aws

Guia do Desenvolvedor

Amazon Lookout for Vision

Amazon Lookout for Vision: Guia do Desenvolvedor

Table of Contents

O que é o Amazon Lookout for Vision?

Você pode usar o Amazon Lookout for Vision para encontrar defeitos visuais em produtos industriais, com precisão e em grande escala. Ele usa visão computacional para identificar componentes ausentes em um produto industrial, danos em veículos ou estruturas, irregularidades nas linhas de produção e até mesmo defeitos minúsculos em pastilhas de silício — ou em qualquer outro item físico em que a qualidade seja importante, como a falta de um capacitor nas placas de circuito impresso.

Benefícios principais

O Amazon Lookout for Vision oferece os seguintes benefícios:

- Melhore os processos de forma rápida e eficiente — Você pode usar o Amazon Lookout for Vision para implementar a inspeção baseada em visão computacional em processos industriais de forma rápida e eficiente, em grande escala. Você pode fornecer apenas 30 imagens básicas boas e o Lookout for Vision pode criar automaticamente um modelo de ML personalizado para detecção de defeitos. Em seguida, você pode processar imagens de câmeras IP, em lote ou em tempo real, para identificar anomalias com rapidez e precisão, como amassados, rachaduras e arranhões.
- Aumente a qualidade da produção rapidamente — Com o Amazon Lookout for Vision, você pode reduzir defeitos nos processos de produção, em tempo real. Ele identifica e relata anomalias visuais em um painel para que você possa agir rapidamente para impedir a ocorrência de mais defeitos, aumentando a qualidade da produção e reduzindo os custos.
- Reduza os custos operacionais — O Amazon Lookout for Vision relata tendências em seus dados de inspeção visual, como identificar processos com a maior taxa de defeitos ou sinalizar variações recentes nos defeitos. Usando essas informações, você pode determinar se deve programar a manutenção na linha de processo ou redirecionar a produção para outra máquina antes que ocorra um tempo de inatividade caro e não planejado.

Você é um usuário final iniciante do Amazon Lookout for Vision?

Se for o primeiro usuário do Amazon Lookout for Vision, recomendamos que leia as seguintes seções na ordem:

1. Configuração do Amazon Lookout for Vision - Nessa seção, você define os detalhes da sua conta.

2. Conceitos básicos do Amazon Lookout for Vision— Nesta seção, você aprenderá a criar seu primeiro modelo Amazon Lookout for Vision.

Configuração do Amazon Lookout for Vision

Nesta seção, você se cadastra em uma conta da AWS e configura o Amazon Lookout for Vision.

Para obter informações sobre as AWS regiões que oferecem suporte ao Amazon Lookout for Vision, consulte Amazon Lookout for Vision Endpoints and Quotas.

Tópicos

Etapa 1: criar uma AWS conta

Nesta etapa, você se inscreve em uma AWS conta e cria um usuário administrativo.

Tópicos

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 https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e digitar 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, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando 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, digite sua senha.

   Para obter ajuda ao fazer login usando o usuário-raiz, consulte Signing in as the root user (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. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda o acesso administrativo para 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 seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  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 Create a permission set 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 Add groups no Guia do usuário do AWS IAM Identity Center .

Etapa 2: Configurar permissões

Para usar o Amazon Lookout for Vision, você precisa de permissões de acesso ao console do Lookout for Vision AWS , às operações do SDK e ao bucket do Amazon S3 que você usa para treinamento de modelos.

> **Note**
>
> Se você usa apenas operações do AWS SDK, pode usar políticas que tenham como escopo as operações do AWS SDK. Para ter mais informações, consulte Configurar permissões do SDK.

Tópicos

- Configurando o acesso ao console com políticas AWS gerenciadas
- Configuração das permissões do bucket do Amazon S3
- Como atribuir permissões

Configurando o acesso ao console com políticas AWS gerenciadas

Use as seguintes políticas AWS gerenciadas para aplicar as permissões de acesso apropriadas ao console do Amazon Lookout for Vision e às operações do SDK.

- AmazonLookoutVisionConsoleFullAccess— permite acesso total ao console do Amazon Lookout for Vision e às operações do SDK. Você precisa de permissões `AmazonLookoutVisionConsoleFullAccess` para criar o bucket do console. Para ter mais informações, consulte Etapa 3: Criar um bucket do console.
- AmazonLookoutVisionConsoleReadOnlyAccess— permite acesso somente de leitura ao console do Amazon Lookout for Vision e às operações do SDK.

Para atribuir permissões, consulte Como atribuir permissões.

Para obter informações sobre políticas AWS gerenciadas, consulte as políticas gerenciadas da AWS.

Configuração das permissões do bucket do Amazon S3

O Amazon Lookout for Vision usa um bucket do Amazon S3 para armazenar os seguintes arquivos:

- Imagens do conjunto de dados: imagens usadas para treinar um modelo. Para ter mais informações, consulte Criar um conjunto de dados.
- Arquivos de manifesto no formato Amazon SageMaker Ground Truth. Por exemplo, a saída do arquivo de manifesto do SageMaker GroundTruth trabalho. Para ter mais informações, consulte Criação de um conjunto de dados usando um arquivo de manifesto Amazon SageMaker Ground Truth.
- O resultado do treinamento do modelo.

Se você usa o console, o Lookout for Vision cria um bucket do Amazon S3 (bucket de console) para gerenciar seus projetos. As políticas gerenciadas `LookoutVisionConsoleReadOnlyAccess` e `LookoutVisionConsoleFullAccess` incluem permissões de acesso do Amazon S3 para o bucket do console.

Você pode usar o bucket do console para armazenar imagens do conjunto de dados e arquivos de manifesto no formato SageMaker Ground Truth. Como alternativa, você pode usar um bucket diferente do Amazon S3. O bucket deve pertencer à sua conta da AWS e deve estar localizado na AWS região em que você está usando o Lookout for Vision.

Para usar um bucket diferente, adicione a política a seguir ao usuário ou ao grupo desejado. Substitua `my-bucket` pelo nome do bucket desejado. Para obter informações sobre como adicionar políticas de IAM, consulte Criação de políticas de IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionS3BucketAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ]
        },
        {
            "Sid": "LookoutVisionS3ObjectAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/*"
            ]
        }
    ]
}
```

Para atribuir permissões, consulte Como atribuir permissões.

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 Criar 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: Criar um bucket do console

Para usar o console Amazon Lookout for Vision, você precisa de um bucket do Amazon S3 conhecido como bucket do console. O bucket do console armazena o seguinte:

- Imagens que você carrega em um conjunto de dados com o console.
- Resultados de treinamento para o treinamento de modelos que você começa com o console.
- Resultados da detecção da avaliação.
- Arquivos de manifesto temporários que o console cria quando você usa o console para criar um conjunto de dados rotulando automaticamente as imagens em um bucket do S3. O console não exclui os arquivos de manifesto.

Quando você abre o console do Amazon Lookout for Vision pela primeira vez em uma AWS nova região, o Lookout for Vision cria o bucket do console em seu nome. Anote o nome do bucket do console porque talvez você precise usar o nome do bucket em operações do AWS SDK ou tarefas do console, como criar um conjunto de dados.

Como alternativa, você pode criar o bucket do console usando o Amazon S3. Use essa abordagem se as políticas de bucket do Amazon S3 não permitirem que o console Amazon Lookout for Vision crie com sucesso o bucket do console. Por exemplo, uma política que proíbe a criação automática de um bucket do Amazon S3.

> **Note**
>
> Se você usa somente o AWS SDK e não o console do Lookout for Vision, não precisará criar o bucket do console. Você pode usar um bucket do S3 diferente com um nome de sua escolha.

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

Tópicos

- Criação do bucket do console com o console Amazon Lookout for Vision
- Criação do bucket do console com o Amazon S3
- Configurações do bucket do console

Criação do bucket do console com o console Amazon Lookout for Vision

Use o procedimento a seguir para criar o bucket do console para uma AWS região com o console Amazon Lookout for Vision. Para obter informações sobre as configurações do bucket do S3 que habilitamos, consulte Configurações do bucket do console.

Para criar o bucket do console usando o console Amazon Lookout for Vision

1. O usuário ou grupo que você está usando deve ter a permissão `AmazonLookoutVisionConsoleFullAccess`. Para ter mais informações, consulte Etapa 2: Configurar permissões.
2. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
3. Na barra de navegação, selecione Selecionar uma região. Em seguida, escolha a AWS região para a qual você deseja criar o bucket do console.
4. Escolha Comece a usar.
5. Se esta for a primeira vez que você abre o console na região atual da AWS, faça o seguinte na caixa de diálogo Configurar pela primeira vez:
    a. Copie o nome do bucket do Amazon S3 exibido. Você precisará dessas informações posteriormente.

b. Escolha Criar bucket do S3 para permitir que o Amazon Lookout for Vision crie o bucket do console em seu nome.

A caixa de diálogo First time set up não será exibida se o bucket do console para a AWS região atual já existir.

6. Feche a janela do navegador.

Criação do bucket do console com o Amazon S3

Você pode usar o Amazon S3 para criar o bucket do console. Você deve criar o bucket com o versionamento do Amazon S3 habilitado. Recomendamos que você use uma configuração de ciclo de vida do Amazon S3 para remover versões não atuais (anteriores) de um objeto e excluir carregamentos fracionados incompletos. Não recomendamos uma configuração de ciclo de vida que exclua as versões atuais de um objeto. Para obter informações sobre as configurações de bucket do S3 que habilitamos para buckets de console que você cria com o console Amazon Lookout for Vision, consulte Configurações do bucket do console.

1. Decida a AWS região na qual você deseja criar um bucket de console. Para obter informações sobre as regiões suportadas, consulte Amazon Lookout for Vision endpoints and quotas.
2. Crie um bucket usando as instruções do console do S3 em Criação de um bucket. Faça o seguinte:

   a. Para a etapa 3, especifique um nome de bucket que seja prefixado com `lookoutvision-`*`region`*`-`*`your-identifier`*. Altere `region` para o código de região que você escolheu na etapa anterior. Altere `your-identifier` para um identificador exclusivo de sua escolha. Por exemplo, `lookoutvision-us-east-1-my-console-bucket-1`.

   b. Para a etapa 4, escolha a AWS região que você deseja usar.
3. Habilite o versionamento para o bucket seguindo as instruções do console do S3 em Habilitar o versionamento em buckets.
4. (Opcional) Especifique uma configuração de ciclo de vida para o bucket seguindo as instruções do console S3 em Definir a configuração do ciclo de vida em um bucket. Faça o seguinte para remover versões não atuais (anteriores) de um objeto e excluir carregamentos incompletos de várias partes. Não é necessário executar as etapas 6, 8, 9, 10.

   a. Para a etapa 5, escolha Aplicar a todos os objetos no bucket.

b. Para a etapa 7, selecione Excluir permanentemente versões não atuais de objetos e Excluir marcadores de exclusão de objetos expirados ou carregamentos fracionados incompletos.

c. Para a etapa 11, insira os números de dias de espera antes de excluir as versões não atuais de um objeto.

d. Para a etapa 12, insira o número de dias de espera antes de excluir carregamentos incompletos de várias partes.

Configurações do bucket do console

Se você criar o bucket do console com o console Amazon Lookout for Vision, habilitaremos as seguintes configurações no bucket do console.

- Versionamento de objetos no bucket do console.
- Criptografia do lado do servidor de objetos no bucket do console.
- Uma configuração de ciclo de vida para a exclusão de objetos não atuais (30 dias) e carregamentos fracionados incompletos (três dias).
- Bloqueie o acesso público ao bucket do console.

Etapa 4: configurar os AWS SDKs AWS CLI e

As etapas a seguir mostram como instalar o AWS Command Line Interface (AWS CLI) e os AWS SDKs. Os exemplos nesta documentação usam os AWS CLI SDKs, Python e Java AWS .

Tópicos

- Instale os AWS SDKS
- Conceder acesso programático
- Configurar permissões do SDK
- Chame uma operação do Amazon Lookout for Vision

Instale os AWS SDKS

Siga as etapas para baixar e configurar os AWS SDKs.

Para configurar o AWS CLI e os AWS SDKs

- Baixe e instale o AWS CLIe os AWS SDKs que você deseja usar. Este guia fornece exemplos para o AWS CLI, Java e Python. Para obter informações sobre a instalação de 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
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI

Qual usuário precisa de acesso programático?	Para	Por
		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 AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para AWS SDKs AWS CLI ou APIs. AWS	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 AWS SDKs e ferramentas, consulte Autenticar usando credencia is de longo prazo no Guia de referência de AWS SDKs e ferramentas. • Para 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 `lookoutvision-access`, por exemplo:

```
session = boto3.Session(profile_name='lookoutvision-access')
lookoutvision_client = session.client("lookoutvision")
```

O usuário que o perfil representa deve ter permissões para chamar as operações do SDK do Lookout for Vision e AWS outras operações do SDK exigidas pelos exemplos. Para ter mais

informações, consulte Configurar permissões do SDK. Para atribuir permissões, consulte Como atribuir permissões.

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 `lookoutvision-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 instância Amazon EC2. O provedor gera e atualiza automaticamente as credenciais para você usando o perfil da instância do Amazon EC2 que você anexa à instância do Amazon EC2. Para obter mais informações, consulte Usar um perfil do IAM para conceder permissões a aplicativos executados em instâncias do Amazon EC2
- 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 serviços da AWS.
- Dispositivo principal do Greengrass: use certificados X.509 para se conectar ao AWS IoT Core usando protocolos TLS de autenticação mútua. Esses certificados permitem que os dispositivos interajam com o AWS IoT sem as credenciais da AWS. O provedor de credenciais do AWS IoT autentica os dispositivos usando o certificado X.509 e emite credenciais do AWS na forma de um token de segurança temporário e de privilégio limitado. Para obter mais informações, consulte Interagir com 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 Lookout for Vision SDK, você precisa de permissões de acesso à API do Lookout for Vision e ao bucket do Amazon S3 usado para o treinamento do modelo.

Tópicos

- Conceder permissões de operação do SDK
- Concessão de permissões do Amazon S3 Bucket
- 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 DetectAnomalies, você precisa de permissão para executar`lookoutvision:DetectAnomalies`. 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.

- AmazonLookoutVisionFullAccess— permite acesso total às operações do Amazon Lookout for Vision SDK.
- AmazonLookoutVisionReadOnlyAccess— permite acesso às operações do SDK somente para leitura.

As políticas gerenciadas do console também fornecem permissões de acesso para operações do SDK. Para ter mais informações, consulte Etapa 2: Configurar permissões.

Para obter informações sobre políticas AWS gerenciadas, consulte as políticas gerenciadas da AWS.

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.

> **Note**
>
> As instruções de introdução exigem permissões `s3:PutObject`. Para ter mais informações, consulte Etapa 1: criar o arquivo de manifesto e fazer upload de imagens.

Para atribuir permissões, consulte Como atribuir permissões.

Concessão de permissões do Amazon S3 Bucket

Para treinar um modelo, você precisa de um bucket do Amazon S3 com as permissões apropriadas para armazenar as imagens, os arquivos de manifesto e o resultado do treinamento. O bucket deve ser de propriedade da sua conta da AWS e deve estar localizado na região da AWS na qual você está usando o Amazon Lookout for Vision.

As políticas gerenciadas somente pelo SDK (`AmazonLookoutVisionFullAccess` e `AmazonLookoutVisionReadOnlyAccess`) não incluem permissões de bucket do Amazon S3 e você precisa aplicar a seguinte política de permissão para acessar os buckets que você usa, incluindo os buckets de console existentes.

As políticas gerenciadas pelo console (`AmazonLookoutVisionConsoleFullAccess` e `AmazonLookoutVisionConsoleReadOnlyAccess`) incluem permissões de acesso ao bucket do console. Se você estiver acessando o bucket do console com operações do SDK e tiver permissões de política gerenciada pelo console, não precisará usar a política a seguir. Para ter mais informações, consulte Etapa 2: Configurar permissões.

Decidindo permissões de tarefas

Use as informações a seguir para decidir quais permissões são necessárias para as tarefas que você deseja realizar.

Criar um conjunto de dados

Para criar um conjunto de dados com CreateDataset, você precisa das seguintes permissões.

- `s3:GetBucketLocation`: permite que o Lookout for Vision valide se o bucket está na mesma região em que você está usando o Lookout for Vision.
- `s3:GetObject`: permite acesso ao arquivo de manifesto especificado no parâmetro `DatasetSource` de entrada. Se quiser especificar uma versão exata do objeto do S3 do arquivo de manifesto, você também precisará do arquivo de manifesto `s3:GetObjectVersion`. Para obter mais informações, consulte Usando o versionamento em buckets do S3.

Como criar um modelo

Para criar um modelo com CreateModel, você precisa das seguintes permissões.

- `s3:GetBucketLocation`: permite que o Lookout for Vision valide se o bucket está na mesma região em que você está usando o Lookout for Vision.
- `s3:GetObject`: permite o acesso às imagens especificadas nos conjuntos de dados de treinamento e teste do projeto.
- `s3:PutObject`: concede a permissão para armazenar resultados de treinamento no bucket especificado. Você especifica a localização do bucket de saída no parâmetro `OutputConfig`. Opcionalmente, você pode definir o escopo das permissões somente para as chaves de objeto especificadas no campo `Prefix` do campo de entrada `S3Location`. Para obter mais informações, consulte OutputConfig.

Acessando imagens, arquivos de manifesto e resultados de treinamento

As permissões de bucket do Amazon S3 não são necessárias para visualizar as respostas da operação do Amazon Lookout for Vision. Você precisará da permissão `s3:GetObject` se quiser acessar imagens, arquivos de manifestos e resultados de treinamento referenciados nas respostas da operação. Se você estiver acessando um objeto do Amazon S3 com versionamento, precisará da permissão `s3:GetObjectVersion`.

Configuração da política de bucket do Amazon S3

Você pode usar a política a seguir para especificar as permissões de bucket do Amazon S3 necessárias para criar um conjunto de dados (`CreateDataset`), criar um modelo (`CreateModel`) e acessar imagens, arquivos de manifesto e resultados de treinamento. Altere o valor de *`my-bucket`* para o nome do bucket que você deseja usar.

Você pode ajustar a política às suas necessidades. Para ter mais informações, consulte Decidindo permissões de tarefas. Adicione a política ao usuário desejado. Para obter mais informações, consulte Criação de políticas de IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionS3BucketAccess",
            "Effect": "Allow",
            "Action": "s3:GetBucketLocation",
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        },
        {
            "Sid": "LookoutVisionS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/*"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        }
    ]
}
```

Para atribuir permissões, consulte Como atribuir permissões.

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 Criar 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 Lookout for Vision

Execute o código a seguir para confirmar que você pode fazer chamadas para a API Amazon Lookout for Vision. 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 `ListProjects`.

Em geral, chamar uma função de exemplo requer um cliente do AWS SDK para Lookout for Vision e quaisquer outros parâmetros necessários. O cliente AWS SDK Lookout for Vision 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 Amazon Lookout for Vision não está disponível em AWS todas as regiões.

Para chamar uma operação Amazon Lookout for Vision

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

   CLI

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

   ```
   aws lookoutvision list-projects \
   --profile lookoutvision-access
   ```

   Python

   Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

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


   from botocore.exceptions import ClientError
   import boto3

   class GettingStarted:

       @staticmethod
   ```

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

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("ARN: " + project["ProjectArn"])
                print()
            print("Done!")
        except ClientError:
            raise
def main():
    session = boto3.Session(profile_name='lookoutvision-access')
    lookoutvision_client = session.client("lookoutvision")

    GettingStarted.list_projects(lookoutvision_client)

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

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

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

package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.ProjectMetadata;
import
 software.amazon.awssdk.services.lookoutvision.paginators.ListProjectsIterable;
```

```
import software.amazon.awssdk.services.lookoutvision.model.ListProjectsRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;

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

public class GettingStarted {

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

    /**
     * Lists the Amazon Lookoutfor Vision projects in the current AWS account
 and
     * AWS Region.
     *
     * @param lfvClient   An Amazon Lookout for Vision client.
     * @return List<ProjectMetadata> Metadata for each project.
     */
    public static List<ProjectMetadata> listProjects(LookoutVisionClient
 lfvClient)
            throws LookoutVisionException {

        logger.log(Level.INFO, "Getting projects:");
        ListProjectsRequest listProjectsRequest = ListProjectsRequest.builder()
                .maxResults(100)
                .build();

        List<ProjectMetadata> projectMetadata = new ArrayList<>();

        ListProjectsIterable projects =
 lfvClient.listProjectsPaginator(listProjectsRequest);

        projects.stream().flatMap(r -> r.projects().stream())
                .forEach(project -> {
                    projectMetadata.add(project);
                    logger.log(Level.INFO, project.projectName());
                });

        logger.log(Level.INFO, "Finished getting projects.");
```

```
        return projectMetadata;

    }

    public static void main(String[] args) throws Exception {

        try {

            // Get the Lookout for Vision client.
            LookoutVisionClient lfvClient = LookoutVisionClient.builder()

 .credentialsProvider(ProfileCredentialsProvider.create("lookoutvision-access"))
                    .build();

            List<ProjectMetadata> projects = Projects.listProjects(lfvClient);

            System.out.printf("Projects%n--------%n");

            for (ProjectMetadata project : projects) {
                System.out.printf("Name: %s%n", project.projectName());
                System.out.printf("ARN: %s%n", project.projectArn());
                System.out.printf("Date: %s%n%n",
 project.creationTimestamp().toString());
            }

        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Could not list projects: {0}: {1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("Could not list projects: %s",
 lfvError.getMessage()));
            System.exit(1);
        }

    }

}
```

Etapa 5: (Opcional) Usando sua própria chave do AWS Key Management Service

É possível usar o AWS Key Management Service (KMS) para gerenciar a criptografia das imagens de entrada que você armazena nos buckets do Amazon S3.

Por padrão, suas imagens são criptografadas com uma chave que a AWS possui e gerencia. Você também pode optar por usar sua própria chave do AWS Key Management Service (KMS). Para obter mais informações, consulte Conceitos do AWS Key Management Service.

Se você quiser usar sua própria chave KMS, use a política a seguir para especificar a chave KMS. Altere *kms_key_arn* para o ARN da chave do KMS (ou ARN do alias do KMS) que você deseja usar. Como alternativa, especifique * para usar qualquer chave do KMS. Para obter informações sobre como adicionar a política a um usuário ou função, consulte Criação de políticas de IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionKmsDescribeAccess",
            "Effect": "Allow",
            "Action": "kms:DescribeKey",
            "Resource": "kms_key_arn"
        },
        {
            "Sid": "LookoutVisionKmsCreateGrantAccess",
            "Effect": "Allow",
            "Action": "kms:CreateGrant",
            "Resource": "kms_key_arn",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "lookoutvision.*.amazonaws.com"
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

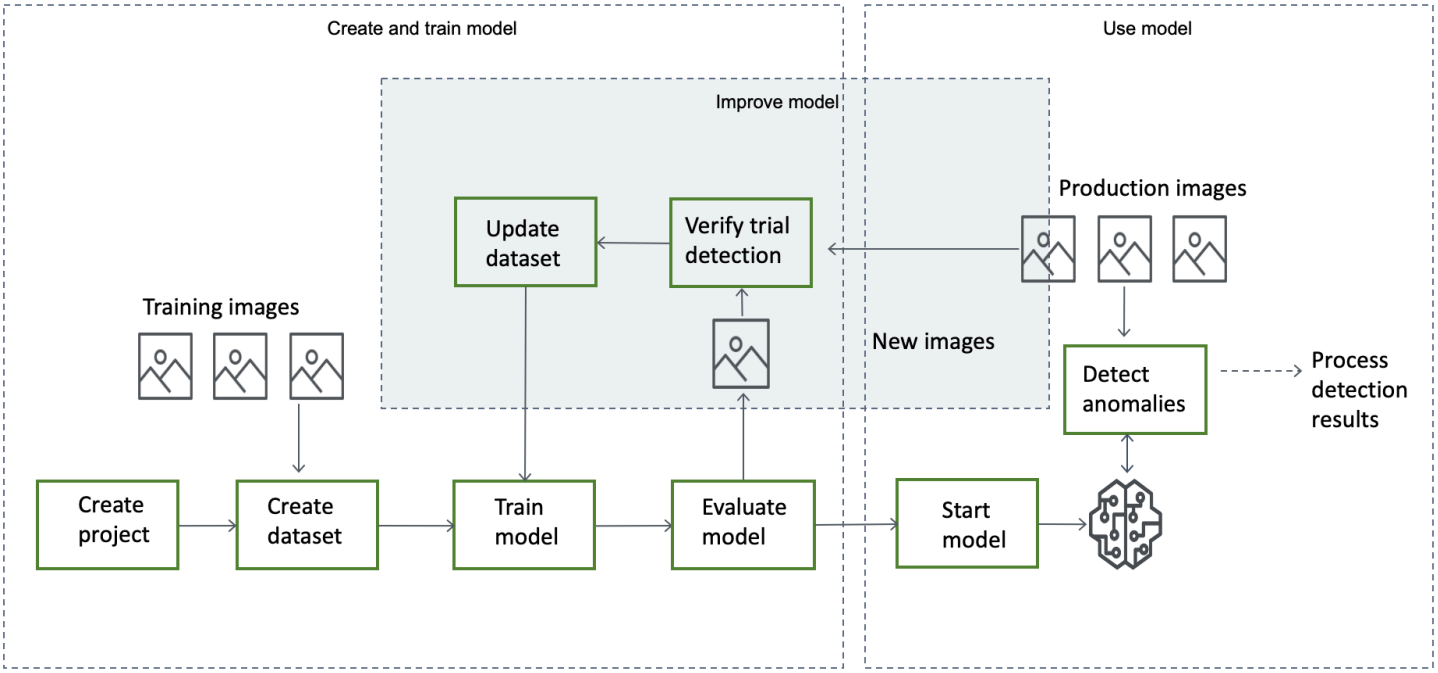
Noções básicas sobre o Amazon Lookout for Vision

Você pode usar o Amazon Lookout for Vision para encontrar defeitos visuais em produtos industriais, com precisão e em grande escala, para tarefas como:

- Detecção de peças danificadas — Detecte danos na qualidade, cor e formato da superfície de um produto durante o processo de fabricação e montagem.
- Identificação de componentes ausentes — Determine os componentes ausentes com base na ausência, presença ou posicionamento de objetos. Por exemplo, falta um capacitor em uma placa de circuito impresso.
- Descobrindo problemas de processo — Detecte defeitos com padrões repetidos, como arranhões repetidos no mesmo ponto em uma pastilha de silicone.

Com o Lookout for Vision, você cria um modelo de visão computacional que prevê a presença de anomalias em uma imagem. Você fornece as imagens que o Amazon Lookout for Vision usa para treinar e testar seu modelo. O Amazon Lookout for Vision fornece métricas que você pode usar para avaliar e melhorar seu modelo treinado. Você pode hospedar o modelo treinado na nuvem AWS ou implantar o modelo em um dispositivo de ponta. Uma operação de API simples retorna as previsões que seu modelo faz.

O fluxo de trabalho geral para criar, avaliar e usar um modelo é o seguinte:



Tópicos

Escolha seu tipo de modelo

Antes de criar um modelo, você deve decidir qual tipo de modelo deseja. Você pode criar dois tipos de modelo: classificação de imagens e segmentação de imagens. Você decide o tipo de modelo a ser criado com base em seu caso de uso.

Modelo de classificação de imagens

Se você só precisa saber se uma imagem contém uma anomalia, mas não precisa saber sua localização, crie um modelo de classificação de imagens. Um modelo de classificação de imagens faz uma previsão sobre se uma imagem contém uma anomalia. A previsão inclui a confiança do modelo na precisão da previsão. O modelo não fornece nenhuma informação sobre a localização de nenhuma anomalia encontrada na imagem.

Modelo de segmentação de imagens

Se você precisar saber a localização de uma anomalia, como a localização de um arranhão, crie um modelo de segmentação de imagem. Os modelos do Amazon Lookout for Vision usam segmentação semântica para identificar os pixels em uma imagem em que os tipos de anomalias (como um arranhão ou uma peça perdida) estão presentes.

> **Note**
>
> Um modelo de segmentação semântica localiza diferentes tipos de anomalia. Ele não fornece informações de instância para anomalias individuais. Por exemplo, se uma imagem contém dois amassados, o Lookout for Vision retorna informações sobre ambos os amassados em uma única entidade que representa o tipo de anomalia de amassado.

Um modelo de segmentação do Amazon Lookout for Vision prevê o seguinte:

Classificação

O modelo retorna uma classificação para uma imagem analisada (normal/anomalia), que inclui a confiança do modelo na previsão. As informações de classificação são calculadas separadamente das informações de segmentação e você não deve presumir uma relação entre elas.

Segmentação

O modelo retorna uma máscara de imagem que marca os pixels em que ocorrem anomalias na imagem. Diferentes tipos de anomalia são codificados por cores de acordo com a cor atribuída ao rótulo de anomalia no conjunto de dados. Um rótulo de anomalia representa o tipo de anomalia. Por exemplo, a máscara azul na imagem a seguir marca a localização de um tipo de anomalia de arranhão encontrado em um carro.

O modelo retorna o código de cor para cada etiqueta de anomalia na máscara. O modelo também retorna a porcentagem de cobertura da imagem que uma etiqueta de anomalia tem.

Com um modelo de segmentação do Lookout for Vision, você pode usar vários critérios para analisar os resultados da análise do modelo. Por exemplo:

- Localização da anomalia — Se você precisar saber a localização das anomalias, use as informações de segmentação para ver as máscaras que cobrem as anomalias.
- Tipos de anomalia — Use as informações de segmentação para decidir se uma imagem contém mais do que um número aceitável de tipos de anomalia.
- Área de cobertura — Use as informações de segmentação para decidir se um tipo de anomalia cobre mais do que uma área aceitável de uma imagem.
- Classificação de imagens — Se você não precisar saber a localização das anomalias, use as informações de classificação para determinar se uma imagem contém anomalias.

Para ver um código demonstrativo, consulte Detectar as anomalias de uma imagem.

Depois de decidir qual tipo de modelo você quer, você cria um projeto e um conjunto de dados para gerenciar seu modelo. Usando rótulos, você pode classificar as imagens como normais ou como anomalias. Os rótulos também identificam informações de segmentação, como máscaras e tipos de anomalias. A forma como você rotula as imagens em seu conjunto de dados determina o tipo de modelo que o Lookout for Vision cria para você.

Rotular um modelo de segmentação de imagens é mais complexo do que rotular um modelo de classificação de imagens. Para treinar um modelo de segmentação, você precisa classificar as imagens de treinamento como normais ou anômalas. Você também precisa definir máscaras e tipos de anomalias para cada imagem anômala. Um modelo de classificação exige apenas que você identifique as imagens de treinamento como normais ou anômalas.

Criar seu modelo

As etapas para criar um modelo são criar um projeto, criar um conjunto de dados e treinar o modelo são as seguintes:

Criar um projeto

Crie um projeto para gerenciar os conjuntos de dados e os modelos que você cria. Um projeto deve ser usado para um único caso de uso, como detectar anomalias em um único tipo de peça da máquina.

Você pode usar o painel do para obter uma visão geral dos seus projetos. Para obter mais informações, consulte Usando o painel do Amazon Lookout for Vision.

Mais informações: Crie seu projeto.

Criação de um conjunto de dados

Para treinar um modelo, o Amazon Lookout for Vision precisa de imagens de objetos normais e anômalos para seu caso de uso. Você fornece essas imagens em um conjunto de dados.

Um conjunto de dados é um conjunto de imagens e rótulos que descrevem essas imagens. As imagens devem representar um único tipo de objeto no qual possam ocorrer anomalias. Para obter mais informações, consulte Preparando imagens para um conjunto de dados.

Com o Amazon Lookout for Vision, você pode ter um projeto que usa um único conjunto de dados ou um projeto que tenha conjuntos de dados de treinamento e teste separados. Recomendamos usar

um único projeto de conjunto de dados, a menos que você queira um controle mais preciso sobre o treinamento, os testes e o ajuste de desempenho.

Você cria um conjunto de dados importando as imagens. Dependendo de como você importa as imagens, elas também podem ser rotuladas. Caso contrário, você usa o console para rotular as imagens.

Importar imagens

Se você criar o conjunto de dados com o console do Lookout for Vision, poderá importar as imagens de uma das seguintes formas:

- Importe imagens do computador local. As imagens não estão rotuladas.
- Importe imagens de um bucket do S3. O Amazon Lookout for Vision pode classificar as imagens usando os nomes das pastas que contêm as imagens. Use `normal` para imagens normais. Use `anomaly` para imagens anômalas. Você não pode atribuir rótulos de segmentação automaticamente.
- Importe um arquivo de manifesto do Amazon SageMaker Ground Truth. As imagens em um arquivo de manifesto são rotuladas. Você pode criar e importar seu próprio arquivo de manifesto. Se você tiver muitas imagens, considere usar o serviço de rotulagem SageMaker Ground Truth. Em seguida, você importa o arquivo de manifesto de saída da tarefa do Amazon SageMaker Ground Truth.

Rotulagem de imagens

Os rótulos descrevem uma imagem em um conjunto de dados. Os rótulos especificam se uma imagem é normal ou anômala (classificação). Os rótulos também descrevem a localização das anomalias em uma imagem (segmentação).

Se suas imagens não estiverem rotuladas, você poderá usar o console para rotulá-las.

Os rótulos que você atribui às imagens em seu conjunto de dados determinam o tipo de modelo que o Lookout for Vision cria:

Classificação de imagens

Para criar um modelo de classificação de imagens, use o console Lookout for Vision para classificar as imagens no conjunto de dados como normais ou como anomalias.

Você também pode usar a operação `CreateDataset` para criar um conjunto de dados a partir de um arquivo de manifesto que inclui informações de classificação.

Segmentação de imagens

Para criar um modelo de segmentação de imagens, use o console do Lookout for Vision para classificar as imagens no conjunto de dados como normais ou como anomalias. Você também especifica máscaras de pixels para áreas anômalas na imagem (se existirem), bem como um rótulo de anomalia para máscaras de anomalia individuais.

Você também pode usar a operação `CreateDataset` para criar um conjunto de dados a partir de um arquivo de manifesto que inclui informações de segmentação e classificação.

Se seu projeto tiver conjuntos de dados de treinamento e teste separados, o Lookout for Vision usa o conjunto de dados de treinamento para aprender e determinar o tipo de modelo. Você deve rotular as imagens em seu conjunto de dados de teste da mesma forma.

Mais informações: Criação do seu conjunto de dados.

Treinar seu modelo

O treinamento cria um modelo e o treina para prever a presença de anomalias nas imagens. Uma nova versão do seu modelo é criada toda vez que você treina.

No início do treinamento, o Amazon Lookout for Vision escolhe o algoritmo mais adequado para treinar seu modelo. O modelo é treinado e depois testado. Em Conceitos básicos do Amazon Lookout for Vision, você treina um único projeto de conjunto de dados, o conjunto de dados é dividido internamente para criar um conjunto de dados de treinamento e um conjunto de dados de teste. Você também pode criar um projeto com conjuntos de dados de treinamento e teste separados. Nessa configuração, o Amazon Lookout for Vision treina seu modelo com o conjunto de dados de treinamento e testa o modelo com o conjunto de dados de teste.

> **Important**
>
> Você será cobrado pelo tempo necessário para treinar seu modelo com sucesso.

Mais informações: Treine seu modelo.

