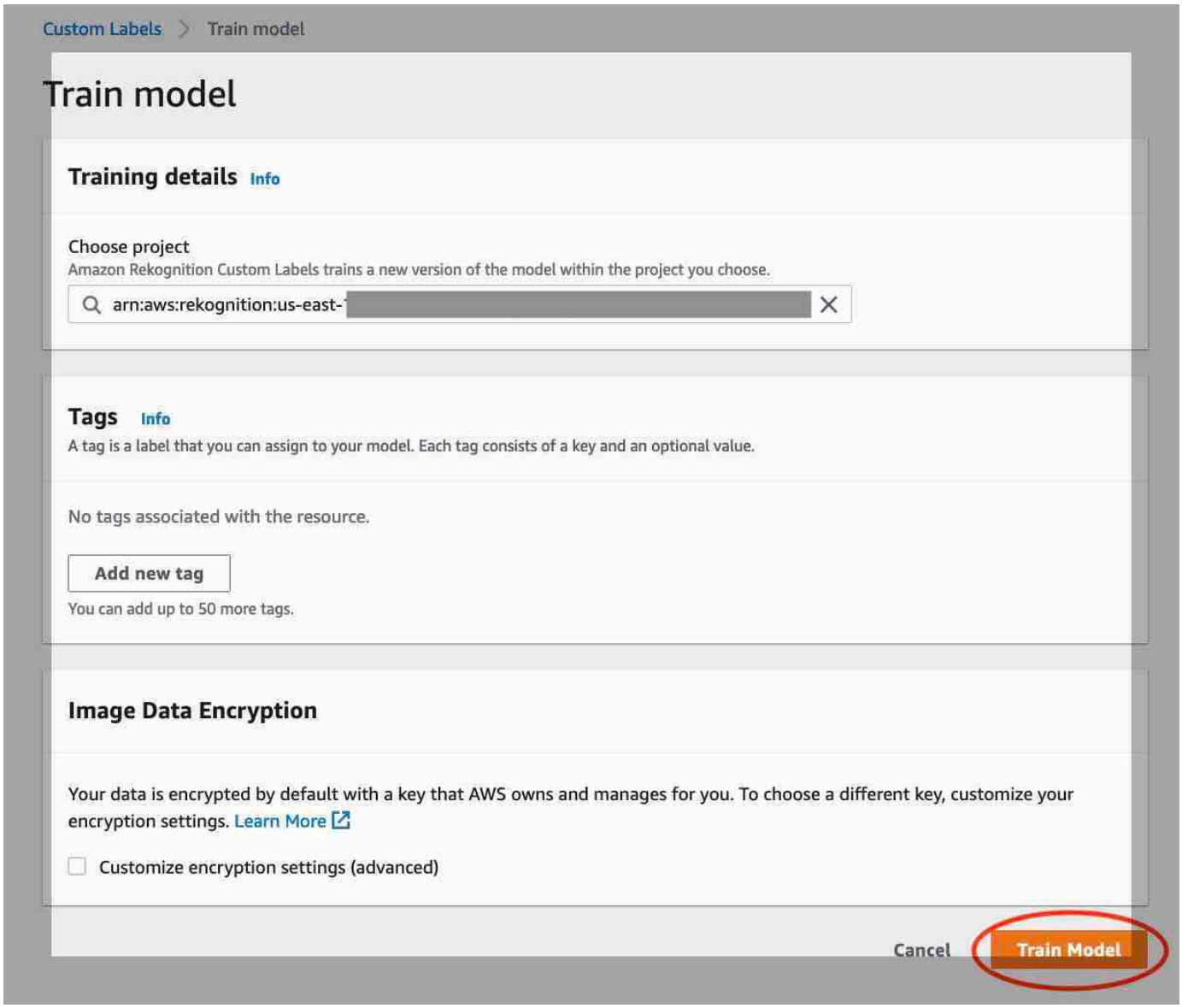
Etapa 7: treine seu modelo

Use as etapas a seguir para treinar seu modelo. Para obter mais informações, consulte Como treinar um modelo do Amazon Rekognition Custom Labels.

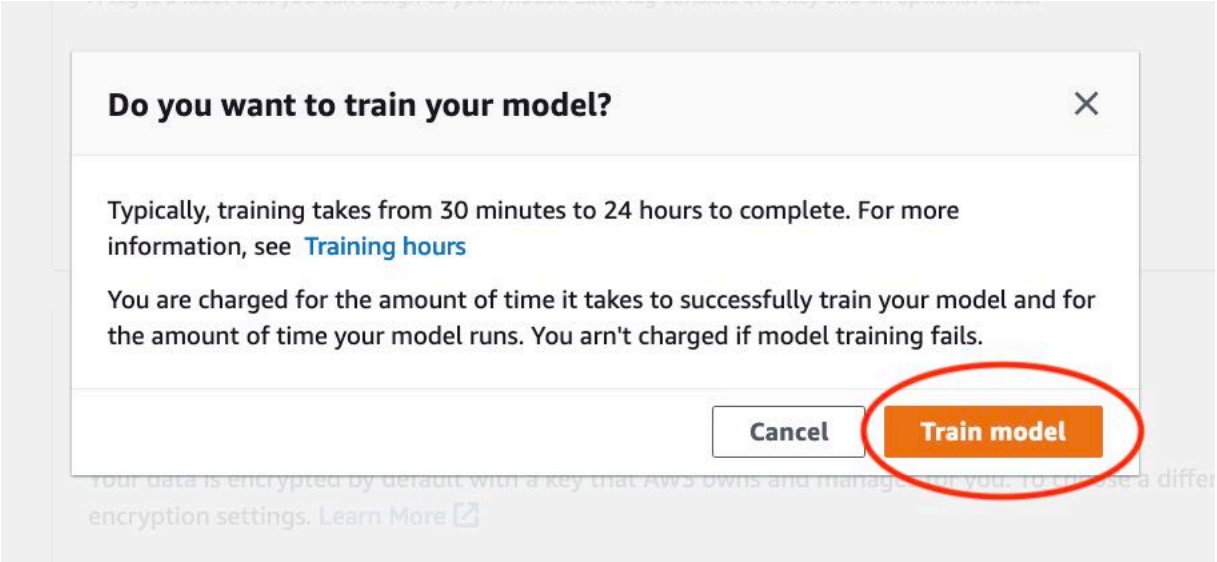
Para treinar seu modelo (console)

1. Na página Conjunto de dados, escolha Treinar modelo.

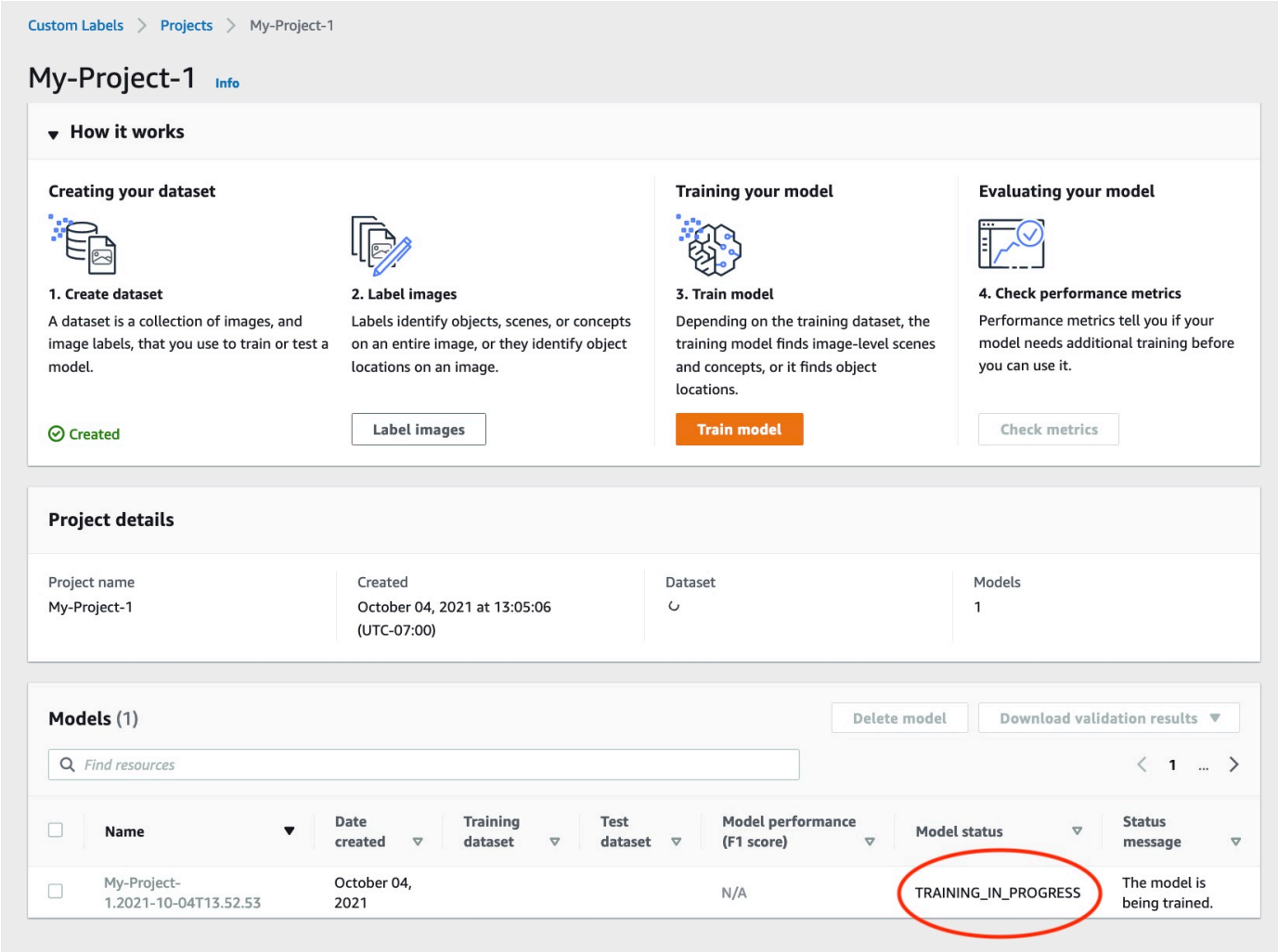
2. Na página Treinar modelo, escolha Treinar modelo. O nome do recurso da Amazon (ARN) do seu projeto está na caixa de edição Escolher projeto.



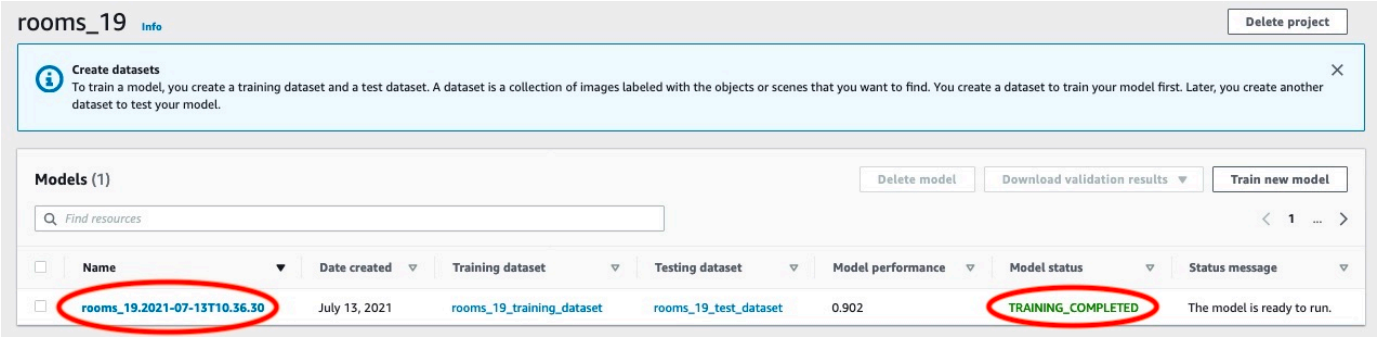
3. Na caixa de diálogo Você quer treinar seu modelo?, escolha Treinar modelo.



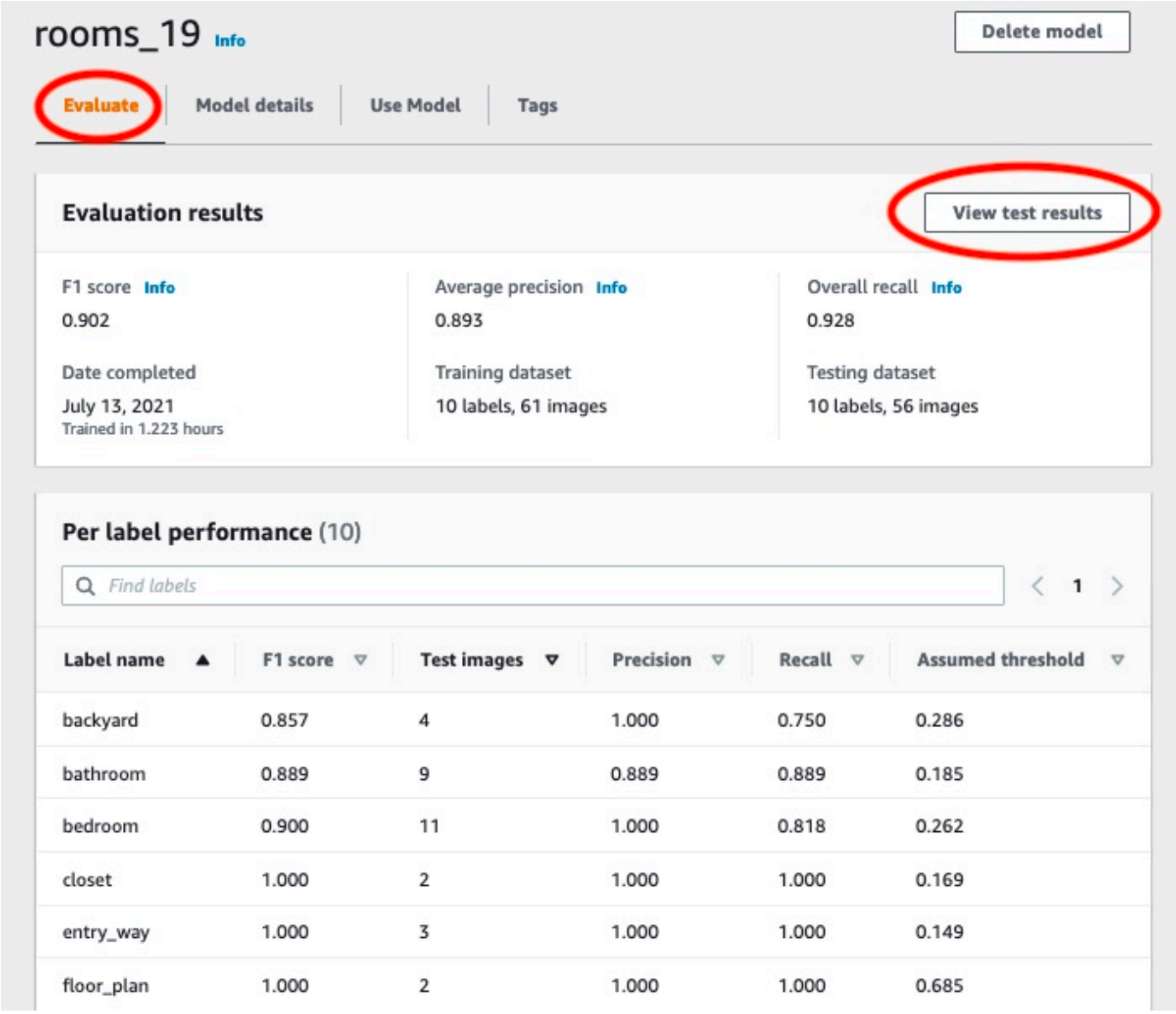
4. Na seção Modelos da página do projeto, pode ver que o treinamento está em andamento. É possível verificar o status atual visualizando a coluna `Model Status` da versão do modelo. O treinamento de um modelo leva algum tempo para ser concluído.



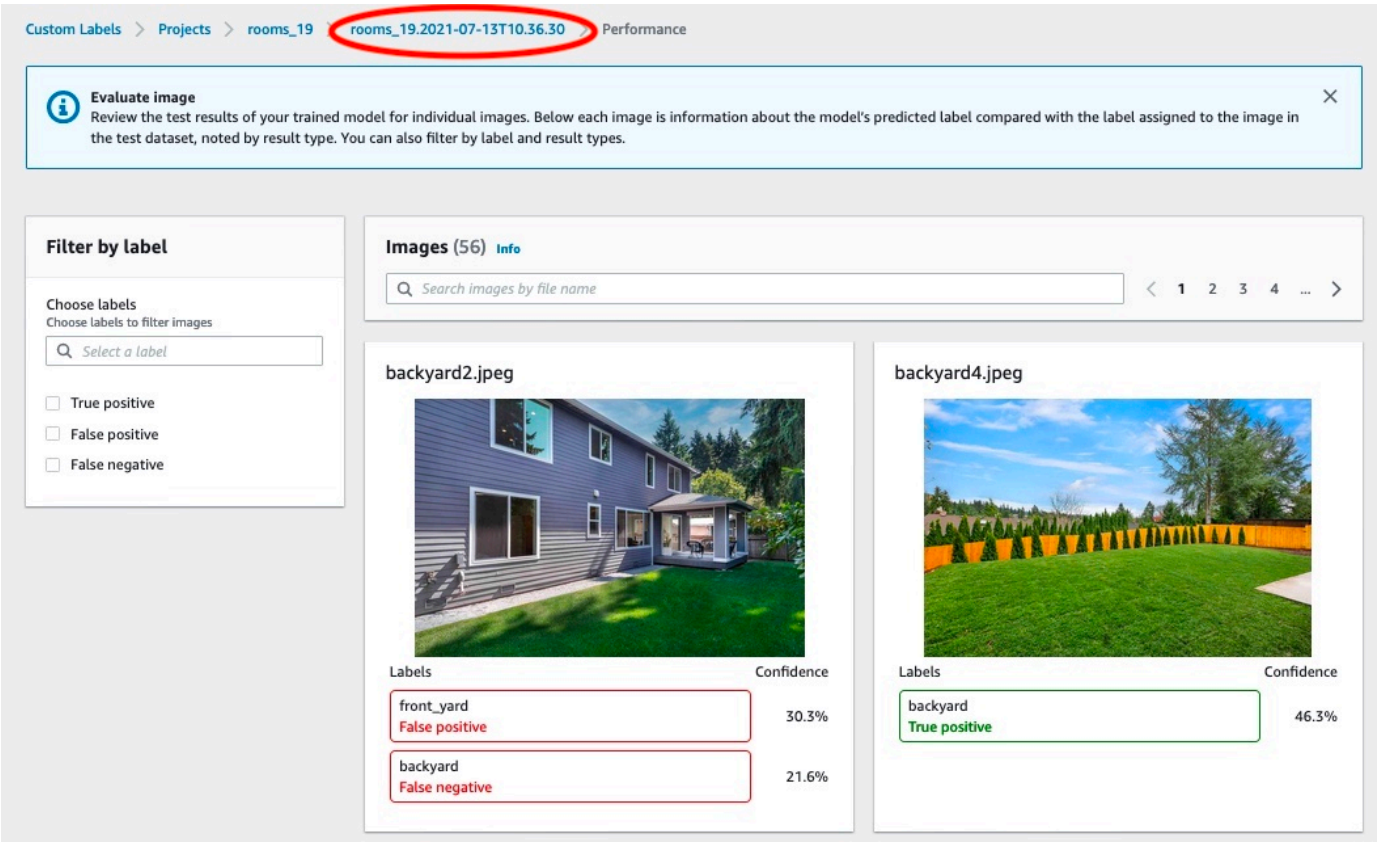
5. Após a conclusão do treinamento, escolha o nome do modelo. O treinamento é concluído quando o status do modelo for TRAINING_COMPLETED.



6. Escolha o botão Avaliar para ver os resultados da avaliação. Para obter informações sobre como avaliar um modelo, consulte Como melhorar um modelo treinado do Amazon Rekognition Custom Labels.
7. Escolha Exibir resultados do teste para ver os resultados de imagens de teste individuais. Para obter mais informações, consulte Métricas para avaliar seu modelo.



8. Depois de visualizar os resultados do teste, escolha o nome do modelo para retornar à página do modelo.



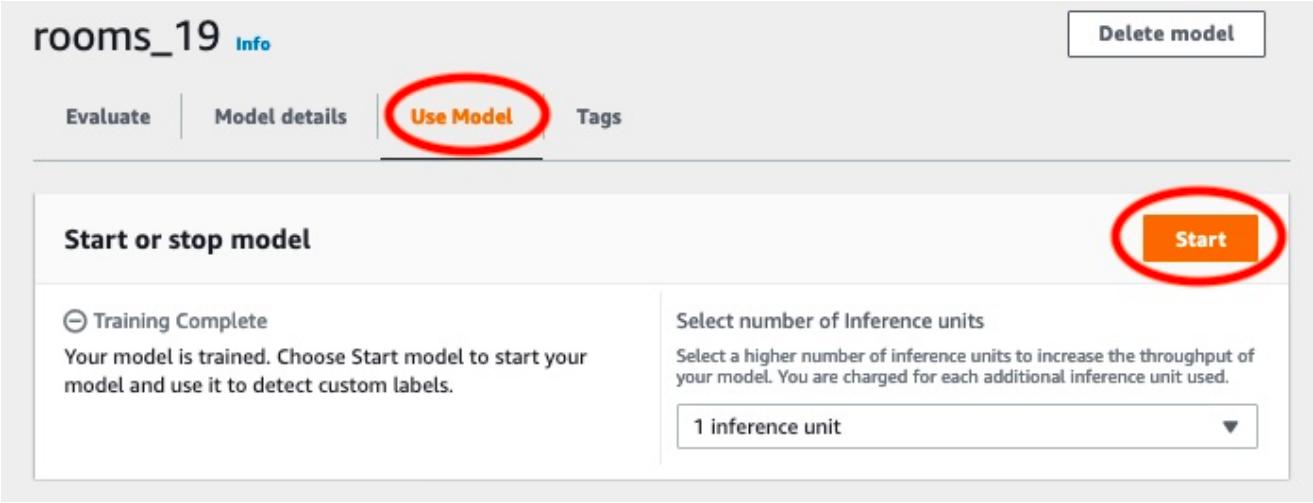
Etapa 8: inicie seu modelo

Nesta etapa, seu modelo é iniciado. Depois que o modelo começar, é possível usá-lo para analisar novas imagens.

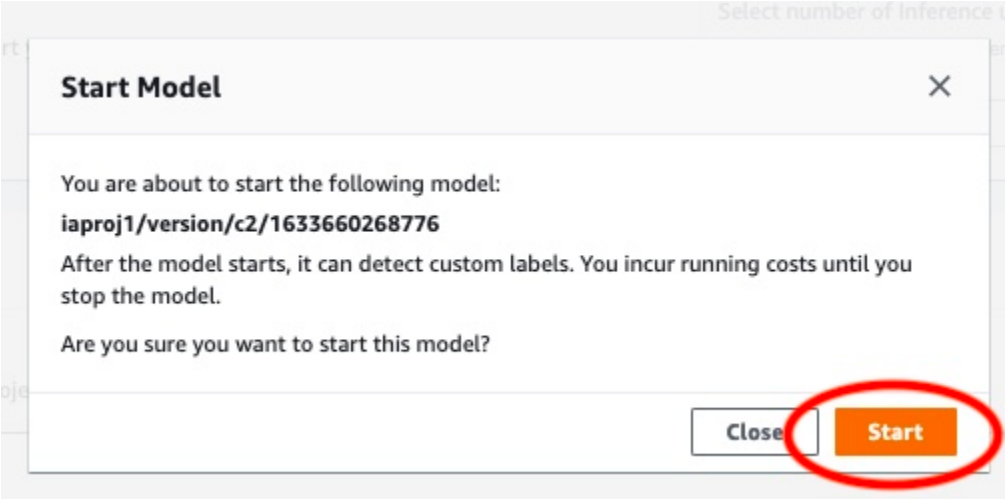
Há uma cobrança pela quantidade de tempo que o modelo é executado. Interrompa seu modelo se você não precisar analisar as imagens. Será possível reiniciar o modelo mais tarde. Para obter mais informações, consulte Como executar um modelo do Amazon Rekognition Custom Labels.

Para iniciar o seu modelo

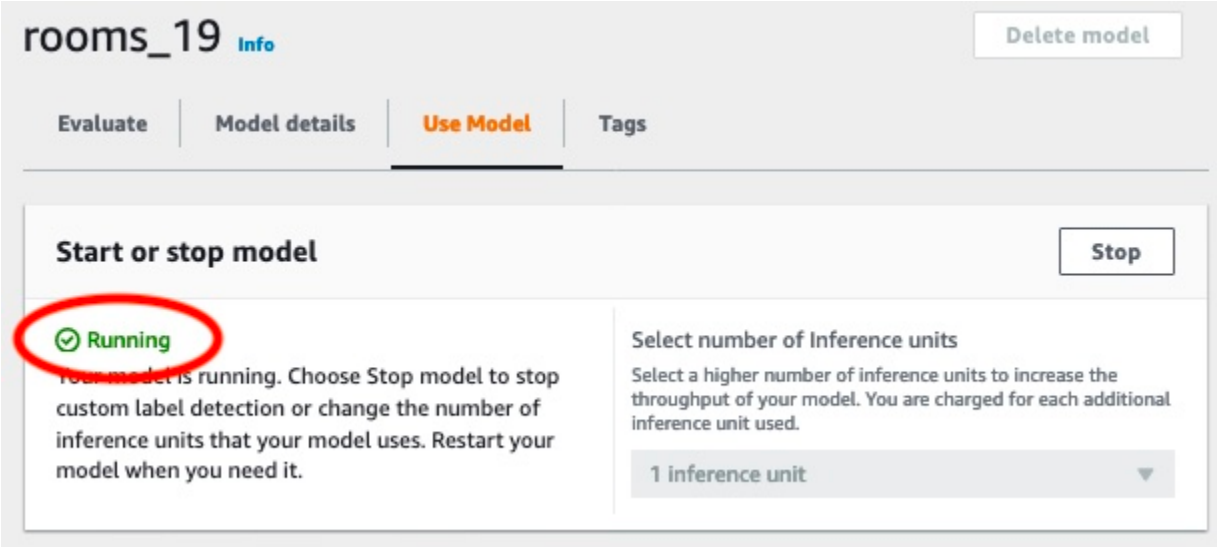
1. Escolha a guia Usar modelo na página do modelo.
2. Na seção Iniciar ou interromper o modelo, faça o seguinte:
    a. Escolha Iniciar.



b. Na caixa de diálogo Iniciar modelo, escolha Iniciar.



3. Espere até que o modelo esteja em execução. O modelo está em execução quando o status na seção Iniciar ou interromper modelo é Executando.



Etapa 9: analise uma imagem com seu modelo

Você analisa uma imagem chamando a DetectCustomLabelsAPI. Nesta etapa, você usa o comando `detect-custom-labels` AWS Command Line Interface (AWS CLI) para analisar uma imagem de exemplo. Você recebe o AWS CLI comando no console Amazon Rekognition Custom Labels. O console configura o AWS CLI comando para usar seu modelo. Só é preciso fornecer uma imagem que esteja armazenada em um bucket do Amazon S3.

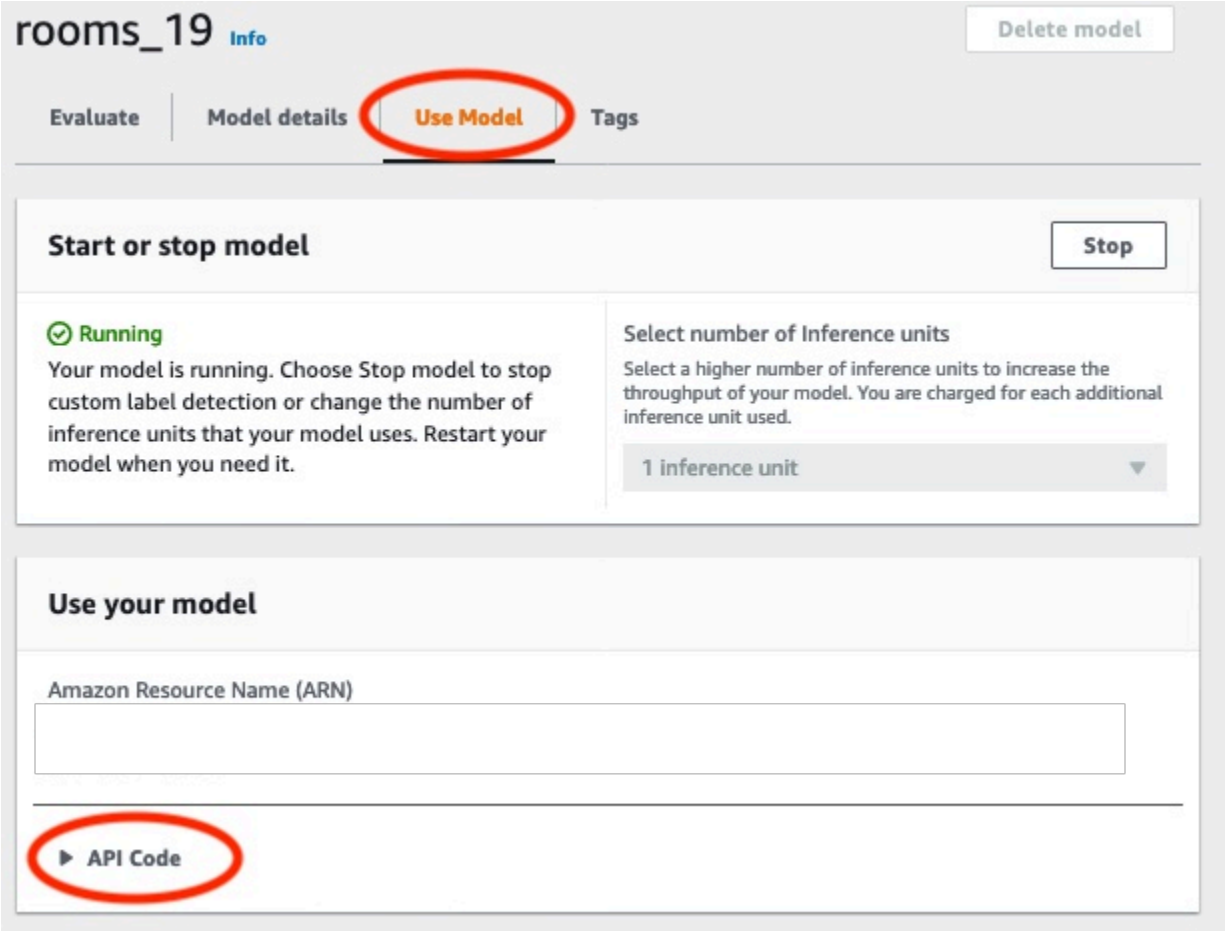
> **Note**
>
> O console também fornece um código de exemplo em Python.

A saída de `detect-custom-labels` inclui uma lista de rótulos encontrados na imagem, caixas delimitadoras (se o modelo encontrar a localização dos objetos) e a confiança que o modelo tem na precisão das previsões.

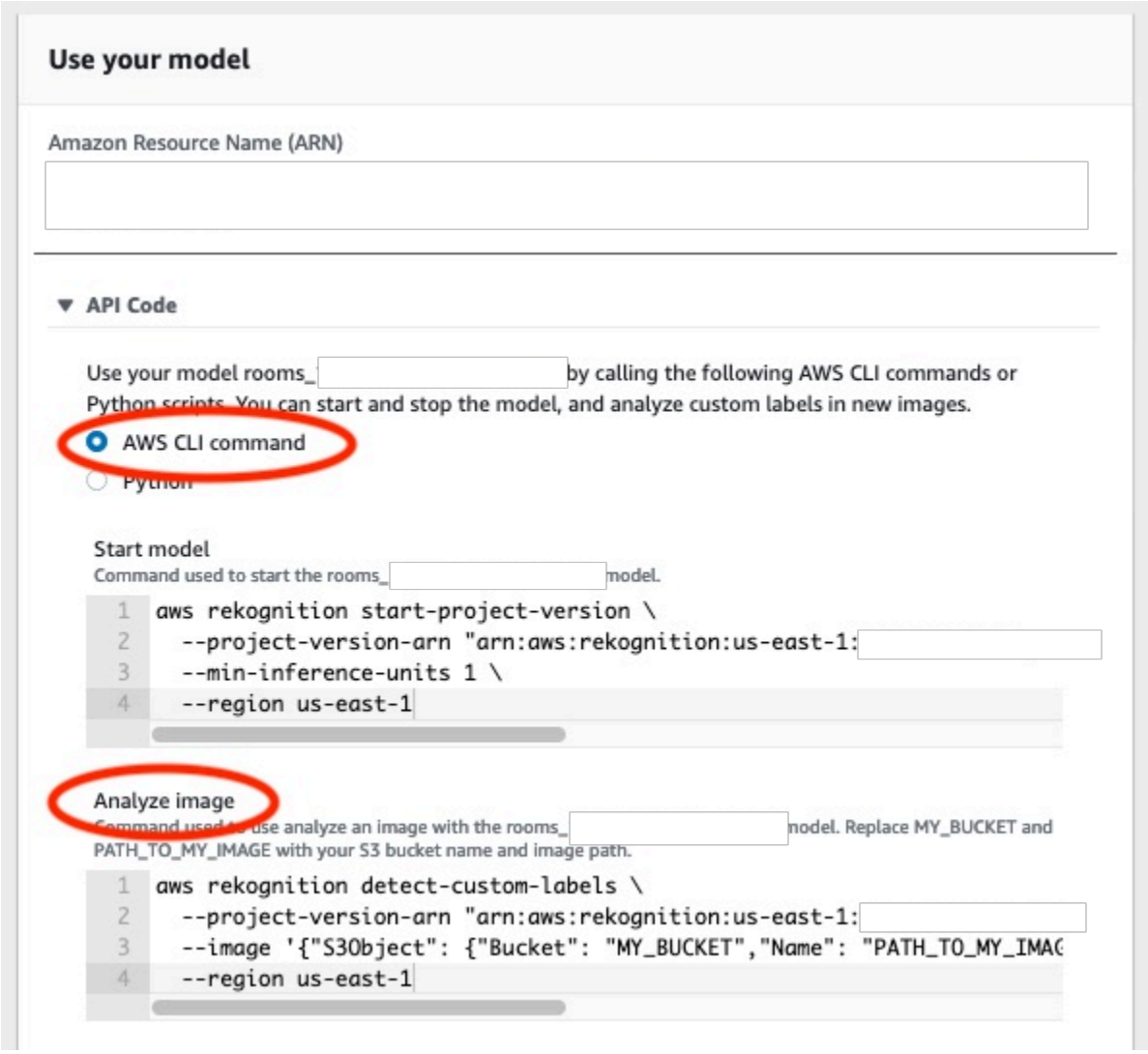
Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.

Para analisar uma imagem (console)

1. Se você ainda não o fez, configure AWS CLI o. Para instruções, consulte the section called "Etapa 4: configurar o AWS CLI e AWS SDKs".
2. Escolha a guia Usar modelo e escolha o código da API.



3. Escolha o Comando da AWS CLI.
4. Na seção Analisar imagem, copie o AWS CLI comando que chama`detect-custom-labels`.



5. Faça upload de uma imagem para um bucket do Amazon S3. Para obter mais informações, consulte Fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service. Se estiver usando imagens do projeto Cômodos, use uma das imagens que você moveu para uma pasta separada em Etapa 1: colete suas imagens.

6. No prompt de comando, digite o AWS CLI comando que você copiou na etapa anterior. Ele se parece com o exemplo a seguir.

   O valor de `--project-version-arn` deve ser o nome do recurso da Amazon (ARN) do seu modelo. O valor de `--region` deve ser a região da AWS na qual você criou o modelo.

   Altere `MY_BUCKET` e `PATH_TO_MY_IMAGE` use o bucket e a imagem do Amazon S3 que você usou na etapa anterior.

Se você estiver usando o perfil custom-labels-access para obter credenciais, adicione o parâmetro `--profile custom-labels-access`.

```
aws rekognition detect-custom-labels \
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET","Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

A saída JSON do comando AWS CLI deverá ser semelhante à seguinte. O `Name` é o nome do rótulo em nível de imagem que o modelo encontrou. A `Confidence` (de 0 a 100) é a confiança do modelo na precisão da previsão.

```
{
    "CustomLabels": [
        {
            "Name": "living_space",
            "Confidence": 83.41299819946289
        }
    ]
}
```

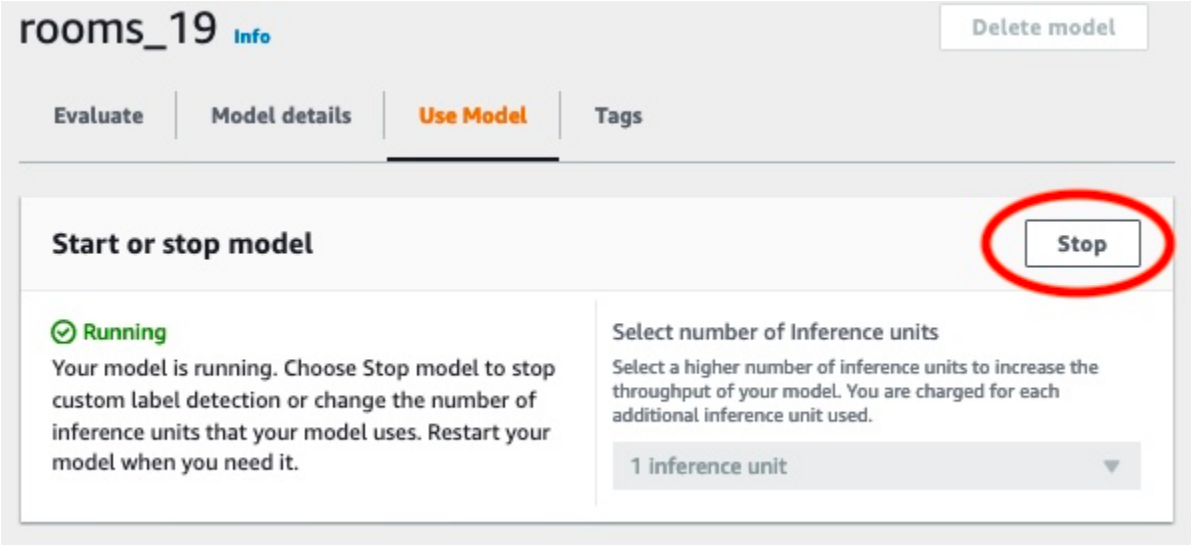
7. Continue usando o modelo para analisar outras imagens. Interrompa o modelo se não estiver mais usando.

Etapa 10: interrompa seu modelo

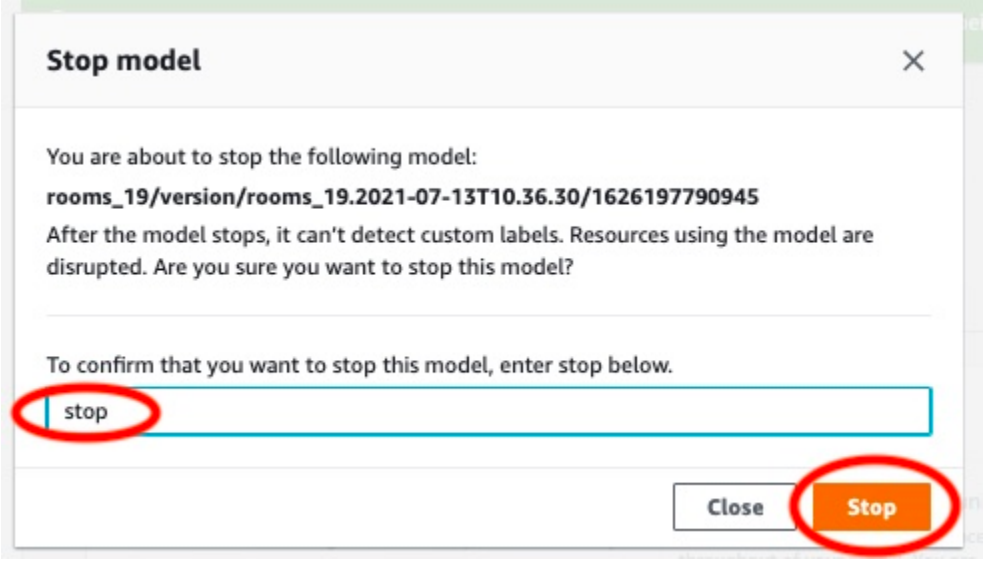
Nesta etapa, seu modelo para de ser executado. Há uma cobrança pela quantidade de tempo que o modelo está em execução. Se terminou de usar o modelo, deve interrompê-lo.

Para interromper seu modelo

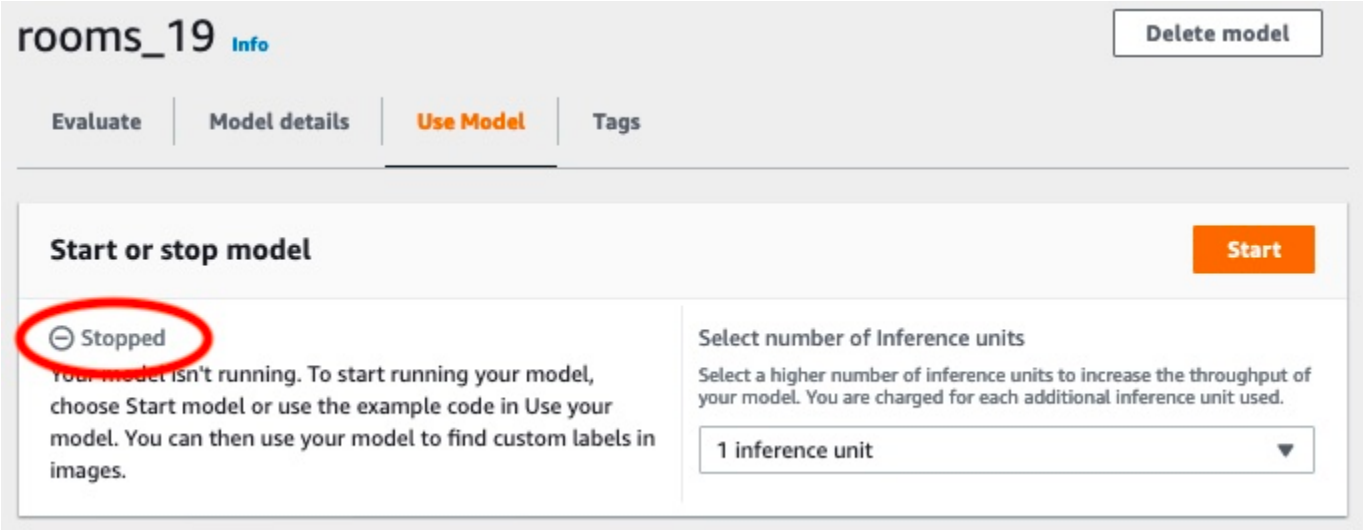
1. Na seção Iniciar ou interromper modelo, escolha Interromper.



2. Na caixa de diálogo Interromper modelo, insira interromper para confirmar que deseja interromper o modelo.



3. Escolha Parar para interromper seu modelo. O modelo foi interrompido quando o status na seção Iniciar ou interromper modelo é Interrompido.


rooms_19
Info
Delete model
Evaluate
Model details
Use Model
Tags
Start or stop model
Start
Stopped
isn't running. To start running your model, choose Start model or use the example code in Use your model. You can then use your model to find custom labels in images.
Select number of Inference units
Select a higher number of inference units to increase the throughput of your model. You are charged for each additional inference unit used.
1 inference unit

Como criar um modelo do Amazon Rekognition Custom Labels

Um modelo é o software que você treina para encontrar conceitos, cenas e objetos exclusivos da sua empresa. Você pode criar um modelo com o console Amazon Rekognition Custom Labels ou com o SDK. AWS Antes de criar um modelo do Amazon Rekognition Custom Labels, recomendamos que você leia Noções básicas do Amazon Rekognition Custom Labels.

Esta seção fornece informações do console e do SDK sobre como criar um projeto, criar conjuntos de dados de treinamento e teste para diferentes tipos de modelo e treinar um modelo. As seções posteriores mostram como melhorar e usar seu modelo. Para um tutorial que mostra como criar e usar um tipo específico de modelo com o console, consulte Classificar imagens.

Tópicos

- Como criar um projeto
- Como criar conjuntos de dados de treinamento e teste
- Como treinar um modelo do Amazon Rekognition Custom Labels
- Como depurar um treinamento de modelo em falha

Como criar um projeto

Um projeto gerencia as versões do modelo, o conjunto de dados de treinamento e o conjunto de dados de teste de um modelo. É possível criar um projeto com o console do Amazon Rekognition Custom Labels ou com a API. Para outras tarefas do projeto, como excluir um projeto, consulte Como gerenciar um projeto do Amazon Rekognition Custom Labels.

Você pode usar etiquetas para categorizar e gerenciar os recursos do Amazon Rekognition Custom Labels, incluindo seus projetos.

A CreateProjectoperação permite que você opcionalmente especifique tags ao criar um novo projeto, fornecendo as tags como pares de valores-chave que você pode usar para categorizar e gerenciar seus recursos.

Como criar um projeto do Amazon Rekognition Custom Labels (console)

É possível usar o console do Amazon Rekognition Custom Labels para criar um projeto. Na primeira vez que você usa o console em uma nova AWS região, o Amazon Rekognition Custom Labels solicita a criação de um bucket do Amazon S3 (bucket do console) em sua conta. AWS Esse bucket será usado para armazenar arquivos do projeto. Não é possível usar o console do Amazon Rekognition Custom Labels a menos que o bucket de console seja criado.

É possível usar o console do Amazon Rekognition Custom Labels para criar um projeto.

Para criar um projeto (console)

1. Faça login no AWS Management Console e abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Usar rótulos personalizados. A página inicial do Amazon Rekognition Custom Labels é exibida.
3. Na página inicial do Amazon Rekognition Custom Labels, escolha Conceitos básicos.
4. No painel esquerdo, selecione Projetos.
5. Escolha Criar projeto.
6. Em Nome do projeto, digite um nome para o seu projeto.
7. Escolha Criar projeto para criar seu projeto.
8. Siga as etapas em Como criar conjuntos de dados de treinamento e teste para criar os conjuntos de dados de treinamento e teste para seu projeto.

Como criar um projeto do Amazon Rekognition Custom Labels (SDK)

Você cria um projeto Amazon Rekognition Custom Labels ligando para. CreateProject A resposta é um nome do recurso da Amazon (ARN) que identifica o projeto. Depois de criar um projeto, você cria conjuntos de dados para treinar e testar um modelo. Para obter mais informações, consulte Como criar conjuntos de dados de treinamento e teste com imagens.

Para criar um projeto (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para criar um projeto.

AWS CLI

O exemplo a seguir cria um projeto e exibe seu ARN.

Altere o valor de `project-name` para o nome do projeto que você deseja criar.

```
aws rekognition create-project --project-name my_project \
 --profile custom-labels-access --"CUSTOM_LABELS" --
tags'{"key1":"value1","key2":"value2"}'
```

Python

O exemplo a seguir cria um projeto e exibe seu ARN. Forneça os seguintes argumentos de linha de comando:

- `project_name`: o nome do projeto que você deseja criar.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_project(rek_client, project_name):
    """
    Creates an Amazon Rekognition Custom Labels project
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: A name for the new prooject.
    """

    try:
        #Create the project.
        logger.info("Creating project: %s",project_name)

        response=rek_client.create_project(ProjectName=project_name)
```

```
        logger.info("project ARN: %s",response['ProjectArn'])

        return response['ProjectArn']


    except ClientError as err:
        logger.exception("Couldn't create project - %s: %s", project_name,
 err.response['Error']['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_name", help="A name for the new project."
    )


def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating project: {args.project_name}")

        # Create the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        project_arn=create_project(rekognition_client,
            args.project_name)

        print(f"Finished creating project: {args.project_name}")
        print(f"ARN: {project_arn}")
```

```
    except ClientError as err:
        logger.exception("Problem creating project: %s", err)
        print(f"Problem creating project: {err}")


if __name__ == "__main__":
    main()
```

Java V2

O exemplo a seguir cria um projeto e exibe seu ARN.

Forneça o seguinte argumento de linha de comando:

- `project_name`: o nome do projeto que você deseja criar.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateProjectRequest;
import software.amazon.awssdk.services.rekognition.model.CreateProjectResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateProject {

    public static final Logger logger =
 Logger.getLogger(CreateProject.class.getName());

    public static String createMyProject(RekognitionClient rekClient, String
 projectName) {

        try {
```

```
            logger.log(Level.INFO, "Creating project: {0}", projectName);
            CreateProjectRequest createProjectRequest =
 CreateProjectRequest.builder().projectName(projectName).build();

            CreateProjectResponse response =
 rekClient.createProject(createProjectRequest);

            logger.log(Level.INFO, "Project ARN: {0} ", response.projectArn());

            return response.projectArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create project: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<project_name> <bucket> <image>
\n\n" + "Where:\n"
                + "   project_name - A name for the new project\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectName = args[0];
        String projectArn = null;
        ;

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();
```

```
            // Create the project
            projectArn = createMyProject(rekClient, projectName);

            System.out.println(String.format("Created project: %s %nProject ARN:
 %s", projectName, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. Observe o nome do ARN do projeto exibido na resposta. Será necessário para criar um modelo.
4. Siga as etapas em Crie conjuntos de dados de treinamento e teste (SDK) para criar os conjuntos de dados de treinamento e teste para seu projeto.

CreateProject solicitação de operação

Veja a seguir o formato da solicitação de CreateProject operação:

```
{
 "AutoUpdate": "string",
 "Feature": "string",
 "ProjectName": "string",
 "Tags": {
 "string": "string"
 }
}
```

Como criar conjuntos de dados de treinamento e teste

Um conjunto de dados é um conjunto de imagens e rótulos que descrevem essas imagens. Seu projeto precisa de um conjunto de dados de treinamento e um conjunto de dados de teste. O Amazon Rekognition Custom Labels usa o conjunto de dados de treinamento para treinar seu modelo. Após o treinamento, o Amazon Rekognition Custom Labels usa o conjunto de dados de teste para verificar se o modelo treinado prevê os rótulos corretos.

Você pode criar conjuntos de dados com o console Amazon Rekognition Custom Labels ou com o SDK. AWS Antes de criar um conjunto de dados, recomendamos a leitura Noções básicas do Amazon Rekognition Custom Labels. Para outras tarefas do conjunto de dados, consulte Como gerenciar conjuntos de dados.

As etapas para criar conjuntos de dados de treinamento e testes para um projeto são:

Para criar os conjuntos de dados de treinamento e teste para seu projeto

1. Determine como você precisa rotular seus conjuntos de dados de treinamento e teste. Para obter mais informações, Como definir os conjuntos de dados.
2. Colete as imagens para seus conjuntos de dados de treinamento e teste. Para obter mais informações, consulte the section called “Como preparar imagens”.
3. Crie os conjuntos de dados de treinamento e teste. Para obter mais informações, consulte Como criar conjuntos de dados de treinamento e teste com imagens. Se você estiver usando o AWS SDK, consulteCrie conjuntos de dados de treinamento e teste (SDK) .
4. Se necessário, adicione rótulos no nível da imagem ou caixas delimitadoras às imagens do seu conjunto de dados. Para obter mais informações, consulte Rotulagem de imagens.

Depois de criar os conjuntos de dados, é possível treinar o modelo.

Tópicos

- Como definir os conjuntos de dados
- Como preparar imagens
- Como criar conjuntos de dados de treinamento e teste com imagens
- Rotulagem de imagens
- Como depurar conjuntos de dados

Como definir os conjuntos de dados

A forma como os conjuntos de dados de treinamento e teste são treinados no projeto determina o tipo de modelo criado. Com o Amazon Rekognition Custom Labels, é possível criar modelos que façam o seguinte:

- Encontre objetos, cenas e conceitos
- Encontre localizações de objetos
- Encontre localizações de marcas

Encontre objetos, cenas e conceitos

O modelo classifica os objetos, cenas e conceitos associados a uma imagem inteira.

É possível criar dois tipos de modelo de classificação, classificação de imagem e classificação de vários rótulos. Para os dois tipos de modelo de classificação, o modelo encontra um ou mais rótulos correspondentes no conjunto completo de rótulos usados para treinamento. Os conjuntos de dados de treinamento e teste exigem pelo menos dois rótulos.

Classificação de imagens

O modelo classifica as imagens como pertencentes a um conjunto de rótulos predefinidos. Por exemplo, convém um modelo que determine se uma imagem contém uma sala de estar. A imagem a seguir pode ter um rótulo de nível de imagem sala_de_estar.

Para este tipo de modelo, adicione um rótulo único em nível de imagem para cada uma das imagens do conjunto de dados de treinamento e teste. Para obter um objeto de exemplo, consulte Classificação de imagens.

Classificação com vários rótulos

O modelo classifica as imagens em várias categorias, como o tipo de flor e se ela tem folhas ou não. Por exemplo, a imagem a seguir pode ter rótulos de nível de imagem mediterranean_spurge e no_leaves.

Para esse tipo de modelo, atribua rótulos em nível de imagem para cada categoria às imagens do conjunto de dados de treinamento e teste. Para obter um objeto de exemplo, consulte Classificação de imagens com vários rótulos.

Como atribuir rótulos em nível de imagem

Se suas imagens estiverem armazenadas em um bucket do Amazon S3, será possível usar nomes de pastas para adicionar automaticamente rótulos em nível de imagem. Para obter mais informações, consulte Importar imagens de um bucket do Amazon S3. Também é possível adicionar rótulos em nível de imagem às imagens depois de criar um conjunto de dados. Para obter mais informações, consulte the section called “Como atribuir rótulos em nível de imagem em uma imagem”. É possível adicionar novos rótulos conforme necessário. Para obter mais informações, consulte Como gerenciar rótulos.

Encontre localizações de objetos

Para criar um modelo que preveja a localização de objetos em suas imagens, você define caixas delimitadoras e rótulos de localização de objetos para as imagens em seus conjuntos de dados de treinamento e teste. Uma caixa delimitadora é uma caixa que envolve firmemente um objeto. Por exemplo, a imagem a seguir mostra as caixas delimitadoras ao redor de um Amazon Echo e um Amazon Echo Dot. Cada caixa delimitadora tem um rótulo atribuído (Amazon Echo ou Amazon Echo Dot).

Para encontrar a localização dos objetos, seus conjuntos de dados precisam de pelo menos um rótulo. Durante o treinamento do modelo, um rótulo adicional é criado automaticamente, representando a área fora das caixas delimitadoras em uma imagem.

Como atribuir caixas delimitadoras

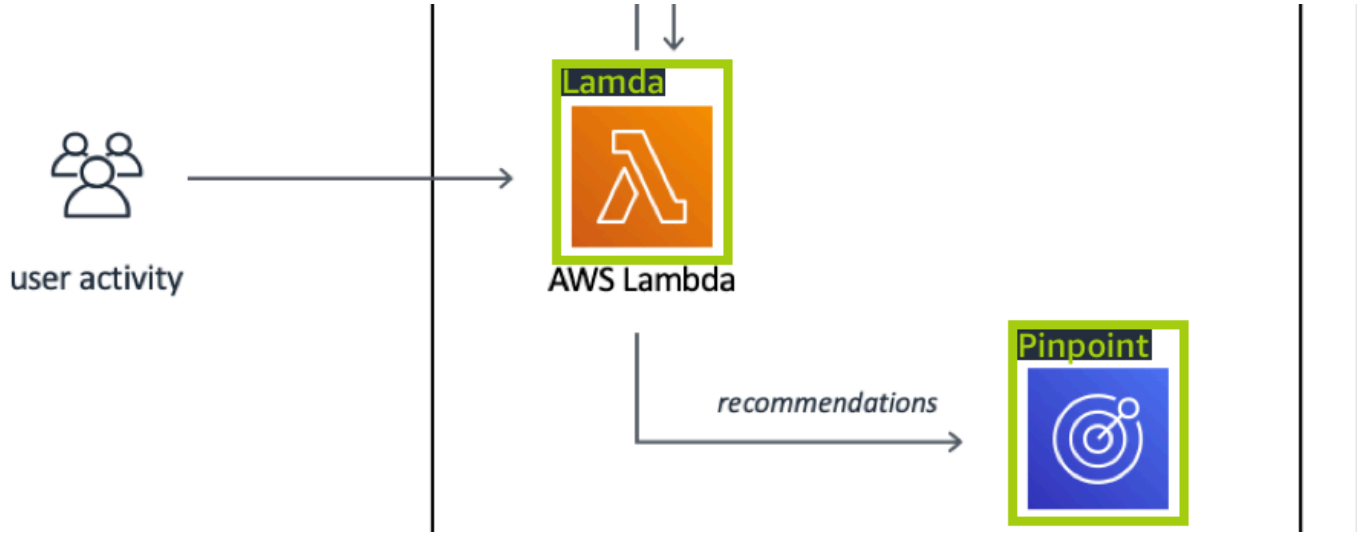
Ao criar seu conjunto de dados, é possível incluir informações da caixa delimitadora para suas imagens. Por exemplo, você pode importar um arquivo de manifesto no formato SageMaker AI Ground Truth que contém caixas delimitadoras. Também é possível adicionar caixas delimitadoras depois de criar um conjunto de dados. Para obter mais informações, consulte Como rotular objetos com caixas delimitadoras. É possível adicionar novos rótulos conforme necessário. Para obter mais informações, consulte Como gerenciar rótulos.

Encontre localizações de marcas

Se quiser encontrar a localização de marcas, como logotipos e personagens animados, é possível usar dois tipos diferentes de imagens para as imagens do seu conjunto de dados de treinamento.

- As imagens que são somente do logotipo. Cada imagem precisa de um único rótulo em nível de imagem que represente o nome do logotipo. Por exemplo, o rótulo em nível de imagem para a imagem a seguir pode ser Lambda.

- As imagens que contêm o logotipo em locais naturais, como um jogo de futebol ou um diagrama arquitetônico. Cada imagem de treinamento precisa de caixas delimitadoras que envolvam cada instância do logotipo. Por exemplo, a imagem a seguir mostra um diagrama arquitetônico com caixas delimitadoras rotuladas ao redor dos logotipos do Lambda AWS e do Amazon Pinpoint.



É recomendável não misturar rótulos em nível de imagem e caixas delimitadoras em suas imagens de treinamento.

As imagens de teste devem ter caixas delimitadoras ao redor das instâncias da marca que você deseja encontrar. É possível dividir o conjunto de dados de treinamento para criar o conjunto de dados de teste, somente se as imagens de treinamento incluírem caixas delimitadoras rotuladas. Se as imagens de treinamento tiverem apenas rótulos em nível de imagem, você deverá criar um conjunto de dados de teste que inclua imagens com caixas delimitadoras rotuladas. Se treinar um modelo para encontrar localizações de marcas, faça Como rotular objetos com caixas delimitadoras e Como atribuir rótulos em nível de imagem em uma imagem de acordo com a forma como você rotula suas imagens.

O projeto de exemplo Detecção de marca mostra como o Amazon Rekognition Custom Labels usa caixas delimitadoras rotuladas para treinar um modelo que encontra a localização dos objetos.

Requisitos de rótulo para tipos de modelo

Use a tabela a seguir para determinar como rotular suas imagens.

É possível combinar rótulos no nível da imagem e imagens rotuladas na caixa delimitadora em um único conjunto de dados. Neste caso, o Amazon Rekognition Custom Labels escolhe se deseja criar um modelo em nível de imagem ou um modelo de localização de objetos.

Exemplo	Imagens de treinamento	Imagens de teste
Classificação de imagens	1 Rótulo em nível de imagem por imagem	1 Rótulo em nível de imagem por imagem
Classificação com vários rótulos	Vários rótulos em nível de imagem por imagem	Vários rótulos em nível de imagem por imagem
Encontre localizações de marcas	rótulos de nível de imagem (também é possível usar caixas delimitadoras rotuladas)	Caixas delimitadoras rotuladas
Encontre localizações de objetos	Caixas delimitadoras rotuladas	Caixas delimitadoras rotuladas

Como preparar imagens

As imagens em seu conjunto de dados de treinamento e teste contêm os objetos, cenas ou conceitos que você deseja que seu modelo encontre.

O conteúdo das imagens deve estar em uma variedade de planos de fundo e iluminação que representem as imagens que você deseja que o modelo treinado identifique.

Esta seção fornece informações sobre as imagens em seu conjunto de dados de treinamento e teste.

Formato da imagem

É possível treinar modelos do Amazon Rekognition Custom Labels com imagens nos formatos PNG e JPEG. Da mesma forma, para detectar o uso de rótulos personalizados usando `DetectCustomLabels`, você precisa de imagens nos formatos PNG e JPEG.

Recomendações de imagem de entrada

O Amazon Rekognition Custom Labels exige imagens para treinar e testar seu modelo. Para preparar suas imagens, considere o seguinte:

- Escolha um domínio específico para o modelo que você deseja criar. Por exemplo, é possível escolher um modelo para vistas panorâmicas e outro modelo para objetos como peças de máquinas. O Amazon Rekognition Custom Labels funcionam melhor se suas imagens estiverem no domínio escolhido.
- Use pelo menos 10 imagens para treinar seu modelo.
- As imagens devem estar no formato PNG ou JPEG.
- Use imagens que mostrem o objeto em uma variedade de iluminações, planos de fundo e resoluções.
- As imagens de treinamento e teste devem ser semelhantes às imagens com as quais você deseja usar o modelo.
- Decida quais rótulos atribuir às imagens.
- Certifique-se de que as imagens sejam suficientemente grandes em termos de resolução. Para obter mais informações, consulte Diretrizes e cotas no Amazon Rekognition Custom Labels.
- Certifique-se de que as oclusões não obscureçam os objetos que você deseja detectar.
- Use imagens com contraste suficiente com o plano de fundo.
- Use imagens claras e nítidas. Evite usar imagens que possam ficar desfocadas devido ao movimento do objeto e da câmera, tanto quanto possível.

- Use uma imagem em que o objeto ocupe uma grande proporção da imagem.
- As imagens em seu conjunto de dados de teste não devem ser imagens que estão no conjunto de dados de treinamento. Eles devem incluir os objetos, cenas e conceitos que o modelo foi treinado para analisar.

Tamanho do conjunto de imagens

O Amazon Rekognition Custom Labels usa um conjunto de imagens para treinar um modelo. No mínimo, você deve usar pelo menos dez imagens para treinar. O Amazon Rekognition Custom Labels armazenas imagens de treinamento e teste em conjuntos de dados. Para obter mais informações, consulte Como criar conjuntos de dados de treinamento e teste com imagens.

Como criar conjuntos de dados de treinamento e teste com imagens

É possível iniciar com um projeto que tenha um único conjunto de dados ou um projeto que tenha conjuntos de dados de treinamento e teste separados. Se você começar com um único conjunto de dados, o Amazon Rekognition Custom Labels divide seu conjunto de dados durante o treinamento para criar um conjunto de dados de treinamento (80%) e um conjunto de dados de teste (20%) para seu projeto. Comece com um único conjunto de dados se quiser que o Amazon Rekognition Custom Labels decida onde as imagens serão usadas para treinamento e teste. Para ter controle total sobre o treinamento, teste e ajuste de desempenho, recomendamos que você inicie seu projeto com os conjuntos de dados de treinamento e teste separados.

É possível criar conjuntos de dados de treinamento e teste para um projeto importando imagens de um dos seguintes locais:

- Importar imagens de um bucket do Amazon S3
- Importar imagens de um computador local
- Usar um arquivo de manifesto para importar imagens
- Copiar conteúdo de um conjunto de dados existente

Se iniciar seu projeto com conjuntos de dados de treinamento e teste separados, poderá usar locais de origem diferentes para cada conjunto de dados.

Dependendo de onde você importa suas imagens, elas podem não estar rotuladas. Por exemplo, imagens importadas de um computador local não estão rotuladas. As imagens importadas de um arquivo de manifesto do Amazon SageMaker AI Ground Truth são rotuladas. É possível usar o

console do Amazon Rekognition Custom Labels para adicionar, alterar e atribuir rótulos. Para obter mais informações, consulte Rotulagem de imagens.

Se as imagens estiverem sendo carregadas com erros, se faltarem imagens ou se faltarem rótulos nas imagens, leia Como depurar um treinamento de modelo em falha.

Para obter mais informações sobre conjuntos de dados, consulte Como gerenciar conjuntos de dados.

Crie conjuntos de dados de treinamento e teste (SDK)

Você pode usar o AWS SDK para criar conjuntos de dados de treinamento e teste.

A operação `CreateDataset` permite que você opte por especificar etiquetas ao criar um conjunto de dados, com o objetivo de categorizar e gerenciar seus recursos.

Conjunto de dados de trenaimento

Você pode usar o AWS SDK para criar um conjunto de dados de treinamento das seguintes formas.

- Use CreateDatasetcom um arquivo de manifesto no formato Amazon Sagemaker fornecido por você. Para obter mais informações, consulte the section called “Criar um arquivo de manifesto”. Para obter um código de exemplo, consulte Criação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth.
- Use `CreateDataset` para copiar um conjunto de dados existente do Amazon Rekognition Custom Labels. Para obter um código de exemplo, consulte Como criar um conjunto de dados usando um conjunto de dados existente (SDK).
- Crie um conjunto de dados vazio com `CreateDataset` e adicione entradas do conjunto de dados posteriormente com UpdateDatasetEntries. Para criar um conjunto de dados vazio, consulte Como adicionar um conjunto de dados a um projeto. Para adicionar imagens a um conjunto de dados, consulte Como adicionar mais imagens (SDK). É necessário adicionar as entradas do conjunto de dados antes de treinar um modelo.

Conjunto de dados de teste

Você pode usar o AWS SDK para criar um conjunto de dados de teste das seguintes formas:

- Use CreateDatasetcom um arquivo de manifesto no formato Amazon Sagemaker fornecido por você. Para obter mais informações, consulte the section called “Criar um arquivo de manifesto”.

Para obter um código de exemplo, consulte Criação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth.

- Use `CreateDataset` para copiar um conjunto de dados existente do Amazon Rekognition Custom Labels. Para obter um código de exemplo, consulte Como criar um conjunto de dados usando um conjunto de dados existente (SDK).
- Crie um conjunto de dados vazio com `CreateDataset` e adicione entradas do conjunto de dados posteriormente com `UpdateDatasetEntries`. Para criar um conjunto de dados vazio, consulte Como adicionar um conjunto de dados a um projeto. Para adicionar imagens a um conjunto de dados, consulte Como adicionar mais imagens (SDK). É necessário adicionar as entradas do conjunto de dados antes de treinar um modelo.
- Divida o conjunto de dados de treinamento em conjuntos de dados de treinamento e teste separados. Primeiro, crie um conjunto de dados de teste vazio com `CreateDataset`. Em seguida, mova 20% das entradas do conjunto de dados de treinamento para o conjunto de dados de teste ligando. DistributeDatasetEntries Para criar um conjunto de dados vazio, consulte Como adicionar um conjunto de dados a um projeto (SDK). Para dividir o conjunto de dados de treinamento, consulte Como distribuir um conjunto de dados de treinamento (SDK).

Importar imagens de um bucket do Amazon S3

As imagens são importadas de um bucket do Amazon S3. Você pode usar o bucket do console ou outro bucket do Amazon S3 em sua AWS conta. Se estiver usando o bucket do console, as permissões necessárias já estão configuradas. Se não estiver usando o bucket do console, consulte Como acessar os buckets externos do Amazon S3.

> **Note**
>
> Você não pode usar o AWS SDK para criar um conjunto de dados diretamente de imagens em um bucket do Amazon S3. Em vez disso, crie um arquivo de manifesto que faça referência aos locais de origem das imagens. Para obter mais informações, consulte Usar um arquivo de manifesto para importar imagens.

Durante a criação do conjunto de dados, é possível escolher atribuir nomes de rótulos às imagens com base no nome da pasta que contém as imagens. As pastas devem ser filhas do caminho da pasta do Amazon S3 especificada na localização da pasta do S3 durante a criação do conjunto de

dados. Para criar um conjunto de dados, consulte Criação de um conjunto de dados importando imagens de um bucket do S3.

Por exemplo, presuma a estrutura de pasta a seguir em um bucket do Amazon S3. Se especificar a localização da pasta do Amazon S3 como S3-bucket/Alexa-devices, as imagens na pasta echo receberão o rótulo echo. Da mesma forma, as imagens na pasta echo-dots recebem o rótulo echo-dot. Os nomes das pastas secundárias mais profundas não são usados para rotular imagens. Em vez disso, é usada a pasta secundária apropriada da localização da pasta do Amazon S3. Por exemplo, as imagens na pasta white-echo-dotsrecebem o rótulo echo-dot. As imagens no nível da localização da pasta S3 (alexa-devices) não têm rótulos atribuídos a elas.

Pastas mais profundas na estrutura de pastas podem ser usadas para rotular imagens especificando uma localização mais profunda da pasta S3. Por exemplo, se você especificar S3- bucket/alexa-devices/echo -dot, as imagens na pasta white-echo-dotserão rotuladas. white-echo-dot As imagens fora do local especificado da pasta s3, como echo, não são importadas.

```
S3-bucket
### alexa-devices
    ### echo
    #   ### echo-image-1.png
    #   ### echo-image-2.png
    #   ### .
    #   ### .
    ### echo-dot
        ### white-echo-dot
        #   ### white-echo-dot-image-1.png
        #   ### white-echo-dot-image-2.png
        #
        ### echo-dot-image-1.png
        ### echo-dot-image-2.png
        ### .
        ### .
```

Recomendamos que você use o bucket do Amazon S3 (bucket do console) criado para você pelo Amazon Rekognition quando você abriu o console pela primeira vez na região atual. AWS Se o bucket do Amazon S3 que você está usando for diferente (externo) do bucket do console, o console solicitará que você configure as permissões apropriadas durante a criação do conjunto de dados. Para obter mais informações, consulte the section called “Etapa 2: configure as permissões do console”.