Avaliar seu modelo

Avalie o desempenho do seu modelo usando as métricas de desempenho criadas durante o teste.

Usando métricas de desempenho, você pode entender melhor o desempenho do seu modelo treinado e decidir se está pronto para usá-lo na produção.

Mais informações: Melhorando seu modelo.

Se as métricas de desempenho indicarem que melhorias são necessárias, você pode adicionar mais dados de treinamento executando uma tarefa de detecção de testes com novas imagens. Depois que a tarefa for concluída, você poderá verificar os resultados e adicionar as imagens verificadas ao seu conjunto de dados de treinamento. Como alternativa, você pode adicionar novas imagens de treinamento diretamente ao conjunto de dados. Em seguida, você treina novamente seu modelo e verifica novamente as métricas de desempenho.

Mais informações: Verificando seu modelo com uma tarefa de detecção de teste.

Use seu modelo

Antes de usar seu modelo na AWS nuvem, você inicia o modelo com a operação startModel. Você pode obter o comando `StartModel` CLI para seu modelo no console.

Mais informações: Inicie seu modelo.

Um modelo treinado do Amazon Lookout for Vision prevê se uma imagem de entrada contém conteúdo normal ou anômalo. Se seu modelo for um modelo de segmentação, a previsão inclui uma máscara de anomalia que marca os pixels onde as anomalias são encontradas.

Para fazer uma previsão com seu modelo, chame a operação DetectAnomalies e passe uma imagem de entrada do seu computador local. Você pode obter o comando CLI que chama `DetectAnomalies` do console.

Mais informações: Detecte anomalias em uma imagem.

> **Important**
>
> Você é cobrado pelo tempo em que seu modelo está em execução.

Se você não estiver mais usando seu modelo, use a operação StopModel para parar o modelo. É possível obter o comando da CLI no console do.

Mais informações: Pare seu modelo.

Use seu modelo em um dispositivo Edge

Você pode usar um modelo do Lookout for Vision em AWS IoT Greengrass Version 2 um dispositivo principal.

Mais informações: Usando seu modelo Amazon Lookout for Vision em um dispositivo de ponta.

Use seu painel

Você pode usar o painel para obter uma visão geral de todos os seus projetos e informações gerais de projetos individuais.

Mais informações: Use seu painel.

Conceitos básicos do Amazon Lookout for Vision

Antes de iniciar estas instruções de introdução, recomendamos que você leiaNoções básicas sobre o Amazon Lookout for Vision.

As instruções de introdução mostram como criar um exemplo de modelo de segmentação de imagens. Se você quiser criar um exemplo de modelo de classificação de imagens, consulteConjunto de dados de classificação de imagens.

Se você quiser experimentar rapidamente um modelo de exemplo, fornecemos exemplos de imagens de treinamento e imagens de máscara. Também fornecemos um script Python que cria um arquivo manifesto de segmentação de imagens. Você usa o arquivo de manifesto para criar um conjunto de dados para seu projeto e não precisa rotular as imagens no conjunto de dados. Ao criar um modelo com suas próprias imagens, você deve rotular as imagens no conjunto de dados. Para obter mais informações, consulte Criar um conjunto de dados.

As imagens que fornecemos são de cookies normais e anômalos. Um biscoito anômalo tem uma rachadura na forma de biscoito. O modelo que você treina com as imagens prevê uma classificação (normal ou anômala) e encontra a área (máscara) das rachaduras em um biscoito anômalo, conforme mostrado no exemplo a seguir.

Tópicos

Etapa 1: criar o arquivo de manifesto e fazer upload de imagens

Neste procedimento, você clona o repositório de documentação do Amazon Lookout for Vision em seu computador. Em seguida, você usa um script Python (versão 3.7 ou superior) para criar um arquivo de manifesto e carregar as imagens de treinamento e as imagens de máscara em um local do Amazon S3 que você especificar. Você usa o arquivo de manifesto para criar seu modelo. Posteriormente, você usa imagens de teste no repositório local para testar seu modelo.

Para criar o arquivo de manifesto e fazer upload de imagens

1. Configure o Amazon Lookout for Vision seguindo as instruções em Configurar o Amazon Lookout for Vision Certifique-se de instalar o AWSSDK for Python.
2. Na AWS região em que você deseja usar o Lookout for Vision, crie um bucket do S3.
3. No bucket do Amazon S3, crie uma pasta chamada `getting-started`.
4. Observe o URI do Amazon S3 e o nome do recurso da Amazon (ARN) da pasta. Você os usa para configurar permissões e executar o script.
5. Certifique-se de que o usuário que está chamando o script tenha permissão para chamar a `s3:PutObject` operação. Você pode usar a seguinte política. Para atribuir permissões, consulte Como atribuir permissões

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Sid": "Statement1",
  "Effect": "Allow",
  "Action": [
   "s3:PutObject"
  ],
  "Resource": [
   "arn:aws:s3::: ARN for S3 folder in step 4/*"
  ]
 }]
}
```

6. Verifique se você tem um perfil local nomeado `lookoutvision-access` e se o usuário do perfil tem a permissão da etapa anterior. Para obter mais informações, consulte Usando um perfil em seu computador local.
7. Faça o download do arquivo zip, getting-started.zip. O arquivo zip contém o conjunto de dados de introdução e o script de configuração.

8. Descompacte o arquivo `getting-started.zip`.
9. No prompt de comando, faça o seguinte:

   a. Navegue para a pasta `getting-started`.

   b. Execute o comando a seguir para criar um arquivo de manifesto e carregar as imagens de treinamento e as máscaras de imagem para o caminho do Amazon S3 que você anotou na etapa 4.

   ```
   python getting_started.py S3-URI-from-step-4
   ```

   c. Quando o script for concluído, anote o caminho para o `train.manifest` arquivo que o script exibe depois`Create dataset using manifest file:`. O caminho deve ser similar aos`s3://path to getting started folder/manifests/train.manifest`.

Etapa 2: Criar o modelo

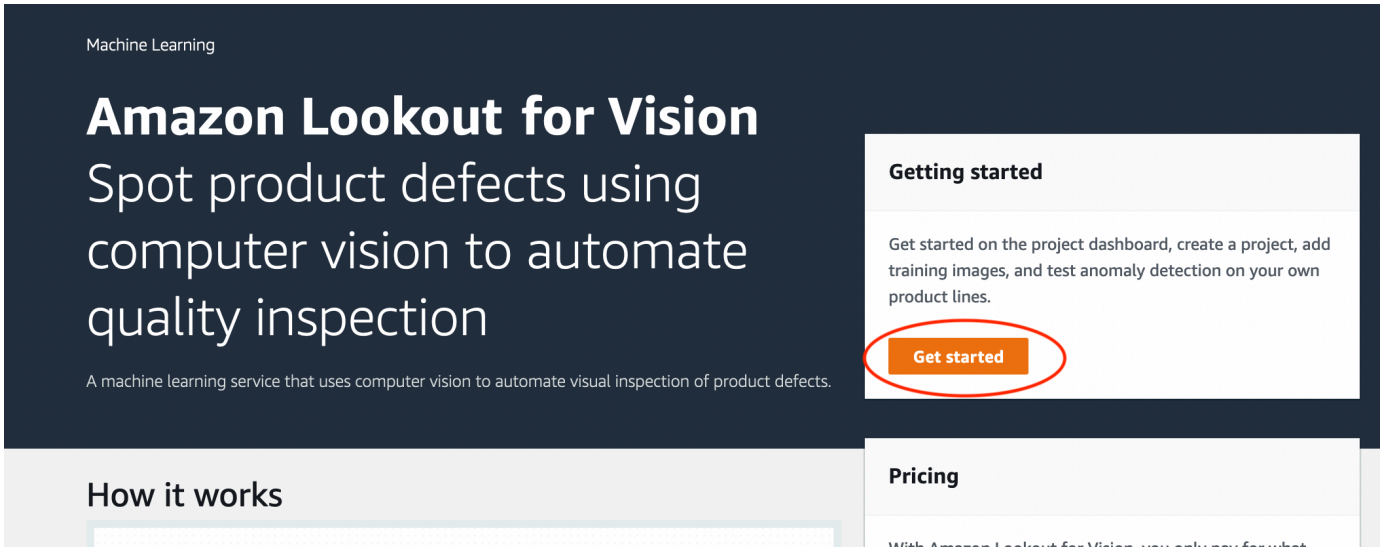
Neste procedimento, você cria um projeto e um conjunto de dados usando as imagens e o arquivo de manifesto que você carregou anteriormente no seu bucket do Amazon S3. Em seguida, você cria o modelo e visualiza os resultados da avaliação do treinamento do modelo.

Como você cria o conjunto de dados a partir do arquivo de manifesto de introdução, não precisa rotular as imagens do conjunto de dados. Ao criar um conjunto de dados com suas próprias imagens, você precisa rotular as imagens. Para obter mais informações, consulte Rotulagem de imagens.

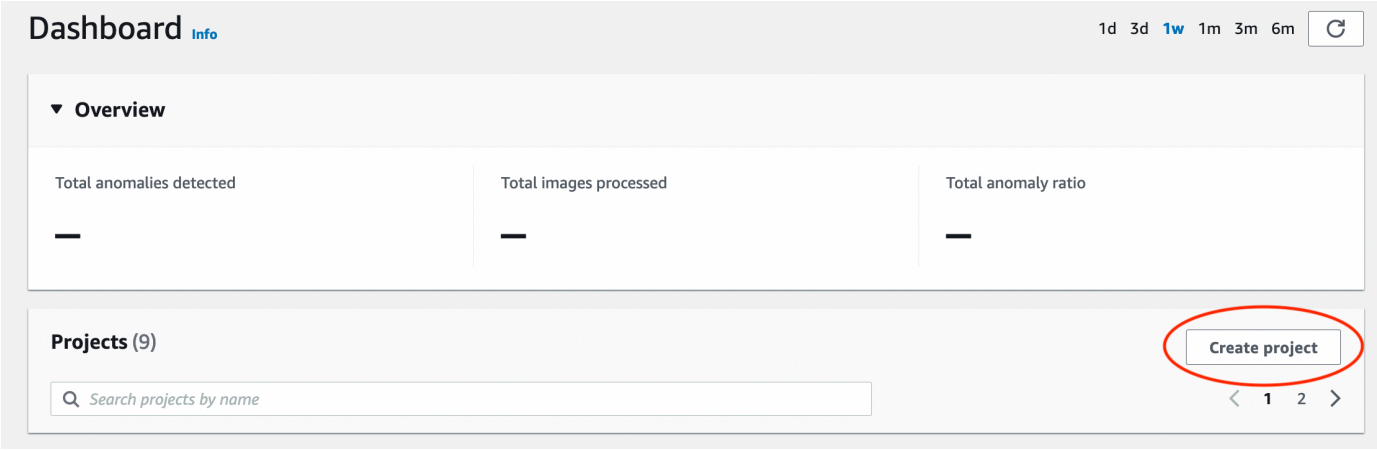
> **Important**
>
> Você é cobrado pelo treinamento bem-sucedido de um modelo.

Como criar um modelo

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Verifique se você está na mesma região da AWS em que criou o bucket do Amazon S3 em Etapa 1: criar o arquivo de manifesto e fazer upload de imagens. Na barra de navegação, escolha o nome da região exibida no momento. Depois, escolha a região para a qual pretende alternar.
3. Escolha Como começar.



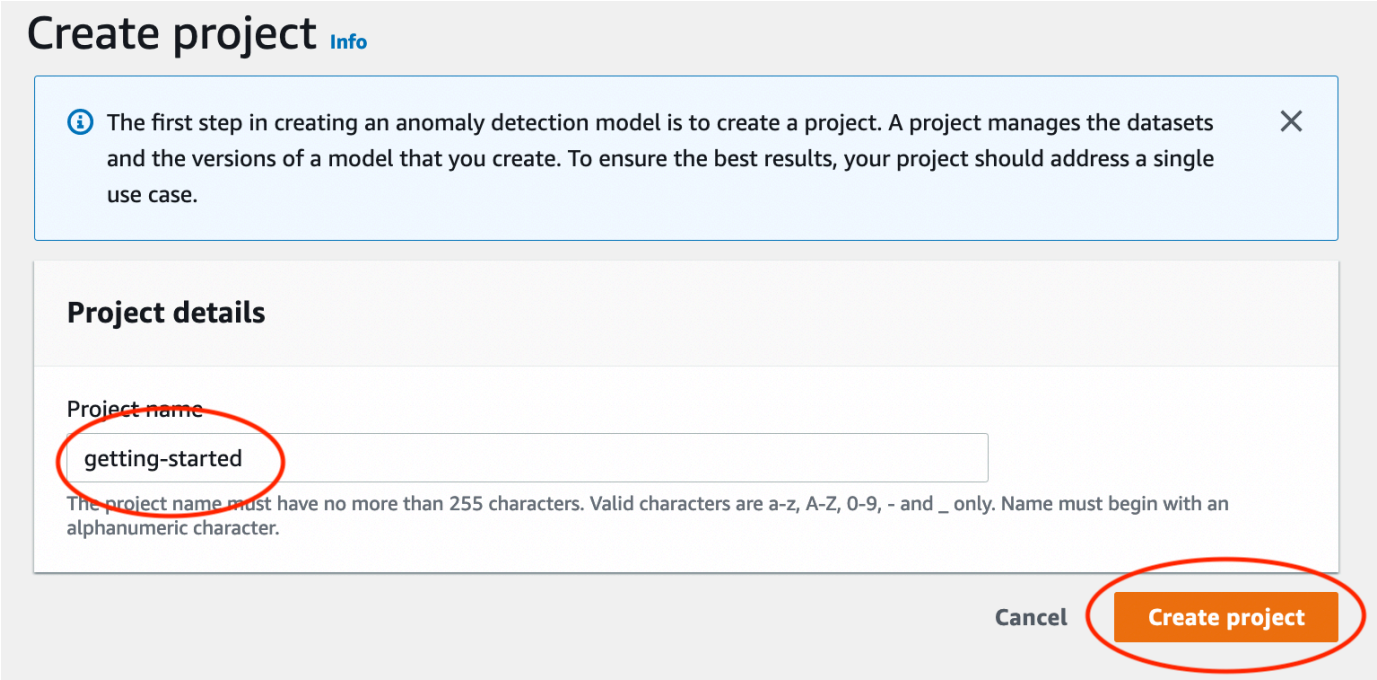
4. Na seção Projetos, escolha Criar projeto).



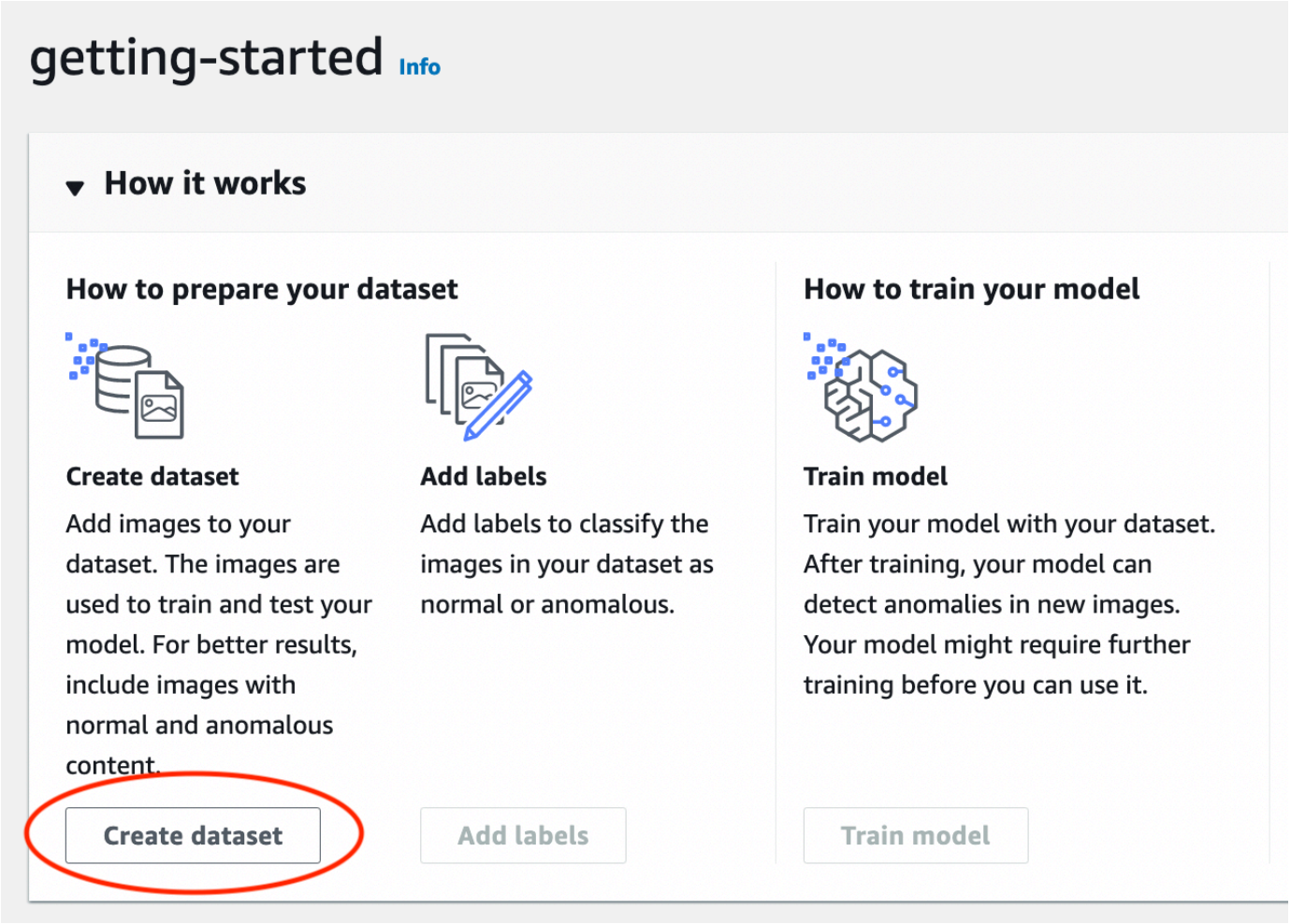
5. Na página Criar política faça o seguinte:

   a. Em Nome do projeto, insira `getting-started`.

   b. Escolha Criar projeto.



6. Na página do projeto, na seção Como funciona, escolha Criar conjunto de dados.



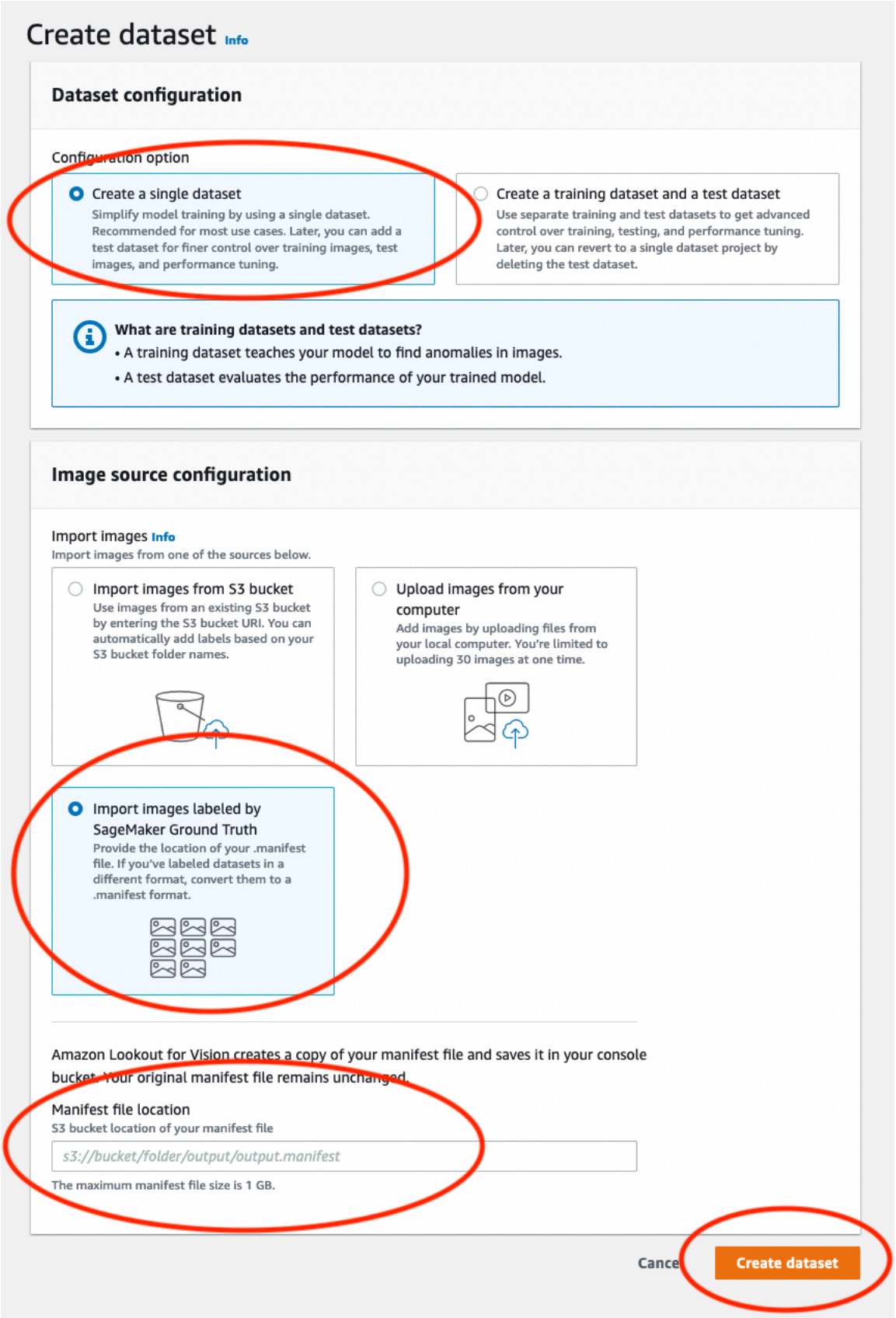
7. Na página Criar banco de dados , faça o seguinte:

   a. Escolha Criar um único conjunto de dados.

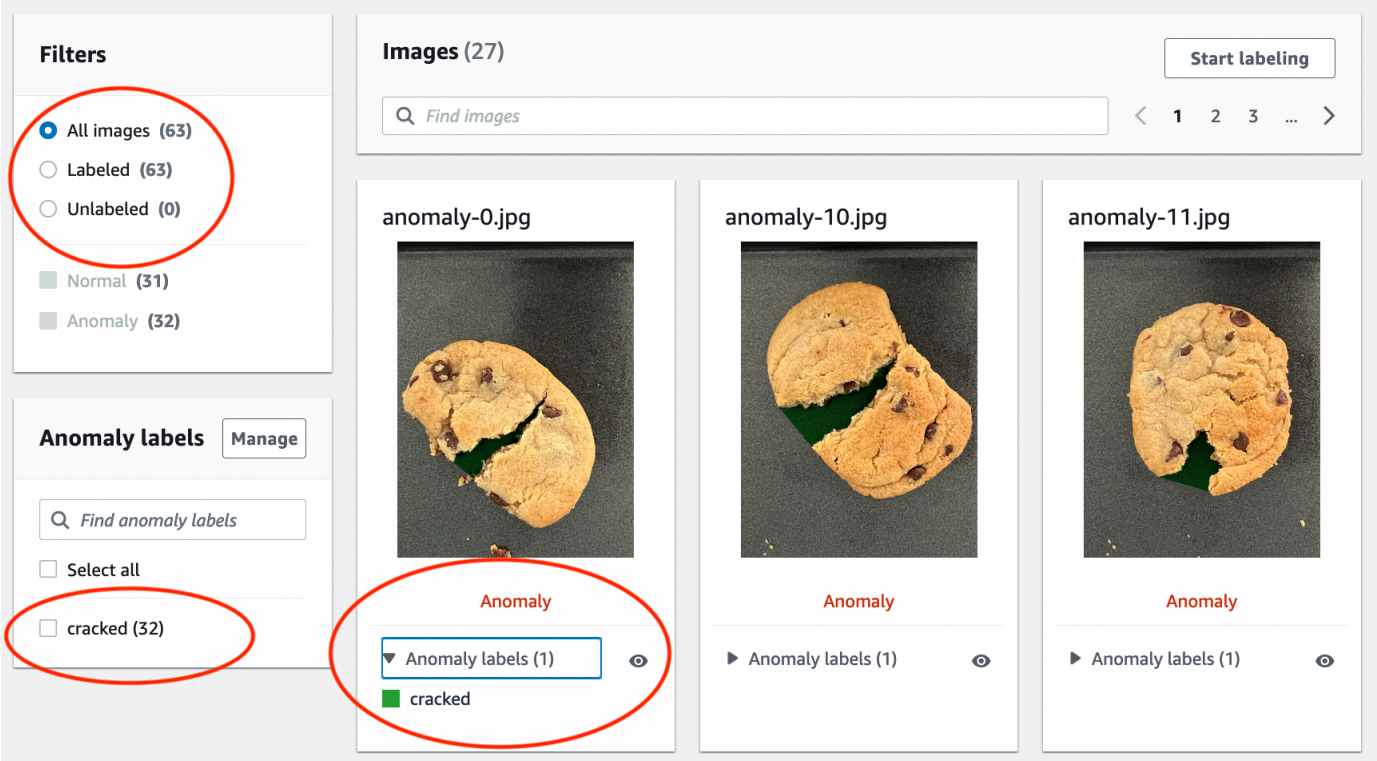
   b. Na seção Configuração da fonte de imagem, escolha Importar imagens rotuladas pelo SageMaker Ground Truth.

   c. Para a localização do arquivo.manifest, insira a localização do arquivo manifesto no Amazon S3 que você anotou na etapa 6.c. de Etapa 1: criar o arquivo de manifesto e fazer upload de imagens A localização do Amazon S3 deve ser semelhante a `s3://path to getting started folder/manifests/train.manifest`

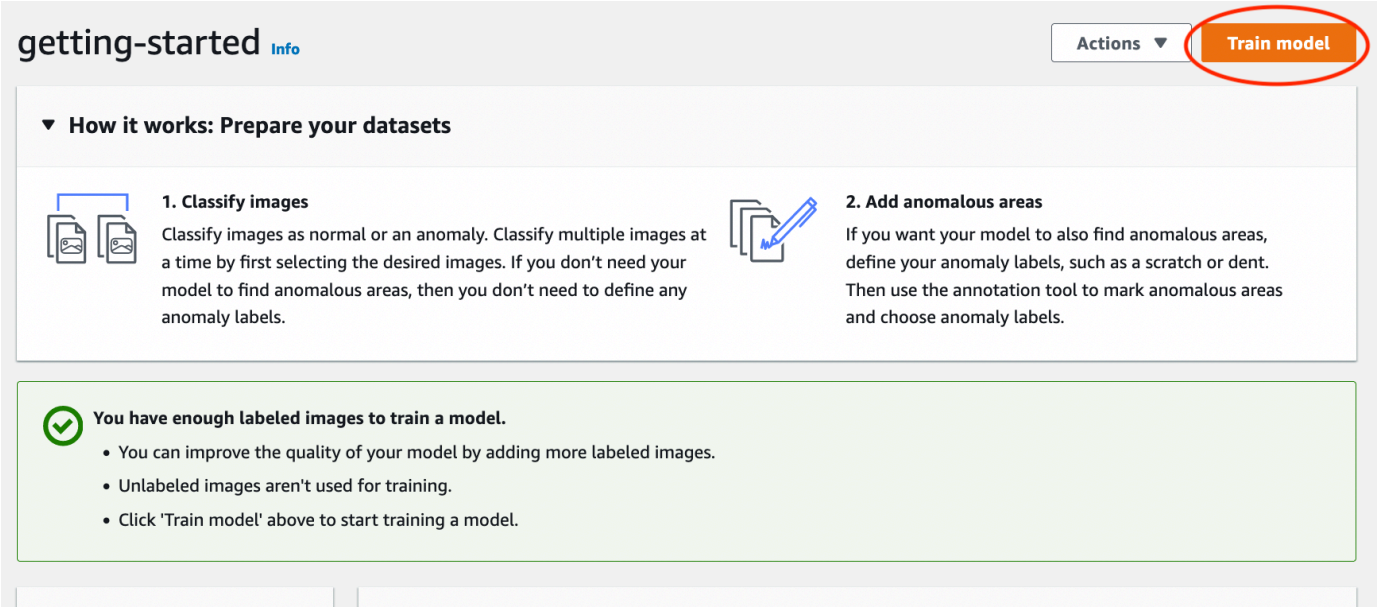
   d. Escolha Criar grupo de conjuntos de dados.


Create dataset Info
Dataset configuration
Configuration option
Create a single dataset
Simplify model training by using a single dataset. Recommended for most use cases. Later, you can add a test dataset for finer control over training images, test images, and performance tuning.
Create a training dataset and a test dataset
Use separate training and test datasets to get advanced control over training, testing, and performance tuning. Later, you can revert to a single dataset project by deleting the test dataset.
What are training datasets and test datasets?
• A training dataset teaches your model to find anomalies in images.
• A test dataset evaluates the performance of your trained model.
Image source configuration
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 URI. 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.
Import images labeled by SageMaker Ground Truth
Provide the location of your .manifest file. If you've labeled datasets in a different format, convert them to a .manifest format.
Amazon Lookout for Vision creates a copy of your manifest file and saves it in your console bucket. Your original manifest file remains unchanged.
Manifest file location
S3 bucket location of your manifest file
s3://bucket/folder/output/output.manifest
The maximum manifest file size is 1 GB.
Cance
Create dataset

8. Na página de detalhes do projeto, na seção Imagens, visualize as imagens do conjunto de dados. Você pode visualizar as informações de classificação e segmentação da imagem (rótulos de máscara e anomalia) para cada imagem do conjunto de dados. Você também pode pesquisar imagens, filtrar imagens pelo status da etiqueta (etiquetada/não rotulada) ou filtrar imagens pelos rótulos de anomalia atribuídos a elas.

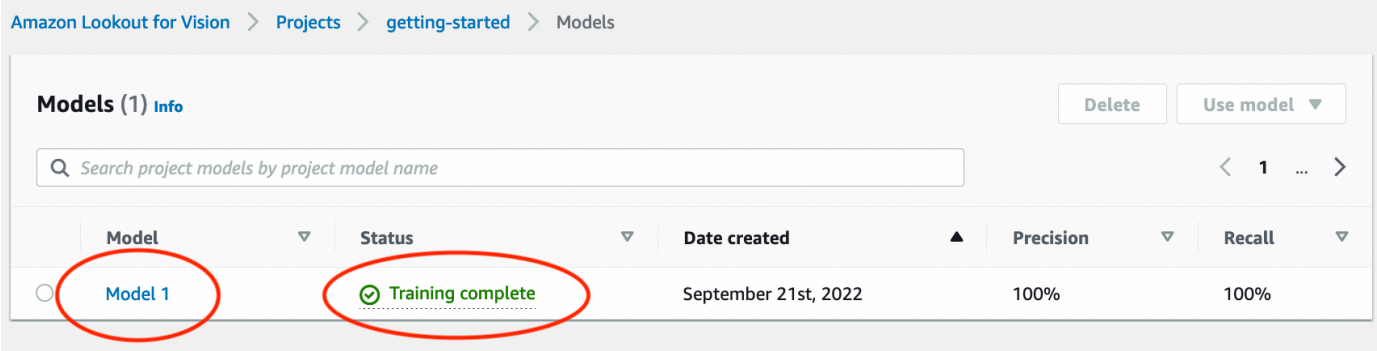


9. Na página de detalhes do projeto, escolha Modelo de trem.



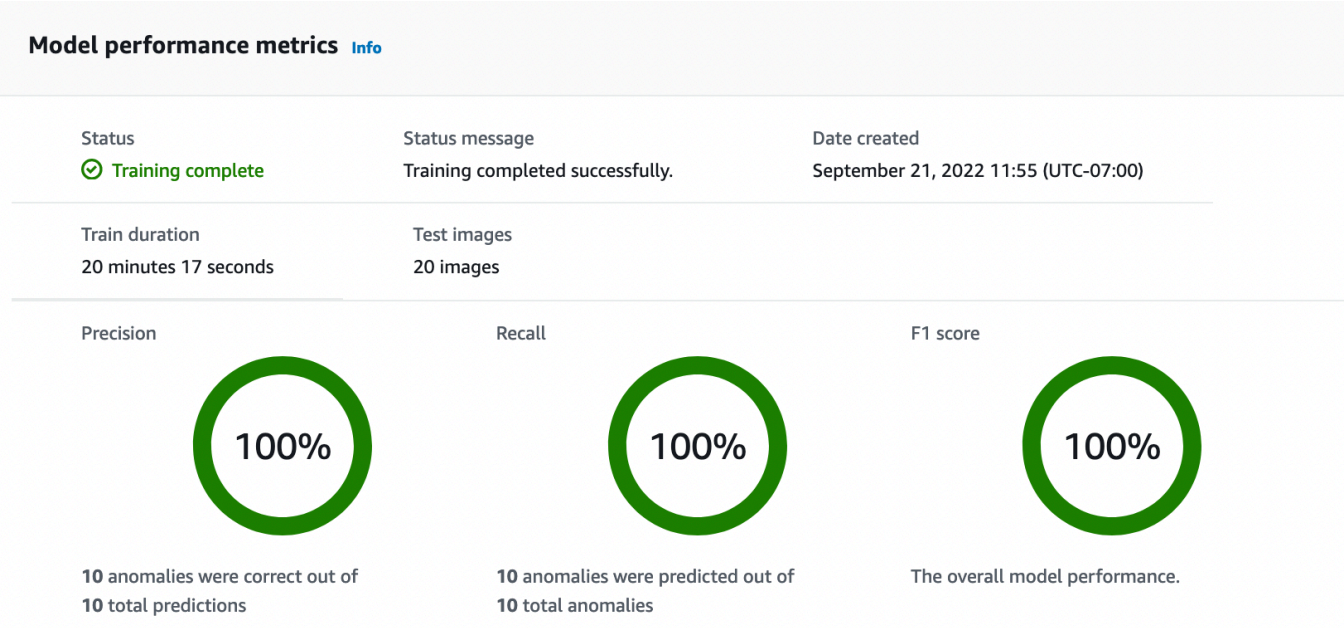
10. Na página de detalhes do modelo do trem, escolha Modelo do trem.

11. No Você quer treinar seu modelo? caixa de diálogo, escolha Modelo do trem.
12. Na página Modelos do projeto, você pode ver que o treinamento começou. Verifique o status atual visualizando a coluna Status da versão do modelo. O treinamento do modelo leva pelo menos 30 minutos para ser concluído. O treinamento foi concluído com sucesso quando o status muda para Treinamento concluído.
13. Quando o treinamento terminar, escolha o modelo Modelo 1 na página Modelos.

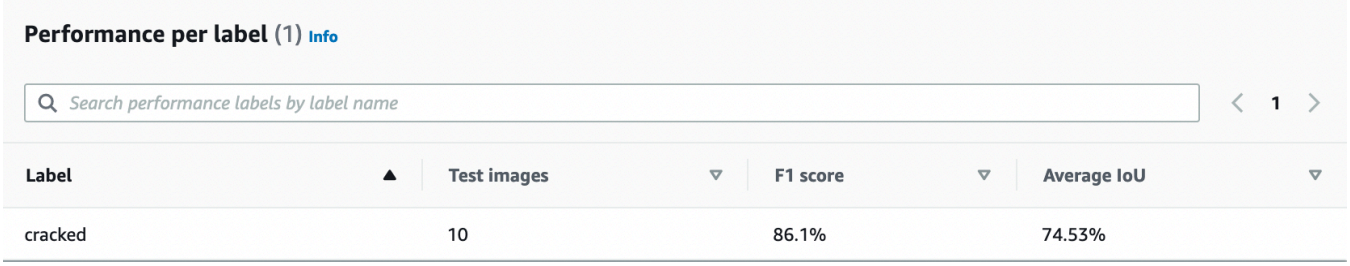


14. Na página de detalhes do modelo, visualize os resultados da avaliação na guia Métricas de desempenho. Existem métricas para o seguinte:

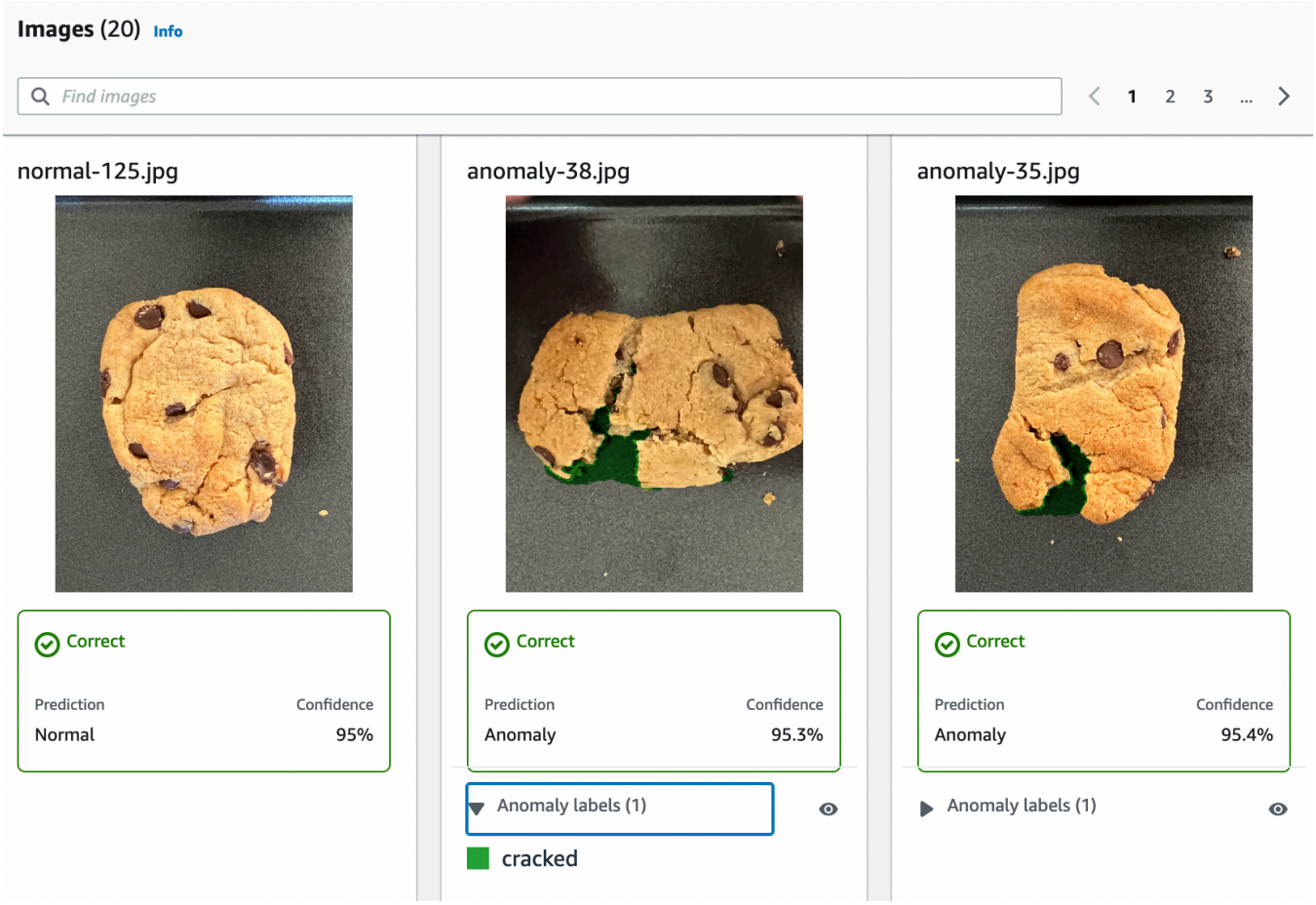
    - Métricas gerais de desempenho do modelo (precisão, recall e pontuação F1) para as previsões de classificação feitas pelo modelo.



    - Métricas de desempenho para rótulos de anomalias encontrados nas imagens de teste (média de IoU, pontuação F1)



- Previsões para imagens de teste (classificação, máscaras de segmentação e rótulos de anomalias)



Como o treinamento do modelo não é determinístico, os resultados da avaliação podem ser diferentes dos resultados mostrados nesta página. Para obter mais informações, consulte Melhoria de um modelo do Amazon Lookout for Vision.

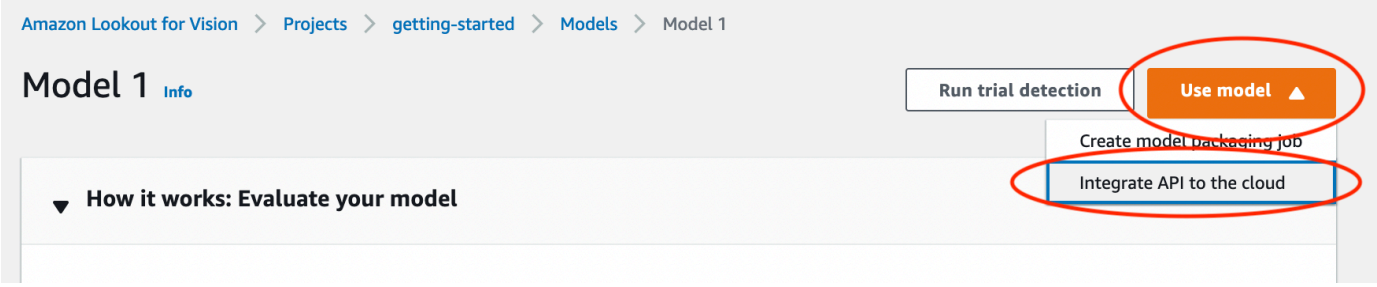
Etapa 3: iniciar o modelo

Nesta etapa, você começa a hospedar o modelo para que ele esteja pronto para analisar imagens. Para obter mais informações, consulte Executando seu modelo treinado do Amazon Lookout for Vision.

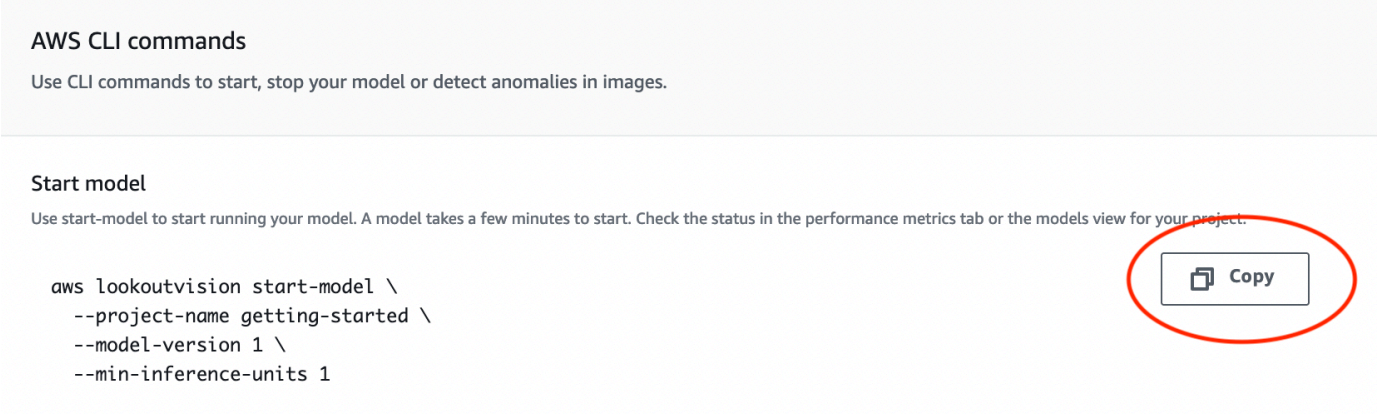
> **Note**
>
> Você é cobrado pela quantidade de tempo que o modelo funciona. Você interrompe seu modeloEtapa 5: interromper o modelo.

Como iniciar o modelo.

1. Na página de detalhes do modelo, escolha Usar modelo e, em seguida, escolha Integrar API à nuvem.



2. Na seção de AWS CLIcomandos, copie o `start-model` AWS CLI comando.



3. Certifique-se de que o AWS CLI esteja configurado para ser executado na mesma AWS região em que você está usando o console Amazon Lookout for Vision. Para alterar a AWS região que o AWS CLI usa, consulteInstale os AWS SDKS.

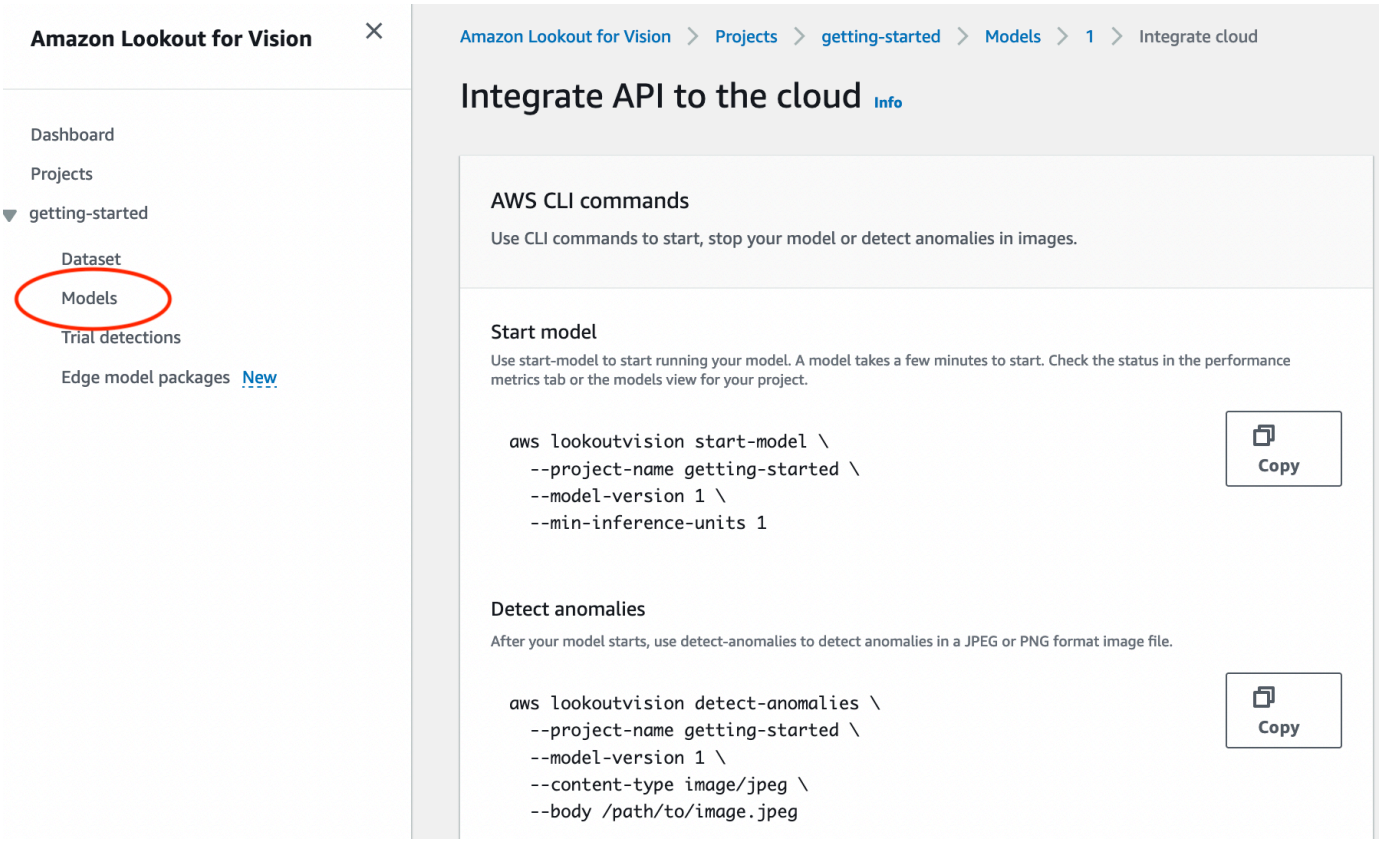
4. No prompt de comando, inicie o modelo digitando o `start-model` comando. Se você estiver usando o `lookoutvision` perfil para obter credenciais, adicione o `--profile lookoutvision-access` parâmetro. Por exemplo:

```
aws lookoutvision start-model \
  --project-name getting-started \
  --model-version 1 \
  --min-inference-units 1 \
  --profile lookoutvision-access
```

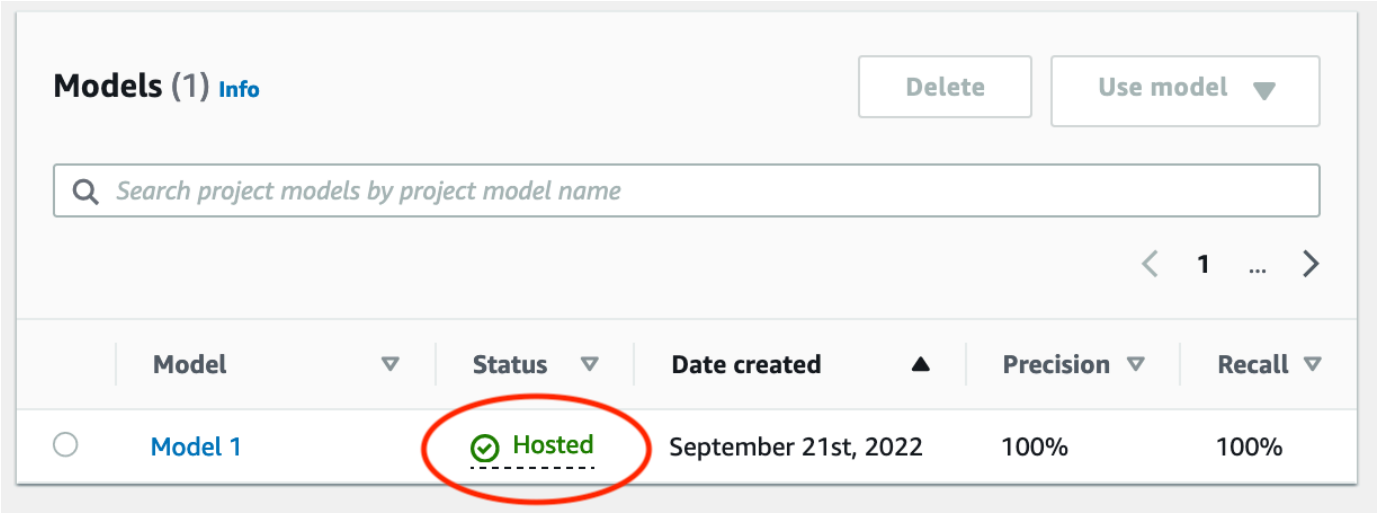
   Se a chamada tiver êxito, a seguinte saída será exibida:

```
{
    "Status": "STARTING_HOSTING"
}
```

5. De volta ao console do , escolha Instâncias no painel de navegação.



6. Espere até que o status do modelo (Modelo 1) na coluna Status exiba Hospedado. Se você já treinou um modelo no projeto, aguarde a conclusão da versão mais recente do modelo.

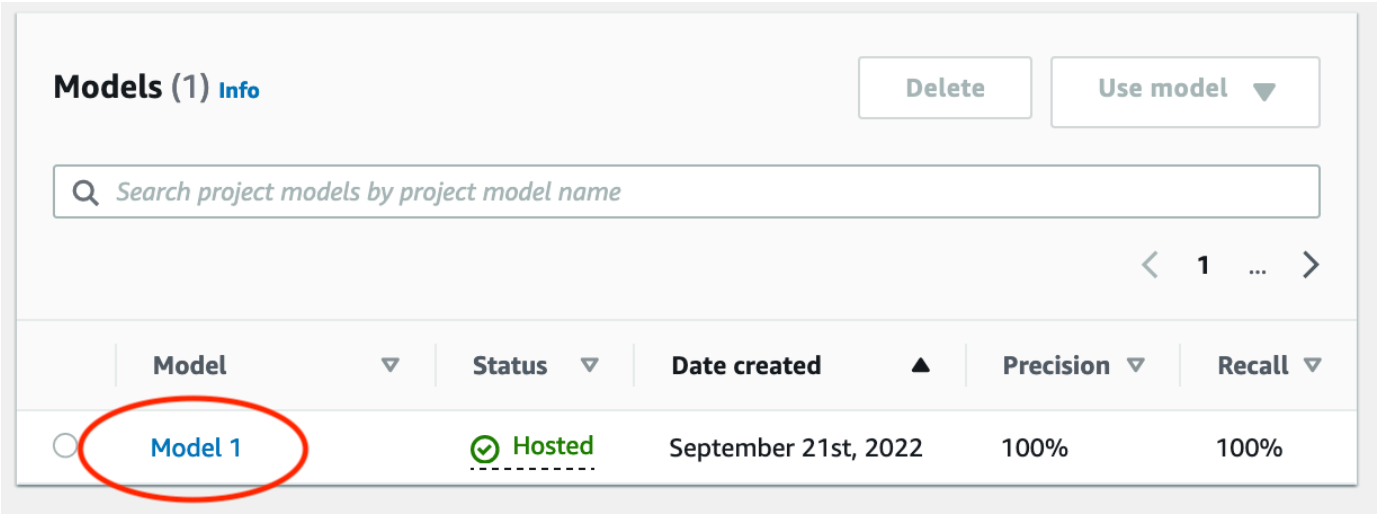


Etapa 4: Analisar uma imagem

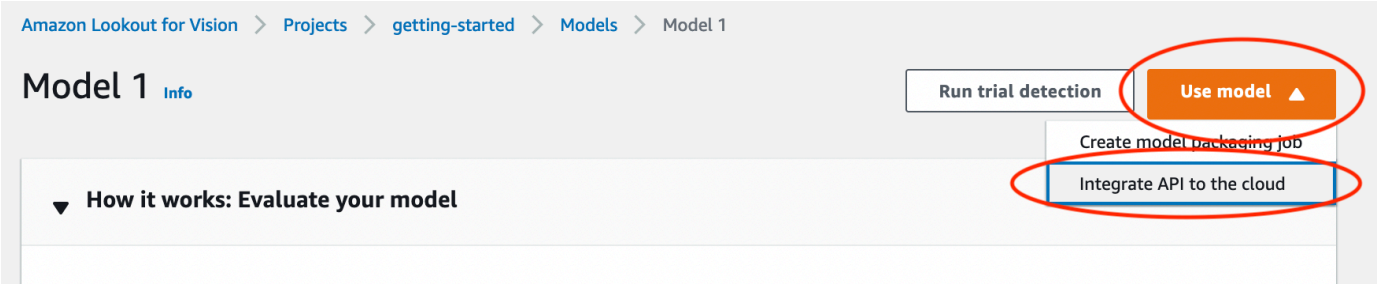
Nesta etapa, você analisa uma imagem com o modelo. Fornecemos exemplos de imagens que você pode usar na pasta de introdução `test-images` no repositório de documentação do Lookout for Vision em seu computador. Para obter mais informações, consulte Detectar as anomalias de uma imagem.

Como analisar uma imagem

1. Na página Modelos, escolha o modelo Modelo 1.



2. Na página de detalhes do modelo, escolha Usar modelo e, em seguida, escolha Integrar API à nuvem.



3. Na seção de AWS CLIcomandos, copie o `detect-anomalies` AWS CLI comando.

4. No prompt de comando, analise uma imagem anômala inserindo o `detect-anomalies` comando da etapa anterior. Para o `--body` parâmetro, especifique uma imagem anômala da `test-images` pasta de introdução do seu computador. Se você estiver usando o `lookoutvision` perfil para obter credenciais, adicione o `--profile lookoutvision-access` parâmetro. Por exemplo:

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/test-images/test-anomaly-1.jpg \
  --profile lookoutvision-access
```

A saída deve ser semelhante à seguinte:

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.983975887298584,
        "Anomalies": [
            {
```

```
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.9818974137306213,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "cracked",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.018102575093507767,
                    "Color": "#23A436"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAAMACA......"
    }
}
```

5. Observe o seguinte sobre a saída:

   - `IsAnomalous`é um booleano para a classificação prevista. `true`se a imagem for anômala, caso contrário. `false`
   - `Confidence` é um valor flutuante que representa a confiança que o Amazon Lookout for Vision tem na previsão. 0 é a menor confiança, 1 é a confiança mais alta.
   - `Anomalies` é uma lista de anomalias encontradas na imagem. `Name` é o rótulo da anomalia. `PixelAnomaly` inclui a área percentual total da anomalia (`TotalPercentageArea`) e uma cor (`Color`) para o rótulo da anomalia. A lista também inclui uma anomalia de "fundo" que cobre a área externa das anomalias encontradas na imagem.
   - `AnomalyMask` é uma imagem de máscara que mostra a localização das anomalias na imagem analisada.

   É possível usar as informações na resposta para exibir uma combinação da imagem analisada e da máscara de anomalia, conforme mostrado no exemplo a seguir. Para ver um código demonstrativo, consulte Exibindo informações de classificação e segmentação.


Classification:
Prediction: Anomalous
Confidence: 99.9%
Segmentation:
Anomaly: cracked. Area: 6.2%

6. No prompt de comando, analise uma imagem normal da `test-images` pasta de introdução. Se você estiver usando o `lookoutvision` perfil para obter credenciais, adicione o `--profile lookoutvision-access` parâmetro. Por exemplo:

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/test-images/test-normal-1.jpg \
  --profile lookoutvision-access
```

   A saída deve ser semelhante à seguinte:

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": false,
        "Confidence": 0.9916400909423828,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 1.0,
                    "Color": "#FFFFFF"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAA....."
    }
}
```

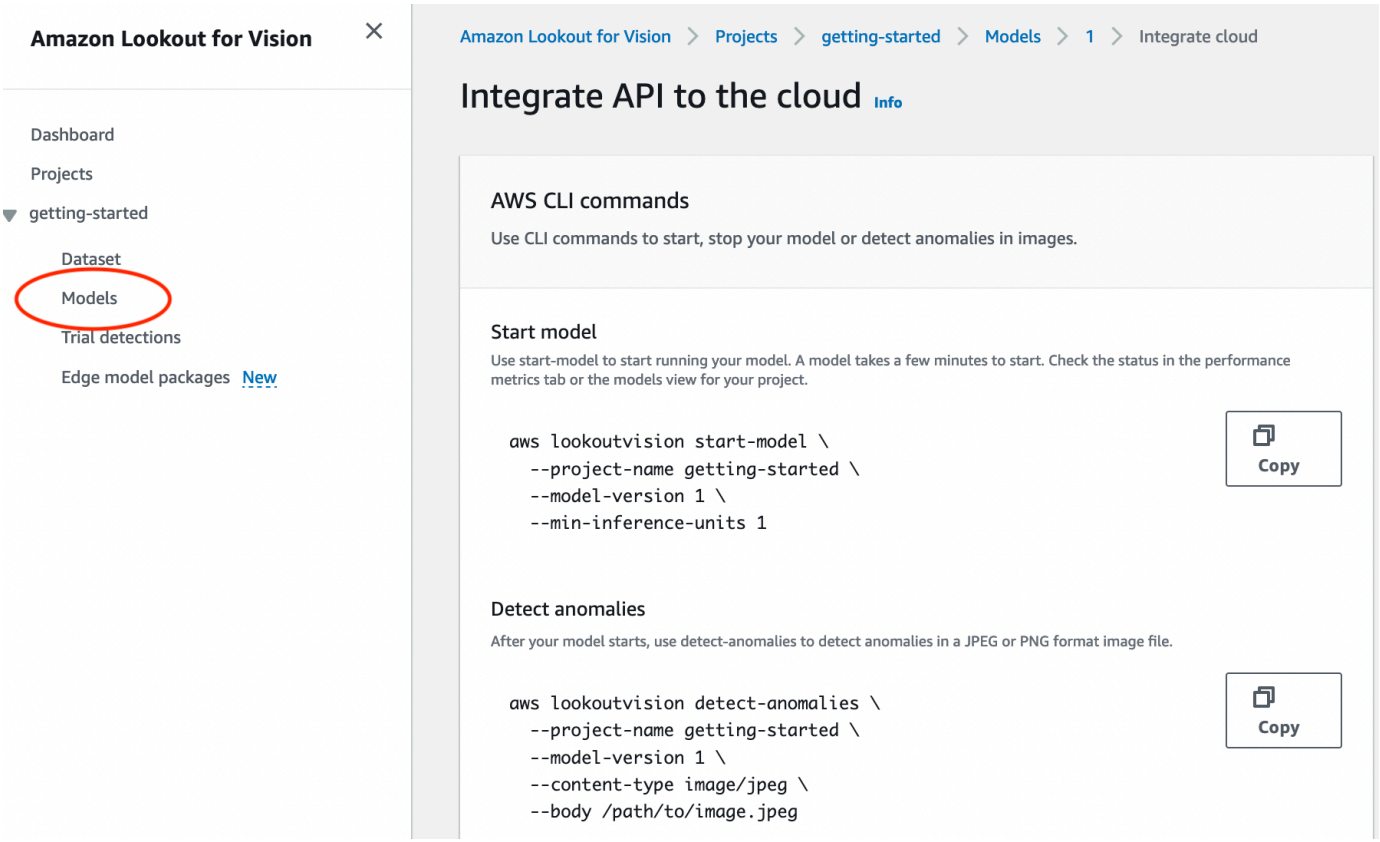
7. Na saída, observe que o `false` valor de `IsAnomalous` classifica a imagem como sem anomalias. Use `Confidence` para ajudar a decidir sua confiança na classificação. Além disso, a `Anomalies` matriz tem apenas o rótulo de `background` anomalia.

Etapa 5: interromper o modelo

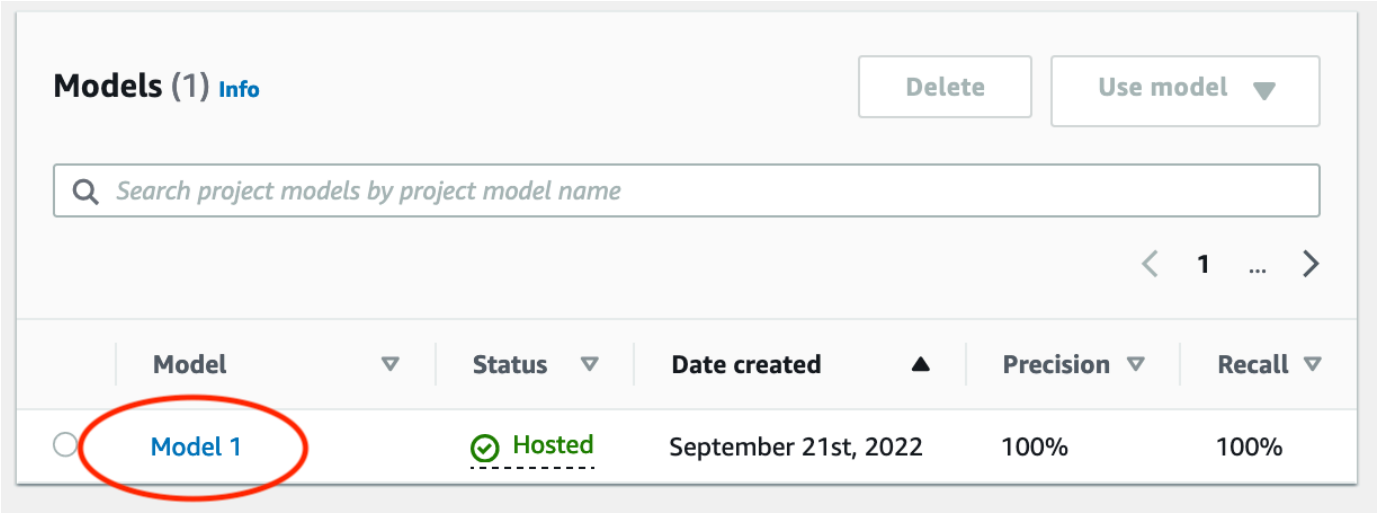
Nesta etapa, você deixa de hospedar o modelo. Você é cobrado pela quantidade de tempo em que seu modelo está em execução. Se você não estiver usando o modelo, interrompa. Você poderá reiniciar o modelo na próxima vez que precisar dele. Para obter mais informações, consulte Iniciar seu modelo do Amazon Lookout for Vision.

Como parar o modelo.

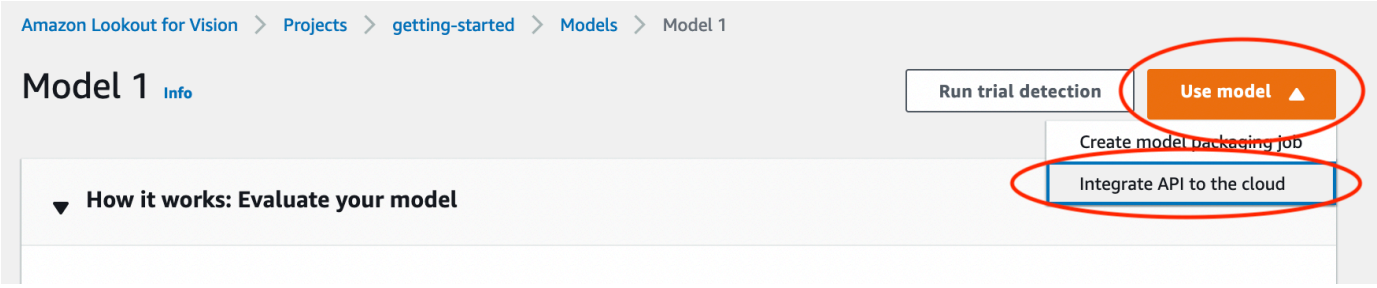
1. No painel de navegação, escolha Modelos.



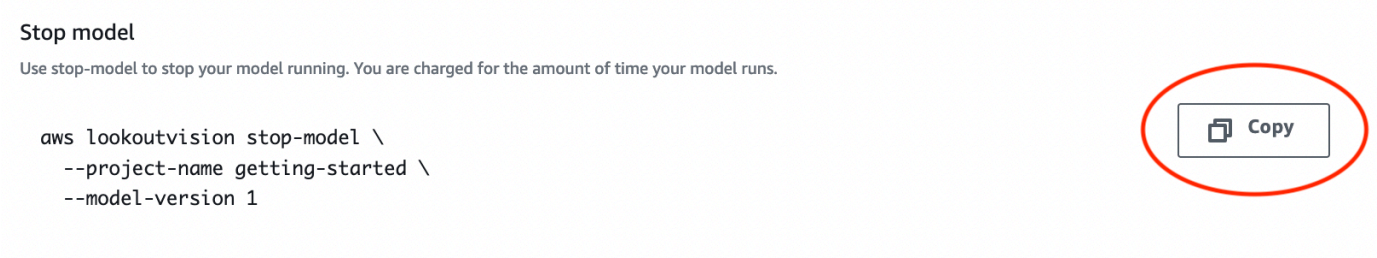
2. Na página Modelos, escolha o modelo Modelo 1.



3. Na página de detalhes do modelo, escolha Usar modelo e, em seguida, escolha Integrar API à nuvem.



4. Na seção de comandos AWS CLI, copie o comando `stop-model` AWS CLI.



5. No prompt de comando, pare o modelo inserindo o `stop-model` AWS CLI comando da etapa anterior. Se você estiver usando o `lookoutvision` perfil para obter credenciais, adicione o `--profile lookoutvision-access` parâmetro. Por exemplo:

```
aws lookoutvision stop-model \
  --project-name getting-started \
  --model-version 1 \
  --profile lookoutvision-access
```

Se a chamada tiver êxito, a seguinte saída será exibida:

```
{
    "Status": "STOPPING_HOSTING"
}
```

6. De volta ao console, escolha Modelos na página de navegação à esquerda.
7. O modelo foi interrompido quando o status do modelo na coluna Status é Treinamento concluído.

Próximas etapas

Quando estiver pronto para criar um modelo com suas próprias imagens, comece seguindo as instruções emCriar seu projeto. As instruções incluem etapas para criar um modelo com o console Amazon Lookout for Vision e com AWS o SDK.

Se você quiser experimentar outros exemplos de conjuntos de dados, consulte Exemplos de códigos e conjuntos de dados.

Criar o modelo do Amazon Lookout for Vision

Um modelo Amazon Lookout for Vision é um modelo de machine learning que prevê a presença de anomalias em novas imagens ao encontrar padrões nas imagens usadas para treinar o modelo. Esta seção mostra como criar e treinar um modelo. Depois de treinar o modelo, avalie seu desempenho. Para ter mais informações, consulte Melhoria de um modelo do Amazon Lookout for Vision.

Antes de criar seu primeiro modelo, recomendamos que você leia Noções básicas sobre o Amazon Lookout for Vision e Conceitos básicos do Amazon Lookout for Vision. Se você estiver usando o AWS SDK, leia Chame uma operação do Amazon Lookout for Vision.

Tópicos

Criar seu projeto

Um projeto do Amazon Lookout for Vision é um agrupamento dos recursos necessários para criar e gerenciar um modelo do Lookout for Vision. Um projeto gerencia o seguinte:

- Conjunto de dados: as imagens e os rótulos de imagem usados para treinar um modelo. Para ter mais informações, consulte Criar um conjunto de dados.
- Modelo: o software que você treina para detectar anomalias. Você pode ter várias versões de um modelo. Para ter mais informações, consulte Treinamento de seu modelo.

Recomendamos que você use um projeto para um único caso de uso, como detectar anomalias em um único tipo de peça da máquina.

> **Note**
>
> Você pode usar AWS CloudFormation para provisionar e configurar projetos do Amazon Lookout for Vision. Para ter mais informações, consulte Criação de recursos do Amazon Lookout for Vision com AWS CloudFormation.

É possível ver seus projetos em Visualizando seus projetos ou abrir o Usando o painel do Amazon Lookout for Vision. Para excluir um modelo, consulte Excluir um modelo.

Tópicos

Criar um projeto (console)

O procedimento a seguir mostra como criar um projeto usando o console.

Criar um projeto (console)

1. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. No painel de navegação à esquerda, escolha Projetos.
3. Escolha Criar projeto.
4. Em Nome do projeto, digite um nome para o seu projeto.
5. Escolha Criar projeto. A página de detalhes do projeto é exibida.
6. Siga as etapas em Criar um conjunto de dados para criar seu conjunto de dados.

Criar um projeto (SDK)

Você usa a CreateProjectoperação para criar um projeto Amazon Lookout for Vision. A resposta de `CreateProject` inclui o nome do projeto e o nome do recurso da Amazon (ARN) do projeto. Depois, ligue CreateDatasetpara adicionar um treinamento e um conjunto de dados de teste ao seu projeto. Para ter mais informações, consulte Criação de um conjunto de dados com um arquivo de manifesto (SDK).

Para ver os projetos que você criou em um projeto, chame `ListProjects`. Para ter mais informações, consulte Visualizando seus projetos.

Para criar um projeto (SDK)

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

   CLI

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

   ```
   aws lookoutvision create-project --project-name project name \
     --profile lookoutvision-access
   ```

   Python

   Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

   ```
       @staticmethod
       def create_project(lookoutvision_client, project_name):
           """
           Creates a new Lookout for Vision project.

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param  project_name: The name for the new project.
           :return project_arn: The ARN of the new project.
           """
           try:
               logger.info("Creating project: %s", project_name)
               response =
   lookoutvision_client.create_project(ProjectName=project_name)
               project_arn = response["ProjectMetadata"]["ProjectArn"]
               logger.info("project ARN: %s", project_arn)
           except ClientError:
               logger.exception("Couldn't create project %s.", project_name)
               raise
           else:
               return project_arn
   ```

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
/**
 * Creates an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return ProjectMetadata Metadata information about the created project.
 */
public static ProjectMetadata createProject(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {

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

        CreateProjectResponse response =
 lfvClient.createProject(createProjectRequest);

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

        return response.projectMetadata();

}
```

3. Siga as etapas em Criação de um conjunto de dados usando um arquivo de manifesto Amazon SageMaker Ground Truth para criar seu conjunto de dados.

Criar um conjunto de dados

Um conjunto de dados contém as imagens e os rótulos atribuídos que você usa para treinar e testar um modelo. Você cria o conjunto de dados do seu projeto com o console Amazon Lookout for Vision

ou com CreateDataseta operação. As imagens do conjunto de dados devem ser rotuladas de acordo com o tipo de modelo que você deseja criar (classificação de imagem ou segmentação de imagem).

Tópicos

Preparando imagens para um conjunto de dados

Você precisa de uma coleção de imagens para criar um conjunto de dados. Suas imagens devem ser arquivos no formato PNG ou JPEG. O número e o tipo de imagens de que você precisa dependem se seu projeto tem um único conjunto de dados ou conjuntos de dados de treinamento e teste separados.

Exclui um conjunto de dados

Para criar um modelo de classificação de imagens, você precisa do seguinte para começar o treinamento:

- Pelo menos 20 imagens de objetos normais.
- Pelo menos 10 imagens de objetos anômalos.

Para criar um modelo de segmentação de imagens, você precisa do seguinte para começar o treinamento:

- Pelo menos 20 imagens de cada tipo de anomalia.
- Cada imagem anômala (imagem com tipos de anomalia presentes) deve ter somente um tipo de anomalia.
- Pelo menos 20 imagens de objetos normais.

Projeto separado de conjunto de dados de treinamento e teste

Para criar um modelo de classificação de imagens, você precisa do seguinte:

- Pelo menos 10 imagens de objetos normais no conjunto de dados de treinamento.
- Pelo menos 10 imagens de objetos normais no conjunto de dados de teste.
- Pelo menos 10 imagens de objetos anômalos no conjunto de dados de teste.

Para criar um modelo de segmentação de imagens, você precisa do seguinte:

- Cada conjunto de dados precisa de pelo menos 10 imagens de cada tipo de anomalia.
- Cada imagem anômala (imagem com tipos de anomalia presentes) deve conter somente um tipo de anomalia.
- Cada conjunto de dados deve ter pelo menos 10 imagens de objetos normais.

Para criar um modelo de maior qualidade, use mais do que o número mínimo de imagens. Se você estiver criando um modelo de segmentação, recomendamos incluir imagens com vários tipos de anomalia, mas elas não são o mínimo que o Lookout for Vision precisa para iniciar o treinamento.

Suas imagens devem ser de um único tipo de objeto. Além disso, você deve ter condições de captura de imagem consistentes, como posicionamento da câmera, iluminação e pose do objeto.

Todas as imagens nos conjuntos de dados de treinamento e teste devem ter as mesmas dimensões. Posteriormente, as imagens que você analisa com seu modelo treinado devem ter as mesmas dimensões das imagens do conjunto de dados de treinamento e teste. Para ter mais informações, consulte Detectar as anomalias de uma imagem.

Todas as imagens de treinamento e teste devem ser imagens exclusivas, preferencialmente de objetos exclusivos. Imagens normais devem capturar as variações normais do objeto que está sendo analisado. Imagens anômalas devem capturar uma amostra diversificada de anomalias.

O Amazon Lookout for Vision oferece exemplos de imagens que você pode usar. Para ter mais informações, consulte Conjunto de dados de classificação de imagens.

Para ver os limites de imagem, consulte Cotas.

Criando o conjuntos de dados

Ao criar o conjunto de dados para seu projeto, você escolhe a configuração inicial do conjunto de dados do seu projeto. Você também escolhe de onde o Lookout for Vision importa as imagens.

Escolha de uma configuração de conjunto de dados para seu projeto

Ao criar o primeiro conjunto de dados em seu projeto, escolha uma das seguintes configurações de conjunto de dados:

- Conjunto de dados único: um único projeto de conjunto de dados usa um único conjunto de dados para treinar e testar o modelo. O uso de um único conjunto de dados simplifica o treinamento, permitindo que o Amazon Lookout for Vision escolha as imagens de treinamento e teste. Durante o treinamento, o Amazon Lookout for Vision divide internamente o conjunto de dados em um conjunto de dados de treinamento e um conjunto de dados de teste. Você não tem acesso aos conjuntos de dados divididos. Recomendamos usar um único projeto de conjunto de dados para a maioria dos cenários.
- Conjuntos de dados de treinamento e teste separados: se você quiser um controle mais preciso sobre treinamento, teste e ajuste de desempenho, você pode configurar seu projeto para ter conjuntos de dados de treinamento e teste separados. Use um conjunto de dados de teste separado se quiser controlar as imagens usadas para o teste ou se já tiver um conjunto de imagens de referência que deseja usar.

Você pode adicionar um conjunto de dados de teste a um projeto de conjunto de dados único existente. O único conjunto de dados então se torna o conjunto de dados de treinamento. Se você remover o conjunto de dados de teste de um projeto com conjuntos de dados de treinamento e teste separados, o projeto se tornará um projeto de conjunto de dados único. Para ter mais informações, consulte Excluir um conjunto de dados.

Importar imagens

Ao criar um conjunto de dados, você escolhe de onde importar as imagens. Dependendo de como você importa as imagens, elas podem já estar rotuladas. Se as imagens não forem rotuladas após a criação do conjunto de dados, consulte Rotulagem de imagens.

Você cria um conjunto de dados e importa suas imagens de uma das seguintes maneiras:

- Importe imagens do computador local. As imagens não estão rotuladas. Você adiciona ou rotula usando o console Lookout for Vision.

- Importe imagens de um bucket do S3. O Amazon Lookout for Vision pode classificar imagens usando os nomes das pastas para rotular as imagens. Use `normal` para imagens normais. Use `anomaly` para imagens anômalas. Você não pode atribuir rótulos de segmentação automaticamente.
- Importe um arquivo de manifesto do Amazon SageMaker Ground Truth, que inclui imagens rotuladas. Você pode criar e importar seu próprio arquivo de manifesto. Se você tiver muitas imagens, considere usar o serviço de rotulagem SageMaker Ground Truth. Em seguida, você importa o arquivo de manifesto de saída do trabalho do Amazon SageMaker Ground Truth. Se necessário, é possível usar o console Lookout for Vision para adicionar ou alterar rótulos.

Se você estiver usando o AWS SDK, você cria um conjunto de dados com um arquivo de manifesto Amazon SageMaker Ground Truth. Para ter mais informações, consulte Criação de um conjunto de dados usando um arquivo de manifesto Amazon SageMaker Ground Truth.

Se, depois de criar seu conjunto de dados, suas imagens forem rotuladas, você poderá treinar o modelo. Se as imagens não estiverem rotuladas, adicione as etiquetas de acordo com o tipo de modelo que você deseja criar. Para ter mais informações, consulte Rotulagem de imagens.

Você pode adicionar mais imagens a um conjunto de dados existente. Para ter mais informações, consulte Adicionar imagens ao seu conjunto de dados.

Criação de um conjunto de dados usando imagens armazenadas em seu computador local

Você pode criar um conjunto de dados usando imagens que são carregadas diretamente do seu computador. Você pode fazer upload de até 30 imagens por vez. Nesse procedimento, você pode criar um único projeto de conjunto de dados ou um projeto com conjuntos de dados de treinamento e teste separados.

> **Note**
>
> Se você acabou de concluir Criar seu projeto, o console deve mostrar o painel do projeto e não é necessário seguir as etapas de 1 a 3.

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

1. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. No painel de navegação à esquerda, escolha Projetos.
3. Na página Projetos, escolha o projeto ao qual você deseja adicionar um conjunto de dados.
4. Na página de detalhes do projeto, escolha Criar conjunto de dados.
5. Escolha a guia Conjunto de dados único ou a guia Conjuntos de dados de treinamento e teste separados e siga as etapas.

   Single dataset

   a. Na seção Configuração do conjunto de dados, escolha Criar um único conjunto de dados.
   b. Na seção Configuração da origem da imagem, escolha Fazer upload de imagens do computador.
   c. Escolha Criar conjunto de dados.
   d. Na página do conjunto de dados, escolha Adicionar imagens.
   e. Escolha as imagens que você deseja carregar no conjunto de dados a partir dos arquivos do seu computador. É possível arrastar as imagens ou escolher as imagens que deseja carregar do seu computador local.
   f. Escolha Fazer upload de imagens.

   Separate training and test datasets

   a. Na seção Configuração do conjunto de dados, escolha Criar um conjunto de dados de treinamento e um conjunto de dados de teste.
   b. Na seção Detalhes do conjunto de dados de treinamento, escolha Fazer upload de imagens do computador.
   c. 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.

d. Escolha Criar conjunto de dados. Uma página do conjunto de dados aparece com uma guia Treinamento e uma guia Teste para os respectivos conjuntos de dados.
e. Escolha Ações e Adicionar imagens ao conjunto de dados de treinamento.
f. 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.
g. Escolha Fazer upload de imagens.
h. Repita as etapas de 5e a 5g. Para a etapa 5e, escolha Ações e escolha Adicionar imagens ao conjunto de dados de teste.

6. Siga as etapas em Rotulagem de imagens para rotular suas imagens.
7. Siga as etapas em Treinamento de seu modelo para treinar o modelo.

Criar um conjunto de dados usando imagens armazenadas em um bucket do Amazon S3

Você pode criar um conjunto de dados usando imagens armazenadas em um bucket do Amazon S3. Com essa opção, você pode usar a estrutura de pastas em seu bucket do Amazon S3 para classificar automaticamente suas imagens. Você pode armazenar as imagens no bucket do console ou em outro bucket do Amazon S3 em sua conta.

Configurando pastas para etiquetagem automática

Durante a criação do conjunto de dados, você pode 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 que você especifica na URI do S3 ao criar o conjunto de dados.

Veja a seguir a pasta `train` das imagens de exemplo de Introdução. Se você especificar o local da pasta do Amazon S3 como `S3-bucket/circuitboard/train/`, as imagens na pasta `normal` receberão o rótulo `Normal`. As imagens na pasta `anomaly` recebem o rótulo `Anomaly`. Os nomes das pastas secundárias mais profundas não são usados para rotular imagens.

```
S3-bucket
  ### circuitboard
    ### train
        ### anomaly
            ### train-anomaly_1.jpg
            ### train-anomaly_2.jpg
```

```
        ### .
        ### .
    ### normal
        ### train-normal_1.jpg
        ### train-normal_2.jpg
        ### .
        ### .
```

Criar um conjunto de dados usando imagens de um bucket do Amazon S3

O procedimento a seguir cria um conjunto de dados usando as imagens de exemplo de classificação armazenadas em um bucket do Amazon S3. Para usar suas próprias imagens, crie a estrutura de pastas descrita em Configurando pastas para etiquetagem automática.

O procedimento também mostra como criar um único projeto de conjunto de dados ou um projeto que usa conjuntos de dados de treinamento e teste separados.

Se você não optar por rotular automaticamente suas imagens, precisará rotular as imagens após a criação dos conjuntos de dados. Para ter mais informações, consulte Classificação de imagens (console).

> **Note**
>
> Se você acabou de concluir Criar seu projeto, o console deve mostrar o painel do projeto e não é necessário seguir as etapas de 1 a 4.

Para criar um conjunto de dados usando imagens armazenadas em um bucket do Amazon S3

1. Se ainda não tiver feito isso, faça o upload das imagens de introdução para o bucket do Amazon S3. Para ter mais informações, consulte Conjunto de dados de classificação de imagens.
2. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
3. No painel de navegação à esquerda, escolha Projetos.
4. Na página Projetos, selecione o projeto ao qual você deseja adicionar um conjunto de dados. A página de detalhes do seu projeto é exibida.
5. Escolha Criar conjunto de dados. A página Criar conjunto de dados é exibida.

> **Tip**
>
> Se você estiver seguindo as instruções de Introdução, escolha Criar um conjunto de dados de treinamento e um conjunto de dados de teste.

6. Escolha a guia Conjunto de dados único ou a guia Conjuntos de dados de treinamento e teste separados e siga as etapas.

   Single dataset

   a. Na seção Configuração do conjunto de dados, escolha Criar um único conjunto de dados.

   b. Insira as informações das etapas de 7 a 9 na seção Configuração da origem da imagem.

   Separate training and test datasets

   a. Na seção Configuração do conjunto de dados, escolha Criar um conjunto de dados de treinamento e um conjunto de dados de teste.

   b. Para o conjunto de dados de treinamento, insira as informações das etapas de 7 a 9 na seção Detalhes do conjunto de dados de treinamento.

   c. Para o conjunto de dados de teste, insira as informações das etapas de 7 a 9 na seção Detalhes do conjunto de dados de teste.

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

7. Escolha Importar imagens do bucket do Amazon S3.

8. Em URI do S3, insira a localização do bucket do Amazon S3 e o caminho da pasta. Altere o `bucket` para o nome do seu bucket do Amazon S3.

   a. Se você estiver criando um único projeto de conjunto de dados ou um conjunto de dados de treinamento, insira o seguinte:

   ```
   s3://bucket/circuitboard/train/
   ```

   b. Se você estiver criando um conjunto de dados de teste, insira o seguinte:

```
s3://bucket/circuitboard/test/
```

9. Escolha Anexar rótulos automaticamente às imagens com base na pasta.
10. Escolha Criar grupo de conjuntos de dados. Uma página do conjunto de dados é aberta com suas imagens rotuladas.
11. Siga as etapas em Treinamento de seu modelo para treinar o modelo.

Criação de um conjunto de dados usando um arquivo de manifesto Amazon SageMaker Ground Truth

Um arquivo de manifesto contém informações sobre as imagens e os rótulos das imagens que você pode usar para treinar e testar um modelo. Você pode armazenar um arquivo de manifesto em um bucket do Amazon S3 e usá-lo para criar um conjunto de dados. Você pode criar seu próprio arquivo de manifesto ou usar um arquivo de manifesto existente, como a saída de um trabalho do Amazon SageMaker Ground Truth.

Tópicos

- Usando um trabalho do Amazon Sagemaker Ground Truth
- Criar um arquivo de manifesto

Usando um trabalho do Amazon Sagemaker Ground Truth

A rotulagem de imagens pode levar um tempo significativo. Por exemplo, pode levar 10 segundos para desenhar com precisão uma máscara ao redor de uma anomalia. Se você tiver centenas de imagens, pode levar várias horas para rotulá-las. Como alternativa a rotular as imagens você mesmo, considere usar o Amazon SageMaker Ground Truth.

Com o Amazon SageMaker Ground Truth, você pode usar funcionários da Amazon Mechanical Turk, uma empresa fornecedora de sua escolha, ou de uma força de trabalho interna e privada para criar um conjunto rotulado de imagens. Para obter mais informações, consulte Use o Amazon SageMaker Ground Truth para rotular dados.

Há um custo para usar o Amazon Mechanical Turk. Além disso, pode levar vários dias para concluir um trabalho de etiquetagem do Amazon Ground Truth. Se o custo for um problema ou se você precisar treinar seu modelo rapidamente, recomendamos que você use o console do Amazon Lookout for Vision para rotular as imagens.

Você pode usar uma tarefa de etiquetagem SageMaker do Amazon Ground Truth para rotular imagens adequadas para modelos de classificação de imagens e modelos de segmentação de imagens. Após a conclusão do trabalho, você usa o arquivo de manifesto de saída para criar um conjunto de dados do Amazon Lookout for Vision.

Classificação de imagens

Para rotular imagens de um modelo de classificação de imagens, crie um trabalho de rotulagem para uma tarefa de Classificação de imagens (etiqueta única).

Segmentação de imagens

Para rotular imagens para um modelo de segmentação de imagens, crie uma tarefa de rotulagem para uma tarefa de classificação de imagens (etiqueta única). Depois, encadeie o trabalho para criar um trabalho de rotulagem para uma tarefa de segmentação semântica de imagem.

Você também pode usar um trabalho de rotulagem para criar um arquivo de manifesto parcial para um modelo de segmentação de imagem. Por exemplo, você pode classificar imagens com uma tarefa de Classificação de Imagem (Etiqueta Única). Depois de criar um conjunto de dados do Lookout for Vision com a saída do trabalho, use o console do Amazon Lookout for Vision para adicionar máscaras de segmentação e rótulos de anomalias às imagens do conjunto de dados.

Rotulando imagens com o Amazon SageMaker Ground Truth

O procedimento a seguir mostra como rotular imagens com as tarefas de rotulagem de imagens SageMaker do Amazon Ground Truth. O procedimento cria um arquivo de manifesto de classificação de imagens e, opcionalmente, encadeia a tarefa de rotulagem de imagens para criar um arquivo de manifesto de segmentação de imagem. Se você quiser que seu projeto tenha um conjunto de dados de teste separado, repita esse procedimento para criar o arquivo de manifesto para o conjunto de dados de teste.

Para rotular imagens com o Amazon SageMaker Ground Truth (Console)

1. Crie um trabalho do Ground Truth para uma tarefa de Classificação de imagem (rótulo único) seguindo as instruções em Create a Labeling Job (Console).

    a. Para a etapa 10, escolha Imagem no menu suspenso Categoria de tarefa e selecione Classificação de imagem (rótulo único) como o tipo de tarefa.

    b. Para a etapa 16, na seção Ferramenta de rotulagem de classificação de imagem (rótulo único), adicione duas etiquetas: normal e anomalia.

2. Espere até que a força de trabalho termine de classificar suas imagens.
3. Se você estiver criando um conjunto de dados para um modelo de segmentação de imagens, faça o seguinte. Do contrário, vá para a etapa 4.

   a. No console do Amazon SageMaker Ground Truth, abra a página Labeling jobs.
   b. Selecione o trabalho que você criou anteriormente. Isso habilita o menu Ações.
   c. No menu Ações, escolha Cadeia. A página de detalhes do trabalho é aberta.
   d. Em Tipo de tarefa, escolha segmentação semântica.
   e. Escolha Próximo.
   f. Na seção Ferramenta de rotulagem de segmentação semântica, adicione rótulos de anomalia para cada tipo de anomalia que você deseja que o modelo encontre.
   g. Escolha Criar.
   h. Espere até que a força de trabalho rotule suas imagens.
4. Abra o console do Ground Truth e acesse a página Trabalhos de rotulagem.
5. Se estiver criando um modelo de classificação de imagem, escolha o trabalho criado na etapa 1. Se você estiver criando um modelo de segmentação de imagem, escolha o trabalho criado na etapa 3.
6. Em Resumo de trabalho de rotulagem, abra o local do S3 em Local de saída do conjunto de dados. Anote a localização do arquivo de manifesto, que deveria ser `s3://`*`output-dataset-location`*`/manifests/output/output.manifest`.
7. Repita esse procedimento se quiser criar um arquivo de manifesto para um conjunto de dados de teste. Caso contrário, siga as instruções em Criando o conjuntos de dados para criar um conjunto de dados com o arquivo de manifesto.

Criando o conjuntos de dados

Use esse procedimento para criar um conjunto de dados em um projeto do Lookout for Vision com o arquivo de manifesto que você anotou na etapa 6 de Rotulando imagens com o Amazon SageMaker Ground Truth. O arquivo de manifesto cria o conjunto de dados de treinamento para um único projeto de conjunto de dados. Se você quiser que seu projeto tenha um conjunto de dados de teste separado, você pode executar outro trabalho do Amazon SageMaker Ground Truth para criar um arquivo de manifesto para o conjunto de dados de teste. Ou você pode criar o arquivo de manifesto por conta própria. Você também pode importar imagens para seu conjunto de dados de teste a partir de um bucket do Amazon S3 ou do seu computador local. (As imagens podem precisar ser rotuladas antes que você possa treinar o modelo).

Esse procedimento pressupõe que seu projeto não tenha conjuntos de dados.

Para criar um conjunto de dados com o Lookout for Vision (Console)

1. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos.
4. Escolha o projeto que você deseja adicionar para usar com o arquivo de manifesto.
5. Na seção Como funciona, escolha Criar conjunto de dados.
6. Escolha a guia Conjunto de dados único ou a guia Conjuntos de dados de treinamento e teste separados e siga as etapas.

   Single dataset

   1. Escolha Criar um único conjunto de dados.
   2. Na seção Configuração da fonte de imagem, escolha Importar imagens rotuladas por SageMaker Ground Truth.
   3. Em Localização do arquivo .manifest, insira a localização do arquivo de manifesto que você anotou na etapa 6 de Rotulando imagens com o Amazon SageMaker Ground Truth.

   Separate training and test datasets

   1. Escolha Criar um conjunto de dados de treinamento e um conjunto de dados de teste.
   2. Na seção Detalhes do conjunto de dados de treinamento, escolha Importar imagens rotuladas por SageMaker Ground Truth.
   3. Em Localização do arquivo .manifest, a localização do arquivo de manifesto que você anotou na etapa 6 de Rotulando imagens com o Amazon SageMaker Ground Truth.
   4. Na seção Detalhes do conjunto de dados de teste, escolha Importar imagens rotuladas por SageMaker Ground Truth.
   5. Em Localização do arquivo .manifest, a localização do arquivo de manifesto que você anotou na etapa 6 de Rotulando imagens com o Amazon SageMaker Ground Truth. Lembre-se de que você precisa de um arquivo de manifesto separado para o conjunto de dados de teste.
7. Selecione Enviar.

8. Siga as etapas em Treinamento de seu modelo para treinar o modelo.

Criar um arquivo de manifesto

Você pode criar um conjunto de dados importando um arquivo de manifesto no formato SageMaker Ground Truth. Se suas imagens estiverem rotuladas em um formato que não seja um arquivo de manifesto SageMaker Ground Truth, use as informações a seguir para criar um arquivo de manifesto no formato SageMaker 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. Existem diferentes formatos para classificação e segmentação de imagens. Os arquivos de manifesto devem ser codificados usando a codificação UTF-8.

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

As imagens referenciadas por um arquivo de manifesto devem estar localizadas no mesmo bucket do Amazon S3. O arquivo de manifesto pode estar em um bucket diferente. Você especifica a localização de uma imagem no campo `source-ref` de uma linha JSON.

Você pode criar um arquivo de manifesto usando código. O notebook Python do Amazon Lookout for Vision Lab mostra como criar um arquivo de manifesto de classificação de imagens para as imagens de exemplo da placa de circuito. Como alternativa, você pode usar o código de exemplo de conjuntos de dados no Repositório de exemplos de AWS código. Você pode criar facilmente um arquivo de manifesto usando um arquivo de valores separados por vírgula (CSV). Para ter mais informações, consulte Criar um arquivo de manifesto de classificação de um arquivo CSV.

Tópicos

- Definindo linhas JSON para classificação de imagens
- Definindo linhas JSON para segmentação de imagens
- Criar um arquivo de manifesto de classificação de um arquivo CSV
- Criação de um conjunto de dados com um arquivo de manifesto (console)
- Criação de um conjunto de dados com um arquivo de manifesto (SDK)

Definindo linhas JSON para classificação de imagens

Você define uma linha JSON para cada imagem que deseja usar em um arquivo de manifesto do Amazon Lookout for Vision. Se você quiser criar um modelo de classificação, a linha JSON deve incluir uma classificação de imagem que seja normal ou uma anomalia. Uma linha JSON está no formato SageMaker Ground Truth Classification Job Output. Um arquivo de manifesto é feito de uma ou mais linhas JSON, uma para cada imagem que você deseja importar.

Para criar um arquivo de manifesto para imagens classificadas

1. Crie um arquivo de texto vazio.
2. Como importar mais de uma vez, você pode criar uma nova versão. Cada linha JSON deve ser semelhante à seguinte:

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

3. Salve o arquivo.

> **Note**
>
> 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 ter mais informações, consulte Criar um arquivo de manifesto.

Linhas de classificação JSON

Nesta seção, você aprenderá a criar uma linha JSON que classifica uma imagem como normal ou anômala.

Usar a detecção de anomalias

A linha JSON a seguir mostra uma imagem rotulada como uma anomalia. Observe que o valor de `class-name` é `anomaly`.

```
{
    "source-ref": "s3: //bucket/image/anomaly/abnormal-1.jpg",
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/auto-label",
        "class-name": "anomaly",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.600",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 1
}
```

Linha JSON normal

A linha JSON a seguir mostra uma imagem rotulada como normal. Observe que o valor de `class-name` é `normal`.

```
{
    "source-ref": "s3: //bucket/image/normal/2020-10-20_12-14-55_613.jpeg",
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/auto-label",
        "class-name": "normal",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.603",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 0
}
```

Chaves e valores de linha JSON

As informações a seguir descrevem as chaves e os valores em uma linha JSON do Amazon Lookout for Vision.

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.

Descrever detectores de anomalias

(Obrigatório) O nome do atributo a ser usado. Use a chave `anomaly-label` ou o nome de outra chave que você escolher. O valor da chave (`0` no exemplo anterior) é exigido pelo Amazon Lookout for Vision, mas não é usado. O manifesto de saída criado pelo Amazon Lookout for Vision converte o valor para `1` para uma imagem anômala e um valor de `0` de para uma imagem normal. O valor de `class-name` determina se a imagem é normal ou anômala.

Deve haver metadados correspondentes identificados pelo nome do campo com -metadata anexado. Por exemplo, `"anomaly-label-metadata"`.

metadados de rótulo de anomalia

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

confiança

(Opcional) No momento, não usado pelo Amazon Lookout for Vision. Se você especificar um valor, use um valor de `1`.

job-name

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

Nome da classe

(Obrigatório) Se a imagem contiver conteúdo normal, especifique `normal`, caso contrário, especifique `anomaly`. Se o valor de `class-name` for qualquer outro valor, a imagem será adicionada ao conjunto de dados como uma imagem sem rótulo. Para rotular uma imagem, consulte Adicionar imagens ao seu conjunto de dados.

human-annotated

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

Data de criação

(Opcional) 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 de anomalia no nível da imagem, o valor é `"groundtruth/image-classification"`.

Definindo linhas JSON para segmentação de imagens

Você define uma linha JSON para cada imagem que deseja usar em um arquivo de manifesto do Amazon Lookout for Vision. Se você quiser criar um modelo de segmentação, a linha JSON deve incluir informações de segmentação e classificação da imagem. Um arquivo de manifesto é feito de uma ou mais linhas JSON, uma para cada imagem que você deseja importar.

Para criar um arquivo de manifesto para imagens segmentadas

1. Crie um arquivo de texto vazio.
2. Como importar mais de uma vez, você pode criar uma nova versão. Cada linha JSON deve ser semelhante à seguinte:

```
{"source-ref":"s3://path-to-image","anomaly-label":1,"anomaly-label-metadata":
{"class-name":"anomaly","creation-date":"2021-10-12T14:16:45.668","human-
annotated":"yes","job-name":"labeling-job/classification-job","type":"groundtruth/
image-classification","confidence":1},"anomaly-mask-ref":"s3://path-to-
image","anomaly-mask-ref-metadata":{"internal-color-map":{"0":{"class-
name":"BACKGROUND","hex-color":"#ffffff","confidence":0.0},"1":{"class-
name":"scratch","hex-color":"#2ca02c","confidence":0.0},"2":{"class-
name":"dent","hex-color":"#1f77b4","confidence":0.0}},"type":"groundtruth/
semantic-segmentation","human-annotated":"yes","creation-
date":"2021-11-23T20:31:57.758889","job-name":"labeling-job/segmentation-job"}}
```

3. Salve o arquivo.

> **Note**
>
> 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 ter mais informações, consulte Criar um arquivo de manifesto.

Linhas JSON de segmentação

Nesta seção, você aprenderá a criar uma linha JSON que inclui informações de segmentação e classificação de uma imagem.

A linha JSON a seguir mostra uma imagem com informações de segmentação e classificação. `anomaly-label-metadata` contém informações de classificação. `anomaly-mask-ref` e `anomaly-mask-ref-metadata` contêm informações de segmentação.

```
{
    "source-ref": "s3://path-to-image",
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "creation-date": "2021-10-12T14:16:45.668",
        "human-annotated": "yes",
        "job-name": "labeling-job/classification-job",
        "type": "groundtruth/image-classification",
        "confidence": 1
    },
    "anomaly-mask-ref": "s3://path-to-image",
    "anomaly-mask-ref-metadata": {
        "internal-color-map": {
            "0": {
                "class-name": "BACKGROUND",
                "hex-color": "#ffffff",
                "confidence": 0.0
            },
            "1": {
                "class-name": "scratch",
                "hex-color": "#2ca02c",
                "confidence": 0.0
            },
            "2": {
                "class-name": "dent",
                "hex-color": "#1f77b4",
                "confidence": 0.0
            }
        },
        "type": "groundtruth/semantic-segmentation",
        "human-annotated": "yes",
        "creation-date": "2021-11-23T20:31:57.758889",
        "job-name": "labeling-job/segmentation-job"
    }
}
```

Chaves e valores de linha JSON

As informações a seguir descrevem as chaves e os valores em uma linha JSON do Amazon Lookout for Vision.

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.

Descrever detectores de anomalias

(Obrigatório) O nome do atributo a ser usado. Use a chave `anomaly-label` ou o nome de outra chave que você escolher. O valor da chave (`1` no exemplo anterior) é exigido pelo Amazon Lookout for Vision, mas não é usado. O manifesto de saída criado pelo Amazon Lookout for Vision converte o valor para `1` para uma imagem anômala e um valor de `0` de para uma imagem normal. O valor de `class-name` determina se a imagem é normal ou anômala.

Deve haver metadados correspondentes identificados pelo nome do campo com -metadata anexado. Por exemplo, `"anomaly-label-metadata"`.

metadados de rótulo de anomalia

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

confiança

(Opcional) No momento, não usado pelo Amazon Lookout for Vision. Se você especificar um valor, use um valor de `1`.

job-name

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

Nome da classe

(Obrigatório) Se a imagem contiver conteúdo normal, especifique `normal`, caso contrário, especifique `anomaly`. Se o valor de `class-name` for qualquer outro valor, a imagem será adicionada ao conjunto de dados como uma imagem sem rótulo. Para rotular uma imagem, consulte Adicionar imagens ao seu conjunto de dados.

human-annotated

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

Data de criação

(Opcional) 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. Use o valor `"groundtruth/image-classification"`.

máscara de anomalia-ref

(Obrigatório) O local da imagem de máscara no Amazon S3. Use `anomaly-mask-ref` para o nome da chave ou use o nome de uma chave de sua escolha. A chave deve terminar com `-ref`. A imagem da máscara deve conter máscaras coloridas para cada tipo de anomalia `internal-color-map`. O formato é `"s3://`*`BUCKET`*`/`*`OBJECT_PATH`*`"`. As imagens em um conjunto de dados importado devem ser armazenadas no mesmo bucket do Amazon S3. A imagem da máscara deve ser uma imagem no formato Portable Network Graphic (PNG).

metadados de máscara de anomalia

(Obrigatório) Metadados de segmentação da imagem. Use `anomaly-mask-ref-metadata` para o nome da chave ou use o nome de uma chave de sua escolha. O nome da chave deve terminar com `-ref-metadata`.

internal-color-map

(Obrigatório) Um mapa de cores mapeado para tipos individuais de anomalias. As cores devem corresponder às cores na imagem da máscara (`anomaly-mask-ref`).

chave

(Obrigatório) A chave do mapa. A entrada `0` deve conter o nome da classe BACKGROUND que representa áreas fora das anomalias na imagem.

Nome da classe

(Obrigatório) O nome do tipo de anomalia, como arranhão ou amassado.

cor hexadecimal

(Obrigatório) A cor hexadecimal do tipo de anomalia, como `#2ca02c`. A cor deve corresponder a uma cor em `anomaly-mask-ref`. O valor do tipo de anomalia `BACKGROUND` é sempre `#ffffff`.

confiança

(Obrigatório) Atualmente não é usado pelo Amazon Lookout for Vision, mas é necessário um valor flutuante.

human-annotated

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

Data de criação

(Opcional) A data e hora do Tempo Universal Coordenado (UTC) em que as informações de segmentação foram criadas.

tipo

(Obrigatório) O tipo de processamento que deve ser aplicado à imagem. Use o valor `"groundtruth/semantic-segmentation"`.

Criar um arquivo de manifesto de classificação de um arquivo CSV

Esse exemplo de script Python simplifica a criação de um arquivo de manifesto de classificação usando um arquivo CSV (Comma Separated Values) para rotular imagens. Você cria o arquivo CSV.

Um arquivo de manifesto descreve as imagens usadas para treinar um modelo. Um arquivo de manifesto é composto por uma ou mais linhas JSON. Cada linha JSON descreve uma única imagem. Para ter mais informações, consulte Definindo linhas JSON para classificação de imagens.

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 Comma-separated values. Para esse script, cada linha no arquivo CSV inclui a localização de uma imagem no S3 e a classificação de anomalias da imagem (`normal` ou `anomaly`). Cada linha é mapeada para uma linha JSON no arquivo de manifesto.

Por exemplo, o arquivo CSV a seguir descreve algumas das imagens nas imagens de exemplo.

```
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg,anomaly
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train-normal_1.jpg,normal
s3://s3bucket/circuitboard/train/normal/train-normal_10.jpg,normal
s3://s3bucket/circuitboard/train/normal/train-normal_11.jpg,normal
```

O script gera linhas JSON para cada linha. Por exemplo, a seguir está a linha JSON para a primeira linha (`s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly`).

```
{"source-ref": "s3://s3bucket/csv_test/train_anomaly_1.jpg","anomaly-label":
 1,"anomaly-label-metadata": {"confidence": 1,"job-name": "labeling-job/anomaly-
classification","class-name": "anomaly","human-annotated": "yes","creation-date":
 "2022-02-04T22:47:07","type": "groundtruth/image-classification"}}
```

Se o 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 as imagens.

Antes de criar o arquivo de manifesto, o script verifica se há imagens duplicadas no arquivo CSV e em qualquer classificação de imagem que não seja `normal` ou `anomaly`. Se forem encontrados erros de imagem ou classificação de imagem duplicados, o script fará o seguinte:

- Registra a primeira entrada de imagem válida para todas as imagens em um arquivo CSV desduplicado.
- Registra ocorrências duplicadas de uma imagem no arquivo de erros.
- Registra as classificações de imagens que não sejam `normal` ou `anomaly` no arquivo de erros.
- Não cria um arquivo de manifesto.