Criação de um conjunto de dados importando imagens de um bucket do S3

O procedimento a seguir mostra como criar um conjunto de dados usando imagens armazenadas no bucket do Console S3. As imagens são automaticamente rotuladas com o nome da pasta na qual estão armazenadas.

Depois de importar suas imagens, é possível adicionar mais imagens, atribuir rótulos e adicionar caixas delimitadoras da página de galeria de um conjunto de dados. Para obter mais informações, consulte Rotulagem de imagens.

Faça upload das suas imagens em um bucket do Amazon Simple Storage Service

1. Crie uma pasta no sistema de arquivos local. Use um nome de pasta, como dispositivos-alexa.
2. Na pasta que você acabou de criar, crie pastas com o nome de cada rótulo que você deseja usar. Por exemplo, echo e echo-dot. A estrutura deve ser semelhante à que vem a seguir.

```
alexa-devices
### echo
#   ### echo-image-1.png
#   ### echo-image-2.png
#   ### .
#   ### .
### echo-dot
    ### echo-dot-image-1.png
    ### echo-dot-image-2.png
    ### .
    ### .
```

3. Coloque as imagens que correspondem a um rótulo na pasta com o mesmo nome do rótulo.
4. Faça login no AWS Management Console e abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
5. Adicione a pasta que você criou na etapa 1 ao bucket do Amazon S3 (bucket do console) criado para você pelo Amazon Rekognition Custom Labels durante a Primeira configuração. Para obter mais informações, consulte Como gerenciar um projeto do Amazon Rekognition Custom Labels.
6. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
7. Escolha Usar rótulos personalizados.
8. Escolha Comece a usar.
9. No painel de navegação esquerdo, selecione Projetos.

10. Na página Projetos, selecione o projeto ao qual você deseja adicionar um conjunto de dados. A página de detalhes do seu projeto é exibida.
11. Escolha Criar conjunto de dados. A página Criar conjunto de dados é exibida.
12. Em Configuração inicial, escolha Iniciar com um único conjunto de dados ou Iniciar com um conjunto de dados de treinamento. Para criar um modelo de maior qualidade, recomendamos começar com conjuntos de dados de treinamento e teste separados.

    Single dataset

    a. Na seção Detalhes do conjunto de dados de treinamento, escolha Importar imagens do bucket do S3.
    b. Na seção Detalhes do conjunto de dados de treinamento, insira as informações das etapas 13 a 15 na seção Configuração da fonte de imagem.

    Separate training and test datasets

    a. Na seção Detalhes do conjunto de dados de treinamento, escolha Importar imagens do bucket do S3.
    b. Na seção Detalhes do conjunto de dados de treinamento, insira as informações das etapas 13 a 15 na seção Configuração da fonte de imagem.
    c. Na seção Detalhes do conjunto de dados de teste, escolha Importar imagens do bucket do S3.
    d. Na seção Detalhes do conjunto de dados de teste, insira as informações das etapas 13 a 15 na seção Configuração da fonte de imagem.
13. Escolha Importar imagens do bucket do Amazon S3.
14. Em S3 URI, insira a localização do bucket do Amazon S3 e o caminho da pasta.
15. Escolha Anexar rótulos automaticamente às imagens com base na pasta.
16. Escolha Criar conjuntos de dados. A página de conjuntos de dados do seu projeto é aberta.
17. Se precisar adicionar ou alterar rótulos, faça Rotulagem de imagens.
18. Siga as etapas em Como treinar um modelo (console) para treinar seu modelo.

Importar imagens de um computador local

As imagens são carregadas diretamente do seu computador. É possível fazer upload de até 30 imagens por vez.

As imagens que você enviar não terão rótulos associados a elas. Para obter mais informações, consulte Rotulagem de imagens. Se tiver muitas imagens para carregar, considere usar um bucket do Amazon S3. Para obter mais informações, consulte Importar imagens de um bucket do Amazon S3.

> **Note**
>
> Você não pode usar o AWS SDK para criar um conjunto de dados com imagens locais. Em vez disso, crie um arquivo de manifesto e carregue as imagens em um bucket do Amazon S3. Para obter mais informações, consulte Usar um arquivo de manifesto para importar imagens.

Para criar um conjunto de dados usando imagens em um computador local (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, selecione o projeto ao qual você deseja adicionar um conjunto de dados. A página de detalhes do seu projeto é exibida.
6. Escolha Criar conjunto de dados. A página Criar conjunto de dados é exibida.
7. Em Configuração inicial, escolha Iniciar com um único conjunto de dados ou Iniciar com um conjunto de dados de treinamento. Para criar um modelo de maior qualidade, recomendamos começar com conjuntos de dados de treinamento e teste separados.

   Single dataset

   a. Na seção Detalhes do conjunto de dados de treinamento, escolha Fazer upload de imagens do seu computador.
   b. Escolha Criar conjunto de dados.
   c. Na página do conjunto de dados do projeto, escolha Adicionar imagens.
   d. Escolha as imagens que você deseja fazer upload no conjunto de dados dos arquivos do seu computador. É possível arrastar as imagens ou escolher as imagens que deseja carregar do seu computador local.
   e. Escolha Fazer upload de imagens.

Separate training and test datasets

a. Na seção Detalhes do conjunto de dados de treinamento, escolha Fazer upload de imagens do seu computador.
b. Na seção Detalhes do conjunto de dados de teste, escolha Fazer upload de imagens do seu computador.

> **Note**
>
> Seus conjuntos de dados de treinamento e teste podem ter fontes de imagem diferentes.

c. Escolha Criar conjuntos de dados. A página do conjunto de dados do seu projeto aparece com uma guia Treinamento e uma guia Teste para os respectivos conjuntos de dados.
d. Escolha Ações e escolha Adicionar imagens ao conjunto de dados de treinamento.
e. Escolha as imagens que você deseja fazer upload no conjunto de dados. É possível arrastar as imagens ou escolher as imagens que deseja carregar do seu computador local.
f. Escolha Fazer upload de imagens.
g. Repita as etapas de 5e a 5g. Para a etapa 5e, escolha Ações e escolha Adicionar imagens ao conjunto de dados de teste.

8. Siga as etapas em Rotulagem de imagens para rotular suas imagens.
9. Siga as etapas em Como treinar um modelo (console) para treinar o modelo.

Usar um arquivo de manifesto para importar imagens

Você pode criar um conjunto de dados usando um arquivo de manifesto no formato Amazon SageMaker AI Ground Truth. Você pode usar o arquivo de manifesto de um trabalho do Amazon SageMaker AI Ground Truth. Se suas imagens e rótulos não estiverem no formato de um arquivo de manifesto do SageMaker AI Ground Truth, você poderá criar um arquivo de manifesto no formato SageMaker AI e usá-lo para importar suas imagens rotuladas.

A operação `CreateDataset` foi atualizada para permitir que você opte por especificar etiquetas ao criar um conjunto de dados. As etiquetas são pares de chave/valor que podem ajudar você a organizar e categorizar os recursos.

Tópicos

- Criação de um conjunto de dados com um arquivo de manifesto do SageMaker AI Ground Truth (console)
- Criação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth
- Criar solicitação de conjunto de dados
- Rotulando imagens com um trabalho do Amazon SageMaker AI Ground Truth
- Criar um arquivo de manifesto
- Importar rótulos ao nível da imagem em arquivos de manifesto
- Localização de objetos em arquivos de manifesto
- Regras de validação para arquivos de manifesto
- Conversão de outros formatos de conjunto de dados em um arquivo de manifesto

Criação de um conjunto de dados com um arquivo de manifesto do SageMaker AI Ground Truth (console)

O procedimento a seguir mostra como criar um conjunto de dados usando um arquivo de manifesto no formato SageMaker AI Ground Truth.

1. Crie um arquivo de manifesto para o conjunto de dados de treinamento seguindo um destes procedimentos:

   - Crie um arquivo de manifesto com um SageMaker AI GroundTruth Job seguindo as instruções em Rotulando imagens com um trabalho do Amazon SageMaker AI Ground Truth.
   - Crie seu próprio arquivo de manifesto seguindo as instruções em Criar um arquivo de manifesto.

   Se quiser criar um conjunto de dados de teste, repita a etapa 1 para criar o conjunto de dados de teste.
2. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
3. Escolha Usar rótulos personalizados.
4. Escolha Comece a usar.
5. No painel de navegação esquerdo, selecione Projetos.

6. Na página Projetos, selecione o projeto ao qual você deseja adicionar um conjunto de dados. A página de detalhes do seu projeto é exibida.
7. Escolha Criar conjunto de dados. A página Criar conjunto de dados é exibida.
8. Em Configuração inicial, escolha Iniciar com um único conjunto de dados ou Iniciar com um conjunto de dados de treinamento. Para criar um modelo de maior qualidade, recomendamos começar com conjuntos de dados de treinamento e teste separados.

   Single dataset

   a. Na seção Detalhes do conjunto de dados de treinamento, escolha Importar imagens rotuladas por SageMaker Ground Truth.
   b. No local do arquivo de manifesto, insira o local do arquivo de manifesto criado na etapa 1.
   c. Escolha Criar conjunto de dados. A página de conjuntos de dados do seu projeto é aberta.

   Separate training and test datasets

   a. Na seção Detalhes do conjunto de dados de treinamento, escolha Importar imagens rotuladas por SageMaker Ground Truth.
   b. No local do arquivo de manifesto, insira o local do arquivo de manifesto do conjunto de dados de treinamento criado na etapa 1.
   c. Na seção Detalhes do conjunto de dados de teste, escolha Importar imagens rotuladas por SageMaker Ground Truth.

      > **Note**
      > Seus conjuntos de dados de treinamento e teste podem ter fontes de imagem diferentes.

   d. No local do arquivo de manifesto, insira o local do arquivo de manifesto do conjunto de dados de teste criado na etapa 1.
   e. Escolha Criar conjuntos de dados. A página de conjuntos de dados do seu projeto é aberta.
9. Se precisar adicionar ou alterar rótulos, faça Rotulagem de imagens.
10. Siga as etapas em Como treinar um modelo (console) para treinar seu modelo.

Criação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth

O procedimento a seguir mostra como criar conjuntos de dados de treinamento ou teste a partir de um arquivo de manifesto usando a CreateDatasetAPI.

Você pode usar um arquivo de manifesto existente, como a saída de um trabalho do SageMaker AI Ground Truth, ou criar seu próprio arquivo de manifesto.

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Crie um arquivo de manifesto para o conjunto de dados de treinamento seguindo um destes procedimentos:

   - Crie um arquivo de manifesto com um SageMaker AI GroundTruth Job seguindo as instruções em Rotulando imagens com um trabalho do Amazon SageMaker AI Ground Truth.
   - Crie seu próprio arquivo de manifesto seguindo as instruções em Criar um arquivo de manifesto.

   Se quiser criar um conjunto de dados de teste, repita a etapa 2 para criar o conjunto de dados de teste.
3. Use o código de exemplo a seguir para criar o conjunto de dados de treinamento e teste.

   AWS CLI

   Use o código a seguir para criar um conjunto de dados. Substitua o seguinte:

   - `project_arn`: o ARN do projeto ao qual você deseja adicionar o conjunto de dados de teste.
   - `type`: o tipo de conjunto de dados que você deseja criar (TREINAMENTO ou TESTE).
   - `bucket`: o bucket que contém o arquivo de manifesto do conjunto de dados.
   - `manifest_file`: o caminho e o nome do arquivo de manifesto.

```
aws rekognition create-dataset --project-arn project_arn \
  --dataset-type type \
  --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",
 "Name": "manifest_file" } } }' \
  --profile custom-labels-access
```

```
  --tags '{"key1": "value1", "key2": "value2"}'
```

Python

Use os valores a seguir para criar um conjunto de dados. Forneça os seguintes parâmetros de linha de comando:

- `project_arn`: o ARN do projeto ao qual você deseja adicionar o conjunto de dados de teste.
- `dataset_type`: o tipo de conjunto de dados que você deseja criar (`train` ou `test`).
- `bucket`: o bucket que contém o arquivo de manifesto do conjunto de dados.
- `manifest_file`: o caminho e o nome do arquivo de manifesto.

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)


import argparse
import logging
import time
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_dataset(rek_client, project_arn, dataset_type, bucket,
 manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    :param bucket: The S3 bucket that contains the manifest file.
    :param manifest_file: The path and filename of the manifest file.
    """
```

```
    try:
        #Create the project
        logger.info("Creating %s dataset for project %s",dataset_type,
project_arn)

        dataset_type = dataset_type.upper()

        dataset_source = json.loads(
            '{ "GroundTruthManifest": { "S3Object": { "Bucket": "'
            + bucket
            + '", "Name": "'
            + manifest_file
            + '" } } }'
        )

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type,
DatasetSource=dataset_source
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s",dataset_arn)

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":
                logger.info("Creating dataset: %s ",dataset_arn)
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished=True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
{dataset_arn}"
```

```
                logger.exception(error_message)
                raise Exception (error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn


    except ClientError as err:
        logger.exception("Couldn't create dataset: %s",err.response['Error']
['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
 (train or test)."
    )

    parser.add_argument(
        "bucket", help="The S3 bucket that contains the manifest file."
    )

    parser.add_argument(
        "manifest_file", help="The path and filename of the manifest file."
    )


def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    try:

        #Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating {args.dataset_type} dataset for project
 {args.project_arn}")

        #Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type,
            args.bucket,
            args.manifest_file)

        print(f"Finished creating dataset: {dataset_arn}")


    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")



if __name__ == "__main__":
    main()
```

Java V2

Use os valores a seguir para criar um conjunto de dados. Forneça os seguintes parâmetros de linha de comando:

- `project_arn`: o ARN do projeto ao qual você deseja adicionar o conjunto de dados de teste.
- `dataset_type`: o tipo de conjunto de dados que você deseja criar (`train` ou `test`).

- `bucket`: o bucket que contém o arquivo de manifesto do conjunto de dados.
- `manifest_file`: o caminho e o nome do arquivo de manifesto.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateDatasetManifestFiles {

    public static final Logger logger =
 Logger.getLogger(CreateDatasetManifestFiles.class.getName());

    public static String createMyDataset(RekognitionClient rekClient, String
 projectArn, String datasetType,
            String bucket, String name) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
 s3://{2}/{3} ",
```

```
                    new Object[] { datasetType, projectArn, bucket, name });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Could not create dataset. Unrecognized
dataset type: {0}", datasetType);
                throw new Exception("Could not create dataset. Unrecognized
dataset type: " + datasetType);

            }

            GroundTruthManifest groundTruthManifest =
GroundTruthManifest.builder()

.s3Object(S3Object.builder().bucket(bucket).name(name).build()).build();

            DatasetSource datasetSource =
DatasetSource.builder().groundTruthManifest(groundTruthManifest).build();

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);
```

```
                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
```

```
            throw e;
        }

    }

    public static void main(String[] args) {

        String datasetType = null;
        String bucket = null;
        String name = null;
        String projectArn = null;
        String datasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
 <dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
 copy the datast to.\n\n"
                + "   dataset_type - the type of the dataset that you want to
 create (train or test).\n\n"
                + "   bucket - the S3 bucket that contains the manifest file.\n
\n"
                + "   name - the location and name of the manifest file within
 the bucket.\n\n";

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];
        bucket = args[2];
        name = args[3];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();
```

```
             // Create the dataset
            datasetArn = createMyDataset(rekClient, projectArn, datasetType,
 bucket, name);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

4. Se precisar adicionar ou alterar rótulos, consulte Como gerenciar rótulos (SDK).
5. Siga as etapas em Treinando um modelo (SDK) para treinar seu modelo.

Criar solicitação de conjunto de dados

Veja a seguir o formato da solicitação de CreateDataset operação:

```
{
"DatasetSource": {
"DatasetArn": "string",
"GroundTruthManifest": {
"S3Object": {
"Bucket": "string",
"Name": "string",
"Version": "string"
}
}
},
"DatasetType": "string",
"ProjectArn": "string",
```

```
"Tags": {
"string": "string"
}
}
```

Rotulando imagens com um trabalho do Amazon SageMaker AI Ground Truth

Com o Amazon SageMaker AI Ground Truth, você pode usar trabalhadores da Amazon Mechanical Turk, uma empresa fornecedora de sua escolha, ou de uma força de trabalho interna e privada, juntamente com o aprendizado de máquina que permite criar um conjunto rotulado de imagens. O Amazon Rekognition Custom Labels SageMaker importa arquivos de manifesto do AI Ground Truth de um bucket do Amazon S3 que você especificar.

O Amazon Rekognition Custom Labels oferece suporte às seguintes tarefas do AI Ground Truth. SageMaker

- Classificação de imagens
- Caixa delimitadora

Os arquivos que você importa são as imagens e um arquivo de manifesto. O arquivo de manifesto contém informações do rótulo e da caixa delimitadora das imagens que você importa.

O Amazon Rekognition precisa de permissões para acessar o bucket do Amazon S3 onde suas imagens são armazenadas. Se estiver usando o bucket de console configurado para você pelo Amazon Rekognition Custom Labels, as permissões necessárias já estão configuradas. Se não estiver usando o bucket do console, consulte Como acessar os buckets externos do Amazon S3.

Criação de um arquivo de manifesto com um trabalho do SageMaker AI Ground Truth (console)

O procedimento a seguir mostra como criar um conjunto de dados usando imagens rotuladas por um trabalho do SageMaker AI Ground Truth. Os arquivos de saída do trabalho são armazenados no bucket do console do Amazon Rekognition Custom Labels.

Para criar um conjunto de dados usando imagens rotuladas por um trabalho do SageMaker AI Ground Truth (console)

1. Faça login no AWS Management Console e abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/

2. No bucket do console, crie uma pasta para armazenar suas imagens de treinamento.

> **Note**
>
> O bucket do console é criado quando você abre pela primeira vez o console Amazon Rekognition Custom Labels em uma região. AWS Para obter mais informações, consulte Como gerenciar um projeto do Amazon Rekognition Custom Labels.

3. Faça upload de suas imagens na pasta que acabou de criar.
4. No bucket do console, crie uma pasta para armazenar a saída do trabalho do Ground Truth.
5. Abra o console de SageMaker IA em https://console.aws.amazon.com/sagemaker/.
6. Crie um trabalho de rotulagem do Ground Truth. Você precisará do Amazon S3 URLs para as pastas que você criou nas etapas 2 e 4. Para obter mais informações, consulte Use Amazon SageMaker Ground Truth for Data Labeling.
7. Observe a localização do arquivo `output.manifest` na pasta que você criou na etapa 4. Ele deve estar na subpasta *`Ground-Truth-Job-Name`*`/manifests/output`.
8. Siga as instruções em Criação de um conjunto de dados com um arquivo de manifesto do SageMaker AI Ground Truth (console) para criar um conjunto de dados com o arquivo de manifesto carregado. Para a etapa 8, na localização do arquivo .manifest, insira a URL do Amazon S3 para a localização que você anotou na etapa anterior. Se você estiver usando o AWS SDK, useCriação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth.
9. Repita as etapas 1 a 6 para criar a tarefa SageMaker AI Ground Truth para seu conjunto de dados de teste.

Criar um arquivo de manifesto

Você pode criar um conjunto de dados de teste ou treinamento importando um arquivo de manifesto no formato SageMaker AI Ground Truth. Se suas imagens estiverem rotuladas em um formato que não seja um arquivo de manifesto do SageMaker AI Ground Truth, use as informações a seguir para criar um arquivo de manifesto no formato SageMaker AI Ground Truth.

Os arquivos de manifesto estão no formato de linhas JSON, onde cada linha é um objeto JSON completo representando as informações de rotulagem de uma imagem. O Amazon Rekognition Custom Labels SageMaker suporta manifestos AI Ground Truth com linhas JSON nos seguintes formatos:

- Saída do trabalho de classificação: use para adicionar rótulos em nível de imagem a uma imagem. Um rótulo em nível de imagem define a classe de cena, conceito ou objeto (se as informações de localização do objeto não forem necessárias) que está em uma imagem. Uma imagem pode ter mais de um rótulo no nível da imagem. Para obter mais informações, consulte Importar rótulos ao nível da imagem em arquivos de manifesto.
- Saída do trabalho da caixa delimitadora: use para rotular a classe e a localização de um ou mais objetos em uma imagem. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

As linhas JSON em nível de imagem e localização (caixa delimitadora) podem ser encadeadas no mesmo arquivo de manifesto.

> **Note**
>
> Os exemplos de linhas JSON nesta seção são formatados para facilitar a leitura.

Ao importar um arquivo de manifesto, o Amazon Rekognition Custom Labels aplica regras de validação para limites, sintaxe e semântica. Para obter mais informações, consulte Regras de validação para arquivos de manifesto.

As imagens referenciadas por um arquivo de manifesto devem estar localizadas no mesmo bucket do Amazon S3. O arquivo de manifesto pode estar localizado em um bucket do Amazon S3 diferente do bucket do Amazon S3 que armazena as imagens. A localização de uma imagem é especificada no campo `source-ref` de uma linha JSON.

O Amazon Rekognition precisa de permissões para acessar o bucket do Amazon S3 onde suas imagens são armazenadas. Se estiver usando o bucket de console configurado para você pelo Amazon Rekognition Custom Labels, as permissões necessárias já estão configuradas. Se não estiver usando o bucket do console, consulte Como acessar os buckets externos do Amazon S3.

Tópicos

- Como criar um arquivo de manifesto

Como criar um arquivo de manifesto

O procedimento a seguir cria um projeto com um conjunto de dados de treinamento e teste. Os conjuntos de dados são criados a partir dos arquivos de manifesto de treinamento e teste que você cria.

Para criar um conjunto de dados usando um arquivo de manifesto no formato SageMaker AI Ground Truth (console)

1. No bucket do console, crie uma pasta para armazenar seus arquivos de manifesto.
2. No bucket do console, crie uma pasta para armazenar suas imagens.
3. Faça upload de suas imagens na pasta que acabou de criar.
4. Crie um arquivo de manifesto no formato SageMaker AI Ground Truth para seu conjunto de dados de treinamento. Para obter mais informações, consulte Importar rótulos ao nível da imagem em arquivos de manifesto e Localização de objetos em arquivos de manifesto.

   > **Important**
   >
   > O valor do campo `source-ref` em cada linha JSON deve ser mapeado para uma imagem que você carregou.

5. Crie um arquivo de manifesto no formato SageMaker AI Ground Truth para seu conjunto de dados de teste.
6. Faça upload de seus arquivos de manifesto na pasta que acabou de criar.
7. Observe a localização do arquivo de manifesto.
8. Siga as instruções em Criação de um conjunto de dados com um arquivo de manifesto do SageMaker AI Ground Truth (console) para criar um conjunto de dados com o arquivo de manifesto carregado. Para a etapa 8, na localização do arquivo .manifest, insira a URL do Amazon S3 para a localização que você anotou na etapa anterior. Se você estiver usando o AWS SDK, useCriação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth.

Importar rótulos ao nível da imagem em arquivos de manifesto

Para importar rótulos em nível de imagem (imagens rotuladas com cenas, conceitos ou objetos que não exigem informações de localização), você adiciona linhas JSON no formato JSON do formato

AI Ground SageMaker Truth Classification Job Output a um arquivo de manifesto. Um arquivo de manifesto é feito de uma ou mais linhas JSON, uma para cada imagem que você deseja importar.

> **Tip**
>
> Para simplificar a criação de um arquivo de manifesto, fornecemos um script em Python que cria um arquivo de manifesto a partir de um arquivo CSV. Para obter mais informações, consulte Como criar um arquivo de manifesto de um arquivo CSV.

Para criar um arquivo de manifesto para rótulos em nível de imagem

1. Crie um arquivo de texto vazio.
2. Adicione uma linha JSON para cada imagem que você deseja importar. Cada linha deve ser semelhante à linha a seguir.

```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/
manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-
metadata":{"confidence":0.95,"job-name":"labeling-job/
testclconsolebucket","class-name":"Echo Dot","human-annotated":"yes","creation-
date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

3. Salve o arquivo. É possível usar a extensão `.manifest`, mas ela não é obrigatória.
4. Crie um conjunto de dados usando o arquivo de manifesto que você criou. Para obter mais informações, consulte Para criar um conjunto de dados usando um arquivo de manifesto no formato SageMaker AI Ground Truth (console).

Linhas JSON em nível de imagem

Nesta seção, é mostrado como criar uma linha JSON para uma única imagem. Considere a seguinte imagem: Uma cena para a imagem a seguir pode ser chamada de Sunrise.

A linha JSON da imagem anterior, com a cena Nascer do sol, pode ser a seguinte.

```
{
    "source-ref": "s3://bucket/images/sunrise.png",
    "testdataset-classification_Sunrise": 1,
    "testdataset-classification_Sunrise-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/testdataset-classification_Sunrise",
        "class-name": "Sunrise",
        "human-annotated": "yes",
        "creation-date": "2020-03-06T17:46:39.176",
        "type": "groundtruth/image-classification"
    }
}
```

Observe as seguintes informações:

source-ref

(Obrigatório) O local no Amazon S3 da imagem. O formato é `"s3://BUCKET/OBJECT_PATH"`. As imagens em um conjunto de dados importado devem ser armazenadas no mesmo bucket do Amazon S3.

testdataset-classification_Sunrise

(Obrigatório) O atributo do rótulo. Escolha o nome do campo. O valor do campo (1 no exemplo anterior) é um identificador de atributo de rótulo. Ele não é usado pelo Amazon Rekognition Custom Labels e pode ter qualquer valor inteiro. Deve haver metadados correspondentes identificados pelo nome do campo com -metadata anexado. Por exemplo, `."testdataset-classification_Sunrise-metadata"`

testdataset-classification_Sunrise-metadados

(Obrigatório) Metadados sobre o atributo do rótulo. O nome do campo deve ser o mesmo do atributo do rótulo com -metadata anexado.

confidence

(Obrigatório) Atualmente não é usado pelo Amazon Rekognition Custom Labels, mas um valor entre 0 e 1 deve ser fornecido.

job-name

(Opcional) Um nome que você escolhe para o trabalho que processa a imagem.

class-name

(Obrigatório) Um nome de classe que você escolhe para a cena ou conceito que se aplica à imagem. Por exemplo, `."Sunrise"`

human-annotated

(Obrigatório) Especifique `"yes"` se a anotação foi preenchida por um humano. Caso contrário, `"no"`.

creation-date

(Obrigatório) A data e hora do Tempo Universal Coordenado (UTC) em que o rótulo foi criado.

tipo

(Obrigatório) O tipo de processamento que deve ser aplicado à imagem. Para rótulos em nível de imagem, o valor é `"groundtruth/image-classification"`.

Como adicionar vários rótulos em nível de imagem a uma imagem

É possível adicionar vários rótulos a uma imagem. Por exemplo, o JSON a seguir adiciona dois rótulos, futebol e bola, a uma única imagem.

```
{
    "source-ref": "S3 bucket location",
    "sport0":0, # FIRST label
    "sport0-metadata": {
        "class-name": "football",
        "confidence": 0.8,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    },
    "sport1":1, # SECOND label
    "sport1-metadata": {
        "class-name": "ball",
        "confidence": 0.8,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    }
}  # end of annotations for 1 image
```

Localização de objetos em arquivos de manifesto

Você pode importar imagens rotuladas com informações de localização de objetos adicionando linhas JSON no formato SageMaker AI Ground Truth Bounding Box Job Output a um arquivo de manifesto.

As informações de localização representam a localização de um objeto em uma imagem. A localização é representada por uma caixa delimitadora que circunda o objeto. A estrutura da caixa delimitadora contém as coordenadas no canto superior esquerdo da caixa delimitadora e a largura

e altura da caixa delimitadora. Uma linha JSON em formato de caixa delimitadora inclui caixas delimitadoras para a localização de um ou mais objetos em uma imagem e a classe de cada objeto na imagem.

Um arquivo de manifesto é feito de uma ou mais linhas JSON, cada linha contém as informações de uma única imagem.

Para criar um arquivo de manifesto para localização de objetos

1. Crie um arquivo de texto vazio.
2. Adicione uma linha JSON para cada imagem que você deseja importar. Cada linha deve ser semelhante à linha a seguir.

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size":
 [{"width": 640, "height": 480, "depth": 3}], "annotations": [{ "class_id":
 1, "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0,
 "top": 65, "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata":
 {"objects": [{ "confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo",
 "1": "Echo Dot"}, "type": "groundtruth/object-detection", "human-annotated":
 "yes", "creation-date": "2013-11-18T02:53:27", "job-name": "my job"}}
```

3. Salve o arquivo. É possível usar a extensão `.manifest`, mas ela não é obrigatória.
4. Crie um conjunto de dados usando o arquivo que você criou. Para obter mais informações, consulte Para criar um conjunto de dados usando um arquivo de manifesto no formato SageMaker AI Ground Truth (console).

Linhas JSON da caixa delimitadora de objetos

Nesta seção, é mostrado como criar uma linha JSON para uma única imagem. A imagem a seguir mostra as caixas delimitadoras ao redor de dispositivos Amazon Echo e um Amazon Echo Dot.

A seguir está a linha JSON da caixa delimitadora da imagem anterior.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
```

```
    "height": 101
   }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2013-11-18T02:53:27",
   "job-name": "my job"
  }
 }
```

Observe as seguintes informações:

source-ref

(Obrigatório) O local no Amazon S3 da imagem. O formato é `"s3://BUCKET/OBJECT_PATH"`. As imagens em um conjunto de dados importado devem ser armazenadas no mesmo bucket do Amazon S3.

bounding-box

(Obrigatório) O atributo do rótulo. Escolha o nome do campo. Contém o tamanho da imagem e as caixas delimitadoras de cada objeto detectado na imagem. Deve haver metadados correspondentes identificados pelo nome do campo com -metadata anexado. Por exemplo, `."bounding-box-metadata"`

image_size

(Obrigatório) Uma matriz de elementos únicos contendo o tamanho da imagem em pixels.

- height: (obrigatória) a altura da imagem em pixels.
- width: (obrigatório) a profundidade da imagem em pixels.
- depth: (obrigatório) o número de canais na imagem. Para imagens RGB, o valor é 3. Não é usado pelo Amazon Rekognition Custom Labels neste momento, mas um valor é obrigatório.

anotações

(Obrigatório) Uma matriz de informações da caixa delimitadora para cada objeto detectado na imagem.

- class_id (obrigatório) mapeia para o rótulo no class-map. No exemplo anterior, o objeto com o class_id de 1 é o Echo Dot na imagem.
- top: (obrigatório) a distância da parte superior da imagem à parte superior da caixa delimitadora, em pixels.
- left: (obrigatório) a distância da esquerda da imagem à esquerda da caixa delimitadora, em pixels.
- width: (obrigatório) a largura da caixa delimitadora em pixels.
- height: (obrigatória) a altura da caixa delimitadora em pixels.

bounding-box-metadados

(Obrigatório) Metadados sobre o atributo do rótulo. O nome do campo deve ser o mesmo do atributo do rótulo com -metadata anexado. Uma matriz de informações da caixa delimitadora para cada objeto detectado na imagem.

Objetos

(Obrigatório) Uma matriz de objetos que estão na imagem. Mapeia para a matriz de anotações por índice. O atributo de confiança não é usado pelo Amazon Rekognition Custom Labels.

class-map

(Obrigatório) Um mapa das classes que se aplicam aos objetos detectados na imagem.

tipo

(Obrigatório) O tipo de trabalho de classificação. `"groundtruth/object-detection"` identifica o trabalho como detecção de objetos.

creation-date

(Obrigatório) A data e hora do Tempo Universal Coordenado (UTC) em que o rótulo foi criado.

human-annotated

(Obrigatório) Especifique `"yes"` se a anotação foi preenchida por um humano. Caso contrário, `"no"`.

job-name

(Opcional) O nome do trabalho que processa a imagem.

Regras de validação para arquivos de manifesto

Ao importar um arquivo de manifesto, o Amazon Rekognition Custom Labels aplica regras de validação para limites, sintaxe e semântica. O esquema SageMaker AI Ground Truth impõe a validação da sintaxe. Para obter mais informações, consulte Saídas. A seguir estão as regras de validação para limites e semântica.

> **Note**
>
> - As regras de invalidade de 20% se aplicam cumulativamente a todas as regras de validação. Se a importação exceder o limite de 20% devido a qualquer combinação, como 15% de JSON inválido e 15% de imagens inválidas, a importação falhará.
> - Cada objeto do conjunto de dados é uma linha no manifesto. Linhas em branco/inválidas também são contadas como objetos do conjunto de dados.
> - As sobreposições são (rótulos comuns entre teste e treinamento)/(rótulos de treinamento).

Tópicos

- Limites
- Semântica

Limites

Validação	Limite	Erro gerado
Tamanho do arquivo de manifesto	Máximo de 1 GB	Erro
Contagem máxima de linhas para um arquivo de manifesto	Máximo de 250 mil objetos do conjunto de dados como linhas em um manifesto.	Erro
Limite inferior no número total de objetos de conjunto de dados válidos por rótulo	>=1	Erro
Limite inferior nos rótulos	>=2	Erro
Limite superior nos rótulos	<= 250	Erro
Mínimo de caixas delimitad oras por imagem	0	Nenhum
Máximo de caixas delimitad oras por imagem	50	Nenhum

Semântica

Validação	Limite	Erro gerado
Manifesto vazio		Erro
Objeto /in-accessible source-ref ausente	Número de objetos menor que 20%	Aviso
Objeto /in-accessible source-ref ausente	Número de objetos > 20%	Erro

Validação	Limite	Erro gerado
Rótulos de teste não presentes no conjunto de dados de treinamento	Pelo menos 50% de sobreposição nos rótulos	Erro
Combinação de exemplos de rótulos versus objetos para o mesmo rótulo em um conjunto de dados. Classificação e detecção da mesma classe em um objeto de conjunto de dados.		Nenhum erro ou aviso
Como sobrepor ativos entre teste e treinamento	Não deve haver uma sobreposição entre os conjuntos de dados de teste e treinamento.	
As imagens em um conjunto de dados devem ser do mesmo bucket	Erro se os objetos estiverem em um bucket diferente	Erro

Conversão de outros formatos de conjunto de dados em um arquivo de manifesto

Você pode usar as informações a seguir para criar arquivos de manifesto no formato Amazon SageMaker AI a partir de uma variedade de formatos de conjuntos de dados de origem. Depois de criar o arquivo de manifesto, use-o para criar um conjunto de dados. Para obter mais informações, consulte Usar um arquivo de manifesto para importar imagens.

Tópicos

- Transformar um conjunto de dados COCO em um formato de arquivo de manifesto
- Transformando arquivos de manifesto com vários rótulos do SageMaker AI Ground Truth
- Como criar um arquivo de manifesto de um arquivo CSV

Transformar um conjunto de dados COCO em um formato de arquivo de manifesto

COCO é um formato para especificar conjuntos de dados de detecção, segmentação e legendagem de objetos em grande escala. Este exemplo em Python mostra como transformar um conjunto de dados no formato de detecção de objetos COCO em um arquivo de manifesto no formato de caixa delimitadora no Amazon Rekognition Custom Labels. Esta seção também inclui informações que podem ser usadas para escrever seu próprio código.

Um arquivo JSON no formato COCO consiste em cinco seções que fornecem informações para um conjunto de dados inteiro. Para obter mais informações, consulte O formato de conjunto de dados COCO.

- `info`: as informações gerais sobre o conjunto de dados.
- `licenses` : informações de licença para as imagens no conjunto de dados.
- `images`: uma lista de imagens no conjunto de dados.
- `annotations`: uma lista de anotações (incluindo caixas delimitadoras) que estão presentes em todas as imagens no conjunto de dados.
- `categories`: uma lista de categorias de rótulo.

São precisas informações das listas `images`, `annotations` e `categories` para criar um arquivo de manifesto do Amazon Rekognition Custom Labels.

Um arquivo de manifesto Amazon Rekognition Custom Labels está no formato de linhas JSON, onde cada linha tem a caixa delimitadora e as informações do rótulo de um ou mais objetos em uma imagem. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

Como mapear objetos COCO para uma linha JSON de rótulos personalizados

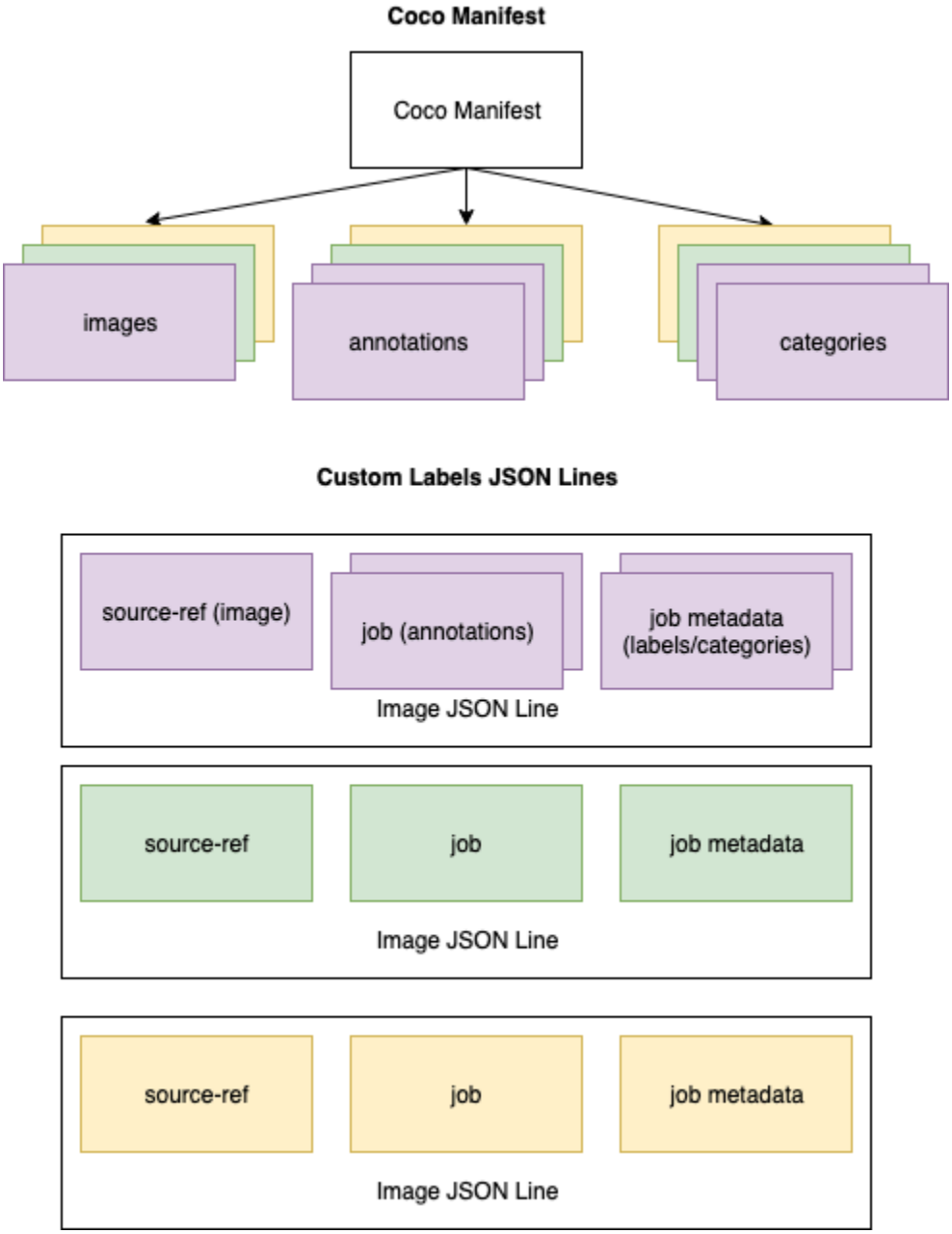
Para transformar um conjunto de dados no formato COCO, você mapeia o conjunto de dados COCO para um arquivo de manifesto do Amazon Rekognition Custom Labels para localização de objetos. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto. Para criar uma linha JSON para cada imagem, o arquivo de manifesto precisa mapear o conjunto de dados `image` COCO e o `annotation` campo do `category` objeto. IDs

Veja a seguir um exemplo de arquivo de manifesto COCO. Para obter mais informações, consulte O formato de conjunto de dados COCO.

```
{
```

```
    "info": {
        "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
 "1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
    },
    "licenses": [
        {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
 "Attribution License"}
    ],
    "images": [
        {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
 "date_captured": "2013-11-15 02:41:42"},
        {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,......167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
        {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
        {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```


O diagrama a seguir mostra como o conjunto de dados COCO lista para um mapa de conjunto de dados às linhas JSON do Amazon Rekognition Custom Labels para uma imagem. Cada linha JSON de uma imagem tem um campo de referência de origem, trabalho e metadados do trabalho. As cores correspondentes indicam informações para uma única imagem. Observe que, no manifesto, uma imagem individual pode ter várias anotações e metadados/categorias.



Para obter os objetos COCO para uma única linha JSON

1. Para cada imagem na lista de imagens, obtenha a anotação da lista de anotações em que o valor do campo de anotação `image_id`corresponda ao campo da imagem `id`.
2. Para cada anotação correspondida na etapa 1, leia a lista `categories` e obtenha cada `category` em que o valor da `id` do campo `category` corresponda ao objeto `annotation` do campo `category_id`.

3. Crie uma linha JSON para a imagem usando os objetos `image`, `annotation` e `category` correspondentes. Para mapear os campos, consulte Como mapear campos de objetos COCO para campos de objeto da linha JSON de rótulos personalizados.
4. Repita as etapas de 1 a 3 até criar linhas JSON para cada objeto `image` na lista `images`.

Para obter um código de exemplo, consulte Como transformar um conjunto de dados COCO.

Como mapear campos de objetos COCO para campos de objeto da linha JSON de rótulos personalizados

Depois de identificar os objetos COCO para uma linha JSON do Amazon Rekognition Custom Labels, você precisa mapear os campos do objeto COCO para os respectivos campos de objeto de linha JSON do Amazon Rekognition Custom Labels. O exemplo a seguir da linha JSON do Amazon Rekognition Custom Labels mapeia uma imagem (`id=000000245915`) para o exemplo anterior de COCO JSON. Observe as seguintes informações:

- `source-ref` é o local da imagem em um bucket do Amazon S3. Se suas imagens COCO não estiverem armazenadas em um bucket do Amazon S3, você precisa movê-las para um bucket do Amazon S3.
- A lista `annotations` contém um objeto `annotation` para cada objeto na imagem. Um objeto `annotation` inclui informações da caixa delimitadora (`top`, `left`, `width`, `height`) e um identificador de rótulo (`class_id`).
- O identificador do rótulo (`class_id`) é mapeado para a lista `class-map` nos metadados. Ele lista os rótulos usados na imagem.

```
{
 "source-ref": "s3://custom-labels-bucket/images/000000245915.jpg",
 "bounding-box": {
  "image_size": {
   "width": 640,
   "height": 480,
   "depth": 3
  },
  "annotations": [{
   "class_id": 0,
   "top": 251,
   "left": 399,
   "width": 155,
```

```
    "height": 101
   }, {
    "class_id": 1,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
 }
```

Use as informações a seguir para mapear os campos do arquivo de manifesto do Amazon Rekognition Custom Labels para os campos JSON do conjunto de dados COCO.

source-ref

O URL do formato S3 para a localização da imagem. A imagem deve ser armazenada em um bucket do S3. Para obter mais informações, consulte source-ref. Se o campo COCO `coco_url` apontar para uma localização de bucket do S3, será possível usar o valor de `coco_url` para o valor de `source-ref`. Como alternativa, é possível mapear o `source-ref` para o campo `file_name` (COCO) e, em seu código de transformação, adicionar o caminho do S3 necessário ao local em que a imagem está armazenada.

bounding-box

Um nome de atributo de rótulo de sua escolha. Para obter mais informações, consulte *bounding-box*.

image_size

O tamanho da imagem em pixels. Mapeia para um objeto `image` na lista de imagens.

- `height`-> `image.height`
- `width`-> `image.width`
- `depth`-> Não é usado pelo Amazon Rekognition Custom Labels, mas um valor deve ser fornecido.

anotações

Uma lista dos objetos `annotation`. Há um `annotation` para cada objeto na imagem.

anotação

Contém informações da caixa delimitadora de uma instância de um objeto na imagem.

- `class_id`-> mapeamento de identificação numérica para a lista de `class-map` do rótulo personalizado.
- `top` -> `bbox[1]`
- `left` -> `bbox[0]`
- `width` -> `bbox[2]`
- `height` -> `bbox[3]`

bounding-box-metadados

Metadados para o atributo de rótulo. Inclui os rótulos e os identificadores dos rótulos. Para obter mais informações, consulte *`bounding-box`*-metadados.

Objetos

Uma matriz de objetos na imagem. Mapas para a lista `annotations` por índice.

Objeto

- `confidence`-> Não é usado pelo Amazon Rekognition Custom Labels, mas um valor (1) é obrigatório.

class-map

Um mapa dos rótulos (classes) que se aplicam aos objetos detectados na imagem. Mapeia para objetos de categoria na lista de categorias.

- `id` -> `category.id`
- `id value` -> `category.name`

type

Deve ser `groundtruth/object-detection`

human-annotated

Especifique `yes` ou no. Para obter mais informações, consulte *`bounding-box`*-metadados.

creation-date -> image.date_captured

A data e a hora da criação da imagem. Mapeia para o campo imagem .date_capture de uma imagem na lista de imagens COCO. O Amazon Rekognition Custom Labels espera que o formato de `creation-date` seja Y-M-DTH:M:S.

job-name

Um nome de trabalho de sua escolha.

O formato de conjunto de dados COCO

Um conjunto de dados COCO consiste de cinco seções de informações que fornecem informações para todo o conjunto de dados. O formato de um conjunto de dados de detecção de objetos COCO está documentado em COCO Data Format.

- info: as informações gerais sobre o conjunto de dados.
- licences: informações de licença para as imagens no conjunto de dados.
- images: uma lista de imagens no conjunto de dados.
- annotations: uma lista de anotações (incluindo caixas delimitadoras) que estão presentes em todas as imagens no conjunto de dados.
- categories: uma lista de categorias de rótulo.

Para criar um manifesto de rótulos personalizados, você usa as listas `images`, `annotations` e `categories` do arquivo de manifesto COCO. As outras seções (`info`, `licences`) não são obrigatórias. Veja a seguir um exemplo de arquivo de manifesto COCO.

```
{
    "info": {
        "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
 "1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
    },
    "licenses": [
        {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
 "Attribution License"}
    ],
    "images": [
        {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
 "date_captured": "2013-11-15 02:41:42"},
        {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,......167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
        {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
        {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```

lista de imagens

As imagens referenciadas por um conjunto de dados COCO são listadas na matriz de imagens. Cada objeto de imagem contém informações sobre a imagem, como o nome do arquivo da imagem. No exemplo de objeto de imagem a seguir, observe as seguintes informações e quais campos são necessários para criar um arquivo de manifesto Amazon Rekognition Custom Labels.

- `id`: (obrigatório) um identificador exclusivo para a imagem. O campo `id` mapeia para o campo `id` na matriz de anotações (onde as informações da caixa delimitadora são armazenadas).
- `license`: (não obrigatória) mapeia para a matriz de licenças.
- `coco_url`: (opcional) o local da imagem.
- `flickr_url`: (não obrigatório) o local da imagem no Flickr.
- `width`: (obrigatório) a largura da imagem.
- `height`: (obrigatório) a altura da imagem.
- `file_name`: (obrigatório) o nome do arquivo de imagem. Neste exemplo, `file_name` e `id` combinam, mas não é um requisito para conjuntos de dados COCO.
- `date_captured`: (obrigatória) a data e a hora em que a imagem foi capturada.

```
{
    "id": 245915,
    "license": 4,
    "coco_url": "http://images.cocodataset.org/val2017/nnnnnnnnnnnn.jpg",
    "flickr_url": "http://farm1.staticflickr.com/88/nnnnnnnnnnnnnnnnnn.jpg",
    "width": 640,
    "height": 480,
    "file_name": "000000245915.jpg",
    "date_captured": "2013-11-18 02:53:27"
}
```

lista de anotações (caixas delimitadoras)

As informações da caixa delimitadora de todos os objetos em todas as imagens são armazenadas na lista de anotações. Um único objeto de anotação contém informações da caixa delimitadora de um único objeto e o rótulo do objeto em uma imagem. Há um objeto de anotação para cada instância de um objeto em uma imagem.

No exemplo a seguir, observe as seguintes informações e quais campos são necessários para criar um arquivo de manifesto Amazon Rekognition Custom Labels.

- `id`: (não obrigatório) o identificador da anotação.
- `image_id`: (obrigatório) corresponde ao `id` da imagem na matriz de imagens.
- `category_id`: (obrigatório) o identificador da etiqueta que identifica o objeto dentro de uma caixa delimitadora. Ele mapeia para o campo `id` da matriz de categorias.
- `iscrowd`: (não obrigatório) especifica se a imagem contém uma multidão de objetos.
- `segmentation`: (não obrigatória) informações de segmentação de objetos em uma imagem. O Amazon Rekognition Custom Labels não é compatível com a segmentação.
- `area`: (não obrigatória) a área da anotação.
- `bbox`: (obrigatório) contém as coordenadas, em pixels, de uma caixa delimitadora ao redor de um objeto na imagem.

```
{
    "id": 1409619,
    "category_id": 1,
    "iscrowd": 0,
    "segmentation": [
        [86.0, 238.8,..........382.74, 241.17]
    ],
    "image_id": 245915,
    "area": 3556.2197000000015,
    "bbox": [86, 65, 220, 334]
}
```

lista de categorias

As informações do rótulo são armazenadas na matriz de categorias. No exemplo de objeto de categoria a seguir, observe as seguintes informações e quais campos são necessários para criar um arquivo de manifesto Amazon Rekognition Custom Labels.

- `supercategory`: (não obrigatória) a categoria principal de uma etiqueta.
- `id`: (obrigatório) o identificador da etiqueta. O campo `id` mapeia para o campo `category_id` em um objeto `annotation`. No exemplo a seguir, o identificador de um ponto de eco é 2.
- `name`: (obrigatório) o nome do rótulo.

```
        {"supercategory": "speaker","id": 2,"name": "echo dot"}
```

Como transformar um conjunto de dados COCO

Use o exemplo em Python a seguir para transformar informações de caixa delimitadora de um conjunto de dados no formato COCO em um arquivo de manifesto no Amazon Rekognition Custom Labels. O código faz upload do arquivo de manifesto criado para o bucket do Amazon S3. O código também fornece um comando da AWS CLI que é possível usar para fazer upload de suas imagens.

Para transformar um conjunto de dados COCO (SDK)

1. Se ainda não tiver feito isso:

   a. Certifique-se de que você tem as permissões `AmazonS3FullAccess`. Para obter mais informações, consulte Configurar permissões do SDK.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.

2. Use o código em Python a seguir para transformar um conjunto de dados COCO. Defina os seguintes valores:

   - `s3_bucket`: o nome do bucket do S3 no qual você deseja armazenar as imagens e o arquivo de manifesto do Amazon Rekognition Custom Labels.
   - `s3_key_path_images`: o caminho para onde você deseja colocar as imagens no bucket do S3 (`s3_bucket`).
   - `s3_key_path_manifest_file`: o caminho para onde você deseja colocar o arquivo de manifesto de rótulos personalizados no bucket do S3 (`s3_bucket`).
   - `local_path`: o caminho local para onde o exemplo abre o conjunto de dados COCO de entrada e também salva o novo arquivo de manifesto do Custom Labels.
   - `local_images_path`: o caminho local para as imagens que você deseja usar para treinamento.
   - `coco_manifest`: o nome do arquivo do conjunto de dados COCO de entrada.
   - `cl_manifest_file`: um nome para o arquivo de manifesto criado pelo exemplo. O arquivo é salvo no local especificado por `local_path`. Por convenção, o arquivo tem a extensão `.manifest`, mas isso não é obrigatório.
   - `job_name`: um nome para o trabalho de rótulos personalizados.

```
import json
import os
```

```
import random
import shutil
import datetime
import botocore
import boto3
import PIL.Image as Image
import io

#S3 location for images
s3_bucket = 'bucket'
s3_key_path_manifest_file = 'path to custom labels manifest file/'
s3_key_path_images = 'path to images/'
s3_path='s3://' + s3_bucket  + '/' + s3_key_path_images
s3 = boto3.resource('s3')

#Local file information
local_path='path to input COCO dataset and output Custom Labels manifest/'
local_images_path='path to COCO images/'
coco_manifest = 'COCO dataset JSON file name'
coco_json_file = local_path + coco_manifest
job_name='Custom Labels job name'
cl_manifest_file = 'custom_labels.manifest'

label_attribute ='bounding-box'

open(local_path + cl_manifest_file, 'w').close()

# class representing a Custom Label JSON line for an image
class cl_json_line:
    def __init__(self,job, img):

        #Get image info. Annotations are dealt with seperately
        sizes=[]
        image_size={}
        image_size["width"] = img["width"]
        image_size["depth"] = 3
        image_size["height"] = img["height"]
        sizes.append(image_size)

        bounding_box={}
        bounding_box["annotations"] = []
        bounding_box["image_size"] = sizes

        self.__dict__["source-ref"] = s3_path + img['file_name']
```

```
        self.__dict__[job] = bounding_box

        #get metadata
        metadata = {}
        metadata['job-name'] = job_name
        metadata['class-map'] = {}
        metadata['human-annotated']='yes'
        metadata['objects'] = []
        date_time_obj = datetime.datetime.strptime(img['date_captured'], '%Y-%m-%d
 %H:%M:%S')
        metadata['creation-date']= date_time_obj.strftime('%Y-%m-%dT%H:%M:%S')
        metadata['type']='groundtruth/object-detection'

        self.__dict__[job + '-metadata'] = metadata


print("Getting image, annotations, and categories from COCO file...")

with open(coco_json_file) as f:

    #Get custom label compatible info
    js = json.load(f)
    images = js['images']
    categories = js['categories']
    annotations = js['annotations']

    print('Images: ' + str(len(images)))
    print('annotations: ' + str(len(annotations)))
    print('categories: ' + str(len (categories)))


print("Creating CL JSON lines...")

images_dict = {image['id']: cl_json_line(label_attribute, image) for image in
 images}

print('Parsing annotations...')
for annotation in annotations:

    image=images_dict[annotation['image_id']]

    cl_annotation = {}
    cl_class_map={}
```

```
    # get bounding box information
    cl_bounding_box={}
    cl_bounding_box['left'] = annotation['bbox'][0]
    cl_bounding_box['top'] = annotation['bbox'][1]

    cl_bounding_box['width'] = annotation['bbox'][2]
    cl_bounding_box['height'] = annotation['bbox'][3]
    cl_bounding_box['class_id'] = annotation['category_id']

    getattr(image, label_attribute)['annotations'].append(cl_bounding_box)


    for category in categories:
         if annotation['category_id'] == category['id']:
            getattr(image, label_attribute + '-metadata')['class-map']
[category['id']]=category['name']


    cl_object={}
    cl_object['confidence'] = int(1)  #not currently used by Custom Labels
    getattr(image, label_attribute + '-metadata')['objects'].append(cl_object)

print('Done parsing annotations')

# Create manifest file.
print('Writing Custom Labels manifest...')

for im in images_dict.values():

    with open(local_path+cl_manifest_file, 'a+') as outfile:
            json.dump(im.__dict__,outfile)
            outfile.write('\n')
            outfile.close()

# Upload manifest file to S3 bucket.
print ('Uploading Custom Labels manifest file to S3 bucket')
print('Uploading'  + local_path + cl_manifest_file + ' to ' +
 s3_key_path_manifest_file)
print(s3_bucket)
s3 = boto3.resource('s3')
s3.Bucket(s3_bucket).upload_file(local_path + cl_manifest_file,
 s3_key_path_manifest_file + cl_manifest_file)

# Print S3 URL to manifest file,
```

```
print ('S3 URL Path to manifest file. ')
print('\033[1m s3://' + s3_bucket + '/' + s3_key_path_manifest_file +
 cl_manifest_file + '\033[0m')

# Display aws s3 sync command.
print ('\nAWS CLI s3 sync command to upload your images to S3 bucket. ')
print ('\033[1m aws s3 sync ' + local_images_path + ' ' + s3_path + '\033[0m')
```

3. Execute o código.
4. Na saída do programa, observe o comando `s3 sync`. Você precisa dele na próxima etapa.
5. No prompt de comando, execute o comando `s3 sync`. Suas imagens são carregadas para o bucket do S3. Se o comando falhar durante o upload, execute-o novamente até que suas imagens locais estejam sincronizadas com o bucket do S3.
6. Na saída do programa, observe o caminho do URL do S3 para o arquivo de manifesto. Você precisa dele na próxima etapa.
7. Siga a instrução em Criação de um conjunto de dados com um arquivo de manifesto do SageMaker AI Ground Truth (console) para criar um conjunto de dados com o arquivo de manifesto carregado. Para a etapa 8, na localização do arquivo .manifest, insira a URL do Amazon S3 que você observou na etapa anterior. Se estiver usando o AWS SDK, use Criação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth.

Transformando arquivos de manifesto com vários rótulos do SageMaker AI Ground Truth

Este tópico mostra como transformar um arquivo de manifesto com vários rótulos do Amazon SageMaker AI Ground Truth em um arquivo de manifesto no formato Amazon Rekognition Custom Labels.

SageMaker Os arquivos de manifesto do AI Ground Truth para trabalhos com vários rótulos são formatados de forma diferente dos arquivos de manifesto no formato Amazon Rekognition Custom Labels. A classificação com vários rótulos ocorre quando uma imagem é classificada em um conjunto de classes, mas pode pertencer a várias classes ao mesmo tempo. Neste caso, a imagem pode ter potencialmente vários rótulos (vários rótulos), como futebol e bola.

Para obter informações sobre trabalhos com vários rótulos do SageMaker AI Ground Truth, consulte Classificação de imagens (vários rótulos). Para obter informações sobre os arquivos de manifesto do Amazon Rekognition Custom Labels em formato de vários rótulos, consulte the section called “Como adicionar vários rótulos em nível de imagem a uma imagem”.

Obtendo o arquivo de manifesto para um trabalho do SageMaker AI Ground Truth

O procedimento a seguir mostra como obter o arquivo manifesto de saída (`output.manifest`) para um trabalho do Amazon SageMaker AI Ground Truth. `output.manifest` será usado como entrada para o próximo procedimento.

Para baixar um arquivo de manifesto de trabalho do SageMaker AI Ground Truth

1. Abra a https://console.aws.amazon.com/sagemaker/.
2. No painel de navegação, escolha Ground Truth e escolha Labeling Jobs.
3. Escolha o trabalho de rotulagem que contém o arquivo de manifesto que você deseja usar.
4. Na página de detalhes, escolha o link em Local do conjunto de dados de saída. O console do Amazon S3 é aberto no local do conjunto de dados.
5. Escolha `Manifests`, `output` e depois `output.manifest`.
6. Escolha Ações de objeto e escolha Download para baixar o arquivo de manifesto.

Transformação de um arquivo de manifesto de SageMaker IA com vários rótulos

O procedimento a seguir cria um arquivo de manifesto Amazon Rekognition Custom Labels no formato de vários rótulos a partir de um arquivo de manifesto AI existente no formato de vários rótulos. SageMaker GroundTruth

> **Note**
>
> Para executar o código, você precisa do Python versão 3 ou superior.

Para transformar um arquivo de manifesto de SageMaker IA com vários rótulos

1. Execute o código em Python a seguir. Forneça o nome do arquivo de manifesto criado em Obtendo o arquivo de manifesto para um trabalho do SageMaker AI Ground Truth como um argumento da linha de comando.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier:  Apache-2.0
"""
Purpose
Shows how to create and Amazon Rekognition Custom Labels format
manifest file from an Amazon SageMaker Ground Truth Image
```

```
Classification (Multi-label) format manifest file.
"""
import json
import logging
import argparse
import os.path

logger = logging.getLogger(__name__)

def create_manifest_file(ground_truth_manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels format manifest file from
    an Amazon SageMaker Ground Truth Image Classification (Multi-label) format
    manifest file.
    :param: ground_truth_manifest_file: The name of the Ground Truth manifest file,
    including the relative path.
    :return: The name of the new Custom Labels manifest file.
    """

    logger.info('Creating manifest file from %s', ground_truth_manifest_file)
    new_manifest_file =
 f'custom_labels_{os.path.basename(ground_truth_manifest_file)}'

    # Read the SageMaker Ground Truth manifest file into memory.
    with open(ground_truth_manifest_file) as gt_file:
        lines = gt_file.readlines()

    #Iterate through the lines one at a time to generate the
    #new lines for the Custom Labels manifest file.
    with open(new_manifest_file, 'w') as the_new_file:
        for line in lines:
            #job_name - The of the Amazon Sagemaker Ground Truth job.
            job_name = ''
            # Load in the old json item from the Ground Truth manifest file
            old_json = json.loads(line)

            # Get the job name
            keys = old_json.keys()
            for key in keys:
                if 'source-ref' not in key and '-metadata' not in key:
                    job_name = key

            new_json = {}
            # Set the location of the image
```

```
            new_json['source-ref'] = old_json['source-ref']

            # Temporarily store the list of labels
            labels = old_json[job_name]

            # Iterate through the labels and reformat to Custom Labels format
            for index, label in enumerate(labels):
                new_json[f'{job_name}{index}'] = index
                metadata = {}
                metadata['class-name'] = old_json[f'{job_name}-metadata']['class-
map'][str(label)]
                metadata['confidence'] = old_json[f'{job_name}-metadata']
['confidence-map'][str(label)]
                metadata['type'] = 'groundtruth/image-classification'
                metadata['job-name'] = old_json[f'{job_name}-metadata']['job-name']
                metadata['human-annotated'] = old_json[f'{job_name}-metadata']
['human-annotated']
                metadata['creation-date'] = old_json[f'{job_name}-metadata']
['creation-date']
                # Add the metadata to new json line
                new_json[f'{job_name}{index}-metadata'] = metadata
            # Write the current line to the json file
            the_new_file.write(json.dumps(new_json))
            the_new_file.write('\n')

    logger.info('Created %s', new_manifest_file)
    return  new_manifest_file

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "manifest_file", help="The Amazon SageMaker Ground Truth manifest file"
        "that you want to use."
    )


def main():
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")
    try:
```

```
        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()
        # Create the manifest file
        manifest_file = create_manifest_file(args.manifest_file)
        print(f'Manifest file created: {manifest_file}')
    except FileNotFoundError as err:
        logger.exception('File not found: %s', err)
        print(f'File not found: {err}. Check your manifest file.')

if __name__ == "__main__":
    main()
```

2. Observe o nome do novo arquivo de manifesto exibido pelo script. Ele será usado na próxima etapa.
3. Faça upload dos arquivos de manifesto para o bucket do Amazon S3 que você deseja usar para armazenar o arquivo de manifesto.

> **Note**
>
> Certifique-se de que o Amazon Rekognition Custom Labels tenha acesso ao bucket do Amazon S3 referenciado no campo `source-ref` das linhas JSON do arquivo de manifesto. Para obter mais informações, consulte Como acessar os buckets externos do Amazon S3. Se seu trabalho do Ground Truth armazena imagens no bucket do console do Amazon Rekognition Custom Labels, você não precisa adicionar permissões.

4. Siga as instruções em Criação de um conjunto de dados com um arquivo de manifesto do SageMaker AI Ground Truth (console) para criar um conjunto de dados com o arquivo de manifesto carregado. Para a etapa 8, na localização do arquivo .manifest, insira a URL do Amazon S3 para a localização do arquivo de manifesto. Se estiver usando o AWS SDK, use Criação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth.

Como criar um arquivo de manifesto de um arquivo CSV

Este exemplo de script em Python simplifica a criação de um arquivo de manifesto usando um arquivo de valores separados por vírgulas (CSV) para rotular imagens. Crie o arquivo CSV. O arquivo de manifesto é adequado para a classificação de imagens com vários rótulos ou

Classificação de imagens com vários rótulos. Para obter mais informações, consulte Encontre objetos, cenas e conceitos.

> **Note**
>
> Este script não cria um arquivo de manifesto adequado para descobrir localizações de objetos ou para encontrar localizações de marcas.

Um arquivo de manifesto descreve as imagens usadas para treinar um modelo. Por exemplo, localizações de imagens e rótulos atribuídos às imagens. Um arquivo de manifesto é composto por uma ou mais linhas JSON. Cada linha JSON descreve uma única imagem. Para obter mais informações, consulte the section called “Importar rótulos ao nível da imagem em arquivos de manifesto”.

Um arquivo CSV representa dados tabulares em várias linhas em um arquivo de texto. Os campos em uma linha são separados por vírgulas. Para obter mais informações, consulte valores separados por vírgula. Para esse script, cada linha em seu arquivo CSV representa uma única imagem e mapeia para uma linha JSON no arquivo de manifesto. Para criar um arquivo CSV para um arquivo de manifesto que seja compatível com a classificação de imagens com vários rótulos, adicione um ou mais rótulos em nível de imagem a cada linha. Para criar um arquivo de manifesto adequado para Classificação de imagens, adicione um único rótulo em nível de imagem a cada linha.

Por exemplo, o arquivo CSV a seguir descreve as imagens no projeto Classificação de imagens com vários rótulos (Flores) Conceitos básicos.

```
camellia1.jpg,camellia,with_leaves
camellia2.jpg,camellia,with_leaves
camellia3.jpg,camellia,without_leaves
helleborus1.jpg,helleborus,without_leaves,not_fully_grown
helleborus2.jpg,helleborus,with_leaves,fully_grown
helleborus3.jpg,helleborus,with_leaves,fully_grown
jonquil1.jpg,jonquil,with_leaves
jonquil2.jpg,jonquil,with_leaves
jonquil3.jpg,jonquil,with_leaves
jonquil4.jpg,jonquil,without_leaves
mauve_honey_myrtle1.jpg,mauve_honey_myrtle,without_leaves
mauve_honey_myrtle2.jpg,mauve_honey_myrtle,with_leaves
mauve_honey_myrtle3.jpg,mauve_honey_myrtle,with_leaves
mediterranean_spurge1.jpg,mediterranean_spurge,with_leaves
```

```
mediterranean_spurge2.jpg,mediterranean_spurge,without_leaves
```

O script gera linhas JSON para cada linha. Por exemplo, a seguir está a linha JSON para a primeira linha (`camellia1.jpg,camellia,with_leaves`).