O arquivo de erros inclui o número da linha em que uma imagem duplicada ou um erro de classificação é encontrado no arquivo CSV de entrada. Use o arquivo CSV de erros para atualizar o arquivo CSV de entrada e, em seguida, execute o script novamente. Como alternativa, use o arquivo CSV de erros para atualizar o arquivo CSV desduplicado, que contém somente entradas de imagem exclusivas e imagens sem erros de classificação de imagem. Execute novamente o script com o arquivo CSV desduplicado atualizado.

Se nenhuma duplicata ou erro for encontrado no arquivo CSV de entrada, o script excluirá o arquivo CSV de imagem desduplicada e o arquivo de erros, pois eles estão vazios.

Nesse procedimento, você cria o arquivo CSV e executa o script Python para criar o arquivo de manifesto. O script foi testado com a versão 3.7 do Python.

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
O nome da imagem ou o caminho do Amazon S3 para a imagem. Por exemplo, `s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg` . Você não pode ter uma mistura de imagens com o caminho do Amazon S3 e imagens sem.	A classificação da anomalia para a imagem (`normal` ou `anomaly`).

   Por exemplo, `s3://s3bucket/circuitboard/train/anomaly/image_10.jpg,anomaly` ou `image_11.jpg,normal`
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.
   - (Opcional) `--s3-path` *`s3://path_to_folder/`*: i 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
"""
Purpose
Shows how to create an Amazon Lookout for Vision manifest file from a CSV file.
The CSV file format is image location,anomaly classification (normal or anomaly)
For example:
s3://s3bucket/circuitboard/train/anomaly/train_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train_1.jpg,normal
```

```
If necessary, use the bucket argument to specify the Amazon S3 bucket folder for
 the images.
"""

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

logger = logging.getLogger(__name__)


def check_errors(csv_file):
    """
    Checks for duplicate images and incorrect classifications in a CSV file.
    If duplicate images or invalid anomaly assignments are found, an errors CSV
 file
    and deduplicated CSV file are created. Only the first
    occurrence of a duplicate is recorded. Other duplicates are recorded in the
 errors file.
    :param csv_file: The source CSV file
    :return: True if errors or duplicates are found, otherwise false.
    """

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

    errors_found = False
    errors_file = f"{os.path.splitext(csv_file)[0]}_errors.csv"
    deduplicated_file = f"{os.path.splitext(csv_file)[0]}_deduplicated.csv"

    with open(csv_file, 'r', encoding="UTF-8") as input_file,\
            open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
            open(errors_file, 'w', encoding="UTF-8") as errors:

        reader = csv.reader(input_file, delimiter=',')
        dedup_writer = csv.writer(dedup)
        error_writer = csv.writer(errors)
        line = 1
        entries = set()
        for row in reader:
```

```
            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            # Record any incorrect classifications.
            if not row[1].lower() == "normal" and not row[1].lower() == "anomaly":
                error_writer.writerow(
                    [line, row[0], row[1], "INVALID_CLASSIFICATION"])
                errors_found = True

            # Write first image entry to dedup file and record duplicates.
            key = row[0]
            if key not in entries:
                dedup_writer.writerow(row)
                entries.add(key)
            else:
                error_writer.writerow([line, row[0], row[1], "DUPLICATE"])
                errors_found = True
            line += 1

    if errors_found:
        logger.info("Errors found check %s.", errors_file)
    else:
        os.remove(errors_file)
        os.remove(deduplicated_file)

    return errors_found


def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Read a CSV file and create an Amazon Lookout for Vision 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 Amazon S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s.", csv_file)

    image_count = 0
    anomalous_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 the CSV file.
        for row in image_classifications:
            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            source_ref = str(s3_path) + row[0]
            classification = 0

            if row[1].lower() == 'anomaly':
                classification = 1
                anomalous_count += 1

           # Create the JSON line.
            json_line = {}
            json_line['source-ref'] = source_ref
            json_line['anomaly-label'] = str(classification)

            metadata = {}
            metadata['confidence'] = 1
            metadata['job-name'] = "labeling-job/anomaly-classification"
            metadata['class-name'] = row[1]
            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['anomaly-label-metadata'] = metadata

            output_file.write(json.dumps(json_line))
            output_file.write('\n')
            image_count += 1

    logger.info("Finished creating manifest file %s.\n"
                "Images: %s\nAnomalous: %s",
                manifest_file,
                image_count,
                anomalous_count)
    return image_count, anomalous_count
```

```
def add_arguments(parser):
    """
    Add 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 Amazon S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the Amazon 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 = ""

        csv_file = args.csv_file
        csv_file_no_extension = os.path.splitext(csv_file)[0]
        manifest_file = csv_file_no_extension + '.manifest'

        # Create manifest file if there are no duplicate images.
        if check_errors(csv_file):
            print(f"Issues found. Use {csv_file_no_extension}_errors.csv "\
                "to view duplicates and errors.")
            print(f"{csv_file}_deduplicated.csv contains the first"\
                "occurrence of a duplicate.\n"
                  "Update as necessary with the correct information.")
```

```
            print(f"Re-run the script with
 {csv_file_no_extension}_deduplicated.csv")
        else:
            print('No duplicates found. Creating manifest file.')

            image_count, anomalous_count = create_manifest_file(csv_file,
 manifest_file, s3_path)

            print(f"Finished creating manifest file: {manifest_file} \n")

            normal_count = image_count-anomalous_count
            print(f"Images processed: {image_count}")
            print(f"Normal: {normal_count}")
            print(f"Anomalous: {anomalous_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 ocorrerem imagens duplicadas ou ocorrerem erros de classificação:

   a. Use o arquivo de erros para atualizar o arquivo CSV desduplicado ou o arquivo CSV de entrada.

   b. Execute o script novamente com o arquivo CSV desduplicado atualizado ou o arquivo CSV de entrada atualizado.

5. Se você planeja usar um conjunto de dados de teste, repita as etapas de 1 a 4 para criar um arquivo de manifesto para seu conjunto de dados de teste.

6. Se necessário, copie as imagens do seu computador 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`). Para copiar as imagens, insira o comando a seguir no prompt de comando.

```
aws s3 cp --recursive your-local-folder/ s3://your-target-S3-location/
```

7. Siga as instruções em Criação de um conjunto de dados com um arquivo de manifesto (console) para criar um conjunto de dados. Se você estiver usando o AWS SDK, consulteCriação de um conjunto de dados com um arquivo de manifesto (SDK).

Criação de um conjunto de dados com um arquivo de manifesto (console)

O procedimento a seguir mostra como criar um conjunto de dados de treinamento ou teste importando um arquivo de manifesto em SageMaker formato armazenado em um bucket do Amazon S3.

Depois de criar o conjunto de dados, você pode adicionar mais imagens ao conjunto de dados ou adicionar rótulos às imagens. Para ter mais informações, consulte Adicionar imagens ao seu conjunto de dados.

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

1. Crie ou use um arquivo de manifesto existente no formato Ground Truth SageMaker compatível com Amazon Lookout for Vision. Para ter mais informações, consulte Criar um arquivo de manifesto.
2. Faça login AWS Management Console e abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
3. Em um bucket do Amazon S3, crie uma pasta para armazenar o arquivo de manifesto.
4. Faça upload do arquivo de manifesto na pasta que você acabou de criar.
5. No bucket do Amazon S3, crie uma pasta para armazenar as imagens.
6. Carregue suas imagens na pasta que você acabou de criar.

   > **Important**
   >
   > O valor do campo `source-ref` em cada linha JSON deve ser mapeado para imagens na pasta.

7. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
8. Escolha Comece a usar.
9. No painel de navegação à esquerda, escolha Projetos.
10. Escolha o projeto que você deseja adicionar para usar com o arquivo de manifesto.
11. Na seção Como funciona, escolha Criar conjunto de dados.
12. Escolha a guia Conjunto de dados único ou a guia Conjuntos de dados de treinamento e teste separados e siga as etapas.

Single dataset

1. Escolha Criar um único conjunto de dados.
2. Na seção Configuração da fonte de imagem, escolha Importar imagens rotuladas por SageMaker Ground Truth.
3. Na Localização do arquivo .manifest, insira a localização do arquivo de manifesto.

Separate training and test datasets

1. Escolha Criar um conjunto de dados de treinamento e um conjunto de dados de teste.
2. Na seção Detalhes do conjunto de dados de treinamento, escolha Importar imagens rotuladas por SageMaker Ground Truth.
3. Na Localização do arquivo .manifest, insira a localização do arquivo de manifesto de treinamento.
4. Na seção Detalhes do conjunto de dados de teste, escolha Importar imagens rotuladas por SageMaker Ground Truth.
5. Na Localização do arquivo .manifest, insira a localização do arquivo de manifesto de teste.

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

13. Selecione Enviar.
14. Siga as etapas em Treinamento de seu modelo para treinar o modelo.

O Amazon Lookout for Vision cria um conjunto de dados na pasta `datasets` do bucket do Amazon S3. O arquivo `.manifest` original permanece inalterado.

Criação de um conjunto de dados com um arquivo de manifesto (SDK)

Você usa a CreateDatasetoperação para criar os conjuntos de dados associados a um projeto Amazon Lookout for Vision.

Se você quiser usar um único conjunto de dados para treinamento e teste, crie um único conjunto de dados com o valor de `DatasetType` definido como `train`. Durante o treinamento, o conjunto

de dados é dividido internamente para criar um conjunto de dados de treinamento e teste. Você não tem acesso aos conjuntos de dados de treinamento e teste divididos. Se você quiser um conjunto de dados de teste separado, faça uma segunda chamada para `CreateDataset` com o valor de `DatasetType` definido como `test`. Durante o treinamento, os conjuntos de dados de treinamento e teste são usados para treinar e testar o modelo.

Opcionalmente, você pode usar o `DatasetSource` parâmetro para especificar a localização de um arquivo de manifesto no formato SageMaker Ground Truth usado para preencher o conjunto de dados. Nesse caso, a chamada para `CreateDataset` é assíncrona. Para verificar o status atual, chame `DescribeDataset`. Para ter mais informações, consulte Visualizar seus conjuntos de dados. Se ocorrer um erro de validação durante a importação, o valor de `Status` será definido como CREATE_FAILED e a mensagem de status (`StatusMessage`) será definida.

> **Tip**
>
> Se você estiver criando um conjunto de dados com o conjunto de dados de exemplo de introdução, use o arquivo de manifesto (`getting-started/dataset-files/manifests/train.manifest`) no qual o script cria. Etapa 1: criar o arquivo de manifesto e fazer upload de imagens
> Se você estiver criando um conjunto de dados com as imagens de exemplo da placa de circuito, você tem duas opções:
>
> 1. Crie o arquivo de manifesto usando código. O notebook Python do Amazon Lookout for Vision Lab mostra como criar o arquivo de manifesto para as imagens de exemplo da placa de circuito. Como alternativa, use o código de exemplo de conjuntos de dados no Repositório de exemplos de AWS código.
> 2. Se você já usou o console Amazon Lookout for Vision para criar um conjunto de dados com as imagens de exemplo da placa de circuito, reutilize os arquivos de manifesto criados para você pelo Amazon Lookout for Vision. Os locais dos arquivos do manifesto de treinamento e teste sãos3://*bucket*/datasets/*project name*/*train or test*/manifests/output/output.manifest.

Se você não especificar `DatasetSource`, será criado um conjunto de dados vazio. Nesse caso, a chamada para `CreateDataset` é síncrona. Posteriormente, você pode rotular imagens para o conjunto de dados UpdateDataset chamando Entries. Para ver um código demonstrativo, consulte Adicionar mais imagens (SDK).

Se você quiser substituir um conjunto de dados, primeiro exclua o conjunto de dados existente DeleteDataseto, em seguida, crie um novo conjunto de dados do mesmo tipo de conjunto de dados chamando. `CreateDataset` Para ter mais informações, consulte Excluir um conjunto de dados.

Depois de criar os conjuntos de dados, você pode criar o modelo. Para ter mais informações, consulte Treinando um modelo (SDK).

Você pode visualizar as imagens rotuladas (linhas JSON) em um conjunto de dados chamando Entries. ListDataset Você pode adicionar imagens rotuladas chamando `UpdateDatasetEntries`.

Para ver informações sobre os conjuntos de dados de teste e treinamento, consulteVisualizar seus conjuntos de dados.

Para criar um conjunto de dados (SDK)

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

   CLI

   Altere os seguintes valores:

   - `project-name` ao nome do projeto ao qual você deseja associar o conjunto de dados.
   - `dataset-type` para o tipo de conjunto de dados que você deseja criar (`trainoutest`).
   - `dataset-source` para o local do arquivo de manifesto no Amazon S3.
   - `Bucket` para o nome do bucket do Amazon S3 que contém o arquivo de manifesto.
   - `Key`ao caminho e nome do arquivo de manifesto no bucket do Amazon S3.

   ```
   aws lookoutvision create-dataset --project-name project\
     --dataset-type train or test\
     --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",
    "Key": "manifest file" } } }' \
     --profile lookoutvision-access
   ```

   Python

   Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
    @staticmethod
    def create_dataset(lookoutvision_client, project_name, manifest_file,
 dataset_type):
        """
        Creates a new Lookout for Vision dataset

        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project_name: The name of the project in which you want to
                             create a dataset.
        :param bucket: The bucket that contains the manifest file.
        :param manifest_file: The path and name of the manifest file.
        :param dataset_type: The type of the dataset (train or test).
        """
        try:
            bucket, key = manifest_file.replace("s3://", "").split("/", 1)
            logger.info("Creating %s dataset type...", dataset_type)
            dataset = {
                "GroundTruthManifest": {"S3Object": {"Bucket": bucket, "Key":
 key}}
            }
            response = lookoutvision_client.create_dataset(
                ProjectName=project_name,
                DatasetType=dataset_type,
                DatasetSource=dataset,
            )
            logger.info("Dataset Status: %s", response["DatasetMetadata"]
["Status"])
            logger.info(
                "Dataset Status Message: %s",
                response["DatasetMetadata"]["StatusMessage"],
            )
            logger.info("Dataset Type: %s", response["DatasetMetadata"]
["DatasetType"])

            # Wait until either created or failed.
            finished = False
            status = ""
            dataset_description = {}
            while finished is False:
                dataset_description = lookoutvision_client.describe_dataset(
                    ProjectName=project_name, DatasetType=dataset_type
                )
                status = dataset_description["DatasetDescription"]["Status"]
```

```
                if status == "CREATE_IN_PROGRESS":
                    logger.info("Dataset creation in progress...")
                    time.sleep(2)
                elif status == "CREATE_COMPLETE":
                    logger.info("Dataset created.")
                    finished = True
                else:
                    logger.info(
                        "Dataset creation failed: %s",
                        dataset_description["DatasetDescription"]
["StatusMessage"],
                    )
                    finished = True

            if status != "CREATE_COMPLETE":
                message = dataset_description["DatasetDescription"]
["StatusMessage"]
                logger.exception("Couldn't create dataset: %s", message)
                raise Exception(f"Couldn't create dataset: {message}")

        except ClientError:
            logger.exception("Service error: Couldn't create dataset.")
            raise
```

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
/**
 * Creates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to create a
 *                     dataset.
 * @param datasetType  The type of dataset that you want to create (train or
 *                     test).
 * @param bucket       The S3 bucket that contains the manifest file.
 * @param manifestFile The name and location of the manifest file within the S3
 *                     bucket.
```

```
     * @return DatasetDescription The description of the created dataset.
     */
    public static DatasetDescription createDataset(LookoutVisionClient lfvClient,
                    String projectName,
                    String datasetType,
                    String bucket,
                    String manifestFile)
                    throws LookoutVisionException, InterruptedException {

            logger.log(Level.INFO, "Creating {0} dataset for project {1}",
                            new Object[] { projectName, datasetType });

            // Build the request. If no bucket supplied, setup for empty dataset
     creation.
            CreateDatasetRequest createDatasetRequest = null;

            if (bucket != null && manifestFile != null) {

                    InputS3Object s3Object = InputS3Object.builder()
                                    .bucket(bucket)
                                    .key(manifestFile)
                                    .build();

                    DatasetGroundTruthManifest groundTruthManifest =
     DatasetGroundTruthManifest.builder()
                                    .s3Object(s3Object)
                                    .build();

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

                    createDatasetRequest = CreateDatasetRequest.builder()
                                    .projectName(projectName)
                                    .datasetType(datasetType)
                                    .datasetSource(datasetSource)
                                    .build();
            } else {
                    createDatasetRequest = CreateDatasetRequest.builder()
                                    .projectName(projectName)
                                    .datasetType(datasetType)
                                    .build();
            }
```

```
        lfvClient.createDataset(createDatasetRequest);

        DatasetDescription datasetDescription = null;

        boolean finished = false;

        // Wait until dataset is created, or failure occurs.
        while (!finished) {

                datasetDescription = describeDataset(lfvClient, projectName,
datasetType);

                switch (datasetDescription.status()) {
                        case CREATE_COMPLETE:
                                logger.log(Level.INFO, "{0}dataset created for
project {1}",
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;
                        case CREATE_IN_PROGRESS:
                                logger.log(Level.INFO, "{0} dataset creating for
project {1}",
                                                new Object[] { datasetType,
projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case CREATE_FAILED:
                                logger.log(Level.SEVERE,
                                                "{0} dataset creation failed for
project {1}. Error {2}",
                                                new Object[] { datasetType,
projectName,

datasetDescription.statusAsString() });
                                finished = true;
                                break;
                        default:
                                logger.log(Level.SEVERE, "{0} error when
creating {1} dataset for project {2}",
```

```
                                                    new Object[] { datasetType,
 projectName,

 datasetDescription.statusAsString() });
                                finished = true;
                                break;

                }
        }

        logger.log(Level.INFO, "Dataset info. Status: {0}\n Message: {1} }",
                        new Object[] { datasetDescription.statusAsString(),
                                        datasetDescription.statusMessage() });

        return datasetDescription;

}
```

3. Treine seu modelo seguindo as etapas emTreinando um modelo (SDK).

Rotulagem de imagens

Você pode usar o console Amazon Lookout for Vision para adicionar ou modificar os rótulos atribuídos às imagens em seu conjunto de dados. Se você estiver usando o SDK, os rótulos fazem parte do arquivo de manifesto para CreateDataseto qual você fornece. Você pode atualizar os rótulos de uma imagem chamando UpdateDatasetEntradas. Para ver um código demonstrativo, consulte Adicionar mais imagens (SDK).

Escolhendo o tipo de modelo

Os rótulos que você atribui às imagens determinam o tipo de modelo que o Lookout for Vision cria. Se seu projeto tiver um conjunto de dados de teste separado, rotule as imagens da mesma forma.

Modelo de classificação de imagens

Para criar um modelo de classificação de imagens, você classifica cada imagem como normal ou anômala. Para cada imagem, façaClassificação de imagens (console).

Modelo de segmentação de imagens

Para criar um modelo de segmentação de imagem, você classifica cada imagem como normal ou anômala. Para cada imagem anômala, você também especifica uma máscara de pixels para cada área anômala na imagem e um rótulo de anomalia para o tipo de anomalia dentro da máscara de pixels. Por exemplo, a máscara azul na imagem a seguir marca a localização de um tipo de anomalia de arranhão em um carro. É possível especificar mais de um tipo de rótulo de anomalia em uma imagem. Para cada imagem, façaSegmentação de imagens (console).

Classificação de imagens (console)

Você usa o console do Lookout for Vision para classificar imagens em um conjunto de dados como normais ou como anomalias. Imagens não classificadas não são usadas para treinar seu modelo.

Se você estiver criando um modelo de segmentação de imagens, pule esse procedimento e faça issoSegmentação de imagens (console), o que inclui etapas para classificar imagens.

> **Note**
>
> Se você acabou de concluirCriar um conjunto de dados, o console deve mostrar atualmente o painel do seu modelo e você não precisa seguir as etapas 1 a 4.

Para classificar suas imagens (console)

1. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. No painel de navegação à esquerda, escolha Projetos.
3. Na página Projetos, escolha o projeto que você deseja usar.
4. No painel de navegação à esquerda do projeto, selecione Conjunto de dados.

5. Se você tiver conjuntos de dados de treinamento e teste separados, escolha a guia do conjunto de dados que deseja usar.
6. Escolha Iniciar rotulagem.
7. Escolha Selecionar todas as imagens nesta página.
8. Se as imagens estiverem normais, escolha Classificar como normal, caso contrário, escolha Classificar como anomalia. Uma etiqueta aparece abaixo de cada imagem.
9. Se precisar alterar o rótulo de uma imagem, faça o seguinte:

    a. Escolha Anomalia ou Normal abaixo da imagem.

    b. Se você não conseguir determinar o rótulo correto para uma imagem, amplie a imagem escolhendo a imagem na galeria.

> **Note**
>
> Você pode filtrar os rótulos das imagens escolhendo o rótulo ou o estado do rótulo desejado na seção Filtros.

10. Repita as etapas 7 a 9 em cada página conforme necessário até que todas as imagens no conjunto de dados tenham sido rotuladas corretamente.
11. Escolha Salvar alterações).
12. Se você terminou de rotular suas imagens, você pode treinar seu modelo.

Segmentação de imagens (console)

Se você estiver criando um modelo de segmentação de imagens, deverá classificar as imagens como normais ou anomalias. Você também deve adicionar informações de segmentação a imagens anômalas. Para especificar as informações de segmentação, primeiro você especifica rótulos de anomalia para cada tipo de anomalia, como um dente ou arranhão, que você deseja que seu modelo encontre. Em seguida, você especifica uma máscara de anomalia e um rótulo de anomalia para cada anomalia em imagens anômalas em seu conjunto de dados.

> **Note**
>
> Se estiver criando um modelo de classificação de imagens, não será necessário segmentar imagens nem especificar rótulos de anomalias.

Tópicos

Usar a detecção de anomalias

Você define um rótulo de anomalia para cada tipo de anomalia que está nas imagens do conjunto de dados.

Usar a detecção de anomalias

1. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. No painel de navegação à esquerda, escolha Projetos.
3. Na página Projetos, escolha o projeto que você deseja usar.
4. No painel de navegação à esquerda do projeto, selecione Conjunto de dados.
5. Em Rótulos de anomalia, escolha Adicionar rótulos de anomalia. Se você já adicionou um rótulo de anomalia, escolha Gerenciar.
6. Na caixa de diálogo , , faça o seguinte:
    a. Insira o rótulo de anomalia que você deseja adicionar e escolha Adicionar rótulo de anomalia.
    b. Repita a etapa anterior até inserir todas as etiquetas de anomalias que deseja que seu modelo encontre.
    c. (Opcional) Escolha o ícone de edição para alterar o nome do rótulo.
    d. (Opcional) Escolha o ícone de exclusão para excluir um novo rótulo de anomalia. Você não pode excluir os tipos de anomalia que seu conjunto de dados está usando atualmente.
7. Escolha Confirmar para adicionar os novos rótulos de anomalia ao conjunto de dados.

Depois de especificar os rótulos de anomalia, rotule as imagens fazendo Formatar uma tabela.

Formatar uma tabela

Para rotular uma imagem para segmentação de imagem, classifique-a como normal ou como uma anomalia. Em seguida, use a ferramenta de anotação para segmentar a imagem desenhando máscaras que cubram bem as áreas de cada tipo de anomalia presente na imagem.

Para rotular uma imagem

1. Se você tiver conjuntos de dados de treinamento e teste separados, escolha a guia do conjunto de dados que deseja usar.
2. Se ainda não tiver feito isso, especifique os tipos de anomalias para o conjunto de dados fazendo. Usar a detecção de anomalias
3. Escolha Iniciar rotulagem.
4. Escolha Selecionar todas as imagens nesta página.
5. Se as imagens estiverem normais, escolha Classificar como normal, caso contrário, escolha Classificar como anomalia.
6. Para alterar o rótulo de uma única imagem, escolha Normal ou Anomalia abaixo da imagem.

> **Note**
>
> Você pode filtrar os rótulos das imagens escolhendo o rótulo ou o estado do rótulo desejado na seção Filtros. É possível classificar por pontuação de confiança na seção Opções de classificação.

7. Para cada imagem anômala, escolha a imagem para abrir a ferramenta de anotação. Adicione informações de segmentação fazendoSegmentar uma imagem com a ferramenta de anotação.
8. Escolha Salvar alterações.
9. Se você terminou de rotular suas imagens, você pode treinar seu modelo.

Segmentar uma imagem com a ferramenta de anotação

Você usa a ferramenta de anotação para segmentar uma imagem marcando áreas anômalas com uma máscara.

Para segmentar uma imagem com a ferramenta de anotação

1. Abra a ferramenta de anotação selecionando a imagem na galeria do conjunto de dados. Se necessário, escolha Iniciar rotulagem para entrar no modo de rotulagem.
2. Na seção Rótulos de anomalia, escolha o rótulo de anomalia que você deseja marcar. Se necessário, escolha Adicionar rótulos de anomalia para adicionar um novo rótulo de anomalia.
3. Escolha uma ferramenta de desenho na parte inferior da página e desenhe máscaras que cubram bem as áreas anômalas da etiqueta de anomalia. A imagem a seguir é um exemplo de uma máscara que cobre bem uma anomalia.

Veja a seguir um exemplo de uma máscara de baixa qualidade que não cobre bem uma anomalia.

4. Se você tiver mais imagens para segmentar, escolha Próximo e repita as etapas 2 e 3.
5. Escolha Enviar e fechar para finalizar a segmentação das imagens.

Treinamento de seu modelo

Depois de criar seus conjuntos de dados e rotular as imagens, você pode treinar seu modelo. Como parte do processo de treinamento, um conjunto de dados de teste é usado. Se você tiver um único projeto de conjunto de dados, as imagens no conjunto de dados serão automaticamente divididas em um conjunto de dados de teste e um conjunto de dados de treinamento como parte do processo de treinamento. Se seu projeto tiver um conjunto de dados de treinamento e um conjunto de dados de teste, eles serão usados para treinar e testar separadamente o conjunto de dados.

Depois que o treinamento for concluído, você poderá avaliar o desempenho do modelo e fazer as melhorias necessárias. Para ter mais informações, consulte Melhoria de um modelo do Amazon Lookout for Vision.

Para treinar seu modelo, o Amazon Lookout for Vision faz uma cópia das imagens originais de treinamento e teste. Por padrão, as imagens copiadas são criptografadas com uma chave que a AWS possui e gerencia. Você também pode optar por usar sua própria chave do AWS Key Management Service (KMS). Para obter mais informações, consulte Conceitos do AWS Key Management Service. Suas imagens de origem não são afetadas.

Você pode atribuir metadados ao seu modelo na forma de tags. Para ter mais informações, consulte Modelos de marcação.

Cada vez que você treina um modelo, uma nova versão do modelo é criada. Se você não precisar mais de uma versão de um modelo, poderá excluí-la. Para ter mais informações, consulte Excluir um modelo.

Você será cobrado pelo tempo necessário para treinar o modelo com sucesso. Para obter mais informações, consulte Horas de treinamento.

Para visualizar os modelos existentes em um projeto, Visualizar as modelos.

> **Note**
>
> Se você acabou de concluir Criar um conjunto de dados ouAdicionar imagens ao seu conjunto de dados. No momento, o console deve mostrar o painel do seu modelo e você não precisa seguir as etapas 1 a 4.

Tópicos

Treinando um modelo (console)

O procedimento a seguir mostra como treinar o modelo usando o console.

Para treinar seu modelo (console)

1. Abra o console do Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. No painel de navegação à esquerda, escolha Projetos.
3. Na página Projetos, escolha o projeto que contém o modelo que você deseja treinar.
4. Na página de detalhes do projeto, escolha Modelo de trem. O botão Treinar modelo está disponível se você tiver imagens rotuladas suficientes para treinar o modelo. Se o botão não estiver disponível, adicione mais imagens até que você tenha imagens rotuladas suficientes.
5. (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.

6. (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 você estiver removendo uma tag salva anteriormente, ela será removida quando você salvar suas alterações.

7. Escolha o modelo do trem.

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

9. Na visualização Modelos, você pode ver que o treinamento foi iniciado e verificar o status atual visualizando a `Status` coluna da versão do modelo. O treinamento de um modelo leva algum tempo para ser concluído.

10. Quando o treinamento terminar, você poderá avaliar seu desempenho. Para ter mais informações, consulte Melhoria de um modelo do Amazon Lookout for Vision.

Treinando um modelo (SDK)

Você usa a CreateModeloperação para iniciar o treinamento, o teste e a avaliação de um modelo. O Amazon Lookout for Vision treina o modelo usando o conjunto de dados de treinamento e teste associado ao projeto. Para ter mais informações, consulte Criar um projeto (SDK).

Toda vez que você chama `CreateModel`, uma nova versão do modelo é criada. A resposta de `CreateModel` inclui a versão do modelo.

Você é cobrado por cada modelo de treinamento bem-sucedido. Use o parâmetro `ClientToken` de entrada para ajudar a evitar cobranças devido a repetições desnecessárias ou acidentais do

treinamento de modelos por seus usuários. `ClientToken` é um parâmetro de entrada idempotente que garante que `CreateModel` seja concluído apenas uma vez para um conjunto específico de parâmetros. Uma chamada repetida de `CreateModel` com o mesmo valor `ClientToken` garante que o treinamento não seja repetido. Se você não fornecer um valor para `ClientToken`, o SDK da AWS que você está usando insere um valor para você. Isso evita que novas tentativas após um erro de rede iniciem vários trabalhos de treinamento, mas você precisará fornecer seu próprio valor para seus próprios casos de uso. Para obter mais informações, consulte CreateModel.

O treinamento demora um pouco para ser concluído. Para verificar o status atual, chame `DescribeModel` e passe o nome do projeto (especificado na chamada para`CreateProject`) e a versão do modelo. O `status` campo indica o status atual do treinamento do modelo. Para ver um código demonstrativo, consulte Visualizando seus modelos (SDK).

Se o treinamento for bem-sucedido, você poderá avaliar o modelo. Para ter mais informações, consulte Melhoria de um modelo do Amazon Lookout for Vision.

Para ver os modelos que você criou em um projeto, chame `ListModels`. Para ver um código demonstrativo, consulte Visualizar as modelos.

Para treinar um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure o AWS CLI e os AWS SDKs. Para ter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Use o código de exemplo a seguir para treinar um modelo.

   CLI

   Altere os seguintes valores:

   - `project-name` ao nome do projeto que contém o modelo que você deseja criar.
   - `output-config` para o local onde você deseja salvar os resultados do treinamento. Substitua os valores a seguir:
     - `output bucket` com o nome do bucket do Amazon S3 em que o Amazon Lookout for Vision salva os resultados do treinamento.
     - `output folder` com o nome da pasta onde deseja salvar os resultados do treinamento.
     - `Key` com o nome de uma chave de tag.
     - `Value`com um valor ao qual associar`tag_key`.

```
aws lookoutvision create-model --project-name "project name"\
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":
 "output folder" } }'\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Python

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
    @staticmethod
    def create_model(
        lookoutvision_client,
        project_name,
        training_results,
        tag_key=None,
        tag_key_value=None,
    ):
        """
        Creates a version of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project in which you want to create
a
                             model.
        :param training_results: The Amazon S3 location where training results
are stored.
        :param tag_key: The key for a tag to add to the model.
        :param tag_key_value - A value associated with the tag_key.
        return: The model status and version.
        """
        try:
            logger.info("Training model...")
            output_bucket, output_folder = training_results.replace("s3://",
"").split(
                "/", 1
            )
            output_config = {
                "S3Location": {"Bucket": output_bucket, "Prefix": output_folder}
            }
```

```
            tags = []
            if tag_key is not None:
                tags = [{"Key": tag_key, "Value": tag_key_value}]

            response = lookoutvision_client.create_model(
                ProjectName=project_name, OutputConfig=output_config, Tags=tags
            )

            logger.info("ARN: %s", response["ModelMetadata"]["ModelArn"])
            logger.info("Version: %s", response["ModelMetadata"]
["ModelVersion"])
            logger.info("Started training...")

            print("Training started. Training might take several hours to
 complete.")

            # Wait until training completes.
            finished = False
            status = "UNKNOWN"
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name,
                    ModelVersion=response["ModelMetadata"]["ModelVersion"],
                )
                status = model_description["ModelDescription"]["Status"]

                if status == "TRAINING":
                    logger.info("Model training in progress...")
                    time.sleep(600)
                    continue

                if status == "TRAINED":
                    logger.info("Model was successfully trained.")
                else:
                    logger.info(
                        "Model training failed: %s ",
                        model_description["ModelDescription"]["StatusMessage"],
                    )
                finished = True
        except ClientError:
            logger.exception("Couldn't train model.")
            raise
        else:
            return status, response["ModelMetadata"]["ModelVersion"]
```

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
/**
 * Creates an Amazon Lookout for Vision model. The function returns after model
 * training completes. Model training can take multiple hours to complete.
 * You are charged for the amount of time it takes to successfully train a
 model.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to create a
 *                    model.
 * @param description A description for the model.
 * @param bucket      The S3 bucket in which Lookout for Vision stores the
 *                    training results.
 * @param folder      The location of the training results within the S3
 *                    bucket.
 * @return ModelDescription The description of the created model.
 */
public static ModelDescription createModel(LookoutVisionClient lfvClient, String
 projectName,
                String description, String bucket, String folder)
                throws LookoutVisionException, InterruptedException {

        logger.log(Level.INFO, "Creating model for project: {0}.", new Object[]
 { projectName });

        // Setup input parameters.
        S3Location s3Location = S3Location.builder()
                        .bucket(bucket)
                        .prefix(folder)
                        .build();

        OutputConfig config = OutputConfig.builder()
                        .s3Location(s3Location)
                        .build();

        CreateModelRequest createModelRequest = CreateModelRequest.builder()
```

```
                        .projectName(projectName)
                        .description(description)
                        .outputConfig(config)
                        .build();

        // Create and train the model.
        CreateModelResponse response =
lfvClient.createModel(createModelRequest);

        String modelVersion = response.modelMetadata().modelVersion();
        boolean finished = false;
        DescribeModelResponse descriptionResponse = null;

        // Wait until training finishes or fails.

        do {
                DescribeModelRequest describeModelRequest =
DescribeModelRequest.builder()
                                .projectName(projectName)
                                .modelVersion(modelVersion)
                                .build();

                descriptionResponse =
lfvClient.describeModel(describeModelRequest);

                switch (descriptionResponse.modelDescription().status()) {
                        case TRAINED:
                                logger.log(Level.INFO, "Model training completed
for project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

                        case TRAINING:
                                logger.log(Level.INFO,
                                                "Model training in progress for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                TimeUnit.SECONDS.sleep(60);

                                break;
```

```
                        case TRAINING_FAILED:
                                logger.log(Level.SEVERE,
                                                "Model training failed for for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "Unexpected error when training
model project {0} version {1}: {2}.",
                                                new Object[] { projectName,
modelVersion,

descriptionResponse.modelDescription()

.status() });
                                finished = true;
                                break;

                }
        } while (!finished);

        return descriptionResponse.modelDescription();

}
```

3. Quando o treinamento terminar, você poderá avaliar seu desempenho. Para ter mais informações, consulte Melhoria de um modelo do Amazon Lookout for Vision.

Treinamento de modelos de solução de

Problemas com seu arquivo de manifesto ou imagens de treinamento podem fazer com que o treinamento do modelo falhe. Antes de treinar novamente seu modelo, verifique os seguintes possíveis problemas.

As cores dos rótulos de anomalias não correspondem à cor das anomalias na imagem da máscara

Se você estiver treinando um modelo de segmentação de imagem, a cor do rótulo de anomalia no arquivo de manifesto deve corresponder à cor que está na imagem da máscara. A linha JSON de uma imagem no arquivo de manifesto tem metadados (`internal-color-map`) que informam ao Amazon Lookout for Vision qual cor corresponde a um rótulo de anomalia. Por exemplo, a cor do rótulo de anomalia `scratch` na linha JSON a seguir é `#2ca02c`.