```
{"source-ref": "s3://bucket/flowers/train/camellia1.jpg","camellia": 1,"camellia-
metadata":{"confidence": 1,"job-name": "labeling-job/camellia","class-name":
 "camellia","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type":
 "groundtruth/image-classification"},"with_leaves": 1,"with_leaves-metadata":
{"confidence": 1,"job-name": "labeling-job/with_leaves","class-name":
 "with_leaves","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type":
 "groundtruth/image-classification"}}
```

No exemplo CSV, o caminho do Amazon S3 para a imagem não está presente. Se seu arquivo CSV não incluir o caminho do Amazon S3 para as imagens, use o argumento da linha de comando `--s3_path` para especificar o caminho do Amazon S3 para a imagem.

O script registra a primeira entrada de cada imagem em um arquivo CSV de imagem desduplicada. O arquivo CSV de imagem desduplicada contém uma única instância de cada imagem encontrada no arquivo CSV de entrada. Outras ocorrências de uma imagem no arquivo CSV de entrada são registradas em um arquivo CSV de imagem duplicado. Se o script encontrar imagens duplicadas, revise o arquivo CSV de imagem duplicada e atualize o arquivo CSV de imagem desduplicada conforme necessário. Execute novamente o script com o arquivo desduplicado. Se nenhuma duplicata for encontrada no arquivo CSV de entrada, o script excluirá o arquivo CSV de imagem desduplicada e a imagem duplicada, pois eles estão vazios. CSVfile

Neste procedimento, o arquivo CSV é criado e o script em Python é executado para criar o arquivo de manifesto.

Para criar um arquivo de manifesto de um arquivo CSV

1. Crie um arquivo CSV com os seguintes campos em cada linha (uma linha por imagem). Não adicione uma linha de cabeçalho ao arquivo CSV.

Campo 1	Campo 2	Campo n
O nome da imagem ou o caminho do Amazon S3 para a imagem. Por	O primeiro rótulo de nível de imagem para a imagem.	Um ou mais rótulos adicionais em nível de imagem separados por vírgulas.

Campo 1	Campo 2	Campo n
exemplo, .`s3://my-bucket/flowers/train/camellia1.jpg` Não é possível ter uma mistura de imagens com o caminho do Amazon S3 e imagens sem ele.		Adicione somente se quiser criar um arquivo de manifesto que seja compatível com a classificação de imagens com vários rótulos.

Por exemplo, `camellia1.jpg,camellia,with_leaves` ou `s3://my-bucket/flowers/train/camellia1.jpg,camellia,with_leaves`

2. Salve o arquivo CSV.
3. Execute o seguinte script em Python. Forneça os seguintes argumentos:
   - `csv_file`: o arquivo CSV que você criou na etapa 1.
   - `manifest_file`: o nome do arquivo de manifesto que você deseja criar.
   - (Opcional) `--s3_path` *`s3://path_to_folder/`*: o caminho do Amazon S3 a ser adicionado aos nomes dos arquivos de imagem (campo 1). Use `--s3_path` se as imagens no campo 1 ainda não contiverem um caminho do S3.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier:  Apache-2.0

from datetime import datetime, timezone
import argparse
import logging
import csv
import os
import json

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service documentation.
Shows how to create an image-level (classification) manifest file from a CSV file.
You can specify multiple image level labels per image.
CSV file format is
image,label,label,..
```

```
If necessary, use the bucket argument to specify the S3 bucket folder for the
 images.
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-gt-cl-
transform.html
"""

logger = logging.getLogger(__name__)


def check_duplicates(csv_file, deduplicated_file, duplicates_file):
    """
    Checks for duplicate images in a CSV file. If duplicate images
    are found, deduplicated_file is the deduplicated CSV file - only the first
    occurence of a duplicate is recorded. Other duplicates are recorded in
 duplicates_file.
    :param csv_file: The source CSV file.
    :param deduplicated_file: The deduplicated CSV file to create. If no duplicates
 are found
    this file is removed.
    :param duplicates_file: The duplicate images CSV file to create. If no
 duplicates are found
    this file is removed.
    :return: True if duplicates are found, otherwise false.
    """

    logger.info("Deduplicating %s", csv_file)

    duplicates_found = False

    # Find duplicates.
    with open(csv_file, 'r', newline='', encoding="UTF-8") as f,\
            open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
            open(duplicates_file, 'w', encoding="UTF-8") as duplicates:

        reader = csv.reader(f, delimiter=',')
        dedup_writer = csv.writer(dedup)
        duplicates_writer = csv.writer(duplicates)

        entries = set()
        for row in reader:
            # Skip empty lines.
            if not ''.join(row).strip():
                continue
```

```
            key = row[0]
            if key not in entries:
                dedup_writer.writerow(row)
                entries.add(key)
            else:
                duplicates_writer.writerow(row)
                duplicates_found = True

    if duplicates_found:
        logger.info("Duplicates found check %s", duplicates_file)

    else:
        os.remove(duplicates_file)
        os.remove(deduplicated_file)

    return duplicates_found


def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Reads a CSV file and creates a Custom Labels classification manifest file.
    :param csv_file: The source CSV file.
    :param manifest_file: The name of the manifest file to create.
    :param s3_path: The S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s", csv_file)

    image_count = 0
    label_count = 0

    with open(csv_file, newline='', encoding="UTF-8") as csvfile,\
            open(manifest_file, "w", encoding="UTF-8") as output_file:

        image_classifications = csv.reader(
            csvfile, delimiter=',', quotechar='|')

        # Process each row (image) in CSV file.
        for row in image_classifications:
            source_ref = str(s3_path)+row[0]

            image_count += 1

            # Create JSON for image source ref.
            json_line = {}
```

```
            json_line['source-ref'] = source_ref

            # Process each image level label.
            for index in range(1, len(row)):
                image_level_label = row[index]

                # Skip empty columns.
                if image_level_label == '':
                    continue
                label_count += 1

               # Create the JSON line metadata.
                json_line[image_level_label] = 1
                metadata = {}
                metadata['confidence'] = 1
                metadata['job-name'] = 'labeling-job/' + image_level_label
                metadata['class-name'] = image_level_label
                metadata['human-annotated'] = "yes"
                metadata['creation-date'] = \
                    datetime.now(timezone.utc).strftime('%Y-%m-%dT%H:%M:%S.%f')
                metadata['type'] = "groundtruth/image-classification"

                json_line[f'{image_level_label}-metadata'] = metadata

                # Write the image JSON Line.
            output_file.write(json.dumps(json_line))
            output_file.write('\n')

    output_file.close()
    logger.info("Finished creating manifest file %s\nImages: %s\nLabels: %s",
                manifest_file, image_count, label_count)

    return image_count, label_count


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "csv_file", help="The CSV file that you want to process."
    )
```

```
    parser.add_argument(
        "--s3_path", help="The S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the S3 path.",
 required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        s3_path = args.s3_path
        if s3_path is None:
            s3_path = ''

        # Create file names.
        csv_file = args.csv_file
        file_name = os.path.splitext(csv_file)[0]
        manifest_file = f'{file_name}.manifest'
        duplicates_file = f'{file_name}-duplicates.csv'
        deduplicated_file = f'{file_name}-deduplicated.csv'

        # Create manifest file, if there are no duplicate images.
        if check_duplicates(csv_file, deduplicated_file, duplicates_file):
            print(f"Duplicates found. Use {duplicates_file} to view duplicates "
                  f"and then update {deduplicated_file}. ")
            print(f"{deduplicated_file} contains the first occurence of a
 duplicate. "
                  "Update as necessary with the correct label information.")
            print(f"Re-run the script with {deduplicated_file}")
        else:
            print("No duplicates found. Creating manifest file.")

            image_count, label_count = create_manifest_file(csv_file,
                                                            manifest_file,
```

```
                                                                s3_path)

            print(f"Finished creating manifest file: {manifest_file} \n"
                  f"Images: {image_count}\nLabels: {label_count}")

    except FileNotFoundError as err:
        logger.exception("File not found: %s", err)
        print(f"File not found: {err}. Check your input CSV file.")


if __name__ == "__main__":
    main()
```

4. Se planeja usar um conjunto de dados de teste, repita as etapas de 1 a 3 para criar um arquivo de manifesto para seu conjunto de dados de teste.
5. Se necessário, copie as imagens para o caminho do bucket do Amazon S3 que você especificou na coluna 1 do arquivo CSV (ou especificado na linha de comando `--s3_path`). É possível usar o seguinte comando AWS do S3.

```
aws s3 cp --recursive your-local-folder s3://your-target-S3-location
```

6. Faça upload dos arquivos de manifesto para o bucket do Amazon S3 que você deseja usar para armazenar o arquivo de manifesto.

> **Note**
>
> Certifique-se de que o Amazon Rekognition Custom Labels tenha acesso ao bucket do Amazon S3 referenciado no campo `source-ref` das linhas JSON do arquivo de manifesto. Para obter mais informações, consulte Como acessar os buckets externos do Amazon S3. Se seu trabalho do Ground Truth armazena imagens no bucket do console do Amazon Rekognition Custom Labels, você não precisa adicionar permissões.

7. Siga as instruções em Criação de um conjunto de dados com um arquivo de manifesto do SageMaker AI Ground Truth (console) para criar um conjunto de dados com o arquivo de manifesto carregado. Para a etapa 8, na localização do arquivo .manifest, insira a URL do Amazon S3 para a localização do arquivo de manifesto. Se estiver usando o AWS SDK, use Criação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth.

Copiar conteúdo de um conjunto de dados existente

Se já criou um conjunto de dados, pode copiar seu conteúdo para um novo conjunto de dados. Para criar um conjunto de dados a partir de um conjunto de dados existente com o AWS SDK, consulte. Como criar um conjunto de dados usando um conjunto de dados existente (SDK)

Para criar um conjunto de dados usando um conjunto de dados personalizados do Amazon Rekognition Custom Labels (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, selecione o projeto ao qual você deseja adicionar um conjunto de dados. A página de detalhes do seu projeto é exibida.
6. Escolha Criar conjunto de dados. A página Criar conjunto de dados é exibida.
7. Em Configuração inicial, escolha Iniciar com um único conjunto de dados ou Iniciar com um conjunto de dados de treinamento. Para criar um modelo de maior qualidade, recomendamos começar com conjuntos de dados de treinamento e teste separados.

   Single dataset

   a. Na seção Detalhes do conjunto de dados de treinamento, escolha Copiar um conjunto de dados existente do Amazon Rekognition Custom Labels.
   b. Na seção Detalhes do conjunto de dados de treinamento, na caixa de edição Conjunto de dados, digite ou selecione o nome do conjunto de dados que você deseja copiar.
   c. Escolha Criar conjunto de dados. A página de conjuntos de dados do seu projeto é aberta.

   Separate training and test datasets

   a. Na seção Detalhes do conjunto de dados de treinamento, escolha Copiar um conjunto de dados existente do Amazon Rekognition Custom Labels.
   b. Na seção Detalhes do conjunto de dados de treinamento, na caixa de edição Conjunto de dados, digite ou selecione o nome do conjunto de dados que você deseja copiar.
   c. Na seção Detalhes do conjunto de dados de teste, escolha Copiar um conjunto de dados existente do Amazon Rekognition Custom Labels.

d. Na seção Detalhes do conjunto de dados de teste, na caixa de edição Conjunto de dados, digite ou selecione o nome do conjunto de dados que você deseja copiar.

> **Note**
>
> Seus conjuntos de dados de treinamento e teste podem ter fontes de imagem diferentes.

e. Escolha Criar conjuntos de dados. A página de conjuntos de dados do seu projeto é aberta.

8. Se precisar adicionar ou alterar rótulos, faça Rotulagem de imagens.
9. Siga as etapas em Como treinar um modelo (console) para treinar seu modelo.

Rotulagem de imagens

Um rótulo identifica um objeto, cena, conceito ou caixa delimitadora ao redor de um objeto em uma imagem. Por exemplo, se seu conjunto de dados contiver imagens de cães, será possível adicionar rótulos para raças de cães.

Depois de importar suas imagens para um conjunto de dados, talvez seja necessário adicionar rótulos às imagens ou corrigir imagens com rótulos incorretos. Por exemplo, imagens não estão rotuladas se forem importadas de um computador local. A galeria de conjuntos de dados é usada para adicionar novos rótulos ao conjunto de dados e atribuir rótulos e caixas delimitadoras às imagens no conjunto de dados.

A forma como as imagens são rotuladas nos conjuntos de dados determina o tipo de modelo que o Amazon Rekognition Custom Labels treina. Para obter mais informações, consulte Como definir os conjuntos de dados.

Tópicos

- Como gerenciar rótulos
- Como atribuir rótulos em nível de imagem em uma imagem
- Como rotular objetos com caixas delimitadoras

Como gerenciar rótulos

É possível gerenciar rótulos usando o console do Amazon Rekognition Custom Labels. Não há uma API específica para gerenciar rótulos. Os rótulos são adicionados ao conjunto de dados quando você cria o conjunto de dados com `CreateDataset` ou quando você adiciona mais imagens ao conjunto de dados com `UpdateDatasetEntries`.

Tópicos

- Como gerenciar rótulos (console)
- Como gerenciar rótulos (SDK)

Como gerenciar rótulos (console)

É possível usar o console do Amazon Rekognition Custom Labels para adicionar, alterar ou remover rótulos de um conjunto de dados. Para adicionar um rótulo a um conjunto de dados, é possível adicionar um novo rótulo que você cria ou importar rótulos de um conjunto de dados existente no Rekognition.

Tópicos

- Adicionar novos rótulos (console)
- Alterar e remover rótulos (console)

Adicionar novos rótulos (console)

É possível especificar novos rótulos que você deseja adicionar ao conjunto de dados.

Adicione rótulos usando a janela de edição

Para adicionar um novo rótulo (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, escolha o projeto que deseja usar. A página de detalhes do seu projeto é exibida.

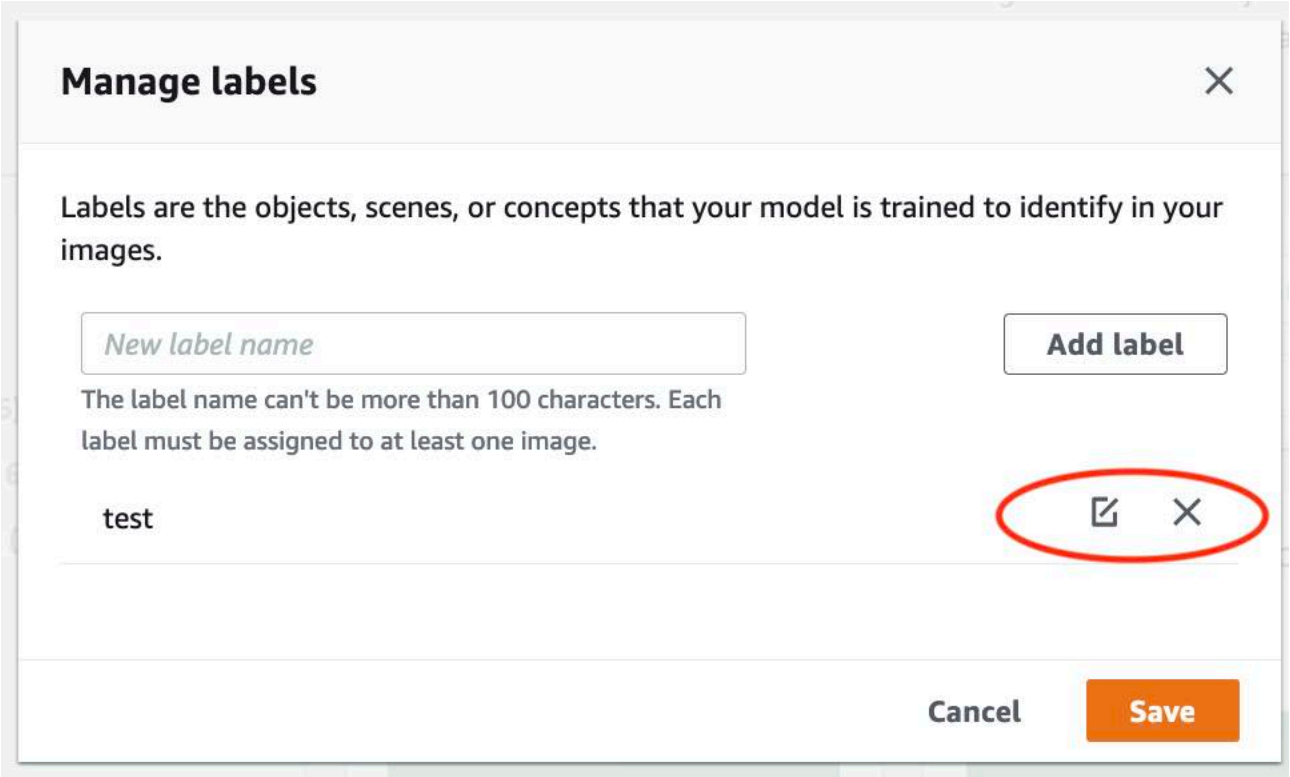
6. Se quiser adicionar rótulos ao seu conjunto de dados de treinamento, escolha a guia Treinamento. Caso contrário, escolha a guia Teste para adicionar rótulos ao conjunto de dados de teste.
7. Escolha Iniciar rotulagem para entrar no modo de rotulagem.
8. Na seção Rótulos da galeria do conjunto de dados, escolha Gerenciar rótulos para abrir a caixa de diálogo Gerenciar rótulos.
9. Na caixa de edição, insira um novo nome de rótulo.
10. Selecione Adicionar novo rótulo.
11. Repita as etapas 9 e 10 até criar todos os rótulos necessários.
12. Escolha Salvar para salvar os rótulos que você adicionou.

Alterar e remover rótulos (console)

É possível renomear ou remover rótulos depois de adicioná-los a um conjunto de dados. Só é possível remover rótulos que não estejam atribuídos a nenhuma imagem.

Para renomear ou remover um rótulo existente (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, escolha o projeto que deseja usar. A página de detalhes do seu projeto é exibida.
6. Se quiser alterar ou excluir rótulos no seu conjunto de dados de treinamento, escolha a guia Treinamento. Caso contrário, escolha a guia Teste para alterar ou excluir rótulos ao conjunto de dados de teste.
7. Escolha Iniciar rotulagem para entrar no modo de rotulagem.
8. Na seção Rótulos da galeria do conjunto de dados, escolha Gerenciar rótulos para abrir a caixa de diálogo Gerenciar rótulos.
9. Escolha o rótulo que você deseja editar ou excluir.



a. Se escolher o ícone de exclusão (X), o rótulo será removido da lista.

b. Se quiser alterar o rótulo, escolha o ícone de edição (lápis e bloco de papel) e insira um novo nome de rótulo na caixa de edição.

10. Escolha Salvar para salvar as alterações.

Como gerenciar rótulos (SDK)

Não há uma API exclusiva que gerencie rótulos de conjuntos de dados. Se criar um conjunto de dados com `CreateDataset`, os rótulos encontrados no arquivo de manifesto ou no conjunto de dados copiado, crie o conjunto inicial de rótulos. Se adicionar mais imagens com a API `UpdateDatasetEntries`, novos rótulos encontrados nas entradas serão adicionados ao conjunto de dados. Para obter mais informações, consulte Como adicionar mais imagens (SDK). Para excluir rótulos de um conjunto de dados, você deve remover todas as anotações de rótulos no conjunto de dados.

Para excluir rótulos de um conjunto de dados

1. Chame `ListDatasetEntries` para obter as entradas do conjunto de dados. Para obter um código de exemplo, consulte Como listar entradas do conjunto de dados (SDK).

2. No arquivo, remova todas as anotações do rótulo. Para obter mais informações, consulte Importar rótulos ao nível da imagem em arquivos de manifesto e the section called “Localização de objetos em arquivos de manifesto”.
3. Use o arquivo para atualizar o conjunto de dados com a API `UpdateDatasetEntries`. Para obter mais informações, consulte Como adicionar mais imagens (SDK).

Como atribuir rótulos em nível de imagem em uma imagem

Os rótulos em nível de imagem são usados para treinar modelos que classificam imagens em categorias. Um rótulo no nível da imagem indica que uma imagem contém um objeto, cena ou conceito. Por exemplo, a imagem a seguir mostra um rio. Se seu modelo classificar as imagens como contendo rios, você adicionaria um rótulo no nível da imagem de rio. Para obter mais informações, consulte Como definir os conjuntos de dados.

Um conjunto de dados que contém rótulos em nível de imagem precisa de pelo menos dois rótulos definidos. Cada imagem precisa de pelo menos um rótulo atribuído que identifique o objeto, a cena ou o conceito na imagem.

Para atribuir rótulos em nível de imagem em uma imagem (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, escolha o projeto que deseja usar. A página de detalhes do seu projeto é exibida.
6. No painel de navegação esquerdo, selecione Conjunto de dados.
7. Se quiser adicionar rótulos ao seu conjunto de dados de treinamento, escolha a guia Treinamento. Caso contrário, escolha a guia Teste para adicionar rótulos ao conjunto de dados de teste.
8. Escolha Iniciar rotulagem para entrar no modo de rotulagem.
9. Na galeria de imagens, selecione uma ou mais imagens às quais você deseja adicionar rótulos. Só é possível selecionar imagens em uma única página de cada vez. Para selecionar uma faixa contígua de imagens em uma página:
    a. Selecione a primeira imagem no intervalo.
    b. Pressione e segure a tecla shift.
    c. Selecione o último intervalo de imagens. As imagens entre a primeira e a segunda imagem também são selecionadas.
    d. Solte a tecla shift.
10. Escolha Atribuir rótulos em nível de imagem.
11. Na caixa de diálogo Atribuir um rótulo em nível de imagem às imagens selecionadas, selecione um rótulo que deseja atribuir à imagem ou imagens.
12. Escolha Atribuir para atribuir um rótulo à imagem.
13. Repita a rotulagem até que cada imagem seja anotada com os rótulos necessários.
14. Escolha Salvar alterações para salvar suas alterações.

Atribuir rótulos em nível de imagem (SDK)

É possível usar a API `UpdateDatasetEntries` para adicionar ou atualizar os rótulos em nível de imagem atribuídos a uma imagem. `UpdateDatasetEntries` usa uma ou mais linhas JSON. Cada linha JSON representa uma única imagem. Para uma imagem com um rótulo em nível de imagem, a linha JSON é semelhante à seguinte.

```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/
manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-metadata":
{"confidence":0.95,"job-name":"labeling-job/testclconsolebucket","class-name":"Echo
 Dot","human-annotated":"yes","creation-
date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

O campo `source-ref` indica a localização da imagem. A linha JSON também inclui os rótulos em nível de imagem atribuídos à imagem. Para obter mais informações, consulte the section called "Importar rótulos ao nível da imagem em arquivos de manifesto".

Para atribuir rótulos em nível de imagem em uma imagem

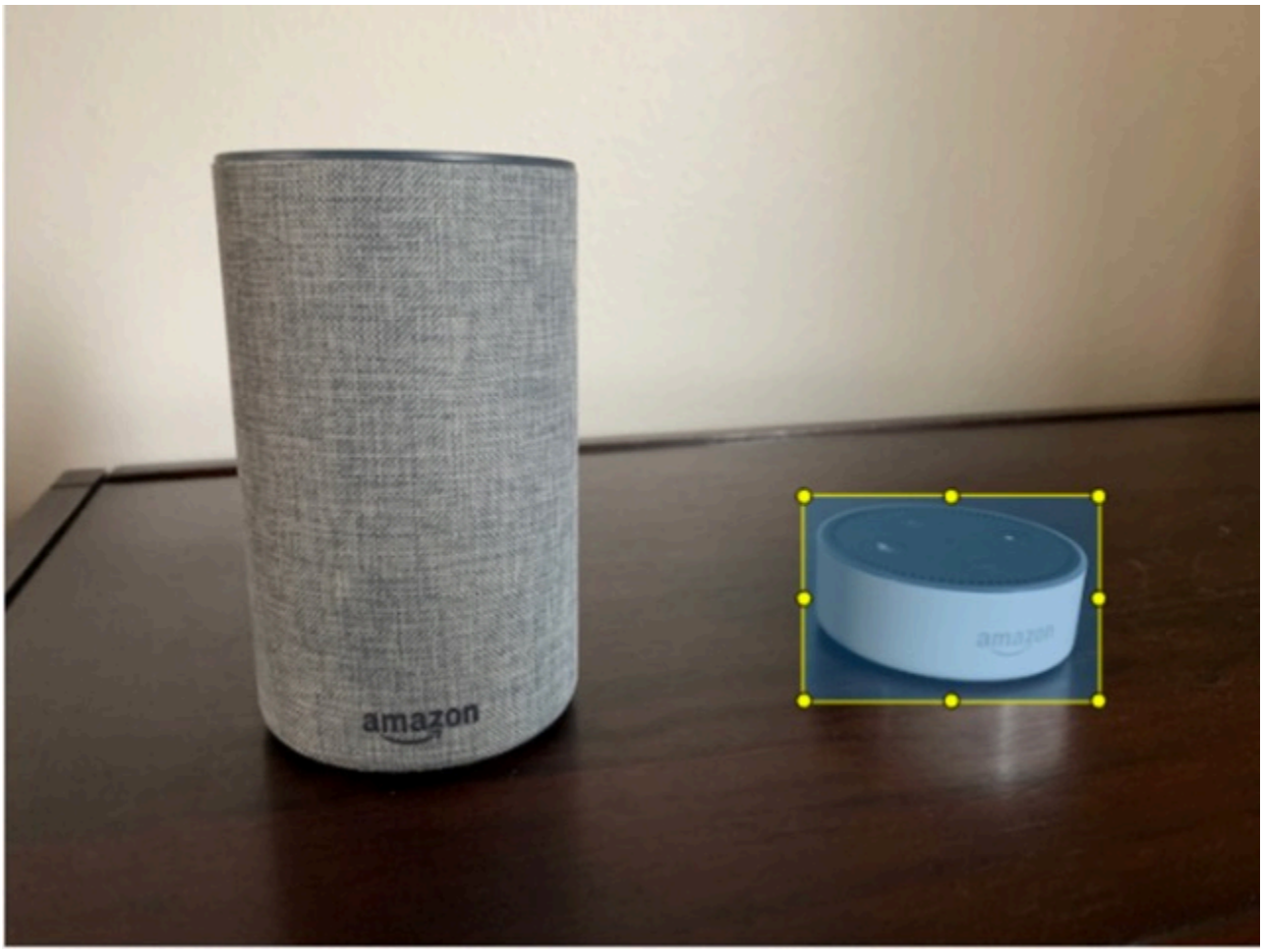
1. Obtenha a linha get JSON para a imagem existente usando `ListDatasetEntries`. Para o campo `source-ref`, especifique a localização da imagem à qual você deseja atribuir o rótulo. Para obter mais informações, consulte Como listar entradas do conjunto de dados (SDK).
2. Atualize a linha JSON retornada na etapa anterior usando as informações em Importar rótulos ao nível da imagem em arquivos de manifesto.
3. Chame `UpdateDatasetEntries` para atualizar a imagem. Para obter mais informações, consulte Como adicionar mais imagens a um conjunto de dados.

Como rotular objetos com caixas delimitadoras

Se quiser que seu modelo detecte a localização de objetos em uma imagem, você deve identificar o que é o objeto e onde ele está na imagem. Uma caixa delimitadora é uma caixa que isola um objeto em uma imagem. As caixas delimitadoras são usadas para treinar um modelo para detectar objetos diferentes na mesma imagem. Identifique os objetos atribuindo rótulos à caixa delimitadora.

> **Note**
>
> Se estiver treinando um modelo para encontrar objetos, cenas e conceitos com rótulos em nível de imagem, não precisará executar essa etapa.

Por exemplo, se você quiser treinar um modelo que detecte dispositivos Amazon Echo Dot, desenhe uma caixa delimitadora ao redor de cada Echo Dot em uma imagem e atribua um rótulo chamado Echo Dot à caixa delimitadora. A imagem a seguir mostra uma caixa delimitadora ao redor de um dispositivo Echo Dot. A imagem também contém um Amazon Echo sem uma caixa delimitadora.



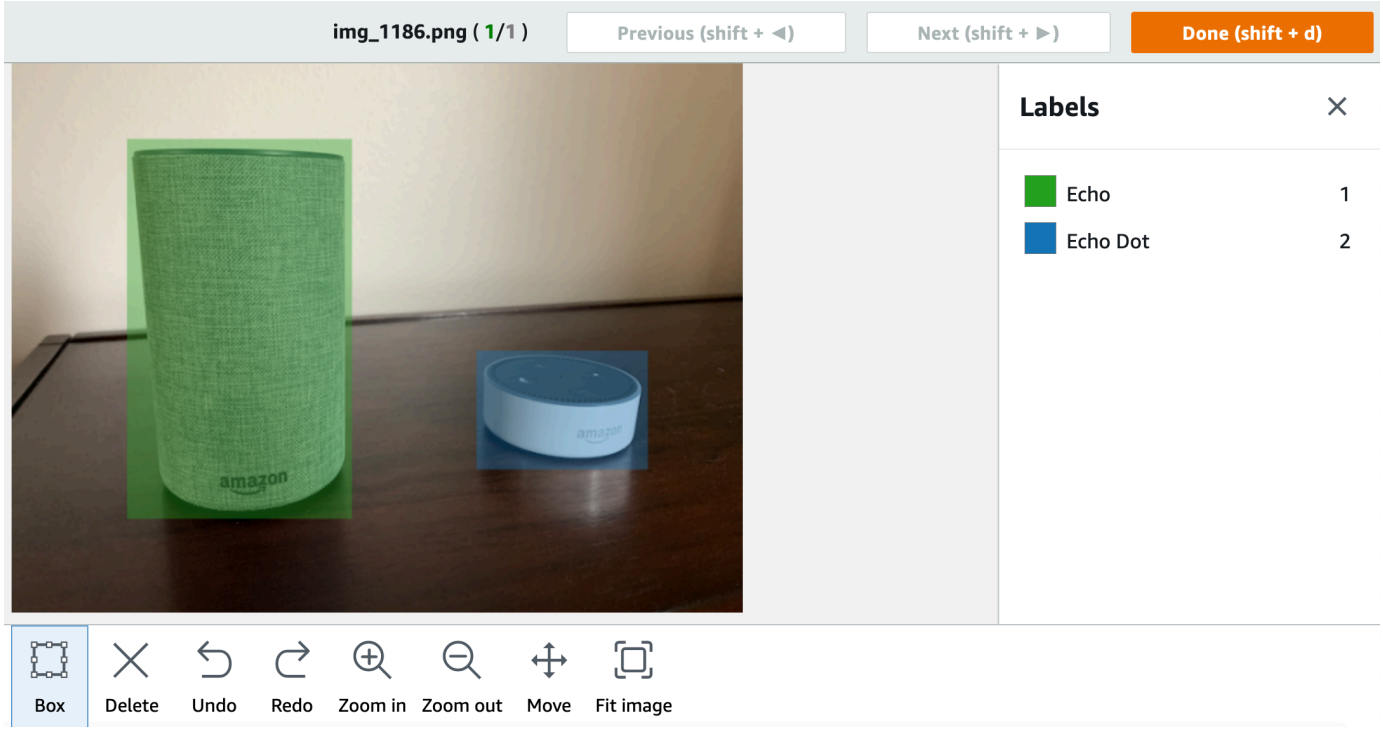
Localize objetos com caixas delimitadoras (console)

Neste procedimento, é usado o console para desenhar caixas delimitadoras ao redor dos objetos em suas imagens. Também é possível identificar objetos na imagem atribuindo rótulos à caixa delimitadora.

> **Note**
>
> Não é possível usar o navegador Safari para adicionar caixas delimitadoras às imagens. Para ver os navegadores compatíveis, consulte Como configurar o Amazon Rekognition Custom Labels.

Antes de adicionar caixas delimitadoras, você deve adicionar pelo menos um rótulo ao conjunto de dados. Para obter mais informações, consulte Adicionar novos rótulos (console).

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, escolha o projeto que deseja usar. A página de detalhes do seu projeto é exibida.
6. Na página de detalhes do projeto, escolha Rotular imagens
7. Se quiser adicionar caixas delimitadoras às imagens do seu conjunto de dados de treinamento, escolha a guia Treinamento. Caso contrário, escolha a guia Teste para adicionar caixas delimitadoras às imagens do conjunto de dados de teste.
8. Escolha Iniciar rotulagem para entrar no modo de rotulagem.
9. Na galeria de imagens, escolha as imagens às quais você deseja adicionar caixas delimitadoras.
10. Escolha Desenhar caixa delimitadora. Uma série de dicas é mostrada antes que o editor da caixa delimitadora seja exibido.
11. No painel Rótulos à direita, selecione o rótulo que deseja atribuir a uma caixa delimitadora.
12. Na ferramenta de desenho, posicione o ponteiro na área superior esquerda do objeto desejado.
13. Pressione o botão esquerdo do mouse e desenhe uma caixa ao redor do objeto. Tente desenhar a caixa delimitadora o mais próximo possível do objeto.
14. Solte o botão do mouse. A caixa delimitadora é destacada.
15. Escolha Próximo se você tiver mais imagens para rotular. Caso contrário, escolha Concluído para finalizar a rotulagem.



16. Repita as etapas de 1 a 7 até criar uma caixa delimitadora em cada imagem que contém objetos.
17. Escolha Salvar alterações para salvar suas alterações.
18. Escolha Sair para sair do modo de rotulagem.

Localize objetos com caixas delimitadoras (SDK)

É possível usar a API `UpdateDatasetEntries` para adicionar ou atualizar as informações de localização do objeto para uma imagem. `UpdateDatasetEntries` usa uma ou mais linhas JSON. Cada linha JSON representa uma única imagem. Para localização de objetos, uma linha JSON terá uma aparência semelhante à seguinte.

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size":
 [{"width": 640, "height": 480, "depth": 3}], "annotations": [{ "class_id": 1,
 "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0, "top": 65,
 "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata": {"objects":
 [{ "confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo", "1": "Echo Dot"},
 "type": "groundtruth/object-detection", "human-annotated": "yes", "creation-date":
 "2013-11-18T02:53:27", "job-name": "my job"}}
```

O campo `source-ref` indica a localização da imagem. A linha JSON também inclui caixas delimitadoras rotuladas para cada objeto na imagem. Para obter mais informações, consulte the section called "Localização de objetos em arquivos de manifesto".

Para atribuir caixas delimitadoras a uma imagem

1. Obtenha a linha get JSON para a imagem existente usando `ListDatasetEntries`. Para o campo `source-ref`, especifique a localização da imagem à qual você deseja atribuir o rótulo no nível da imagem. Para obter mais informações, consulte Como listar entradas do conjunto de dados (SDK).
2. Atualize a linha JSON retornada na etapa anterior usando as informações em Localização de objetos em arquivos de manifesto.
3. Chame `UpdateDatasetEntries` para atualizar a imagem. Para obter mais informações, consulte Como adicionar mais imagens a um conjunto de dados.

Como depurar conjuntos de dados

Durante a criação do conjunto de dados, podem ocorrer dois tipos de erro: erros terminais e erros não terminais. Erros terminais não impedem a criação ou a atualização do conjunto de dados. Erros não terminais não impedem a criação ou a atualização do conjunto de dados.

Tópicos

- Depuração de erros terminais do conjunto de dados
- Depuração de erros não terminais do conjunto de dados

Depuração de erros terminais do conjunto de dados

Há dois tipos de erros de terminal: erros de arquivo que causam falha na criação do conjunto de dados e erros de conteúdo que o Amazon Rekognition Custom Labels remove do conjunto de dados. A criação do conjunto de dados falhará se houver muitos erros de conteúdo.

Tópicos

- Erros terminais no arquivo
- Erros terminais de conteúdo

Erros terminais no arquivo

A seguir estão os erros de arquivo. É possível obter informações sobre erros de arquivo chamando `DescribeDataset` e verificando os campos `Status` e `StatusMessage`. Para obter um código de exemplo, consulte Como descrever um conjunto de dados (SDK).

- ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT
- ERROR_MANIFEST_SIZE_TOO_LARGE.
- ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM
- ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET
- ERROR_TOO_MANY_RECORDS_IN_ERROR
- ERROR_MANIFEST_TOO_MANY_LABELS
- ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

Mensagem de erro

A extensão ou o conteúdo do arquivo de manifesto são inválidos.

O arquivo de manifesto de treinamento ou teste não tem uma extensão de arquivo ou seu conteúdo é inválido.

Para corrigir o erro ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

- Verifique as seguintes possíveis causas nos arquivos de manifesto de treinamento e teste.
  - O arquivo de manifesto não tem uma extensão. Por convenção, a extensão do arquivo é `.manifest`.
  - Não foi possível encontrar o bucket ou a chave do Amazon S3 para o arquivo de manifesto.

ERROR_MANIFEST_SIZE_TOO_LARGE

Mensagem de erro

O tamanho do arquivo de manifesto excede o tamanho máximo permitido.

O tamanho do arquivo do manifesto de treinamento ou teste (em bytes) é muito grande. Para obter mais informações, consulte Diretrizes e cotas no Amazon Rekognition Custom Labels. Um arquivo

de manifesto pode ter menos do que o número máximo de linhas JSON e ainda exceder o tamanho máximo do arquivo.

Não é possível pode usar o console do Amazon Rekognition Custom Labels para corrigir o erro O tamanho do arquivo de manifesto excede o tamanho máximo permitido.

Para corrigir o erro ERROR_MANIFEST_SIZE_TOO_LARGE

1. Verifique quais manifestos de treinamento e teste excedem o tamanho máximo do arquivo.
2. Reduza o número de linhas JSON nos arquivos de manifesto que são muito grandes. Para obter mais informações, consulte Criar um arquivo de manifesto.

ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM

Mensagem de erro

O arquivo de manifesto tem muitas linhas.

Mais informações

O número de linhas JSON (número de imagens) no arquivo de manifesto é maior que o limite permitido. O limite é diferente para modelos em nível de imagem e modelos de localização de objetos. Para obter mais informações, consulte Diretrizes e cotas no Amazon Rekognition Custom Labels.

Os erros de linha JSON são validados até que o número de linhas JSON atinja o limite de `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir o erro `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`.

Para corrigir **`ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`**

- Reduza o número de linhas JSON no manifesto. Para obter mais informações, consulte Criar um arquivo de manifesto.

ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

Mensagem de erro

As permissões de bucket do S3 estão incorretas.

O Amazon Rekognition Custom Labels não tem permissões para um ou mais buckets que contêm os arquivos de manifesto de treinamento e teste.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Para corrigir o erro ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

- Verifique as permissões dos buckets contendo os manifestos de treinamento e teste. Para obter mais informações, consulte Etapa 2: configure as permissões do console do Amazon Rekognition Custom Labels.

ERROR_TOO_MANY_RECORDS_IN_ERROR

Mensagem de erro

O arquivo de manifesto tem muitos erros terminais.

Para corrigir **`ERROR_TOO_MANY_RECORDS_IN_ERROR`**

- Reduza o número de linhas JSON (imagens) com erros terminais de conteúdo. Para obter mais informações, consulte Erros terminais de conteúdo do manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_MANIFEST_TOO_MANY_LABELS

Mensagem de erro

O arquivo de manifesto tem muitos rótulos.

Mais informações

O número de rótulos exclusivos no manifesto (conjunto de dados) é maior do que o limite permitido. Se o conjunto de dados de treinamento for dividido para criar um conjunto de dados de teste, o número de rótulos será determinado após a divisão.

Para corrigir ERROR_MANIFEST_TOO_MANY_LABELS (console)

- Remova os rótulos do conjunto de dados. Para obter mais informações, consulte Como gerenciar rótulos. Os rótulos são removidos automaticamente das imagens e das caixas delimitadoras em seu conjunto de dados.

Para corrigir ERROR_MANIFEST_TOO_MANY_LABELS (linha JSON)

- Manifestos com linhas JSON em nível de imagem: se a imagem tiver um único rótulo, remova a linha JSON das imagens que usa o rótulo desejado. Se a linha JSON contiver vários rótulos, remova somente o objeto JSON do rótulo desejado. Para obter mais informações, consulte Como adicionar vários rótulos em nível de imagem a uma imagem.

  Manifestos com linhas JSON de localização do objeto: remova a caixa delimitadora e as informações de rótulo associadas ao rótulo que você deseja remover. Faça isso para cada linha JSON que contém o rótulo desejado. É necessário remover o rótulo da matriz `class-map` e os objetos correspondentes na matriz `objects` e `annotations`. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

Mensagem de erro

O arquivo de manifesto não tem imagens rotuladas suficientes para distribuir o conjunto de dados.

A distribuição do conjunto de dados ocorre quando o Amazon Rekognition Custom Labels divide um conjunto de dados de treinamento para criar um conjunto de dados de teste. Também é possível dividir um conjunto de dados chamando a API `DistributeDatasetEntries`.

Para corrigir o erro ERROR_MANIFEST_TOO_MANY_LABELS

- Adicione mais imagens rotuladas ao conjunto de dados de treinamento.

Erros terminais de conteúdo

A seguir estão os erros terminais de conteúdo. Durante a criação do conjunto de dados, as imagens com erros terminais de conteúdo são removidas do conjunto de dados. O conjunto de dados ainda pode ser usado para treinamento. Se houver muitos erros de conteúdo, a criação do conjunto de dados falhará. Os erros terminais de conteúdo relacionados às operações do conjunto de dados não são exibidos no console nem retornados de `DescribeDataset` ou outra API. Se perceber que imagens ou anotações estão faltando em seus conjuntos de dados, verifique os seguintes problemas nos arquivos de manifesto do conjunto de dados:

- O comprimento de uma linha JSON é muito longo. O tamanho máximo é de 100 mil caracteres.

- O valor `source-ref` está ausente em uma linha JSON.
- O formato de um valor `source-ref` em uma linha JSON é inválido.
- O conteúdo de uma linha JSON não é válido.
- O valor de um campo `source-ref` aparece mais de uma vez. Uma imagem só pode ser referenciada uma vez em um conjunto de dados.

Para obter informações sobre o campo `source-ref`, consulte Criar um arquivo de manifesto.

Depuração de erros não terminais do conjunto de dados

A seguir estão os erros não terminais que podem ocorrer durante a criação ou atualização do conjunto de dados. Estes erros podem invalidar uma linha JSON inteira ou invalidar anotações em uma linha JSON. Se uma linha JSON tiver um erro, ela não será usada para treinamento. Se uma anotação em uma linha JSON tiver um erro, a linha JSON ainda será usada para treinamento, mas sem a anotação quebrada. Para obter mais informações sobre linhas JSON, consulte Criar um arquivo de manifesto.

É possível acessar erros não terminais do console e chamando a API `ListDatasetEntries`. Para obter mais informações, consulte Como listar entradas do conjunto de dados (SDK).

Os seguintes erros também são retornados durante o treinamento. É recomendável a correção desses erros antes de treinar seu modelo. Para obter mais informações, consulte Erros não terminais de validação de linha JSON.

- ERROR_NO_LABEL_ATTRIBUTES
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT
- ERROR_NO_VALID_LABEL_ATTRIBUTES
- ERROR_INVALID_BOUNDING_BOX
- ERROR_INVALID_IMAGE_DIMENSION
- ERROR_BOUNDING_BOX_TOO_SMALL
- ERROR_NO_VALID_ANNOTATIONS
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_TOO_MANY_BOUNDING_BOXES

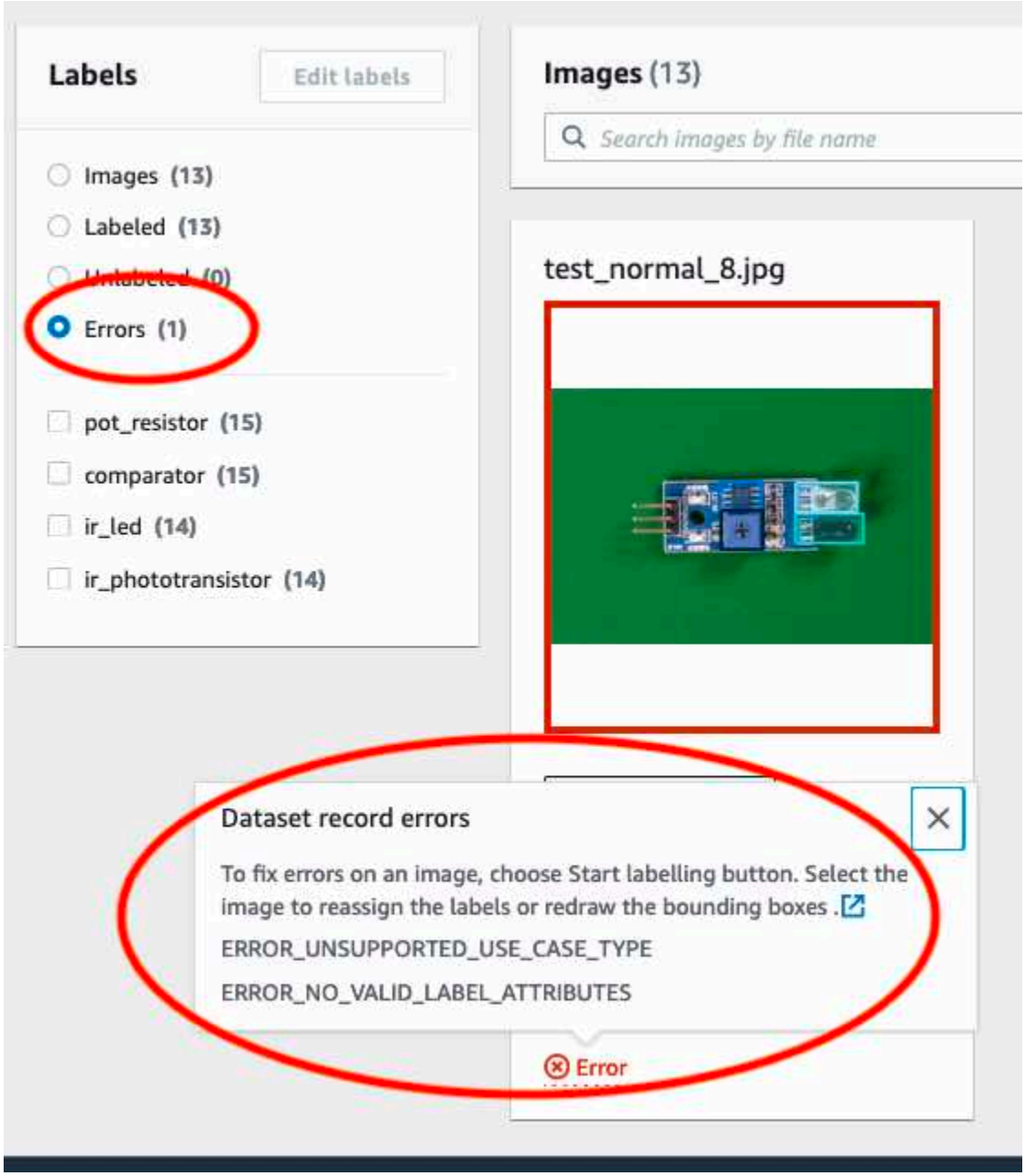
- ERROR_UNSUPPORTED_USE_CASE_TYPE
- ERROR_INVALID_LABEL_NAME_LENGTH

Como acessar erros não terminais

É possível usar o console para descobrir quais imagens em um conjunto de dados não têm erros terminais. Também é possível chamar, chamar a API `ListDatasetEntries` para receber as mensagens de erro. Para obter mais informações, consulte Como listar entradas do conjunto de dados (SDK).

Para acessar erros não terminais (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, escolha o projeto que deseja usar. A página de detalhes do seu projeto é exibida.
6. Se quiser visualizar erros não terminais em seu conjunto de dados de treinamento, escolha a guia Treinamento. Caso contrário, escolha a guia Teste para visualizar erros não terminais em seu conjunto de dados de teste.
7. Na seção Rótulos da galeria do conjunto de dados, escolha Erros. A galeria do conjunto de dados é filtrada para mostrar somente imagens com erros.
8. Escolha Erro abaixo de uma imagem para ver o código do erro. Use as informações em Erros não terminais de validação de linha JSON para corrigir o erro.



Como treinar um modelo do Amazon Rekognition Custom Labels

É possível treinar um modelo usando o console do Amazon Rekognition Custom Labels ou pela API Amazon Rekognition Custom Labels. Se o treinamento do modelo falhar, use as informações em Como depurar um treinamento de modelo em falha para encontrar a causa da falha.

> **Note**
>
> Há uma cobrança pelo tempo necessário para treinar um modelo com êxito. Normalmente, o treinamento leva de 30 minutos a 24 horas para ser concluído. Para obter mais informações, consulte Horas de treinamento.

Uma nova versão de um modelo é criada toda vez que ele é treinado. O Amazon Rekognition Custom Labels cria um nome para o modelo que é uma combinação do nome do projeto e do timestamp de quando o modelo foi criado.

Para treinar seu modelo, o Amazon Rekognition Custom Labels faz uma cópia das imagens originais de treinamento e teste. Por padrão, as imagens copiadas são criptografadas em repouso com uma chave que a AWS possui e gerencia. Também é possível optar por usar a sua própria AWS KMS key. Se usa sua própria chave do KMS, precisará das permissões a seguir na chave do KMS.

- kms:CreateGrant
- kms:DescribeKey

Para obter mais informações, consulte Conceitos do AWS Key Management Service. Suas imagens de origem não são afetadas.

É possível usar a criptografia do lado do servidor (SSE-KMS) para criptografar as imagens de treinamento e teste em seu bucket do Amazon S3, antes de serem copiadas pelo Amazon Rekognition Custom Labels. Para permitir que as etiquetas personalizadas do Amazon Rekognition acessem suas imagens AWS , sua conta precisa das seguintes permissões na chave KMS.

- kms:GenerateDataKey
- kms:Decrypt

Para obter mais informações, consulte Como proteger dados usando criptografia do lado do servidor com chaves KMS armazenadas no AWS Key Management Service (SSE-KMS).

Depois de treinar um modelo, é possível avaliar seu desempenho e fazer melhorias. Para obter mais informações, consulte Como melhorar um modelo treinado do Amazon Rekognition Custom Labels.

Para outras tarefas do modelo, como atribuir tag a um modelo, consulte Como gerenciar um modelo do Amazon Rekognition Custom Labels.

Tópicos

- Como treinar um modelo (console)
- Treinando um modelo (SDK)

Como treinar um modelo (console)

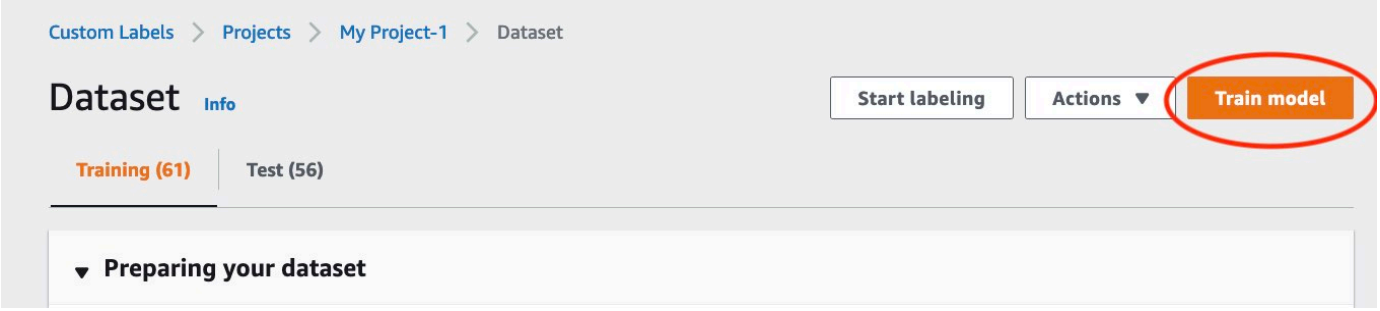
É possível usar o console do Amazon Rekognition Custom Labels para treinar um modelo.

O treinamento requer um conjunto de dados de treinamento e um conjunto de dados de teste. Se seu projeto não tiver um conjunto de dados de teste, o console do Amazon Rekognition Custom Labels divide o conjunto de dados de treinamento durante o treinamento para criar um para seu projeto. As imagens escolhidas são uma amostra representativa e não são usadas no conjunto de dados de treinamento. É recomendável dividir seu conjunto de dados de treinamento somente se não tiver um conjunto de dados de teste alternativo que possa usar. A divisão de um conjunto de dados de treinamento reduz o número de imagens disponíveis para treinamento.

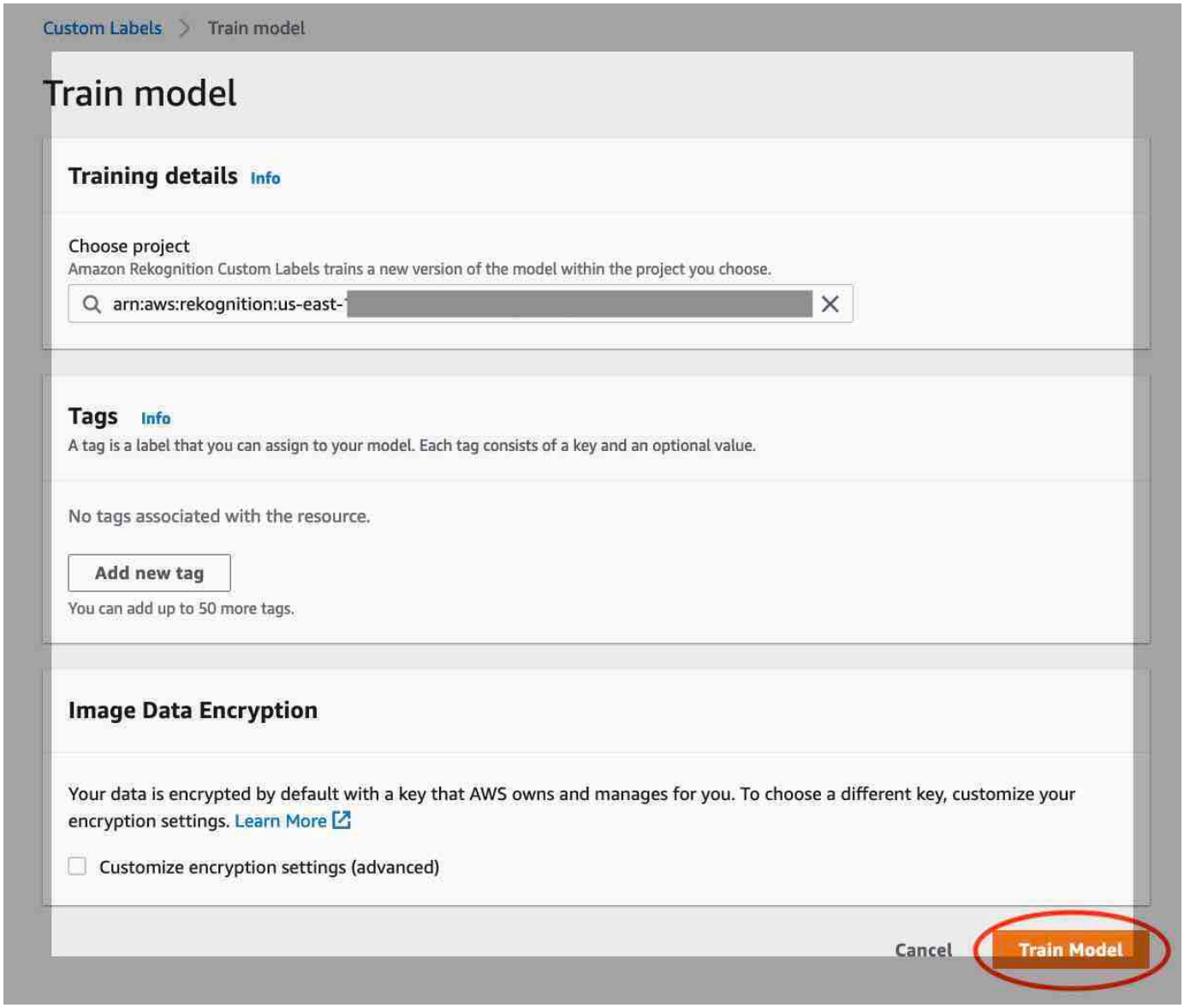
> **Note**
>
> Há uma cobrança pelo tempo necessário para treinar um modelo. Para obter mais informações, consulte Horas de treinamento.

Para treinar seu modelo (console)

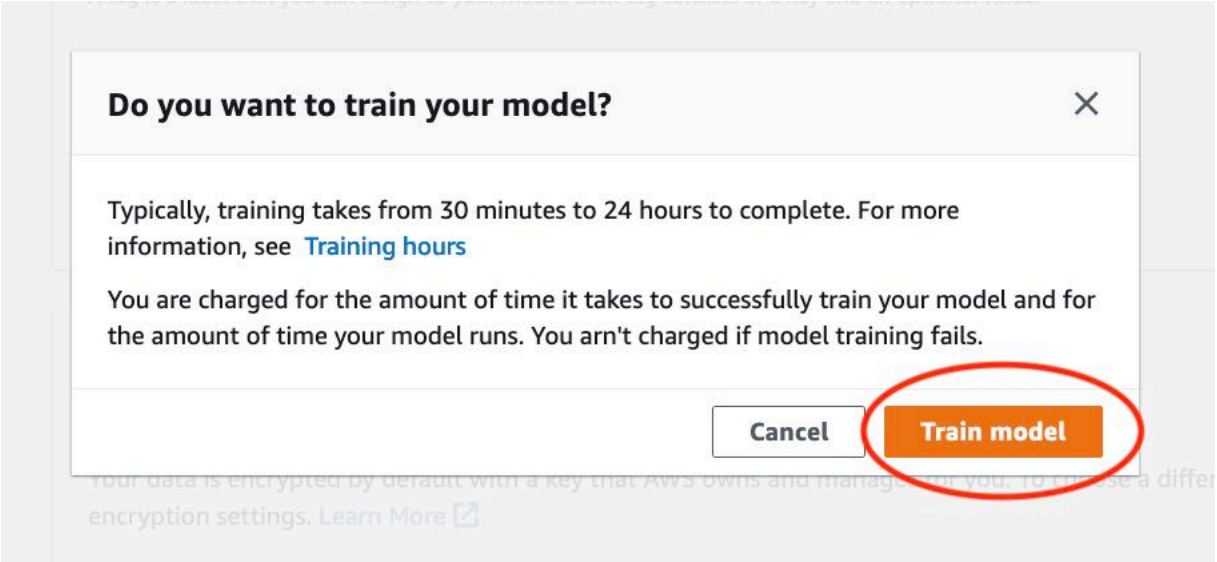
1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. No painel de navegação esquerdo, selecione Projetos.
4. Na página Projetos, escolha o projeto que contém o modelo que deseja treinar.
5. Na página Projeto, escolha Treinar modelo.



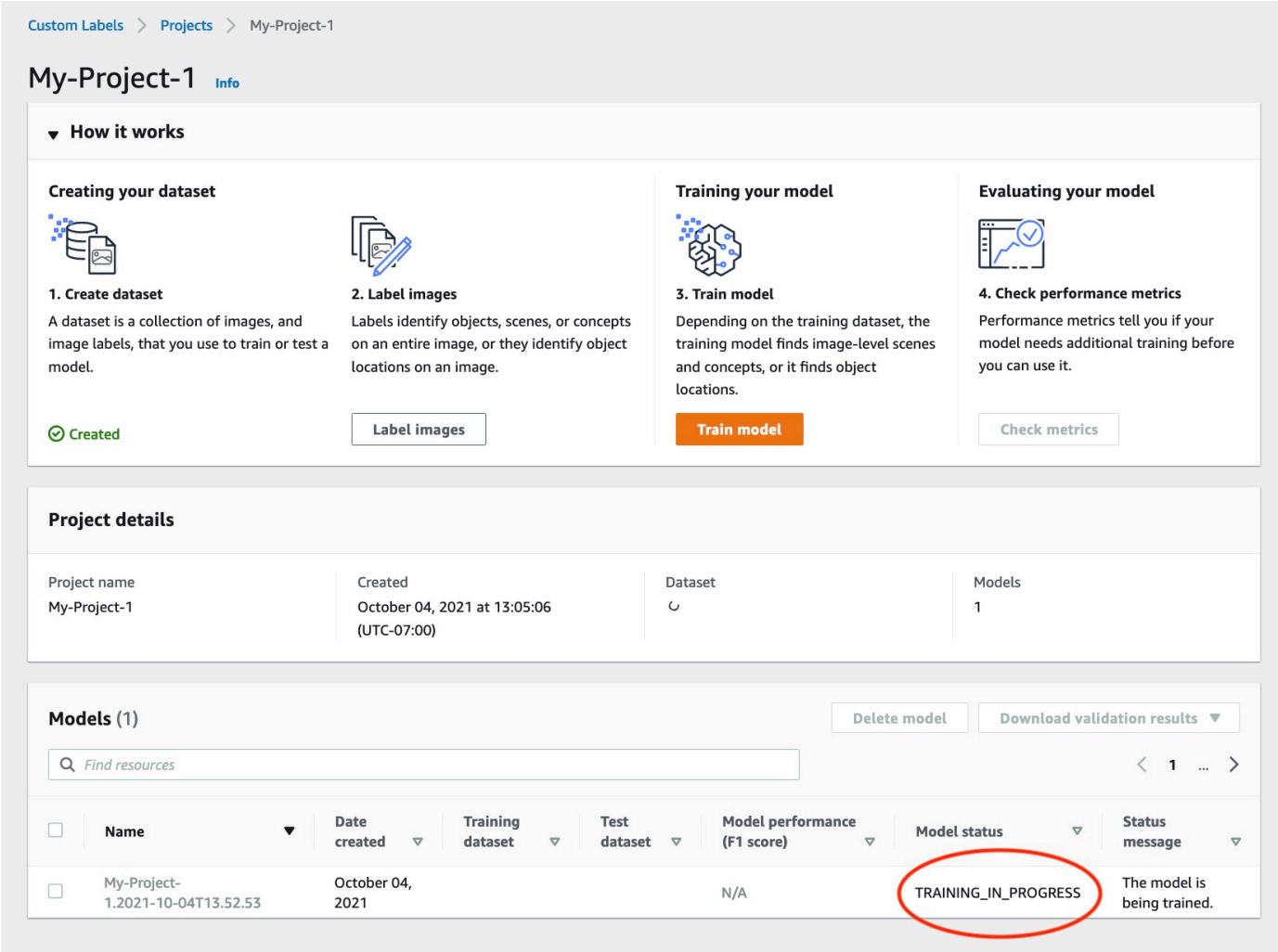
6. (Opcional) Se você quiser usar sua própria chave de criptografia do AWS KMS, faça o seguinte:
   a. Em Criptografia de dados de imagem, escolha Personalizar configurações de criptografia (avançado).
   b. Em encryption.aws_kms_key, insira o nome do recurso da Amazon (ARN) da sua chave ou escolha uma chave do AWS KMS existente. Para criar uma nova chave, escolha Criar uma chave do AWS IMS.
7. (Opcional) se quiser adicionar tags ao seu modelo, faça o seguinte:
   a. Na seção Tags, escolha Adicionar nova tag.
   b. Insira o seguinte:
      i. O nome da chave em Chave.
      ii. O valor da chave em Valor.
   c. Para adicionar mais tags, repita as etapas 6a e 6b.
   d. (Opcional) Se deseja remover uma tag, selecione Remover ao lado da tag que você deseja remover. Se estiver removendo uma tag salva anteriormente, ela será removida quando você salvar suas alterações.
8. Na página Treinar modelo, escolha Treinar modelo. O nome do recurso da Amazon (ARN) do seu projeto deve estar na caixa de edição Escolher projeto. Caso contrário, insira o ARN do seu projeto.



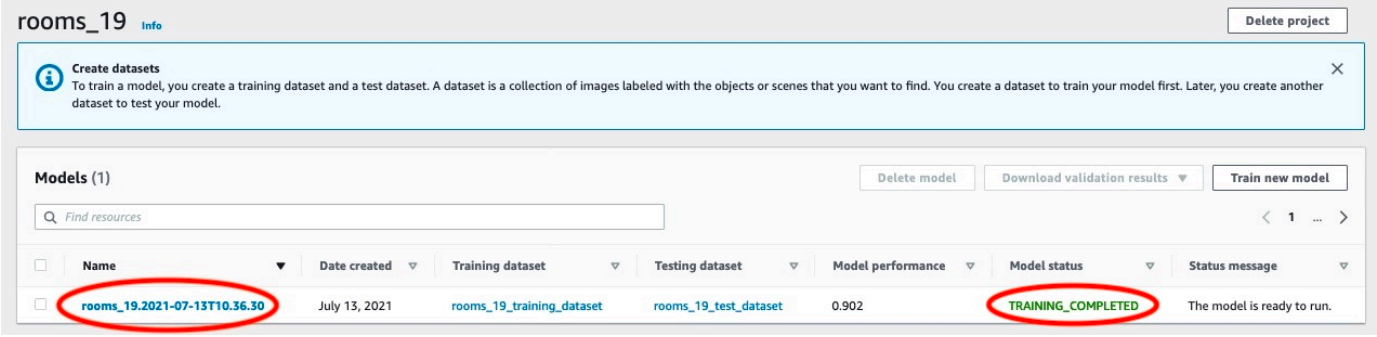
9. Na caixa de diálogo Você quer treinar seu modelo?, escolha Treinar modelo.



10. Na seção Modelos da página do projeto, pode verificar o status atual na coluna `Model Status`, onde o treinamento está em andamento. O treinamento de um modelo demora para ser concluído.



11. Após a conclusão do treinamento, escolha o nome do modelo. O treinamento é concluído quando o status do modelo for TRAINING_COMPLETED. Se o treinamento falhar, leia Como depurar um treinamento de modelo em falha.



12. Próxima etapa: avalie seu modelo. Para obter mais informações, Como melhorar um modelo treinado do Amazon Rekognition Custom Labels.

Treinando um modelo (SDK)

Você treina um modelo ligando CreateProjectVersion. Para treinar um modelo, as seguintes informações são necessárias:

- Nome: um nome exclusivo para a versão do modelo.
- ARN do projeto: o nome do recurso da Amazon (ARN) do projeto que gerencia o modelo.
- Local dos resultados do treinamento: o local do Amazon S3 em que os resultados são colocados. É possível usar o mesmo local do bucket do console do Amazon S3 ou escolher um local diferente. A recomendação é escolher um local diferente, pois isso permite a definição de permissões e evitar possíveis conflitos de nomenclatura com os resultados do treinamento do uso do console do Amazon Rekognition Custom Labels.

O treinamento usa os conjuntos de dados de treinamento e teste associados ao projeto. Para obter mais informações, consulte Como gerenciar conjuntos de dados.

> **Note**
>
> Você tem a opção de especificar arquivos de manifesto do conjunto de dados de treinamento e teste que são externos a um projeto. Se abrir o console após treinar um modelo com arquivos de manifesto externos, o Amazon Rekognition Custom Labels criará os conjuntos de dados para você usando o último conjunto de arquivos de manifesto usado para treinamento. Não é mais possível treinar uma versão de modelo para o projeto especificando arquivos de manifesto externos. Para obter mais informações, consulte CreatePrjectVersion.

A resposta de `CreateProjectVersion` é um ARN que você usa para identificar a versão do modelo em solicitações subsequentes. Também é possível usar o ARN para proteger a versão do modelo. Para obter mais informações, consulte Como proteger projetos do Amazon Rekognition Custom Labels.

O treinamento de uma versão do modelo demora para ser concluído. Os exemplos em Python e Java neste tópico usam esperadores para aguardar a conclusão do treinamento. Um agentes de espera é um métodos utilitário que sonda um determinado estado para verificar se ele ocorreu em um cliente. Como alternativa, é possível obter o status atual do treinamento ao chamar `DescribeProjectVersions`. O treinamento é concluído quando o valor do campo `Status` for

`TRAINING_COMPLETED`. Depois que o treinamento for concluído, será possível avaliar a qualidade do modelo analisando os resultados da avaliação.

Como treinar um modelo (SDK)

O exemplo a seguir mostra como treinar um modelo usando os conjuntos de dados de treinamento e teste associados a um projeto.

Para treinar um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir para treinar um projeto.

   AWS CLI

   O exemplo a seguir cria um modelo. O conjunto de dados de treinamento é dividido para criar o conjunto de dados de teste. Substitua o seguinte:

   - `my_project_arn` com o nome do recurso da Amazon (ARN) do projeto.
   - `version_name` com um nome exclusivo de versão de sua escolha.
   - `output_bucket` com o nome do bucket do Amazon S3 no qual o Amazon Rekognition Custom Labels salva os resultados do treinamento.
   - `output_folder` com o nome da pasta em que os resultados do treinamento são salvos.
   - (parâmetro opcional) `--kms-key-id` com identificador para sua chave mestra de cliente do AWS Key Management Service.

```
aws rekognition create-project-version \
  --project-arn project_arn \
  --version-name version_name \
  --output-config '{"S3Bucket":"output_bucket", "S3KeyPrefix":"output_folder"}'
 \
  --profile custom-labels-access
```

   Python

   O exemplo a seguir cria um modelo. Forneça os seguintes argumentos de linha de comando:

   - `project_arn`: o nome do recurso da Amazon (ARN) do projeto.

- `version_name`: um nome exclusivo de versão para o modelo de sua escolha.
- `output_bucket`: o nome do bucket do Amazon S3 no qual o Amazon Rekognition Custom Labels salva os resultados do treinamento.
- `output_folder`: o nome da pasta em que os resultados do treinamento são salvos.

Você tem a opção de fornecer os seguintes parâmetros de linha de comando para anexar uma tag ao seu modelo:

- `tag`: um nome de tag da sua escolha que você deseja anexar ao modelo.
- `tag_value`, o valor da tag.