```
{
    "source-ref": "s3://path-to-image",
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "creation-date": "2021-10-12T14:16:45.668",
        "human-annotated": "yes",
        "job-name": "labeling-job/classification-job",
        "type": "groundtruth/image-classification",
        "confidence": 1
    },
    "anomaly-mask-ref": "s3://path-to-image",
    "anomaly-mask-ref-metadata": {
        "internal-color-map": {
            "0": {
                "class-name": "BACKGROUND",
                "hex-color": "#ffffff",
                "confidence": 0.0
            },
            "1": {
                "class-name": "scratch",
                "hex-color": "#2ca02c",
                "confidence": 0.0
            },
            "2": {
                "class-name": "dent",
                "hex-color": "#1f77b4",
                "confidence": 0.0
            }
        },
        "type": "groundtruth/semantic-segmentation",
        "human-annotated": "yes",
```

```
        "creation-date": "2021-11-23T20:31:57.758889",
        "job-name": "labeling-job/segmentation-job"
    }
}
```

Se as cores na imagem da máscara não corresponderem aos valores em `hex-color`, o treinamento falhará e você precisará atualizar o arquivo de manifesto.

Para atualizar os valores de cor em um arquivo de manifesto

1. Usando um editor de texto, abra o arquivo de manifesto que você usou para criar o conjunto de dados.
2. Para cada linha JSON (imagem), verifique se as cores (`hex-color`) dentro do `internal-color-map` campo correspondem às cores dos rótulos de anomalia na imagem da máscara.

   Você pode obter a localização da imagem da máscara *`anomaly-mask`*`-ref` no campo. Baixe a imagem para o seu computador e use o código a seguir para obter as cores em uma imagem.

   ```
   from PIL import Image
   img = Image.open('path to local copy of mask file')
   colors = img.convert('RGB').getcolors() #this converts the mode to RGB
   for color in colors:
       print('#%02x%02x%02x' % color[1])
   ```

3. Para cada imagem com uma atribuição de cor incorreta, atualize o `hex-color` campo na linha JSON da imagem.
4. Salve o arquivo manifesto.
5. Exclua o conjunto de dados existente do projeto.
6. Crie um novo conjunto de dados no projeto com o arquivo de manifesto atualizado.
7. Treine o modelo.

Como alternativa, nas etapas 5 e 6, você pode atualizar imagens individuais no conjunto de dados chamando a operação UpdateDatasetEntries e fornecendo linhas JSON atualizadas para as imagens que você deseja atualizar. Para ver um código demonstrativo, consulte Adicionar mais imagens (SDK).

As imagens de máscara não estão no formato PNG

Se você estiver treinando um modelo de segmentação de imagens, as imagens da máscara devem estar no formato PNG. Se você criar um conjunto de dados a partir de um arquivo de manifesto, verifique se as imagens de máscara às quais você faz referência `anomaly-mask-ref` estão no formato PNG. Se as imagens da máscara não estiverem no formato PNG, você precisará convertê-las para o formato PNG. Não é suficiente renomear a extensão de um arquivo de imagem para `.png`.

As imagens de máscara que você cria no console Amazon Lookout for Vision ou com SageMaker uma tarefa do Ground Truth são criadas no formato PNG. Você não precisa alterar o formato dessas imagens.

Para corrigir imagens de máscara em formato não PNG em um arquivo de manifesto

1. Usando um editor de texto, abra o arquivo de manifesto que você usou para criar o conjunto de dados.
2. Para cada linha JSON (imagem), certifique-se de que a imagem faça `anomaly-mask-ref` referência a uma imagem no formato PNG. Para ter mais informações, consulte Criar um arquivo de manifesto.
3. Baixe o arquivo manifesto.
4. Exclua o conjunto de dados existente do projeto.
5. Crie um novo conjunto de dados no projeto com o arquivo de manifesto atualizado.
6. Treine o modelo.

Os rótulos de segmentação ou classificação são imprecisos ou ausentes

Rótulos ausentes ou imprecisos podem fazer com que o treinamento falhe ou crie um modelo com baixo desempenho. Recomendamos que você rotule todas as imagens em seu conjunto de dados. Se você não rotular todas as imagens e o treinamento do modelo falhar ou se seu modelo tiver um desempenho ruim, adicione mais imagens.

Verifique o seguinte:

- Se você estiver criando um modelo de segmentação, as máscaras devem cobrir rigorosamente as anomalias nas imagens do seu conjunto de dados. Para verificar as máscaras em seu conjunto de dados, visualize as imagens na galeria de conjuntos de dados do projeto. Se necessário,

redesenhe as máscaras de imagem. Para ter mais informações, consulte Segmentação de imagens (console).

- Certifique-se de que as imagens anômalas nas imagens do seu conjunto de dados sejam classificadas. Se você estiver criando um modelo de segmentação de imagens, certifique-se de que imagens anômalas tenham rótulos de anomalia e máscaras de imagem.

  É importante lembrar qual tipo de modelo (segmentação ou classificação) você está criando. Um modelo de classificação não exige máscaras de imagem em imagens anômalas. Não adicione máscaras às imagens do conjunto de dados destinadas a um modelo de classificação.

  Para atualizar os rótulos ausentes

  1. Abra a galeria de conjuntos de dados do projeto.
  2. Filtre imagens sem rótulos para ver quais imagens não têm rótulos.
  3. Execute um destes procedimentos:
     - Se você estiver criando um modelo de classificação de imagens, classifique cada imagem sem rótulo.
     - Se você estiver criando um modelo de segmentação de imagem, classifique e segmente cada imagem sem rótulo.
  4. Se você estiver criando um modelo de segmentação de imagem, adicione máscaras a qualquer imagem anômala classificada que não tenha máscaras.
  5. Treine o modelo.

Se você optar por não corrigir rótulos incorretos ou ausentes, recomendamos adicionar mais imagens rotuladas ou remover as imagens afetadas do conjunto de dados. Você pode adicionar mais do console ou usando a operação UpdateDatasetEntradas. Para ter mais informações, consulte Adicionar imagens ao seu conjunto de dados.

Se você optar por remover as imagens, deverá recriar o conjunto de dados sem as imagens afetadas, pois não é possível excluir uma imagem de um conjunto de dados. Para ter mais informações, consulte Removendo imagens do seu conjunto de dados.

Melhoria de um modelo do Amazon Lookout for Vision

Durante o treinamento, o Lookout for Vision testa seu modelo com o conjunto de dados de teste e usa os resultados para criar métricas de desempenho. Você pode usar métricas de desempenho para avaliar o desempenho do seu modelo. Se necessário, você pode tomar medidas para melhorar seus conjuntos de dados e, em seguida, retreinar seu modelo.

Se estiver satisfeito com o desempenho do seu modelo, poderá começar a usá-lo. Para obter mais informações, consulte Executando seu modelo treinado do Amazon Lookout for Vision.

Tópicos

Etapa 1: Avalie o desempenho do seu modelo

Você pode acessar as métricas de desempenho no console e na operação DescribeModel. O Amazon Lookout for Vision fornece métricas resumidas de desempenho para o conjunto de dados de teste e os resultados previstos para todas as imagens individuais. Se seu modelo for um modelo de segmentação, o console também mostrará métricas resumidas para cada rótulo de anomalia.

Para ver as métricas de desempenho e as previsões de imagens de teste no console, consulte Visualização de métricas de desempenho (console). Para obter informações sobre como acessar métricas de desempenho e previsões de imagens de teste com a operação `DescribeModel`, consulte Visualização de métricas de performance.

Métricas de classificação

O Amazon Lookout for Vision fornece as seguintes métricas resumidas para as classificações que um modelo faz durante o teste:

- Precisão
- Recall
- Pontuação de F1

Métricas do modelo de segmentação de imagens

Se o modelo for um modelo de segmentação de imagens, o Amazon Lookout for Vision fornece métricas resumidas de classificação de imagens e métricas resumidas de desempenho para cada rótulo de anomalia:

- Pontuação de F1
- Interseção média sobre a União (IoU)

Precisão

A métrica de precisão responde à pergunta: quando o modelo prevê que uma imagem contém uma anomalia, com que frequência essa previsão está correta?

A precisão é uma métrica útil para situações em que o custo de um falso positivo é alto. Por exemplo, o custo de remover uma peça da máquina que não está com defeito de uma máquina montada.

O Amazon Lookout for Vision fornece um valor métrico de precisão resumida para todo o conjunto de dados de teste.

A precisão é a fração de anomalias previstas corretamente (verdadeiros positivos) sobre todas as anomalias previstas (verdadeiros e falsos positivos). A fórmula para precisão é a seguinte.

Valor de precisão = verdadeiros positivos/ (verdadeiros positivos + falsos positivos)

Os valores possíveis para precisão variam de 0 a 1. O console Amazon Lookout for Vision exibe a precisão como um valor percentual (0—100).

Um valor de precisão mais alto indica que mais anomalias previstas estão corretas. Por exemplo, suponha que seu modelo preveja que 100 imagens são anômalas. Se 85 das previsões estiverem corretas (os verdadeiros positivos) e 15 estiverem incorretas (os falsos positivos), a precisão será calculada da seguinte forma:

85 verdadeiros positivos / (85 verdadeiros positivos + 15 falsos positivos) = valor de precisão de 0,85

No entanto, se o modelo prediz apenas 40 imagens corretamente de 100 previsões de anomalias, o valor de precisão resultante será menor em 0,40 (ou seja, 40 / (40 + 60) = 0,40). Nesse caso, seu modelo está fazendo mais previsões incorretas do que previsões corretas. Para corrigir isso,

considere fazer melhorias em seu modelo. Para obter mais informações, consulte Etapa 2: Melhore seu modelo.

Para obter mais informações, consulte Precisão e recall.

Recall

A métrica de recall responde à pergunta: Do número total de imagens anômalas no conjunto de dados de teste, quantas são corretamente previstas como anômalas?

A métrica de recall é útil para situações em que o custo de um falso negativo é alto. Por exemplo, quando o custo de não remover uma peça defeituosa é alto. O Amazon Lookout for Vision fornece um valor resumido da métrica de recall para todo o conjunto de dados de teste.

O recall é a fração das imagens de teste anômalas que foram detectadas corretamente. É uma medida da frequência com que o modelo pode prever corretamente uma imagem anômala, quando ela está realmente presente nas imagens do seu conjunto de dados de teste. A fórmula para recall é calculada da seguinte forma:

Valor de recuperação = verdadeiros positivos/ (verdadeiros positivos+falsos negativos)

O intervalo para recall é de 0 a 1. O console Amazon Lookout for Vision exibe o recall como um valor percentual (0—100).

Um valor de recall mais alto indica que mais imagens anômalas foram identificadas corretamente. Por exemplo, suponha que o conjunto de dados de teste contenha 100 imagens anômalas. Se o modelo detectar corretamente 90 das 100 imagens anômalas, o recall será o seguinte:

90 verdadeiros positivos / (90 verdadeiros positivos + 10 falsos negativos) = valor de recall de 0,90

Um valor de recall de 0,90 indica que seu modelo está prevendo corretamente a maioria das imagens anômalas no conjunto de dados de teste. Se o modelo predizer corretamente apenas 20 das imagens anômalas, o recall será menor em 0,20 (ou seja, 20 / (20 + 80) = 0,20).

Nesse caso, considere a possibilidade de aprimorar seu modelo. Para obter mais informações, consulte Etapa 2: Melhore seu modelo.

Para obter mais informações, consulte Precisão e revocação.

Pontuação de F1

O Amazon Lookout for Vision fornece uma pontuação média de desempenho do modelo para o conjunto de dados de teste. Especificamente, o desempenho do modelo para classificação de anomalias é medido pela métrica de pontuação F1, que é a média harmônica das pontuações de precisão e recall.

A pontuação F1 é uma medida agregada que leva em consideração a precisão e o recall. A pontuação de performance do modelo é um valor entre 0 e 1. Quanto maior o valor, melhor o desempenho do modelo em termos de recuperação e precisão. Por exemplo, para um modelo com precisão de 0,9 e recall de 1,0, a pontuação F1 é 0,947.

Se o modelo não estiver 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 na F1 é baixa, o que indica problemas com o modelo.

Para obter mais informações, consulte pontuação F1.

Interseção média sobre a União (IoU)

A porcentagem média de sobreposição entre as máscaras de anomalia nas imagens de teste e as máscaras de anomalia que o modelo prevê para as imagens de teste. O Amazon Lookout for Vision retorna a média de IoU para cada etiqueta de anomalia e só é retornada por modelos de segmentação de imagem.

Um valor percentual baixo indica que o modelo não está combinando com precisão as máscaras previstas para uma etiqueta com as máscaras nas imagens de teste.

A imagem a seguir tem um IOU baixo. A máscara laranja é a previsão do modelo e não cobre bem a máscara azul que representa a máscara em uma imagem de teste.

A imagem a seguir tem um IOU maior. A máscara azul (imagem de teste) está bem coberta pela máscara laranja (máscara prevista).

Resultados dos testes

Durante o teste, o modelo prevê a classificação de cada imagem de teste no conjunto de dados de teste. O resultado de cada previsão é comparado ao rótulo (normal ou anomalia) da imagem de teste correspondente da seguinte forma:

- Prever corretamente que uma imagem é anômala é considerado um verdadeiro positivo.
- Prever incorretamente que uma imagem é anômala é considerado falso positivo.
- Prever corretamente que uma imagem é normal é considerado um verdadeiro negativo.
- Prever incorretamente que uma imagem é normal é considerado um falso negativo.

Se o modelo for um modelo de segmentação, ele também prevê máscaras e rótulos de anomalias para a localização das anomalias na imagem de teste.

O Amazon Lookout for Vision usa os resultados das comparações para gerar as métricas de desempenho.

Etapa 2: Melhore seu modelo

As métricas de desempenho podem mostrar que você pode melhorar seu modelo. Por exemplo, se o modelo não detectar todas as anomalias no conjunto de dados de teste, seu modelo tem baixa recuperação (ou seja, a métrica de recuperação tem um valor baixo). Se precisar melhorar seu modelo, considere o seguinte:

- Verifique se as imagens do conjunto de dados de treinamento e teste estão devidamente rotuladas.

- Reduza a variabilidade das condições de captura de imagem, como iluminação e pose do objeto, e treine seu modelo em objetos do mesmo tipo.
- Certifique-se de que suas imagens mostrem somente o conteúdo necessário. Por exemplo, se seu projeto detectar anomalias em peças de máquinas, certifique-se de que nenhum outro objeto esteja em suas imagens.
- Adicione mais imagens rotuladas aos seus conjuntos de dados de treinamento e teste. Se seu conjunto de dados de teste tem excelente recuperação e precisão, mas o modelo tem um desempenho ruim quando implantado, seu conjunto de dados de teste pode não ser representativo o suficiente e você precisa ampliá-lo.
- Se seu conjunto de dados de teste resultar em recuperação e precisão insuficientes, considere a correspondência entre as anomalias e as condições de captura de imagem nos conjuntos de dados de treinamento e teste. Se suas imagens de treinamento não representarem as anomalias e condições esperadas, mas as imagens nas imagens de teste sim, adicione imagens ao conjunto de dados de treinamento com as anomalias e condições esperadas. Se as imagens do conjunto de dados de teste não estiverem nas condições esperadas, mas as imagens de treinamento estiverem, atualize o conjunto de dados de teste.

  Para obter mais informações, consulte Adicionando mais imagens. Uma forma alternativa de adicionar imagens rotuladas ao seu conjunto de dados de treinamento é executar uma tarefa de detecção de teste e verificar os resultados. Em seguida, você pode adicionar as imagens verificadas ao conjunto de dados de treinamento. Para obter mais informações, consulte Verificando seu modelo com uma tarefa de detecção de teste.
- Certifique-se de ter imagens normais e anômalas suficientemente diversas em seu conjunto de dados de treinamento e teste. As imagens devem representar o tipo de imagem normal e anômala que seu modelo encontrará. Por exemplo, ao analisar placas de circuito, suas imagens normais devem representar as variações na posição e na soldagem dos componentes, como resistores e transistores. As imagens anômalas devem representar os diferentes tipos de anomalias que o sistema pode encontrar, como componentes perdidos ou perdidos.
- Se seu modelo tiver uma média baixa de IOU para os tipos de anomalias detectados, verifique as saídas da máscara do modelo de segmentação. Para alguns casos de uso, como arranhões, o modelo pode produzir arranhões muito próximos aos arranhões reais nas imagens de teste, mas com uma baixa sobreposição de pixels. Por exemplo, duas linhas paralelas separadas por 1 pixel de distância. Nesses casos, o IOU médio é um indicador não confiável para medir o sucesso da previsão.

- Se o tamanho da imagem for pequeno ou a resolução da imagem for baixa, considere capturar imagens em uma resolução maior. As dimensões da imagem podem variar de 64 x 64 pixels até 4096 pixels x 4096 pixels.
- Se o tamanho da anomalia for pequeno, considere dividir as imagens em blocos separados e usar as imagens lado a lado para treinamento e teste. Isso permite que o modelo veja defeitos em um tamanho maior em uma imagem.

Depois de melhorar seu conjunto de dados de treinamento e teste, treine novamente e reavalie seu modelo. Para obter mais informações, consulte Treinamento de seu modelo.

Se as métricas mostrarem que seu modelo tem desempenho aceitável, você pode verificar seu desempenho adicionando os resultados de uma tarefa de detecção de teste ao conjunto de dados de teste. Após a reciclagem, as métricas de desempenho devem confirmar as métricas de desempenho do treinamento anterior. Para obter mais informações, consulte Verificando seu modelo com uma tarefa de detecção de teste.

Visualização de métricas de performance

Você pode obter métricas de desempenho no console e chamando a operação `DescribeModel`.

Tópicos

- Visualização de métricas de desempenho (console)
- Visualização de métricas de performance

Visualização de métricas de desempenho (console)

Depois que o treinamento for concluído, o console exibirá as métricas de desempenho.

O console Amazon Lookout for Vision mostra as seguintes métricas de desempenho para as classificações feitas durante os testes:

- Precisão
- Recall
- Pontuação de F1

Se o modelo for um modelo de segmentação, o console também mostrará as seguintes métricas de desempenho para cada rótulo de anomalia:

- O número de imagens de teste em que a etiqueta de anomalia foi encontrada.
- Pontuação de F1
- Interseção média sobre a União (IoU)

A seção de visão geral dos resultados do teste mostra o total de previsões corretas e incorretas para imagens no conjunto de dados de teste. Você também pode ver as atribuições de etiquetas previstas e reais para imagens individuais no conjunto de dados de teste.

O procedimento a seguir mostra como obter as métricas de performance da visualização da lista de modelos de um projeto.

Para visualizar as métricas de performance

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos.
4. Na visualização de projetos, escolha o projeto que contém a versão do modelo que você deseja visualizar.
5. No painel de navegação, abaixo do nome do projeto, escolha Modelos.
6. Na visualização da lista de modelos, escolha as versões do modelo que você deseja visualizar.
7. Na página de detalhes do modelo, visualize as métricas de desempenho na guia Métricas de desempenho.
8. Observe o seguinte:

    a. A seção Métricas de desempenho do modelo do modelo contém métricas gerais do modelo (precisão, recall, pontuação F1) para as previsões de classificação que o modelo fez para as imagens de teste.

    b. Se o modelo for um modelo de segmentação de imagem, a seção Desempenho por etiqueta conterá o número de imagens de teste nas quais a etiqueta de anomalia foi encontrada. Você também vê métricas (pontuação F1, média de IOU) para cada rótulo de anomalia.

    c. A seção Visão geral dos resultados do teste fornece os resultados de cada imagem de teste que o Lookout for Vision usa para avaliar o modelo. A tabela inclui os seguintes campos:

- O número total de previsões de classificação corretas (verdadeiro positivo) e incorretas (falso negativo) (normal ou anomalia) para todas as imagens de teste.
- A previsão de classificação para cada imagem de teste. Se você ver Correto abaixo de uma imagem, a classificação prevista corresponde à classificação real da imagem. Caso contrário, o modelo não classificou corretamente a imagem.
- Com um modelo de segmentação de imagem, você vê rótulos de anomalia que o modelo atribuiu à imagem e máscaras na imagem que correspondem às cores dos rótulos de anomalia.

Visualização de métricas de performance

Você pode usar a operação DescribeModel para obter as métricas de desempenho resumidas (classificação) do modelo, o manifesto da avaliação e os resultados da avaliação de um modelo.

Obtendo as métricas resumidas de desempenho

As métricas resumidas de desempenho das previsões de classificação feitas pelo modelo durante o teste (Precisão,Recall e Pontuação de F1) são retornadas no `Performance` campo retornado por uma chamada para `DescribeModel`.

```
        "Performance": {
            "F1Score": 0.8,
            "Recall": 0.8,
            "Precision": 0.9
        },
```

O campo `Performance` não inclui as métricas de desempenho do rótulo de anomalia retornadas por um modelo de segmentação. Você pode obtê-los do campo `EvaluationResult`. Para obter mais informações, consulte Analisando o resultado da avaliação.

Para obter informações sobre métricas resumidas de desempenho, consulte Etapa 1: Avalie o desempenho do seu modelo. Para ver um código demonstrativo, consulte Visualizando seus modelos (SDK).

Usando o manifesto de avaliação

O manifesto de avaliação fornece métricas de previsão de teste para as imagens individuais usadas para testar um modelo. Para cada imagem no conjunto de dados de teste, uma linha JSON contém

as informações originais do teste (verdade fundamental) e a previsão do modelo para a imagem. O Amazon Lookout for Vision armazena o manifesto de avaliação em um bucket do Amazon S3. Você pode obter a localização do campo `EvaluationManifest` na resposta da operação `DescribeModel`.

```
"EvaluationManifest": {
            "Bucket": "lookoutvision-us-east-1-nnnnnnnnnn",
            "Key": "my-sdk-project-model-output/EvaluationManifest-my-sdk-
project-1.json"
        }
```

O formato final do nome do arquivo é `EvaluationManifest-`*`project name`*`.json`. Para ver um código demonstrativo, consulte Visualizar as modelos.

No exemplo de linha JSON a seguir, essa `class-name` é a verdade fundamental para o conteúdo da imagem. Neste exemplo, a imagem contém uma anomalia. O campo `confidence` mostra a confiança que o Amazon Lookout for Vision tem na previsão.

```
{
    "source-ref"*: "s3://customerbucket/path/to/image.jpg",
    "source-ref-metadata": {
        "creation-date": "2020-05-22T21:33:37.201882"
    },

    // Test dataset ground truth
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "type": "groundtruth/image-classification",
        "human-annotated": "yes",
        "creation-date": "2020-05-22T21:33:37.201882",
        "job-name": "labeling-job/anomaly-detection"
    },
    // Anomaly label detected by Lookout for Vision
    "anomaly-label-detected": 1,
    "anomaly-label-detected-metadata": {
        "class-name": "anomaly",
        "confidence": 0.9,
        "type": "groundtruth/image-classification",
        "human-annotated": "no",
        "creation-date": "2020-05-22T21:33:37.201882",
```

```
        "job-name": "training-job/anomaly-detection",
        "model-arn": "lookoutvision-some-model-arn",
        "project-name": "lookoutvision-some-project-name",
        "model-version": "lookoutvision-some-model-version"
    }
}
```

Analisando o resultado da avaliação

O resultado da avaliação tem as seguintes métricas de desempenho agregadas (classificação) para todo o conjunto de imagens de teste:

- Precisão
- Recall
- Curva ROC (não mostrada no console)
- Precisão média (não mostrada no console)
- Pontuação de F1

O resultado da avaliação também inclui o número de imagens usadas para treinar e testar o modelo.

Se o modelo for um modelo de segmentação, o resultado da avaliação também incluirá as seguintes métricas para cada rótulo de anomalia encontrado no conjunto de dados de teste:

- Precisão
- Recall
- Pontuação de F1
- Interseção média sobre a União (IoU)

O Amazon Lookout for Vision armazena o resultado da avaliação em um bucket do Amazon S3. Você pode obter a localização verificando o valor de `EvaluationResult` na resposta da operação `DescribeModel`.

```
"EvaluationResult": {
            "Bucket": "lookoutvision-us-east-1-nnnnnnnnnn",
            "Key": "my-sdk-project-model-output/EvaluationResult-my-sdk-project-1.json"
        }
```

O formato do nome do arquivo é `EvaluationResult-`*`project name`*`.json`. Para ver um exemplo, consulte Visualizar as modelos.

O esquema a seguir mostra o resultado da avaliação.

```
   {
       "Version": 1,
       "EvaluationDetails":
       {
           "ModelArn": "string", // The Amazon Resource Name (ARN) of the model
version.
           "EvaluationEndTimestamp": "string",   // The UTC date and time that
evaluation finished.
           "NumberOfTrainingImages": int,        // The number of images that were
successfully used for training.
           "NumberOfTestingImages": int          // The number of images that were
successfully used for testing.
       },
       "AggregatedEvaluationResults":
       {
           "Metrics":
           {                       //Classification metrics.
               "ROCAUC": float,            // ROC area under the curve.
               "AveragePrecision": float,  // The average precision of the model.
               "Precision": float,         // The overall precision of the model.
               "Recall": float,            // The overall recall of the model.
               "F1Score": float,           // The overal F1 score for the model.

               "PixelAnomalyClassMetrics":     //Segmentation metrics.
               [
                   {
                       "Precision": float,       // The precision for the anomaly
label.
                       "Recall": float,          // The recall for the anomaly label.
                       "F1Score": float,         // The F1 score for the anomaly
label.
                       "AIOU" : float,           // The average Intersection Over
Union for the anomaly label.
                       "ClassName": "string"   // The anomaly label.
                   }
               ]
           }
       }
```

```
    }
```

Verificando seu modelo com uma tarefa de detecção de teste

Se quiser verificar ou melhorar a qualidade do seu modelo, você pode executar uma tarefa de detecção de teste. Uma tarefa de detecção de teste detecta anomalias nas novas imagens que você fornece.

Você pode verificar os resultados da detecção e adicionar as imagens verificadas ao seu conjunto de dados. Se você tiver conjuntos de dados de treinamento e teste separados, as imagens verificadas serão adicionadas ao conjunto de dados de treinamento.

Você pode verificar imagens do seu computador local ou imagens localizadas em um bucket do Amazon S3. Se você quiser adicionar imagens verificadas ao conjunto de dados, as imagens localizadas em um bucket do S3 devem estar no mesmo bucket do S3 que as imagens do seu conjunto de dados.

> **Note**
>
> Para executar uma tarefa de detecção de teste, certifique-se de que seu bucket do S3 tenha o versionamento ativado. Para obter mais informações, consulte Usar versionamento. O bucket do console é criado com o controle de versão ativado.

Por padrão, suas imagens são criptografadas com uma chave que a AWS possui e gerencia. Você também pode escolher usar sua própria chave do AWS Key Management Service (KMS). Para obter mais informações, consulte Conceitos do AWS Key Management Service.

Tópicos

Executando uma tarefa de detecção de testes

Execute as etapas a seguir para executar uma tarefa de detecção de teste.

Para executar uma detecção de teste (console)

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos.
4. Na visualização de projetos, escolha o projeto que contém a versão do modelo que você deseja visualizar.
5. No painel de navegação, abaixo do nome do projeto, escolha Detecções de teste.
6. Na visualização de detecções de teste, escolha Executar detecção de teste.
7. Na página Executar detecção de teste, insira um nome para sua tarefa de detecção de teste em Nome da tarefa.
8. Em Escolher modelo, escolha a versão desse modelo que você deseja usar.
9. Importe as imagens de acordo com a fonte das imagens da seguinte forma:

   - Se você estiver importando suas imagens de origem de um bucket do Amazon S3, insira o URI do S3.

     > **Tip**
     >
     > Se você estiver usando as imagens de exemplo de introdução, use a pasta extra_images. O URI do Amazon S3 é. `s3://your bucket/circuitboard/extra_images`

   - Se você estiver carregando imagens do seu computador, adicione as imagens depois de escolher Detectar anomalias.
10. (Opcional) Se você quiser usar sua própria chave de criptografia do AWS KMS, faça o seguinte:

    a. Para Criptografia de dados de imagem, escolha Personalizar configurações de criptografia (avançado).

    b. Em encryption.aws_kms_key, insira o nome de recurso da Amazon (ARN) da sua chave ou escolha uma de AWS KMS existente. Para criar uma nova chave, escolha Criar uma chave AWS IMS.
11. Escolha Detectar anomalias e, em seguida, escolha Executar detecção de teste para iniciar a tarefa de detecção de teste.
12. Verifique o status atual na visualização de detecções de testes. A detecção do teste pode demorar um pouco para ser concluída.

Verificando os resultados da detecção do ensaio

Verificar os resultados de um teste de detecção pode ajudar você a melhorar seu modelo.

Se as métricas de desempenho forem ruins, melhore seu modelo executando um teste de detecção e, em seguida, adicione imagens verificadas ao conjunto de dados (conjunto de dados de treinamento, se você tiver conjuntos de dados separados).

Se as métricas de desempenho do modelo forem boas, mas os resultados de uma detecção de teste forem ruins, você poderá melhorar seu modelo adicionando imagens verificadas ao conjunto de dados (conjunto de dados de treinamento). Se você tiver um conjunto de dados de teste separado, considere adicionar mais imagens ao conjunto de dados de teste.

Depois de adicionar imagens verificadas ao seu conjunto de dados, treine novamente e reavalie seu modelo. Para obter mais informações, consulte Treinamento de seu modelo.

Para verificar os resultados de uma detecção de teste

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. No painel de navegação à esquerda, escolha Projetos.
3. Na página Projetos, escolha o projeto que deseja usar. O painel do seu projeto é exibido.
4. No painel de navegação à esquerda, selecione Detecções de teste.
5. Escolha a detecção de teste que você deseja verificar.
6. Na página de detecção de testes, escolha Verificar previsões da máquina.
7. Escolha Selecionar todas as imagens nesta página.
8. Se as previsões estiverem corretas, escolha Verificar como correta. Caso contrário, escolha Verificar como incorreto. A previsão e a pontuação de confiança da previsão são mostradas abaixo de cada imagem.
9. Se precisar alterar o rótulo de uma imagem, faça o seguinte:
    a. Escolha Correto ou Incorreto abaixo da imagem.
    b. Se você não conseguir determinar o rótulo correto para uma imagem, amplie a imagem escolhendo a imagem na galeria.

> **Note**
>
> Você pode filtrar os rótulos das imagens escolhendo o rótulo ou o estado do rótulo desejado na seção Filtros. Você pode classificar por pontuação de confiança na seção Opções de classificação.

10. Se seu modelo for um modelo de segmentação e a máscara ou etiqueta de anomalia de uma imagem estiver errada, escolha Área anômala abaixo da imagem e abra a ferramenta de anotação. Atualize as informações de segmentação fazendo Corrigindo rótulos de segmentação com a ferramenta de anotação.
11. Repita as etapas 7 a 10 em cada página conforme necessário até que todas as imagens tenham sido verificadas.
12. Escolha Adicionar imagens verificadas ao conjunto de dados. Se você tiver conjuntos de dados separados, as imagens serão adicionadas ao conjunto de dados de treinamento.
13. Treinar seu modelo do Para obter mais informações, consulte the section called “Treinamento de seu modelo”.

Corrigindo rótulos de segmentação com a ferramenta de anotação

Você usa a ferramenta de anotação para segmentar uma imagem marcando áreas anômalas com uma máscara.

Para corrigir os rótulos de segmentação de uma imagem com a ferramenta de anotação

1. Abra a ferramenta de anotação selecionando a área anômala abaixo de uma imagem na galeria do conjunto de dados.
2. Se o rótulo de anomalia de uma máscara não estiver correto, escolha a máscara e, em seguida, escolha o rótulo de anomalia correto em Rótulos de anomalia. Se necessário, escolha Adicionar rótulo de anomalia para adicionar um novo rótulo de anomalia.
3. Se a máscara não estiver correta, escolha uma ferramenta de desenho na parte inferior da página e desenhe máscaras que cubram bem as áreas anômalas da etiqueta de anomalia. A imagem a seguir é um exemplo de uma máscara que cobre bem uma anomalia.

Veja a seguir um exemplo de uma máscara de baixa qualidade que não cobre bem uma anomalia.

4. Se você tiver mais imagens para corrigir, escolha Próximo e repita as etapas 2 e 3.
5. Escolha Enviar e fechar para concluir a atualização das imagens.

Executando seu modelo treinado do Amazon Lookout for Vision

Para detectar anomalias em imagens com seu modelo, você deve primeiro iniciar seu modelo com a operação startModel. O console do Amazon Lookout for Vision fornece comandos AWS CLI que é possível utilizar para iniciar e parar o modelo. Esta seção inclui um exemplo de código que você pode usar.

Depois que seu modelo for iniciado, você poderá usar a operação `DetectAnomalies` para detectar anomalias em uma imagem. Para obter mais informações, consulte Detectar as anomalias de uma imagem.

Tópicos

Unidades de inferência

Quando você inicia seu modelo, o Amazon Lookout for Vision provisiona no mínimo um recurso computacional, conhecido como unidade de inferência. Você especifica o número de unidades de inferência a serem usadas no parâmetro `MinInferenceUnits` de entrada da `StartModel` API. A alocação padrão para um modelo é 1 unidade de inferência.

> **Important**
>
> Você é cobrado 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 você configura a execução do seu modelo. Por exemplo, se você iniciar o modelo com duas unidades de inferência e usar o modelo por 8 horas, você será cobrado por 16 horas de inferência (8 horas de tempo de execução* duas unidades de inferência). Para obter mais informações, consulte Definição de preço do Amazon Lookout for Vision. Se você não interromper explicitamente seu modelo chamando StopModel, você será cobrado mesmo que não esteja analisando ativamente as imagens com seu modelo.

As transações por segundo (TPS) que uma única unidade de inferência suporta são afetadas pelo seguinte:

- O algoritmo que o Lookout for Vision usa para treinar o modelo. Quando você treina um modelo, vários modelos são treinados. O Lookout for Vision seleciona o modelo com o melhor desempenho com base no tamanho do conjunto de dados e em sua composição de imagens normais e anômalas.
- Imagens de alta resolução exigem mais tempo para análise.
- Imagens menores (medidas em MBs) são analisadas mais rapidamente do que imagens maiores.

Gerenciando a produtividade com unidades de inferência

Você pode aumentar ou diminuir a produtividade do seu modelo, dependendo das demandas da sua aplicação. Para aumentar a produtividade, 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 a taxa de transferência suportada 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 essa abordagem se seu modelo tiver uma taxa de transferência 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 escala automática

Se seu modelo precisar acomodar picos de demanda, o Amazon Lookout for Vision pode escalar automaticamente o número de unidades de inferência que seu modelo usa. À medida que a demanda aumenta, o Amazon Lookout for Vision adiciona unidades de inferência adicionais ao modelo e as remove quando a demanda diminui.

Para permitir que o Lookout for Vision escale 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 você não especificar um número máximo de unidades, o Lookout for Vision 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**
>
> Você não pode definir o número máximo de unidades de inferência com o console Lookout for Vision. Em vez disso, especifique o parâmetro de entrada `MaxInferenceUnits` para a operação `StartModel`.

O Lookout for Vision fornece as seguintes métricas do Amazon CloudWatch Logs que você pode usar para determinar o status atual do dimensionamento automático de um modelo.

Métrica	Descrição
`DesiredInferenceUnits`	O número de unidades de inferência para as quais o Lookout for Vision está aumentando ou diminuindo.
`InServiceInferenceUnits`	O número de unidades de inferência que o modelo está usando.

Se `DesiredInferenceUnits` = `InServiceInferenceUnits`, o Lookout for Vision não está atualmente escalando o número de unidades de inferência.

Se `DesiredInferenceUnits` > `InServiceInferenceUnits`, o Lookout for Vision está aumentando para o valor de. `DesiredInferenceUnits`

Se `DesiredInferenceUnits` < `InServiceInferenceUnits`, o Lookout for Vision está diminuindo para o valor de. `DesiredInferenceUnits`

Para obter mais informações sobre as métricas retornadas pelo Lookout for Vision e as dimensões de filtragem, consulte Monitoramento do Lookout for Vision com o Amazon CloudWatch.

Para saber o número máximo de unidades de inferência que você solicitou para um modelo, chame DescribeModel e verifique o campo na resposta `MaxInferenceUnits`.

Uma VPC com zonas de disponibilidade e uma zona Wavelength.

Amazon Lookout for Vision; distribui unidades de inferência em várias zonas de disponibilidade em uma região AWS para oferecer maior disponibilidade. Para obter 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 DetectAnomalies são redistribuídas nas unidades de inferência restantes. Essas chamadas são bem-sucedidas se não excederem as transações por segundo (TPS) suportadas das unidades de inferência restantes. Depois que a AWS repara 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 Lookout for Vision 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.

Iniciar seu modelo do Amazon Lookout for Vision

Antes de usar um modelo do Amazon Lookout for Vision para detectar anomalias, você deve primeiro iniciar o modelo. Você inicia um modelo chamando a API StartModel e transmitindo o seguinte:

- ProjectName — o nome do projeto que contém o modelo que deseja iniciar.
- ModelVersion — A versão do modelo que você deseja iniciar.
- minInferenceUnits — O número mínimo de unidades de inferência. Para obter mais informações, consulte Unidades de inferência.
- (Opcional) MaxInferenceUnits — O número máximo de unidades de inferência que o Amazon Lookout for Vision pode usar para escalar automaticamente o modelo. Para obter mais informações, consulte Unidades de inferência de escala automática.

O console do Amazon Lookout for Vision fornece um código de exemplo que é possível utilizar para iniciar e parar um modelo.

> **Note**
>
> Você é cobrado pela quantidade de tempo em que seu modelo está em execução. Para interromper um modelo em execução, consulteParar o modelo do Amazon Lookout for Vision.
> Você pode usar a AWS SDK para visualizar os modelos em execução em todas as regiões da AWS nas quais o Lookout for Vision está disponível. Por exemplo, código, consulte find_running_models.py.

Tópicos

- Iniciar seu modelo (console)
- Iniciando seu modelo Amazon Lookout for Vision (SDK)

Iniciar seu modelo (console)

O console Amazon Lookout for Vision fornece AWS CLI um comando que você pode usar para iniciar um modelo. Depois que o modelo for iniciado, você poderá começar a detectar anomalias nas imagens. Para obter mais informações, consulte Detectar as anomalias de uma imagem.

Para iniciar um modelo (console)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
3. Escolha Como começar.
4. No painel de navegação à esquerda, escolha 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. Na página de detalhes do modelo, escolha Usar modelo e, em seguida, escolha Integrar API à nuvem.

> **Tip**
>
> Se você quiser implantar seu modelo em um dispositivo de borda, escolha Criar tarefa de empacotamento de modelo. Para obter mais informações, consulte Empacotar seu modelo Amazon Lookout for Vision.

8. Em comandos da AWS CLI, copie o comando da AWS CLI que chama `start-model`.
9. No prompt de comando, digite o comando `start-model` que você copiou na etapa anterior. Se você estiver usando o perfil `lookoutvision` para obter credenciais, adicione o parâmetro `--profile lookoutvision-access`.
10. No console, escolha Modelos na página de navegação à esquerda.
11. Verifique na coluna Status o status atual do modelo. Quando o status é Hospedado, você pode usar o modelo para detectar anomalias nas imagens. Para obter mais informações, consulte Detectar as anomalias de uma imagem.

Iniciando seu modelo Amazon Lookout for Vision (SDK)

Você inicia um modelo chamando a operação startModel.

Um modelo pode demorar um pouco para começar. Você pode verificar o status atual chamando DescribeModel. Para obter mais informações, consulte Visualizar as modelos.

Para iniciar seu modelo (SDK)

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

   CLI

   Altere os seguintes valores:

   - `project-name` ao nome do projeto que contém o modelo que deseja iniciar.
   - `model-version` para a versão do modelo que deseja iniciar.
   - `--min-inference-units` ao número de unidades de inferência que você deseja usar.

- (Opcional) `--max-inference-units` até o número máximo de unidades de inferência que o Amazon Lookout for Vision pode usar para escalar automaticamente o modelo.