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)


import argparse
import logging
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def train_model(rek_client, project_arn, version_name, output_bucket,
 output_folder, tag_key, tag_key_value):
    """
    Trains an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to train a
 model.
    :param version_name: A version for the model.
    :param output_bucket: The S3 bucket that hosts training output.
    :param output_folder: The path for the training output within output_bucket
    :param tag_key: The name of a tag to attach to the model. Pass None to
 exclude
    :param tag_key_value: The value of the tag. Pass None to exclude
```

```
    """

    try:
        #Train the model

        status=""
        logger.info("training model version %s for project %s",
            version_name, project_arn)


        output_config = json.loads(
            '{"S3Bucket": "'
            + output_bucket
            + '", "S3KeyPrefix": "'
            + output_folder
            + '" }  '
        )

        tags={}

        if tag_key is not None and tag_key_value is not None:
            tags = json.loads(
                '{"' + tag_key + '":"' + tag_key_value + '"}'
            )


        response=rek_client.create_project_version(
            ProjectArn=project_arn,
            VersionName=version_name,
            OutputConfig=output_config,
            Tags=tags
        )

        logger.info("Started training: %s", response['ProjectVersionArn'])

        # Wait for the project version training to complete.

        project_version_training_completed_waiter =
rek_client.get_waiter('project_version_training_completed')
        project_version_training_completed_waiter.wait(ProjectArn=project_arn,
        VersionNames=[version_name])


        # Get the completion status.
```

```
 describe_response=rek_client.describe_project_versions(ProjectArn=project_arn,
            VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            logger.info("Status: %s", model['Status'])
            logger.info("Message: %s", model['StatusMessage'])
            status=model['Status']


        logger.info("finished training")

        return response['ProjectVersionArn'], status

    except ClientError as err:
        logger.exception("Couldn't create model: %s", err.response['Error']
['Message'] )
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to train a
 model"
    )

    parser.add_argument(
        "version_name", help="A version name of your choosing."
    )

    parser.add_argument(
        "output_bucket", help="The S3 bucket that receives the training
 results."
    )

    parser.add_argument(
        "output_folder", help="The folder in the S3 bucket where training
 results are stored."
    )

    parser.add_argument(
```

```
        "--tag_name",  help="The name of a tag to attach to the model",
 required=False
    )

    parser.add_argument(
        "--tag_value",  help="The value for the tag.", required=False
    )




def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Training model version {args.version_name} for project
 {args.project_arn}")

        # Train the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        model_arn, status=train_model(rekognition_client,
            args.project_arn,
            args.version_name,
            args.output_bucket,
            args.output_folder,
            args.tag_name,
            args.tag_value)


        print(f"Finished training model: {model_arn}")
        print(f"Status: {status}")


    except ClientError as err:
        logger.exception("Problem training model: %s", err)
```

```
        print(f"Problem training model: {err}")
    except Exception as err:
        logger.exception("Problem training model: %s", err)
        print(f"Problem training model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

O exemplo a seguir treina um modelo. Forneça os seguintes argumentos de linha de comando:

- `project_arn`: o nome do recurso da Amazon (ARN) do projeto.
- `version_name`: um nome exclusivo de versão para o modelo de sua escolha.
- `output_bucket`: o nome do bucket do Amazon S3 no qual o Amazon Rekognition Custom Labels salva os resultados do treinamento.
- `output_folder`: o nome da pasta em que os resultados do treinamento são salvos.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.CreateProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
```

```
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class TrainModel {

    public static final Logger logger =
 Logger.getLogger(TrainModel.class.getName());

    public static String trainMyModel(RekognitionClient rekClient, String
 projectArn, String versionName,
            String outputBucket, String outputFolder) {

        try {

            OutputConfig outputConfig =
 OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

            logger.log(Level.INFO, "Training Model for project {0}",
 projectArn);
            CreateProjectVersionRequest createProjectVersionRequest =
 CreateProjectVersionRequest.builder()

 .projectArn(projectArn).versionName(versionName).outputConfig(outputConfig).build();

            CreateProjectVersionResponse response =
 rekClient.createProjectVersion(createProjectVersionRequest);

            logger.log(Level.INFO, "Model ARN: {0}",
 response.projectVersionArn());
            logger.log(Level.INFO, "Training model...");

            // wait until training completes

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
 DescribeProjectVersionsRequest.builder()
                    .versionNames(versionName)
                    .projectArn(projectArn)
                    .build();
```

```
            RekognitionWaiter waiter = rekClient.waiter();

            WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionTrainingCompleted(describeProjectVersionsRequest);

            Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {
                System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
projectVersionDescription.statusAsString());
                System.out.println("Message: " +
projectVersionDescription.statusMessage());
            }

            return response.projectVersionArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String versionName = null;
        String projectArn = null;
        String projectVersionArn = null;
        String bucket = null;
        String location = null;
```

```
        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
 <output_bucket> <output_folder>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that you want to use.
 \n\n"
                + "   version_name - A version name for the model.\n\n"
                + "   output_bucket - The S3 bucket in which to place the
 training output. \n\n"
                + "   output_folder - The folder within the bucket that the
 training output is stored in. \n\n";

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        versionName = args[1];
        bucket = args[2];
        location = args[3];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Train model
            projectVersionArn = trainMyModel(rekClient, projectArn, versionName,
 bucket, location);

            System.out.println(String.format("Created model: %s for Project ARN:
 %s", projectVersionArn, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }
```

```
    }

}
```

3. Se o treinamento falhar, leia Como depurar um treinamento de modelo em falha.

Como depurar um treinamento de modelo em falha

É possível encontrar erros durante o treinamento do modelo. O Amazon Rekognition Custom Labels relata erros de treinamento no console e na resposta do. DescribeProjectVersions

Os erros são terminais (o treinamento não pode continuar) ou não terminais (o treinamento pode continuar). Para erros relacionados ao conteúdo dos conjuntos de dados de treinamento e teste, é possível baixar os resultados da validação (um resumo do manifesto e manifestos de validação de treinamento e teste). Use os códigos de erro nos resultados da validação para encontrar mais informações nesta seção. Esta seção também fornece informações sobre erros do arquivo de manifesto (erros terminais que ocorrem antes da validação do conteúdo do arquivo de manifesto).

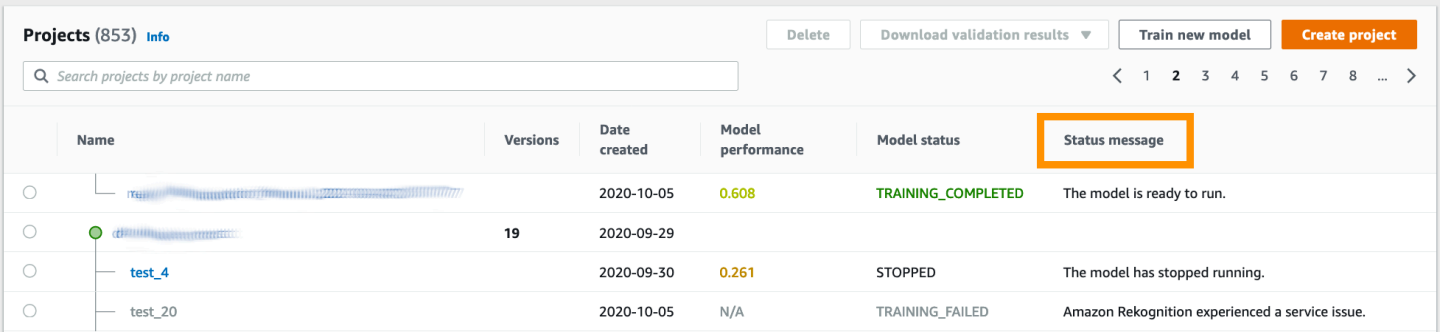
> **Note**
>
> Um manifesto é o arquivo usado para armazenar o conteúdo de um conjunto de dados.

É possível corrigir alguns erros usando o console do Amazon Rekognition Custom Labels. Outros erros podem exigir que você atualize os arquivos de manifesto de treinamento ou teste. Talvez seja necessário fazer outras alterações, como as permissões do IAM. Para obter mais informações, consulte a documentação de erros individuais.

Erros terminais

Os erros terminais interrompem o treinamento de um modelo. Há três categorias de erros terminais de treinamento: erros de serviço, erros de arquivo de manifesto e erros de conteúdo manifesto.

No console, o Amazon Rekognition Custom Labels mostra erros terminais para um modelo na coluna Mensagem de status da página de projetos. Painel de gerenciamento de projetos mostrando a lista de projetos com nome, versões, data de criação, desempenho do modelo e mensagem de status indicando o estado do modelo, como “em treinamento”, “concluído” ou “reprovado”.



Se você estiver usando o AWS SDK, poderá descobrir se ocorreu um erro no arquivo de manifesto do terminal ou no conteúdo do manifesto do terminal verificando a resposta de. DescribeProjectVersions Neste caso, o valor `Status` é `TRAINING_FAILED` e o campo `StatusMessage` contém o erro.

Erros de serviço

Os erros terminais do serviço ocorrem quando o Amazon Rekognition enfrenta um problema de serviço e não consegue continuar o treinamento. Por exemplo, a falha de outro serviço do qual o Amazon Rekognition Custom Labels depende. O Amazon Rekognition Custom Labels relata erros de serviço no console quando o Amazon Rekognition teve um problema de serviço. Se você usa o AWS SDK, os erros de serviço que ocorrem durante o treinamento são apresentados como uma `InternalServerError` exceção por CreateProjectVersione. DescribeProjectVersions

Se ocorrer um erro de serviço, tente novamente treinar o modelo. Se o treinamento continuar falhando, entre em contato com o AWS Support e inclua todas as informações de erro relatadas com o erro do serviço.

Lista de erros terminais do arquivo de manifesto

Os erros do arquivo de manifesto são erros terminais, nos conjuntos de dados de treinamento e teste, que ocorrem no nível do arquivo ou em vários arquivos. Os erros do arquivo de manifesto são detectados antes que o conteúdo dos conjuntos de dados de treinamento e teste seja validado. Os erros do arquivo de manifesto impedem o relatório de erros não terminais de validação. Por exemplo, um arquivo de manifesto de treinamento vazio gera um erro O arquivo de manifesto está vazio. Como o arquivo está vazio, nenhum erro não terminais de validação de linha JSON pode ser relatado. O resumo do manifesto também não foi criado.

Você deve corrigir os erros do arquivo de manifesto antes de treinar seu modelo.

A seguir, são listados os erros do arquivo de manifesto.

- A extensão ou o conteúdo do arquivo de manifesto são inválidos.
- O arquivo de manifesto está vazio.
- O tamanho do arquivo de manifesto excede o tamanho máximo permitido.
- Não é possível gravar no bucket S3 de saída.
- As permissões de bucket do S3 estão incorretas.

Lista de erros terminais de conteúdo do manifesto

Erros de conteúdo manifesto são erros terminais relacionados ao conteúdo em um manifesto. Por exemplo, se você receber o erro O arquivo de manifesto contém imagens rotuladas insuficientes por rótulo para realizar a divisão automática, o treinamento não poderá ser concluído, pois não há imagens rotuladas suficientes no conjunto de dados de treinamento para criar um conjunto de dados de teste.

Além de ser relatado no console e na resposta do `DescribeProjectVersions`, o erro é relatado no resumo do manifesto junto com quaisquer outros erros terminais de conteúdo do manifesto. Para obter mais informações, consulte Noções básicas sobre o resumo do manifesto.

Erros não terminais de linha JSON também são relatados em manifestos separados de resultados de validação de treinamento e teste. Os erros não terminais de linha JSON encontrados pelo Amazon Rekognition Custom Labels não estão necessariamente relacionados aos erros de conteúdo do manifesto que interrompem o treinamento. Para obter mais informações, consulte Noções básicas sobre treinar e testar manifestos de resultados de validação.

Os erros do arquivo de manifesto devem ser corrigidos antes de treinar seu modelo.

A seguir estão as mensagens de erro para erros de conteúdo manifesto.

- O arquivo de manifesto contém muitas linhas inválidas.
- O arquivo de manifesto contém imagens de vários buckets do S3.
- ID de proprietário inválido para imagens do bucket S3.
- O arquivo de manifesto contém imagens rotuladas insuficientes por rótulo para realizar a divisão automática.
- O arquivo de manifesto tem poucos rótulos.
- O arquivo de manifesto tem muitos rótulos.
- Menos de {}% de sobreposição de rótulos entre os arquivos de manifesto de treinamento e teste.

- O arquivo de manifesto tem poucos rótulos utilizáveis.
- Menos de {}% de sobreposição de rótulos utilizáveis entre os arquivos de manifesto de treinamento e teste.
- Falha ao copiar imagens do bucket do S3.

Lista de erros não terminais de validação de linha JSON

Os erros de validação da linha JSON são erros não terminais que não exigem que o Amazon Rekognition Custom Labels pare de treinar um modelo.

Os erros de validação da linha JSON não são mostrados no console.

Nos conjuntos de dados de treinamento e teste, uma linha JSON representa as informações de treinamento ou teste de uma única imagem. Os erros de validação em uma linha JSON, como uma imagem inválida, são relatados nos manifestos de validação de treinamento e teste. O Amazon Rekognition Custom Labels conclui o treinamento usando as outras linhas JSON válidas que estão no manifesto. Para obter mais informações, consulte Noções básicas sobre treinar e testar manifestos de resultados de validação. Para obter informações sobre as regras de validação, consulte Regras de validação para arquivos de manifesto.

> **Note**
>
> O treinamento falhará se houver muitos erros de linha JSON.

É recomendável a correção de erros não terminais de linha JSON, pois eles podem causar erros futuros ou afetar o treinamento do modelo.

O Amazon Rekognition Custom Labels pode gerar os seguintes erros não terminais de validação de linha JSON.

- A chave source-ref está ausente.
- O formato do valor source-ref é inválido.
- Nenhum atributo de rótulo encontrado.
- O formato {} do atributo do rótulo é inválido.
- O formato dos metadados do atributo do rótulo é inválido.
- Nenhum atributo de rótulo válido encontrado.

- Uma ou mais caixas delimitadoras não têm um valor de confiança.
- Um ou mais IDs de classe estão faltando no mapa de classe.
- A linha JSON tem um formato inválido.
- A imagem é inválida. Verifique o caminho e/ou as propriedades da imagem do S3.
- A caixa delimitadora tem valores fora do quadro.
- A altura e a largura da caixa delimitadora são muito pequenas.
- Há mais caixas delimitadoras do que o máximo permitido.
- Nenhuma anotação válida encontrada.

Noções básicas sobre o resumo do manifesto

O resumo do manifesto contém as seguintes informações.

- Informações de erro sobre Lista de erros terminais de conteúdo do manifesto encontradas durante a validação.
- Informações de localização do erro para Lista de erros não terminais de validação de linha JSON nos conjuntos de dados de treinamento e teste.
- Estatísticas de erro, como o número total de linhas JSON inválidas encontradas nos conjuntos de dados de treinamento e teste.

O resumo do manifesto é criado durante o treinamento, se não houver Lista de erros terminais do arquivo de manifesto. Para obter a localização do arquivo de resumo do manifesto (manifest_summary.json), consulte Como obter os resultados de validação.

> **Note**
>
> Os erros de serviço e os erros do arquivo de manifesto não são relatados no resumo do manifesto. Para obter mais informações, consulte Erros terminais.

Para obter mais informações sobre erros específicos no conteúdo do manifesto, consulte Erros terminais de conteúdo do manifesto.

Formato de arquivo de resumo do manifesto

Um arquivo de manifesto tem 2 seções: `statistics` e `errors`.

estatísticas

`statistics` contém informações sobre os erros nos conjuntos de dados de treinamento e teste.

- `training`: as estatísticas e erros encontrados no conjunto de dados de treinamento.
- `testing`: as estatísticas e erros encontrados no conjunto de dados de teste.

Os objetos na matriz `errors` contêm o código de erro e a mensagem para erros de conteúdo do manifesto.

A matriz `error_line_indices` contém os números de linha de cada linha JSON no manifesto de treinamento ou teste que tem um erro. Para obter mais informações, consulte Como corrigir erros de treinamento.

erros

Erros abrangendo o conjunto de dados de treinamento e teste. Por exemplo, um ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP ocorre quando não há rótulos utilizáveis suficientes que se sobreponham aos conjuntos de dados de treinamento e teste.

```
{
    "statistics": {
        "training":
            {
                "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
 OBJECT_LOCALIZATION and NOT_DETERMINED
                "total_json_lines": Number,   # Total number json lines (images) in the
  training manifest.
                "valid_json_lines": Number,   # Total number of JSON Lines (images)
 that can be used for training.
                "invalid_json_lines": Number, # Total number of invalid JSON Lines.
 They are not used for training.
                "ignored_json_lines": Number, # JSON Lines that have a valid schema but
 have no annotations. The aren't used for training and aren't counted as invalid.
                "error_json_line_indices": List[int], # Contains a list of line numbers
 for JSON line errors in the training dataset.
                "errors": [
                    {
                        "code": String, # Error code for a training manifest content
 error.
```

```
                        "message": String # Description for a training manifest content
 error.
                    }
                ]
            },
        "testing":
            {
                "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
 OBJECT_LOCALIZATION and NOT_DETERMINED
                "total_json_lines": Number, # Total number json lines (images) in the
 manifest.
                "valid_json_lines": Number,  # Total number of JSON Lines (images) that
 can be used for testing.
                "invalid_json_lines": Number, # Total number of invalid JSON Lines.
 They are not used for testing.
                "ignored_json_lines": Number, # JSON Lines that have a valid schema but
 have no annotations. They aren't used for testing and aren't counted as invalid.
                "error_json_line_indices": List[int], # contains a list of error record
 line numbers in testing dataset.
                "errors": [
                    {
                        "code": String,   # # Error code for a testing manifest content
 error.
                        "message": String # Description for a testing manifest content
 error.
                    }
                ]
            }
    },
    "errors": [
        {
            "code": String, # # Error code for errors that span the training and
 testing datasets.
            "message": String # Description of the error.
        }
    ]
}
```

Exemplo de resumo do manifesto

O exemplo a seguir é um resumo parcial do manifesto que mostra um erro terminal de conteúdo do manifesto (ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST). A matriz

`error_json_line_indices` contém os números de linha dos erros não terminais de linha JSON no manifesto de validação de treinamento ou teste correspondente.

```
{
    "errors": [],
    "statistics": {
        "training": {
            "use_case": "NOT_DETERMINED",
            "total_json_lines": 301,
            "valid_json_lines": 146,
            "invalid_json_lines": 155,
            "ignored_json_lines": 0,
            "errors": [
                {
                    "code": "ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST",
                    "message": "The manifest file contains too many invalid rows."
                }
            ],
            "error_json_line_indices": [
                15,
                16,
                17,
                22,
                23,
                24,
                 .
                 .
                 .
                 .
                300
            ]
        },
        "testing": {
            "use_case": "NOT_DETERMINED",
            "total_json_lines": 15,
            "valid_json_lines": 13,
            "invalid_json_lines": 2,
            "ignored_json_lines": 0,
            "errors": [],
            "error_json_line_indices": [
                13,
                15
            ]
```

```
        }
    }
}
```

Noções básicas sobre treinar e testar manifestos de resultados de validação

Durante o treinamento, o Amazon Rekognition Custom Labels cria manifestos de resultados de validação para conter erros não terminais de linha JSON. Os manifestos dos resultados da validação são cópias dos conjuntos de dados de treinamento e teste com informações de erro adicionadas. É possível acessar os manifestos de validação após a conclusão do treinamento. Para obter mais informações, consulte Como obter os resultados de validação. O Amazon Rekognition Custom Labels também cria um resumo do manifesto que inclui informações gerais sobre erros de linha JSON, como locais de erros e contagens de erros de linha JSON. Para obter mais informações, consulte Noções básicas sobre o resumo do manifesto.

> **Note**
>
> Os resultados da validação (manifestos de resultados de validação de treinamento e teste e resumo do manifesto) são criados somente se não houver Lista de erros terminais do arquivo de manifesto.

Um manifesto contém linhas JSON para cada imagem no conjunto de dados. Nos manifestos dos resultados da validação, as informações de erro da linha JSON são adicionadas às linhas JSON em que ocorrem erros.

Um erro de linha JSON é um erro não terminal relacionado a uma única imagem. Um erro não terminal de validação pode invalidar toda a linha JSON ou apenas uma parte. Por exemplo, se a imagem referenciada em uma linha JSON não estiver no formato PNG ou JPG, ocorrerá um erro ERROR_INVALID_IMAGE e toda a linha JSON será excluída do treinamento. O treinamento continua com outras linhas JSON válidas.

Em uma linha JSON, um erro pode significar que a linha JSON ainda pode ser usada para treinamento. Por exemplo, se o valor esquerdo de uma das quatro caixas delimitadoras associadas a um rótulo for negativo, o modelo ainda será treinado usando as outras caixas delimitadoras válidas. As informações de erro da linha JSON são retornadas para a caixa delimitadora inválida

(ERROR_INVALID_BOUNDING_BOX). Neste exemplo, as informações do erro são adicionadas ao objeto `annotation` em que o erro ocorre.

Os erros de aviso, como WARNING_NO_ANNOTATIONS, não são usados para treinamento e contam como linhas JSON ignoradas (`ignored_json_lines`) no resumo do manifesto. Para obter mais informações, consulte Noções básicas sobre o resumo do manifesto. Além disso, as linhas JSON ignoradas não contam para o limite de erro de 20% para treinamento e teste.

Para obter informações sobre erros não terminais específicos de validação de dados, consulte Erros não terminais de validação de linha JSON.

> **Note**
>
> Se houver muitos erros de validação de dados, o treinamento será interrompido e um erro terminal de ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST será relatado no resumo do manifesto.

Para obter informações sobre como corrigir erros da linha JSON, consulte Como corrigir erros de treinamento.

Formato de erro de linha JSON

O Amazon Rekognition Custom Labels adiciona informações de erros não terminais de validação ao nível da imagem e ao formato de localização de objetos JSON Lines. Para obter mais informações, consulte the section called “Criar um arquivo de manifesto”.

Erros no nível da imagem

O exemplo a seguir mostra as matrizes `Error` em uma linha JSON em nível de imagem. Há dois conjuntos de erros. Erros relacionados aos metadados do atributo do rótulo (neste exemplo, metadados esportivos) e erros relacionados à imagem. Um erro inclui um código de erro (código), mensagem de erro (mensagem). Para obter mais informações, consulte Importar rótulos ao nível da imagem em arquivos de manifesto.

```
{
    "source-ref": String,
    "sport": Number,
    "sport-metadata": {
        "class-name": String,
```

```
        "confidence": Float,
        "type": String,
        "job-name": String,
        "human-annotated": String,
        "creation-date": String,
        "errors": [
            {
                "code": String, # error codes for label
                "message": String # Description and additional contextual details of
 the error
            }
        ]
    },
    "errors": [
        {
            "code": String, # error codes for image
            "message": String # Description and additional contextual details of the
 error
        }
    ]
}
```

Erros de localização de objetos

O exemplo a seguir mostra as matrizes de erro em uma linha JSON de localização de objetos. A linha JSON contém informações de matriz `Errors` para campos nas seguintes seções da linha JSON. Cada objeto `Error` inclui o código de erro e a mensagem de erro.

- label attribute: erros nos campos do atributo do rótulo. Consulte `bounding-box` no exemplo.
- anotações: os erros de anotação (caixas delimitadoras) são armazenados na matriz `annotations` dentro do atributo de rótulo.
- label attribute-metadata: erros nos metadados do atributo do rótulo. Consulte `bounding-box-metadata` no exemplo.
- image: erros não relacionados aos campos de atributo do rótulo, anotação e metadados do atributo do rótulo.

Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

```
{
    "source-ref": String,
```

```
    "bounding-box": {
        "image_size": [
            {
                "width": Int,
                "height": Int,
                "depth":Int,
            }
        ],
        "annotations": [
            {
                "class_id": Int,
                "left": Int,
                "top": Int,
                "width": Int,
                "height": Int,
                "errors": [   # annotation field errors
                    {
                        "code": String, # annotation field error code
                        "message": String # Description and additional contextual
details of the error
                    }
                ]
            }
        ],
        "errors": [ #label attribute field errors
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
the error
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": Float
            }
        ],
        "class-map": {
            String: String
        },
        "type": String,
        "human-annotated": String,
        "creation-date": String,
```

```
        "job-name": String,
        "errors": [  #metadata field errors
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
 the error
            }
        ]
    },
   "errors": [  # image errors
        {
            "code": String, # error code
            "message": String # Description and additional contextual details of the
 error
        }
    ]
}
```

Exemplo de erro de linha JSON

A seguinte linha JSON de localização de objetos (formatada para facilitar a leitura) mostra um erro ERROR_BOUNDING_BOX_TOO_SMALL. Neste exemplo, as dimensões da caixa delimitadora (altura e largura) não são maiores que 1 x 1.

```
{
    "source-ref": "s3://bucket/Manifests/images/199940-1791.jpg",
    "bounding-box": {
        "image_size": [
            {
                "width": 3000,
                "height": 3000,
                "depth": 3
            }
        ],
        "annotations": [
            {
                "class_id": 1,
                "top": 0,
                "left": 0,
                "width": 1,
                "height": 1,
                "errors": [
                    {
```

```
                        "code": "ERROR_BOUNDING_BOX_TOO_SMALL",
                        "message": "The height and width of the bounding box is too
 small."
                    }
                ]
            },
            {
                "class_id": 0,
                "top": 65,
                "left": 86,
                "width": 220,
                "height": 334
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": 1
            },
            {
                "confidence": 1
            }
        ],
        "class-map": {
            "0": "Echo",
            "1": "Echo Dot"
        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2019-11-20T02:57:28.288286",
        "job-name": "my job"
    }
}
```

Como obter os resultados de validação

Os resultados da validação contêm informações de erro para Lista de erros terminais de conteúdo do manifesto e Lista de erros não terminais de validação de linha JSON. Há três arquivos de resultados de validação.

- training_manifest_with_validation.json: uma cópia do arquivo de manifesto do conjunto de dados de treinamento com informações de erro de linha JSON adicionadas.

- testing_manifest_with_validation.json: uma cópia do arquivo de manifesto do conjunto de dados de teste com informações de erro de linha JSON adicionadas.
- manifest_summary.json: um resumo dos erros do conteúdo do manifesto e dos erros da linha JSON encontrados nos conjuntos de dados de treinamento e teste. Para obter mais informações, consulte Noções básicas sobre o resumo do manifesto.

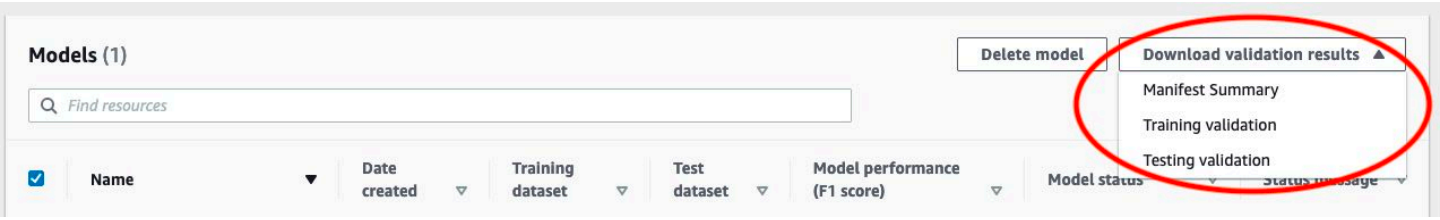
Para obter informações sobre o conteúdo dos manifestos de validação de treinamento e teste, consulte Como depurar um treinamento de modelo em falha.

> **Note**
>
> - Os resultados da validação são criados somente se nenhum Lista de erros terminais do arquivo de manifesto for gerado durante o treinamento.
> - Se ocorrer um erro de serviço após a validação do manifesto de treinamento e teste, os resultados da validação serão criados, mas a resposta DescribeProjectVersionsnão incluirá os locais dos arquivos dos resultados da validação.

Depois que o treinamento for concluído ou falhar, você poderá baixar os resultados da validação usando o console Amazon Rekognition Custom Labels ou obter a localização do bucket do Amazon S3 chamando a API. DescribeProjectVersions

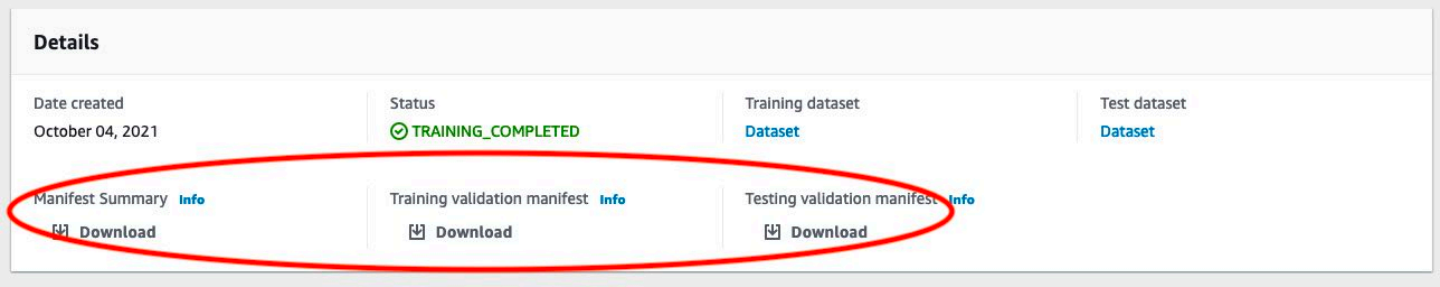
Como obter resultados de validação (console)

Se estiver usando o console para treinar seu modelo, poderá baixar os resultados da validação da lista de modelos de um projeto, conforme mostrado no diagrama a seguir. O painel Modelos mostra os resultados de treinamento e validação do modelo com a opção de baixar os resultados da validação.



Também é possível acessar o download dos resultados da validação na página de detalhes de um modelo. A página de detalhes mostra os detalhes do conjunto de dados com status, conjuntos de

dados de treinamento e teste e links de download para resumo do manifesto, manifesto de validação de treinamento e manifesto de validação de teste.



Para obter mais informações, consulte Como treinar um modelo (console).

Como obter resultados de validação (SDK)

Após a conclusão do treinamento do modelo, o Amazon Rekognition Custom Labels armazena os resultados da validação no bucket do Amazon S3 especificado durante o treinamento. Você pode obter a localização do bucket do S3 chamando a DescribeProjectVersionsAPI após a conclusão do treinamento. Para treinar um modelo, consulte Treinando um modelo (SDK).

Um ValidationDataobjeto é retornado para o conjunto de dados de treinamento (TrainingDataResult) e o conjunto de dados de teste (TestingDataResult). O manifesto resumido é retornado no `ManifestSummary`.

Depois de obter a localização do bucket do Amazon S3, é possível baixar os resultados da validação. Para obter mais informações, consulte Como fazer download de um objeto de um bucket do S3?. Você também pode usar a operação GetObject.

Para obter dados de validação (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o exemplo a seguir para obter a localização dos resultados da validação.

   Python

   Substitua `project_arn` pelo nome do recurso da Amazon (ARN) do projeto que contém o modelo. Para obter mais informações, consulte Como gerenciar um projeto do Amazon Rekognition Custom Labels. Substitua `version_name` pelo nome da versão do modelo. Para obter mais informações, consulte Treinando um modelo (SDK).

```
import boto3
import io
from io import BytesIO
import sys
import json


def describe_model(project_arn, version_name):

    client=boto3.client('rekognition')

    response=client.describe_project_versions(ProjectArn=project_arn,
        VersionNames=[version_name])

    for model in response['ProjectVersionDescriptions']:
        print(json.dumps(model,indent=4,default=str))

def main():

    project_arn='project_arn'
    version_name='version_name'

    describe_model(project_arn, version_name)

if __name__ == "__main__":
    main()
```

3. Na saída do programa, observe o campo `Validation` dentro dos objetos `TestingDataResult` e `TrainingDataResult`. O manifesto resumido está no `ManifestSummary`.

Como corrigir erros de treinamento

O resumo do manifesto é usado para identificar Lista de erros terminais de conteúdo do manifesto e Lista de erros não terminais de validação de linha JSON encontrados durante o treinamento. Os erros de conteúdo do manifesto devem ser corrigidos. Também é recomendável a correção de erros não terminais da linha JSON. Para obter mais informações sobre erros específicos, consulte Erros não terminais de validação de linha JSON e Erros terminais de conteúdo do manifesto.

É possível fazer correções no conjunto de dados de treinamento ou teste usado para treinamento. Como alternativa, é possível fazer as correções nos arquivos de manifesto de validação de treinamento e teste e usá-los para treinar o modelo.

Depois de fazer as correções, você precisa importar os manifestos atualizados e treinar novamente o modelo. Para obter mais informações, consulte Criar um arquivo de manifesto.

O procedimento a seguir mostra como usar o resumo do manifesto para corrigir erros terminal de conteúdo do manifesto. O procedimento também mostra como localizar e corrigir erros da linha JSON nos manifestos de validação de treinamento e teste.

Para corrigir erros de treinamento do Amazon Rekognition Custom Labels

1. Baixe os arquivos de resultados da validação. Os nomes dos arquivos são training_manifest_with_validation.json, testing_manifest_with_validation.json e manifest_summary.json. Para obter mais informações, consulte Como obter os resultados de validação.
2. Abra o arquivo de resumo do manifesto (manifest_summary.json).
3. Corrija quaisquer erros no resumo do manifesto. Para obter mais informações, consulte Noções básicas sobre o resumo do manifesto.
4. No resumo do manifesto, itere por meio da matriz `error_line_indices` em `training` e corrija os erros em `training_manifest_with_validation.json` nos números de linha JSON correspondentes. Para obter mais informações, consulte the section called “Noções básicas sobre treinar e testar manifestos de resultados de validação”.
5. Itere por meio da matriz `error_line_indices` em `testing` e corrija os erros em `testing_manifest_with_validation.json` nos números de linha JSON correspondentes.
6. Treine novamente o modelo usando os arquivos de manifesto de validação como conjuntos de dados de treinamento e teste. Para obter mais informações, consulte the section called “Como treinar um modelo”.

Se você estiver usando o AWS SDK e optar por corrigir os erros nos arquivos de manifesto de dados de validação de treinamento ou teste, use a localização dos arquivos de manifesto de dados de validação nos parâmetros TestingDatade entrada TrainingDatae para CreateProjectVersion. Para obter mais informações, consulte Treinando um modelo (SDK).

Precedência de erro de linha JSON

Os erros de linha JSON a seguir são detectados primeiro. Se algum desses erros ocorrer, a validação dos erros da linha JSON será interrompida. Esses erros devem ser corrigidos antes de poder corrigir qualquer um dos outros erros da linha JSON

- MISSING_SOURCE_REF
- ERROR_INVALID_SOURCE_REF_FORMAT
- ERROR_NO_LABEL_ATTRIBUTES
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_INVALID_JSON_LINE

Erros terminais do arquivo de manifesto

Este tópico descreve o Lista de erros terminais do arquivo de manifesto. Os erros do arquivo de manifesto não têm um código de erro associado. Os manifestos dos resultados da validação não são criados quando ocorre um erro terminal no arquivo de manifesto. Para obter mais informações, consulte Noções básicas sobre o resumo do manifesto. Os erros terminais do manifesto impedem o relatório de Erros não terminais de validação de linha JSON.

A extensão ou o conteúdo do arquivo de manifesto são inválidos.

O arquivo de manifesto de treinamento ou teste não tem uma extensão de arquivo ou seu conteúdo é inválido.

Para corrigir o erro A extensão ou o conteúdo do arquivo de manifesto são inválidos.

- Verifique as seguintes possíveis causas nos arquivos de manifesto de treinamento e teste.
  - O arquivo de manifesto não tem uma extensão. Por convenção, a extensão do arquivo é `.manifest`.
  - Não foi possível encontrar o bucket ou a chave do Amazon S3 para o arquivo de manifesto.

O arquivo de manifesto está vazio.

O arquivo de manifesto de treinamento ou teste usado para treinamento existe, mas está vazio. O arquivo de manifesto precisa de uma linha JSON para cada imagem que você usa para treinamento e teste.

Para corrigir o erro O arquivo de manifesto está vazio.

1. Verifique quais manifestos de treinamento ou teste estão vazios.
2. Adicione linhas JSON ao arquivo de manifesto vazio. Para obter mais informações, consulte Criar um arquivo de manifesto. Como alternativa, crie um novo conjunto de dados com o console. Para obter mais informações, consulte the section called “Como criar conjuntos de dados com imagens”.

O tamanho do arquivo de manifesto excede o tamanho máximo permitido.

O tamanho do arquivo do manifesto de treinamento ou teste (em bytes) é muito grande. Para obter mais informações, consulte Diretrizes e cotas no Amazon Rekognition Custom Labels. Um arquivo de manifesto pode ter menos do que o número máximo de linhas JSON e ainda exceder o tamanho máximo do arquivo.

Não é possível pode usar o console do Amazon Rekognition Custom Labels para corrigir o erro O tamanho do arquivo de manifesto excede o tamanho máximo permitido.

Para corrigir o erro O tamanho do arquivo de manifesto excede o tamanho máximo permitido.

1. Verifique quais manifestos de treinamento e teste excedem o tamanho máximo do arquivo.
2. Reduza o número de linhas JSON nos arquivos de manifesto que são muito grandes. Para obter mais informações, consulte Criar um arquivo de manifesto.

As permissões de bucket do S3 estão incorretas.

O Amazon Rekognition Custom Labels não tem permissões para um ou mais buckets que contêm os arquivos de manifesto de treinamento e teste.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Para corrigir o erro As permissões do bucket do S3 estão incorretas.

- Verifique as permissões dos buckets contendo os manifestos de treinamento e teste. Para obter mais informações, consulte Etapa 2: configure as permissões do console do Amazon Rekognition Custom Labels.

Não é possível gravar no bucket S3 de saída.

O serviço não consegue gerar os arquivos de saída do treinamento.

Para corrigir o erro Não é possível gravar no bucket S3 de saída.

- Verifique se as informações do bucket do Amazon S3 no parâmetro OutputConfigde entrada para estão corretas CreateProjectVersion.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Erros terminais de conteúdo do manifesto

Este tópico descreve o Lista de erros terminais de conteúdo do manifesto relatado no resumo do manifesto. O resumo do manifesto inclui um código de erro e uma mensagem para cada erro detectado. Para obter mais informações, consulte Noções básicas sobre o resumo do manifesto. Os erros terminais de conteúdo do manifesto não interrompem o relatório de Lista de erros não terminais de validação de linha JSON.

ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

Mensagem de erro

O arquivo de manifesto contém muitas linhas inválidas.

Mais informações

Ocorre um erro `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST` se houver muitas linhas JSON com conteúdo inválido.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir um erro `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST`.

Para corrigir ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

1. Verifique se há erros de linha JSON no manifesto. Para obter mais informações, consulte Noções básicas sobre treinar e testar manifestos de resultados de validação.
2. Corrija linhas JSON que contêm erros. Para obter mais informações, consulte Erros não terminais de validação de linha JSON.

ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS

Mensagem de erro

O arquivo de manifesto contém imagens de vários buckets do S3.

Mais informações

Um manifesto só pode referenciar imagens armazenadas em um único bucket. Cada linha JSON armazena a localização de uma imagem no Amazon S3 no valor de `source-ref`. No exemplo a seguir, o nome do bucket é my-bucket.

```
"source-ref": "s3://my-bucket/images/sunrise.png"
```

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Para corrigir **`ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS`**

- Certifique-se de que todas as suas imagens estejam no mesmo bucket do Amazon S3 e que o valor de `source-ref` em cada linha JSON faça referência ao bucket em que suas imagens estão armazenadas. Como alternativa, escolha um bucket preferido do Amazon S3 e remova as linhas JSON em que o `source-ref` não faz referência ao seu bucket preferido.

ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET

Mensagem de erro

As permissões para o bucket do S3 de imagens são inválidas.

Mais informações

As permissões do bucket do Amazon S3 que contém as imagens estão incorretas.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Para corrigir **`ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET`**

- Verifique as permissões do bucket que contém as imagens. O valor de `source-ref` para uma imagem contém a localização do bucket.

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

Mensagem de erro

ID de proprietário inválido para imagens do bucket S3.

Mais informações

O proprietário do bucket que contém as imagens de treinamento ou teste é diferente do proprietário do bucket que contém o manifesto de treinamento ou teste. É possível usar o seguinte comando para encontrar o dono de um bucket.

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

O `OWNER ID` deve corresponder aos buckets que armazenam as imagens e os arquivos de manifesto.

Para corrigir ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

1. Escolha o proprietário desejado dos buckets de treinamento, teste, saída e imagem. O proprietário deve ter permissões para usar o Amazon Rekognition Custom Labels.
2. Para cada bucket que não seja atualmente de propriedade do proprietário desejado, crie um novo bucket do Amazon S3 de propriedade do proprietário preferencial.
3. Copie o conteúdo antigo do bucket para o novo bucket. Para obter mais informações, consulte Como copiar objetos entre buckets do Amazon S3?.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

Mensagem de erro

O arquivo de manifesto contém imagens rotuladas insuficientes por rótulo para realizar a divisão automática.

Mais informações

Durante o treinamento do modelo, é possível criar um conjunto de dados de teste usando 20% das imagens do conjunto de dados de treinamento. ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT ocorre quando não há imagens suficientes para criar um conjunto de dados de teste aceitável.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Para corrigir ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

- Adicione mais imagens rotuladas ao seu conjunto de dados de treinamento. É possível adicionar imagens no console do Amazon Rekognition Custom Labels adicionando imagens ao conjunto de dados de treinamento ou adicionando linhas JSON ao seu manifesto de treinamento. Para obter mais informações, consulte Como gerenciar conjuntos de dados.

ERROR_MANIFEST_TOO_FEW_LABELS

Mensagem de erro

O arquivo de manifesto tem poucos rótulos.

Mais informações

Os conjuntos de dados de treinamento e teste exigem um número mínimo de rótulos. O mínimo depende se o conjunto de dados treina/testa um modelo para detectar rótulos em nível de imagem (classificação) ou se o modelo detecta localizações de objetos. Se o conjunto de dados de treinamento for dividido para criar um conjunto de dados de teste, o número de rótulos será determinado após a divisão do conjunto de dados de treinamento. Para obter mais informações, consulte Diretrizes e cotas no Amazon Rekognition Custom Labels.

Para corrigir ERROR_MANIFEST_TOO_FEW_LABELS (console)

1. Adicione mais rótulos novos ao conjunto de dados. Para obter mais informações, consulte Como gerenciar rótulos.
2. Adicione os novos rótulos às imagens no conjunto de dados. Se seu modelo detectar rótulos em nível de imagem, consulte Como atribuir rótulos em nível de imagem em uma imagem. Se seu modelo detectar localizações de objetos, consulte the section called “Como rotular objetos com caixas delimitadoras”.

Para corrigir ERROR_MANIFEST_TOO_FEW_LABELS (linha JSON)

- Adicione linhas JSON para novas imagens que tenham novos rótulos. Para obter mais informações, consulte Criar um arquivo de manifesto. Se seu modelo detectar rótulos em nível de imagem, você adicionará novos nomes de rótulos ao campo `class-name`. Por exemplo, o rótulo da imagem a seguir é Nascer do sol.

```
{
    "source-ref": "s3://bucket/images/sunrise.png",
    "testdataset-classification_Sunrise": 1,
    "testdataset-classification_Sunrise-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/testdataset-classification_Sunrise",
        "class-name": "Sunrise",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256",
        "type": "groundtruth/image-classification"
    }
}
```

  Se o seu modelo detectar a localização de objetos, adicione novos rótulos ao `class-map`, conforme mostrado no exemplo a seguir.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
```

```
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

É necessário mapear a tabela do mapa de classes para as anotações da caixa delimitadora. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

ERROR_MANIFEST_TOO_MANY_LABELS

Mensagem de erro

O arquivo de manifesto tem muitos rótulos.

Mais informações

O número de rótulos exclusivos no manifesto (conjunto de dados) é maior do que o limite permitido. Se o conjunto de dados de treinamento for dividido para criar um conjunto de dados de teste, o número de rótulos será determinado após a divisão.

Para corrigir ERROR_MANIFEST_TOO_MANY_LABELS (console)

- Remova os rótulos do conjunto de dados. Para obter mais informações, consulte Como gerenciar rótulos. Os rótulos são removidos automaticamente das imagens e das caixas delimitadoras em seu conjunto de dados.

Para corrigir ERROR_MANIFEST_TOO_MANY_LABELS (linha JSON)

- Manifestos com linhas JSON em nível de imagem: se a imagem tiver um único rótulo, remova a linha JSON das imagens que usa o rótulo desejado. Se a linha JSON contiver vários rótulos, remova somente o objeto JSON do rótulo desejado. Para obter mais informações, consulte Como adicionar vários rótulos em nível de imagem a uma imagem.

  Manifestos com linhas JSON de localização do objeto: remova a caixa delimitadora e as informações de rótulo associadas ao rótulo que você deseja remover. Faça isso para cada linha JSON que contém o rótulo desejado. É necessário remover o rótulo da matriz `class-map` e os objetos correspondentes na matriz `objects` e `annotations`. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

ERROR_INSUFFICIENT_LABEL_OVERLAP

Mensagem de erro

Menos de {}% de sobreposição de rótulos entre os arquivos de manifesto de treinamento e teste.

Mais informações

Há menos de 50% de sobreposição entre os nomes dos rótulos do conjunto de dados de teste e os nomes dos rótulos do conjunto de dados de treinamento.

Para corrigir ERROR_INSUFFICIENT_LABEL_OVERLAP (console)

- Remova os rótulos do conjunto de dados de treinamento. Como alternativa, adicione rótulos mais comuns ao seu conjunto de dados de teste. Para obter mais informações, consulte Como gerenciar rótulos. Os rótulos são removidos automaticamente das imagens e das caixas delimitadoras em seu conjunto de dados.

Para corrigir ERROR_INSUFFICIENT_LABEL_OVERLAP removendo rótulos do conjunto de dados de treinamento (linha JSON)

- Manifestos com linhas JSON em nível de imagem: se a imagem tiver um único rótulo, remova a linha JSON da imagem que usa o rótulo desejado. Se a linha JSON contiver vários rótulos, remova somente o objeto JSON do rótulo desejado. Para obter mais informações, consulte Como adicionar vários rótulos em nível de imagem a uma imagem. Faça isso para cada linha JSON no manifesto que contém o rótulo que você deseja remover.

  Manifestos com linhas JSON de localização do objeto: remova a caixa delimitadora e as informações de rótulo associadas ao rótulo que você deseja remover. Faça isso para cada linha JSON que contém o rótulo desejado. É necessário remover o rótulo da matriz `class-map` e os objetos correspondentes na matriz `objects` e `annotations`. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

Para corrigir ERROR_INSUFFICIENT_LABEL_OVERLAP adicionando rótulos comuns ao conjunto de dados de teste (linha JSON)

- Adicione linhas JSON ao conjunto de dados de teste que incluem imagens rotuladas com rótulos que já estão no conjunto de dados de treinamento. Para obter mais informações, consulte Criar um arquivo de manifesto.

ERROR_MANIFEST_TOO_FEW_USABLE_LABELS

Mensagem de erro

O arquivo de manifesto tem poucos rótulos utilizáveis.

Mais informações

Um manifesto de treinamento pode conter linhas JSON no formato de rótulo no nível da imagem e no formato de localização do objeto. Dependendo do tipo encontrado no manifesto de treinamento, o Amazon Rekognition Custom Labels escolhe criar um modelo que detecta rótulos em nível de imagem ou um modelo que detecta a localização dos objetos. O Amazon Rekognition Custom Labels filtra registros JSON válidos para linhas JSON que não estão no formato escolhido. ERROR_MANIFEST_TOO_FEW_USABLE_LABELS ocorre quando o número de rótulos no manifesto do tipo de modelo escolhido é insuficiente para treinar o modelo.

É necessário um mínimo de um rótulo para treinar um modelo que detecta rótulos em nível de imagem. É necessário um mínimo de dois rótulos para treinar um modelo que localize o objeto.

Para corrigir ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (console)

1. Verifique o campo `use_case` no resumo do manifesto.
2. Adicione mais rótulos ao conjunto de dados de treinamento para o caso de uso (nível da imagem ou localização do objeto) que corresponda ao valor de `use_case`. Para obter mais informações, consulte Como gerenciar rótulos. Os rótulos são removidos automaticamente das imagens e das caixas delimitadoras em seu conjunto de dados.

Para corrigir ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (linha JSON)

1. Verifique o campo `use_case` no resumo do manifesto.
2. Adicione mais rótulos ao conjunto de dados de treinamento para o caso de uso (nível da imagem ou localização do objeto) que corresponda ao valor de `use_case`. Para obter mais informações, consulte Criar um arquivo de manifesto.

ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP

Mensagem de erro

Menos de {}% de sobreposição de rótulos utilizáveis entre os arquivos de manifesto de treinamento e teste.

Mais informações

Um manifesto de treinamento pode conter linhas JSON no formato de rótulo no nível da imagem e no formato de localização do objeto. Dependendo dos formatos encontrados no manifesto de treinamento, o Amazon Rekognition Custom Labels escolhe criar um modelo que detecta rótulos em nível de imagem ou um modelo que detecta a localização dos objetos. O Amazon Rekognition Custom Labels não usa registros JSON válidos para linhas JSON que não estão no formato escolhido. ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP ocorre quando há menos de 50% de sobreposição entre os rótulos de teste e treinamento usados.

Para corrigir ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP (console)

- Remova os rótulos do conjunto de dados de treinamento. Como alternativa, adicione rótulos mais comuns ao seu conjunto de dados de teste. Para obter mais informações, consulte Como gerenciar rótulos. Os rótulos são removidos automaticamente das imagens e das caixas delimitadoras em seu conjunto de dados.

Para corrigir ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP removendo rótulos do conjunto de dados de treinamento (linha JSON)

- Conjuntos de dados usados para detectar rótulos no nível da imagem: se a imagem tiver um único rótulo, remova a linha JSON da imagem que usa o rótulo desejado. Se a linha JSON contiver vários rótulos, remova somente o objeto JSON do rótulo desejado. Para obter mais informações, consulte Como adicionar vários rótulos em nível de imagem a uma imagem. Faça isso para cada linha JSON no manifesto que contém o rótulo que você deseja remover.

  Conjuntos de dados usados para detectar localizações do objeto: remova a caixa delimitadora e as informações de rótulo associadas ao rótulo que você deseja remover. Faça isso para cada linha JSON que contém o rótulo desejado. É necessário remover o rótulo da matriz `class-map` e os objetos correspondentes na matriz `objects` e `annotations`. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

Para corrigir ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP adicionando rótulos comuns ao conjunto de dados de teste (linha JSON)

- Adicione linhas JSON ao conjunto de dados de teste que incluem imagens rotuladas com rótulos que já estão no conjunto de dados de treinamento. Para obter mais informações, consulte Criar um arquivo de manifesto.

ERROR_FAILED_IMAGES_S3_COPY

Mensagem de erro

Falha ao copiar imagens do bucket do S3.

Mais informações

O serviço não conseguiu copiar nenhuma das imagens em seu conjunto de dados.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Para corrigir ERROR_FAILED_IMAGES_S3_COPY

1. Verifique as permissões de suas imagens.
2. Se você estiver usando AWS KMS, verifique a política do bucket. Para obter mais informações, consulte Descriptografando arquivos criptografados com AWS Key Management Service.

O arquivo de manifesto tem muitos erros terminais.

Há muitas linhas JSON com erros terminais de conteúdo.

Para corrigir **`ERROR_TOO_MANY_RECORDS_IN_ERROR`**

- Reduza o número de linhas JSON (imagens) com erros terminais de conteúdo. Para obter mais informações, consulte Erros terminais de conteúdo do manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Erros não terminais de validação de linha JSON

Este tópico lista os erros não terminais de validação da linha JSON relatados pelo Amazon Rekognition Custom Labels durante o treinamento. Os erros são relatados no manifesto de validação de treinamento e teste. Para obter mais informações, consulte Noções básicas sobre treinar e testar manifestos de resultados de validação. É possível corrigir um erro não terminal de linha JSON atualizando a linha JSON no arquivo de manifesto de treinamento ou teste. Também é possível remover a linha JSON do manifesto, mas isso pode reduzir a qualidade do seu modelo. Se houver

muitos erros não terminais de validação, talvez seja mais fácil recriar o arquivo de manifesto. Normalmente, erros de validação ocorrem em arquivos de manifesto criados manualmente. Para obter mais informações, consulte Criar um arquivo de manifesto. Para obter informações sobre a correção de erros de validação, consulte Como corrigir erros de treinamento. Alguns erros podem ser corrigidos usando o console do Amazon Rekognition Custom Labels.

ERROR_MISSING_SOURCE_REF

Mensagem de erro

A chave source-ref está ausente.

Mais informações

O campo `source-ref` da linha JSON fornece a localização de uma imagem no Amazon S3. Este erro ocorre quando a chave `source-ref` está ausente ou é digitada incorretamente. Normalmente, este erro ocorre em um arquivo de manifesto criado manualmente. Para obter mais informações, consulte Criar um arquivo de manifesto.

Para corrigir **`ERROR_MISSING_SOURCE_REF`**

1. Verifique se a chave `source-ref` está presente e se está digitada corretamente. Uma chave `source-ref` e um valor completos são semelhantes ao seguinte `"source-ref": "s3://bucket/path/image"`.
2. Atualize a chave `source-ref` na linha JSON. Como alternativa, remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_INVALID_SOURCE_REF_FORMAT

Mensagem de erro

O formato do valor source-ref é inválido.

Mais informações

A chave `source-ref` está presente na linha JSON, mas o esquema do caminho do Amazon S3 está incorreto. Por exemplo, o caminho é `https://...` em vez de `S3://...`. Um erro

ERROR_INVALID_SOURCE_REF_FORMAT normalmente ocorre em arquivos de manifesto criados manualmente. Para obter mais informações, consulte Criar um arquivo de manifesto.

Para corrigir **`ERROR_INVALID_SOURCE_REF_FORMAT`**

1. Verifique se o esquema é `"source-ref": "s3://bucket/path/image"`. Por exemplo, `"source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg"`.
2. Atualize ou remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este `ERROR_INVALID_SOURCE_REF_FORMAT`.

ERROR_NO_LABEL_ATTRIBUTES

Mensagem de erro

Nenhum atributo de rótulo encontrado.

Mais informações

O atributo do rótulo ou o nome da chave `-metadata` do atributo do rótulo (ou ambos) é inválido ou está ausente. No exemplo a seguir, `ERROR_NO_LABEL_ATTRIBUTES` ocorre sempre que a chave `bounding-box` ou `bounding-box-metadata` (ou ambas) está ausente. Para obter mais informações, consulte Criar um arquivo de manifesto.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
```

```
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
 }
```

Normalmente, ocorre um erro `ERROR_NO_LABEL_ATTRIBUTES` em um arquivo de manifesto criado manualmente. Para obter mais informações, consulte Criar um arquivo de manifesto.

Para corrigir **`ERROR_NO_LABEL_ATTRIBUTES`**

1. Verifique se as chaves de `-metadata` do identificador do atributo do rótulo e do identificador do atributo do rótulo estão presentes e se os nomes das chaves estão escritos corretamente.
2. Atualize ou remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir `ERROR_NO_LABEL_ATTRIBUTES`.

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

Mensagem de erro

O formato {} do atributo do rótulo é inválido.

Mais informações

O esquema da chave de atributo do rótulo está ausente ou é inválido. Um erro ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT geralmente ocorre em arquivos de manifesto criados manualmente. Para obter mais informações, consulte Criar um arquivo de manifesto.

Para corrigir **`ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`**

1. Verifique se a seção Linha JSON da chave de atributo do rótulo está correta. No exemplo de localização do objeto a seguir, os objetos `image_size` e `annotations` devem estar corretos. A chave do atributo do rótulo é nomeada `bounding-box`.

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }, {
  "class_id": 0,
  "top": 65,
  "left": 86,
  "width": 220,
  "height": 334
 }]
},
```

2. Atualize ou remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT

Mensagem de erro

O formato dos metadados do atributo do rótulo é inválido.

Mais informações

O esquema da chave de metadados do atributo do rótulo está ausente ou é inválido. Um erro ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT geralmente ocorre em arquivos de manifesto criados manualmente. Para obter mais informações, consulte Criar um arquivo de manifesto.

Para corrigir **`ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`**

1. Verifique se o esquema da linha JSON para a chave de metadados do atributo label é semelhante ao exemplo a seguir. A chave de metadados do atributo do rótulo é nomeada `bounding-box-metadata`.

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
 "class-map": {
  "0": "Echo",
  "1": "Echo Dot"
 },
 "type": "groundtruth/object-detection",
 "human-annotated": "yes",
 "creation-date": "2018-10-18T22:18:13.527256",
 "job-name": "my job"
}
```

2. Atualize ou remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_NO_VALID_LABEL_ATTRIBUTES

Mensagem de erro

Nenhum atributo de rótulo válido encontrado.

Mais informações

Nenhum atributo de rótulo válido foi encontrado na linha JSON. O Amazon Rekognition Custom Labels verifica tanto o atributo do rótulo quanto o identificador do atributo do rótulo. Um erro ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT geralmente ocorre em arquivos de manifesto criados manualmente. Para obter mais informações, consulte Criar um arquivo de manifesto.

Se uma linha JSON não estiver em um formato de manifesto de SageMaker IA compatível, os rótulos personalizados do Amazon Rekognition marcarão a linha JSON como inválida e um erro será relatado. `ERROR_NO_VALID_LABEL_ATTRIBUTES` Atualmente, o Amazon Rekognition Custom Labels é compatível com trabalhos de classificação e formatos de caixa delimitadora. Para obter mais informações, consulte Criar um arquivo de manifesto.

Para corrigir **`ERROR_NO_VALID_LABEL_ATTRIBUTES`**

1. Verifique se o JSON da chave do atributo do rótulo e dos metadados do atributo do rótulo está correto.
2. Atualize ou remova a linha JSON do arquivo de manifesto. Para obter mais informações, consulte the section called “Criar um arquivo de manifesto”.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE

Mensagem de erro

Uma ou mais caixas delimitadoras não têm um valor de confiança.

Mais informações

A chave de confiança está ausente para uma ou mais caixas delimitadoras de localização de objetos. A chave de confiança de uma caixa delimitadora está nos metadados do atributo de rótulo, conforme mostrado no exemplo a seguir. Um erro ERROR_MISSING_BOUNDING_BOX_CONFIDENCE geralmente ocorre em arquivos de manifesto criados manualmente. Para obter mais informações, consulte the section called “Localização de objetos em arquivos de manifesto”.

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
```

Para corrigir **`ERROR_MISSING_BOUNDING_BOX_CONFIDENCE`**

1. Verifique se a matriz `objects` no atributo do rótulo contém o mesmo número de chaves de confiança que há objetos na matriz `annotations` do atributo do rótulo.
2. Atualize ou remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_MISSING_CLASS_MAP_ID

Mensagem de erro

Um ou mais IDs de classe estão faltando no mapa de classe.

Mais informações

O objeto `class_id` em uma anotação (caixa delimitadora) não tem uma entrada correspondente no mapa da classe de metadados do atributo de rótulo (`class-map`). Para obter mais informações, consulte Localização de objetos em arquivos de manifesto. Um erro ERROR_MISSING_CLASS_MAP_ID geralmente ocorre em arquivos de manifesto criados manualmente.

Para corrigir ERROR_MISSING_CLASS_MAP_ID

1. Verifique se o valor `class_id` em cada objeto de anotação (caixa delimitadora) tem um valor correspondente na matriz `class-map`, conforme mostrado no exemplo a seguir. A matriz `annotations` e a matriz `class_map` devem ter o mesmo número de elementos.

   ```
   {
    "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
    "bounding-box": {
     "image_size": [{
      "width": 640,
   ```

```
    "height": 480,
    "depth": 3
   }],
   "annotations": [{
    "class_id": 1,
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
   }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
 }
```

2. Atualize ou remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_INVALID_JSON_LINE

Mensagem de erro

A linha JSON tem um formato inválido.

Mais informações

Um caractere inesperado foi encontrado na linha JSON. A linha JSON é substituída por uma nova linha JSON que contém somente as informações de erro. Um erro ERROR_INVALID_JSON_LINE normalmente ocorre em arquivos de manifesto criados manualmente. Para obter mais informações, consulte the section called “Localização de objetos em arquivos de manifesto”.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

Para corrigir **`ERROR_INVALID_JSON_LINE`**

1. Abra o arquivo de manifesto e navegue até a linha JSON em que ocorre o erro ERROR_INVALID_JSON_LINE.
2. Verifique se a linha JSON não contém caracteres inválidos e se os caracteres obrigatórios `;` ou `,` não estão faltando.
3. Atualize ou remova a linha JSON do arquivo de manifesto.

ERROR_INVALID_IMAGE

Mensagem de erro

A imagem é inválida. Verifique o caminho e/ou as propriedades da imagem do S3.

Mais informações

O arquivo referenciado por `source-ref` não é uma imagem válida. As possíveis causas incluem a proporção da imagem, o tamanho da imagem e o formato da imagem.

Para obter mais informações, consulte Diretrizes e cotas.

Para corrigir **`ERROR_INVALID_IMAGE`**

1. Verifique o seguinte:
   - A proporção da imagem é menor que 20:1.
   - O tamanho da imagem é maior do que 15 MB
   - A imagem está no formato PNG ou JPEG.
   - O caminho até a imagem em `source-ref` está correto.
   - A dimensão mínima da imagem é maior que 64 pixels x 64 pixels.

- A dimensão máxima da imagem é menor que 4096 pixels x 4096 pixels.

2. Atualize ou remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_INVALID_IMAGE_DIMENSION

Mensagem de erro

As dimensões da imagem não estão em conformidade com as dimensões permitidas.

Mais informações

A imagem referenciada por `source-ref` não está em conformidade com as dimensões de imagem permitidas. A dimensão mínima é 64 pixels. A dimensão máxima é de 4096 pixels. `ERROR_INVALID_IMAGE_DIMENSION` é relatado para imagens com caixas delimitadoras.

Para obter mais informações, consulte Diretrizes e cotas.

Para corrigir **`ERROR_INVALID_IMAGE_DIMENSION`** (console)

1. Atualize a imagem no bucket do Amazon S3 com dimensões que o Amazon Rekognition Custom Labels pode processar.
2. No console do Amazon Rekognition Custom Labels, faça o seguinte:

   a. Remova as caixas delimitadoras existentes da imagem.

   b. Adicione novamente as caixas delimitadoras à imagem.

   c. Salve as alterações.

   Para obter mais informações, Como rotular objetos com caixas delimitadoras.

Para corrigir **`ERROR_INVALID_IMAGE_DIMENSION`** (SDK)

1. Atualize a imagem no bucket do Amazon S3 com dimensões que o Amazon Rekognition Custom Labels pode processar.
2. Obtenha a linha JSON existente para a imagem ListDatasetEntrieschamando. Para o parâmetro de entrada `SourceRefContains`, especifique a localização do Amazon S3 e o nome do arquivo da imagem.

3. Ligue UpdateDatasetEntriese forneça a linha JSON para a imagem. Certifique-se de que o valor de `source-ref` corresponda à localização da imagem no bucket do Amazon S3. Atualize as anotações da caixa delimitadora para que correspondam às dimensões da caixa delimitadora necessárias para a imagem atualizada.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2013-11-18T02:53:27",
  "job-name": "my job"
 }
```

```
}
```

ERROR_INVALID_BOUNDING_BOX

Mensagem de erro

A caixa delimitadora tem valores fora do quadro.

Mais informações

As informações da caixa delimitadora especificam uma imagem que está fora do quadro da imagem ou contém valores negativos.

Para obter mais informações, consulte Diretrizes e cotas.

Para corrigir **`ERROR_INVALID_BOUNDING_BOX`**

1. Verifique os valores das caixas delimitadoras na matriz `annotations`.

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

2. Atualize, ou como alternativa, remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_NO_VALID_ANNOTATIONS

Mensagem de erro

Nenhuma anotação válida encontrada.

Mais informações

Nenhum dos objetos de anotação na linha JSON contém informações válidas da caixa delimitadora.

Para corrigir **`ERROR_NO_VALID_ANNOTATIONS`**

1. Atualize a matriz `annotations` para incluir objetos de caixa delimitadora válidos. Além disso, verifique se as informações correspondentes da caixa delimitadora (`confidence` e `class_map`) nos metadados do atributo do rótulo estão corretas. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [
     {
   "class_id": 1,    #annotation object
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [
  >{
```

```
    "confidence": 1          #confidence  object
   },
         {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",    #label
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
 }
```

2. Atualize, ou como alternativa, remova a linha JSON do arquivo de manifesto.

Não é possível usar o console do Amazon Rekognition Custom Labels para corrigir este erro.

ERROR_BOUNDING_BOX_TOO_SMALL

Mensagem de erro

A altura e a largura da caixa delimitadora são muito pequenas.

Mais informações

As dimensões da caixa delimitadora (altura e largura) devem ser maiores que 1 x 1 pixels.

Durante o treinamento, o Amazon Rekognition Custom Labels redimensiona uma imagem se alguma de suas dimensões for maior que 1.280 pixels (as imagens de origem não são afetadas). As alturas e larguras das caixas delimitadoras resultantes devem ser maiores que 1 x 1 pixels. A localização da caixa delimitadora é armazenada na matriz `annotations` de uma linha JSON de localização do objeto. Para ter mais informações, consulte Localização de objetos em arquivos de manifesto

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
```

```
    "class_id": 1,
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
   }]
  },
```

As informações do erro são adicionadas ao objeto de anotação.

Para corrigir ERROR_BOUNDING_BOX_TOO_SMALL

- Escolha uma das seguintes opções.
  - Aumente o tamanho das caixas delimitadoras que são muito pequenas.
  - Remova as caixas delimitadoras que são muito pequenas. Para obter informações sobre como remover uma caixa delimitadora, consulte ERROR_TOO_MANY_BOUNDING_BOXES.
  - Remova a imagem (linha JSON) do manifesto.

ERROR_TOO_MANY_BOUNDING_BOXES

Mensagem de erro

Há mais caixas delimitadoras do que o máximo permitido.

Mais informações

Há mais caixas delimitadoras do que o limite permitido (50). É possível remover o excesso de caixas delimitadoras no console do Amazon Rekognition Custom Labels ou pode removê-las da linha JSON.

Para corrigir **`ERROR_TOO_MANY_BOUNDING_BOXES`** (console).

1. Decida quais caixas delimitadoras remover.
2. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
3. Escolha Usar rótulos personalizados.
4. Escolha Começar.
5. No painel de navegação esquerdo, selecione o projeto que contém o conjunto de dados que você deseja usar.

6. Na seção Conjuntos de dados, selecione o conjunto de dados que desejar usar.
7. Na página da galeria do conjunto de dados, escolha Iniciar rotulagem para entrar no modo de rotulagem.
8. Escolha a imagem da qual você deseja remover as caixas delimitadoras.
9. Escolha Desenhar caixa delimitadora.
10. Na ferramenta de desenho, selecione a caixa delimitadora que deseja excluir.
11. Pressione a tecla "delete" no teclado para excluir a caixa delimitadora.
12. Repita as duas etapas anteriores até excluir caixas delimitadoras suficientes.
13. Escolha Concluído
14. Escolha Salvar alterações para salvar suas alterações.
15. Escolha Sair para sair do modo de rotulagem.

Para corrigir ERROR_TOO_MANY_BOUNDING_BOXES (linha JSON).

1. Abra o arquivo de manifesto e navegue até a linha JSON em que ocorre o erro ERROR_TOO_MANY_BOUNDING_BOXES.
2. Remova o seguinte para cada caixa delimitadora que você deseja remover.
   - Remova o objeto necessário `annotation` da matriz `annotations`.
   - Remova o objeto `confidence` correspondente da matriz `objects` nos metadados de atributo do rótulo.
   - Se não for mais usado por outras caixas delimitadoras, remova o rótulo do `class-map`.

   Use o exemplo a seguir para identificar quais itens remover.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [
     {
```

```
    "class_id": 1,    #annotation object
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
   }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [
   >{
    "confidence": 1          #confidence  object
   },
         {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",    #label
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
 }
```

WARNING_UNANNOTATED_RECORD

Mensagem de aviso

O registro não está anotado.

Mais informações

Uma imagem adicionada a um conjunto de dados usando o console do Amazon Rekognition Custom Labels não foi rotulada. A linha JSON da imagem não é usada para treinamento.

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "warnings": [
        {
            "code": "WARNING_UNANNOTATED_RECORD",
            "message": "Record is unannotated."
        }
    ]
}
```

Para corrigir WARNING_UNANNOTATED_RECORD

- Rotule a imagem usando o console do Amazon Rekognition Custom Labels. Para obter instruções, consulte Como atribuir rótulos em nível de imagem em uma imagem.

WARNING_NO_ANNOTATIONS

Mensagem de aviso

Nenhuma anotação fornecida.

Mais informações

Uma linha JSON no formato de localização de objetos não contém nenhuma informação da caixa delimitadora, apesar de ser anotada por um humano (`human-annotated = yes`). A linha JSON é válida, mas não é usada para treinamento. Para obter mais informações, consulte Noções básicas sobre treinar e testar manifestos de resultados de validação.

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "bounding-box": {
        "image_size": [
            {
                "width": 640,
                "height": 480,
                "depth": 3
            }
        ],
        "annotations": [
```

```
        ],
        "warnings": [
            {
                "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
                "message": "No attribute annotations were found."
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [

        ],
        "class-map": {

        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2013-11-18 02:53:27",
        "job-name": "my job"
    },
    "warnings": [
        {
            "code": "WARNING_NO_ANNOTATIONS",
            "message": "No annotations were found."
        }
    ]
}
```

Para corrigir WARNING_NO_ANNOTATIONS

- Escolha uma das seguintes opções.
  - Adicione as informações da caixa delimitadora (`annotations`) à linha JSON. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.
  - Remova a imagem (linha JSON) do manifesto.

WARNING_NO_ATTRIBUTE_ANNOTATIONS

Mensagem de aviso

Nenhuma anotação de atributo fornecida.

Mais informações

Uma linha JSON no formato de localização de objetos não contém nenhuma informação de anotação da caixa delimitadora, apesar de ser anotada por um humano (`human-annotated = yes`). A matriz `annotations` não está presente ou não está preenchida. A linha JSON é válida, mas não é usada para treinamento. Para obter mais informações, consulte Noções básicas sobre treinar e testar manifestos de resultados de validação.

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "bounding-box": {
        "image_size": [
            {
                "width": 640,
                "height": 480,
                "depth": 3
            }
        ],
        "annotations": [

        ],
        "warnings": [
            {
                "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
                "message": "No attribute annotations were found."
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [

        ],
        "class-map": {

        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
```

```
        "creation-date": "2013-11-18 02:53:27",
        "job-name": "my job"
    },
    "warnings": [
        {
            "code": "WARNING_NO_ANNOTATIONS",
            "message": "No annotations were found."
        }
    ]
}
```

Para corrigir WARNING_NO_ATTRIBUTE_ANNOTATIONS

- Escolha uma das seguintes opções.
  - Adicione um ou mais objetos de `annotation` da caixa delimitadora à linha JSON. Para obter mais informações, consulte Localização de objetos em arquivos de manifesto.
  - Remova o atributo da caixa delimitadora.
  - Remova a imagem (linha JSON) do manifesto. Se existirem outros atributos de caixa delimitadora válidos na linha JSON, é possível remover apenas o atributo inválido da caixa delimitadora da linha JSON.

ERROR_UNSUPPORTED_USE_CASE_TYPE

Mensagem de aviso

Mais informações

O valor do campo `type` não é `groundtruth/image-classification` ou `groundtruth/object-detection`. Para obter mais informações, consulte Criar um arquivo de manifesto.

```
{
    "source-ref": "s3://bucket/test_normal_8.jpg",
    "BB": {
        "annotations": [
            {
                "left": 1768,
                "top": 1007,
                "width": 448,
                "height": 295,
                "class_id": 0
```

```
            },
            {
                "left": 1794,
                "top": 1306,
                "width": 432,
                "height": 411,
                "class_id": 1
            },
            {
                "left": 2568,
                "top": 1346,
                "width": 710,
                "height": 305,
                "class_id": 2
            },
            {
                "left": 2571,
                "top": 1020,
                "width": 644,
                "height": 312,
                "class_id": 3
            }
        ],
        "image_size": [
            {
                "width": 4000,
                "height": 2667,
                "depth": 3
            }
        ]
    },
    "BB-metadata": {
        "job-name": "labeling-job/BB",
        "class-map": {
            "0": "comparator",
            "1": "pot_resistor",
            "2": "ir_phototransistor",
            "3": "ir_led"
        },
        "human-annotated": "yes",
        "objects": [
            {
                "confidence": 1
            },
```

```
            {
                "confidence": 1
            },
            {
                "confidence": 1
            },
            {
                "confidence": 1
            }
        ],
        "creation-date": "2021-06-22T09:58:34.811Z",
        "type": "groundtruth/wrongtype",
        "cl-errors": [
            {
                "code": "ERROR_UNSUPPORTED_USE_CASE_TYPE",
                "message": "The use case type of the BB-metadata label attribute
 metadata is unsupported. Check the type field."
            }
        ]
    },
    "cl-metadata": {
        "is_labeled": true
    },
    "cl-errors": [
        {
            "code": "ERROR_NO_VALID_LABEL_ATTRIBUTES",
            "message": "No valid label attributes found."
        }
    ]
}
```

Para corrigir ERROR_UNSUPPORTED_USE_CASE_TYPE

- Escolha uma das seguintes opções:
  - Altere o valor do campo `type` para `groundtruth/image-classification` ou`groundtruth/object-detection`, dependendo do tipo de modelo que você deseja criar. Para obter mais informações, consulte Criar um arquivo de manifesto.
  - Remova a imagem (linha JSON) do manifesto.

ERROR_INVALID_LABEL_NAME_LENGTH

Mais informações

O tamanho do nome de um rótulo é muito longo. O tamanho máximo é de 256 caracteres.

Para corrigir ERROR_INVALID_LABEL_NAME_LENGTH

- Escolha uma das seguintes opções:
  - Reduza o tamanho do nome do rótulo para 256 caracteres ou menos.
  - Remova a imagem (linha JSON) do manifesto.

Como melhorar um modelo treinado do Amazon Rekognition Custom Labels

Quando o treinamento é concluído, você avalia o desempenho do modelo. Para ajudar você, o Amazon Rekognition Custom Labels fornece métricas resumidas e métricas de avaliação para cada rótulo. Para obter informações sobre as métricas disponíveis, consulte Métricas para avaliar seu modelo. Para melhorar seu modelo usando métricas, consulte Como melhorar um modelo do Amazon Rekognition Custom Labels.

Se estiver satisfeito com a precisão do seu modelo, poderá começar a usá-lo. Para obter mais informações, consulte Como executar um modelo do Amazon Rekognition Custom Labels.

Tópicos

- Métricas para avaliar seu modelo
- Como acessar as métricas de avaliação (console)
- Como acessar as métricas de avaliação (SDK) do Amazon Rekognition Custom Labels
- Como melhorar um modelo do Amazon Rekognition Custom Labels

Métricas para avaliar seu modelo

Depois que seu modelo é treinado, o Amazon Rekognition Custom Labels retorna métricas de testes de modelo que podem ser usadas para avaliar o desempenho do seu modelo. Este tópico descreve as métricas disponíveis para você e como entender se seu modelo treinado está funcionando bem.

O console do Amazon Rekognition Custom Labels fornece as seguintes métricas como um resumo dos resultados do treinamento e como métricas para cada rótulo:

- Precisão
- Recall
- F1

Cada métrica que fornecemos é uma métrica comumente usada para avaliar o desempenho de um modelo de machine learning. O Amazon Rekognition Custom Labels retorna métricas para os resultados dos testes em todo o conjunto de dados de teste, junto com métricas para cada rótulo

personalizado. Também é possível analisar o desempenho do seu modelo personalizado treinado para cada imagem em seu conjunto de dados de teste. Para obter mais informações, consulte Como acessar as métricas de avaliação (console).

Como avaliar o desempenho do modelo

Durante o teste, o Amazon Rekognition Custom Labels prevê se uma imagem de teste contém um rótulo personalizado. A pontuação de confiança é um valor que quantifica a certeza da previsão do modelo.

Se a pontuação de confiança de um rótulo personalizado exceder o valor limite, a saída do modelo incluirá esse rótulo. As previsões podem ser categorizadas das seguintes formas:

- Verdadeiro positivo: o modelo do Amazon Rekognition Custom Labels prevê corretamente a presença de um rótulo personalizado em uma imagem de teste. Ou seja, o rótulo previsto também é um rótulo de "verdade fundamental" para esta imagem. Por exemplo, o Amazon Rekognition Custom Labels retorna corretamente um rótulo de bola de futebol quando uma bola de futebol está presente em uma imagem.
- Falso positivo: o modelo do Amazon Rekognition Custom Labels prevê incorretamente a presença de um rótulo personalizado em uma imagem de teste. Ou seja, o rótulo previsto não é um rótulo de "verdade fundamental" para a imagem. Por exemplo, o Amazon Rekognition Custom Labels retorna um rótulo de bola de futebol, mas não há nenhum rótulo de bola de futebol na "verdade absoluta" para essa imagem.
- Falso negativo: o modelo do Amazon Rekognition Custom Labels não prevê a presença de um rótulo personalizado na imagem, mas a "verdade fundamental" dessa imagem inclui esse rótulo. Por exemplo, o Amazon Rekognition Custom Labels não retorna um rótulo personalizado de "bola de futebol" para uma imagem que contém uma bola de futebol.
- Verdadeiro positivo: o modelo do Amazon Rekognition Custom Labels prevê corretamente a ausência de um rótulo personalizado em uma imagem de teste. Por exemplo, o Amazon Rekognition Custom Labels não retorna um rótulo de bola de futebol para uma imagem que não contém uma bola de futebol.

O console fornece acesso a valores verdadeiros positivos, falsos positivos e falsos negativos para cada imagem em seu conjunto de dados de teste. Para obter mais informações, consulte Como acessar as métricas de avaliação (console).

Estes resultados de previsão são usados para calcular as seguintes métricas para cada rótulo e um agregado para todo o conjunto de testes. As mesmas definições se aplicam às previsões feitas pelo modelo no nível da caixa delimitadora, com a distinção de que todas as métricas são calculadas sobre cada caixa delimitadora (previsão ou "verdade fundamental") em cada imagem de teste.

Interseção sobre união (IoU) e detecção de objetos

Interseção sobre União (IoU) mede a porcentagem de sobreposição entre duas caixas delimitadoras de objetos em sua área combinada. O intervalo é de 0 (menor sobreposição) a 1 (sobreposição completa). Durante o teste, uma caixa delimitadora prevista está correta quando o IoU da caixa delimitadora de "verdade fundamental" e da caixa delimitadora prevista é de pelo menos 0,5.

Limite assumido

O Amazon Rekognition Custom Labels calcula automaticamente um valor limite assumido (0-1) para cada um de seus rótulos personalizados. Você não pode definir o valor limite assumido para um rótulo personalizado. O limite assumido para cada rótulo é o valor acima do qual uma previsão é contada como verdadeiro ou falso positivo. Ele é definido com base no seu conjunto de dados de teste. O limite assumido é calculado com base na melhor pontuação F1 alcançada no conjunto de dados de teste durante o treinamento do modelo.

É possível obter o valor do limite assumido para um rótulo a partir dos resultados de treinamento do modelo. Para obter mais informações, consulte Como acessar as métricas de avaliação (console).

Normalmente, as alterações nos valores-limite assumidos são usadas para melhorar a precisão e o recall de um modelo. Para obter mais informações, consulte Como melhorar um modelo do Amazon Rekognition Custom Labels. Como não é possível definir o limite assumido de um modelo para um rótulo, é possível obter os mesmos resultados analisando uma imagem `DetectCustomLabels` e especificando o parâmetro de entrada `MinConfidence`. Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.

Precisão

O Amazon Rekognition Custom Labels fornece métricas de precisão para cada rótulo e uma métrica de precisão média para todo o conjunto de dados de teste.

A precisão é a fração das previsões corretas (verdadeiros positivos) sobre todas as previsões do modelo (verdadeiros e falsos positivos) no limite assumido para um rótulo individual. À medida que o

limite aumenta, o modelo pode fazer menos previsões. Em geral, entretanto, ele terá uma proporção maior de verdadeiros positivos sobre falsos positivos em comparação com um limite mais baixo. Os valores possíveis para precisão variam de 0 a 1 e os valores mais altos indicam uma maior precisão.

Por exemplo, quando o modelo prevê que uma bola de futebol está presente em uma imagem, com que frequência essa previsão está correta? Suponha que haja uma imagem com oito bolas de futebol e cinco pedras. Se o modelo prevê 9 bolas de futebol (oito previstas corretamente e um falso positivo), a precisão para este exemplo é 0,89. No entanto, se o modelo previu 13 bolas de futebol na imagem com oito previsões corretas e cinco incorretas, a precisão resultante será menor.

Para obter mais informações, consulte Precisão e recall.

Recall

O Amazon Rekognition Custom Labels fornece métricas de recall médio para cada rótulo e uma métrica de recall médio para todo o conjunto de dados de teste.

Recall é a fração dos rótulos do conjunto de testes que foram previstos corretamente acima do limite assumido. É uma medida da frequência com que o modelo pode prever corretamente um rótulo personalizado quando ele está realmente presente nas imagens do seu conjunto de testes. O intervalo para recall é de 0 a 1. Valores mais altos indicam um recall maior.

Por exemplo, se uma imagem contém oito bolas de futebol, quantas delas são detectadas corretamente? Neste exemplo, em que uma imagem tem oito bolas de futebol e cinco pedras, se o modelo detectar cinco das bolas de futebol, o valor de recall será 0,62. Se, após o retreinamento, o novo modelo detectar 9 bolas de futebol, incluindo todas as 8 que estavam presentes na imagem, o valor de recall será 1,0.

Para obter mais informações, consulte Precisão e recall.

F1

O Amazon Rekognition Custom Labels usa a métrica de pontuação F1 para medir o desempenho médio do modelo de cada rótulo e o desempenho médio do modelo de todo o conjunto de dados de teste.

O desempenho do modelo é uma medida agregada que leva em consideração a precisão e o recall de todos os rótulos (por exemplo, pontuação F1 ou precisão média). A pontuação de desempenho do modelo é um valor entre 0 e 1. Quanto maior o valor, melhor o desempenho do modelo em termos

de recall e precisão. Especificamente, o desempenho do modelo para tarefas de classificação é comumente medido pela pontuação F1. Esta pontuação é a média harmônica das pontuações de precisão e recall no limite assumido. Por exemplo, para um modelo com precisão de 0,9 e recall de 1,0, a pontuação F1 é 0,947.

Um valor alto para a pontuação F1 indica que o modelo está funcionando bem tanto em termos de precisão quanto de recall. Se o modelo não está funcionando bem, por exemplo, com uma baixa precisão de 0,30 e um alto recall de 1,0, a pontuação F1 é 0,46. Da mesma forma, se a precisão for alta (0,95) e o recall for baixo (0,20), a pontuação F1 será 0,33. Em ambos os casos, a pontuação F1 é baixa e indica problemas com o modelo.

Para obter mais informações, consulte Pontuação F1.

Uso de métricas do

Para um determinado modelo que você treinou e dependendo da sua aplicação, é possível fazer uma troca entre precisão e recall usando o parâmetro de entrada `MinConfidence` para `DetectCustomLabels`. Com um valor `MinConfidence` mais alto, você geralmente obtém uma maior precisão (previsões mais corretas de bolas de futebol), mas um menor recall (mais bolas de futebol reais serão perdidas). Com um valor `MinConfidence` menor, você obtém um maior recall (mais bolas de futebol reais previstas corretamente), mas uma menor precisão (mais dessas previsões estarão erradas). Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.

As métricas também informam sobre as etapas que é possível tomar para melhorar o desempenho do modelo, se necessário. Para obter mais informações, consulte Como melhorar um modelo do Amazon Rekognition Custom Labels.

> **Note**
>
> `DetectCustomLabels` retorna previsões que variam de 0 a 100, que correspondem à faixa métrica de 0-1.

Como acessar as métricas de avaliação (console)

Durante o teste, o modelo é avaliado quanto ao seu desempenho em relação ao conjunto de dados de teste. Os rótulos no conjunto de dados de teste são considerados como "verdade fundamental",

pois representam o que a imagem real representa. Durante o teste, o modelo faz previsões usando o conjunto de dados de teste. Os rótulos previstos são comparados com os rótulos de "verdade fundamental" e os resultados estão disponíveis na página de avaliação do console.

O console do Amazon Rekognition Custom Labels mostra métricas resumidas para todo o modelo e métricas para rótulos individuais. As métricas disponíveis no console são recall, precisão, pontuação F1, confiança e limite de confiança. Para obter mais informações, consulte Como melhorar um modelo treinado do Amazon Rekognition Custom Labels.

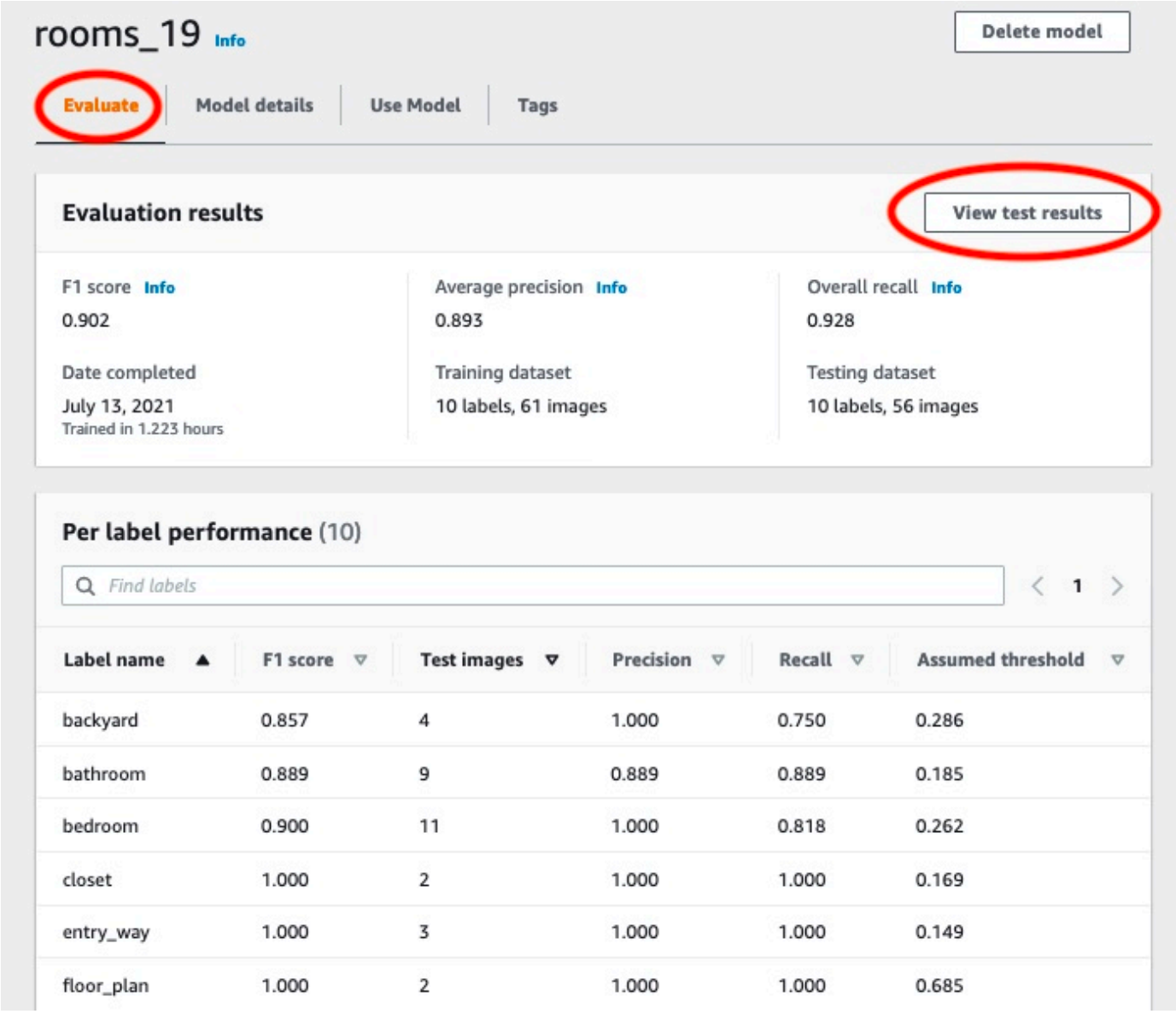
É possível usar o console para se concentrar em métricas individuais. Por exemplo, para investigar problemas de precisão em um rótulo, é possível filtrar os resultados do treinamento por rótulo e por resultados falsos positivos. Para obter mais informações, consulte Métricas para avaliar seu modelo.

Após o treinamento, o conjunto de dados de treinamento é somente para leitura. Se decidir melhorar o modelo, poderá copiar o conjunto de dados de treinamento para um novo conjunto de dados. A cópia do conjunto de dados é usada para treinar uma nova versão do modelo.

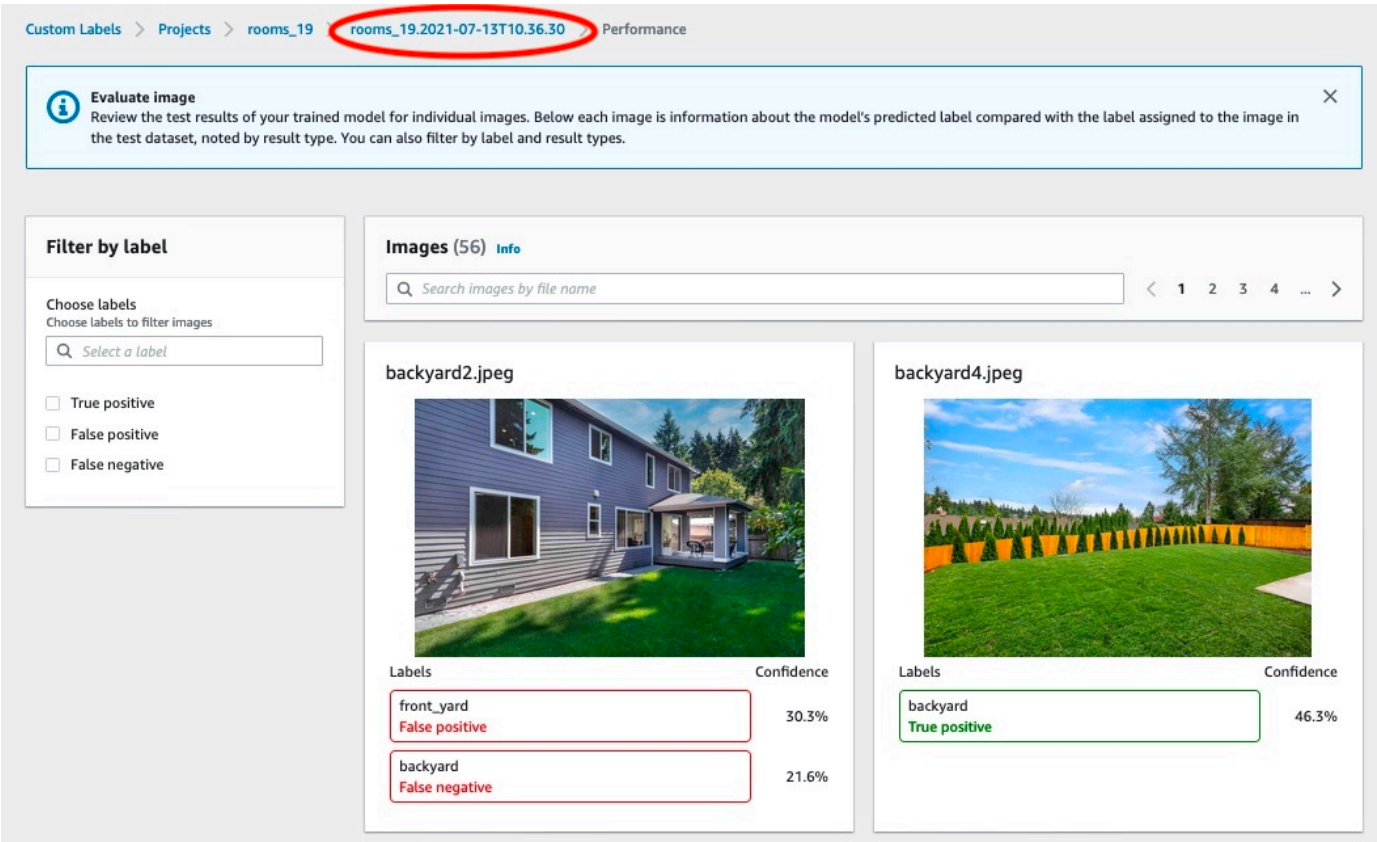
Nesta etapa, o console é usado para acessar os resultados do treinamento no console.

Para acessar as métricas de avaliação (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, escolha o projeto que contém o modelo treinado que deseja avaliar.
6. Na seção Modelos, escolha o modelo que deseja avaliar.
7. Escolha a guia Avaliar para ver os resultados da avaliação. Para obter informações sobre como avaliar um modelo, consulte Como melhorar um modelo treinado do Amazon Rekognition Custom Labels.
8. Escolha Exibir resultados do teste para ver os resultados de imagens de teste individuais. Para obter mais informações, consulte Métricas para avaliar seu modelo. A captura de tela a seguir do resumo da avaliação de modelo mostra a pontuação F1, a precisão média e o recall geral de 6 rótulos com resultados de testes e métricas de desempenho. Detalhes sobre o uso do modelo treinado também são fornecidos.



9. Depois de visualizar os resultados do teste, escolha o nome do projeto para retornar à página do modelo. A página de resultados do teste mostra imagens com rótulos previstos e pontuações de confiança para um modelo de machine learning treinado nas categorias de imagens de quintal e jardim da frente. Duas imagens de exemplo são exibidas.



10. Use as métricas para avaliar o desempenho do modelo. Para obter mais informações, consulte Como melhorar um modelo do Amazon Rekognition Custom Labels.

Como acessar as métricas de avaliação (SDK) do Amazon Rekognition Custom Labels

A DescribeProjectVersionsoperação fornece acesso a métricas além das fornecidas no console.

Assim como o console, `DescribeProjectVersions` fornece acesso às seguintes métricas como informações resumidas dos resultados do teste e como resultados do teste para cada rótulo:

- Precisão
- Recall
- F1

O limite médio para todas os rótulos e o limite para rótulos individuais são retornados.

`DescribeProjectVersions` também fornece acesso às seguintes métricas para a classificação e detecção de imagens (localização do objeto na imagem).

- Matriz de confusão para a classificação de imagens. Para obter mais informações, consulte Como visualizar a matriz de confusão para um modelo.
- Precisão média (mAP) para a detecção de imagens.
- Recall médio (mAR) para a detecção de imagens.

`DescribeProjectVersions` também fornece acesso a valores verdadeiros positivos, falsos positivos, falsos negativos e verdadeiros negativos. Para obter mais informações, consulte Métricas para avaliar seu modelo.

A métrica de pontuação F1 agregada é retornada diretamente por `DescribeProjectVersions`. Outras métricas podem ser acessadas de arquivos de Acessar o arquivo de resumo do modelo e Interpretar o snapshot do manifesto de avaliação armazenados em um bucket do Amazon S3. Para obter mais informações, consulte Como acessar o arquivo de resumo e o snapshot do manifesto de avaliação (SDK).

Tópicos

- Acessar o arquivo de resumo do modelo
- Interpretar o snapshot do manifesto de avaliação
- Como acessar o arquivo de resumo e o snapshot do manifesto de avaliação (SDK)
- Como visualizar a matriz de confusão para um modelo
- Referência: arquivo de resumo dos resultados do treinamento

Acessar o arquivo de resumo do modelo

O arquivo de resumo contém informações sobre os resultados da avaliação sobre o modelo como um todo e as métricas de cada rótulo. As métricas são precisão, recall, pontuação F1. O valor limite para o modelo também é fornecido. O local do arquivo de resumo pode ser acessado a partir do objeto `EvaluationResult` retornado por `DescribeProjectVersions`. Para obter mais informações, consulte Referência: arquivo de resumo dos resultados do treinamento.

Veja a seguir um exemplo de arquivo de resumo.

```
{
```

```
  "Version": 1,
  "AggregatedEvaluationResults": {
    "ConfusionMatrix": [
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "CAP",
        "Value": 0.9948717948717949
      },
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "WATCH",
        "Value": 0.00854700854700854 8
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "CAP",
        "Value": 0.1794871794871795
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "WATCH",
        "Value": 0.7008547008547008
      }
    ],
    "F1Score": 0.9726959470546408,
    "Precision": 0.9719115848331294,
    "Recall": 0.9735042735042735
  },
  "EvaluationDetails": {
    "EvaluationEndTimestamp": "2019-11-21T07:30:23.910943",
    "Labels": [
      "CAP",
      "WATCH"
    ],
    "NumberOfTestingImages": 624,
    "NumberOfTrainingImages": 5216,
    "ProjectVersionArn": "arn:aws:rekognition:us-east-1:nnnnnnnnn:project/my-project/
version/v0/1574317227432"
  },
  "LabelEvaluationResults": [
    {
      "Label": "CAP",
      "Metrics": {
        "F1Score": 0.9794344473007711,
```

```
        "Precision": 0.9819587628865979,
        "Recall": 0.9769230769230769,
        "Threshold": 0.9879502058029175
      },
      "NumberOfTestingImages": 390
    },
    {
      "Label": "WATCH",
      "Metrics": {
        "F1Score": 0.9659574468085106,
        "Precision": 0.961864406779661,
        "Recall": 0.9700854700854701,
        "Threshold": 0.014450683258473873
      },
      "NumberOfTestingImages": 234
    }
  ]
}
```

Interpretar o snapshot do manifesto de avaliação

O snapshot do manifesto de avaliação contém informações detalhadas sobre os resultados do teste. O snapshot inclui a classificação de confiança de cada previsão. Também inclui a classificação da previsão em comparação com a classificação real da imagem (verdadeiro positivo, verdadeiro negativo, falso positivo ou falso negativo).

Os arquivos são um snapshot, pois somente imagens que podem ser usadas para teste e treinamento estão incluídas. As imagens que não podem ser verificadas, como imagens no formato errado, não são incluídas no manifesto. O local do snapshot de teste pode ser acessado a partir do objeto `TestingDataResult` retornado por `DescribeProjectVersions`. O local do snapshot de treinamento pode ser acessado a partir do objeto `TrainingDataResult` retornado por `DescribeProjectVersions`.

O instantâneo está no formato de saída do manifesto SageMaker AI Ground Truth com campos adicionados para fornecer informações adicionais, como o resultado da classificação binária de uma detecção. O trecho a seguir mostra os campos adicionais.

```
"rekognition-custom-labels-evaluation-details": {
    "version": 1,
    "is-true-positive": true,
    "is-true-negative": false,
```

```
    "is-false-positive": false,
    "is-false-negative": false,
    "is-present-in-ground-truth": true
    "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"]
}
```

- versão: a versão do formato do `rekognition-custom-labels-evaluation-details` campo no snapshot do manifesto.
- is-true-positive... — A classificação binária da previsão com base em como a pontuação de confiança se compara ao limite mínimo do rótulo.
- is-present-in-ground-true — Verdadeiro se a previsão feita pelo modelo estiver presente nas informações verdadeiras básicas usadas para treinamento, caso contrário, falso. Este valor não se baseia no fato de a pontuação de confiança exceder o limite mínimo calculado pelo modelo.
- ground-truth-labeling-jobs— Uma lista dos campos de verdade fundamentais na linha do manifesto que são usados para treinamento.

Para obter informações sobre o formato de manifesto do SageMaker AI Ground Truth, consulte Output.

Veja a seguir um exemplo de um snapshot do manifesto de teste que mostra métricas para classificação de imagens e detecção de objetos.

```
// For image classification
{
  "source-ref": "s3://amzn-s3-demo-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": 1,
  "rekognition-custom-labels-training-0-metadata": {
    "confidence": 1.0,
    "job-name": "rekognition-custom-labels-training-job",
    "class-name": "Football",
    "human-annotated": "yes",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification"
  },
  "rekognition-custom-labels-evaluation-0": 1,
  "rekognition-custom-labels-evaluation-0-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-name": "Football",
    "human-annotated": "no",
```

```
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification",
    "rekognition-custom-labels-evaluation-details": {
      "version": 1,
      "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
      "is-true-positive": true,
      "is-true-negative": false,
      "is-false-positive": false,
      "is-false-negative": false,
      "is-present-in-ground-truth": true
    }
  }
}


// For object detection
{
  "source-ref": "s3://amzn-s3-demo-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-training-0-metadata": {
    "job-name": "rekognition-custom-labels-training-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "yes",
```

```
    "objects": [
      {
        "confidence": 1.0
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  },
  "rekognition-custom-labels-evaluation": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-evaluation-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "no",
    "objects": [
      {
        "confidence": 0.95,
        "rekognition-custom-labels-evaluation-details": {
          "version": 1,
          "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
          "is-true-positive": true,
          "is-true-negative": false,
```

```
          "is-false-positive": false,
          "is-false-negative": false,
          "is-present-in-ground-truth": true
        }
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  }
}
```

Como acessar o arquivo de resumo e o snapshot do manifesto de avaliação (SDK)

Para obter resultados de treinamento, você liga DescribeProjectVersions. Para obter um código de exemplo, consulte Como descrever um modelo (SDK).

A localização das métricas é retornada na resposta `ProjectVersionDescription` de `DescribeProjectVersions`.

- `EvaluationResult`: a localização do arquivo de resumo.
- `TestingDataResult`: a localização do instantâneo do manifesto de avaliação usado para testes.

A pontuação F1 e a localização do arquivo de resumo são retornadas em `EvaluationResult`. Por exemplo:

```
"EvaluationResult": {
                "F1Score": 1.0,
                "Summary": {
                    "S3Object": {
                        "Bucket": "echo-dot-scans",
                        "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
                    }
                }
            }
```

O snapshot do manifesto de avaliação é armazenado no local especificado no parâmetro de entrada `--output-config` que você especificou em Treinando um modelo (SDK).

> **Note**
>
> A quantidade de tempo, em segundos, que você recebe pelo treinamento é retornada em `BillableTrainingTimeInSeconds`.

Para obter informações sobre as métricas que são retornadas pelo Amazon Rekognition Custom Labels, consulte Como acessar as métricas de avaliação (SDK) do Amazon Rekognition Custom Labels.

Como visualizar a matriz de confusão para um modelo

Uma matriz de confusão permite que você veja os rótulos que seu modelo confunde com outros rótulos em seu modelo. Ao usar uma matriz de confusão, é possível focar suas melhorias no modelo.

Durante a avaliação do modelo, o Amazon Rekognition Custom Labels cria uma matriz de confusão usando as imagens de teste para identificar rótulos mal identificados (confusos). O Amazon Rekognition Custom Labels só criam uma matriz de confusão para modelos de classificação. A matriz de classificação pode ser acessada a partir do arquivo de resumo que o Amazon Rekognition Custom Labels cria durante o treinamento do modelo. Não é possível visualizar a matriz de confusão no console do Amazon Rekognition Custom Labels.

Tópicos

- Como usar uma matriz de confusão
- Como obter a matriz de confusão para um modelo

Como usar uma matriz de confusão

A tabela a seguir é a matriz de confusão do projeto de exemplo classificação de imagens de Cômodos. Os cabeçalhos das colunas são os rótulos (rótulos de "verdade fundamental") atribuídos às imagens de teste. Os cabeçalhos das linhas são os rótulos que o modelo prevê para as imagens de teste. Cada célula é a porcentagem de previsões de um rótulo (linha) que deve ser o rótulo de "verdade fundamental" (coluna). Por exemplo, 67% das previsões para banheiros foram rotuladas corretamente como banheiros. 33% por cento dos banheiros foram rotulados incorretamente como cozinhas. Um modelo de alto desempenho tem altos valores de células quando o rótulo previsto corresponde ao rótulo de "verdade fundamental". É possível vê-los como uma linha diagonal do primeiro ao último rótulo previsto e os rótulos de "verdade fundamental". Se o valor de uma célula for

0, nenhuma previsão foi feita para o rótulo previsto da célula, que deveria ser o rótulo de "verdade fundamental" da célula.

> **Note**
>
> Como os modelos não são determinísticos, os valores das células da matriz de confusão que você obtém ao treinar o projeto Cômodos podem ser diferentes da tabela a seguir.

A matriz de confusão identifica áreas nas quais focar. Por exemplo, a matriz de confusão mostra que 50% das vezes o modelo confundiu armários com quartos. Nesta situação, devem ser adicionadas mais imagens de armários e quartos ao seu conjunto de dados de treinamento. Verifique também se as imagens existentes do armário e do quarto estão corretamente rotuladas. Isto deve ajudar o modelo a distinguir melhor os dois rótulos. Para adicionar mais imagens a um conjunto de dados, consulte Como adicionar mais imagens a um conjunto de dados.

Embora a matriz de confusão seja útil, é importante saber outras métricas. Por exemplo, 100% das previsões encontraram corretamente o rótulo floor_plan, que indica um excelente desempenho. No entanto, o conjunto de dados de teste tem apenas duas imagens com o rótulo floor_plan. Ele também tem 11 imagens com o rótulo sala_de_estar. Este desequilíbrio também está no conjunto de dados de treinamento (13 imagens sala_de_estar e duas imagens de armário). Para obter uma avaliação mais precisa, equilibre os conjuntos de dados de treinamento e teste adicionando mais imagens de rótulos sub-representados (plantas baixas neste exemplo). Para obter o número de imagens de teste por rótulo, consulte Como acessar as métricas de avaliação (console).

A tabela a seguir é um exemplo de matriz de confusão, comparando o rótulo previsto (no eixo y) com o rótulo da verdade fundamental:

Rótulo previsto	quintal	banheiro	quarto	closet	entry_way	floor_plan	front_yard	kitchen	sala_de_estar	patio
quintal	75%	0%	0%	0%	0%	0%	33%	0%	0%	0%
banheiro	0%	67%	0%	0%	0%	0%	0%	0%	0%	0%
quarto	0%	0%	82%	50%	0%	0%	0%	0%	9%	0%
closet	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%

Rótulo previsto	quintal	banheiro	quarto	closet	entry_way	floor_plan	front_yard	kitchen	sala_de_estar	patio
entry_way	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%
floor_plan	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
front_yard	25%	0%	0%	0%	0%	0%	67%	0%	0%	0%
kitchen	0%	33%	0%	0%	0%	0%	0%	88%	0%	0%
sala_de_estar	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
patio	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

Como obter a matriz de confusão para um modelo

O código a seguir usa as DescribeProjectVersionsoperações DescribeProjectse para obter o arquivo de resumo de um modelo. Em seguida, ele usa o arquivo de resumo para exibir a matriz de confusão do modelo.

Para exibir a matriz de confusão de um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para exibir a matriz de confusão de um modelo. Forneça os seguintes argumentos de linha de comando:

   - `project_name`: o nome do projeto que você deseja usar. É possível obter o nome do projeto na página de projetos no console do Amazon Rekognition Custom Labels.
   - `version_name`: a versão do modelo que você deseja usar. É possível obter o nome da versão na página de detalhes do projeto no console do Amazon Rekognition Custom Labels.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose

Shows how to display the confusion matrix for an Amazon Rekognition Custom labels
 image
classification model.
"""


import json
import argparse
import logging
import boto3
import pandas as pd
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def get_model_summary_location(rek_client, project_name, version_name):
    """
    Get the summary file location for a model.

    :param rek_client: A Boto3 Rekognition client.
    :param project_arn: The Amazon Resource Name (ARN) of the project that contains
 the model.
    :param model_arn: The Amazon Resource Name (ARN) of the model.
    :return: The location of the model summary file.
    """

    try:
        logger.info(
            "Getting summary file for model %s in project %s.", version_name,
 project_name)

        summary_location = ""

        # Get the project ARN from the project name.
        response = rek_client.describe_projects(ProjectNames=[project_name])

        assert len(response['ProjectDescriptions']) > 0, \
            f"Project {project_name} not found."
```

```
        project_arn = response['ProjectDescriptions'][0]['ProjectArn']

        # Get the summary file location for the model.
        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        assert len(describe_response['ProjectVersionDescriptions']) > 0, \
            f"Model {version_name} not found."

        model=describe_response['ProjectVersionDescriptions'][0]

        evaluation_results=model['EvaluationResult']

        summary_location=(f"s3://{evaluation_results['Summary']['S3Object']
['Bucket']}"
                          f"/{evaluation_results['Summary']['S3Object']
['Name']}")

        return summary_location

    except ClientError as err:
        logger.exception(
            "Couldn't get summary file location: %s", err.response['Error']
['Message'])
        raise


def show_confusion_matrix(summary):
    """
    Shows the confusion matrix for an Amazon Rekognition Custom Labels
    image classification model.
    :param summary: The summary file JSON object.
    """
    pd.options.display.float_format = '{:.0%}'.format

    # Load the model summary JSON into a DataFrame.

    summary_df = pd.DataFrame(
        summary['AggregatedEvaluationResults']['ConfusionMatrix'])

    # Get the confusion matrix.
    confusion_matrix = summary_df.pivot_table(index='PredictedLabel',
```

```
                                                    columns='GroundTruthLabel',
                                                    fill_value=0.0).astype(float)

    # Display the confusion matrix.
    print(confusion_matrix)


def get_summary(s3_resource, summary):
    """
    Gets the summary file.
    : return: The summary file in bytes.
    """
    try:
        summary_bucket, summary_key = summary.replace(
            "s3://", "").split("/", 1)

        bucket = s3_resource.Bucket(summary_bucket)
        obj = bucket.Object(summary_key)
        body = obj.get()['Body'].read()
        logger.info(
            "Got summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't get summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
        raise
    else:
        return body


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    : param parser: The command line parser.
    """

    parser.add_argument(
        "project_name", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to describe."
    )
```

```
def main():
    """
    Entry point for script.
    """

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get the command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Showing confusion matrix for: {args.version_name} for project
 {args.project_name}.")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
        s3_resource = session.resource('s3')

        # Get the summary file for the model.
        summary_location = get_model_summary_location(rekognition_client,
 args.project_name,
                                                      args.version_name
                                                      )
        summary = json.loads(get_summary(s3_resource, summary_location))

        # Check that the confusion matrix is available.
        assert 'ConfusionMatrix' in summary['AggregatedEvaluationResults'], \
            "Confusion matrix not found in summary. Is the model a classification
 model?"

        # Show the confusion matrix.
        show_confusion_matrix(summary)
        print("Done")

    except ClientError as err:
        logger.exception("Problem showing confusion matrix: %s", err)
        print(f"Problem describing model: {err}")
```

```
    except AssertionError as err:
        logger.exception(
            "Error: %s.\n", err)
        print(
            f"Error: {err}\n")


if __name__ == "__main__":
    main()
```

Referência: arquivo de resumo dos resultados do treinamento

O resumo dos resultados do treinamento contém métricas que é possível usar para avaliar seu modelo. O arquivo de resumo também é usado para exibir métricas na página de resultados do treinamento do console. O arquivo de resumo é armazenado em um bucket do Amazon S3 após o treinamento. Para obter o arquivo de resumo, chame `DescribeProjectVersion`. Para obter um código de exemplo, consulte Como acessar o arquivo de resumo e o snapshot do manifesto de avaliação (SDK).

Arquivo de resumo

O JSON a seguir é o formato do arquivo de resumo.

EvaluationDetails (seção 3)

Informações gerais sobre a tarefa de treinamento. Isso inclui o ARN do projeto ao qual o modelo pertence (`ProjectVersionArn`)), a data e a hora em que o treinamento foi concluído, a versão do modelo que foi avaliada (`EvaluationEndTimestamp`) e uma lista de rótulos detectados durante o treinamento (`Labels`). Também está incluído o número de imagens usadas para treinamento (`NumberOfTrainingImages`) e avaliação (`NumberOfTestingImages`).

AggregatedEvaluationResults (seção 1)

É possível usar `AggregatedEvaluationResults` para avaliar o desempenho geral do modelo treinado quando usado com o conjunto de dados de teste. As métricas agregadas estão incluídas para as métricas `Precision`, `Recall` e `F1Score`. Para detecção de objetos (a localização do objeto em uma imagem), as métricas `AverageRecall` (mAR) e `AveragePrecision` (mAP) são retornadas. Para classificação (o tipo de objeto em uma imagem), uma métrica de matriz de confusão é retornada.

LabelEvaluationResults (seção 2)

É possível usar `labelEvaluationResults` para avaliar o desempenho de rótulos individuais. Os rótulos são classificados pela pontuação F1 de cada rótulo. As métricas incluídas são `Precision`, `Recall`, `F1Score` e `Threshold` (usadas para classificação).

O nome do arquivo é formatado da seguinte maneira: `EvaluationSummary-ProjectName-VersionName.json`.

```
{
  "Version": "integer",
  // section-3
  "EvaluationDetails": {
    "ProjectVersionArn": "string",
    "EvaluationEndTimestamp": "string",
    "Labels": "[string]",
    "NumberOfTrainingImages": "int",
    "NumberOfTestingImages": "int"
  },
  // section-1
  "AggregatedEvaluationResults": {
    "Metrics": {
      "Precision": "float",
      "Recall": "float",
      "F1Score": "float",
      // The following 2 fields are only applicable to object detection
      "AveragePrecision": "float",
      "AverageRecall": "float",
      // The following field is only applicable to classification
      "ConfusionMatrix":[
        {
          "GroundTruthLabel": "string",
          "PredictedLabel": "string",
          "Value": "float"
        },
        ...
      ],
    }
  },
  // section-2
  "LabelEvaluationResults": [
    {
```

```
      "Label": "string",
      "NumberOfTestingImages", "int",
      "Metrics": {
        "Threshold": "float",
        "Precision": "float",
        "Recall": "float",
        "F1Score": "float"
      },
    },
    ...
  ]
}
```

Como melhorar um modelo do Amazon Rekognition Custom Labels

O desempenho dos modelos de machine learning depende muito de fatores , como a complexidade e a variabilidade de seus rótulos personalizados (os objetos e cenas específicos nos quais você está interessado), a qualidade e o poder representativo do conjunto de dados de treinamento fornecido e as estruturas do modelo e os métodos de machine learning usados para treinar o modelo.

O Amazon Rekognition Custom Labels simplifica esse processo e não é necessário nenhum conhecimento em machine learning. No entanto, o processo de criação de um bom modelo geralmente envolve iterações sobre dados e melhorias no modelo para alcançar o desempenho desejado. Veja a seguir informações sobre como melhorar seu modelo.

Dados

Em geral, é possível melhorar a qualidade do seu modelo com quantidades maiores de dados de melhor qualidade. Use imagens de treinamento que mostrem claramente o objeto ou a cena e não estejam repletas de itens desnecessários. Para delimitar caixas ao redor de objetos, use imagens de treinamento que mostrem o objeto totalmente visível e não obstruído por outros objetos.

Certifique-se de que seus conjuntos de dados de treinamento e teste correspondam ao tipo de imagem sobre a qual você eventualmente executará a inferência. Para objetos, como logotipos, nos quais você tem apenas alguns exemplos de treinamento, forneça caixas delimitadoras ao redor do logotipo nas imagens de teste. Estas imagens representam ou retratam os cenários nos quais você deseja localizar o objeto.

Para adicionar imagens a um conjunto de dados de treinamento ou teste, consulte Como adicionar mais imagens a um conjunto de dados.

Como reduzir falsos positivos (melhor precisão)

- Primeiro, verifique se o aumento do limite assumido permite manter as previsões corretas e, ao mesmo tempo, diminuir os falsos positivos. Em algum momento, isso tem ganhos decrescentes devido à compensação entre precisão e recall de um determinado modelo. Não é possível definir o limite assumido para um rótulo, mas pode obter o mesmo resultado especificando um valor alto para o parâmetro de entrada `MinConfidence` para `DetectCustomLabels`. Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.
- Talvez veja um ou mais de seus rótulos personalizados de interesse (A) serem confundidos consistentemente com a mesma classe de objetos (mas não com um rótulo no qual você esteja interessado) (B). Para ajudar, adicione B como um rótulo de classe de objeto ao seu conjunto de dados de treinamento (junto com as imagens nas quais você obteve o falso positivo). Efetivamente, você está ajudando o modelo a aprender a prever B e não A por meio das novas imagens de treinamento. Para adicionar imagens a um conjunto de dados de treinamento, consulte Como adicionar mais imagens a um conjunto de dados.
- É possível descobrir que o modelo está confuso com dois de seus rótulos personalizados (A e B): prevê-se que a imagem de teste com o rótulo A tenha o rótulo B e vice-versa. Nesse caso, primeiro verifique se há imagens com rótulos incorretos em seus conjuntos de treinamento e teste. Use a galeria de conjuntos de dados para gerenciar os rótulos atribuídos a um conjunto de dados. Para obter mais informações, consulte Como gerenciar rótulos. Além disso, adicionar mais imagens de treinamento relacionadas a esse tipo de confusão ajudará um modelo retreinado a discriminar melhor entre A e B. Para adicionar imagens a um conjunto de dados de treinamento, consulte Como adicionar mais imagens a um conjunto de dados.

Como reduzir falsos negativos (melhor recall)

- Use um valor menor para o limite assumido. Não é possível definir o limite assumido para um rótulo, mas pode obter o mesmo resultado especificando um valor menor para o parâmetro de entrada `MinConfidence` para `DetectCustomLabels`. Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.
- Use exemplos melhores para modelar a variedade do objeto e das imagens nas quais eles aparecem.
- Divida seu rótulo em duas classes que sejam mais fáceis de aprender. Por exemplo, em vez de biscoitos bons e ruins, é possível querer biscoitos bons, biscoitos queimados e biscoitos quebrados para ajudar o modelo a aprender melhor cada conceito exclusivo.

Como executar um modelo do Amazon Rekognition Custom Labels

Se estiver satisfeito com o desempenho do modelo, poderá começar a usá-lo. Você pode iniciar e interromper um modelo usando o console ou o AWS SDK. O console também inclui exemplos de operações do SDK que podem ser usados.

Tópicos

- Unidades de inferência
- Zonas de disponibilidade
- Como iniciar um modelo do Amazon Rekognition Custom Labels
- Como interromper um modelo do Amazon Rekognition Custom Labels
- Duração do relatório de execução e unidades de inferência usadas

Unidades de inferência

Ao iniciar seu modelo, especifique o número de recursos computacionais, conhecidos como unidade de inferência, que o modelo usa.

> **Important**
>
> Há uma cobrança pelo número de horas em que seu modelo está em execução e pelo número de unidades de inferência que seu modelo usa enquanto está em execução com base em como a execução do seu modelo é configurada. Por exemplo, se você iniciar o modelo com duas unidades de inferência e usar o modelo por oito horas, haverá uma cobrança por 16 horas de inferência (oito horas de tempo de execução * duas unidades de inferência). Para obter mais informações, consulte Horas de inferência. Se não interromper seu modelo explicitamente, haverá uma cobrança mesmo que não esteja analisando ativamente as imagens com seu modelo.

As transações por segundo (TPS) compatíveis com uma única unidade de inferência são afetadas por:

- Um modelo que detecta rótulos em nível de imagem (classificação) geralmente tem um TPS maior do que um modelo que detecta e localiza objetos com caixas delimitadoras (detecção de objetos).
- A complexidade do modelo.
- Uma imagem de alta resolução requer mais tempo para análise.
- Mais objetos em uma imagem exigem mais tempo para análise.
- Imagens menores são analisadas mais rapidamente do que imagens maiores.
- Uma imagem passada como bytes de imagem é analisada mais rapidamente do que primeiro fazer upload da imagem em um bucket do Amazon S3 e referenciar a imagem carregada. As imagens passadas como bytes de imagem devem ter menos de 4,0 MB. É recomendável usar bytes de imagem para processamento de imagens quase em tempo real e quando o tamanho da imagem for menor que 4,0 MB. Por exemplo, imagens capturadas de uma câmera IP.
- O processamento de imagens armazenadas em um bucket do Amazon S3 é mais rápido do que baixar as imagens, convertê-las em bytes de imagem e passar os bytes da imagem para análise.
- Analisar uma imagem já armazenada em um bucket do Amazon S3 é provavelmente mais rápido do que analisar a mesma imagem passada como bytes de imagem. Isso é bem válido se o tamanho da imagem for maior.

Se o número de chamadas para `DetectCustomLabels` exceder o máximo de TPS compatíveis com a soma das unidades de inferência que um modelo usa, o Amazon Rekognition Custom Labels retornará uma exceção `ProvisionedThroughputExceededException`.

Como gerenciar o throughput com unidades de inferência

É possível aumentar ou diminuir o throughput do seu modelo, dependendo das demandas da sua aplicação. Para aumentar o throughput, use unidades de inferência adicionais. Cada unidade de inferência adicional aumenta sua velocidade de processamento em uma unidade de inferência. Para obter informações sobre como calcular o número de unidades de inferência necessárias, consulte Calcular unidades de inferência para os modelos Amazon Rekognition Custom Labels e Amazon Lookout for Vision. Se quiser alterar o throughput compatível do modelo, você tem duas opções:

Adicionar ou remover unidades de inferência manualmente

Pare o modelo e reinicie com o número necessário de unidades de inferência. A desvantagem dessa abordagem é que o modelo não pode receber solicitações durante a reinicialização e não pode ser usado para lidar com picos de demanda. Use esta abordagem se seu modelo tiver um throughput

estável e seu caso de uso puder tolerar de 10 a 20 minutos de tempo de inatividade. Um exemplo seria se você quiser fazer chamadas em lote para seu modelo usando uma programação semanal.

Unidades de inferência de ajuste de escala automático

Se seu modelo precisar acomodar picos de demanda, o Amazon Rekognition Custom Labels pode escalar automaticamente o número de unidades de inferência que seu modelo usa. À medida que a demanda aumenta, o Amazon Rekognition Custom Labels adiciona unidades de inferência adicionais ao modelo e as remove quando a demanda diminui.

Para permitir que o Amazon Rekognition Custom Labels escalem automaticamente as unidades de inferência de um modelo, inicie o modelo e defina o número máximo de unidades de inferência que ele pode usar usando o parâmetro `MaxInferenceUnits`. Definir um número máximo de unidades de inferência permite gerenciar o custo de execução do modelo limitando o número de unidades de inferência disponíveis para ele. Se não especificar um número máximo de unidades, o Amazon Rekognition Custom Labels não escalará automaticamente seu modelo, usando apenas o número de unidades de inferência com as quais você começou. Para obter informações sobre o número máximo de unidades de inferência, consulte Service Quotas.

Também é possível especificar um número mínimo de unidades de inferência usando o parâmetro `MinInferenceUnits`. Isto permite que você especifique o throughput mínimo para seu modelo, em que uma única unidade de inferência representa uma hora de tempo de processamento.

> **Note**
>
> É possível pode definir o número máximo de unidades de inferência com o console do Amazon Rekognition Custom Labels. Em vez disso, especifique o parâmetro de entrada `MaxInferenceUnits` para a operação `StartProjectVersion`.

O Amazon Rekognition Custom Labels fornece CloudWatch as seguintes métricas do Amazon Logs que você pode usar para determinar o status atual de escalabilidade automática de um modelo.

Métrica	Descrição
`DesiredInferenceUnits`	O número de unidades de inferência que o Amazon Rekognition Custom Labels está aumentando ou diminuindo.

Métrica	Descrição
`InServiceInferenceUnits`	O número de unidades de inferência que o modelo está usando.

Se `DesiredInferenceUnits` =`InServiceInferenceUnits`, o Amazon Rekognition Custom Labels não está escalando o número de unidades de inferência neste momento.

Se `DesiredInferenceUnits` < `InServiceInferenceUnits`, o Amazon Rekognition Custom Labels está aumentando a escala verticalmente para o valor de `DesiredInferenceUnits`.

Se `DesiredInferenceUnits` < `InServiceInferenceUnits`, o Amazon Rekognition Custom Labels está reduzindo a escala verticalmente para o valor de `DesiredInferenceUnits`.

Para obter mais informações sobre as métricas retornadas pelos rótulos personalizados do Amazon Rekognition e pelas dimensões de filtragem, consulte métricas do Rekognition. CloudWatch

Para descobrir o número máximo de unidades de inferência que você solicitou para um modelo, chame `DescribeProjectsVersion` e verifique o campo `MaxInferenceUnits` na resposta. Para obter um código de exemplo, consulte Como descrever um modelo (SDK).

Zonas de disponibilidade

O Amazon Rekognition Custom Labels distribui unidades de inferência em várias zonas de disponibilidade em uma região da AWS para oferecer maior disponibilidade. Para ter mais informações, consulte zonas de disponibilidade. Para ajudar a proteger seus modelos de produção contra interrupções na zona de disponibilidade e falhas na unidade de inferência, inicie seus modelos de produção com pelo menos duas unidades de inferência.

Se ocorrer uma interrupção na zona de disponibilidade, todas as unidades de inferência na zona de disponibilidade ficarão indisponíveis e a capacidade do modelo será reduzida. As chamadas para DetectCustomLabelssão redistribuídas nas unidades de inferência restantes. Estas chamadas têm sucesso se não excederem as transações por segundo (TPS) compatíveis com as unidades de inferência restantes. Depois que a AWS reparar a zona de disponibilidade, as unidades de inferência são reiniciadas e a capacidade total é restaurada.

Se uma única unidade de inferência falhar, o Amazon Rekognition Custom Labels iniciará automaticamente uma nova unidade de inferência na mesma zona de disponibilidade. A capacidade do modelo é reduzida até que a nova unidade de inferência seja iniciada.

Como iniciar um modelo do Amazon Rekognition Custom Labels

Você pode começar a executar um modelo de etiquetas personalizadas do Amazon Rekognition usando o console ou usando a operação. StartProjectVersion

> **Important**
>
> Há uma cobrança pelo número de horas em que seu modelo está em execução e pelo número de unidades de inferência que seu modelo usa enquanto está em execução. Para obter mais informações, consulte Como executar um modelo do Amazon Rekognition Custom Labels.

A inicialização de um modelo pode levar alguns minutos para ser concluída. Para verificar o status atual da prontidão do modelo, verifique a página de detalhes do projeto ou uso DescribeProjectVersions.

Depois que o modelo é iniciado, você usa DetectCustomLabels, para analisar imagens usando o modelo. Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado. O console também fornece um código de exemplo para chamar `DetectCustomLabels`.

Tópicos

- Como iniciar um modelo do Amazon Rekognition Custom Labels (console)
- Como iniciar um modelo do Amazon Rekognition Custom Labels (SDK)

Como iniciar um modelo do Amazon Rekognition Custom Labels (console)

Use o procedimento a seguir para começar a executar um modelo do Amazon Rekognition Custom Labels com o console. Você pode iniciar o modelo diretamente do console ou usar o código AWS SDK fornecido pelo console.

Para iniciar um modelo (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.

5. Na página de recursos Projetos, selecione o projeto que contém o modelo treinado que você deseja iniciar.
6. Na seção Modelos, escolha o modelo que deseja iniciar.
7. Escolha a guia Usar modelo.
8. Execute um destes procedimentos:

   Start model using the console

   Na seção Iniciar ou interromper o modelo, faça o seguinte:

   1. Selecione o número de unidades de inferência que deseja usar. Para obter mais informações, consulte Como executar um modelo do Amazon Rekognition Custom Labels.
   2. Escolha Iniciar.
   3. Na caixa de diálogo Iniciar modelo, escolha Iniciar.

   Start model using the AWS SDK

   Na seção Use seu modelo, faça o seguinte:

   1. Escolha Código da API.
   2. Escolha AWS CLI ou Python.
   3. Em Iniciar modelo, copie o código de exemplo.
   4. Use o código de exemplo para iniciar seu modelo. Para obter mais informações, consulte Como iniciar um modelo do Amazon Rekognition Custom Labels (SDK).
9. Para voltar à página de visão geral do projeto, escolha o nome do seu projeto na parte superior da página.
10. Na seção Modelo, verifique o status do modelo. Quando o status do modelo é EXECUTANDO, é possível usar o modelo para analisar imagens. Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.

Como iniciar um modelo do Amazon Rekognition Custom Labels (SDK)

Você inicia um modelo chamando a StartProjectVersionAPI e passando o Amazon Resource Name (ARN) do modelo no parâmetro de `ProjectVersionArn` entrada. Também é possível especificar o número de unidades de inferência que deseja usar. Para obter mais informações, consulte Como executar um modelo do Amazon Rekognition Custom Labels.

Um modelo pode demorar um pouco para ser iniciado. Os exemplos em Python e Java neste tópico usam esperadores para aguardar o início do modelo. Um agentes de espera é um métodos utilitário que sonda um determinado estado para verificar se ele ocorreu em um cliente. Como alternativa, você pode verificar o status atual ligando DescribeProjectVersions.

Para iniciar um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir para iniciar um modelo.

   CLI

   Altere o valor de `project-version-arn` para o ARN do modelo que você deseja iniciar. Altere o valor de `--min-inference-units` para o número de unidades de inferência que você deseja usar. Você tem a opção de alterar `--max-inference-units` para o número máximo de unidades de inferência que o Amazon Rekognition Custom Labels pode usar para escalar automaticamente o modelo.

   ```
   aws rekognition start-project-version  --project-version-arn model_arn \
      --min-inference-units minimum number of units \
      --max-inference-units maximum number of units \
      --profile custom-labels-access
   ```

   Python

   Forneça os seguintes parâmetros de linha de comando:

   - `project_arn`: o ARN do projeto que contém o modelo que você deseja iniciar.
   - `model_arn`: o ARN do modelo que você deseja iniciar.
   - `min_inference_units`: o número de unidades de inferência que você deseja usar.
   - (Opcional) `--max_inference_units` O número máximo de unidades de inferência que o Amazon Rekognition Custom Labels pode usar para escalar automaticamente o modelo.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0
   ```

```
"""
Purpose
Shows how to start running an Amazon Lookout for Vision model.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    :return: The model status
    """

    logger.info("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]

    models = rek_client.describe_project_versions(ProjectArn=project_arn,
                                                  VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:

        logger.info("Status: %s", model['StatusMessage'])
        return model["Status"]

    error_message = f"Model {model_arn} not found."
    logger.exception(error_message)
    raise Exception(error_message)


def start_model(rek_client, project_arn, model_arn, min_inference_units,
 max_inference_units=None):
    """
    Starts the hosting of an Amazon Rekognition Custom Labels model.
```

```
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that contains the
    model that you want to start hosting.
    :param min_inference_units: The number of inference units to use for
 hosting.
    :param max_inference_units: The number of inference units to use for auto-
scaling
    the model. If not supplied, auto-scaling does not happen.
    """

    try:
        # Start the model
        logger.info(f"Starting model: {model_arn}. Please wait....")

        if max_inference_units is None:
            rek_client.start_project_version(ProjectVersionArn=model_arn,

 MinInferenceUnits=int(min_inference_units))
        else:
            rek_client.start_project_version(ProjectVersionArn=model_arn,
                                             MinInferenceUnits=int(
                                                 min_inference_units),

 MaxInferenceUnits=int(max_inference_units))

        # Wait for the model to be in the running state
        version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]
        project_version_running_waiter = rek_client.get_waiter(
            'project_version_running')
        project_version_running_waiter.wait(
            ProjectArn=project_arn, VersionNames=[version_name])

        # Get the running status
        return get_model_status(rek_client, project_arn, model_arn)

    except ClientError as err:
        logger.exception("Client error: Problem starting model: %s", err)
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
```

```
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains that the model
 you want to start."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to start."
    )
    parser.add_argument(
        "min_inference_units", help="The minimum number of inference units to
 use."
    )
    parser.add_argument(
        "--max_inference_units",  help="The maximum number of inference units to
 use for auto-scaling the model.", required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Start the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status = start_model(rekognition_client,
                             args.project_arn, args.model_arn,
                             args.min_inference_units,
                             args.max_inference_units)

        print(f"Finished starting model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
```

```
        error_message = f"Client error: Problem starting model: {err}"
        logger.exception(error_message)
        print(error_message)

    except Exception as err:
        error_message = f"Problem starting model:{err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Forneça os seguintes parâmetros de linha de comando:

- `project_arn`: o ARN do projeto que contém o modelo que você deseja iniciar.
- `model_arn`: o ARN do modelo que você deseja iniciar.
- `min_inference_units`: o número de unidades de inferência que você deseja usar.
- (Opcional) `max_inference_units`: o número máximo de unidades de inferência que o Amazon Rekognition Custom Labels pode usar para escalar automaticamente o modelo. Se não especificar um valor, o escalonamento automático não acontecerá.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.StartProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartProjectVersionResponse;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class StartModel {

    public static final Logger logger =
 Logger.getLogger(StartModel.class.getName());



    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void startMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn,
            Integer minInferenceUnits, Integer maxInferenceUnits
            ) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Starting model: {0}", modelArn);

            StartProjectVersionRequest startProjectVersionRequest = null;

            if (maxInferenceUnits == null) {
                startProjectVersionRequest =
 StartProjectVersionRequest.builder()
                    .projectVersionArn(modelArn)
```

```
                    .minInferenceUnits(minInferenceUnits)
                    .build();
            }
            else {
                startProjectVersionRequest =
StartProjectVersionRequest.builder()
                        .projectVersionArn(modelArn)
                        .minInferenceUnits(minInferenceUnits)
                        .maxInferenceUnits(maxInferenceUnits)
                        .build();

            }

            StartProjectVersionResponse response =
rekClient.startProjectVersion(startProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString() );


            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);


            // wait until model starts

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .versionNames(versionName)
                    .projectArn(projectArn)
                    .build();

            RekognitionWaiter waiter = rekClient.waiter();

            WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionRunning(describeProjectVersionsRequest);

            Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();
```

```
            DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {
                if(projectVersionDescription.status() ==
ProjectVersionStatus.RUNNING) {
                    logger.log(Level.INFO, "Model is running" );

                }
                else {
                    String error = "Model training failed: " +
projectVersionDescription.statusAsString() + " "
                            + projectVersionDescription.statusMessage() + " " +
modelArn;
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);
                }

            }


        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not start model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String modelArn = null;
        String projectArn = null;
        Integer minInferenceUnits = null;
        Integer maxInferenceUnits = null;




        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
<min_inference_units> <max_inference_units>\n\n" + "Where:\n"
```

```
                + "   project_arn - The ARN of the project that contains the
 model that you want to start. \n\n"
                + "   model_arn - The ARN of the model version that you want to
 start.\n\n"
                + "   min_inference_units - The number of inference units to
 start the model with.\n\n"
                + "   max_inference_units - The maximum number of inference
 units that Custom Labels can use to "
                + "   automatically scale the model. If the value is null,
 automatic scaling doesn't happen.\n\n";

        if (args.length < 3  || args.length >4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        modelArn = args[1];
        minInferenceUnits=Integer.parseInt(args[2]);

        if (args.length == 4) {
            maxInferenceUnits = Integer.parseInt(args[3]);
        }

        try {


            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Start the model.
            startMyModel(rekClient, projectArn, modelArn, minInferenceUnits,
 maxInferenceUnits);


            System.out.println(String.format("Model started: %s", modelArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
```

```
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }



    }

}
```

Como interromper um modelo do Amazon Rekognition Custom Labels

Você pode parar de executar um modelo de etiquetas personalizadas do Amazon Rekognition usando o console ou usando a operação. StopProjectVersion

Tópicos

- Como interromper um modelo do Amazon Rekognition Custom Labels (console)
- Como interromper um modelo do Amazon Rekognition Custom Labels (SDK)

Como interromper um modelo do Amazon Rekognition Custom Labels (console)

Use o procedimento a seguir para interromper um modelo em execução do Amazon Rekognition Custom Labels com o console. Você pode parar o modelo diretamente do console ou usar o código AWS SDK fornecido pelo console.

Para interromper um modelo (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.

5. Na página de recursos Projetos, selecione o projeto que contém o modelo treinado que você deseja interromper.
6. Na seção Modelos, escolha o modelo que deseja interromper.
7. Escolha a guia Usar modelo.
8. Stop model using the console

    1. Na seção Iniciar ou interromper modelo, escolha Interromper.
    2. Na caixa de diálogo Interromper modelo, insira interromper para confirmar que deseja interromper o modelo.
    3. Escolha Parar para interromper seu modelo.

    Stop model using the AWS SDK

    Na seção Use seu modelo, faça o seguinte:

    1. Escolha Código da API.
    2. Escolha AWS CLI ou Python.
    3. Em Interromper modelo, copie o código de exemplo.
    4. Use o código de exemplo para interromper seu modelo. Para obter mais informações, consulte Como interromper um modelo do Amazon Rekognition Custom Labels (SDK).
9. Escolha o nome do seu projeto na parte superior da página para voltar à página de visão geral do projeto.
10. Na seção Modelo, verifique o status do modelo. O modelo foi interrompido quando o status do modelo é INTERROMPIDO.

Como interromper um modelo do Amazon Rekognition Custom Labels (SDK)

Você interrompe um modelo chamando a StopProjectVersionAPI e passando o Amazon Resource Name (ARN) do modelo no parâmetro de `ProjectVersionArn` entrada.

Um modelo pode demorar um pouco para parar. Para verificar o status atual, chame `DescribeProjectVersions`.

Para interromper um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir para interromper um modelo em execução.

   CLI

   Altere o valor de `project-version-arn` para o ARN da versão do modelo que você deseja interromper.

   ```
   aws rekognition stop-project-version --project-version-arn "model arn" \
     --profile custom-labels-access
   ```

   Python

   O exemplo a seguir interrompe um modelo que já está em execução.

   Forneça os seguintes parâmetros de linha de comando:

   - `project_arn`: o ARN do projeto que contém o modelo que você deseja interromper.
   - `model_arn`: o ARN do modelo que você deseja interromper.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Shows how to stop a running Amazon Lookout for Vision model.
   """

   import argparse
   import logging
   import time
   import boto3

   from botocore.exceptions import ClientError


   logger = logging.getLogger(__name__)
   ```

```
def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    """

    logger.info ("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name=(model_arn.split("version/",1)[1]).rpartition('/')[0]

    # Get the model status.
    models=rek_client.describe_project_versions(ProjectArn=project_arn,
    VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:
        logger.info("Status: %s",model['StatusMessage'])
        return model["Status"]

    # No model found.
    logger.exception("Model %s not found.", model_arn)
    raise Exception("Model %s not found.", model_arn)


def stop_model(rek_client, project_arn, model_arn):
    """
    Stops a running Amazon Rekognition Custom Labels Model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to stop running.
    :param model_arn:  The ARN of the model (ProjectVersion) that you want to
 stop running.
    """

    logger.info("Stopping model: %s", model_arn)

    try:
        # Stop the model.
        response=rek_client.stop_project_version(ProjectVersionArn=model_arn)

        logger.info("Status: %s", response['Status'])
```

```
        # stops when hosting has stopped or failure.
        status = ""
        finished = False

        while finished is False:

            status=get_model_status(rek_client, project_arn, model_arn)

            if status == "STOPPING":
                logger.info("Model stopping in progress...")
                time.sleep(10)
                continue
            if status == "STOPPED":
                logger.info("Model is not running.")
                finished = True
                continue

            error_message = f"Error stopping model. Unexepected state: {status}"
            logger.exception(error_message)
            raise Exception(error_message)

        logger.info("finished. Status %s", status)
        return status

    except ClientError as err:
        logger.exception("Couldn't stop model - %s: %s",
           model_arn,err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to stop."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to stop."
    )
```

```
def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Stop the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status=stop_model(rekognition_client, args.project_arn, args.model_arn)

        print(f"Finished stopping model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        logger.exception("Problem stopping model:%s",err)
        print(f"Failed to stop model: {err}")

    except Exception as err:
        logger.exception("Problem stopping model:%s", err)
        print(f"Failed to stop model: {err}")

if __name__ == "__main__":
    main()
```

Java V2

Forneça os seguintes parâmetros de linha de comando:

- `project_arn`: o ARN do projeto que contém o modelo que você deseja interromper.
- `model_arn`: o ARN do modelo que você deseja interromper.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
```

```
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.StopProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StopProjectVersionResponse;


import java.util.logging.Level;
import java.util.logging.Logger;

public class StopModel {

    public static final Logger logger =
 Logger.getLogger(StopModel.class.getName());

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void stopMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn)
            throws Exception, RekognitionException {
```

```
        try {

            logger.log(Level.INFO, "Stopping {0}", modelArn);

            StopProjectVersionRequest stopProjectVersionRequest =
StopProjectVersionRequest.builder()
                    .projectVersionArn(modelArn).build();

            StopProjectVersionResponse response =
rekClient.stopProjectVersion(stopProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString());

            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);

            // wait until model stops

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .projectArn(projectArn).versionNames(versionName).build();

            boolean stopped = false;

            // Wait until create finishes

            do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                        .projectVersionDescriptions()) {

                    ProjectVersionStatus status =
projectVersionDescription.status();
```

```
                    logger.log(Level.INFO, "stopping model: {0} ", modelArn);

                    switch (status) {

                    case STOPPED:
                        logger.log(Level.INFO, "Model stopped");
                        stopped = true;
                        break;

                    case STOPPING:
                        Thread.sleep(5000);
                        break;

                    case FAILED:
                        String error = "Model stopping failed: " +
projectVersionDescription.statusAsString() + " "
                                + projectVersionDescription.statusMessage() + "
" + modelArn;
                        logger.log(Level.SEVERE, error);
                        throw new Exception(error);

                    default:
                        String unexpectedError = "Unexpected stopping state: "
                                + projectVersionDescription.statusAsString() + "
"
                                + projectVersionDescription.statusMessage() + "
" + modelArn;
                        logger.log(Level.SEVERE, unexpectedError);
                        throw new Exception(unexpectedError);
                    }
                }

            } while (stopped == false);

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not stop model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {
```

```
        String modelArn = null;
        String projectArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>\n
\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 model that you want to stop. \n\n"
                + "   model_arn - The ARN of the model version that you want to
 stop.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        modelArn = args[1];


        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Stop model
            stopMyModel(rekClient, projectArn, modelArn);

            System.out.println(String.format("Model stopped: %s", modelArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }
```

```
    }

}
```

Duração do relatório de execução e unidades de inferência usadas

Se você treinou e iniciou seu modelo depois de agosto de 2022, pode usar a CloudWatch métrica da `InServiceInferenceUnits` Amazon para determinar por quantas horas um modelo foi executado e o número de unidades de inferência usadas durante essas horas.

> **Note**
>
> Se você tiver apenas um modelo em uma AWS região, também poderá obter o tempo de execução do modelo rastreando chamadas bem-sucedidas de entrada `StartprojectVersion` e `StopProjectVersion` entrada CloudWatch. Essa abordagem não funciona se você executar mais de um modelo na AWS região, pois as métricas não incluem informações sobre o modelo.
> Como alternativa, você pode usar AWS CloudTrail para rastrear chamadas para `StartProjectVersion` e `StopProjectVersion` (o que inclui o ARN do modelo no `requestParameters` campo do histórico de eventos). CloudTrail os eventos são limitados a 90 dias, mas você pode armazenar eventos por até 7 anos em um CloudTraillago.

O procedimento a seguir cria gráficos para o seguinte:

- O número de horas em que um modelo foi executado.
- O número de unidades de inferência que um modelo usou.

É possível escolher um período de até 15 meses no passado. Para obter mais informações sobre a retenção de métricas, consulte Retenção de métricas.

Para determinar a duração do modelo e as unidades de inferência usadas para um modelo

1. Faça login no AWS Management Console e abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação esquerdo, escolha Todas as métricas em Métricas.