```
aws lookoutvision start-model --project-name "project name"\
  --model-version model version\
  --min-inference-units minimum number of units\
  --max-inference-units max number of units \
  --profile lookoutvision-access
```

Python

Esse código foi retirado do repositório GitHub de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
    @staticmethod
    def start_model(
            lookoutvision_client, project_name, model_version,
min_inference_units, max_inference_units = None):
        """
        Starts the hosting of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name:  The name of the project that contains the version
of the
                              model that you want to start hosting.
        :param model_version: The version of 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: (Optional) The maximum number of inference
units that
        Lookout for Vision can use to automatically scale the model.
        """
        try:
            logger.info(
                "Starting model version %s for project %s", model_version,
project_name)

            if max_inference_units is None:
                lookoutvision_client.start_model(
                    ProjectName = project_name,
```

```
                ModelVersion = model_version,
                MinInferenceUnits = min_inference_units)

        else:
            lookoutvision_client.start_model(
                ProjectName = project_name,
                ModelVersion = model_version,
                MinInferenceUnits = min_inference_units,
                MaxInferenceUnits = max_inference_units)

        print("Starting hosting...")

        status = ""
        finished = False

        # Wait until hosted or failed.
        while finished is False:
            model_description = lookoutvision_client.describe_model(
                ProjectName=project_name, ModelVersion=model_version)
            status = model_description["ModelDescription"]["Status"]

            if status == "STARTING_HOSTING":
                logger.info("Host starting in progress...")
                time.sleep(10)
                continue

            if status == "HOSTED":
                logger.info("Model is hosted and ready for use.")
                finished = True
                continue

            logger.info("Model hosting failed and the model can't be used.")
            finished = True

        if status != "HOSTED":
            logger.error("Error hosting model: %s", status)
            raise Exception(f"Error hosting model: {status}")
    except ClientError:
        logger.exception("Couldn't host model.")
        raise
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
/**
 * Starts hosting an Amazon Lookout for Vision model. Returns when the model has
 * started or if hosting fails. You are charged for the amount of time that a
 * model is hosted. To stop hosting a model, use the StopModel operation.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
 *                    want to host.
 * @modelVersion The version of the model that you want to host.
 * @minInferenceUnits The number of inference units to use for hosting.
 * @maxInferenceUnits The maximum number of inference units that Lookout for
 *                    Vision can use for automatically scaling the model. If the
 *                    value is null, automatic scaling doesn't happen.
 * @return ModelDescription The description of the model, which includes the
 *         model hosting status.
 */
public static ModelDescription startModel(LookoutVisionClient lfvClient, String
 projectName, String modelVersion,
        Integer minInferenceUnits, Integer maxInferenceUnits) throws
 LookoutVisionException, InterruptedException {

    logger.log(Level.INFO, "Starting Model version {0} for project {1}.",
            new Object[] { modelVersion, projectName });

    StartModelRequest startModelRequest = null;

    if (maxInferenceUnits == null) {

        startModelRequest =
 StartModelRequest.builder().projectName(projectName).modelVersion(modelVersion)
                .minInferenceUnits(minInferenceUnits).build();
    } else {
        startModelRequest =
 StartModelRequest.builder().projectName(projectName).modelVersion(modelVersion)

 .minInferenceUnits(minInferenceUnits).maxInferenceUnits(maxInferenceUnits).build();
    }
```

```
   // Start hosting the model.
   lfvClient.startModel(startModelRequest);

   DescribeModelRequest describeModelRequest =
DescribeModelRequest.builder().projectName(projectName)
           .modelVersion(modelVersion).build();

   ModelDescription modelDescription = null;

   boolean finished = false;
   // Wait until model is hosted or failure occurs.
   do {

       modelDescription =
lfvClient.describeModel(describeModelRequest).modelDescription();

       switch (modelDescription.status()) {

       case HOSTED:
           logger.log(Level.INFO, "Model version {0} for project {1} is
running.",
                   new Object[] { modelVersion, projectName });
           finished = true;
           break;

       case STARTING_HOSTING:
           logger.log(Level.INFO, "Model version {0} for project {1} is
starting.",
                   new Object[] { modelVersion, projectName });

           TimeUnit.SECONDS.sleep(60);

           break;
       case HOSTING_FAILED:
           logger.log(Level.SEVERE, "Hosting failed for model version {0} for
project {1}.",
                   new Object[] { modelVersion, projectName });
           finished = true;
           break;

       default:
           logger.log(Level.SEVERE, "Unexpected error when hosting model
version {0} for project {1}: {2}.",
```

```
                    new Object[] { projectName, modelVersion,
 modelDescription.status() });
            finished = true;
            break;

        }

    } while (!finished);

    logger.log(Level.INFO, "Finished starting model version {0} for project {1}
 status: {2}",
            new Object[] { modelVersion, projectName,
 modelDescription.statusMessage() });

    return modelDescription;

}
```

3. Se a saída do código for `Model is hosted and ready for use`, você poderá usar o modelo para detectar anomalias nas imagens. Para obter mais informações, consulte Detectar as anomalias de uma imagem.

Parar o modelo do Amazon Lookout for Vision

Para interromper um modelo em execução, você chama a operação `StopModel` e passa o seguinte:

- Projeto - O nome do projeto que contém o modelo que você deseja interromper.
- ModelVersion — A versão do modelo que você deseja interromper.

O console Amazon Lookout for Vision fornece um código de exemplo que você pode usar para parar um modelo.

> **Note**
>
> Você é cobrado pela quantidade de tempo em que seu modelo está em execução.

Tópicos

- Parar o modelo (console)

- Interrompendo seu modelo Amazon Lookout for Vision (SDK)

Parar o modelo (console)

Execute as etapas do procedimento a seguir para parar o modelo usando o console.

Como interromper seu modelo (console)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
3. Escolha Como começar.
4. No painel de navegação à esquerda, escolha Projetos.
5. Na página Recursos de projetos, selecione o projeto que contém o modelo em execução que você deseja interromper.
6. Na seção Modelos, escolha o modelo que deseja parar.
7. Na página de detalhes do modelo, escolha Usar modelo e, em seguida, escolha Integrar API à nuvem.
8. Em comandos da AWS CLI, copie o comando da AWS CLI que chama `stop-model`.
9. No prompt de comando, digite o comando `stop-model` que você copiou na etapa anterior. Se você estiver usando o perfil `lookoutvision` para obter credenciais, adicione o parâmetro `--profile lookoutvision-access`.
10. No console, escolha Modelos na página de navegação à esquerda.
11. Verifique a coluna Status para ver o status atual do modelo. O modelo parou quando o valor da coluna Status é Treinamento concluído.

Interrompendo seu modelo Amazon Lookout for Vision (SDK)

Você interrompe um modelo chamando a operação StopModel.

Um modelo pode demorar um pouco para parar. Para verificar o status atual, chame `DescribeModel`.

Para parar seu modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Use o código de exemplo para interromper um modelo em execução.

   CLI

   Altere os seguintes valores:

   - `project-name` ao nome do projeto que contém o modelo que deseja interromper.
   - `model-version` para a versão do modelo que deseja interromper.

```
aws lookoutvision stop-model --project-name "project name"\
    --model-version model version \
  --profile lookoutvision-access
```

   Python

   Esse código foi retirado do repositório GitHub de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
    @staticmethod
    def stop_model(lookoutvision_client, project_name, model_version):
        """
        Stops a running Lookout for Vision Model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the version
of
                             the model that you want to stop hosting.
        :param model_version:  The version of the model that you want to stop
hosting.
        """
        try:
            logger.info("Stopping model version %s for %s", model_version,
project_name)
            response = lookoutvision_client.stop_model(
                ProjectName=project_name, ModelVersion=model_version
            )
            logger.info("Stopping hosting...")
```

```
            status = response["Status"]
            finished = False

            # Wait until stopped or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version
                )
                status = model_description["ModelDescription"]["Status"]

                if status == "STOPPING_HOSTING":
                    logger.info("Host stopping in progress...")
                    time.sleep(10)
                    continue

                if status == "TRAINED":
                    logger.info("Model is no longer hosted.")
                    finished = True
                    continue

                logger.info("Failed to stop model: %s ", status)
                finished = True

            if status != "TRAINED":
                logger.error("Error stopping model: %s", status)
                raise Exception(f"Error stopping model: {status}")
        except ClientError:
            logger.exception("Couldn't stop hosting model.")
            raise
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
/**
 * Stops the hosting an Amazon Lookout for Vision model. Returns when model has
 * stopped or if hosting fails.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
```

```
 *                    want to stop hosting.
 * @modelVersion The version of the model that you want to stop hosting.
 * @return ModelDescription The description of the model, which includes the
 *         model hosting status.
 */

public static ModelDescription stopModel(LookoutVisionClient lfvClient, String
 projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        logger.log(Level.INFO, "Stopping Model version {0} for project {1}.",
                        new Object[] { modelVersion, projectName });

        StopModelRequest stopModelRequest = StopModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        // Stop hosting the model.

        lfvClient.stopModel(stopModelRequest);

        DescribeModelRequest describeModelRequest =
 DescribeModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        ModelDescription modelDescription = null;

        boolean finished = false;
        // Wait until model is stopped or failure occurs.
        do {

                modelDescription =
 lfvClient.describeModel(describeModelRequest).modelDescription();

                switch (modelDescription.status()) {

                        case TRAINED:
                                logger.log(Level.INFO, "Model version {0} for
 project {1} has stopped.",
```

```
                                                    new Object[] { modelVersion,
 projectName });
                                finished = true;
                                break;

                        case STOPPING_HOSTING:
                                logger.log(Level.INFO, "Model version {0} for
 project {1} is stopping.",
                                                    new Object[] { modelVersion,
 projectName });

                                TimeUnit.SECONDS.sleep(60);

                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                    "Unexpected error when stopping
 model version {0} for project {1}: {2}.",
                                                    new Object[] { projectName,
 modelVersion,

 modelDescription.status() });
                                finished = true;
                                break;

                }

        } while (!finished);

        logger.log(Level.INFO, "Finished stopping model version {0} for project
 {1} status: {2}",
                        new Object[] { modelVersion, projectName,
 modelDescription.statusMessage() });

        return modelDescription;

}
```

Detectar as anomalias de uma imagem

Para detectar anomalias em uma imagem com um modelo treinado do Amazon Lookout for Vision, você chama a operação DetectAnomalies. O resultado de `DetectAnomalies` inclui uma previsão booleana que classifica a imagem como contendo uma ou mais anomalias e um valor de confiança para a previsão. Se o modelo for um modelo de segmentação de imagem, o resultado também incluirá uma máscara colorida mostrando as posições dos diferentes tipos de anomalias.

As imagens que você fornece `DetectAnomalies` devem ter as mesmas dimensões de largura e altura das imagens que você usou para treinar o modelo.

A `DetectAnomalies` aceita imagens no formato PNG ou JPG. Recomendamos que as imagens estejam no mesmo formato de codificação e compressão que as usadas para treinar o modelo. Por exemplo, se você treinar o modelo com imagens no formato PNG, chame `DetectAnomalies` com imagens no formato PNG.

Antes de ligar `DetectAnomalies`, você deve primeiro iniciar seu modelo com a `StartModel` operação. Para obter mais informações, consulte Iniciar seu modelo do Amazon Lookout for Vision. Você é cobrado pela quantidade de tempo, em minutos, que um modelo é executado e pelo número de unidades de detecção de anomalias que seu modelo usa. Se você não estiver usando um modelo, use a `StopModel` operação para parar seu modelo. Para obter mais informações, consulte Parar o modelo do Amazon Lookout for Vision.

Tópicos

Como chamar o DetectAnomalias

Para chamar `DetectAnomalies`, especifique o seguinte:

- Projeto — O nome do projeto que contém o modelo que deseja usar.
- ModelVersion — A versão do modelo que você deseja usar.

- ContentType — O tipo de imagem que você deseja analisar. Os valores válidos são `image/png` (imagens no formato PNG) e `image/jpeg` (imagens no formato JPG).
- Corpo — Os bytes binários não codificados que representam a imagem.

  A imagem deve ter as mesmas dimensões das imagens usadas para treinar o modelo.

O exemplo a seguir mostra como chamar `DetectAnomalies`. Você pode usar a resposta da função dos exemplos em Python e Java para chamar funções em Determinar se uma imagem é anômala

AWS CLI

Esse comando da AWS CLI exibe a saída JSON da operação da CLI `DetectAnomalies`. Alterar os valores dos seguintes parâmetros de entrada:

- `project name` com o nome do projeto que você deseja usar.
- `model version` com a versão do modelo que você deseja usar.
- `content type` com o tipo de imagem que você deseja usar. Os valores válidos são `image/png` (imagens no formato PNG) e `image/jpeg` (imagens no formato JPG).
- `file name` com o caminho e o nome do arquivo da imagem que você deseja usar. Certifique-se de que o tipo de arquivo corresponda ao valor de `content-type`.

```
aws lookoutvision detect-anomalies  --project-name project name\
    --model-version model version\
    --content-type content type\
    --body file name \
    --profile lookoutvision-access
```

Python

Para obter o exemplo de código completo, consulte GitHub.

```
    def detect_anomalies(lookoutvision_client, project_name, model_version, photo):
        """
        Calls DetectAnomalies using the supplied project, model version, and image.
        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project: The project that contains the model that you want to use.
        :param model_version: The version of the model that you want to use.
        :param photo: The photo that you want to analyze.
        :return: The DetectAnomalyResult object that contains the analysis results.
```

```
        """

        image_type = imghdr.what(photo)
        if image_type == "jpeg":
            content_type = "image/jpeg"
        elif image_type == "png":
            content_type = "image/png"
        else:
            logger.info("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
        with open(photo, "rb") as image:
            response = lookoutvision_client.detect_anomalies(
                ProjectName=project_name,
                ContentType=content_type,
                Body=image.read(),
                ModelVersion=model_version)

        return response['DetectAnomalyResult']
```

Java V2

```
  public static DetectAnomalyResult detectAnomalies(LookoutVisionClient lfvClient,
String projectName,
            String modelVersion,
            String photo) throws IOException, LookoutVisionException {
        /**
         * Creates an Amazon Lookout for Vision dataset from a manifest file.
         * Returns after Lookout for Vision creates the dataset.
         *
         * @param lfvClient    An Amazon Lookout for Vision client.
         * @param projectName  The name of the project in which you want to create a
         *                     dataset.
         * @param modelVersion The version of the model that you want to use.
         *
         * @param photo        The photo that you want to analyze.
         *
         * @return DetectAnomalyResult The analysis result from DetectAnomalies.
         */
```

```
        logger.log(Level.INFO, "Processing local file: {0}", photo);

        // Get image bytes.

        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
        byte[] imageBytes = imageSDKBytes.asByteArray();

        // Get the image type. Can be image/jpeg or image/png.
        String contentType = getImageType(imageBytes);

        // Detect anomalies in the supplied image.
        DetectAnomaliesRequest request =
DetectAnomaliesRequest.builder().projectName(projectName)
                .modelVersion(modelVersion).contentType(contentType).build();

        DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
                RequestBody.fromBytes(imageBytes));

        /*
         * Tip: You can also use the following to analyze a local file.
         * Path path = Paths.get(photo);
         * DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
path);
         */
        DetectAnomalyResult result = response.detectAnomalyResult();

        String prediction = "Prediction: Normal";

        if (Boolean.TRUE.equals(result.isAnomalous())) {
            prediction = "Prediction: Anomalous";
        }

        // Convert confidence to percentage.
        NumberFormat defaultFormat = NumberFormat.getPercentInstance();
        defaultFormat.setMinimumFractionDigits(1);
        String confidence = String.format("Confidence: %s",
defaultFormat.format(result.confidence()));

        // Log classification result.
        String photoPath = "File: " + photo;
        String[] imageLines = { photoPath, prediction, confidence };
        logger.log(Level.INFO, "Image: {0}\nAnomalous: {1}\nConfidence {2}",
imageLines);
```

```
        return result;

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.
    private static String getImageType(byte[] image) throws IOException {

        InputStream is = new BufferedInputStream(new ByteArrayInputStream(image));
        String mimeType = URLConnection.guessContentTypeFromStream(is);

        logger.log(Level.INFO, "Image type: {0}", mimeType);

        if (mimeType.equals("image/jpeg") || mimeType.equals("image/png")) {
            return mimeType;
        }
        // Not a supported file type.
        logger.log(Level.SEVERE, "Unsupported image type: {0}", mimeType);
        throw new IOException(String.format("Wrong image type. %s format isn't
supported.", mimeType));
    }
```

Entender a resposta do DetectAnomalias

A resposta `DetectAnomalies` varia de acordo com o tipo do modelo que você treina (modelo de classificação ou modelo de segmentação). Em ambos os casos, a resposta é um objeto DetectAanomalyResult.

Modelo de classificação

Se seu modelo for um Modelo de classificação de imagens, a resposta de `DetectAnomalies` conterá o seguinte:

- isAnomalous — Um indicador booleano de que a imagem contém uma ou mais anomalias.
- Confiança — A confiança que o Amazon Lookout for Vision tem na precisão da previsão da anomalia (`IsAnomalous`). `Confidence`é um valor de ponto flutuante entre 0 e 1. Um valor mais alto indica uma maior confiança.
- Fonte — Informações sobre a imagem passada para `DetectAnomalies`.

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996867775917053
    }
}
```

Você determina se uma imagem é anômala verificando o `IsAnomalous` campo e confirmando que o `Confidence` valor é alto o suficiente para suas necessidades.

Se você achar que os valores de confiança retornados por `DetectAnomalies` são muito baixos, considere retreinar o modelo. Para ver um código demonstrativo, consulte Classificação.

Modelo de segmentação

Se seu modelo for um Modelo de segmentação de imagens, a resposta inclui informações de classificação e informações de segmentação, como uma máscara de imagem e tipos de anomalia. As informações de classificação são calculadas separadamente das informações de segmentação e você não deve presumir uma relação entre elas. Se você não receber informações de segmentação na resposta, verifique se você tem a versão mais recente do AWS SDK instalada (AWS Command Line Interface, se estiver usando a AWS CLI). Para ver um código demonstrativo, consulte Segmentação e Exibindo informações de classificação e segmentação.

- isAnomalous (classificação) — Um indicador booleano que classifica a imagem como normal ou anômala.
- Confiança (classificação) — A confiança que o Amazon Lookout for Vision tem na precisão da classificação da imagem (`IsAnomalous`). `Confidence` é um valor de ponto flutuante entre 0 e 1. Um valor mais alto indica uma maior confiança.
- Fonte — Informações sobre a imagem passada para `DetectAnomalies`.
- AnomalyMask (segmentação) — Uma máscara de pixels que cobre anomalias encontradas na imagem analisada. Pode haver várias anomalias na imagem. A cor dos mapas de uma máscara indica o tipo de anomalia. As cores da máscara são mapeadas para as cores atribuídas aos tipos de anomalias no conjunto de dados de treinamento. Para encontrar o tipo de anomalia a partir da cor da máscara, verifique `Color` o campo `PixelAnomaly` de cada anomalia retornada na lista

`Anomalies`. Para ver um código demonstrativo, consulte Exibindo informações de classificação e segmentação.

- Anomalias (segmentação) — Uma lista de anomalias encontradas na imagem. Cada anomalia inclui o tipo de anomalia (`Name`) e as informações de pixel (`PixelAnomaly`). `TotalPercentageArea` é a área percentual da imagem que a anomalia cobre. `Color` é a cor da máscara para a anomalia.

  O primeiro elemento na lista é sempre um tipo de anomalia que representa o fundo da imagem (`BACKGROUND`) e não deve ser considerado uma anomalia. O Amazon Lookout for Vision adiciona automaticamente o tipo de anomalia em segundo plano à resposta. Você não precisa declarar um tipo de anomalias de fundo do seu conjunto de dados.


```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996814727783203,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.998999834060669,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "scratch",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0004034999874420464,
                    "Color": "#7ED321"
                }
            },
            {
                "Name": "dent",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0005966666503809392,
                    "Color": "#4DD8FF"
                }
            }
```

```
        ],
        "AnomalyMask": "iVBORw0....."
    }
}
```

Determinar se uma imagem é anômala

Você pode determinar se uma imagem é anômala de várias maneiras. O método que você escolhe depende do seu caso de uso. A seguir estão as possíveis soluções.

Tópicos

- Classificação
- Segmentação

Classificação

`IsAnomalous` classifica uma imagem como anômala, use o campo `Confidence` para ajudar a decidir se a imagem é realmente anômala. Um valor mais alto indica maior confiança. Por exemplo, você pode decidir que um produto está com defeito somente se a confiança for superior a 80%. Você pode classificar imagens analisadas por modelos de classificação ou por modelos de segmentação de imagens.

Python

Para obter o exemplo de código completo, consulte GitHub.

```
   def reject_on_classification(image, prediction, confidence_limit):
       """
       Returns True if the anomaly confidence is greater than or equal to
       the supplied confidence limit.
       :param image: The name of the image file that was analyzed.
       :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
       :param confidence_limit: The minimum acceptable confidence. Float value
between 0 and 1.
       :return: True if the error condition indicates an anomaly, otherwise False.
       """

       reject = False
```

```
       logger.info("Checking classification for %s", image)

       if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:
           reject = True
           reject_info=(f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) is greater"
                   f" than limit ({confidence_limit:.2%})")
           logger.info("%s", reject_info)

       if not reject:
           logger.info("No anomalies found.")
       return reject
```

Java V2

```
   public static boolean rejectOnClassification(String image, DetectAnomalyResult
prediction, float minConfidence) {
       /**
        * Rejects an image based on its anomaly classification and prediction
        * confidence
        *
        * @param image         The file name of the analyzed image.
        * @param prediction    The prediction for an image analyzed with
        *                      DetectAnomalies.
        * @param minConfidence The minimum acceptable confidence for the prediction
        *                      (0-1).
        *
        * @return boolean True if the image is anomalous, otherwise False.
        */

       Boolean reject = false;

       logger.log(Level.INFO, "Checking classification for {0}", image);

       String[] logParameters = { prediction.confidence().toString(),
String.valueOf(minConfidence) };

       if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
           logger.log(Level.INFO, "Rejected: Anomaly confidence {0} is greater than
confidence limit {1}",
```

```
                    logParameters);
            reject = true;
        }
        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;

    }
```

Segmentação

Se seu modelo for um modelo de segmentação de imagem, você poderá usar as informações de segmentação para determinar se uma imagem contém anomalias. Você também pode usar um modelo de segmentação de imagem para classificar as imagens. Por exemplo, código que obtém e exibe máscaras de imagem, consulte Exibindo informações de classificação e segmentação

Área de anomalias

Use a porcentagem de cobertura (`TotalPercentageArea`) de uma anomalia na imagem. Por exemplo, você pode decidir que um produto está com defeito se a área de uma anomalia for maior que 1% da imagem.

Python

Para obter o exemplo de código completo, consulte GitHub.

```
   def reject_on_coverage(image, prediction, confidence_limit, anomaly_label,
coverage_limit):
        """
        Checks if the coverage area of an anomaly is greater than the coverage limit
and if
        the prediction confidence is greater than the confidence limit.
        :param image: The name of the image file that was analyzed.
        :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
        :param confidence_limit: The minimum acceptable confidence (float 0-1).
        :anomaly_label: The anomaly label for the type of anomaly that you want to
check.
        :coverage_limit: The maximum acceptable percentage coverage of an anomaly
(float 0-1).
```

```
        :return: True if the error condition indicates an anomaly, otherwise False.
        """

        reject = False

        logger.info("Checking coverage for %s", image)

        if prediction['IsAnomalous'] and prediction['Confidence'] >=
 confidence_limit:
            for anomaly in prediction['Anomalies']:
                if (anomaly['Name'] == anomaly_label and
                        anomaly['PixelAnomaly']['TotalPercentageArea'] >
 (coverage_limit)):
                    reject = True
                    reject_info=(f"Rejected: Anomaly confidence
 ({prediction['Confidence']:.2%}) "
                        f"is greater than limit ({confidence_limit:.2%}) and
 {anomaly['Name']} "
                        f"coverage ({anomaly['PixelAnomaly']
['TotalPercentageArea']:.2%}) "
                        f"is greater than limit ({coverage_limit:.2%})")

                    logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")

        return reject
```

Java V2

```
   public static Boolean rejectOnCoverage(String image, DetectAnomalyResult
 prediction, float minConfidence,
            String anomalyType, float maxCoverage) {
        /**
         * Rejects an image based on a maximum allowable coverage area for an
 anomaly
         * type.
         *
         * @param image         The file name of the analyzed image.
         * @param prediction    The prediction for an image analyzed with
         *                      DetectAnomalies.
```

```
        * @param minConfidence The minimum acceptable confidence for the prediction
        *                      (0-1).
        * @param anomalyTypes  The anomaly type to check.
        * @param maxCoverage   The maximum allowable coverage area of the anomaly
type.
        *                      (0-1).
        *
        * @return boolean True if the coverage area of the anomaly type exceeds the
        *         maximum allowed, otherwise False.
        */

       Boolean reject = false;

       logger.log(Level.INFO, "Checking coverage for {0}", image);

       if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
           for (Anomaly anomaly : prediction.anomalies()) {

               if (Objects.equals(anomaly.name(), anomalyType)
                       && anomaly.pixelAnomaly().totalPercentageArea() >=
maxCoverage) {

                   String[] logParameters = { prediction.confidence().toString(),
                           String.valueOf(minConfidence),

String.valueOf(anomaly.pixelAnomaly().totalPercentageArea()),
                           String.valueOf(maxCoverage) };
                   logger.log(Level.INFO,
                           "Rejected: Anomaly confidence {0} is greater than
confidence limit {1} and " +
                                   "{2} anomaly type coverage is higher than
coverage limit {3}\n",
                           logParameters);
                   reject = true;

               }
           }
       }

       if (Boolean.FALSE.equals(reject))
           logger.log(Level.INFO, ": No anomalies found.");

       return reject;
```

```
    }
```

Número de tipos de anomalias

Use uma contagem dos diferentes tipos de anomalias (`Name`) encontrados na imagem. Por exemplo, você pode decidir que um produto está com defeito se houver mais de dois tipos de anomalia presentes.

Python

Para obter o exemplo de código completo, consulte GitHub.

```
   def reject_on_anomaly_types(image, prediction, confidence_limit,
anomaly_types_limit):
       """
       Checks if the number of anomaly types is greater than than the anomaly types
       limit and if the prediction confidence is greater than the confidence limit.
       :param image: The name of the image file that was analyzed.
       :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
       :param confidence: The minimum acceptable confidence. Float value between 0
and 1.
       :param anomaly_types_limit: The maximum number of allowable anomaly types
(Integer).
       :return: True if the error condition indicates an anomaly, otherwise False.
       """

       logger.info("Checking number of anomaly types for %s",image)

       reject = False

       if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:

           anomaly_types = {anomaly['Name'] for anomaly in prediction['Anomalies']\
               if anomaly['Name'] != 'background'}

           if len (anomaly_types) > anomaly_types_limit:
                   reject = True
                   reject_info = (f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) "
                   f"is greater than limit ({confidence_limit:.2%}) and "
```

```
                    f"the number of anomaly types ({len(anomaly_types)-1}) is "
                    f"greater than the limit ({anomaly_types_limit})")

                    logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")
        return reject
```

Java V2

```
   public static Boolean rejectOnAnomalyTypeCount(String image, DetectAnomalyResult
prediction,
            float minConfidence, Integer maxAnomalyTypes) {

        /**
         * Rejects an image based on a maximum allowable number of anomaly types.
         *
         * @param image           The file name of the analyzed image.
         * @param prediction      The prediction for an image analyzed with
         *                        DetectAnomalies.
         * @param minConfidence   The minimum acceptable confidence for the
predictio
         *                        (0-1).
         * @param maxAnomalyTypes The maximum allowable number of anomaly types.
         *
         * @return boolean True if the image contains more than the maximum allowed
         *         anomaly types, otherwise False.
         */

        Boolean reject = false;

        logger.log(Level.INFO, "Checking coverage for {0}", image);

        Set<String> defectTypes = new HashSet<>();

        if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
            for (Anomaly anomaly : prediction.anomalies()) {
                defectTypes.add(anomaly.name());
            }
            // Reduce defect types by one to account for 'background' anomaly type.
            if ((defectTypes.size() - 1) > maxAnomalyTypes) {
```

```
                String[] logParameters = { prediction.confidence().toString(),
                        String.valueOf(minConfidence),
                        String.valueOf(defectTypes.size()),
                        String.valueOf(maxAnomalyTypes) };
                logger.log(Level.INFO, "Rejected: Anomaly confidence {0} is >=
minimum confidence {1} and " +
                        "the number of anomaly types {2} > the allowable number of
anomaly types {3}\n", logParameters);
                reject = true;
            }

        }

        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;
    }
```

Exibindo informações de classificação e segmentação

Este exemplo mostra a imagem analisada e sobrepõe os resultados da análise. Se a resposta incluir uma máscara de anomalia, a máscara será mostrada nas cores dos tipos de anomalia associados.

Para mostrar informações de classificação e segmentação de imagens

1. Se você ainda não tiver feito isso, instale o .
   a. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
   b. Treinar seu modelo do
   c. Comece seu modelo.
2. Certifique-se de que o usuário `DetectAnomalies` que está ligando tenha acesso à versão do modelo que você deseja usar. Para obter mais informações, consulte Configurar permissões do SDK.
3. Use o seguinte código:

Python

O código de exemplo a seguir detecta anomalias em uma imagem que você fornece. A linha de comando da aceita as seguintes opções:

- Determine o nome do perfil da que você deseja usar.
- `version`— a versão do modelo, dentro do projeto, que você deseja usar.
- `image`— o caminho e o arquivo de um arquivo de imagem local (formato JPEG ou PNG).