3. No painel inferior, escolha a guia Origem.
4. Certifique-se de que o botão Painel está selecionado.
5. Na caixa de edição, substitua o JSON existente pelo JSON a seguir. Altere os seguintes valores:
   - `Project_Name`: o projeto que contém o modelo que você deseja representar graficamente.
   - `Version_Name`: a versão do modelo que você deseja representar graficamente.
   - `AWS_Region`— A AWS região que contém o modelo. Verifique se o CloudWatch console está na mesma AWS região, verificando o seletor de região na barra de navegação na parte superior da página. Atualize conforme for necessário.

```
{
    "sparkline": true,
    "metrics": [
        [
            {
                "expression": "SUM(m1)*m1",
                "label": "Inference units used",
                "id": "e1"
            }
        ],
        [
            {
                "expression": "DATAPOINT_COUNT(m1)*m1/m1",
                "label": "Hours running",
                "id": "e2"
            }
        ],
        [
            "AWS/Rekognition",
            "InServiceInferenceUnits",
            "ProjectName",
            "Project_Name",
            "VersionName",
            "Version_Name",
            {
                "id": "m1",
                "visible": false
            }
        ]
    ],
```

```
    "view": "singleValue",
    "stacked": false,
    "region": "AWS_Region",
    "stat": "Average",
    "period": 3600,
    "title": "Hours run and inference units used"
}
```

6. Selecione Atualizar.
7. Na parte superior da página, escolha uma linha do tempo. É preciso ver os números das unidades de inferência usadas e as horas de execução durante a linha do tempo. As lacunas no gráfico indicam momentos em que o modelo não estava em execução. A captura de tela do console abaixo mostra as unidades de inferência usadas e as horas de execução ao longo de períodos, com um tempo personalizado de 2 semanas definido e os valores mais altos de 214 unidades de inferência e 209 horas de execução.

8. (Opcional) Adicione o gráfico a um painel, selecione Ações e Adicionar ao painel: aprimorado.

Como analisar uma imagem com um modelo treinado

Para analisar uma imagem com um modelo treinado de etiquetas personalizadas do Amazon Rekognition, você chama a API. DetectCustomLabels O resultado `DetectCustomLabels` é uma previsão de que a imagem contém objetos, cenas ou conceitos específicos.

Para chamar `DetectCustomLabels`, você deve especificar o seguinte:

- O nome do recurso da Amazon (ARN) do modelo do Amazon Rekognition Custom Labels que deseja usar.
- A imagem com a qual você deseja que o modelo faça uma previsão. É possível fornecer uma imagem de entrada como uma matriz de bytes de imagem (bytes de imagem codificados em base64) ou como um objeto do Amazon S3. Para obter mais informações, consulte Imagem.

Os rótulos personalizados são retornados em uma matriz de objetos Custom Label. Cada rótulo personalizado representa um único objeto, cena ou conceito encontrado na imagem. Um rótulo personalizado inclui:

- Um rótulo para o objeto, cena ou conceito encontrado na imagem.
- Uma caixa delimitadora para objetos encontrados na imagem. As coordenadas da caixa delimitadora e mostram onde o texto está localizado na imagem de origem. Os valores das coordenadas são uma proporção do tamanho geral da imagem. Para obter mais informações, consulte BoundingBox. `DetectCustomLabels`retorna caixas delimitadoras somente se o modelo for treinado para detectar a localização dos objetos.
- A confiança que o Amazon Rekognition Custom Labels tem na precisão do rótulo e da caixa delimitadora.

Para filtrar os rótulos com base na confiança da detecção, especifique um valor para `MinConfidence` que corresponda ao nível de confiança desejado. Por exemplo, se você precisar ter muita confiança na previsão, especifique um valor alto para `MinConfidence`. Para obter todos os rótulos, independentemente da confiança, especifique um valor de `MinConfidence` de 0.

O desempenho do seu modelo é medido, em parte, pelas métricas de recall e precisão calculadas durante o treinamento do modelo. Para obter mais informações, consulte Métricas para avaliar seu modelo.

Para aumentar a precisão do seu modelo, defina um valor maior para `MinConfidence`. Para obter mais informações, consulte Como reduzir falsos positivos (melhor precisão).

Para aumentar o recall do seu modelo, use um valor menor para `MinConfidence`. Para obter mais informações, consulte Como reduzir falsos negativos (melhor recall).

Se você não especificar um valor para `MinConfidence`, o Amazon Rekognition Custom Labels retornará um rótulo com base no limite assumido para esse rótulo. Para obter mais informações, consulte Limite assumido. É possível obter o valor do limite assumido para um rótulo a partir dos resultados de treinamento do modelo. Para obter mais informações, consulte Como treinar um modelo (console).

Ao usar o parâmetro `MinConfidence` de entrada, você está especificando um limite desejado para a chamada. Os rótulos detectados com uma confiança abaixo do valor de `MinConfidence` não são retornados na resposta. Além disso, o limite assumido para um rótulo não afeta a inclusão do rótulo na resposta.

> **Note**
>
> As métricas do Amazon Rekognition Custom Labels expressam um limite assumido como um valor de ponto flutuante entre 0-1. O intervalo de `MinConfidence` normaliza o limite para um valor percentual (0-100). As respostas de confiança também DetectCustomLabels são retornadas como uma porcentagem.

Talvez queira especificar um limite para rótulos específicos. Por exemplo, quando a métrica de precisão é aceitável para o Rótulo A, mas não para o Rótulo B. Ao especificar um limite diferente (`MinConfidence`), considere o seguinte.

- Se estiver interessado apenas em um único rótulo (A), defina o valor de `MinConfidence` para o valor limite desejado. Na resposta, as previsões para o rótulo A são retornadas (junto com outros rótulos) somente se a confiança for maior que `MinConfidence`. É preciso filtrar todos os outros rótulos retornados.
- Se quiser aplicar limites diferentes a vários rótulos, faça o seguinte:
  1. Use um valor de 0 para `MinConfidence`. Um valor 0 garante que todos os rótulos sejam retornados, independentemente da confiança na detecção.
  2. Para cada rótulo retornado, aplique o limite desejado verificando se a confiança do rótulo é maior do que o limite que você deseja para o rótulo.

Para obter mais informações, consulte Como melhorar um modelo treinado do Amazon Rekognition Custom Labels.

Se achar que os valores de confiança retornados por `DetectCustomLabels` são muito baixos, considere retreinar o modelo. Para obter mais informações, consulte Como treinar um modelo do Amazon Rekognition Custom Labels. É possível restringir o número de rótulos personalizados retornados de `DetectCustomLabels` especificando o parâmetro de entrada `MaxResults`. Os resultados são retornados classificados da maior confiança para a mais baixa.

Para outros exemplos que chamam `DetectCustomLabels`, consulte Exemplos de rótulos personalizados.

Para obter informações sobre a segurança do `DetectCustomLabels`, consulte Protegendo DetectCustomLabels.

Para detectar rótulos personalizados (API)

1. Se ainda não tiver feito isso:

   a. Certifique-se de que você tem as permissões `DetectCustomLabels` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Configurar permissões do SDK.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.

2. Treine e implante seu modelo. Para obter mais informações, consulte Como criar um modelo do Amazon Rekognition Custom Labels.

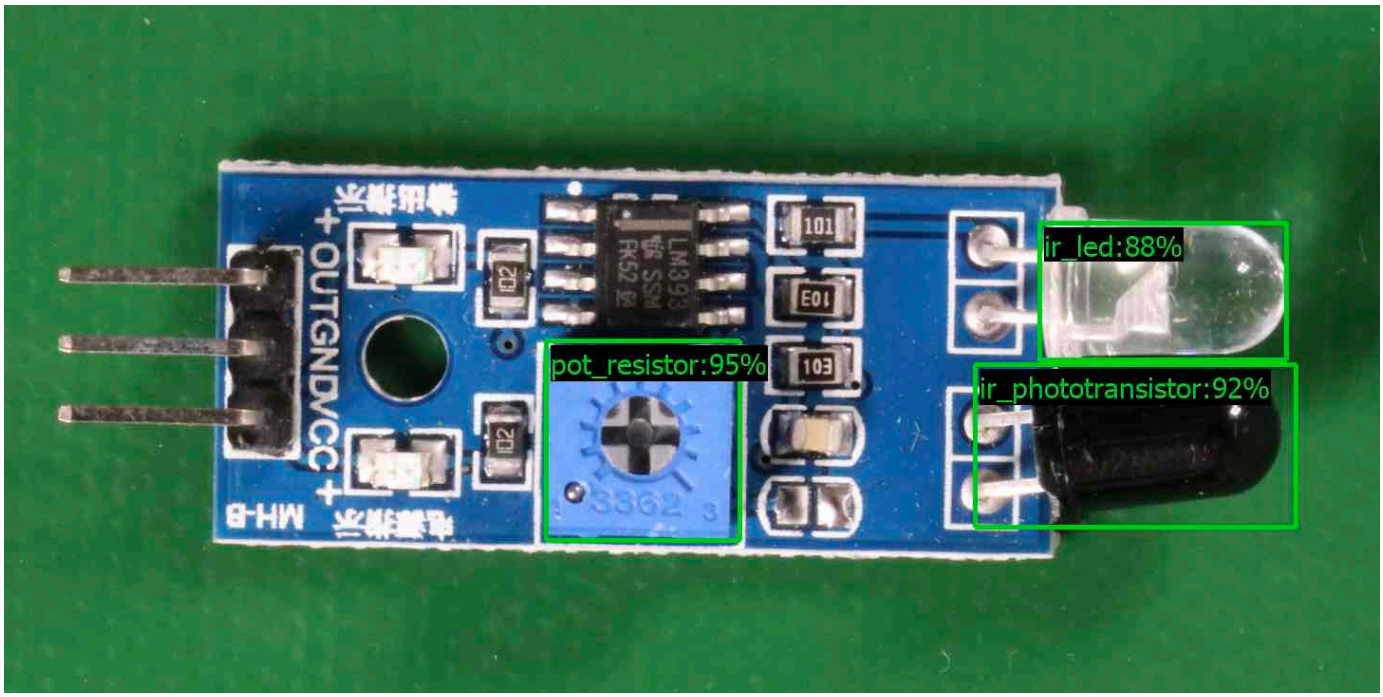
3. Certifique-se de que o usuário que está chamando `DetectCustomLabels` tenha acesso ao modelo usado na etapa 2. Para obter mais informações, consulte Protegendo DetectCustomLabels.

4. Faça upload de uma imagem que deseja analisar para um bucket do S3.

   Para obter mais informações, consulte Fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service. Os exemplos em Python, Java e Java 2 também mostram como usar um arquivo de imagem local para transmitir uma imagem usando bytes brutos. O arquivo deve ter menos de 4 MB.

5. Use os exemplos a seguir para chamar a operação `DetectCustomLabels`. Os exemplos em Python e Java mostram a imagem e sobrepõem os resultados da análise, semelhante à imagem

a seguir. As imagens a seguir contêm caixas delimitadoras e rótulos para uma placa de circuito com potenciômetro, fototransistor infravermelho e componentes de LED.



AWS CLI

Esse AWS CLI comando exibe a saída JSON para a operação da `DetectCustomLabels` CLI. Altere os valores dos parâmetros de entrada a seguir.

- `bucket` com o nome do bucket do Amazon S3 que você usou na etapa 4.
- `image` com o nome do arquivo de imagem de entrada que você carregou na etapa 4.
- `projectVersionArn` com o ARN do modelo que você deseja usar.

```
aws rekognition detect-custom-labels --project-version-arn model_arn \
   --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}' \
   --min-confidence 70 \
   --profile custom-labels-access
```

Python

O código de exemplo a seguir exibe caixas delimitadoras e rótulos de nível de imagem encontrados em uma imagem.

Para analisar uma imagem local, execute o programa e forneça os seguintes argumentos de linha de comando:

- O ARN do modelo com o qual deseja analisar a imagem.
- O nome e a localização de um arquivo de imagem local.

Para analisar uma imagem armazenada em um bucket do Amazon S3, execute o programa e forneça os seguintes argumentos de linha de comando:

- O ARN do modelo com o qual deseja analisar a imagem.
- O nome e a localização de uma imagem no bucket do Amazon S3 usado na etapa 4.
- `--bucket`*`bucket name`*— O bucket do Amazon S3 que você usou na etapa 4.

Observe que este exemplo pressupõe que você usa o Pillow versão 8.0.0 ou posterior.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Amazon Rekognition Custom Labels detection example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/detecting-custom-
labels.html
Shows how to detect custom labels by using an Amazon Rekognition Custom Labels
 model.
The image can be stored on your local computer or in an Amazon S3 bucket.
"""

import io
import logging
import argparse
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def analyze_local_image(rek_client, model, photo, min_confidence):
    """
    Analyzes an image stored as a local file.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
 want to use.
    :param photo: The name and file path of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        logger.info("Analyzing local file: %s", photo)
        image = Image.open(photo)
        image_type = Image.MIME[image.format]

        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file:
 {photo}"
            )

        # get images bytes for call to detect_anomalies
        image_bytes = io.BytesIO()
        image.save(image_bytes, format=image.format)
        image_bytes = image_bytes.getvalue()

        response = rek_client.detect_custom_labels(Image={'Bytes': image_bytes},
                                                   MinConfidence=min_confidence,
                                                   ProjectVersionArn=model)

        show_image(image, response)
        return len(response['CustomLabels'])

    except ClientError as client_err:
        logger.error(format(client_err))
        raise
    except FileNotFoundError as file_error:
        logger.error(format(file_error))
        raise
```

```
def analyze_s3_image(rek_client, s3_connection, model, bucket, photo,
 min_confidence):
    """
    Analyzes an image stored in the specified S3 bucket.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
 want to use.
    :param bucket: The name of the S3 bucket that contains the image that you
 want to analyze.
    :param photo: The name of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        # Get image from S3 bucket.

        logger.info("analyzing bucket: %s image: %s", bucket, photo)
        s3_object = s3_connection.Object(bucket, photo)
        s3_response = s3_object.get()

        stream = io.BytesIO(s3_response['Body'].read())
        image = Image.open(stream)

        image_type = Image.MIME[image.format]

        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file:
 {photo}")

        ImageDraw.Draw(image)

        # Call DetectCustomLabels.
        response = rek_client.detect_custom_labels(
            Image={'S3Object': {'Bucket': bucket, 'Name': photo}},
            MinConfidence=min_confidence,
            ProjectVersionArn=model)

        show_image(image, response)
        return len(response['CustomLabels'])

    except ClientError as err:
```

```
        logger.error(format(err))
        raise


def show_image(image, response):
    """
    Displays the analyzed image and overlays analysis results
    :param image: The analyzed image
    :param response: the response from DetectCustomLabels
    """
    try:
        font_size = 40
        line_width = 5

        img_width, img_height = image.size
        draw = ImageDraw.Draw(image)

        # Calculate and display bounding boxes for each detected custom label.
        image_level_label_height = 0

        for custom_label in response['CustomLabels']:
            confidence = int(round(custom_label['Confidence'], 0))
            label_text = f"{custom_label['Name']}:{confidence}%"
            fnt = ImageFont.truetype('Tahoma.ttf', font_size)
            text_left, text_top, text_right, text_bottom = draw.textbbox((0, 0),
 label_text, fnt)
            text_width, text_height = text_right - text_left, text_bottom -
 text_top

            logger.info("Label: %s", custom_label['Name'])
            logger.info("Confidence:  %s", confidence)

            # Draw bounding boxes, if present
            if 'Geometry' in custom_label:
                box = custom_label['Geometry']['BoundingBox']
                left = img_width * box['Left']
                top = img_height * box['Top']
                width = img_width * box['Width']
                height = img_height * box['Height']

                logger.info("Bounding box")
                logger.info("\tLeft: {0:.0f}".format(left))
                logger.info("\tTop: {0:.0f}".format(top))
                logger.info("\tLabel Width: {0:.0f}".format(width))
```

```
                logger.info("\tLabel Height: {0:.0f}".format(height))

                points = (
                    (left, top),
                    (left + width, top),
                    (left + width, top + height),
                    (left, top + height),
                    (left, top))
                # Draw bounding box and label text
                draw.line(points, fill="limegreen", width=line_width)
                draw.rectangle([(left + line_width, top+line_width),
                                (left + text_width + line_width, top +
 line_width + text_height)], fill="black")
                draw.text((left + line_width, top + line_width),
                          label_text, fill="limegreen", font=fnt)

            # draw image-level label text.
            else:
                draw.rectangle([(10, image_level_label_height),
                                (text_width + 10, image_level_label_height
+text_height)], fill="black")
                draw.text((10, image_level_label_height),
                          label_text, fill="limegreen", font=fnt)

                image_level_label_height += text_height

        image.show()

    except Exception as err:
        logger.error(format(err))
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
```

```
        "image", help="The path and file name of the image that you want to
 analyze"
    )
    parser.add_argument(
        "--bucket",  help="The bucket that contains the image. If not supplied,
 image is assumed to be a local file.", required=False
    )


def main():

    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        label_count = 0
        min_confidence = 50

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        if args.bucket is None:
            # Analyze local image.
            label_count = analyze_local_image(rekognition_client,
                                              args.model_arn,
                                              args.image,
                                              min_confidence)
        else:
            # Analyze image in S3 bucket.
            s3_connection = session.resource('s3')
            label_count = analyze_s3_image(rekognition_client,
                                           s3_connection,
                                           args.model_arn,
                                           args.bucket,
                                           args.image,
                                           min_confidence)

        print(f"Custom labels detected: {label_count}")
```

```
    except ClientError as client_err:
        print("A service client error occurred: " +
              format(client_err.response["Error"]["Message"]))

    except ValueError as value_err:
        print("A value error occurred: " + format(value_err))

    except FileNotFoundError as file_error:
        print("File not found error: " + format(file_error))

    except Exception as err:
        print("An error occurred: " + format(err))


if __name__ == "__main__":
    main()
```

Java

O código de exemplo a seguir exibe caixas delimitadoras e rótulos de nível de imagem encontrados em uma imagem.

Para analisar uma imagem local, execute o programa e forneça os seguintes argumentos de linha de comando:

- O ARN do modelo com o qual deseja analisar a imagem.
- O nome e a localização de um arquivo de imagem local.

Para analisar uma imagem armazenada em um bucket do Amazon S3, execute o programa e forneça os seguintes argumentos de linha de comando:

- O ARN do modelo com o qual deseja analisar a imagem.
- O nome e a localização de uma imagem no bucket do Amazon S3 usado na etapa 4.
- O bucket do Amazon S3 que contém a imagem que você usou na etapa 4.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
```

```
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import java.io.FileNotFoundException;
import java.awt.font.FontRenderContext;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CustomLabel;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.util.IOUtils;

// Calls DetectCustomLabels and displays a bounding box around each detected
 image.
public class DetectCustomLabels extends JPanel {

    private transient DetectCustomLabelsResult response;
```

```
   private transient Dimension dimension;
   private transient BufferedImage image;

   public static final Logger logger =
Logger.getLogger(DetectCustomLabels.class.getName());

   // Finds custom labels in an image stored in an S3 bucket.
   public DetectCustomLabels(AmazonRekognition rekClient,
           AmazonS3 s3client,
           String projectVersionArn,
           String bucket,
           String key,
           Float minConfidence) throws AmazonRekognitionException,
AmazonS3Exception, IOException {

       logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
Object[] { bucket, key });

       // Get image from S3 bucket and create BufferedImage
       com.amazonaws.services.s3.model.S3Object s3object =
s3client.getObject(bucket, key);
       S3ObjectInputStream inputStream = s3object.getObjectContent();
       image = ImageIO.read(inputStream);

       // Set image size
       setWindowDimensions();

       DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
               .withProjectVersionArn(projectVersionArn)
               .withImage(new Image().withS3Object(new
S3Object().withName(key).withBucket(bucket)))
               .withMinConfidence(minConfidence);

       // Call DetectCustomLabels

       response = rekClient.detectCustomLabels(request);
       logFoundLabels(response.getCustomLabels());
       drawLabels();

   }

   // Finds custom label in a local image file.
   public DetectCustomLabels(AmazonRekognition rekClient,
           String projectVersionArn,
```

```
            String photo,
            Float minConfidence)
            throws IOException, AmazonRekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);

        // Get image bytes and buffered image
        ByteBuffer imageBytes;
        try (InputStream inputStream = new FileInputStream(new File(photo))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
        }

        // Get image for display
        InputStream imageBytesStream;
        imageBytesStream = new ByteArrayInputStream(imageBytes.array());

        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        image = ImageIO.read(imageBytesStream);
        ImageIO.write(image, "jpg", baos);

        // Set image size
        setWindowDimensions();

        // Analyze image
        DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
                .withProjectVersionArn(projectVersionArn)
                .withImage(new Image()
                        .withBytes(imageBytes))
                .withMinConfidence(minConfidence);

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.getCustomLabels());

        drawLabels();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
        logger.info("Custom labels found");
        if (customLabels.isEmpty()) {
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");
```

```
        } else {
            for (CustomLabel customLabel : customLabels) {
                logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                        new Object[] { customLabel.getName(),
customLabel.getConfidence() });
            }

        }
    }

    // Sets window dimensions to 1/2 screen size, unless image is smaller
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);

    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    public void drawLabels() {
        // Draws bounding boxes (if present) and label text.

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
```

```
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.getCustomLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.getName();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
            if (customLabel.getGeometry() != null) {

                BoundingBox box = customLabel.getGeometry().getBoundingBox();
                float left = imageWidth * box.getLeft();
                float top = imageHeight * box.getTop();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                        textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

                // Draw bounding box around label location
                g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.getWidth())),
                        Math.round((imageHeight * box.getHeight())));
```

```
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
 textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

    public static void main(String args[]) throws Exception {

        String photo = null;
        String bucket = null;
        String projectVersionArn = null;
        float minConfidence = 50;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
                + "   bucket - The S3 bucket that contains the image. Don't
 specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
 assumed to be
        // in an S3 bucket.

        if (args.length < 2 || args.length > 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectVersionArn = args[0];
```

```
        photo = args[1];

        if (args.length == 3) {
            bucket = args[2];
        }

        DetectCustomLabels panel = null;

        try {

            AWSCredentialsProvider provider =new
ProfileCredentialsProvider("custom-labels-access");

            AmazonRekognition rekClient =
AmazonRekognitionClientBuilder.standard()
                    .withCredentials(provider)
                    .withRegion(Regions.US_WEST_2)
                    .build();


            AmazonS3 s3client = AmazonS3ClientBuilder.standard()
            .withCredentials(provider)
            .withRegion(Regions.US_WEST_2)
            .build();

            // Create frame and panel.
            JFrame frame = new JFrame("Custom Labels");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            if (args.length == 2) {
                // Analyze local image
                panel = new DetectCustomLabels(rekClient, projectVersionArn,
photo, minConfidence);
            } else {
                // Analyze image in S3 bucket
                panel = new DetectCustomLabels(rekClient, s3client,
projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (AmazonRekognitionException rekError) {
```

```
            String errorMessage = "Rekognition client error: " +
 rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
 fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (AmazonS3Exception s3Error) {
            String errorMessage = "S3 error: " + s3Error.getErrorMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
}
```

Java V2

O código de exemplo a seguir exibe caixas delimitadoras e rótulos de nível de imagem encontrados em uma imagem.

Para analisar uma imagem local, execute o programa e forneça os seguintes argumentos de linha de comando:

- `projectVersionArn`: o ARN do modelo com o qual você deseja analisar a imagem.
- `photo`: o nome e a localização de um arquivo de imagem local.

Para analisar uma imagem armazenada em um bucket do S3, execute o programa e forneça os seguintes argumentos de linha de comando:

- O ARN do modelo com o qual deseja analisar a imagem.

- O nome e a localização de uma imagem no bucket do S3 usado na etapa 4.
- O bucket do Amazon S3 que contém a imagem que você usou na etapa 4.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.CustomLabel;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.NoSuchBucketException;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import java.awt.*;
import java.awt.font.FontRenderContext;
import java.awt.image.BufferedImage;
```

```
import javax.imageio.ImageIO;
import javax.swing.*;

import java.util.logging.Level;
import java.util.logging.Logger;

// Calls DetectCustomLabels on an image. Displays bounding boxes or
// image level labels found in the image.
public class ShowCustomLabels extends JPanel {

    private transient BufferedImage image;
    private transient DetectCustomLabelsResponse response;
    private transient Dimension dimension;
    public static final Logger logger =
 Logger.getLogger(ShowCustomLabels.class.getName());

    // Finds custom labels in an image stored in an S3 bucket.
    public ShowCustomLabels(RekognitionClient rekClient,
            S3Client s3client,
            String projectVersionArn,
            String bucket,
            String key,
            Float minConfidence) throws RekognitionException,
 NoSuchBucketException, NoSuchKeyException, IOException {

        logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
 Object[] { bucket, key });
        // Get image from S3 bucket and create BufferedImage
        GetObjectRequest requestObject =
 GetObjectRequest.builder().bucket(bucket).key(key).build();
        ResponseBytes<GetObjectResponse> result =
 s3client.getObject(requestObject, ResponseTransformer.toBytes());
        ByteArrayInputStream bis = new
 ByteArrayInputStream(result.asByteArray());
        image = ImageIO.read(bis);

        // Set image size
        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        S3Object s3Object = S3Object.builder().bucket(bucket).name(key).build();

        Image s3Image = Image.builder().s3Object(s3Object).build();
```

```
        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(s3Image)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);
        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Finds custom label in a local image file.
    public ShowCustomLabels(RekognitionClient rekClient,
            String projectVersionArn,
            String photo,
            Float minConfidence)
            throws IOException, RekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);
        // Get image bytes and buffered image
        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageBytes = SdkBytes.fromInputStream(sourceStream);
        ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageBytes.asByteArray());
        image = ImageIO.read(inputStream);

        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        Image localImageBytes = Image.builder().bytes(imageBytes).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(localImageBytes)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Sets window dimensions to 1/2 screen size, unless image is smaller
```

```
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);

    }

    // Draws bounding boxes (if present) and label text.
    public void drawLabels() {

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.customLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.name();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
```

```
            if (customLabel.geometry() != null) {

                BoundingBox box = customLabel.geometry().boundingBox();
                float left = imageWidth * box.left();
                float top = imageHeight * box.top();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                        textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

                // Draw bounding box around label location
                g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.width())),
                        Math.round((imageHeight * box.height())));
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
        logger.info("Custom labels found:");
        if (customLabels.isEmpty()) {
```

```
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
 min confidence.");

        }
        else {
        for (CustomLabel customLabel : customLabels) {
            logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                    new Object[] { customLabel.name(),
 customLabel.confidence() } );
            }
        }
    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    public static void main(String args[]) throws Exception {

        String photo = null;
        String bucket = null;
        String projectVersionArn = null;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
                + "   bucket - The S3 bucket that contains the image. Don't
 specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
 assumed to be
        // in an S3 bucket.

        if (args.length < 2 || args.length > 3) {
```

```
            System.out.println(USAGE);
            System.exit(1);
        }

        projectVersionArn = args[0];
        photo = args[1];

        if (args.length == 3) {
            bucket = args[2];
        }

        float minConfidence = 50;

        ShowCustomLabels panel = null;

        try {
            // Get the Rekognition client


            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            S3Client s3Client = S3Client.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Create frame and panel.
            JFrame frame = new JFrame("Custom Labels");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            if (args.length == 2) {
                // Analyze local image
                panel = new ShowCustomLabels(rekClient, projectVersionArn,
 photo, minConfidence);
            } else {
                // Analyze image in S3 bucket
```

```
                panel = new ShowCustomLabels(rekClient, s3Client,
projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (RekognitionException rekError) {

            String errorMessage = "Rekognition client error: " +
rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchKeyException bucketError) {
            String errorMessage = String.format("Image not found: %s in bucket
%s.", photo, bucket);
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchBucketException bucketError) {
            String errorMessage = "Bucket not found: " + bucket;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
}
```

DetectCustomLabels solicitação de operação

Na operação `DetectCustomLabels`, você fornece uma imagem de entrada como uma matriz de bytes codificada em base64 ou como uma imagem armazenada em um bucket do Amazon S3. O exemplo de solicitação JSON a seguir mostra a imagem carregada de um bucket do Amazon S3.

```
{
    "ProjectVersionArn": "string",
     "Image":{
        "S3Object":{
            "Bucket":"string",
            "Name":"string",
            "Version":"string"
         }
    },
    "MinConfidence": 90,
    "MaxLabels": 10,
}
```

DetectCustomLabels resposta da operação

A resposta do JSON a seguir da operação `DetectCustomLabels` mostra os rótulos personalizados detectados na imagem a seguir.

```
{
    "CustomLabels": [
        {
            "Name": "MyLogo",
            "Confidence": 77.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

Como gerenciar os recursos do Amazon Rekognition Custom Labels

Esta seção fornece uma visão geral dos recursos do Amazon Rekognition Custom Labels usados para treinar e gerenciar um modelo. Também estão incluídas informações gerais sobre como usar o AWS SDK para treinar e usar um modelo.

O Amazon Rekognition Custom Labels depende de três recursos diferentes para detectar rótulos personalizados: projetos, conjuntos de dados e modelos.

- Projetos: usados para agrupar outros recursos, como conjuntos de dados, versões de modelos e avaliações de modelos.
- Conjuntos de dados: define imagens e metadados associados para uso em modelos de treinamento e teste. Você pode criar um conjunto de dados usando um arquivo de manifesto no formato SageMaker AI ou copiando um conjunto de dados existente do Amazon Rekognition Custom Labels.
- Modelos: o modelo matemático que prevê a presença de objetos, cenas e conceitos nas imagens ao identificar padrões nas imagens usadas para treinar o modelo.

Tópicos

- Como gerenciar um projeto do Amazon Rekognition Custom Labels
- Como gerenciar conjuntos de dados
- Como gerenciar um modelo do Amazon Rekognition Custom Labels

Como gerenciar um projeto do Amazon Rekognition Custom Labels

No Amazon Rekognition Custom Labels, um projeto é usado para gerenciar os modelos criados para um caso de uso específico. Um projeto gerencia conjuntos de dados, treinamento de modelos, versões de modelos, avaliação de modelos e a execução dos modelos do seu projeto.

Tópicos

- Como excluir um projeto do Amazon Rekognition Custom Labels
- Como descrever um projeto (SDK)

- Como criar um projeto com o AWS CloudFormation

Como excluir um projeto do Amazon Rekognition Custom Labels

Você pode excluir um projeto usando o console do Amazon Rekognition ou chamando a API. DeleteProject Para excluir um projeto, exclua primeiro cada modelo associado. Um projeto ou modelo excluído não pode ser recuperado.

Tópicos

- Como excluir um projeto do Amazon Rekognition Custom Labels (console)
- Como excluir um projeto do Amazon Rekognition Custom Labels (SDK)

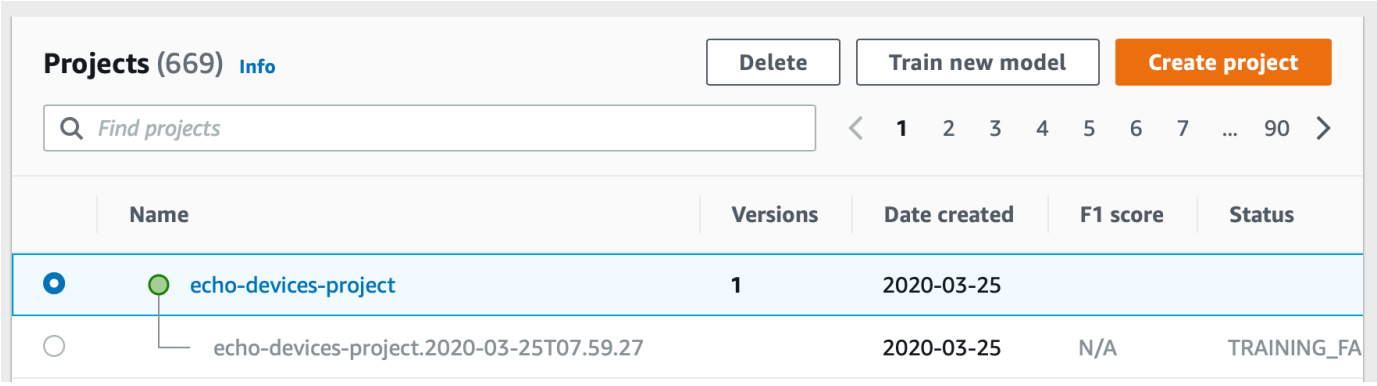
Como excluir um projeto do Amazon Rekognition Custom Labels (console)

É possível excluir um projeto da página de projetos ou excluir um projeto da página de detalhes de um projeto. O procedimento a seguir mostra como excluir um projeto usando a página de projetos.

O console do Amazon Rekognition Custom Labels exclui modelos e conjuntos de dados associados para você durante a exclusão do projeto. Não é possível excluir um projeto se algum de seus modelos estiver em execução ou em treinamento. Para interromper um modelo em execução, consulte Como interromper um modelo do Amazon Rekognition Custom Labels (SDK). Se um modelo estiver sendo treinado, espere até que ele termine antes de excluir o projeto.

Para excluir um projeto de (console)

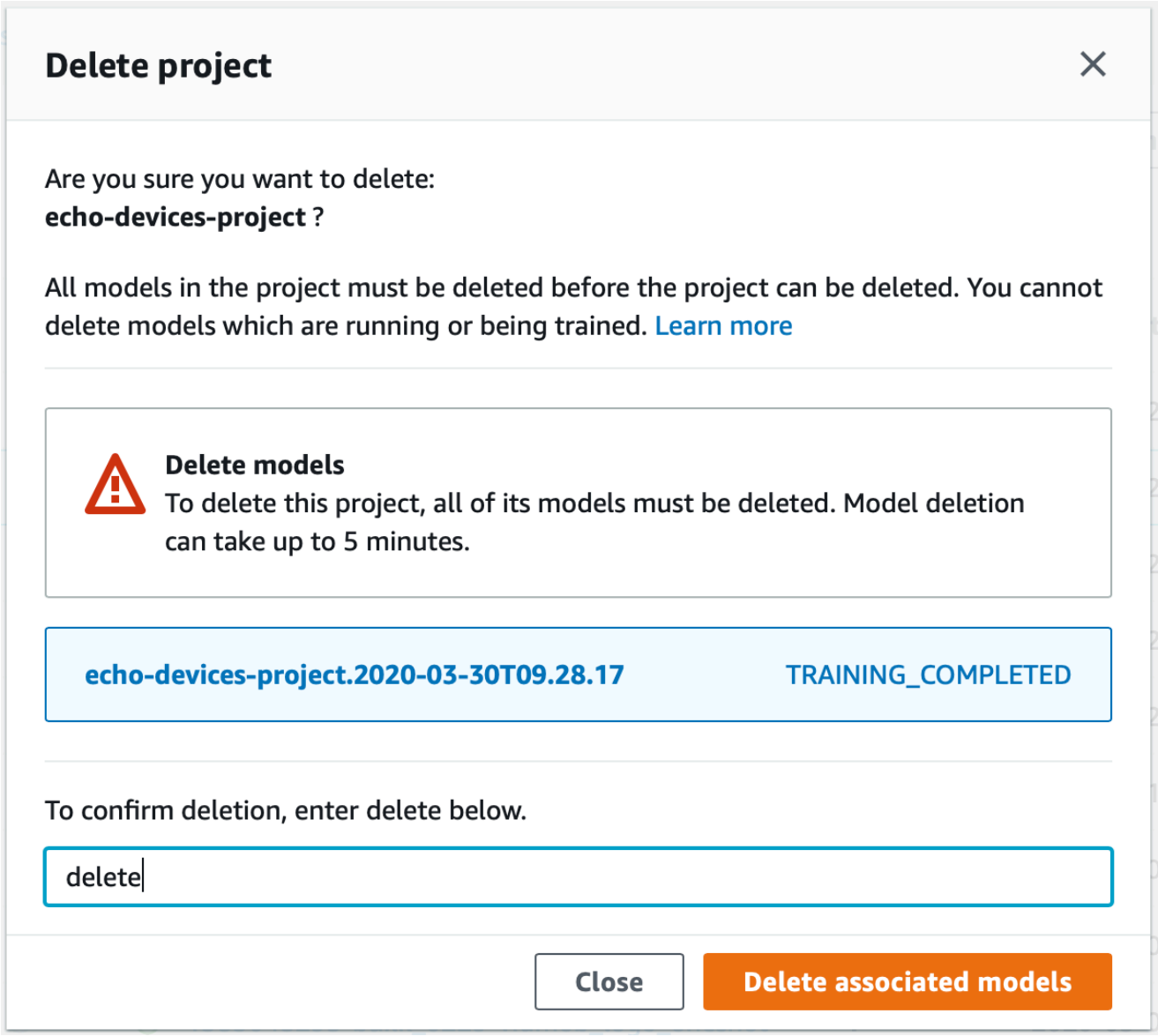
1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na página Projetos, selecione o botão ao lado do projeto que deseja excluir. A lista de projetos é exibida echo-devices-project, com 1 versão criada em 25/03/2020 e opções para excluir, treinar novo modelo ou criar projeto.



6. Escolha Excluir, no alto da página. A caixa de diálogo Excluir projeto é exibida.
7. Se o projeto não tiver modelos associados:
   a. insira Excluir para excluir o projeto.
   b. Escolha Excluir para excluir o projeto.
8. Se o projeto tiver modelos ou conjuntos de dados associados:
   a. Insira excluir para confirmar que deseja excluir os modelos e os conjuntos de dados.
   b. Escolha Excluir modelos associados, Excluir conjuntos de dados associados ou Excluir conjuntos de dados e modelos associados, dependendo se o modelo tem conjuntos de dados, modelos ou ambos. A exclusão do modelo pode demorar um pouco para ser concluída.

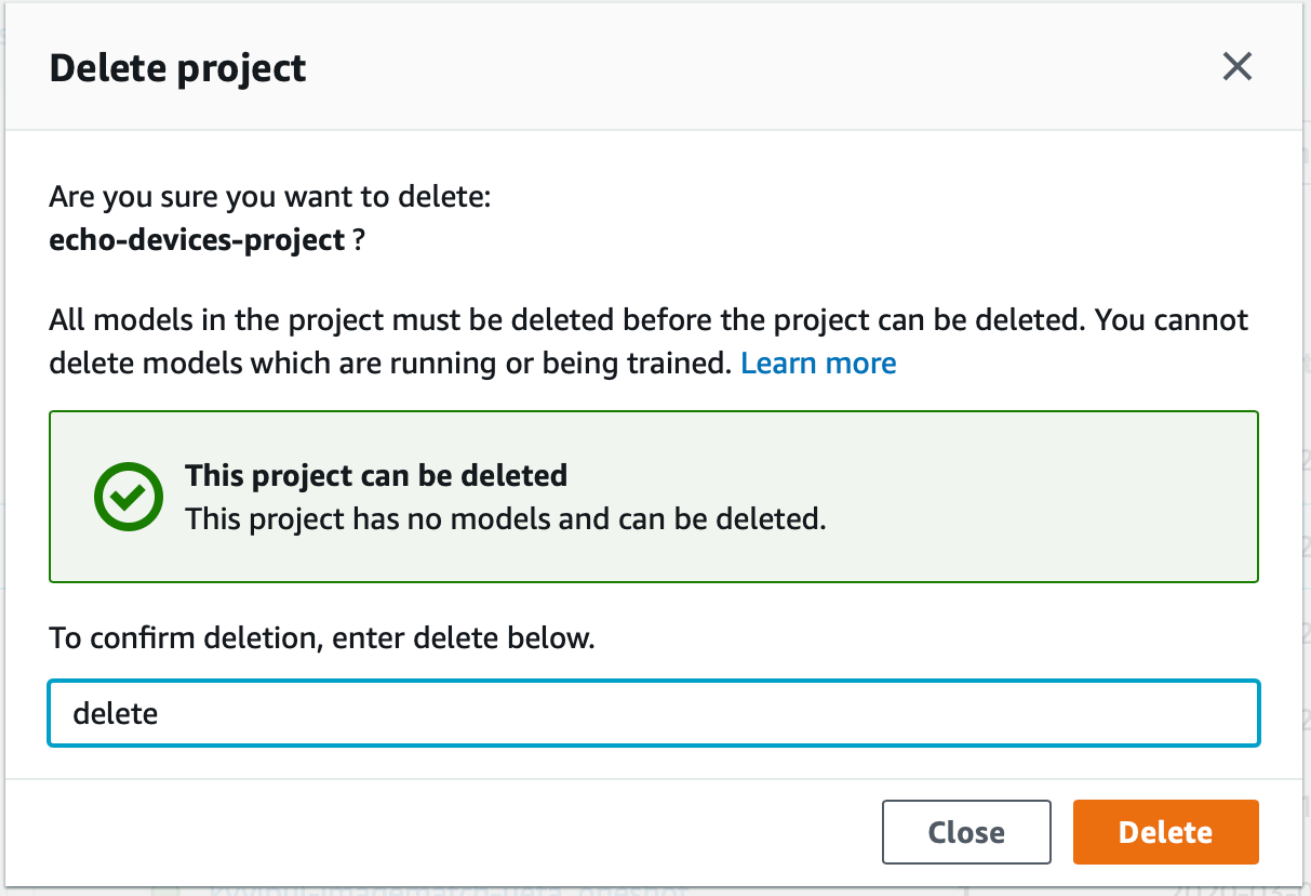
> **Note**
>
> O console não pode excluir modelos que estejam em treinamento ou em execução. Tente novamente depois de interromper qualquer modelo em execução listado e espere até que os modelos listados como treinamento terminem.
> Se Fechar a caixa de diálogo durante a exclusão do modelo, os modelos ainda serão excluídos. Posteriormente, é possível excluir o projeto repetindo esse procedimento.

O painel para excluir um modelo fornece instruções explícitas para excluir modelos associados.



c. Insira excluir para confirmar que você deseja excluir o projeto.

d. Escolha Excluir para excluir o projeto.



Como excluir um projeto do Amazon Rekognition Custom Labels (SDK)

Você exclui um projeto Amazon Rekognition Custom Labels DeleteProjectchamando e fornecendo o Amazon Resource Name (ARN) do projeto que você deseja excluir. Para obter ARNs os projetos em sua AWS conta, ligue DescribeProjects. A resposta inclui uma matriz de ProjectDescriptionobjetos. O ARN do projeto é o campo `ProjectArn`. É possível usar o nome do projeto para identificar o ARN do projeto. Por exemplo, `arn:aws:rekognition:us-east-1:123456789010:project/`*`project name`*`/1234567890123`.

Antes de excluir um projeto, exclua primeiro todos os modelos e conjuntos de dados no projeto. Para ter mais informações, consulte Como excluir um modelo do Amazon Rekognition Custom Labels (SDK) e Como excluir um conjunto de dados.

O projeto pode demorar alguns instantes para ser excluído. Durante esse período, o status do projeto é `DELETING`. O projeto será excluído se uma chamada subsequente para DescribeProjects não incluir o projeto que você excluiu.

Para excluir um projeto (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para excluir um projeto.

   AWS CLI

   Altere o valor de `project-arn` para o nome do projeto que você deseja excluir.

```
aws rekognition delete-project --project-arn project_arn \
  --profile custom-labels-access
```

   Python

   Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

   - `project_arn`: o ARN do projeto que você deseja excluir.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels project example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/mp-delete-
project.html
Shows how to delete an existing Amazon Rekognition Custom Labels project.
You must first delete any models and datasets that belong to the project.
"""

import argparse
import logging
import time
import boto3


from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_project(rek_client, project_arn):
    """
    Deletes an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to delete.
    """

    try:
        # Delete the project
        logger.info("Deleting project: %s", project_arn)

        response = rek_client.delete_project(ProjectArn=project_arn)

        logger.info("project status: %s",response['Status'])

        deleted = False

        logger.info("waiting for project deletion: %s", project_arn)

        # Get the project name
        start = find_forward_slash(project_arn, 1) + 1
        end = find_forward_slash(project_arn, 2)
        project_name = project_arn[start:end]

        project_names = [project_name]

        while deleted is False:
```

```
            project_descriptions = rek_client.describe_projects(
                ProjectNames=project_names)['ProjectDescriptions']

            if len(project_descriptions) == 0:
                deleted = True

            else:
                time.sleep(5)

        logger.info("project deleted: %s",project_arn)

        return True

    except ClientError as err:
        logger.exception(
            "Couldn't delete project - %s: %s",
            project_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that you want to delete."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Deleting project: {args.project_arn}")
```

```
        # Delete the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        delete_project(rekognition_client,
                       args.project_arn)

        print(f"Finished deleting project: {args.project_arn}")

    except ClientError as err:
        error_message = f"Problem deleting project: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- `project_arn`: o ARN do projeto que você deseja excluir.

```
/*
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.List;
import java.util.Objects;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectResponse;
```

```
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProject {

    public static final Logger logger =
 Logger.getLogger(DeleteProject.class.getName());

    public static void deleteMyProject(RekognitionClient rekClient, String
 projectArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting project: {0}", projectArn);

            // Delete the project

            DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder().projectArn(projectArn).build();
            DeleteProjectResponse response =
 rekClient.deleteProject(deleteProjectRequest);

            logger.log(Level.INFO, "Status: {0}", response.status());

            // Wait until deletion finishes

            Boolean deleted = false;

            do {

        DescribeProjectsRequest describeProjectsRequest =
 DescribeProjectsRequest.builder().build();
                    DescribeProjectsResponse describeResponse =
 rekClient.describeProjects(describeProjectsRequest);
                    List<ProjectDescription> projectDescriptions =
 describeResponse.projectDescriptions();

                    deleted = true;
```

```
                    for (ProjectDescription projectDescription :
 projectDescriptions) {

                        if (Objects.equals(projectDescription.projectArn(),
 projectArn)) {
                            deleted = false;
                            logger.log(Level.INFO, "Not deleted: {0}",
 projectDescription.projectArn());
                            Thread.sleep(5000);
                            break;
                        }
                    }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Project deleted: {0} ", projectArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<project_arn>\n\n" + "Where:\n"
             + "   project_arn - The ARN of the project that you want to delete.
\n\n";

        if (args.length != 1) {
             System.out.println(USAGE);
          System.exit(1);
        }

        String projectArn = args[0];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
                .region(Region.US_WEST_2)
```

```
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .build();

            // Delete the project.
            deleteMyProject(rekClient, projectArn);

            System.out.println(String.format("Project deleted: %s",
 projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }
    }

}
```

Como descrever um projeto (SDK)

É possível usar a API `DescribeProjects` para obter informações sobre seus projetos.

Para descrever um projeto (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir para descrever um projeto. Substitua `project_name` pelo nome do esquema que deseja descrever. Se não especificar `--project-names`, as descrições de todos os projetos são retornadas.

AWS CLI

```
aws rekognition describe-projects --project-names project_name \
  --profile custom-labels-access
```

Python

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- project_name : o nome do projeto que deseja descrever. Se não especificar um nome, as descrições de todos os projetos são retornadas.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to describe an Amazon Rekognition Custom Labels project.
"""
import argparse
import logging
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def display_project_info(project):
    """
    Displays information about a Custom Labels project.
    :param project: The project that you want to display information about.
    """
    print(f"Arn: {project['ProjectArn']}")
    print(f"Status: {project['Status']}")

    if len(project['Datasets']) == 0:
        print("Datasets: None")
    else:
        print("Datasets:")
```

```
    for dataset in project['Datasets']:
        print(f"\tCreated: {str(dataset['CreationTimestamp'])}")
        print(f"\tType: {dataset['DatasetType']}")
        print(f"\tARN: {dataset['DatasetArn']}")
        print(f"\tStatus: {dataset['Status']}")
        print(f"\tStatus message: {dataset['StatusMessage']}")
        print(f"\tStatus code: {dataset['StatusMessageCode']}")
        print()
    print()


def describe_projects(rek_client, project_name):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The project you want to describe. Pass None to describe
 all projects.
    """

    try:
        # Describe the project
        if project_name is None:
            logger.info("Describing all projects.")
        else:
            logger.info("Describing project: %s.",project_name)

        if project_name is None:
            response = rek_client.describe_projects()
        else:
            project_names = json.loads('["' + project_name + '"]')
            response = rek_client.describe_projects(ProjectNames=project_names)

        print('Projects\n--------')
        if len(response['ProjectDescriptions']) == 0:
            print("Project(s) not found.")
        else:
            for project in response['ProjectDescriptions']:
                display_project_info(project)

        logger.info("Finished project description.")

    except ClientError as err:
        logger.exception(
            "Couldn't describe project - %s: %s",
```

```
            project_name,err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "--project_name",  help="The name of the project that you want to
 describe.", required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)

        args = parser.parse_args()

        print(f"Describing projects: {args.project_name}")

        # Describe the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_projects(rekognition_client,
                          args.project_name)

        if args.project_name is None:
            print("Finished describing all projects.")
        else:
            print("Finished describing project %s.", args.project_name)

    except ClientError as err:
```

```
        error_message = f"Problem describing project: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- `project_name`: o ARN do projeto que você deseja descrever. Se não especificar um nome, as descrições de todos os projetos são retornadas.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DescribeProjects {

    public static final Logger logger =
 Logger.getLogger(DescribeProjects.class.getName());
```

```
   public static void describeMyProjects(RekognitionClient rekClient, String
projectName) {

       DescribeProjectsRequest descProjects = null;

       // If a single project name is supplied, build projectNames argument

       List<String> projectNames = new ArrayList<String>();

       if (projectName == null) {
           descProjects = DescribeProjectsRequest.builder().build();
       } else {
           projectNames.add(projectName);
           descProjects =
DescribeProjectsRequest.builder().projectNames(projectNames).build();
       }

       // Display useful information for each project.

       DescribeProjectsResponse resp =
rekClient.describeProjects(descProjects);

       for (ProjectDescription projectDescription : resp.projectDescriptions())
{

           System.out.println("ARN: " + projectDescription.projectArn());
           System.out.println("Status: " +
projectDescription.statusAsString());
           if (projectDescription.hasDatasets()) {
               for (DatasetMetadata datasetDescription :
projectDescription.datasets()) {
                   System.out.println("\tdataset Type: " +
datasetDescription.datasetTypeAsString());
                   System.out.println("\tdataset ARN: " +
datasetDescription.datasetArn());
                   System.out.println("\tdataset Status: " +
datasetDescription.statusAsString());
               }
           }
           System.out.println();
       }

   }
```

```
    public static void main(String[] args) {

        String projectArn = null;

        // Get command line arguments

        final String USAGE = "\n" + "Usage: " + "<project_name>\n\n" + "Where:
\n"
                + "   project_name - (Optional) The name of the project that you
 want to describe. If not specified, all projects "
                + "are described.\n\n";

        if (args.length > 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        if (args.length == 1) {
            projectArn = args[0];
        }

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Describe projects

            describeMyProjects(rekClient, projectArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

Como criar um projeto com o AWS CloudFormation

O Amazon Rekognition Custom Labels é AWS CloudFormation integrado ao, um serviço que ajuda você a modelar e AWS configurar seus recursos para que você possa gastar menos tempo criando e gerenciando seus recursos e infraestrutura. Você cria um modelo que descreve todos os AWS recursos que você deseja e se AWS CloudFormation encarrega de provisionar e configurar esses recursos para você.

Você pode usar AWS CloudFormation para provisionar e configurar projetos de etiquetas personalizadas do Amazon Rekognition.

Ao usar AWS CloudFormation, você pode reutilizar seu modelo para configurar seus projetos de etiquetas personalizadas do Amazon Rekognition de forma consistente e repetida. Basta descrever seus projetos uma vez e provisionar os mesmos projetos repetidamente em várias AWS contas e regiões.

Etiquetas e modelos personalizados do Amazon Rekognition AWS CloudFormation

Para provisionar e configurar recursos para o Amazon Rekognition Custom Labels e serviços relacionados, é preciso entender os modelos do AWS CloudFormation. Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os recursos que você deseja provisionar em suas AWS CloudFormation pilhas. Se você não estiver familiarizado com JSON ou YAML, você pode usar o AWS CloudFormation Designer para ajudá-lo a começar a usar modelos. AWS CloudFormation Para obter mais informações, consulte O que é o Designer AWS CloudFormation ? no Manual do usuário do AWS CloudFormation .

Para obter informações de referência sobre projetos do Amazon Rekognition Custom Labels, incluindo exemplos de modelos JSON e YAML, consulte Referência do tipo de recurso do Rekognition.

Saiba mais sobre AWS CloudFormation

Para saber mais sobre isso AWS CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- AWS CloudFormation Guia do usuário

- AWS CloudFormation API Reference
- AWS CloudFormation Guia do usuário da interface de linha de comando

Como gerenciar conjuntos de dados

Um conjunto de dados contém as imagens e os rótulos atribuídos que você usa para treinar ou testar um modelo. Os tópicos desta seção mostram como gerenciar um conjunto de dados com o console Amazon Rekognition Custom Labels e o SDK. AWS

Tópicos

- Como adicionar um conjunto de dados a um projeto
- Como adicionar mais imagens a um conjunto de dados
- Como criar um conjunto de dados usando um conjunto de dados existente (SDK)
- Como descrever um conjunto de dados (SDK)
- Como listar entradas do conjunto de dados (SDK)
- Como distribuir um conjunto de dados de treinamento (SDK)
- Como excluir um conjunto de dados

Como adicionar um conjunto de dados a um projeto

É possível adicionar um conjunto de dados de treinamento ou teste a um projeto existente. Se quiser substituir um conjunto de dados existente, primeiro exclua o conjunto de dados existente. Para obter mais informações, consulte Como excluir um conjunto de dados. Em seguida, adicione o novo conjunto de dados.

Tópicos

- Como adicionar um conjunto de dados a um projeto (console)
- Como adicionar um conjunto de dados a um projeto (SDK)

Como adicionar um conjunto de dados a um projeto (console)

É possível adicionar um conjunto de dados de treinamento ou teste a um projeto usando o console do Amazon Rekognition Custom Labels.

Para adicionar um conjunto de dados a um projeto

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Usar rótulos personalizados. A página inicial do Amazon Rekognition Custom Labels é exibida.
3. No painel de navegação esquerdo, selecione Projetos. A visualização Projetos é exibida.
4. Escolha o projeto ao qual você deseja adicionar um conjunto de dados.
5. No painel de navegação esquerdo, abaixo do nome do projeto, escolha Conjuntos de dados.
6. Se o projeto não tiver um conjunto de dados existente, a página Criar conjunto de dados será exibida. Faça o seguinte:

   a. Na página Criar conjunto de dados, insira as informações da fonte da imagem. Para obter mais informações, consulte the section called “Como criar conjuntos de dados com imagens”.

   b. Escolha Criar conjunto de dados para criar o conjunto de dados.
7. Se o projeto tiver um conjunto de dados existente (treinamento ou teste), a página de detalhes do projeto será exibida. Faça o seguinte:

   a. Na página de detalhes do projeto, escolha Ações.

   b. Se quiser adicionar um conjunto de dados de treinamento, escolha Criar conjunto de dados de teste.

   c. Se quiser adicionar um conjunto de dados de teste, escolha Criar conjunto de dados de teste.

   d. Na página Criar conjunto de dados, insira as informações da fonte da imagem. Para obter mais informações, consulte the section called “Como criar conjuntos de dados com imagens”.

   e. Escolha Criar conjunto de dados para criar o conjunto de dados.
8. Adicione imagens ao seu conjunto de dados. Para obter mais informações, consulte Como adicionar mais imagens (console).
9. Adicione rótulos ao seu conjunto de dados. Para obter mais informações, consulte Adicionar novos rótulos (console).
10. Adicione rótulos às suas imagens. Se estiver adicionando rótulos em nível de imagem, consulte the section called “Como atribuir rótulos em nível de imagem em uma imagem”. Se estiver adicionando caixas delimitadoras, consulte Como rotular objetos com caixas delimitadoras. Para obter mais informações, consulte Como definir os conjuntos de dados.

Como adicionar um conjunto de dados a um projeto (SDK)

É possível adicionar um conjunto de dados de treinamento ou teste a um projeto existente das seguintes maneiras:

- Criar um conjunto de dados usando um arquivo de manifesto. Para obter mais informações, consulte Criação de um conjunto de dados com um arquivo de manifesto (SDK) do SageMaker AI Ground Truth.
- Crie um conjunto de dados vazio e preencha o conjunto de dados depois. O exemplo a seguir mostra como criar um conjunto de dados vazio. Para adicionar entradas depois de criar um conjunto de dados vazio, consulte Como adicionar mais imagens a um conjunto de dados.

Para adicionar um conjunto de dados a um projeto (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use os exemplos a seguir para adicionar linhas JSON a um conjunto de dados.

   CLI

   Substitua `project_arn` pelo projeto ao qual deseja adicionar o conjunto de dados. Substitua `dataset_type` por `TRAIN` para criar um conjunto de dados de treinamento ou `TEST` para criar um conjunto de dados de teste.

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type dataset_type \
     --profile custom-labels-access
   ```

   Python

   Use o código a seguir para criar um conjunto de dados. Forneça as seguintes opções de linha de comando:

   - `project_arn`: o ARN do projeto ao qual você deseja adicionar o conjunto de dados de teste.
   - `type`: o tipo de conjunto de dados que você deseja criar (treinamento ou teste).

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   ```

```
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_empty_dataset(rek_client, project_arn, dataset_type):
    """
    Creates an empty Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    """

    try:
        #Create the dataset.
        logger.info("Creating empty %s dataset for project %s",
            dataset_type, project_arn)

        dataset_type=dataset_type.upper()

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s", dataset_arn)

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":
```

```
                logger.info(("Creating dataset: %s ", dataset_arn))
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished=True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception(error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn

    except ClientError as err:
        logger.exception("Couldn't create dataset: %s", err.response['Error']
['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the empty dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the empty dataset that you want to
 create (train or test)."
    )
```

```
def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating empty {args.dataset_type} dataset for project
 {args.project_arn}")

        # Create the empty dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_empty_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type.lower())

        print(f"Finished creating empty dataset: {dataset_arn}")


    except ClientError as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Use o código a seguir para criar um conjunto de dados. Forneça as seguintes opções de linha de comando:

- `project_arn`: o ARN do projeto ao qual você deseja adicionar o conjunto de dados de teste.
- `type`: o tipo de conjunto de dados que você deseja criar (treinamento ou teste).

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateEmptyDataset {

    public static final Logger logger =
 Logger.getLogger(CreateEmptyDataset.class.getName());

    public static String createMyEmptyDataset(RekognitionClient rekClient,
 String projectArn, String datasetType)
            throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating empty {0} dataset for project :
 {1}",
                    new Object[] { datasetType.toString(), projectArn });
```

```
            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
datasetType);
                throw new Exception("Unrecognized dataset type: " +
datasetType);

            }

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)
                    .datasetType(requestDatasetType).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until updates finishes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());
```

```
                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }


    public static void main(String args[]) {
```

```
        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>\n
\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
 copy the datast to.\n\n"
                + "   dataset_type - the type of the empty dataset that you want
 to create (train or test).\n\n";


        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the dataset
            datasetArn = createMyEmptyDataset(rekClient, projectArn,
 datasetType);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
```

```
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. Adicione imagens ao conjunto de dados. Para obter mais informações, consulte Como adicionar mais imagens (SDK).

Como adicionar mais imagens a um conjunto de dados

É possível adicionar mais imagens aos seus conjuntos de dados usando o console do Amazon Rekognition Custom Labels ou chamando a API `UpdateDatasetEntries`.

Tópicos

- Como adicionar mais imagens (console)
- Como adicionar mais imagens (SDK)

Como adicionar mais imagens (console)

Ao usar o console do Amazon Rekognition Custom Labels, as imagens são carregadas do seu computador local. As imagens são adicionadas ao local do bucket do Amazon S3 (console ou externo) onde as imagens usadas para criar o conjunto de dados são armazenadas.

Para adicionar mais imagens ao seu conjunto de dados (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Usar rótulos personalizados. A página inicial do Amazon Rekognition Custom Labels é exibida.
3. No painel de navegação esquerdo, selecione Projetos. A visualização Projetos é exibida.
4. Escolha o projeto que você deseja usar.
5. No painel de navegação esquerdo, abaixo do nome do projeto, escolha Conjunto de dados.
6. Escolha Ações e selecione o conjunto de dados ao qual você deseja adicionar imagens.

7. Escolha as imagens que você deseja fazer upload no conjunto de dados. É possível arrastar as imagens ou escolher as imagens que deseja carregar do seu computador local. É possível fazer upload de até 30 imagens por vez.
8. Escolha Fazer upload de imagens.
9. Escolha Salvar alterações.
10. Rotule as imagens. Para obter mais informações, consulte Rotulagem de imagens.

Como adicionar mais imagens (SDK)

`UpdateDatasetEntries` atualiza ou adiciona linhas JSON a um arquivo de manifesto. Passe as linhas JSON como um objeto de dados codificado em byte64 no campo `GroundTruth`. Se você estiver usando um AWS SDK para fazer chamadas`UpdateDatasetEntries`, o SDK codifica os dados para você. Cada linha JSON contém informações para uma única imagem, como rótulos atribuídos ou informações da caixa delimitadora. Por exemplo:

```
{"source-ref":"s3://bucket/image","BB":{"annotations":
[{"left":1849,"top":1039,"width":422,"height":283,"class_id":0},
{"left":1849,"top":1340,"width":443,"height":415,"class_id":1},
{"left":2637,"top":1380,"width":676,"height":338,"class_id":2},
{"left":2634,"top":1051,"width":673,"height":338,"class_id":3}],"image_size":
[{"width":4000,"height":2667,"depth":3}]},"BB-metadata":{"job-name":"labeling-job/
BB","class-map":
{"0":"comparator","1":"pot_resistor","2":"ir_phototransistor","3":"ir_led"},"human-
annotated":"yes","objects":[{"confidence":1},{"confidence":1},{"confidence":1},
{"confidence":1}],"creation-date":"2021-06-22T10:11:18.006Z","type":"groundtruth/
object-detection"}}
```

Para obter mais informações, consulte Criar um arquivo de manifesto.

Use o campo `source-ref` como uma chave para identificar as imagens que você deseja atualizar. Se o conjunto de dados não contiver um valor de campo `source-ref` correspondente, a linha JSON será adicionada como uma nova imagem.

Para adicionar mais imagens a um conjunto de dados (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use os exemplos a seguir para adicionar linhas JSON a um conjunto de dados.

CLI

Substitua o valor de `GroundTruth` pelas linhas JSON que você deseja usar. É necessário escapar de qualquer caractere especial dentro da linha JSON.

```
aws rekognition update-dataset-entries\
  --dataset-arn dataset_arn \
  --changes '{"GroundTruth" : "{\"source-ref\":\"s3://your_bucket/your_image
\",\"BB\":{\"annotations\":[{\"left\":1776,\"top\":1017,\"width\":458,\"height
\":317,\"class_id\":0},{\"left\":1797,\"top\":1334,\"width\":418,\"height
\":415,\"class_id\":1},{\"left\":2597,\"top\":1361,\"width\":655,\"height
\":329,\"class_id\":2},{\"left\":2581,\"top\":1020,\"width\":689,\"height
\":338,\"class_id\":3}],\"image_size\":[{\"width\":4000,\"height\":2667,
\"depth\":3}]},\"BB-metadata\":{\"job-name\":\"labeling-job/BB\",\"class-map
\":{\"0\":\"comparator\",\"1\":\"pot_resistor\",\"2\":\"ir_phototransistor\",
\"3\":\"ir_led\"},\"human-annotated\":\"yes\",\"objects\":[{\"confidence\":1},
{\"confidence\":1},{\"confidence\":1},{\"confidence\":1}],\"creation-date\":
\"2021-06-22T10:10:48.492Z\",\"type\":\"groundtruth/object-detection\"}}" }' \
  --cli-binary-format raw-in-base64-out \
  --profile custom-labels-access
```

Python

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- dataset_arn: o ARN do conjunto de dados que você deseja atualizar.
- updates_file : o arquivo que contém as atualizações da linha JSON.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to add entries to an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import time
import json
```

```
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def update_dataset_entries(rek_client, dataset_arn, updates_file):
    """
    Adds dataset entries to an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that yuo want to update.
    :param updates_file: The manifest file of JSON Lines that contains the
 updates.
    """

    try:
        status=""
        status_message=""

        # Update dataset entries.
        logger.info("Updating dataset %s", dataset_arn)


        with open(updates_file) as f:
            manifest_file = f.read()


        changes=json.loads('{ "GroundTruth" : ' +
            json.dumps(manifest_file) +
            '}')

        rek_client.update_dataset_entries(
            Changes=changes, DatasetArn=dataset_arn
        )

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']
            status_message=dataset['DatasetDescription']['StatusMessage']

            if status == "UPDATE_IN_PROGRESS":
```

```
                logger.info("Updating dataset: %s ", dataset_arn)
                time.sleep(5)
                continue

            if status == "UPDATE_COMPLETE":
                logger.info("Dataset updated: %s : %s : %s",
                    status, status_message, dataset_arn)
                finished=True
                continue

            if status == "UPDATE_FAILED":
                error_message = f"Dataset update failed: {status} :
 {status_message} : {dataset_arn}"
                logger.exception(error_message)
                raise Exception (error_message)

            error_message = f"Failed. Unexpected state for dataset update:
 {status} : {status_message} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        logger.info("Added entries to dataset")

        return status, status_message


    except ClientError as err:
        logger.exception("Couldn't update dataset: %s", err.response['Error']
['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to update."
    )

    parser.add_argument(
```

```
        "updates_file", help="The manifest file of JSON Lines that contains the
 updates."
    )

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        #get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Updating dataset {args.dataset_arn} with entries from
 {args.updates_file}.")

        # Update the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status, status_message=update_dataset_entries(rekognition_client,
            args.dataset_arn,
            args.updates_file)

        print(f"Finished updates dataset: {status} : {status_message}")


    except ClientError as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")

    except Exception as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

- dataset_arn: o ARN do conjunto de dados que você deseja atualizar.
- update_file : o arquivo que contém as atualizações da linha JSON.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetChanges;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesRequest;
import
 software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesResponse;

import java.io.FileInputStream;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;

public class UpdateDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(UpdateDatasetEntries.class.getName());

    public static String updateMyDataset(RekognitionClient rekClient, String
 datasetArn,
            String updateFile
            ) throws Exception, RekognitionException {
```

```
        try {

            logger.log(Level.INFO, "Updating dataset {0}",
                    new Object[] { datasetArn});


            InputStream sourceStream = new FileInputStream(updateFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            DatasetChanges datasetChanges = DatasetChanges.builder()
                    .groundTruth(sourceBytes).build();

            UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
UpdateDatasetEntriesRequest.builder()
                    .changes(datasetChanges)
                    .datasetArn(datasetArn)
                    .build();

            UpdateDatasetEntriesResponse response =
rekClient.updateDatasetEntries(updateDatasetEntriesRequest);

            boolean updated = false;

            //Wait until update completes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(datasetArn).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

                switch (status) {

                case UPDATE_COMPLETE:
```

```
                    logger.log(Level.INFO, "Dataset updated");
                    updated = true;
                    break;

                case UPDATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case UPDATE_FAILED:
                    String error = "Dataset update failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
datasetArn;
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected update state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
datasetArn;
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (updated == false);

            return datasetArn;

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not update dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String updatesFile = null;
        String datasetArn = null;
```

```
        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
 <updates_file>\n\n" + "Where:\n"
                + "   dataset_arn - the ARN of the dataset that you want to
 update.\n\n"
                + "   update_file - The file that includes in JSON Line updates.
\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        datasetArn = args[0];
        updatesFile = args[1];


        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

             // Update the dataset
            datasetArn = updateMyDataset(rekClient, datasetArn, updatesFile);

            System.out.println(String.format("Dataset updated: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

Como criar um conjunto de dados usando um conjunto de dados existente (SDK)

O procedimento a seguir mostra como criar um conjunto de dados a partir de um conjunto de dados existente usando a CreateDatasetoperação.

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir para criar um conjunto de dados ao copiar outro conjunto de dados.

   AWS CLI

   Use o código a seguir para criar o conjunto de dados. Substitua o seguinte:

   - `project_arn`: o ARN do projeto ao qual você deseja adicionar o conjunto de dados.
   - `dataset_type`: com o tipo de conjunto de dados (`TRAIN` ou`TEST`) que você deseja criar no projeto.
   - `dataset_arn`: com o ARN do conjunto de dados que você deseja copiar.

```
aws rekognition create-dataset --project-arn project_arn \
  --dataset-type dataset_type \
  --dataset-source '{ "DatasetArn" : "dataset_arn" }' \
  --profile custom-labels-access
```

   Python

   O exemplo a seguir cria um conjunto de dados usando um conjunto de dados existente e exibe seu ARN.

   Para executar o programa, forneça os seguintes argumentos de linha de comando:

   - `project_arn`: o ARN do projeto que você deseja usar.
   - `dataset_type`: o tipo do conjunto de dados do projeto que você deseja criar (`train` ou `test`).

- `dataset_arn`: o ARN do conjunto de dados do qual você deseja criar o conjunto de dados.

```
# Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def create_dataset_from_existing_dataset(rek_client, project_arn, dataset_type,
 dataset_arn):
    """
    Creates an Amazon Rekognition Custom Labels dataset using an existing
 dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    :param dataset_arn: The ARN of the existing dataset that you want to use.
    """

    try:
        # Create the dataset

        dataset_type=dataset_type.upper()

        logger.info(
            "Creating %s dataset for project %s from dataset %s.",
                dataset_type,project_arn, dataset_arn)

        dataset_source = json.loads(
```

```
            '{ "DatasetArn": "' + dataset_arn + '"}'
        )

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type,
DatasetSource=dataset_source
        )

        dataset_arn = response['DatasetArn']

        logger.info("New dataset ARN: %s", dataset_arn)

        finished = False
        while finished is False:

            dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

            status = dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":

                logger.info(("Creating dataset: %s ", dataset_arn))
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished = True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
{dataset_arn}"
                logger.exception(error_message)
                raise Exception(error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
{status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn

    except ClientError as err:
```

```
        logger.exception(
            "Couldn't create dataset: %s",err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
 (train or test)."
    )

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to copy from."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Creating {args.dataset_type} dataset for project
 {args.project_arn}")

        # Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
```

```
        rekognition_client = session.client("rekognition")

        dataset_arn = create_dataset_from_existing_dataset(rekognition_client,
                                    args.project_arn,
                                    args.dataset_type,
                                    args.dataset_arn)

        print(f"Finished creating dataset: {dataset_arn}")

    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

O exemplo a seguir cria um conjunto de dados usando um conjunto de dados existente e exibe seu ARN.

Para executar o programa, forneça os seguintes argumentos de linha de comando:

- `project_arn`: o ARN do projeto que você deseja usar.
- `dataset_type`: o tipo do conjunto de dados do projeto que você deseja criar (`train` ou `test`).
- `dataset_arn`: o ARN do conjunto de dados do qual você deseja criar o conjunto de dados.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateDatasetExisting {

    public static final Logger logger =
 Logger.getLogger(CreateDatasetExisting.class.getName());

    public static String createMyDataset(RekognitionClient rekClient, String
 projectArn, String datasetType,
            String existingDatasetArn) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
 dataset {2} ",
                    new Object[] { datasetType.toString(), projectArn,
 existingDatasetArn });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
 datasetType);
```

```
                throw new Exception("Unrecognized dataset type: " +
datasetType);

            }

            DatasetSource datasetSource =
DatasetSource.builder().datasetArn(existingDatasetArn).build();

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until create finishes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
```

```
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;
        String datasetSourceArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
```

```
                + "   dataset_type - the type of the dataset that you want to
 create (train or test).\n\n"
                + "   dataset_arn - the ARN of the dataset that you want to copy
 from.\n\n";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];
        datasetSourceArn = args[2];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the dataset
            datasetArn = createMyDataset(rekClient, projectArn, datasetType,
 datasetSourceArn);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

Como descrever um conjunto de dados (SDK)

É possível usar a API `DescribeDataset` para obter informações sobre um conjunto de dados.

Para descrever um conjunto de dados (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir para descrever um conjunto de dados.

   AWS CLI

   Altere o valor de `dataset-arn` para o ARN do conjunto de dados que você deseja escrever.

   ```
   aws rekognition describe-dataset --dataset-arn dataset_arn \
     --profile custom-labels-access
   ```

   Python

   Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

   - dataset_arn: o ARN do conjunto de dados que você deseja descrever.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Shows how to describe an Amazon Rekognition Custom Labels dataset.
   """

   import argparse
   import logging
   import boto3

   from botocore.exceptions import ClientError

   logger = logging.getLogger(__name__)
   ```

```
def describe_dataset(rek_client, dataset_arn):
    """
    Describes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to describe.

    """

    try:
        # Describe the dataset
        logger.info("Describing dataset %s", dataset_arn)

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        description = dataset['DatasetDescription']

        print(f"Created: {str(description['CreationTimestamp'])}")
        print(f"Updated: {str(description['LastUpdatedTimestamp'])}")
        print(f"Status: {description['Status']}")
        print(f"Status message: {description['StatusMessage']}")
        print(f"Status code: {description['StatusMessageCode']}")
        print("Stats:")
        print(
            f"\tLabeled entries: {description['DatasetStats']
['LabeledEntries']}")
        print(
            f"\tTotal entries: {description['DatasetStats']['TotalEntries']}")
        print(f"\tTotal labels: {description['DatasetStats']['TotalLabels']}")

    except ClientError as err:
        logger.exception("Couldn't describe dataset: %s",
                         err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
```

```
        "dataset_arn", help="The ARN of the dataset that you want to describe."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Describing dataset {args.dataset_arn}")

        # Describe the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_dataset(rekognition_client, args.dataset_arn)

        print(f"Finished describing dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message=f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

- dataset_arn: o ARN do conjunto de dados que você deseja descrever.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStats;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class DescribeDataset {

    public static final Logger logger =
 Logger.getLogger(DescribeDataset.class.getName());

    public static void describeMyDataset(RekognitionClient rekClient, String
 datasetArn) {

        try {

            DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();
            DescribeDatasetResponse describeDatasetResponse =
 rekClient.describeDataset(describeDatasetRequest);

            DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();
            DatasetStats datasetStats = datasetDescription.datasetStats();

            System.out.println("ARN: " + datasetArn);
```

```
            System.out.println("Created: " +
 datasetDescription.creationTimestamp().toString());
            System.out.println("Updated: " +
 datasetDescription.lastUpdatedTimestamp().toString());
            System.out.println("Status: " +
 datasetDescription.statusAsString());
            System.out.println("Message: " +
 datasetDescription.statusMessage());
            System.out.println("Total Labels: " +
 datasetStats.totalLabels().toString());
            System.out.println("Total entries: " +
 datasetStats.totalEntries().toString());
            System.out.println("Entries with labels: " +
 datasetStats.labeledEntries().toString());
            System.out.println("Entries with at least 1 error: " +
 datasetStats.errorEntries().toString());

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            throw rekError;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
                + "   dataset_arn - The ARN of the dataset that you want to
 describe.\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String datasetArn = args[0];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
```

```
            .region(Region.US_WEST_2)
            .build();

             // Describe the dataset.
            describeMyDataset(rekClient, datasetArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Como listar entradas do conjunto de dados (SDK)

É possível usar a API `ListDatasetEntries` para listar as linhas JSON de cada imagem em um conjunto de dados. Para obter mais informações, consulte Criar um arquivo de manifesto.

Para listar entradas do conjunto de dados (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir: liste as entradas em um conjunto de dados.

   AWS CLI

   Altere o valor de `dataset-arn` para o ARN do conjunto de dados que você deseja listar.

   ```
   aws rekognition list-dataset-entries --dataset-arn dataset_arn \
     --profile custom-labels-access
   ```

   Para listar somente linhas JSON com erros, especifique `has-errors`.

   ```
   aws rekognition list-dataset-entries --dataset-arn dataset_arn \
     --has-errors \
   ```

```
  --profile custom-labels-access
```

Python

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- dataset_arn: o ARN do conjunto de dados que você deseja listar.
- show_errors_only: especifique `true` se deseja ver somente erros. `false`, caso contrário.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to list the entries in an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def list_dataset_entries(rek_client, dataset_arn, show_errors):
    """
    Lists the entries in an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataet that you want to use.
    """

    try:
        # List the entries.
        logger.info("Listing dataset entries for the dataset %s.", dataset_arn)

        finished = False
        count = 0
        next_token = ""
        show_errors_only = False
```

```
        if show_errors.lower() == "true":
            show_errors_only = True

        while finished is False:

            response = rek_client.list_dataset_entries(
                DatasetArn=dataset_arn,
                HasErrors=show_errors_only,
                MaxResults=100,
                NextToken=next_token)

            count += len(response['DatasetEntries'])

            for entry in response['DatasetEntries']:
                print(entry)

            if 'NextToken' not in response:
                finished = True
                logger.info("No more entries. Total:%s", count)
            else:
                next_token = next_token = response['NextToken']
                logger.info("Getting more entries. Total so far :%s", count)

    except ClientError as err:
        logger.exception(
            "Couldn't list dataset: %s",
             err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to list."

    )

    parser.add_argument(
        "show_errors_only", help="true if you want to see errors only. false
 otherwise."
```

```
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Listing entries for  dataset {args.dataset_arn}")

        # List the dataset entries.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_dataset_entries(rekognition_client,
                             args.dataset_arn,
                             args.show_errors_only)

        print(f"Finished listing entries for dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- dataset_arn: o ARN do conjunto de dados que você deseja listar.
- show_errors_only: especifique `true` se deseja ver somente erros. `false`, caso contrário.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.ListDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.paginators.ListDatasetEntriesIterable;


import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ListDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(ListDatasetEntries.class.getName());

    public static void listMyDatasetEntries(RekognitionClient rekClient, String
 datasetArn, boolean showErrorsOnly)
            throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Listing dataset {0}", new Object[]
 { datasetArn });

            ListDatasetEntriesRequest listDatasetEntriesRequest =
 ListDatasetEntriesRequest.builder()
```

```
.hasErrors(showErrorsOnly).datasetArn(datasetArn).maxResults(1).build();

           ListDatasetEntriesIterable datasetEntriesList = rekClient
                   .listDatasetEntriesPaginator(listDatasetEntriesRequest);

           datasetEntriesList.stream().flatMap(r ->
r.datasetEntries().stream())
                   .forEach(datasetEntry ->
System.out.println(datasetEntry.toString()));

       } catch (RekognitionException e) {
           logger.log(Level.SEVERE, "Could not update dataset: {0}",
e.getMessage());
           throw e;
       }

   }

   public static void main(String args[]) {

       boolean showErrorsOnly = false;
       String datasetArn = null;

       final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
<updates_file>\n\n" + "Where:\n"
               + "   dataset_arn - the ARN of the dataset that you want to
update.\n\n"
               + "   show_errors_only - true to show only errors. false
otherwise.\n\n";

       if (args.length != 2) {
           System.out.println(USAGE);
           System.exit(1);
       }

       datasetArn = args[0];
       if (args[1].toLowerCase().equals("true")) {

           showErrorsOnly = true;
       }

       try {
```

```
            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // list the dataset entries.

            listMyDatasetEntries(rekClient, datasetArn, showErrorsOnly);

            System.out.println(String.format("Finished listing entries for :
 %s", datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Como distribuir um conjunto de dados de treinamento (SDK)

O Amazon Rekognition Custom Labels exige um conjunto de dados de treinamento e um conjunto de dados de teste para treinar o seu modelo.

Se você estiver usando a API, poderá usá-la para distribuir 20% do conjunto de dados de treinamento em um conjunto de dados de teste vazio. DistributeDatasetEntries Distribuir o conjunto de dados de treinamento pode ser útil se houver apenas um único arquivo de manifesto disponível. Use o arquivo de manifesto único para criar seu conjunto de dados de treinamento. Em seguida, crie um conjunto de dados de teste vazio e use `DistributeDatasetEntries` para preencher o conjunto de dados de teste.

> **Note**
>
> Se estiver usando o console do Amazon Rekognition Custom Labels e começar com um único projeto de conjunto de dados, o Amazon Rekognition Custom Labels divide (distribui) o conjunto de dados de treinamento, durante o treinamento, para criar um conjunto de dados de teste. 20% das entradas do conjunto de dados de treinamento são movidas para o conjunto de dados de teste.

Para distribuir um conjunto de dados de treinamento (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Crie um projeto. Para obter mais informações, consulte Como criar um projeto do Amazon Rekognition Custom Labels (SDK).
3. Crie seu conjunto de dados de treinamento. Para obter informações sobre conjuntos de dados, consulte Como criar conjuntos de dados de treinamento e teste.
4. Criar um conjunto de dados de teste vazio.
5. Use o código de exemplo a seguir para distribuir 20% das entradas do conjunto de dados de treinamento no conjunto de dados de teste. Você pode obter os Amazon Resource Names (ARN) para os conjuntos de dados de um projeto ligando para. DescribeProjects Para obter um código de exemplo, consulte Como descrever um projeto (SDK).

   AWS CLI

   Altere o valor de `training_dataset-arn` e `test_dataset_arn` com o ARNS dos conjuntos de dados que você deseja usar.

   ```
   aws rekognition distribute-dataset-entries --datasets ['{"Arn":
    "training_dataset_arn"}, {"Arn": "test_dataset_arn"}'] \
     --profile custom-labels-access
   ```

   Python

   Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

   - training_dataset_arn: o ARN do conjunto de dados de teste do qual as entradas são distribuídas.

- test_dataset_arn: o ARN do conjunto de dados de teste para o qual as entradas são distribuídas.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def check_dataset_status(rek_client, dataset_arn):
    """
    Checks the current status of a dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The dataset that you want to check.
    :return: The dataset status and status message.
    """
    finished = False
    status = ""
    status_message = ""

    while finished is False:

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        status = dataset['DatasetDescription']['Status']
        status_message = dataset['DatasetDescription']['StatusMessage']

        if status == "UPDATE_IN_PROGRESS":

            logger.info("Distributing dataset: %s ", dataset_arn)
            time.sleep(5)
            continue

        if status == "UPDATE_COMPLETE":
```

```
            logger.info(
                "Dataset distribution complete: %s : %s : %s",
                    status, status_message, dataset_arn)
            finished = True
            continue

        if status == "UPDATE_FAILED":
            logger.exception(
                "Dataset distribution failed: %s : %s : %s",
                    status, status_message, dataset_arn)
            finished = True
            break

        logger.exception(
            "Failed. Unexpected state for dataset distribution: %s : %s : %s",
            status, status_message, dataset_arn)
        finished = True
        status_message = "An unexpected error occurred while distributing the
 dataset"
        break

    return status, status_message


def distribute_dataset_entries(rek_client, training_dataset_arn,
 test_dataset_arn):
    """
    Distributes 20% of the supplied training dataset into the supplied test
 dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param training_dataset_arn: The ARN of the training dataset that you
 distribute entries from.
    :param test_dataset_arn: The ARN of the test dataset that you distribute
 entries to.
    """

    try:
        # List dataset labels.
        logger.info("Distributing training dataset entries (%s) into test
 dataset (%s).",
            training_dataset_arn,test_dataset_arn)


        datasets = json.loads(
```

```
            '[{"Arn" : "' + str(training_dataset_arn) + '"},{"Arn" : "' +
 str(test_dataset_arn) + '"}]')

        rek_client.distribute_dataset_entries(
            Datasets=datasets
        )

        training_dataset_status, training_dataset_status_message =
 check_dataset_status(
            rek_client, training_dataset_arn)
        test_dataset_status, test_dataset_status_message = check_dataset_status(
            rek_client, test_dataset_arn)

        if training_dataset_status == 'UPDATE_COMPLETE' and test_dataset_status
 == "UPDATE_COMPLETE":
            print("Distribution complete")
        else:
            print("Distribution failed:")
            print(
                f"\ttraining dataset: {training_dataset_status} :
 {training_dataset_status_message}")
            print(
                f"\ttest dataset: {test_dataset_status} :
 {test_dataset_status_message}")

    except ClientError as err:
        logger.exception(
            "Couldn't distribute dataset: %s",err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "training_dataset_arn", help="The ARN of the training dataset that you
 want to distribute from."
    )

    parser.add_argument(
```

```
        "test_dataset_arn", help="The ARN of the test dataset that you want to
 distribute to."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Distributing training dataset entries
 ({args.training_dataset_arn}) "\
            f"into test dataset ({args.test_dataset_arn}).")

        # Distribute the datasets.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        distribute_dataset_entries(rekognition_client,
                                   args.training_dataset_arn,
                                   args.test_dataset_arn)

        print("Finished distributing datasets.")

    except ClientError as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem listing dataset labels: {err}")
    except Exception as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem distributing datasets: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- training_dataset_arn: o ARN do conjunto de dados de teste do qual as entradas são distribuídas.
- test_dataset_arn: o ARN do conjunto de dados de teste para o qual as entradas são distribuídas.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DistributeDataset;
import
 software.amazon.awssdk.services.rekognition.model.DistributeDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DistributeDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(DistributeDatasetEntries.class.getName());

    public static DatasetStatus checkDatasetStatus(RekognitionClient rekClient,
 String datasetArn)
            throws Exception, RekognitionException {
```

```
        boolean distributed = false;
        DatasetStatus status = null;

        // Wait until distribution completes

        do {

            DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();
            DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

            DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

            status = datasetDescription.status();

            logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

            switch (status) {

            case UPDATE_COMPLETE:
                logger.log(Level.INFO, "Dataset updated");
                distributed = true;
                break;

            case UPDATE_IN_PROGRESS:
                Thread.sleep(5000);
                break;

            case UPDATE_FAILED:
                String error = "Dataset distribution failed: " +
datasetDescription.statusAsString() + " "
                        + datasetDescription.statusMessage() + " " + datasetArn;
                logger.log(Level.SEVERE, error);
                break;

            default:
                String unexpectedError = "Unexpected distribution state: " +
datasetDescription.statusAsString() + " "
                        + datasetDescription.statusMessage() + " " + datasetArn;
                logger.log(Level.SEVERE, unexpectedError);
```

```
            }

        } while (distributed == false);

        return status;

    }

    public static void distributeMyDatasetEntries(RekognitionClient rekClient,
String trainingDatasetArn,
            String testDatasetArn) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Distributing {0} dataset to {1} ",
                    new Object[] { trainingDatasetArn, testDatasetArn });

            DistributeDataset distributeTrainingDataset =
DistributeDataset.builder().arn(trainingDatasetArn).build();

            DistributeDataset distributeTestDataset =
DistributeDataset.builder().arn(testDatasetArn).build();

            ArrayList<DistributeDataset> datasets = new ArrayList();

            datasets.add(distributeTrainingDataset);
            datasets.add(distributeTestDataset);

            DistributeDatasetEntriesRequest distributeDatasetEntriesRequest =
DistributeDatasetEntriesRequest.builder()
                    .datasets(datasets).build();

            rekClient.distributeDatasetEntries(distributeDatasetEntriesRequest);

            DatasetStatus trainingStatus = checkDatasetStatus(rekClient,
trainingDatasetArn);
            DatasetStatus testStatus = checkDatasetStatus(rekClient,
testDatasetArn);

            if (trainingStatus == DatasetStatus.UPDATE_COMPLETE && testStatus ==
DatasetStatus.UPDATE_COMPLETE) {
                logger.log(Level.INFO, "Successfully distributed dataset: {0}",
trainingDatasetArn);
```

```
            } else {

                throw new Exception("Failed to distribute dataset: " +
 trainingDatasetArn);
            }

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not distribute dataset: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String trainingDatasetArn = null;
        String testDatasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<training_dataset_arn>
 <test_dataset_arn>\n\n" + "Where:\n"
                + "   training_dataset_arn - the ARN of the dataset that you
 want to distribute from.\n\n"
                + "   test_dataset_arn - the ARN of the dataset that you want to
 distribute to.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        trainingDatasetArn = args[0];
        testDatasetArn = args[1];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Distribute the dataset
```

```
            distributeMyDatasetEntries(rekClient, trainingDatasetArn,
 testDatasetArn);

            System.out.println("Datasets distributed.");

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Como excluir um conjunto de dados

É possível excluir os conjuntos de dados de treinamento e teste de um projeto.

Tópicos

- Como excluir um conjunto de dados (console)
- Como excluir um conjunto de dados do Amazon Rekognition Custom Labels (SDK)

Como excluir um conjunto de dados (console)

Use o procedimento a seguir para excluir um conjunto de dados. Depois, se o projeto tiver um conjunto de dados restante (treinamento ou teste), a página de detalhes do projeto será exibida. Se o projeto não tiver conjuntos de dados restantes, a página Criar conjunto de dados será exibida.

Se excluir o conjunto de dados de treinamento, deverá criar um novo conjunto de dados de treinamento para o projeto antes de treinar um modelo. Para obter mais informações, consulte Como criar conjuntos de dados de treinamento e teste com imagens.

Se excluir o conjunto de dados de teste, poderá treinar um modelo sem criar um novo conjunto de dados de teste. Durante o treinamento, o conjunto de dados de treinamento é dividido para criar

um novo conjunto de dados de teste para o projeto. A divisão do conjunto de dados de treinamento reduz o número de imagens disponíveis para treinamento. Para manter a qualidade, recomendamos criar um novo conjunto de dados de teste antes de treinar um modelo. Para obter mais informações, consulte Como adicionar um conjunto de dados a um projeto.

Para excluir um conjunto de dados

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Usar rótulos personalizados. A página inicial do Amazon Rekognition Custom Labels é exibida.
3. No painel de navegação esquerdo, selecione Projetos. A visualização Projetos é exibida.
4. Selecione o tópico que contém o conjunto de dados que você deseja excluir.
5. No painel de navegação esquerdo, abaixo do nome do projeto, escolha Conjunto de dados
6. Escolha Ações.
7. Para excluir o conjunto de dados de treinamento, escolha Excluir conjunto de dados de treinamento.
8. Para excluir o conjunto de dados de teste, escolha Excluir conjunto de dados.
9. Na caixa de diálogo Excluir conjunto de dados de treinamento ou teste, insira excluir para confirmar que você deseja excluir o conjunto de dados.
10. Escolha Excluir conjunto de dados de treinamento ou teste para excluir o conjunto de dados.

Como excluir um conjunto de dados do Amazon Rekognition Custom Labels (SDK)

Você exclui um conjunto de dados Amazon Rekognition Custom Labels DeleteDatasetchamando e fornecendo o Amazon Resource Name (ARN) do conjunto de dados que você deseja excluir. Para obter os conjuntos ARNs de dados de treinamento e teste em um projeto, ligue DescribeProjects. A resposta inclui uma matriz de ProjectDescriptionobjetos. O conjunto de dados ARNs (`DatasetArn`) e os tipos de conjunto de dados (`DatasetType`) estão na `Datasets` lista.

Se excluir o conjunto de dados de treinamento, precisará criar um novo conjunto de dados de treinamento para o projeto antes de treinar um modelo. Se excluir o conjunto de dados de teste, precisará criar um novo conjunto de dados de teste antes de treinar o modelo. Para obter mais informações, consulte Como adicionar um conjunto de dados a um projeto (SDK).

Para excluir um conjunto de dados (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para excluir um conjunto de dados.

   AWS CLI

   Altere o valor de `dataset-arn` para o ARN dos conjuntos de dados que você deseja excluir.

```
aws rekognition delete-dataset --dataset-arn dataset-arn \
  --profile custom-labels-access
```

   Python

   Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

   - dataset_arn: o ARN do conjunto de dados que você deseja excluir.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to delete an Amazon Rekognition Custom Labels dataset.
"""
import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def delete_dataset(rek_client, dataset_arn):
    """
    Deletes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to delete.
```

```
    """

    try:
        # Delete the dataset,
        logger.info("Deleting dataset: %s", dataset_arn)

        rek_client.delete_dataset(DatasetArn=dataset_arn)

        deleted = False

        logger.info("waiting for dataset deletion %s", dataset_arn)

        # Dataset might not be deleted yet, so wait.
        while deleted is False:
            try:
                rek_client.describe_dataset(DatasetArn=dataset_arn)
                time.sleep(5)
            except ClientError as err:
                if err.response['Error']['Code'] == 'ResourceNotFoundException':
                    logger.info("dataset deleted: %s", dataset_arn)
                    deleted = True
                else:
                    raise

        logger.info("dataset deleted: %s", dataset_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete dataset - %s: %s",
                         dataset_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to delete."
    )
```

```
def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Deleting dataset: {args.dataset_arn}")

        # Delete the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        delete_dataset(rekognition_client,
                       args.dataset_arn)

        print(f"Finished deleting dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem deleting dataset: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- dataset_arn: o ARN do conjunto de dados que você deseja excluir.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
```

```
*/
package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteDataset {

    public static final Logger logger =
 Logger.getLogger(DeleteDataset.class.getName());

    public static void deleteMyDataset(RekognitionClient rekClient, String
 datasetArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting dataset: {0}", datasetArn);

            // Delete the dataset

            DeleteDatasetRequest deleteDatasetRequest =
 DeleteDatasetRequest.builder().datasetArn(datasetArn).build();

            DeleteDatasetResponse response =
 rekClient.deleteDataset(deleteDatasetRequest);

            // Wait until deletion finishes

            DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();

            Boolean deleted = false;

            do {
```

```
                try {

                    rekClient.describeDataset(describeDatasetRequest);
                    Thread.sleep(5000);
                } catch (RekognitionException e) {
                    String errorCode = e.awsErrorDetails().errorCode();
                    if (errorCode.equals("ResourceNotFoundException")) {
                        logger.log(Level.INFO, "Dataset deleted: {0}",
datasetArn);
                        deleted = true;
                    } else {
                        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
                        throw e;
                    }

                }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Dataset deleted: {0} ", datasetArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
                + "   dataset_arn - The ARN of the dataset that you want to
delete.\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String datasetArn = args[0];
```

```
        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();


            // Delete the dataset
            deleteMyDataset(rekClient, datasetArn);

            System.out.println(String.format("Dataset deleted: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Como gerenciar um modelo do Amazon Rekognition Custom Labels

O modelo Amazon Rekognition Custom Labels é um modelo matemático que prevê a presença de objetos, cenas e conceitos em novas imagens. Ele faz isso descobrindo padrões nas imagens usadas para treinar o modelo. Esta seção mostra como treinar um modelo, avaliar seu desempenho

e fazer melhorias. Também mostra como disponibilizar um modelo para uso e como excluir um modelo quando você não precisar mais dele.

Tópicos

- Como excluir um modelo do Amazon Rekognition Custom Labels
- Como marcar um modelo
- Como descrever um modelo (SDK)
- Como copiar um modelo do Amazon Rekognition Custom Labels (SDK)

Como excluir um modelo do Amazon Rekognition Custom Labels

Você pode excluir um modelo usando o console Amazon Rekognition Custom Labels ou usando a API. DeleteProjectVersion Não é possível excluir um modelo se ele estiver em execução ou em treinamento. Para interromper a execução de um modelo, use a StopProjectVersionAPI. Para obter mais informações, consulte Como interromper um modelo do Amazon Rekognition Custom Labels (SDK). Se um modelo estiver sendo treinado, espere até que ele termine antes de excluí-lo.

Um modelo excluído não pode ser recuperado.

Tópicos

- Como excluir um modelo do Amazon Rekognition Custom Labels (console)
- Como excluir um modelo do Amazon Rekognition Custom Labels (SDK)

Como excluir um modelo do Amazon Rekognition Custom Labels (console)

O procedimento a seguir mostra como excluir um modelo de uma página de detalhes do projeto. Também é possível excluir um modelo da página de detalhes de um modelo.

Para excluir um canal (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Usar rótulos personalizados.
3. Escolha Comece a usar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Selecione o tópico que contém o modelo que você deseja excluir. A página de detalhes do projeto é aberta.

6. Na seção Modelos, escolha os modelos que deseja excluir.

> **Note**
>
> Se o modelo não puder ser selecionado, ele está em execução ou em treinamento e não pode ser excluído. Verifique o campo Status e tente novamente após interromper o modelo em execução ou espere até que o treinamento termine.

7. Escolha Excluir modelo e a caixa de diálogo Excluir modelo será exibida.
8. Insira excluir para confirmar a exclusão.
9. Escolha Excluir para excluir o modelo. A exclusão do modelo pode demorar um pouco para ser concluída.

> **Note**
>
> Se Fechar a caixa de diálogo durante a exclusão do modelo, os modelos ainda serão excluídos.

Como excluir um modelo do Amazon Rekognition Custom Labels (SDK)

Você exclui um modelo de etiquetas personalizadas do Amazon Rekognition DeleteProjectVersionchamando e fornecendo o Amazon Resource Name (ARN) do modelo que você deseja excluir. É possível obter o ARN do modelo na seção Use seu modelo da página de detalhes do modelo no console do Amazon Rekognition Custom Labels. Como alternativa, ligue DescribeProjectVersionsse forneça o seguinte.

- O ARN do projeto (`ProjectArn`) ao qual o trabalho está associado.
- O nome da versão (`VersionNames`) do modelo.

O ARN do modelo é o `ProjectVersionArn` campo no ProjectVersionDescriptionobjeto, a partir da `DescribeProjectVersions` resposta.

Não é possível excluir um modelo se ele estiver em execução ou em treinamento. Para determinar se o modelo está em execução ou em treinamento, ligue DescribeProjectVersionsse verifique o `Status` campo do ProjectVersionDescriptionobjeto do modelo. Para interromper a execução de um modelo, use a StopProjectVersionAPI. Para obter mais informações, consulte Como interromper um

modelo do Amazon Rekognition Custom Labels (SDK). É preciso esperar que um modelo termine o treinamento antes de excluí-lo.

Para excluir um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para excluir um modelo.

   AWS CLI

   Altere o valor de `project-version-arn` para o nome do projeto que você deseja excluir.

```
aws rekognition delete-project-version --project-version-arn model_arn \
  --profile custom-labels-access
```

   Python

   Forneça os seguintes parâmetros de linha de comando

   - `project_arn`: o ARN do projeto que contém o modelo que você deseja excluir.
   - `model_arn`: o ARN da versão do modelo que você deseja excluir.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to delete an existing Amazon Rekognition Custom Labels model.
"""


import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_model(rek_client, project_arn, model_arn):
    """
    Deletes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param model_arn: The ARN of the model version that you want to delete.
    """

    try:
        # Delete the model
        logger.info("Deleting dataset: {%s}", model_arn)

        rek_client.delete_project_version(ProjectVersionArn=model_arn)

        # Get the model version name
        start = find_forward_slash(model_arn, 3) + 1
        end = find_forward_slash(model_arn, 4)
        version_name = model_arn[start:end]

        deleted = False

        # model might not be deleted yet, so wait deletion finishes.
        while deleted is False:
            describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
            if len(describe_response['ProjectVersionDescriptions']) == 0:
                deleted = True
            else:
                logger.info("Waiting for model deletion %s", model_arn)
                time.sleep(5)
```

```
        logger.info("model deleted: %s", model_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete model - %s: %s",
                         model_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to delete."
    )

    parser.add_argument(
        "model_arn", help="The ARN of the model version that you want to
 delete."
    )


def confirm_model_deletion(model_arn):
    """
    Confirms deletion of the model. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(f"Are you sure you wany to delete model {model_arn} ?\n", model_arn)

    start = input("Enter delete to delete your model: ")
    if start == "delete":
        return True
    else:
        return False


def main():
```

```
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        if confirm_model_deletion(args.model_arn) is True:
            print(f"Deleting model: {args.model_arn}")

            # Delete the model.
            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            delete_model(rekognition_client,
                         args.project_arn,
                         args.model_arn)

            print(f"Finished deleting model: {args.model_arn}")
        else:
            print(f"Not deleting model {args.model_arn}")

    except ClientError as err:
        print(f"Problem deleting model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

- `project_arn`: o ARN do projeto que contém o modelo que você deseja excluir.
- `model_arn`: o ARN da versão do modelo que você deseja excluir.

```
//Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)
```

```
import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.services.rekognition.RekognitionClient;

import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteModel {

    public static final Logger logger =
 Logger.getLogger(DeleteModel.class.getName());

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void deleteMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn)
            throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting model: {0}", projectArn);

            // Delete the model

            DeleteProjectVersionRequest deleteProjectVersionRequest =
 DeleteProjectVersionRequest.builder()
```

```
                    .projectVersionArn(modelArn).build();

            DeleteProjectVersionResponse response =
                    rekClient.deleteProjectVersion(deleteProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.status());

            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);

            Boolean deleted = false;

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .projectArn(projectArn).versionNames(versionName).build();

            // Wait until model is deleted.

            do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                if
(describeProjectVersionsResponse.projectVersionDescriptions().size()==0) {
                    logger.log(Level.INFO, "Waiting for model deletion: {0}",
modelArn);
                    Thread.sleep(5000);
                } else {
                    deleted = true;
                    logger.log(Level.INFO, "Model deleted: {0}", modelArn);
                }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Model deleted: {0}", modelArn);

        } catch (
```

```
        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn> <model_arn>\n\n"
+ "Where:\n"
                + "   project_arn - The ARN of the project that contains the
model that you want to delete.\n\n"
                + "   model_version - The ARN of the model that you want to
delete.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String modelVersion = args[1];

        try {

            RekognitionClient rekClient = RekognitionClient.builder().build();

            // Delete the model
            deleteMyModel(rekClient, projectArn, modelVersion);

            System.out.println(String.format("model deleted: %s",
modelVersion));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
            System.exit(1);
        }
```

```
        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Como marcar um modelo

É possível identificar, organizar, pesquisar e filtrar seus modelos do Amazon Rekognition usando tags. Cada tag é um rótulo que consiste em um valor e uma chave definida pelo usuário. Por exemplo, para ajudar a determinar o faturamento de seus modelos, marque seus modelos com uma chave `Cost center` e adicione o número do centro de custo apropriado como valor. Para obter mais informações, consulte Como rotular recursos da AWS.

Use tags para:

- Acompanhe o faturamento de um modelo usando tags de alocação de custos. Para obter mais informações, consulte Usar tags de alocação de custos.
- Controle o acesso a um modelo usando o AWS Identity and Access Management (IAM). Para obter mais informações, consulte Controlar o acesso a recursos da AWS usando tags.
- Automatize o gerenciamento de modelos. Por exemplo, é possível executar scripts de inicialização ou interrupção automatizados que desativam modelos de desenvolvimento fora do horário comercial para reduzir os custos. Para obter mais informações, consulte Como executar um modelo do Amazon Rekognition Custom Labels.

Você pode marcar modelos usando o console do Amazon Rekognition ou usando o. AWS SDKs

Tópicos

- Como marcar modelos (console)
- Como visualizar tags de modelo
- Como marcar modelos (SDK)

Como marcar modelos (console)

É possível usar o console do Rekognition para adicionar tags aos modelos, visualizar as tags anexadas a um modelo e remover tags.

Adicionar e remover rótulos

Este procedimento explica como adicionar ou remover tags de um modelo existente. Também é possível adicionar tags a um modelo novo quando ele for treinado. Para obter mais informações, consulte Como treinar um modelo do Amazon Rekognition Custom Labels.

Para adicionar tags ou remover tags de um modelo existente usando o console

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Começar.
3. No painel de navegação, selecione Projetos.
4. Na página de recursos Projetos, selecione o projeto que contém o modelo ao qual você deseja atribuir uma tag.
5. No painel de navegação, no projeto que você escolheu anteriormente, escolha Modelos.
6. Na seção Modelos, escolha o modelo ao qual deseja atribuir uma tag.
7. Na página de detalhes do modelo, escolha a guia Tags.
8. Na seção Tags, escolha Gerenciar tags.
9. Na página Gerenciar tags, escolha Adicionar nova tag.
10. Insira uma Chave e um Valor.
    a. Em Chave, insira o nome da chave.
    b. Em Valor, insira um valor.
11. Para adicionar mais tags, repita as etapas 9 e 10.
12. (Opcional) Para remover uma tag, selecione Remover ao lado da tag que você deseja remover. Se estiver removendo uma tag salva anteriormente, ela será removida quando você salvar suas alterações.
13. Escolha Salvar alterações para salvar suas alterações.

Como visualizar tags de modelo

É possível usar o console do Amazon Rekognition para visualizar as tags anexadas a um modelo.

Para visualizar as tags anexadas a todos os modelos em um projeto, você deve usar o AWS SDK. Para obter mais informações, consulte Como listar de tags de modelo.

Para visualizar as tags anexadas a um modelo

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Começar.
3. No painel de navegação, selecione Projetos.
4. Na página de recursos Projetos, selecione o projeto que contém o modelo cuja tag você deseja visualizar.
5. No painel de navegação, no projeto que você escolheu anteriormente, escolha Modelos.
6. Na seção Modelos, escolha o modelo cujo rótulo você deseja exibir.
7. Na página de detalhes do modelo, escolha a guia Tags. As tags são mostradas na seção Tags.

Como marcar modelos (SDK)

Você pode usar o AWS SDK para:

- Adicionar tags a um novo modelo
- Adicionar tags a um modelo existente
- Liste as tags anexadas a um modelo
- Remover tags de um modelo

As tags nos AWS CLI exemplos a seguir estão no formato a seguir.

```
--tags '{"key1":"value1","key2":"value2"}'
```

Como alternativa, é possível usar este formato.

```
--tags key1=value1,key2=value2
```

Se você não instalou o AWS CLI, consulteEtapa 4: configurar o AWS CLI e AWS SDKs.

Como adicionar tags a um novo modelo

Você pode adicionar tags a um modelo ao criá-lo usando a CreateProjectVersionoperação. Especifique uma ou mais tags no parâmetro `Tags` de entrada da matriz.

```
aws rekognition create-project-version --project-arn project arn \
  --version-name version_name \
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":  "output
 folder" } }' \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

Para obter informações sobre como criar e treinar um modelo, consulte Treinando um modelo (SDK).

Como adicionar tags a um modelo existente

Para adicionar uma ou mais tags a um modelo existente, use a TagResourceoperação. Especifique o nome do recurso da Amazon (ARN) (`ResourceArn`) do modelo e as tags (`Tags`) que deseja adicionar. O exemplo a seguir mostra como adicionar duas tags.

```
aws rekognition tag-resource --resource-arn resource-arn \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

Você pode obter o ARN de um modelo ligando para. CreateProjectVersion

Como listar de tags de modelo

Para listar as tags anexadas a um modelo, use a ListTagsForResourceoperação e especifique o ARN do modelo ()`ResourceArn`. A resposta é um mapa de tags e valores anexados ao modelo especificado.

```
aws rekognition list-tags-for-resource --resource-arn resource-arn \
  --profile custom-labels-access
```

A saída exibe uma lista das tags anexadas ao modelo.

```
{
    "Tags": {
        "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
    }
}
```

Para ver quais modelos em um projeto têm uma tag específica, chame `DescribeProjectVersions` para obter uma lista de modelos. Depois, chame `ListTagsForResource` para cada modelo na resposta de `DescribeProjectVersions`. Inspecione a resposta de `ListTagsForResource` para ver se a tag necessária está presente.

O exemplo a seguir do Python 3 mostra como pesquisar em todos os seus projetos uma chave e um valor de tag específicos. A saída inclui o ARN do projeto e o ARN do modelo em que uma chave correspondente é encontrada.

Para pesquisar um valor de tag

1. Salve o código a seguir em um arquivo chamado `find_tag.py`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to find a tag value that's associated with models within
your Amazon Rekognition Custom Labels projects.
"""
import logging
import argparse
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def find_tag_in_projects(rekognition_client, key, value):
    """
    Finds Amazon Rekognition Custom Label models tagged with the supplied key and
 key value.
    :param rekognition_client: An Amazon Rekognition boto3 client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    return: A list of matching model versions (and model projects) that were found.
    """
    try:

        found_tags = []
        found = False
```

```
        projects = rekognition_client.describe_projects()
        # Iterate through each project and models within a project.
        for project in projects["ProjectDescriptions"]:
            logger.info("Searching project: %s ...", project["ProjectArn"])

            models = rekognition_client.describe_project_versions(
                ProjectArn=(project["ProjectArn"])
            )

            for model in models["ProjectVersionDescriptions"]:
                logger.info("Searching model %s", model["ProjectVersionArn"])

                tags = rekognition_client.list_tags_for_resource(
                    ResourceArn=model["ProjectVersionArn"]
                )

                logger.info(
                    "\tSearching model: %s for tag: %s value: %s.",
                    model["ProjectVersionArn"],
                    key,
                    value,
                )
                # Check if tag exists.

                if key in tags["Tags"]:
                    if tags["Tags"][key] == value:
                        found = True
                        logger.info(
                            "\t\tMATCH: Project: %s: model version %s",
                            project["ProjectArn"],
                            model["ProjectVersionArn"],
                        )
                        found_tags.append(
                            {
                                "Project": project["ProjectArn"],
                                "ModelVersion": model["ProjectVersionArn"],
                            }
                        )

        if found is False:
            logger.info("No match for Tag %s with value %s.", key, value)
        return found_tags
    except ClientError as err:
```

```
        logger.info("Problem finding tags: %s. ", format(err))
        raise


def main():
    """
    Entry point for example.
    """
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    # Set up command line arguments.
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")

    args = parser.parse_args()
    key = args.tag
    value = args.value

    print(f"Searching your models for tag: {key} with value: {value}.")


    session = boto3.Session(profile_name='custom-labels-access')
    rekognition_client = session.client("rekognition")

    # Get tagged models for all projects.
    tagged_models = find_tag_in_projects(rekognition_client, key, value)

    print("Matched models\n--------------")
    if len(tagged_models) > 0:
        for model in tagged_models:
            print(
                "Project: {project}\nModel version: {version}\n".format(
                    project=model["Project"], version=model["ModelVersion"]
                )
            )

    else:
        print("No matches found.")

    print("Done.")
```

```
if __name__ == "__main__":
    main()
```

2. No prompt de comando, digite o seguinte. *value*Substitua *key* e pelo nome e valor da chave que você deseja encontrar.

```
python find_tag.py key value
```

Como excluir as tags de um modelo

Para remover uma ou mais tags de um modelo, use a UntagResourceoperação. Especifique o ARN do modelo (`ResourceArn`) e as chaves de tag (`Tag-Keys`) que deseja remover.

```
aws rekognition untag-resource --resource-arn resource-arn \
  --tag-keys '["key1","key2"]' \
  --profile custom-labels-access
```

Como alternativa, é possível especificar `tag-keys` neste formato.

```
--tag-keys key1,key2
```

Como descrever um modelo (SDK)

É possível usar a API `DescribeProjectVersions` para obter informações sobre uma versão de um modelo. Se não especificar `VersionName`, `DescribeProjectVersions` retornará as descrições de todas as versões do modelo no projeto.

Para descrever um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código de exemplo a seguir para descrever uma versão de um modelo.

   AWS CLI

   Altere o valor de `project-arn` para o ARN do projeto que você deseja descrever. Altere o valor de `version-name` para a versão do modelo que você deseja descrever.

```
aws rekognition describe-project-versions --project-arn project_arn \
  --version-names version_name \
  --profile custom-labels-access
```

Python

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- project_arn: o ARN do projeto que deseja descrever.
- model_version: a versão do modelo que deseja descrever.

Por exemplo: `python describe_model.py` *`project_arn model_version`*

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to describe an Amazon Rekognition Custom Labels model.
"""
import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def describe_model(rek_client, project_arn, version_name):
    """
    Describes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the prject that contains the model.
    :param version_name: The version name of the model that you want to
 describe.
    """

    try:
        # Describe the model
        logger.info("Describing model: %s for project %s",
```

```
                    version_name, project_arn)

        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print(f"Created: {str(model['CreationTimestamp'])} ")
            print(f"ARN: {str(model['ProjectVersionArn'])} ")
            if 'BillableTrainingTimeInSeconds' in model:
                print(
                    f"Billing training time (minutes):
 {str(model['BillableTrainingTimeInSeconds']/60)} ")
            print("Evaluation results: ")
            if 'EvaluationResult' in model:
                evaluation_results = model['EvaluationResult']
                print(f"\tF1 score: {str(evaluation_results['F1Score'])}")
                print(
                    f"\tSummary location: s3://{evaluation_results['Summary']
['S3Object']['Bucket']}/{evaluation_results['Summary']['S3Object']['Name']}")

            if 'ManifestSummary' in model:
                print(
                    f"Manifest summary location: s3://{model['ManifestSummary']
['S3Object']['Bucket']}/{model['ManifestSummary']['S3Object']['Name']}")
            if 'OutputConfig' in model:
                print(
                    f"Training output location: s3://{model['OutputConfig']
['S3Bucket']}/{model['OutputConfig']['S3KeyPrefix']}")
            if 'MinInferenceUnits' in model:
                print(
                    f"Minimum inference units:
 {str(model['MinInferenceUnits'])}")
            if 'MaxInferenceUnits' in model:
                print(
                    f"Maximum Inference units:
 {str(model['MaxInferenceUnits'])}")

            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])

    except ClientError as err:
        logger.exception(
            "Couldn't describe model: %s", err.response['Error']['Message'])
```

```
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to
 describe."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Describing model: {args.version_name} for project
 {args.project_arn}.")

        # Describe the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_model(rekognition_client, args.project_arn,
                       args.version_name)

        print(
            f"Finished describing model: {args.version_name} for project
 {args.project_arn}.")
```

```
    except ClientError as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- project_arn: o ARN do projeto que deseja descrever.
- model_version: a versão do modelo que deseja descrever.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.EvaluationResult;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;

public class DescribeModel {

    public static final Logger logger =
 Logger.getLogger(DescribeModel.class.getName());

    public static void describeMyModel(RekognitionClient rekClient, String
 projectArn, String versionName) {

        try {

            // If a single version name is supplied, build request argument

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
 null;

            if (versionName == null) {
                describeProjectVersionsRequest =
 DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                        .build();
            } else {
                describeProjectVersionsRequest =
 DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                        .versionNames(versionName).build();
            }

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
 rekClient
                    .describeProjectVersions(describeProjectVersionsRequest);

            for (ProjectVersionDescription projectVersionDescription :
 describeProjectVersionsResponse
                    .projectVersionDescriptions()) {

                System.out.println("ARN: " +
 projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
 projectVersionDescription.statusAsString());
                System.out.println("Message: " +
 projectVersionDescription.statusMessage());
```

```
                if (projectVersionDescription.billableTrainingTimeInSeconds() !=
null) {
                    System.out.println(
                            "Billable minutes: " +
(projectVersionDescription.billableTrainingTimeInSeconds() / 60));
                }

                if (projectVersionDescription.evaluationResult() != null) {
                    EvaluationResult evaluationResult =
projectVersionDescription.evaluationResult();

                    System.out.println("F1 Score: " +
evaluationResult.f1Score());
                    System.out.println("Summary location: s3://" +
evaluationResult.summary().s3Object().bucket() + "/"
                            + evaluationResult.summary().s3Object().name());
                }

                if (projectVersionDescription.manifestSummary() != null) {
                    GroundTruthManifest manifestSummary =
projectVersionDescription.manifestSummary();
                    System.out.println("Manifest summary location: s3://" +
manifestSummary.s3Object().bucket() + "/"
                            + manifestSummary.s3Object().name());

                }

                if (projectVersionDescription.outputConfig() != null) {
                    OutputConfig outputConfig =
projectVersionDescription.outputConfig();
                    System.out.println(
                            "Training output: s3://" + outputConfig.s3Bucket() +
"/" + outputConfig.s3KeyPrefix());
                }

                if (projectVersionDescription.minInferenceUnits() != null) {
                    System.out.println("Min inference units: " +
projectVersionDescription.minInferenceUnits());
                }

                System.out.println();

            }
```

```
        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            throw rekError;
        }

    }

    public static void main(String args[]) {

        String projectArn = null;
        String versionName = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <version_name>\n
\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 models you want to describe.\n\n"
                + "   version_name - (optional) The version name of the model
 that you want to describe. \n\n"
                + "                  If you don't specify a value, all model
 versions are described.\n\n";

        if (args.length > 2 || args.length == 0) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];

        if (args.length == 2) {
            versionName = args[1];
        }

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // Describe the model
            describeMyModel(rekClient, projectArn, versionName);
```

```
            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Como copiar um modelo do Amazon Rekognition Custom Labels (SDK)

Você pode usar a CopyProjectVersionoperação para copiar uma versão do modelo Amazon Rekognition Custom Labels de um projeto de origem do Amazon Rekognition Custom Labels para um projeto de destino. O projeto de destino pode estar em uma AWS conta diferente ou na mesma AWS conta. Um cenário típico é copiar um modelo testado de uma AWS conta de desenvolvimento para uma AWS conta de produção.

Como alternativa, é possível treinar o modelo na conta de destino com o conjunto de dados de origem. Usar a operação `CopyProjectVersion` tem as seguintes vantagens:

- O comportamento do modelo é consistente. O treinamento de modelos não é determinístico e não é garantido que dois modelos treinados com o mesmo conjunto de dados façam as mesmas previsões. Copiar o modelo com `CopyProjectVersion` ajuda a garantir que o comportamento do modelo copiado seja consistente com o modelo de origem e que você não precise testá-lo novamente.
- O treinamento de modelos não é necessário. Isto economiza dinheiro, pois você é cobrado por cada treinamento bem-sucedido de um modelo.

Para copiar um modelo para uma AWS conta diferente, você deve ter um projeto Amazon Rekognition Custom Labels na conta de destino. AWS Para obter informações sobre como criar um projeto, consulte Como criar um projeto. Certifique-se de criar o projeto na AWS conta de destino.

Uma política de projetos é uma política baseada em recursos que define permissões de cópia para a versão do modelo que você deseja copiar. Você precisará usar uma política de projeto quando o projeto de destino estiver em uma AWS conta diferente do projeto de origem.

Não é preciso usar uma política de projeto ao copiar versões do modelo na mesma conta. No entanto, é possível optar por uma política de projetos em projetos entre contas se quiser ter mais controle sobre esses recursos.

Você anexa a política do projeto ao projeto de origem chamando a PutProjectPolicyoperação.

Você não pode usar `CopyProjectVersion` para copiar um modelo para um projeto em uma AWS região diferente. Além disso, você não pode copiar um modelo com o console do Amazon Rekognition Custom Labels. Nestes casos, é possível treinar o modelo no projeto de destino com os conjuntos de dados usados para treinar o modelo de origem. Para obter mais informações, consulte Como treinar um modelo do Amazon Rekognition Custom Labels.

Para copiar um modelo de um projeto de origem para um projeto de destino, faça o seguinte:

Para copiar um modelo

1. Crie um documento de política do projeto.
2. Anexe a política de projeto ao projeto de origem.
3. Copie o modelo com a operação `CopyProjectVersion`.

Para remover uma política de projeto de um projeto, chame DeleteProjectPolicy. Para obter uma lista das políticas do projeto anexadas a um projeto, ligue ListProjectPolicies.

Tópicos

- Como criar um documento de política do projeto
- Como anexar uma política de projeto (SDK)
- Como copiar um modelo (SDK)
- Como listar políticas do projeto (SDK)
- Como excluir uma política de projeto (SDK)

Como criar um documento de política do projeto

O Rekognition Custom Labels usa uma política baseada em recursos, conhecida como política de projeto, para gerenciar permissões de cópia para uma versão modelo. Uma política de projeto é um documento no formato JSON.

Uma política de projeto permite ou nega a permissão de uma entidade principal para copiar uma versão do modelo de um projeto de origem para um projeto de destino. Você precisará de uma política de projeto se o projeto de destino estiver em uma AWS conta diferente. Isso também vale se o projeto de destino estiver na mesma conta da AWS do projeto de origem e você quiser restringir o acesso a versões específicas do modelo. Por exemplo, talvez você queira negar permissões de cópia para uma função específica do IAM em uma AWS conta.

O exemplo a seguir permite que a entidade principal `arn:aws:iam::111111111111:role/Admin` copie a versão do modelo `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::111111111111:role/Admin"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"arn:aws:rekognition:us-east-1:111111111111:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

> **Note**
>
> `Action`, `Resource`, `Principal` e `Effect` são campos obrigatórios em um documento de política de projetos.
> A única `action` compatível é `rekognition:CopyProjectVersion`.
> `NotAction`, `NotResource` e `NotPrincipal` são campos proibidos e não devem estar presentes no documento de política de projetos.

Se você não especificar uma política de projeto, um diretor na mesma AWS conta do projeto de origem ainda poderá copiar um modelo, se o diretor tiver uma política baseada em identidade, como, por exemplo `AmazonRekognitionCustomLabelsFullAccess`, que dê permissão para ligar. `CopyProjectVersion`

O procedimento a seguir cria um arquivo de documento de política de projeto que pode ser usado com o exemplo em Python em Como anexar uma política de projeto (SDK). Se você estiver usando o `put-project-policy` AWS CLI comando, forneça a política do projeto como uma string JSON.

Para criar um documento de política do projeto

1. Em um editor de texto, crie o documento a seguir. Altere os seguintes valores:

   - Efeito: especifique `ALLOW` para conceder permissão de cópia. Especifique `DENY` para negar a permissão de cópia.
   - Entidade principal: a entidade principal que deseja permitir ou negar o acesso às versões do modelo que você especificar em `Resource`. Por exemplo, você pode especificar o principal da conta da AWS para uma AWS conta diferente. Não restringimos as entidades principais que podem ser usados. Para obter mais informações, consulte Especificar uma entidade principal.
   - Recurso: o nome do recurso da Amazon (ARN) da versão do modelo para a qual você deseja especificar permissões de cópia. Se quiser conceder permissões para todas as versões do modelo dentro do projeto de origem, use o seguinte formato: `arn:aws:rekognition:region:account:project/source project/version/*`

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"ALLOW or DENY",
      "Principal":{
        "AWS":"principal"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"Model version ARN"
    }
  ]
}
```

2. Salve a política do projeto no seu computador.
3. Anexe a política de projeto ao projeto de origem seguindo as instruções em Como anexar uma política de projeto (SDK).

Como anexar uma política de projeto (SDK)

Você anexa uma política de projeto a um projeto Amazon Rekognition Custom Labels chamando a operação. PutProjectpolicy

Anexe várias políticas de projeto a um projeto chamando `PutProjectPolicy` para cada política de projeto que você deseja adicionar. É possível anexar até cinco políticas de projeto a um projeto. Se precisar anexar mais políticas de projeto, solicite um aumento de limite.

Ao anexar pela primeira vez uma política de projeto exclusiva a um projeto, não especifique uma ID de revisão no parâmetro de entrada `PolicyRevisionId`. A resposta de `PutProjectPolicy` é um ID de revisão da política do projeto que o Amazon Rekognition Custom Labels cria para você. É possível usar o ID da revisão para atualizar ou excluir a revisão mais recente de uma política do projeto. O Amazon Rekognition Custom Labels mantém somente a revisão mais recente da política de um projeto. Se tentar atualizar ou excluir uma revisão anterior de uma política de projeto, receberá um erro `InvalidPolicyRevisionIdException`.

Para atualizar uma política de projeto existente, especifique o ID de revisão da política de projeto no parâmetro de entrada `PolicyRevisionId`. Você pode obter a revisão das IDs políticas do projeto em um projeto ligando para ListProjectPolicies.

Depois de anexar uma política de projeto a um projeto de origem, é possível copiar o modelo do projeto de origem para o projeto de destino. Para obter mais informações, consulte Como copiar um modelo (SDK).

Para remover uma política de projeto de um projeto, chame DeleteProjectPolicy. Para obter uma lista das políticas do projeto anexadas a um projeto, ligue ListProjectPolicies.

Para anexar uma política de projeto a um projeto (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Crie um documento de política do projeto.

3. Use o código a seguir para anexar a política do projeto ao projeto, na AWS conta confiável, que contém a versão do modelo que você deseja copiar. Para obter o ARN do projeto, ligue. DescribeProjects Para obter a versão do modelo ARN, ligue. DescribeProjectVersions

   AWS CLI

   Altere os seguintes valores:

   - `project-arn`para o ARN do projeto de origem na AWS conta confiável que contém a versão do modelo que você deseja copiar.
   - `policy-name` para um nome de política que você escolher.
   - `principal` Para a entidade principal que deseja permitir ou negar o acesso às versões do modelo que você especificar no `Model version ARN`.
   - `project-version-arn` para o ARN da versão do modelo que você deseja copiar.

   Se quiser atualizar uma política de projeto existente, especifique o parâmetro `policy-revision-id` e forneça o ID de revisão da política de projeto desejada.

```
aws rekognition put-project-policy \
  --project-arn project-arn \
  --policy-name policy-name \
  --policy-document '{ "Version":"2012-10-17", "Statement":
[{ "Effect":"ALLOW or DENY", "Principal":{ "AWS":"principal" },
 "Action":"rekognition:CopyProjectVersion", "Resource":"project-version-
arn" }]}' \
  --profile custom-labels-access
```

   Python

   Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

   - `project_arn`: o ARN do projeto de origem ao qual você deseja anexar a política do projeto.
   - `policy_name`: um nome de política que você escolher.
   - `project_policy`: o arquivo que contém o documento de política do projeto.
   - `policy_revision_id`: (optional). Se quiser atualizar uma revisão existente da política de um projeto, especifique o ID de revisão da política de projeto.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to attach a project policy to an Amazon Rekognition Custom Labels
 project.
"""

import boto3
import argparse
import logging
import json
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def put_project_policy(rek_client, project_arn, policy_name,
 policy_document_file, policy_revision_id=None):
    """
    Attaches a project policy to an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param policy_name: A name for the project policy.
    :param project_arn: The Amazon Resource Name (ARN) of the source project
    that you want to attach the project policy to.
    :param policy_document_file: The JSON project policy document to
    attach to the source project.
    :param policy_revision_id: (Optional) The revision of an existing policy to
 update.
    Pass None to attach new policy.
    :return The revision ID for the project policy.
    """

    try:

        policy_document_json = ""
        response = None
```

```
        with open(policy_document_file, 'r') as policy_document:
            policy_document_json = json.dumps(json.load(policy_document))

        logger.info(
            "Attaching %s project_policy to project %s.",
            policy_name, project_arn)

        if policy_revision_id is None:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                     PolicyName=policy_name,

 PolicyDocument=policy_document_json)

        else:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                     PolicyName=policy_name,

 PolicyDocument=policy_document_json,

 PolicyRevisionId=policy_revision_id)

        new_revision_id = response['PolicyRevisionId']

        logger.info(
            "Finished creating project policy %s. Revision ID: %s",
            policy_name, new_revision_id)

        return new_revision_id

    except ClientError as err:
        logger.exception(
            "Couldn't attach %s project policy to project %s: %s }",
            policy_name, project_arn, err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
```

```
        "project_arn",  help="The Amazon Resource Name (ARN) of the project "
        "that you want to attach the project policy to."
    )
    parser.add_argument(
        "policy_name",  help="A name for the project policy."

    )

    parser.add_argument(
        "project_policy",  help="The file containing the project policy JSON"
    )

    parser.add_argument(
        "--policy_revision_id",  help="The revision of an existing policy to
 update. "
        "If you don't supply a value, a new project policy is created.",
        required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)

        args = parser.parse_args()

        print(f"Attaching policy to {args.project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")


        # Attach a new policy or update an existing policy.

        response = put_project_policy(rekognition_client,
                                      args.project_arn,
                                      args.policy_name,
```

```
                                        args.project_policy,
                                        args.policy_revision_id)

        print(
            f"project policy {args.policy_name} attached to project
 {args.project_arn}")
        print(f"Revision ID: {response}")

    except ClientError as err:
        print("Problem attaching project policy: %s", err)


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- `project_arn`: o ARN do projeto de origem ao qual você deseja anexar a política do projeto.
- `project_policy_name`: um nome de política que você escolher.
- `project_policy_document`: o arquivo que contém o documento de política do projeto.
- `project_policy_revision_id`: (optional). Se quiser atualizar uma revisão existente da política de um projeto, especifique o ID de revisão da política de projeto.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.PutProjectPolicyRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class PutProjectPolicy {

    public static final Logger logger =
 Logger.getLogger(PutProjectPolicy.class.getName());

    public static void putMyProjectPolicy(RekognitionClient rekClient, String
 projectArn, String projectPolicyName,
             String projectPolicyFileName, String projectPolicyRevisionId)
 throws IOException {

        try {

            Path filePath = Path.of(projectPolicyFileName);



            String policyDocument = Files.readString(filePath);

            String[] logArguments = new String[] { projectPolicyFileName,
 projectPolicyName };

            PutProjectPolicyRequest putProjectPolicyRequest = null;

            logger.log(Level.INFO, "Attaching Project policy: {0} to project:
 {1}", logArguments);

            // Attach the project policy.

            if (projectPolicyRevisionId == null) {
                putProjectPolicyRequest =
 PutProjectPolicyRequest.builder().projectArn(projectArn)

 .policyName(projectPolicyName).policyDocument(policyDocument).build();
            } else {
                putProjectPolicyRequest =
 PutProjectPolicyRequest.builder().projectArn(projectArn)

 .policyName(projectPolicyName).policyRevisionId(projectPolicyRevisionId)
                        .policyDocument(policyDocument)
```

```
                        .build();
            }

            rekClient.putProjectPolicy(putProjectPolicyRequest);

            logger.log(Level.INFO, "Attached Project policy: {0} to project:
{1}", logArguments);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: "
                + "<project_arn> <project_policy_name> <policy_document>
<project_policy_revision_id>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that you want to
attach the project policy to.\n\n"
                + "   project_policy_name - A name for the project policy.\n\n"
                + "   project_policy_document - The file name of the project
policy.\n\n"
                + "   project_policy_revision_id - (Optional) The revision ID of
the project policy that you want to update.\n\n";

        if (args.length < 3 || args.length > 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String projectPolicyName = args[1];
        String projectPolicyDocument = args[2];
        String projectPolicyRevisionId = null;

        if (args.length == 4) {
            projectPolicyRevisionId = args[3];
```

```
        }

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Attach the project policy.
            putMyProjectPolicy(rekClient, projectArn, projectPolicyName,
 projectPolicyDocument,
                    projectPolicyRevisionId);

            System.out.println(
                    String.format("project policy %s: attached to project: %s",
 projectPolicyName, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (IOException intError) {
            logger.log(Level.SEVERE, "Exception while reading policy document:
 {0}", intError.getMessage());
            System.exit(1);
        }

    }

}
```

4. Copie a versão do modelo seguindo as instruções em Como copiar um modelo (SDK).

Como copiar um modelo (SDK)

É possível usar a API `CopyProjectVersion` para copiar uma versão do modelo de um projeto de origem para um projeto de destino. O projeto de destino pode estar em uma AWS conta diferente, mas deve estar na mesma AWS região. Se o projeto de destino estiver em uma AWS conta diferente (ou se você quiser conceder permissões específicas para uma versão do modelo copiada em uma AWS conta), você deverá anexar uma política de projeto ao projeto de origem. Para obter mais informações, consulte Como criar um documento de política do projeto. A API `CopyProjectVersion` requer acesso ao bucket do Amazon S3.

O modelo copiado inclui os resultados do treinamento para o modelo de origem, mas não inclui os conjuntos de dados de origem.

A AWS conta de origem não tem propriedade sobre o modelo copiado em uma conta de destino, a menos que você configure as permissões apropriadas.

Para copiar um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Anexe uma política de projeto ao projeto de origem seguindo as instruções em Como anexar uma política de projeto (SDK).
3. Se você estiver copiando o modelo para uma AWS conta diferente, verifique se você tem um projeto na AWS conta de destino.
4. Use o código a seguir para copiar a versão do modelo para um projeto de destino.

   AWS CLI

   Altere os seguintes valores:

   - `source-project-arn` para o ARN do projeto de origem que contém a versão do modelo que você deseja copiar.
   - `source-project-version-arn` para o ARN da versão do modelo que você deseja copiar.
   - `destination-project-arn` para o ARN do projeto de destino no qual você deseja copiar o modelo.
   - `version-name` para um nome de versão do modelo no projeto de destino.

- `bucket` para o bucket do S3 para o qual você deseja copiar os resultados do treinamento para o modelo de origem.
- `folder` para a pasta no `bucket` para a qual você deseja copiar os resultados do treinamento para o modelo de origem.
- (Opcional) `kms-key-id` para o ID da chave do AWS Key Management Service para o modelo.
- (Opcional) `key` para uma chave de tag de sua escolha.
- (Opcional) `value` para um valor de tag de sua escolha.

```
aws rekognition copy-project-version \
  --source-project-arn source-project-arn \
  --source-project-version-arn source-project-version-arn \
  --destination-project-arn destination-project-arn \
  --version-name version-name \
  --output-config '{"S3Bucket":"bucket","S3KeyPrefix":"folder"}' \
  --kms-key-id arn:myKey \
  --tags '{"key":"key"}' \
  --profile custom-labels-access
```

Python

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- `source_project_arn`— o ARN do projeto de origem na AWS conta de origem que contém a versão do modelo que você deseja copiar.
- `source_project_version-arn`— o ARN da versão do modelo na AWS conta de origem que você deseja copiar.
- `destination_project_arn`: o ARN do projeto de destino para o qual você deseja copiar o modelo.
- `destination_version_name`: um nome de versão para o modelo no projeto de destino.
- `training_results`: a localização do S3 para o qual você deseja copiar os resultados do treinamento para o modelo de origem.
- (Opcional) `kms_key_id` para o ID da chave do AWS Key Management Service para o modelo.
- (Opcional) `tag_name` para uma chave de tag de sua escolha.

- (Opcional) `tag_value` para um valor de tag de sua escolha.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def copy_model(
    rekognition_client, source_project_arn, source_project_version_arn,
        destination_project_arn, training_results, destination_version_name):
    """
    Copies a version of a Amazon Rekognition Custom Labels model.

    :param rekognition_client: A Boto3 Amazon Rekognition Custom Labels client.
    :param source_project_arn: The ARN of the source project that contains the
    model that you want to copy.
    :param source_project_version_arn: The ARN of the model version that you
 want
    to copy.
    :param destination_project_Arn: The ARN of the project that you want to copy
 the model
    to.
    :param training_results: The Amazon S3 location where training results for
 the model
    should be stored.
    return: The model status and version.
    """
    try:
        logger.info("Copying model...%s from %s to %s ",
 source_project_version_arn,
                    source_project_arn,
                    destination_project_arn)

        output_bucket, output_folder = training_results.replace(
            "s3://", "").split("/", 1)
```

```
        output_config = {"S3Bucket": output_bucket,
                         "S3KeyPrefix": output_folder}

        response = rekognition_client.copy_project_version(
            DestinationProjectArn=destination_project_arn,
            OutputConfig=output_config,
            SourceProjectArn=source_project_arn,
            SourceProjectVersionArn=source_project_version_arn,
            VersionName=destination_version_name
        )

        destination_model_arn = response["ProjectVersionArn"]

        logger.info("Destination model ARN: %s", destination_model_arn)

        # Wait until training completes.
        finished = False
        status = "UNKNOWN"
        while finished is False:
            model_description =
 rekognition_client.describe_project_versions(ProjectArn=destination_project_arn,
                    VersionNames=[destination_version_name])
            status = model_description["ProjectVersionDescriptions"][0]
["Status"]

            if status == "COPYING_IN_PROGRESS":
                logger.info("Model copying in progress...")
                time.sleep(60)
                continue

            if status == "COPYING_COMPLETED":
                logger.info("Model was successfully copied.")

            if status == "COPYING_FAILED":
                logger.info(
                    "Model copy failed: %s ",
                    model_description["ProjectVersionDescriptions"][0]
["StatusMessage"])

            finished = True
    except ClientError:
        logger.exception("Couldn't copy model.")
        raise
    else:
```

```
        return destination_model_arn, status


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "source_project_arn",
        help="The ARN of the project that contains the model that you want to
 copy."
    )

    parser.add_argument(
        "source_project_version_arn",
        help="The ARN of the model version that you want to copy."
    )

    parser.add_argument(
        "destination_project_arn",
        help="The ARN of the project which receives the copied model."
    )

    parser.add_argument(
        "destination_version_name",
        help="The version name for the model in the destination project."
    )

    parser.add_argument(
        "training_results",
        help="The S3 location in the destination account that receives the
 training results for the copied model."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:
```

```
        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Copying model version {args.source_project_version_arn} to project
 {args.destination_project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        # Copy the model.

        model_arn, status = copy_model(rekognition_client,
                                       args.source_project_arn,
                                       args.source_project_version_arn,
                                       args.destination_project_arn,
                                       args.training_results,
                                       args.destination_version_name,
                                       )

        print(f"Finished copying model: {model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        print(f"Problem copying model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- `source_project_arn`— o ARN do projeto de origem na AWS conta de origem que contém a versão do modelo que você deseja copiar.
- `source_project_version-arn`— o ARN da versão do modelo na AWS conta de origem que você deseja copiar.
- `destination_project_arn`: o ARN do projeto de destino para o qual você deseja copiar o modelo.

- `destination_version_name`: um nome de versão para o modelo no projeto de destino.
- `output_bucket`: o bucket do S3 para o qual você deseja copiar os resultados do treinamento para o modelo de origem.
- `output_folder`: a pasta no S3 para a qual você deseja copiar os resultados do treinamento para a versão do modelo de origem.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CopyProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.CopyProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CopyModel {

    public static final Logger logger =
 Logger.getLogger(CopyModel.class.getName());

    public static ProjectVersionDescription copyMyModel(RekognitionClient
 rekClient,
            String sourceProjectArn,
            String sourceProjectVersionArn,
```

```
            String destinationProjectArn,
            String versionName,
            String outputBucket,
            String outputFolder) throws InterruptedException {

        try {

            OutputConfig outputConfig =
OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

            String[] logArguments = new String[] { versionName,
sourceProjectArn, destinationProjectArn };

            logger.log(Level.INFO, "Copying model {0} for from project {1} to
project {2}", logArguments);

            CopyProjectVersionRequest copyProjectVersionRequest =
CopyProjectVersionRequest.builder()
                    .sourceProjectArn(sourceProjectArn)
                    .sourceProjectVersionArn(sourceProjectVersionArn)
                    .versionName(versionName)
                    .destinationProjectArn(destinationProjectArn)
                    .outputConfig(outputConfig)
                    .build();

            CopyProjectVersionResponse response =
rekClient.copyProjectVersion(copyProjectVersionRequest);

            logger.log(Level.INFO, "Destination model ARN: {0}",
response.projectVersionArn());
            logger.log(Level.INFO, "Copying model...");

            // wait until copying completes.

            boolean finished = false;

            ProjectVersionDescription copiedModel = null;

            while (Boolean.FALSE.equals(finished)) {
                DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                        .versionNames(versionName)
                        .projectArn(destinationProjectArn)
                        .build();
```

```
                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                        .projectVersionDescriptions()) {

                    copiedModel = projectVersionDescription;

                    switch (projectVersionDescription.status()) {

                        case COPYING_IN_PROGRESS:
                            logger.log(Level.INFO, "Copying model...");
                            Thread.sleep(5000);
                            continue;

                        case COPYING_COMPLETED:
                            finished = true;
                            logger.log(Level.INFO, "Copying completed");
                            break;

                        case COPYING_FAILED:
                            finished = true;
                            logger.log(Level.INFO, "Copying failed...");
                            break;

                        default:
                            finished = true;
                            logger.log(Level.INFO, "Unexpected copy status %s",
                                    projectVersionDescription.statusAsString());
                            break;

                    }

                }

            }

            logger.log(Level.INFO, "Finished copying model {0} for from project
{1} to project {2}", logArguments);
```

```
            return copiedModel;

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String sourceProjectArn = null;
        String sourceProjectVersionArn = null;
        String destinationProjectArn = null;
        String versionName = null;
        String bucket = null;
        String location = null;

        final String USAGE = "\n" + "Usage: "
                + "<source_project_arn> <source_project_version_arn>
<destination_project_arn> <version_name> <output_bucket> <output_folder>\n\n"
                + "Where:\n"
                + "   source_project_arn - The ARN of the project that contains
the model that you want to copy. \n\n"
                + "   source_project_version_arn - The ARN of the project that
contains the model that you want to copy. \n\n"
                + "   destination_project_arn - The ARN of the destination
project that you want to copy the model to. \n\n"
                + "   version_name - A version name for the copied model.\n\n"
                + "   output_bucket - The S3 bucket in which to place the
training output. \n\n"
                + "   output_folder - The folder within the bucket that the
training output is stored in. \n\n";

        if (args.length != 6) {
            System.out.println(USAGE);
            System.exit(1);
        }

        sourceProjectArn = args[0];
        sourceProjectVersionArn = args[1];
        destinationProjectArn = args[2];
        versionName = args[3];
```

```
        bucket = args[4];
        location = args[5];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Copy the model.
            ProjectVersionDescription copiedModel = copyMyModel(rekClient,
                    sourceProjectArn,
                    sourceProjectVersionArn,
                    destinationProjectArn,
                    versionName,
                    bucket,
                    location);

            System.out.println(String.format("Model copied: %s Status: %s",
                    copiedModel.projectVersionArn(),
                    copiedModel.statusMessage()));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Como listar políticas do projeto (SDK)

Você pode usar a ListProjectPoliciesoperação para listar as políticas do projeto que estão anexadas a um projeto Amazon Rekognition Custom Labels.

Para listar as políticas de projeto a um projeto (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para listar as políticas do projeto.

   AWS CLI

   Altere `project-arn` para o nome do recurso da Amazon do projeto para o qual você deseja listar as políticas do projeto em anexo.

   ```
   aws rekognition list-project-policies \
     --project-arn project-arn \
     --profile custom-labels-access
   ```

   Python

   Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

   - project_arn: o nome do recurso da Amazon do projeto para o qual você deseja listar as políticas do projeto em anexo.

   Por exemplo: `python list_project_policies.py project_arn`

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Amazon Rekognition Custom Labels model example used in the service
    documentation:
   https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
   sdk.html
   Shows how to list the project policies in an Amazon Rekogntion Custom Labels
    project.
   """
   ```

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def display_project_policy(project_policy):
    """
    Displays information about a Custom Labels project policy.
    :param project_policy: The project policy (ProjectPolicy)
    that you want to display information about.
    """
    print(f"Policy name: {(project_policy['PolicyName'])}")
    print(f"Project Arn: {project_policy['ProjectArn']}")
    print(f"Document: {(project_policy['PolicyDocument'])}")
    print(f"Revision ID: {(project_policy['PolicyRevisionId'])}")
    print()



def list_project_policies(rek_client, project_arn):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The Amazon Resource Name of the project you want to use.
    """

    try:

        max_results = 5
        pagination_token = ''
        finished = False

        logger.info("Listing project policies in: %s.", project_arn)
        print('Projects\n--------')
        while not finished:

            response = rek_client.list_project_policies(
                ProjectArn=project_arn, MaxResults=max_results,
 NextToken=pagination_token)
```

```
            for project in response['ProjectPolicies']:
                display_project_policy(project)

            if 'NextToken' in response:
                pagination_token = response['NextToken']
            else:
                finished = True

        logger.info("Finished listing project policies.")

    except ClientError as err:
        logger.exception(
            "Couldn't list policies for - %s: %s",
            project_arn,err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn",  help="The Amazon Resource Name of the project for which
 you want to list project policies."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Listing project policies in: {args.project_arn}")
```

```
        # List the project policies.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_project_policies(rekognition_client,
                              args.project_arn)

    except ClientError as err:
        print(f"Problem list project_policies: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- project_arn: o ARN do projeto que tem as políticas do projeto que você deseja listar.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesRequest;
import
 software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectPolicy;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class ListProjectPolicies {
```

```
   public static final Logger logger =
Logger.getLogger(ListProjectPolicies.class.getName());

   public static void listMyProjectPolicies(RekognitionClient rekClient, String
projectArn) {

       try {

           logger.log(Level.INFO, "Listing project policies for project: {0}",
projectArn);

           // List the project policies.

           Boolean finished = false;
           String nextToken = null;

           while (Boolean.FALSE.equals(finished)) {

               ListProjectPoliciesRequest listProjectPoliciesRequest =
ListProjectPoliciesRequest.builder()
                       .maxResults(5)
                       .projectArn(projectArn)
                       .nextToken(nextToken)
                       .build();

               ListProjectPoliciesResponse response =
rekClient.listProjectPolicies(listProjectPoliciesRequest);

               for (ProjectPolicy projectPolicy : response.projectPolicies()) {

                   System.out.println(String.format("Name: %s",
projectPolicy.policyName()));
                   System.out.println(String.format("Revision ID: %s\n",
projectPolicy.policyRevisionId()));

               }

               nextToken = response.nextToken();

               if (nextToken == null) {
                   finished = true;

               }
```

```
            }

            logger.log(Level.INFO, "Finished listing project policies for
 project: {0}", projectArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn> \n\n" + "Where:
\n"
                + "   project_arn - The ARN of the project with the project
 policies that you want to list.\n\n";
        ;

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // List the project policies.
            listMyProjectPolicies(rekClient, projectArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
```

```
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Como excluir uma política de projeto (SDK)

Você pode usar a DeleteProjectPolicyoperação para excluir uma revisão de uma política de projeto existente de um projeto Amazon Rekognition Custom Labels. Se você quiser excluir todas as revisões de uma política de projeto que estão anexadas a um projeto, use ListProjectPoliciespara obter a revisão IDs de cada política de projeto anexada ao projeto. Em seguida, chame `DeletePolicy` para cada nome de política.

Para excluir uma revisão de uma política de projeto (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para excluir uma política de projeto.

   DeletePolicy levaProjectARN, PolicyName PolicyRevisionId e. ProjectARNe PolicyName são necessários para essa API. PolicyRevisionIdé opcional, mas pode ser incluído para fins de atualizações atômicas.

   AWS CLI

   Altere os seguintes valores:

   - `policy-name` para o nome da política de projetos que você deseja excluir.
   - `policy-revision-id` para o ID de revisão da política de projetos que você deseja excluir.
   - `project-arn` para o nome do recurso da Amazon do projeto que contém a revisão da política do projeto que você deseja excluir.

```
aws rekognition delete-project-policy \
```

```
    --policy-name policy-name \
    --policy-revision-id policy-revision-id \
    --project-arn project-arn \
  --profile custom-labels-access
```

Python

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- `policy-name`: o nome da política de projetos que você deseja excluir.
- `project-arn`: o nome do recurso da Amazon do projeto que contém a revisão da política do projeto que você deseja excluir.
- `policy-revision-id`: o ID de revisão da política de projetos que você deseja excluir.

Por exemplo: `python delete_project_policy.py` *`policy_name project_arn policy_revision_id`*

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to delete a revision of a project policy.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def delete_project_policy(rekognition_client, policy_name,  project_arn,
 policy_revision_id=None):
    """
```

```
    Deletes a project policy.

    :param rekognition_client: A Boto3 Amazon Rekognition client.
    :param policy_name: The name of the project policy that you want to delete.
    :param policy_revision_id: The revsion ID for the project policy that you
 want to delete.
    :param project_arn: The Amazon Resource Name of the project that contains
 the project policy
    that you want to delete.
    """
    try:
        logger.info("Deleting project policy: %s", policy_name)

        if policy_revision_id is None:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                ProjectArn=project_arn)

        else:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                PolicyRevisionId=policy_revision_id,
                ProjectArn=project_arn)

        logger.info("Deleted project policy: %s", policy_name)
    except ClientError:
        logger.exception("Couldn't delete project policy.")
        raise


def confirm_project_policy_deletion(policy_name):
    """
    Confirms deletion of the project policy. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(
        f"Are you sure you wany to delete project policy {policy_name} ?\n",
 policy_name)

    delete = input("Enter delete to delete your project policy: ")
    if delete == "delete":
        return True
    else:
        return False
```

```
def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "policy_name", help="The ARN of the project that contains the project
 policy that you want to delete."
    )

    parser.add_argument(
        "project_arn", help="The ARN of the project project policy you want to
 delete."
    )

    parser.add_argument(
        "--policy_revision_id", help="(Optional) The revision ID of the project
 policy that you want to delete.",
        required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        if confirm_project_policy_deletion(args.policy_name) is True:
            print(f"Deleting project_policy: {args.policy_name}")

            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            # Delete the project policy.
```

```
            delete_project_policy(rekognition_client,
                                  args.policy_name,
                                  args.project_arn,
                                  args.policy_revision_id)

            print(f"Finished deleting project policy: {args.policy_name}")
        else:
            print(f"Not deleting project policy {args.policy_name}")
    except ClientError as err:
        print(f"Couldn't delete project policy in {args.policy_name}: {err}")



if __name__ == "__main__":
    main()
```

Java V2

Use o seguinte código: Forneça os seguintes parâmetros de linha de comando:

- `policy-name`: o nome da política de projetos que você deseja excluir.
- `project-arn`: o nome do recurso da Amazon do projeto que contém a revisão da política do projeto que você deseja excluir.
- `policy-revision-id`: o ID de revisão da política de projetos que você deseja excluir.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectPolicyRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProjectPolicy {

    public static final Logger logger =
 Logger.getLogger(DeleteProjectPolicy.class.getName());

    public static void deleteMyProjectPolicy(RekognitionClient rekClient, String
 projectArn,
            String projectPolicyName,
            String projectPolicyRevisionId)
            throws InterruptedException {

        try {
            String[] logArguments = new String[] { projectPolicyName,
 projectPolicyRevisionId };

            logger.log(Level.INFO, "Deleting: Project policy: {0} revision:
 {1}", logArguments);

            // Delete the project policy.

            DeleteProjectPolicyRequest deleteProjectPolicyRequest =
 DeleteProjectPolicyRequest.builder()
                    .policyName(projectPolicyName)
                    .policyRevisionId(projectPolicyRevisionId)
                    .projectArn(projectArn).build();

            rekClient.deleteProjectPolicy(deleteProjectPolicyRequest);

            logger.log(Level.INFO, "Deleted: Project policy: {0} revision: {1}",
 logArguments);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }
```

```
    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn>
 <project_policy_name> <project_policy_revision_id>\n\n"
                + "Where:\n"
                + "   project_arn - The ARN of the project that has the project
 policy that you want to delete.\n\n"
                + "   project_policy_name - The name of the project policy that
 you want to delete.\n\n"
                + "   project_policy_revision_id - The revision of the project
 policy that you want to delete.\n\n";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String projectPolicyName = args[1];
        String projectPolicyRevisionId = args[2];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Delete the project policy.
            deleteMyProjectPolicy(rekClient, projectArn, projectPolicyName,
 projectPolicyRevisionId);

            System.out.println(String.format("project policy deleted: %s
 revision: %s", projectPolicyName,
                    projectPolicyRevisionId));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }
```

```
        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Exemplos de rótulos personalizados

Esta seção contém exemplos que mostram como você usa os recursos do Amazon Rekognition Custom Labels.

Exemplo	Descrição
Melhorar um modelo com Model Feedback	Mostra como melhorar um modelo usando a verificação humana para criar um novo conjunto de dados de treinamento.
Demonstração do Amazon Rekognition Custom Labels	Demonstração de uma interface de usuário que exibe os resultados de uma chamada para `DetectCustomLabels` .
Detectar rótulos personalizados em vídeos	Mostra como `DetectCustomLabels` pode ser usado com quadros extraídos de um vídeo.
Analisando imagens com uma AWS Lambda função	Mostra como `DetectCustomLabels` pode ser usado com uma função do Lambda.
Como criar um arquivo de manifesto de um arquivo CSV	Mostra como usar um arquivo CSV para criar um arquivo de manifesto adequado para descobrir objetos, cenas e conceitos associado s a uma imagem inteira (classificação).

Melhorar um modelo com Model Feedback

A solução Model Feedback permite que você forneça comentários sobre as previsões do seu modelo e faça melhorias usando a verificação humana. Dependendo do caso de uso, é possível ter sucesso com um conjunto de dados de treinamento que tenha apenas algumas imagens. Um conjunto de treinamento anotado maior pode ser necessário para criar um modelo mais preciso. Ao usar a solução Model Feedback, é possível criar um conjunto de dados maior por meio da assistência do modelo.

Para instalar e configurar a solução Model Feedback, consulte Solução Model Feedback.

O fluxo de trabalho para a melhoria contínua do modelo é o seguinte:

1. Treine a primeira versão do seu modelo (possivelmente com um pequeno conjunto de dados de treinamento).
2. Forneça um conjunto de dados sem anotações para a solução Model Feedback.
3. A solução Model Feedback usa o modelo atual. Ele inicia trabalhos de verificação humana para anotar um novo conjunto de dados.
4. Com base no feedback humano, a solução Model Feedback gera um arquivo de manifesto que você usa para criar um novo modelo.

Demonstração do Amazon Rekognition Custom Labels

A demonstração de etiquetas personalizadas do Amazon Rekognition mostra uma interface de usuário que analisa imagens do seu computador local usando a API. DetectCustomLabels

O aplicativo mostra informações sobre os modelos de etiquetas personalizadas do Amazon Rekognition em sua conta. AWS Depois de selecionar um modelo em execução, é possível analisar uma imagem do seu computador local. Se necessário, é possível iniciar um modelo. Também é possível interromper um modelo em execução. O aplicativo mostra a integração com outros serviços da AWS, como Amazon Cognito, Amazon S3 e Amazon. CloudFront

Para obter mais informações, consulte Demonstração do Amazon Rekognition Custom Labels.

Detectar rótulos personalizados em vídeos

O exemplo a seguir mostra como é possível usar `DetectCustomLabels` com quadros extraídos de um vídeo. O código foi testado com arquivos de vídeo nos formatos mov e mp4.

Como usar **`DetectCustomLabels`** com quadros capturados

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
2. Certifique-se de que você tem as permissões `rekognition:DetectCustomLabels` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.

3. Considere o código de exemplo a seguir. Altere o valor de `videoFile` para o nome de arquivo de vídeo. Altere o valor de `projectVersionArn` para o nome do recurso da Amazon (ARN) do seu modelo do Amazon Rekognition Custom Labels.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to analyze a local video with an Amazon Rekognition Custom Labels model.
"""
import argparse
import logging
import json
import math
import cv2
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def analyze_video(rek_client, project_version_arn, video_file):
    """
    Analyzes a local video file with an Amazon Rekognition Custom Labels model.
    Creates a results JSON file based on the name of the supplied video file.
    :param rek_client: A Boto3 Amazon Rekognition client.
    :param project_version_arn: The ARN of the Custom Labels model that you want to
 use.
    :param video_file: The video file that you want to analyze.
    """

    custom_labels = []
    cap = cv2.VideoCapture(video_file)
    frame_rate = cap.get(5)  # Frame rate.
    while cap.isOpened():
        frame_id = cap.get(1)  # Current frame number.
        print(f"Processing frame id: {frame_id}")
        ret, frame = cap.read()
        if ret is not True:
            break
        if frame_id % math.floor(frame_rate) == 0:
```

```
            has_frame, image_bytes = cv2.imencode(".jpg", frame)

            if has_frame:
                response = rek_client.detect_custom_labels(
                    Image={
                        'Bytes': image_bytes.tobytes(),
                    },
                    ProjectVersionArn=project_version_arn
                )

            for elabel in response["CustomLabels"]:
                elabel["Timestamp"] = (frame_id/frame_rate)*1000
                custom_labels.append(elabel)

    print(custom_labels)

    with open(video_file + ".json", "w", encoding="utf-8") as f:
        f.write(json.dumps(custom_labels))

    cap.release()


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_version_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "video_file", help="The local path to the video that you want to analyze."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:
        # Get command line arguments.
```

```
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        analyze_video(rekognition_client,
                     args.project_version_arn, args.video_file)

    except ClientError as err:
        print(f"Couldn't analyze video: {err}")


if __name__ == "__main__":
    main()
```

Analisando imagens com uma AWS Lambda função

AWS Lambda é um serviço de computação que permite executar código sem provisionar ou gerenciar servidores. Por exemplo, você pode analisar imagens enviadas de um aplicativo móvel sem precisar criar um servidor para hospedar o código do aplicativo. As instruções a seguir mostram como criar uma função do Lambda em Python que chame DetectCustomLabels. A função analisa uma imagem fornecida e retorna uma lista de rótulos encontrados na imagem. As instruções incluem um exemplo de código em Python que mostra como chamar a função do Lambda com uma imagem em um bucket do Amazon S3 ou uma imagem fornecida por um computador local.

Tópicos

- Etapa 1: criar uma AWS Lambda função (console)
- Etapa 2: (opcional) crie uma camada (console)
- Etapa 3: adicione o código em Python (console)
- Etapa 4: teste sua função do Lambda

Etapa 1: criar uma AWS Lambda função (console)

Nesta etapa, você cria uma AWS função vazia e uma função de execução do IAM que permite que sua função chame a `DetectCustomLabels` operação. Ele também concede acesso ao bucket

do Amazon S3 que armazena imagens para análise. Também é possível especificar variáveis de ambiente para o seguinte:

- O modelo do Amazon Rekognition Custom Labels que você deseja que sua função do Lambda use.
- O limite de confiança que você deseja que o modelo use.

Posteriormente, você adiciona o código-fonte e, opcionalmente, uma camada à função do Lambda.

Para criar uma AWS Lambda função (console)

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha a opção Criar função. Para obter mais informações, consulte Criar uma função do Lambda no console.
3. Escolha as seguintes opções:
   - Escolha Criar do zero.
   - Insira um valor para Nome da função.
   - Em Runtime, escolha Python 3.10.
4. Escolha Criar função para criar a função do AWS Lambda .
5. Em sua página da função, escolha a guia Configuração.
6. No painel Variáveis de ambiente, escolha Editar.
7. Adicione as seguintes variáveis de ambiente: Para cada variável, escolha Adicionar variável de ambiente, e insira a chave e o valor da variável.

Chave	Valor
MODEL_ARN	O nome do recurso da Amazon (ARN) do modelo que deseja quer que sua função do Lambda use. É possível obter o ARN do modelo na guia Usar modelo da página de detalhes do modelo no console do Amazon Rekognition Custom Labels.

Chave	Valor
CONFIDENCE	O valor mínimo (de 0 a 100) da confiança do modelo na previsão de um rótulo. A função do Lambda não retorna rótulos com valores de confiança inferiores a esse valor.

8. Escolha Salvar para salvar as variáveis de ambiente.
9. No painel Permissões, em Nome do perfil, selecione a o perfil de execução para abri-lo no console do IAM.
10. Na guia Permissões, escolha Adicionar permissões e Criar política em linha.
11. Escolha JSON e substitua a política padrão com a política a seguir.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "rekognition:DetectCustomLabels",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectCustomLabels"
        }
    ]
}
```

12. Escolha Próximo.
13. Em Detalhes da política, insira um nome para a política, como DetectCustomLabels-access.
14. Escolha Criar política.
15. Se estiver armazenando imagens para análise em um bucket do Amazon S3, repita as etapas 10 a 14.

    a. Para a etapa 11, use a política a seguir. *bucket/folder path*Substitua as imagens que você deseja analisar pelo bucket e pelo caminho da pasta do Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Access",
```

```
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket/folder path/*"
        }
    ]
}
```

b. Para a etapa 13, escolha um nome de política diferente, como S3Bucket-access.

Etapa 2: (opcional) crie uma camada (console)

Para executar este exemplo, não é preciso executar esta etapa. A `DetectCustomLabels` operação está incluída no ambiente padrão do Lambda Python como parte do AWS SDK for Python (Boto3). Se outras partes da sua função do Lambda precisarem de atualizações de AWS serviço recentes que não estejam no ambiente padrão do Lambda Python, siga esta etapa para adicionar a versão mais recente do SDK do Boto3 como uma camada à sua função.

Primeiro, você cria um arquivo .zip que pode conter o SDK do Boto3. Uma camada é criada o arquivo de arquivos.zip é adicionado à camada. Para obter mais informações, consulte Como usar camadas com sua função do Lambda.

Para criar e adicionar uma camada (console)

1. Abra um prompt de comando e execute os comandos a seguir.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Observe o nome do arquivo zip (boto3-layer.zip). Ele será necessário na etapa 6 deste procedimento.
3. Abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
4. No painel de navegação, escolha Camadas.
5. Escolha Criar camada.
6. Insira valores para Nome e Descrição.
7. Escolha Fazer upload de um arquivo .zip e escolha Fazer upload.
8. Na caixa de diálogo, escolha o arquivo de arquivos.zip (boto3-layer.zip) criada na etapa 1 desse procedimento.

9. Para runtimes compatíveis, escolha Python 3.9.
10. Escolha Criar para criar a camada.
11. Escolha o ícone do menu do painel de navegação.
12. Selecione Funções no painel de navegação.
13. Na lista de recursos, escolha a função que você criou em Etapa 1: criar uma AWS Lambda função (console).
14. Escolha a guia Código.
15. Na seção Camadas, escolha Adicionar uma camada.
16. Escolha camadas personalizadas.
17. Em Camadas personalizadas, escolha o nome da camada que você inseriu na etapa 6.
18. Em Versão, escolha a versão da camada, que deve ser 1.
19. Escolha Adicionar.

Etapa 3: adicione o código em Python (console)

Nesta etapa, o código em Python é adicionado à sua função do Lambda usando o editor de código do console do Lambda. O código analisa uma imagem fornecida com `DetectCustomLabels` e retorna uma lista de rótulos encontrados na imagem. A imagem fornecida pode estar localizada em um bucket do Amazon S3 ou fornecida como bytes de imagem codificados em byte64.

Para adicionar um código em Python (console)

1. Se não estiver no console do Lambda, faça o seguinte:
   a. Abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
   b. Abra a função do Lambda que você criou em Etapa 1: criar uma AWS Lambda função (console).
2. Escolha a guia Código.
3. Em Código-fonte, substitua o código em lambda_function.py pelo seguinte:

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
An AWS lambda function that analyzes images with an the Amazon Rekognition
```

```
Custom Labels model.
"""
import json
import base64
from os import environ
import logging
import boto3

from botocore.exceptions import ClientError

# Set up logging.
logger = logging.getLogger(__name__)

# Get the model ARN and confidence.
model_arn = environ['MODEL_ARN']
min_confidence = int(environ.get('CONFIDENCE', 50))

# Get the boto3 client.
rek_client = boto3.client('rekognition')


def lambda_handler(event, context):
    """
    Lambda handler function
    param: event: The event object for the Lambda function.
    param: context: The context object for the lambda function.
    return: The labels found in the image passed in the event
    object.
    """

    try:

        # Determine image source.
        if 'image' in event:
            # Decode the image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image = {'Bytes': img_b64decoded}


        elif 'S3Object' in event:
            image = {'S3Object':
                     {'Bucket':  event['S3Object']['Bucket'],
                      'Name': event['S3Object']['Name']}
```

```
                    }

        else:
            raise ValueError(
                'Invalid source. Only image base 64 encoded image bytes or S3Object
are supported.')


        # Analyze the image.
        response = rek_client.detect_custom_labels(Image=image,
            MinConfidence=min_confidence,
            ProjectVersionArn=model_arn)

        # Get the custom labels
        labels = response['CustomLabels']

        lambda_response = {
            "statusCode": 200,
            "body": json.dumps(labels)
        }

    except ClientError as err:
        error_message = f"Couldn't analyze image. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message
            }
        }
        logger.error("Error function %s: %s",
            context.invoked_function_arn, error_message)

    except ValueError as val_error:
        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": "ValueError",
                "ErrorMessage": format(val_error)
            }
        }
        logger.error("Error function %s: %s",
```

```
            context.invoked_function_arn, format(val_error))

    return lambda_response
```

4. Escolha Implantar para implantar sua função do Lambda.

Etapa 4: teste sua função do Lambda

Nesta etapa, o código em Python é usado em seu computador para passar uma imagem local, ou uma imagem em um bucket do Amazon S3, para sua função do Lambda. As imagens passadas de um computador local devem ter menos de 6.291.456 bytes. Se suas imagens forem maiores, faça o upload das imagens em um bucket do Amazon S3 e chame o script com o caminho do Amazon S3 para a imagem. Para obter mais informações sobre como fazer upload de arquivos para um bucket do Amazon S3, consulte Fazer upload de objetos.

Certifique-se de executar o código na mesma AWS região em que você criou a função Lambda. Você pode visualizar a AWS região da sua função Lambda na barra de navegação da página de detalhes da função no console do Lambda.

Se a AWS Lambda função retornar um erro de tempo limite, estenda o período de tempo limite da função Lambda. Para obter mais informações, consulte Configurando o tempo limite da função (console).

Para obter mais informações sobre como invocar uma função Lambda a partir do seu código, consulte AWS Lambda Invocando funções.

Para testar a função do Lambda

1. Certifique-se de que você tem a permissão `lambda:InvokeFunction`. É possível usar a política a seguir.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InvokeLambda",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
```

```
    ]
}
```

É possível obter o ARN para sua função do Lambda na visão geral da função no console do Lambda.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .

- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.

- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

2. Instale e configure o AWS SDK para Python. Para obter mais informações, consulte Etapa 4: configurar o AWS CLI e AWS SDKs.
3. Inicie o modelo que você especificou na etapa 7 de Etapa 1: criar uma AWS Lambda função (console).
4. Salve o código a seguir em um arquivo chamado `client.py`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Test code for running the Amazon Rekognition Custom Labels Lambda
function example code.
"""

import argparse
```

```
import logging
import base64
import json
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def analyze_image(function_name, image):
    """Analyzes an image with an AWS Lambda function.
    :param image: The image that you want to analyze.
    :return The status and classification result for
    the image analysis.
    """

    lambda_client = boto3.client('lambda')

    lambda_payload = {}

    if image.startswith('s3://'):
        logger.info("Analyzing image from S3 bucket: %s", image)
        bucket, key = image.replace("s3://", "").split("/", 1)
        s3_object = {
            'Bucket': bucket,
            'Name': key
        }
        lambda_payload = {"S3Object": s3_object}

    # Call the lambda function with the image.
    else:
        with open(image, 'rb') as image_file:
            logger.info("Analyzing local image image: %s ", image)
            image_bytes = image_file.read()
            data = base64.b64encode(image_bytes).decode("utf8")

            lambda_payload = {"image": data}

    response = lambda_client.invoke(FunctionName=function_name,
                                    Payload=json.dumps(lambda_payload))

    return json.loads(response['Payload'].read().decode())
```

```
def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "function", help="The name of the AWS Lambda function that you want " \
        "to use to analyze the image.")
    parser.add_argument(
        "image", help="The local image that you want to analyze.")


def main():
    """
    Entrypoint for script.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Get analysis results.
        result = analyze_image(args.function, args.image)
        status = result['statusCode']

        if status == 200:
            labels = result['body']
            labels = json.loads(labels)
            print(f"There are {len(labels)} labels in the image.")
            for custom_label in labels:
                confidence = int(round(custom_label['Confidence'], 0))
                print(
                    f"Label: {custom_label['Name']}: Confidence: {confidence}%")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")
```

```
    except ClientError as error:
        logging.error(error)
        print(error)


if __name__ == "__main__":
    main()
```

5. Execute o código. Para o argumento da linha de comando, forneça o nome da função do Lambda e a imagem que você deseja analisar. É possível fornecer um caminho para uma imagem local ou o caminho do S3 para uma imagem armazenada em um bucket do Amazon S3. Por exemplo:

```
python client.py function_name s3://bucket/path/image.jpg
```

   Se a imagem estiver em um bucket do Amazon S3, certifique-se de que seja o mesmo bucket que você especificou na etapa 15 de Etapa 1: criar uma AWS Lambda função (console).

   Se for bem-sucedida, a saída será uma lista de rótulos encontrados na imagem. Se nenhum rótulo for retornado, considere reduzir o valor de confiança que você definiu na etapa 7 do Etapa 1: criar uma AWS Lambda função (console).

6. Se você tiver concluído a função do Lambda e o modelo não for usado por outras aplicações, interrompa o modelo. Inicie o modelo na próxima vez que quiser usar a função do Lambda.

Segurança

É possível proteger o gerenciamento de seus projetos, modelos e a operação `DetectCustomLabels` que seus clientes usam para detectar rótulos personalizados.

Para obter mais informações sobre como proteger o Amazon Rekognition, consulte Amazon Rekognition Security.

Como proteger projetos do Amazon Rekognition Custom Labels

É possível proteger seus projetos do Amazon Rekognition Custom Labels ao especificar as permissões em nível de recurso que são especificadas nas políticas baseadas em identidade. Para obter mais informações, consulte Políticas baseadas em identidade e políticas baseadas em recurso.

Os recursos do Amazon Rekognition Custom Labels que podem ser protegidos são:

Recurso	Formato de nome do recurso da Amazon
Projeto	arn:aws:rekognition: *:project/ /datetime *project_name*
Modelo	arn:aws:rekognition: *:project/ /version/ / datetime *project_name* *name*

O seguinte exemplo de política mostra como conceder permissão a uma identidade:

- Descreve todos os projetos.
- Crie, inicie, pare e use um modelo específico para inferência.
- Crie um projeto. Crie e descreva um modelo específico.
- Negue a criação de um projeto específico.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllResources",
            "Effect": "Allow",
```

```
            "Action": "rekognition:DescribeProjects",
            "Resource": "*"
        },
        {
            "Sid": "SpecificProjectVersion",
            "Effect": "Allow",
            "Action": [
                "rekognition:StopProjectVersion",
                "rekognition:StartProjectVersion",
                "rekognition:DetectCustomLabels",
                "rekognition:CreateProjectVersion"
            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
        },
        {
            "Sid": "SpecificProject",
            "Effect": "Allow",
            "Action": [
                "rekognition:CreateProject",
                "rekognition:DescribeProjectVersions",
                "rekognition:CreateProjectVersion"
            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/*"
        },
        {
            "Sid": "ExplicitDenyCreateProject",
            "Effect": "Deny",
            "Action": [
                "rekognition:CreateProject"
            ],
            "Resource": ["arn:aws:rekognition:*:*:project/SampleProject/*"]
        }
    ]
}
```

Protegendo DetectCustomLabels

A identidade usada para detectar etiquetas personalizadas pode ser diferente da identidade que gerencia os modelos do Amazon Rekognition Custom Labels.

É possível proteger o acesso de uma identidade para `DetectCustomLabels` aplicando uma política à identidade. O seguinte exemplo restringe o acesso a `DetectCustomLabels` somente a um

modelo específico. A identidade não tem acesso a nenhuma das outras operações do Amazon Rekognition.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:DetectCustomLabels"


            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
        }
    ]
}
```

Políticas gerenciadas pela AWS

Fornecemos a política `AmazonRekognitionCustomLabelsFullAccess` AWS gerenciada que você pode usar para controlar o acesso às etiquetas personalizadas do Amazon Rekognition. Para obter mais informações, consulte a política gerenciada da AWS: AmazonRekognitionCustomLabelsFullAccess.

Diretrizes e cotas no Amazon Rekognition Custom Labels

As seções a seguir fornecem diretrizes e cotas ao usar o Amazon Rekognition Custom Labels.

Regiões compatíveis

Para obter uma lista das AWS regiões em que os rótulos personalizados do Amazon Rekognition estão disponíveis, consulte Regiões e endpoints da AWS na Referência geral da Amazon Web Services.

Cotas

A seguir, veja uma lista de limites do Amazon Rekognition Custom Labels. Para obter informações sobre os limites que podem ser alterados, consulte Limites de serviço da AWS. Para alterar um limite, consulte Criar caso.

Treinamento

- Os formatos de arquivo compatíveis são os formatos de imagem PNG e JPEG.
- O número máximo de conjuntos de dados de treinamento em uma versão de um modelo é 1.
- O tamanho máximo do arquivo de manifesto do conjunto de dados é de 1 GB.
- O número mínimo de rótulos exclusivos por conjunto de dados de objetos, cenas e conceitos (classificação) é 2.
- O número mínimo de rótulos exclusivos por conjunto de dados de localização do objeto (detecção) é 1.
- O número máximo de rótulos exclusivos por manifesto é 250.
- O número mínimo de imagens por rótulo é 1.
- O número máximo de imagens por conjunto de dados de localização do objeto (detecção) é de 250 mil.

  O limite para as AWS regiões Ásia-Pacífico (Mumbai) e Europa (Londres) é de 28.000 imagens.
- O número máximo de imagens por conjunto de dados de objetos, cenas e conceitos (classificação) é 500 mil. O padrão é 250 mil. Para solicitar um aumento, consulte Criar caso.

  O limite para as AWS regiões Ásia-Pacífico (Mumbai) e Europa (Londres) é de 28.000 imagens. Não é possível solicitar um aumento de limite.

- O número máximo de rótulos por imagem é 50.
- O número mínimo de caixas delimitadoras em uma imagem é 0.
- O número máximo de caixas delimitadoras em uma imagem é 50.
- A dimensão mínima da imagem do arquivo de imagem em um bucket do Amazon S3 é de 64 pixels x 64 pixels.
- A dimensão máxima da imagem do arquivo de imagem em um bucket do Amazon S3 é de 4.096 pixels x 4.096 pixels.
- O tamanho máximo do arquivo para uma imagem em um bucket do Amazon S3 é 15 MB.
- A proporção máxima da imagem é 20:1.

Teste

- O número máximo de conjuntos de dados de teste em uma versão de um modelo é 1.
- O tamanho máximo do arquivo de manifesto do conjunto de dados é de 1 GB.
- O número mínimo de rótulos exclusivos por conjunto de dados de objetos, cenas e conceitos (classificação) é 2.
- O número mínimo de rótulos exclusivos por conjunto de dados de localização do objeto (detecção) é 1.
- O número máximo de rótulos exclusivos por conjunto de dados é 250.
- O número mínimo de imagens por rótulo é 0.
- O número máximo de imagens por rótulo é 1.000.
- O número máximo de imagens por conjunto de dados de localização do objeto (detecção) é de 250 mil.

  O limite para as AWS regiões Ásia-Pacífico (Mumbai) e Europa (Londres) é de 7.000 imagens.
- O número máximo de imagens por conjunto de dados de objetos, cenas e conceitos (classificação) é 500 mil. O padrão é 250 mil. Para solicitar um aumento, consulte Criar caso.

  O limite para as AWS regiões Ásia-Pacífico (Mumbai) e Europa (Londres) é de 7.000 imagens. Não é possível solicitar um aumento de limite.
- O número mínimo de rótulos por imagem por manifesto é 0.
- O número máximo de rótulos por imagem por manifesto é 50.
- O número mínimo de caixas delimitadoras em uma imagem por manifesto é 0.

- O número máximo de caixas delimitadoras em uma imagem por manifesto é 50.
- A dimensão mínima da imagem de um arquivo de imagem em um bucket do Amazon S3 é de 64 pixels x 64 pixels.
- A dimensão máxima da imagem de um arquivo de imagem em um bucket do Amazon S3 é de 4.096 pixels x 4.096 pixels.
- O tamanho máximo do arquivo para uma imagem em um bucket do Amazon S3 é 15 MB.
- Os formatos de arquivo compatíveis são os formatos de imagem PNG e JPEG.
- A proporção máxima da imagem é 20:1.

Detecção

- O tamanho máximo das imagens passadas como bytes brutos é de 4 MB.
- O tamanho máximo do arquivo para uma imagem em um bucket do Amazon S3 é 15 MB.
- A dimensão mínima da imagem de um arquivo de imagem de entrada (armazenado em um bucket do Amazon S3 ou fornecido como bytes de imagem) é de 64 pixels x 64 pixels.
- A dimensão máxima da imagem de um arquivo de imagem de entrada (armazenado em um Amazon S3 ou fornecido como bytes de imagem) é de 4.096 pixels x 4.096 pixels.
- Os formatos de arquivo compatíveis são os formatos de imagem PNG e JPEG.
- A proporção máxima da imagem é 20:1.

Cópia do modelo

- O número máximo de políticas de projeto que podem ser anexadas a um projeto é 5.
- O número máximo de trabalhos de cópia simultânea em um destino é 5.

Referência da API do Amazon Rekognition Custom Labels

A API do Amazon Rekognition Custom Labels está documentada como parte do conteúdo de referência da API Amazon Rekognition. Esta é uma lista das operações da API do Amazon Rekognition Custom Labels com links para o tópico de referência apropriado da API do Amazon Rekognition. Além disso, os links de referência da API contidos neste documento levam ao tópico de referência da API do Guia do Desenvolvedor Amazon Rekognition adequado. Para obter informações sobre como usar a API, consulte

Esta seção fornece uma visão geral do fluxo de trabalho para treinar e usar um modelo de etiquetas personalizadas do Amazon Rekognition com o console e o SDK. AWS

> **Note**
>
> O Amazon Rekognition Custom Labels agora gerencia conjuntos de dados dentro de um projeto. Você pode criar conjuntos de dados para seus projetos com o console e com o AWS

SDK. Se você já usou o Amazon Rekognition Custom Labels, talvez seja necessário associar

seus conjuntos de dados antigos a um novo projeto. Para ter mais informações, consulte Etapa 6: (opcional) associe conjuntos de dados anteriores com novos projetos

Tópicos

- Decida o tipo do seu modelo
- Criar um modelo
- Melhore seu modelo
- Executar seu modelo
- Analisar uma imagem
- Interrompa seu modelo

Decida o tipo do seu modelo

Primeiro, decida qual tipo de modelo deseja treinar, o que depende de suas metas comerciais. Por exemplo, é possível treinar um modelo para encontrar seu logotipo em publicações nas redes sociais, identificar seus produtos nas prateleiras das lojas ou classificar peças de máquinas em uma linha de montagem.

O Amazon Rekognition Custom Labels pode treinar os seguintes tipos de modelo:

- Encontre objetos, cenas e conceitos
- Encontre localizações de objetos
- Encontre a localização das marcas

Para ajudar a decidir qual tipo de modelo treinar, o Amazon Rekognition Custom Labels fornece exemplos de projetos que podem ser usados. Para obter mais informações, consulte Conceitos básicos do Amazon Rekognition Custom Labels.

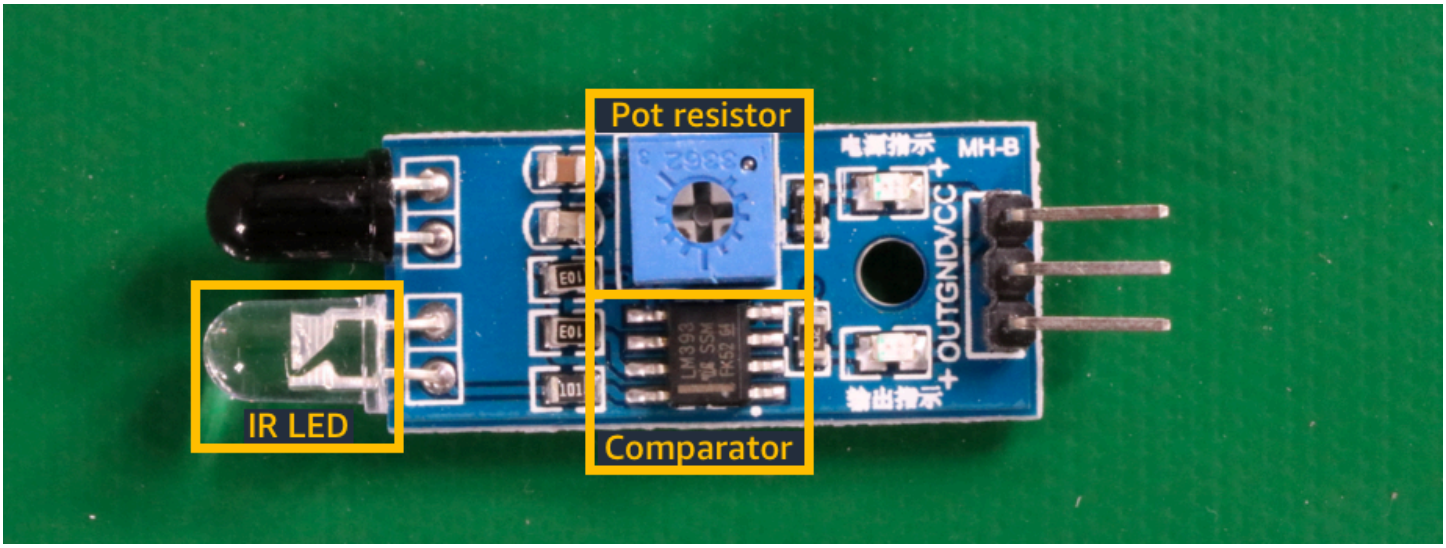
Encontre objetos, cenas e conceitos

O modelo prevê classificações para os objetos, cenas e conceitos associados a uma imagem inteira. Por exemplo, é possível treinar um modelo que determine se uma imagem contém uma atração turística ou não. Para obter um objeto de exemplo, consulte Classificação de imagens. A imagem a seguir de um lago é um exemplo do tipo de imagem em que você pode reconhecer objetos, cenas e conceitos.

Como alternativa, é possível treinar um modelo que categorize as imagens em várias categorias. Por exemplo, a imagem anterior pode ter categorias como cor do céu, reflexo ou lago. Para obter um objeto de exemplo, consulte Classificação de imagens com vários rótulos.

Encontre localizações de objetos

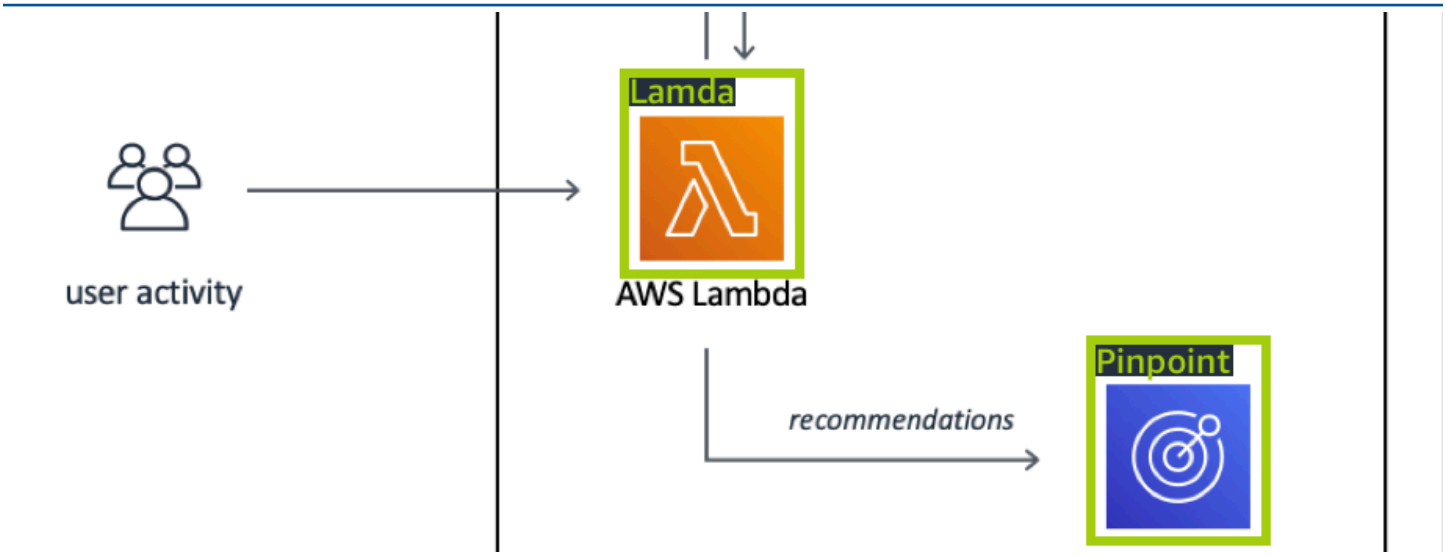
O modelo prevê a localização de um objeto em uma imagem. A previsão inclui informações da caixa delimitadora para a localização do objeto e um rótulo que identifica o objeto dentro da caixa delimitadora. Por exemplo, a imagem a seguir mostra as caixas delimitadoras em torno de várias partes de uma placa de circuito, como um comparador ou potenciômetro.



O projeto de exemplo Localização de objetos mostra como o Amazon Rekognition Custom Labels usa caixas delimitadoras rotuladas para treinar um modelo que encontra a localização dos objetos.

Encontre a localização das marcas

O Amazon Rekognition Custom Labels pode treinar um modelo que encontra a localização de marcas, como logotipos, em uma imagem. A previsão inclui informações da caixa delimitadora para a localização da marca e um rótulo que identifica o objeto dentro da caixa delimitadora. Para obter um objeto de exemplo, consulte Detecção de marca. A imagem a seguir é um exemplo de algumas das marcas que o modelo pode detectar.



Criar um modelo

As etapas para criar um modelo são as seguintes: criar um projeto, criar conjuntos de dados de treinamento e teste, e treinar o modelo.

Criar um projeto

Um projeto é um grupo de recursos necessários para criar e gerenciar versões de um modelo do Amazon Rekognition Custom Labels. Um projeto gerencia o seguinte:

- Conjuntos de dados: as imagens e os rótulos de imagem usados para treinar um modelo. Um projeto tem um conjunto de dados de treinamento e um conjunto de dados de teste.

- Modelos: o software que você treina para encontrar conceitos, cenas e objetos exclusivos da sua empresa. É possível ter várias versões de um modelo em um projeto.

É recomendável usar um projeto para um único caso de uso, como descobrir peças da placa de circuito em uma placa de circuito.

Você pode criar um projeto com o console Amazon Rekognition Custom Labels e com a API. CreateProject Para obter mais informações, consulte Como criar um projeto.

Crie conjuntos de dados de treinamento e teste

Um conjunto de dados é um conjunto de imagens e rótulos que descrevem essas imagens. No projeto, é criado um conjunto de dados de treinamento e um conjunto de dados de teste que o Amazon Rekognition Custom Labels usa para treinar e testar seu modelo.

Um rótulo identifica um objeto, cena, conceito ou caixa delimitadora ao redor de um objeto em uma imagem. Os rótulos são atribuídos a uma imagem inteira (nível de imagem) ou são atribuídos a uma caixa delimitadora que circunda um objeto em uma imagem.

> **Important**
>
> A forma como você rotula as imagens em seus conjuntos de dados determina o tipo de modelo que o Amazon Rekognition Custom Labels cria. Por exemplo, para treinar um modelo que encontre objetos, cenas e conceitos, você atribui rótulos de nível de imagem às imagens em seus conjuntos de dados de treinamento e teste. Para obter mais informações, consulte Como definir os conjuntos de dados.

As imagens devem estar nos formatos PNG e JPEG, e você deve seguir as recomendações das imagens de entrada. Para obter mais informações, consulte Como preparar imagens.

Crie conjuntos de dados de treinamento e teste (console)

É possível iniciar um projeto com um único conjunto de dados ou com conjuntos de dados de treinamento e teste separados. Se você começar com um único conjunto de dados, o Amazon Rekognition Custom Labels divide seu conjunto de dados durante o treinamento para criar um conjunto de dados de treinamento (80%) e um conjunto de dados de teste (20%) para seu projeto. Comece com um único conjunto de dados se quiser que o Amazon Rekognition Custom Labels decida quais imagens serão usadas para treinamento e teste. Para ter controle total sobre o treinamento, teste e ajuste de desempenho, recomendamos que você inicie seu projeto com os conjuntos de dados de treinamento e teste separados.

Para criar os conjuntos de dados para um projeto, importe as imagens das seguintes maneiras:

- Importe imagens do seu computador local.
- Importe imagens de um bucket do S3. O Amazon Rekognition Custom Labels podem rotular as imagens usando os nomes das pastas que contêm as imagens.
- Importe um arquivo de manifesto do Amazon SageMaker AI Ground Truth.
- Copie um conjunto de dados existente do Amazon Rekognition Custom Labels.

Para obter mais informações, consulte Como criar conjuntos de dados de treinamento e teste com imagens.

Dependendo de onde você importa suas imagens, elas podem não estar rotuladas. Por exemplo, imagens importadas de um computador local não estão rotuladas. As imagens importadas de um arquivo de manifesto do Amazon SageMaker AI Ground Truth são rotuladas. É possível usar o console do Amazon Rekognition Custom Labels para adicionar, alterar e atribuir rótulos. Para obter mais informações, consulte Rotulagem de imagens.

Para criar seus conjuntos de dados de treinamento e teste com o console, consulte Como criar conjuntos de dados de treinamento e teste com imagens. Para ver um tutorial que inclui a criação de conjuntos de dados de treinamento e teste, consulte Classificar imagens.

Crie conjuntos de dados de treinamento e teste (SDK)

Para criar seus conjuntos de dados de treinamento e teste, use a API `CreateDataset`. É possível criar um conjunto de dados usando um arquivo de manifesto no formato Amazon Sagemaker ou

copiando um conjunto de dados existente do Amazon Rekognition Custom Labels. Para obter mais informações, consulte Crie conjuntos de dados de treinamento e teste (SDK) . Se necessário, é possível criar o seu próprio arquivo de manifesto. Para obter mais informações, consulte the section called “Criar um arquivo de manifesto”.

Treinar seu modelo

Treine seu modelo com o conjunto de dados de treinamento. Uma nova versão de um modelo é criada toda vez que ele é treinado. Durante o treinamento, o Amazon Rekognition Custom Labels testa o desempenho do seu modelo treinado. É possível usar os resultados para avaliar e melhorar seu modelo. O treinamento demora para ser concluído. Só há uma cobrança por um treinamento de modelo com êxito. Para obter mais informações, consulte Como treinar um modelo do Amazon Rekognition Custom Labels. Se o treinamento do modelo falhar, o Amazon Rekognition Custom Labels fornecerá informações de depuração que podem ser usadas. Para obter mais informações, consulte Como depurar um treinamento de modelo em falha.

Treinar seu modelo (console)

Para treinar seu modelo com o console, consulte Como treinar um modelo (console).

Treinando um modelo (SDK)

Você treina um modelo de etiquetas personalizadas do Amazon Rekognition ligando para. CreateProjectVersion Para obter mais informações, consulte Treinando um modelo (SDK).

Melhore seu modelo

Durante o teste, o Amazon Rekognition Custom Labels cria métricas de avaliação que podem ser usadas para melhorar seu modelo treinado.

Avalie seu modelo

Avalie o desempenho do seu modelo usando as métricas de desempenho criadas durante o teste. As métricas de desempenho, como F1, precisão e recall, permitem que você entenda o desempenho do seu modelo treinado e decida se está pronto para usá-lo na produção. Para obter mais informações, consulte Métricas para avaliar seu modelo.

Avaliar um modelo (console)

Para visualizar as métricas de desempenho, consulte Como acessar as métricas de avaliação (console).

Avaliar um modelo (SDK)

Para obter métricas de desempenho, você liga DescribeProjectVersionspara obter os resultados do teste. Para obter mais informações, consulte Como acessar as métricas de avaliação (SDK) do Amazon Rekognition Custom Labels. Os resultados do teste incluem métricas não disponíveis no console, como uma matriz de confusão para resultados de classificação. Os resultados do teste são retornados nos seguintes formatos:

- Pontuação F1: um valor único que representa o desempenho geral de precisão e recall do modelo. Para obter mais informações, consulte F1.

- Localização do arquivo de resumo: i resumo do teste inclui métricas de avaliação agregadas para todo o conjunto de dados de teste e métricas para cada rótulo individual. `DescribeProjectVersions` retorna o bucket do S3 e a localização da pasta do arquivo de resumo. Para obter mais informações, consulte Acessar o arquivo de resumo do modelo.
- Localização do snapshot do manifesto de avaliação: o snapshot contém detalhes sobre os resultados do teste, incluindo as classificações de confiança e os resultados dos testes de classificação binária, como falsos positivos. `DescribeProjectVersions` retorna o bucket do S3 e a localização da pasta dos arquivos de snapshot. Para obter mais informações, consulte Interpretar o snapshot do manifesto de avaliação.

Melhore seu modelo

Se forem necessárias melhorias, é possível adicionar mais imagens de treinamento ou melhorar a rotulagem do conjunto de dados. Para obter mais informações, consulte Como melhorar um modelo do Amazon Rekognition Custom Labels. Também é possível dar feedback sobre as previsões que seu modelo faz e usá-lo para fazer melhorias em seu modelo. Para obter mais informações, consulte Melhorar um modelo com Model Feedback.

Melhore seu modelo (console)

Para adicionar imagens a um conjunto de dados, consulte Como adicionar mais imagens a um conjunto de dados. Para adicionar ou alterar rótulos, consulte the section called “Rotulagem de imagens”.

Para treinar seu modelo novamente, consulte Como treinar um modelo (console).

Melhore seu modelo (SDK)

Para adicionar imagens a um conjunto de dados ou alterar a rotulagem de uma imagem, use a API `UpdateDatasetEntries`. `UpdateDatasetEntries` atualiza ou adiciona linhas JSON a um arquivo de manifesto. Cada linha JSON contém informações para uma única imagem, como rótulos atribuídos ou informações da caixa delimitadora. Para obter mais informações, consulte Como adicionar mais imagens (SDK). Para visualizar as entradas em um conjunto de dados, use a API `ListDatasetEntries`.

Para treinar seu modelo novamente, consulte Como treinar um modelo (SDK).

Executar seu modelo

Antes de usar seu modelo, você inicia o modelo usando o console do Amazon Rekognition Custom Labels ou a API `StartProjectVersion`. Há uma cobrança pela quantidade de tempo que o modelo é executado. Para obter mais informações, consulte Como executar um modelo treinado.

Executar seu modelo (console)

Para iniciar o seu modelo usando o console, consulte Como iniciar um modelo do Amazon Rekognition Custom Labels (console).

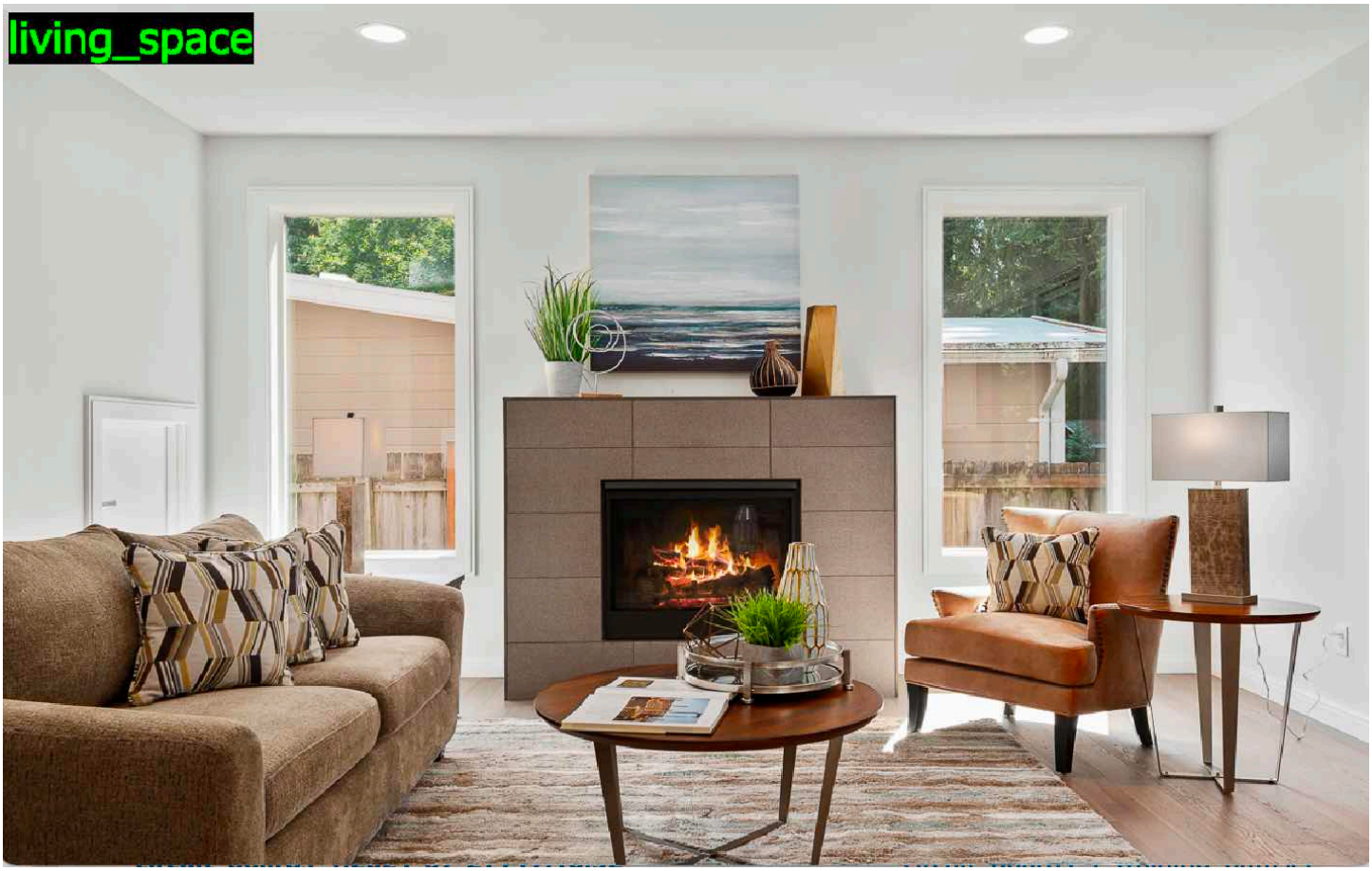
Executar seu modelo

Você começa a ligar para sua modelo StartProjectVersion. Para obter mais informações, consulte Como iniciar um modelo do Amazon Rekognition Custom Labels (SDK).

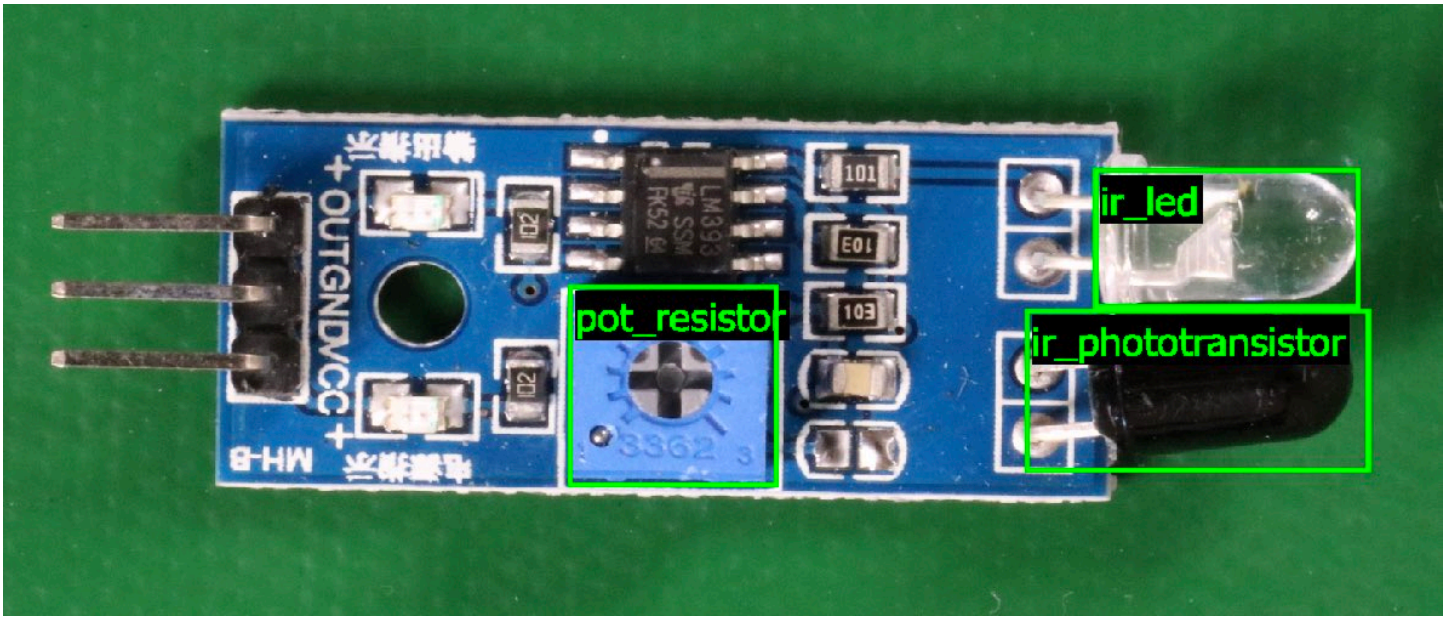
Analisar uma imagem

Para analisar uma imagem com seu modelo, você usa a API `DetectCustomLabels`. É possível especificar uma imagem local ou uma imagem armazenada em um bucket do S3. A operação também requer o nome do recurso da Amazon (ARN) do modelo que deseja utilizar.

Se seu modelo encontrar objetos, cenas e conceitos, a resposta incluirá uma lista de rótulos em nível de imagem encontrados na imagem. Por exemplo, a imagem a seguir mostra os rótulos no nível da imagem encontrados usando o projeto de exemplo Cômodos.



Se o modelo encontrar a localização dos objetos, a resposta incluirá uma lista de caixas delimitadoras rotuladas encontradas na imagem. Uma caixa delimitadora representa a localização de um objeto em uma imagem. É possível usar as informações da caixa delimitadora para desenhar uma caixa delimitadora ao redor de um objeto. Por exemplo, a imagem a seguir mostra caixas delimitadoras ao redor das partes da placa de circuito encontradas usando o projeto de exemplo de Placas de circuito.



Para obter mais informações, consulte Como analisar uma imagem com um modelo treinado.

Interrompa seu modelo

Há uma cobrança pelo tempo que o modelo está em execução. Se não estiver mais usando seu modelo, interrompa o modelo usando o console do Amazon Rekognition Custom Labels ou usando a API `StopProjectVersion`. Para obter mais informações, consulte Como interromper um modelo do Amazon Rekognition Custom Labels.

Interrompa seu modelo (console)

Para interromper a execução de um modelo com o console, consulte Como interromper um modelo do Amazon Rekognition Custom Labels (console).

Interrompa seu modelo (SDK)

Para interromper a execução de um modelo, ligue StopProjectVersion. Para obter mais informações, consulte Como interromper um modelo do Amazon Rekognition Custom Labels (SDK).

.

Como treinar seu modelo

Projetos

- CreateProject— Cria seu projeto Amazon Rekognition Custom Labels, que é um agrupamento lógico de recursos (imagens, rótulos, modelos) e operações (treinamento, avaliação e detecção).
- DeleteProject— Exclui um projeto Amazon Rekognition Custom Labels.
- DescribeProjects— Retorna uma lista de todos os seus projetos de etiquetas personalizadas do Amazon Rekognition.

Políticas do projeto

- PutProjectPolicy— Anexa uma política de projeto a um projeto Amazon Rekognition Custom Labels em uma conta confiável. AWS
- ListProjectPolicies— Retorna uma lista das políticas do projeto anexadas a um projeto.
- DeleteProjectPolicy— Exclui uma política de projeto existente.

Conjuntos de dados

- CreateDataset— Cria um conjunto de dados Amazon Rekognition Custom Labels.
- DeleteDataset— Exclui um conjunto de dados Amazon Rekognition Custom Labels.
- DescribeDataset— Descreve um conjunto de dados Amazon Rekognition Custom Labels.
- DistributeDatasetEntries— Distribui as entradas (imagens) em um conjunto de dados de treinamento entre o conjunto de dados de treinamento e o conjunto de dados de teste de um projeto.
- ListDatasetEntries— Retorna uma lista de entradas (imagens) em um conjunto de dados Amazon Rekognition Custom Labels.
- ListDatasetLabels— Retorna uma lista de etiquetas atribuídas a um conjunto de dados Amazon Rekognition Custom Labels.
- UpdateDatasetEntries— Adiciona ou atualiza entradas (imagens) em um conjunto de dados Amazon Rekognition Custom Labels.

Modelos da

- CreateProjectVersion— Treina seu modelo de etiquetas personalizadas do Amazon Rekognition.
- CopyProjectVersion— Copia seu modelo de etiquetas personalizadas do Amazon Rekognition.
- DeleteProjectVersion— Exclui um modelo de etiquetas personalizadas do Amazon Rekognition.
- DescribeProjectVersions— Retorna uma lista de todos os modelos de etiquetas personalizadas do Amazon Rekognition em um projeto específico.

Tags

- TagResource— Adiciona uma ou mais tags de valor-chave a um modelo de etiquetas personalizadas do Amazon Rekognition.
- UntagResource— Remove uma ou mais tags de um modelo de etiquetas personalizadas do Amazon Rekognition.

Como usar seu modelo

- DetectCustomLabels— Analisa uma imagem com seu modelo de etiquetas personalizadas.
- StartProjectVersion— Inicia seu modelo de etiquetas personalizadas.
- StopProjectVersion— Interrompe seu modelo de etiquetas personalizadas.

Histórico da documentação do Amazon Rekognition Custom Labels

A tabela a seguir descreve as mudanças importantes em cada versão do Guia do desenvolvedor do Amazon Rekognition Custom Labels. Para receber notificações sobre atualizações dessa documentação, é possível inscrever-se em um feed RSS.

- Atualização mais recente da documentação: 19 de abril de 2023

Alteração	Descrição	Data
Foi adicionado o tópico de duração do modelo	Mostra como obter o número de horas de execução e as unidades de inferência usadas por um modelo. Para obter mais informações, consulte Como relatar a duração da execução e das unidades de inferência usadas.	19 de abril de 2023
Conteúdo do conjunto de dados reorganizado	O conteúdo da criação do arquivo de manifesto foi movido para Arquivo de manifesto. Os tópicos de conversão de conjuntos de dados foram movidos para Conversão de outros formatos de conjunto de dados em um arquivo de manifesto.	20 de fevereiro de 2023
Atualizou a orientação do IAM para AWS WAF	Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações,	15 de fevereiro de 2023

	consulte Práticas recomendadas de segurança no IAM.	
Visualizar a matriz de confusão de um modelo de classificação	O console do Amazon Rekognition Custom Labels não mostra a matriz de confusão de um modelo de classificação. Em vez disso, você pode usar o AWS SDK para obter e mostrar uma matriz de confusão. Para obter mais informações, consulte Como visualizar a matriz de confusão de um modelo.	4 de janeiro de 2023
Exemplo de função do Lambda atualizado	O exemplo da função do Lambda agora mostra como analisar imagens passadas de um arquivo local ou de um bucket do Amazon S3. Para obter mais informações, consulte Análise de imagens com uma função do AWS Lambda.	2 de dezembro de 2022
O Amazon Rekognition Custom Labels agora podem copiar modelos treinados	Agora você pode copiar um modelo treinado de uma AWS AWS conta para outra na mesma AWS região. Para obter mais informações, consulte Como copiar um modelo do Amazon Rekognition Custom Labels (SDK).	16 de agosto de 2022

Agora o Amazon Rekognition Custom Labels pode escalar automaticamente as unidades de inferência.	Para ajudar com picos de demanda, o Amazon Rekogniti on Custom Labels agora pode escalar automatic amente o número de unidades de inferência que seu modelo usa. Para obter mais informações, consulte Como executar modelos treinados do Amazon Rekognition Custom Labels.	16 de agosto de 2022
Criar um arquivo de manifesto de um arquivo CSV	Agora é possível simplificar a criação de um arquivo de manifesto usando um script que lê as informações de classificação de imagens de um arquivo CSV. Para obter mais informações, consulte Como criar um arquivo de manifesto de um arquivo CSV.	2 de fevereiro de 2022
O Amazon Rekognition Custom Labels agora gerencia conjuntos de dados com projetos	É possível usar projetos para gerenciar os conjuntos de dados de treinamento e teste usados para criar um modelo. Para obter mais informações, consulte Noções básicas do Amazon Rekognition Custom Labels.	1º de novembro de 2021

O Amazon Rekognition Custom Labels é integrado ao AWS CloudFormation	Você pode usar AWS CloudFormation para provisionar e configurar projetos de etiquetas personalizadas do Amazon Rekognition. Para obter mais informações, consulte Criação de um projeto com AWS CloudFormation.	21 de outubro de 2021
Experiência de introdução atualizada	Agora o console do Amazon Rekognition Custom Labels inclui tutoriais em vídeo e exemplos de projetos. Para obter mais informações, consulte Conceitos básicos do Amazon Rekognition Custom Labels.	22 de julho de 2021
Informações atualizadas sobre limites e uso de métricas	Informações sobre como definir um valor limite desejado usando o parâmetro de entrada `MinConfidence` para `DetectCustomLabels`. Para obter mais informações, consulte Análise de uma imagem com um modelo treinado.	8 de junho de 2021
AWS KMS key Suporte adicionado	Agora é possível usar sua própria chave KMS para criptografar suas imagens de treinamento e teste. Para obter mais informações, consulte Como treinar um modelo.	19 de maio de 2021

Foram adicionadas tags	O Amazon Rekognition Custom Labels agora é compatível com marcação. É possível usar tags para identificar, organizar, pesquisar e filtrar seus modelos do Amazon Rekogniti on Custom Labels. Para obter mais informações, consulte Como atribuir tags em um modelo.	25 de março de 2021
Tópico de configuração atualizado	Informações de configura ção atualizadas sobre como criptografar arquivos de treinamento. Para obter mais informações, consulte Etapa 5: (opcional) Como criptografar arquivos de treinamento.	18 de março de 2021
Tópico de cópia do conjunto de dados adicionado	Informações sobre como copiar um conjunto de dados para uma AWS região diferente. Para obter mais informações, consulte Cópia de um conjunto de dados para uma AWS região diferente.	5 de março de 2021

Foi adicionado o tópico de transformação de manifesto GroundTruth multirótulo da Amazon SageMaker AI	Informações sobre como transformar um manifesto no formato de GroundTruth vários rótulos do Amazon SageMaker AI em um arquivo de manifesto no formato Amazon Rekognition Custom Labels. Para obter mais informações, consulte Transformação de arquivos de manifesto do SageMaker AI Ground Truth com vários rótulos.	22 de fevereiro de 2021
Informações de depuração adicionadas para treinamento de modelos	Agora é possível usar manifestos de resultados de validação para obter informações detalhadas de depuração sobre erros de treinamento de modelos. Para obter mais informações, consulte Como depurar um treinamento de modelo com falha.	8 de outubro de 2020
Informações e exemplos de transformação COCO adicionados	Informações sobre como transformar um conjunto de dados no formato de detecção de objetos COCO em um arquivo de manifesto Amazon Rekognition Custom Labels. Para obter mais informações, consulte Como transformar conjuntos de dados COCO.	2 de setembro de 2020

O Amazon Rekognition Custom Labels agora é compatível com treinamentos com um único objeto	Para criar um modelo do Amazon Rekognition Custom Labels que encontre a localização de um único objeto, agora é possível criar um conjunto de dados que requer apenas um rótulo. Para obter mais informaçõ es, consulte Como desenhar caixas delimitadoras.	25 de junho de 2020
Operações de exclusão de projeto e modelo adicionadas	Agora é possível excluir projetos e modelos do Amazon Rekognition Custom Labels com o console e com a API. Para obter mais informações, consulte Como excluir um modelo do Amazon Rekognition Custom Labels e Como excluir um projeto do Amazon Rekognition Custom Labels	1 de abril de 2020
Foram adicionados exemplos em Java	Foram adicionados exemplos em Java que abrangem criação de projetos, treinamen to de modelos, execução de modelos e análise de imagens.	13 de dezembro de 2019
Novo atributo e guia	Esta é a versão inicial do atributo do Amazon Rekogniti on Custom Labels e do Amazon Rekognition Custom Labels Developer Guide.	3 de dezembro de 2019

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.