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Shows how to detect and show anomalies in an image using a trained Amazon
 Lookout
for Vision model. The script displays the analysed image and overlays mask and
 analysis
output.
"""

import argparse
import logging
import io
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class ShowAnomalies:
    """
    Class to detect and show anomalies in an image analyzed by detect_anomalies.
    """



    @staticmethod
    def draw_line(draw, text, fnt,  y_coordinate, color):
        """
        Draws a line of text on the supplied drawing surface.
        :param draw: The surface on which to draw the text.
```

```
        :param text: The text to draw in the drawing surface.
        :param fnt: The font for the text.
        :param y_coordinate: The y position for the text.
        :param color: The color for the text.
        :returns The y coordinate for the next line of text.
        """
        text_width, text_height = draw.textsize(text, fnt)
        draw.rectangle([(10, y_coordinate), (text_width + 10,
                                             y_coordinate + text_height)],
 fill="black")
        draw.text((10, y_coordinate), text, fill=color, font=fnt)

        y_coordinate += text_height

        return y_coordinate

    @staticmethod
    def draw_analysis_text(image, analysis):
        """
        Draws classification and segmentation info onto supplied image
        overlay analysis results on an image analyzed by detect_anomalies.
        :param analysis: The response from a call to detect_anomalies.
        :returns Nothing
        """

        ## Calculate a reasonable font size based on image width.
        font_size = int(image.size[0]/32)

        fnt = ImageFont.truetype('/Library/Fonts/Tahoma.ttf', font_size)

        draw = ImageDraw.Draw(image)

        y_coordinate = 0

        # Draw classification information.
        prediction = "Anomalous" if analysis["DetectAnomalyResult"]
["IsAnomalous"] \
            else "Normal"

        confidence = analysis["DetectAnomalyResult"]["Confidence"]
        found_anomalies = analysis["DetectAnomalyResult"]['Anomalies']
        segmentation_info = False

        logger.info("Prediction: %s", format(prediction))
```

```
        logger.info("Confidence: %s", format(confidence))

        y_coordinate = 0
        y_coordinate = ShowAnomalies.draw_line(
            draw, "Classification", fnt, y_coordinate, "white")
        y_coordinate = ShowAnomalies.draw_line(
            draw, f"  Prediction: {prediction}", fnt, y_coordinate, "white")
        y_coordinate = ShowAnomalies.draw_line(
            draw, f"  Confidence: {confidence:.2%}", fnt, y_coordinate, "white")

        # Draw segmentation information, if present.
        if (len(found_anomalies)) > 1:
            logger.info("Anomalies:")

            y_coordinate = ShowAnomalies.draw_line(
                draw, "Segmentation:", fnt, y_coordinate, "white")
            for i in range(1, len(found_anomalies)):

                # Only display info if more than 0% coverage found.
                percent_coverage = found_anomalies[i]['PixelAnomaly']
['TotalPercentageArea']
                if percent_coverage > 0:
                    segmentation_info = True
                    logger.info("  %s", found_anomalies[i]['Name'])
                    logger.info("    Color: %s",
                                found_anomalies[i]['PixelAnomaly']['Color'])
                    logger.info("    Area: %s", percent_coverage)
                    y_coordinate = ShowAnomalies.draw_line(
                        draw,
                        f"  Anomaly: {found_anomalies[i]['Name']}. Area:
 {percent_coverage:.2%}",
                        fnt,
                        y_coordinate,
                        found_anomalies[i]['PixelAnomaly']['Color'])

            if not segmentation_info:
                y_coordinate = ShowAnomalies.draw_line(
                    draw, "No segmentation information found.", fnt,
 y_coordinate, "white")




    @staticmethod
```

```
   def show_anomaly_prediction(lookoutvision_client, project_name,
model_version, photo):
       """
       Detects anomalies in an image (jpg/png) by using your Amazon Lookout for
Vision
       model. Displays the image and overlays prediction information text.
       :param lookoutvision_client: An Amazon Lookout for Vision Boto3 client.
       :param project_name: The name of the project that contains the model
that
       you want to use.
       :param model_version: The version of the model that you want to use.
       :param photo: The path and name of the image in which you want to detect
       anomalies.
       """
       try:

           logger.info("Detecting anomalies in %s", photo)

           image = Image.open(photo)
           image_type = Image.MIME[image.format]

           # Check that image type is valid.
           if image_type not in ("image/jpeg", "image/png"):
               logger.info("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()

           # Analyze the image.
           response = lookoutvision_client.detect_anomalies(
               ProjectName=project_name,
               ContentType=image_type,
               Body=image_bytes,
               ModelVersion=model_version
           )

           # Overlay mask onto analyzed image.
           image_mask_bytes = response["DetectAnomalyResult"]["AnomalyMask"]
```

```
            image_mask = Image.open(io.BytesIO(image_mask_bytes))

            final_img = Image.blend(image, image_mask, 0.5) \
                if response["DetectAnomalyResult"]["IsAnomalous"] else image

            # Overlay analysis output on image.
            ShowAnomalies.draw_analysis_text(final_img, response)

            final_img.show()

        except ClientError as err:
            logger.info(format(err))
            raise


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

    parser.add_argument(
        "project", help="The project containing the model that you want to use."
    )
    parser.add_argument(
        "version", help="The version of the model that you want to use."
    )
    parser.add_argument(
        "image",
        help="The file that you want to analyze. "
        "Supply a local file path.",
    )


def main():
    """
    Entrypoint for anomaly detection example.
    """

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

        session = boto3.Session(
```

```
            profile_name='lookoutvision-access')

        lookoutvision_client = session.client("lookoutvision")

        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

        add_arguments(parser)

        args = parser.parse_args()

        # Analyze the image and show results.
        ShowAnomalies.show_anomaly_prediction(
            lookoutvision_client, args.project, args.version, args.image
        )

    except ClientError as err:
        print("A service error occured: " +
              format(err.response["Error"]["Message"]))
    except FileNotFoundError as err:
        print("The supplied file couldn't be found: " + err.filename)
    except ValueError as err:
        print("A value error occured. " + format(err))
    else:
        print("Successfully completed analysis.")


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

Java 2

O código de exemplo a seguir detecta anomalias em uma imagem que você fornece. A linha de comando da aceita as seguintes opções:

- Determine o nome do perfil da que você deseja usar.
- `version`— a versão do modelo, dentro do projeto, que você deseja usar.
- `image`— o caminho e o arquivo de um arquivo de imagem local (formato JPEG ou PNG).

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

```
*/


package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.Anomaly;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesResponse;
import software.amazon.awssdk.services.lookoutvision.model.DetectAnomalyResult;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;

import java.io.BufferedInputStream;
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.net.URLConnection;

import java.text.NumberFormat;
import java.awt.*;
import java.awt.font.LineMetrics;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

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

// Finds anomalies on a supplied image.
public class ShowAnomalies extends JPanel {
/**
 * Finds and displays anomalies on a supplied image.
 */

    private static final long serialVersionUID = 1L;
```

```
   private transient BufferedImage image;
   private transient BufferedImage maskImage;
   private transient Dimension dimension;
   public static final Logger logger =
Logger.getLogger(ShowAnomalies.class.getName());

   // Constructor. Finds anomalies in a local image file.
   public ShowAnomalies(LookoutVisionClient lfvClient, String projectName,
String modelVersion,
           String photo) throws IOException, LookoutVisionException {

       logger.log(Level.INFO, "Processing local file: {0}", photo);

       maskImage = null;

       // Get image bytes and buffered image.
       InputStream sourceStream = new FileInputStream(new File(photo));
       SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
       byte[] imageBytes = imageSDKBytes.asByteArray();
       ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageSDKBytes.asByteArray());
       image = ImageIO.read(inputStream);

       // Get the image type. Can be image/jpeg or image/png.
       String contentType = getImageType(imageBytes);

       // Set the size of the window that shows the image.
       setWindowDimensions();

       // Detect anomalies in the supplied image.
       DetectAnomaliesRequest request =
DetectAnomaliesRequest.builder().projectName(projectName)
               .modelVersion(modelVersion).contentType(contentType).build();

       DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
               RequestBody.fromBytes(imageBytes));

       /*
        * Tip: You can also use the following to analyze a local file.
        * Path path = Paths.get(photo);
        * DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
path);
        */
       DetectAnomalyResult result = response.detectAnomalyResult();
```

```
        if (result.anomalyMask() != null){
            SdkBytes maskSDKBytes = result.anomalyMask();

            ByteArrayInputStream maskInputStream = new
ByteArrayInputStream(maskSDKBytes.asByteArray());
            maskImage = ImageIO.read(maskInputStream);
        }

        drawImageInfo(result);

    }

    // 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;
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getWidth() < dimension.width || image.getHeight() <
dimension.height) {
            dimension.width = image.getWidth();
            dimension.height = image.getHeight();
        }
        setPreferredSize(dimension);

    }

    private int drawLine(Graphics2D g2d, String line, FontMetrics metrics, int
yPos, Color color) {
    /**
    * Draws a line of text at the spsecified y position and color.
    * confidence
    *
    * @param g2D The Graphics2D object for the image.
    * @param line The line of text to draw.
    * @param metrics The font information to use.
    * @param yPos The y position for the line of text.
    *
    * @return  The yPos for the next line of text.
    */
```

```
        int indent = 10;

        // Get text height, width, and descent.
        int textWidth = metrics.stringWidth(line);
        LineMetrics lm = metrics.getLineMetrics(line, g2d);
        int textHeight = (int) lm.getHeight();
        int descent = (int) lm.getDescent();

        int y2Pos = (yPos + textHeight) - descent;

        // Draw black rectangle.
        g2d.setColor(Color.BLACK);
        g2d.fillRect(indent, yPos, textWidth, textHeight);

        // Draw text.
        g2d.setColor(color);
        g2d.drawString(line, indent, y2Pos);

        yPos += textHeight;

        return yPos;

    }

    public void drawImageInfo(DetectAnomalyResult result) {
    /**
     * Draws the results from DetectAnomalies onto the output image.
     *
     * @param result The response from a call to
     *               DetectAnomalies.
     *
     */

        // Set up drawing.
        Graphics2D g2d = image.createGraphics();

        if (result.anomalyMask() != null){
            Composite composite = g2d.getComposite();
            g2d.setComposite(AlphaComposite.SrcOver.derive(0.5f));
            int x = (image.getWidth() - maskImage.getWidth()) / 2;
            int y = (image.getHeight() - maskImage.getHeight()) / 2;
            g2d.drawImage(maskImage, x, y, null);
            // Set composite for overlaying text.
            g2d.setComposite(composite);
```

```
        }

        //Calculate font size based on arbitary 32 pixel image width.
        int fontSize = (image.getWidth() / 32);


        g2d.setFont(new Font("Tahoma", Font.PLAIN, fontSize));
        Font font = g2d.getFont();
        FontMetrics metrics = g2d.getFontMetrics(font);

        // Get classification information.

        String prediction = "Prediction: Normal";

        if (Boolean.TRUE.equals(result.isAnomalous())) {
            prediction = "Prediction: Anomalous";
        }

        // Convert prediction to percentage.
        NumberFormat defaultFormat = NumberFormat.getPercentInstance();
        defaultFormat.setMinimumFractionDigits(1);
        String confidence = String.format("Confidence: %s",
defaultFormat.format(result.confidence()));

        // Draw classification information.
        int yPos = 0;

        yPos = drawLine(g2d, "Classification:", metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, prediction, metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, confidence, metrics, yPos, Color.WHITE);

        // Draw segmentation info.
        yPos = drawLine(g2d, "Segmentation:", metrics, yPos, Color.WHITE);

        // Ignore background label, so size must be > 1
        if (result.anomalies().size() > 1) {
            for (Anomaly anomaly : result.anomalies()) {
                if (anomaly.name().equals("background"))
                    continue;
                String label = String.format("Anomaly: %s. Area: %s",
anomaly.name(),

defaultFormat.format(anomaly.pixelAnomaly().totalPercentageArea()));
```

```
                Color anomalyColor =
Color.decode((anomaly.pixelAnomaly().color()));
                yPos = drawLine(g2d, label, metrics, yPos, anomalyColor);

            }

        } else {
            drawLine(g2d, "None found.", metrics, yPos, Color.WHITE);

        }

        g2d.dispose();

    }

    @Override
    public void paintComponent(Graphics g)
    /**
     * Draws the image and analysis results.
     *
     * @param g The Graphics context object for drawing.
     *          DetectAnomalies.
     *
     */
    {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.

    private String getImageType(byte[] image) throws IOException
    /**
     * Gets the file type of a supplied image. Raises an exception if the image
     * isn't compatible with with Amazon Lookout for Vision.
     *
     * @param image The image that you want to check.
     *
     * @return String The type of the image.
     */
```

```
   {
       InputStream is = new BufferedInputStream(new
ByteArrayInputStream(image));
       String mimeType = URLConnection.guessContentTypeFromStream(is);

       logger.log(Level.INFO, "Image type: {0}", mimeType);

       if (mimeType.equals("image/jpeg") || mimeType.equals("image/png")) {
           return mimeType;
       }
       // Not a supported file type.
       logger.log(Level.SEVERE, "Unsupported image type: {0}", mimeType);
       throw new IOException(String.format("Wrong image type. %s format isn't
supported.", mimeType));
   }

   public static void main(String[] args) throws Exception {

       String photo = null;
       String projectName = null;
       String modelVersion = null;

       final String USAGE = "\n" +
               "Usage:\n" +
               "    DetectAnomalies <project> <version> <image> \n\n" +
               "Where:\n" +
               "    project - The Lookout for Vision project.\n\n" +
               "    version - The version of the model within the project.\n\n"
+
               "    image - The path and filename of a local image. \n\n";

       try {

           if (args.length != 3) {
               System.out.println(USAGE);
               System.exit(1);
           }

           projectName = args[0];
           modelVersion = args[1];
           photo = args[2];
           ShowAnomalies panel = null;
```

```
            // Get the Lookout for Vision client.
            LookoutVisionClient lfvClient = LookoutVisionClient.builder()

.credentialsProvider(ProfileCredentialsProvider.create("lookoutvision-access"))
                .build();

            // Create frame and panel.
            JFrame frame = new JFrame(photo);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            panel = new ShowAnomalies(lfvClient, projectName, modelVersion,
 photo);

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);


        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Lookout for Vision client error: {0}:
 {1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("lookout for vision client error:
 %s", lfvError.getMessage()));
            System.exit(1);

        } catch (FileNotFoundException fileError) {
            logger.log(Level.SEVERE, "Could not find file: {0}",
 fileError.getMessage());
            System.out.println(String.format("Could not find file: %s",
 fileError.getMessage()));
            System.exit(1);

        } catch (IOException ioError) {
            logger.log(Level.SEVERE, "IO error {0}", ioError.getMessage());
            System.out.println(String.format("IO error: %s",
 ioError.getMessage()));
            System.exit(1);
        }

    }
}
```

4. Se você não planeja continuar usando seu modelo, interrompa seu modelo.

Encontrando anomalias com uma função AWS Lambda

O AWS Lambda é um serviço de computação que permite executar código sem o provisionamento ou gerenciamento de 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 chama DetectAnomalias. A função analisa uma imagem fornecida e retorna uma classificação para a presença de anomalias nessa imagem. As instruções incluem um exemplo de código Python que mostra como chamar a função Lambda com uma imagem em um bucket do Amazon S3 ou uma imagem fornecida por um computador local.

Tópicos

- Etapa 3: Criar uma função do AWS Lambda (console)
- Etapa 2: (Opcional) Criar uma camada (console)
- Etapa 3: Adicionar código do Python (console)
- Etapa 4: testar sua função do Lambda

Etapa 3: Criar uma função do AWS Lambda (console)

Nesta etapa, você cria uma função AWS vazia e uma função de execução do IAM que permite que sua função chame a operação `DetectAnomalies`. Ele também concede acesso ao bucket do Amazon S3 que armazena imagens para análise. Você também especifica variáveis de ambiente para o seguinte:

- A versão do projeto e modelo do Amazon Lookout for Vision que você deseja que sua função 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 Lambda.

Para criar uma função AWS Lambda (console)

1. Faça login no AWS Management Console e abra o console AWS Lambda em https://console.aws.amazon.com/lambda/.
2. Escolha 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 3.8.
4. Escolha Criar função para criar a função do AWS Lambda.
5. Em sua página da função Lambda, 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, em seguida, insira a chave e o valor da variável.

Chave	Valor
NOME_DO_PROJETO	O projeto Lookout for Vision que contém o modelo que você deseja usar.
VERSÃO_MODELO	A versão do mecanismo de banco de dados que você deseja usar.
confidence	O valor mínimo (0-100) para a confiança do modelo de que a previsão é anômala. Se a confiança for menor, a classificação é considerada normal.

8. Escolha Salvar para salvar as variáveis de ambiente.
9. No painel Permissões, em Nome da função, escolha a função de execução para abrir a função no console do IAM.
10. Na guia Permissões, escolha Adicionar permissões), Criar política em linha.
11. Escolha JSON e substitua a política padrão com a política a seguir.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "lookoutvision:DetectAnomalies",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectAnomaliesAccess"
        }
    ]
}
```

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. Substitua o *caminho do bucket/pasta* pelo bucket do Amazon S3 e o caminho da pasta para as imagens que deseja analisar.

    ```
    {
        "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) Criar uma camada (console)

Para executar esse exemplo, não é preciso executar essa etapa. A operação `DetectAnomalies` está incluída no ambiente padrão do Lambda Python como parte do SDK for Python (Boto3) da

AWS. Se outras partes da sua função do Lambda precisarem de atualizações da 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, crie um arquivo .zip que contém o SDK do Boto3. Em seguida, você cria uma camada e adiciona o arquivo de arquivos.zip à camada. Para obter mais informações, consulte Usar camadas com sua função do Lambda.

Para criar e adicionar uma camada (console)

1. Abra um prompt de comando e digite os seguintes comandos.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Anote o nome do arquivo zip (boto3-layer.zip). Você precisará dela na etapa 6 deste procedimento.
3. Abra o console do AWS Lambda 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 Arquivo .zip e, em seguida, escolha Fazer upload.
8. Na caixa de diálogo, escolha o arquivo de arquivos.zip (boto3-layer.zip) que você criou na etapa 1 desse procedimento.
9. Para tempos de execução compatíveis, escolha Python 3.9.
10. Escolha Criar para criar a tarefa.
11. Escolha o ícone do menu do painel de navegação.
12. No painel de navegação, escolha Funções.
13. Na lista de recursos, selecione a função que você criou no Etapa 3: Criar uma função do AWS Lambda (console).
14. Escolha a guia Código.
15. Na área 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: Adicionar código do Python (console)

Nesta etapa, você adiciona código Python à sua função Lambda usando o editor de código do console Lambda. O código analisa uma imagem fornecida com `DetectAnomalies` e retorna uma classificação (verdadeira se a imagem for anômala, falsa se a imagem for normal). A imagem fornecida pode estar localizada em um bucket do Amazon S3 ou fornecida como bytes de imagem codificados em byte64.

Para adicionar código Python (console)

1. Se você não estiver no console do Lambda, faça o seguinte:
   a. Abra o console do AWS Lambda em https://console.aws.amazon.com/lambda/.
   b. Abra a função Lambda que você criou em Etapa 3: Criar uma função do AWS Lambda (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 Amazon Lookout for Vision
 model.
"""
import base64
import imghdr
from os import environ
from io import BytesIO
import logging
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)
```

```
# Get the model and confidence.
project_name = environ['PROJECT_NAME']
model_version = environ['MODEL_VERSION']
min_confidence = int(environ.get('CONFIDENCE', 50))

lookoutvision_client = boto3.client('lookoutvision')


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:

        file_name = ""

        # Determine image source.
        if 'image' in event:
            # Decode the encoded image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image_type = get_image_type(img_b64decoded)
            image = BytesIO(img_b64decoded)
            file_name = event['filename']

        elif 'S3Object' in event:
            bucket = boto3.resource('s3').Bucket(event['S3Object']['Bucket'])
            image_object = bucket.Object(event['S3Object']['Name'])
            image = image_object.get().get('Body').read()
            image_type = get_image_type(image)
            file_name = f"s3://{event['S3Object']['Bucket']}/{event['S3Object']
['Name']}"

        else:
            raise ValueError(
                'Invalid image source. Only base 64 encoded image bytes or images
 in S3 buckets are supported.')
```

```
        # Analyze the image.
        response = lookoutvision_client.detect_anomalies(
            ProjectName=project_name,
            ContentType=image_type,
            Body=image,
            ModelVersion=model_version)

        reject = reject_on_classification(
            response['DetectAnomalyResult'],
confidence_limit=float(environ['CONFIDENCE'])/100)

        status = "anomalous" if reject else "normal"

        lambda_response = {
            "statusCode": 200,
            "body": {
                "Reject": reject,
                "RejectMessage": f"Image {file_name} is {status}."
            }
        }

    except ClientError as err:
        error_message = f"Couldn't analyze {file_name}. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message,
                "Image": file_name
            }
        }
        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),
```

```
                "Image": event['filename']
            }
        }
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, format(val_error))

    return lambda_response

def get_image_type(image):
    """
    Gets the format of the image. Raises an error
    if the type is not PNG or JPEG.
    :param image: The image that you want to check.
    :return The type of the image.

    """
    image_type = imghdr.what(None, image)

    if image_type == "jpeg":
        content_type = "image/jpeg"
    elif image_type == "png":
        content_type = "image/png"
    else:
        logger.info("Invalid image type")
        raise ValueError(
            "Invalid file format. Supply a jpeg or png format file.")
    return content_type


def reject_on_classification(prediction, confidence_limit):
    """
    Returns True if the anomaly confidence is greater than or equal to
    the supplied confidence limit.
    :param image: The name of the image file that was analyzed.
    :param prediction: The DetectAnomalyResult object returned from DetectAnomalies
    :param confidence_limit: The minimum acceptable confidence. Float value between
 0 and 1.
    :return: True if the error condition indicates an anomaly, otherwise False.
    """

    reject = False

    if prediction['IsAnomalous'] and prediction['Confidence'] >= confidence_limit:
        reject = True
```

```
        reject_info = (f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) is greater"
                       f" than limit ({confidence_limit:.2%})")
        logger.info("%s", reject_info)

    if not reject:
        logger.info("No anomalies found.")
    return reject
```

4. Escolha Deploy para implantar sua função do Lambda.

Etapa 4: testar sua função do Lambda

Nesta etapa, você usa o código Python em seu computador para passar uma imagem local, ou uma imagem em um bucket do Amazon S3, para sua função Lambda. As imagens passadas de um computador local devem ter menos de 6291456 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 informações sobre o upload de arquivos de imagem para um bucket do Amazon S3, consulte Upload de objetos.

Certifique-se de executar o código na mesma região da AWS em que você criou a função Lambda. Você pode visualizar a região da AWS para sua função Lambda na barra de navegação da página de detalhes da função no console do Lambda.

Se a função AWS Lambda retornar um erro de tempo limite, estenda o período de tempo limite da função da função Lambda. Para obter mais informações, consulte Configuração do tempo limite da função (console).

Para obter mais informações sobre como invocar uma função Lambda a partir do seu código, consulte Invocando funções AWS Lambda.

Para testar sua função Lambda

1. Se ainda não tiver feito isso, faça o seguinte:

    a. Certifique-se de que o usuário que está usando o código do cliente tenha permissão `lambda:InvokeFunction`. Você pode usar as seguintes permissões.

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

```
        {
            "Sid": "LambdaPermission",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

Você pode obter o ARN para sua função Lambda na visão geral da função no console Lambda.

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

- Usuários e grupos no AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Create a permission set (Criação de um conjunto de permissões) no Guia do usuário do AWS IAM Identity Center.

- Usuários gerenciados no IAM usando um provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criar 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 Creating a role for an IAM user (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.

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

c. Inicie o modelo que você especificou na etapa 7 do Etapa 3: Criar uma função do AWS Lambda (console).

2. 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: Shows how to call the anomaly detection
AWS Lambda function.
"""
from botocore.exceptions import ClientError

import argparse
import logging
import base64
import json
import boto3
from os import environ

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, "filename": image}

    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()

        # Analyze image and display results.

        result = analyze_image(args.function, args.image)

        status = result['statusCode']

        if status == 200:
            classification = result['body']
            print(f"classification: {classification['Reject']}")
            print(f"Message: {classification['RejectMessage']}")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")
```

```
    except ClientError as error:
        logging.error(error)
        print(error)


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

3. Execute o código. Para o argumento da linha de comando, forneça o nome da função Lambda e o caminho para uma imagem local que você deseja analisar. Por exemplo:

```
python client.py function_name /bucket/path/image.jpg
```

   Se for bem-sucedida, a saída será uma classificação das anomalias encontradas na imagem. Se uma classificação não for retornada, considere reduzir o valor de confiança que você definiu na etapa 7 do Etapa 3: Criar uma função do AWS Lambda (console).

4. Se você tiver concluído a função Lambda e o modelo não for usado por outros aplicativos, interrompa o modelo. Lembre-se de iniciar o modelo na próxima vez que quiser usar a função Lambda.

Usando seu modelo Amazon Lookout for Vision em um dispositivo de ponta

Você pode usar seu modelo Amazon Lookout for Vision em dispositivos periféricos gerenciados peloAWS IoT Greengrass Version 2. O AWS IoT Greengrass é um serviço de nuvem e tempo de execução de ponta da Internet das Coisas (IoT) de código aberto. Você pode usá-lo para criar, implantar e gerenciar aplicativos de IoT em seus dispositivos. Para obter mais informações, consulte AWS IoT Greengrass.

Você implanta os mesmos modelos do Amazon Lookout for Vision que você treinou na nuvem em dispositivos de AWS IoT Greengrass V2 ponta compatíveis. Em seguida, você pode usar seu modelo implantado para realizar a detecção de anomalias no local, como no chão de fábrica, sem transmitir dados continuamente para a nuvem. Dessa forma, você pode minimizar os custos de largura de banda e detectar anomalias localmente com a análise de imagens em tempo real.

> **Tip**
>
> Antes de implantar um modelo do Lookout for Vision AWS IoT Greengrass com, recomendamos que você leia AWS IoT Greengrass Version 2o guia do desenvolvedor. Para obter mais informações, consulte O que é o AWS IoT Greengrass?

Para usar um modelo do Lookout for Vision em AWS IoT Greengrass V2 um dispositivo principal, você implanta o modelo e o software de suporte como componentes do dispositivo principal. Um componente é um módulo de software, como o modelo Lookout for Vision, executado em um dispositivo principal do Greengrass. Existem duas formas de componente. Um componente personalizado é um componente que você cria e só pode ser acessado por você. Também é conhecido como um componente privado. Um componente AWS fornecido é um componente pré-construído que AWS fornece. Também é conhecido como um componente público. Para obter mais informações, consulte https://docs.aws.amazon.com/greengrass/v2/developerguide/public-components.html.

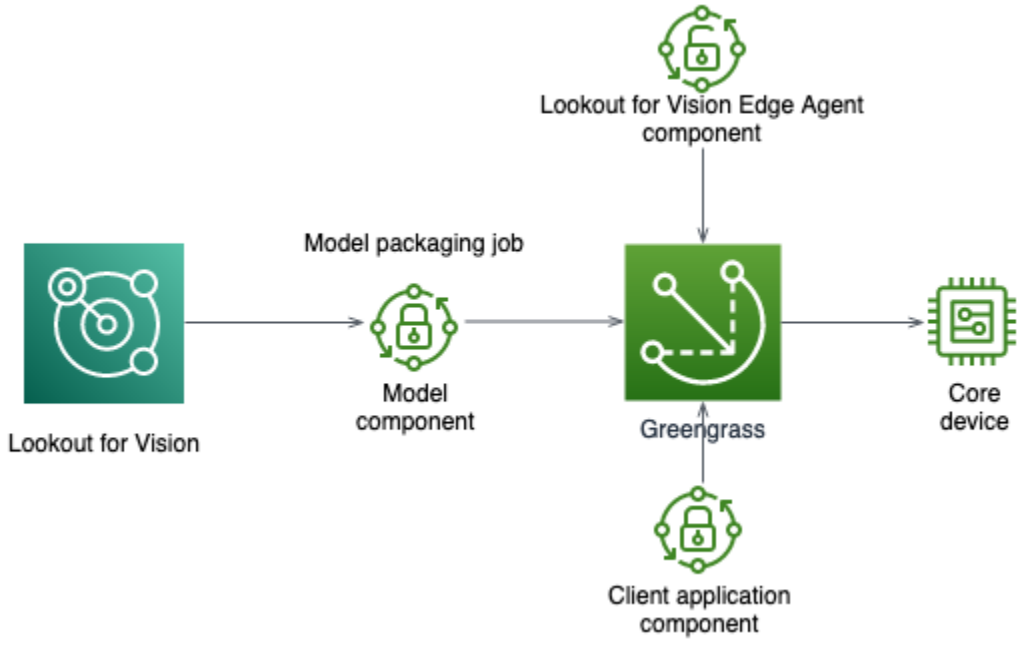
Os componentes que você implanta em um dispositivo principal para um modelo e software de suporte do Lookout for Vision são:

- Componente do modelo. Um componente personalizado que contém seu modelo do Lookout for Vision. Para criar o componente do modelo, você usa o Lookout for Vision para criar um

trabalho de empacotamento do modelo. Um trabalho de empacotamento de modelo cria um componente para o modelo e o disponibiliza como um componente personalizado dentro deleAWS IoT Greengrass V2. Para obter mais informações, consulte Empacotar seu modelo Amazon Lookout for Vision.

- Componente do aplicativo cliente. Um componente personalizado que você cria que implementa o código para suas necessidades comerciais. Por exemplo, encontrar placas de circuito anômalas a partir de imagens tiradas após a montagem. Para obter mais informações, consulte Escrevendo seu componente de aplicativo cliente.
- Componente Amazon Lookout for Vision Edge Agent. Um componente AWS fornecido que fornece uma API para usar e gerenciar seu modelo. Por exemplo, o código no componente do aplicativo cliente pode usar a `DetectAnomalies` API para detectar anomalias nas imagens. O componente Lookout for Vision Edge Agent é uma dependência do componente do modelo. Ele é instalado automaticamente no dispositivo principal quando você implanta o componente do modelo. Para obter mais informações, consulte Referência da API do Amazon Lookout for Vision.

Depois de criar o componente modelo e o componente do aplicativo cliente, você pode usar AWS IoT Greengrass V2 para implantar os componentes e dependências no dispositivo principal. Para obter mais informações, consulte Implantando seus componentes em um dispositivo.



> **Important**
>
> As previsões que seu modelo faz `DetectAnomalies` em um dispositivo principal podem ser diferentes das previsões feitas usando o mesmo modelo hospedado na nuvem. Recomendamos que você teste seu modelo em um dispositivo principal antes de usá-lo em um ambiente de produção.
> Para reduzir as incompatibilidades de previsão entre modelos hospedados em dispositivos e modelos hospedados na nuvem, recomendamos aumentar o número de imagens normais e anômalas em seu conjunto de dados de treinamento. Não recomendamos reutilizar imagens existentes para aumentar o tamanho do conjunto de dados de treinamento.

Implantação de um modelo e componente de aplicativo cliente em um dispositivo AWS IoT Greengrass Version 2 principal

O procedimento para implantar um modelo Amazon Lookout for Vision e um componente de aplicativo cliente em AWS IoT Greengrass Version 2 um dispositivo principal é o seguinte:

1. Configure seus dispositivos principais comAWS IoT Greengrass Version 2.
2. Crie um trabalho de empacotamento de modelo usando o Lookout for Vision. O trabalho cria seu componente de modelo.
3. Escreva um componente do aplicativo cliente. O componente implementa sua lógica de negócios.
4. Implante o componente do modelo e o componente do aplicativo cliente no dispositivo principal usandoAWS IoT Greengrass V2.

Depois que os componentes e dependências forem implantados no dispositivo principal, você poderá usar o modelo no dispositivo principal.

> **Note**
>
> Você deve usar a mesma AWS região e AWS conta para criar e implantar seu modelo e componente de aplicativo cliente do Lookout for Vision.

AWS IoT Greengrass Version 2requisitos do dispositivo principal

Para usar um modelo Amazon Lookout for Vision em AWS IoT Greengrass Version 2 um dispositivo principal, seu modelo tem vários requisitos do dispositivo principal.

Tópicos

Dispositivos, arquiteturas de chip e sistemas operacionais testados

Esperamos que o Amazon Lookout for Vision funcione no seguinte hardware:

- Arquiteturas de CPU
  - X86_64 (versão de 64 bits do conjunto de instruções x86)
  - Aarch64 (CPU ARMv8 de 64 bits)
- (Somente inferência acelerada por GPU) Acelerador de GPU NVIDIA com capacidade de memória suficiente (pelo menos 6,0 GB para um modelo em execução).

A equipe do Amazon Lookout for Vision testou os modelos do Lookout for Vision nos seguintes dispositivos, arquiteturas de chips e sistemas operacionais.

Dispositivos

Dispositivo	Sistema operacional	Arquitetura	Accelerator	Opções do compilador
jetson_xavier (NVIDIA® Jetson AGX Xavier)	Linux	Aarch64	NVIDIA	`{"gpu-code": "sm_72", "trt-ver": "7.1.3", "cuda-ver": "10.2"}`

Dispositivo	Sistema operacional	Arquitetura	Accelerator	Opções do compilador
				`{"gpu-code": "sm_72", "trt-ver": "8.2.1", "cuda-ver": "10.2"}`
g4dn.xlarge (instâncias EC2 (G4) com GPUs NVIDIA T4 Tensor Core)	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}`
g5.xlarge (instâncias EC2 (G5) com GPUs NVIDIA A10G Tensor Core)	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_80", "trt-ver": "8.2.0", "cuda-ver": "11.2"}`
c5.2xlarge (Instâncias C5 do Amazon EC2)	Linux	X86_64/X86-64	CPU	`{"mcpu": "core-avx2"}`

Memória e armazenamento do dispositivo principal

Para executar um único modelo e o Amazon Lookout for Vision Edge Agent, seu dispositivo principal tem os seguintes requisitos de memória e armazenamento. Talvez você precise de mais memória e armazenamento para o componente do aplicativo cliente.

- Armazenamento — Pelo menos 1,5 GB.
- Memória — Pelo menos 6,0 GB para um modelo em execução.

Software necessário

Um dispositivo principal requer o seguinte software.

Dispositivos Jetson

Se o seu dispositivo principal for um dispositivo Jetson, você precisará do seguinte software instalado no dispositivo principal.

Software	Versões com suporte
SDK do Jetpack	4.4 a 4.6.1
Python e ambiente virtual Python para o Lookout for Vision Edge Agent versão 1.x	3.8 ou 3.9

X86 hardware

Se o seu dispositivo principal usa hardware x86, você precisa do seguinte software instalado no dispositivo principal.

Inferência de CPU

Software	Versões com suporte
Python e ambiente virtual Python para o Lookout for Vision Edge Agent versão 1.x	3.8 ou 3.9

Inferência acelerada por GPU

As versões do software variam de acordo com a microarquitetura da GPU NVIDIA que você usa.

GPU NVIDIA com microarquitetura anterior ao Ampere (a capacidade de computação é inferior a 8,0)

Software necessário para uma GPU NVIDIA com uma microarquitetura anterior ao Ampere (capacidade de computação menor que 8,0). O `gpu-code` deve ser menor que `sm_80`.

Software	Versões com suporte
NVIDIA CUDA	10.2
NVIDIA TensorRT	Pelo menos 7.1.3 e menos de 8.0.0
Python e ambiente virtual Python para o Lookout for Vision Edge Agent versão 1.x	3.8 ou 3.9

GPU NVIDIA com microarquitetura Ampere (capacidade computacional 8.0)

Software necessário para uma GPU NVIDIA com a microarquitetura Ampere (a capacidade de computação é 8.0). O `gpu-code` deve ser `sm_80`.

Software	Versões com suporte
NVIDIA CUDA	11.2
NVIDIA TensorRT	8.2.0
Python e ambiente virtual Python para o Lookout for Vision Edge Agent versão 1.x	3.8 ou 3.9

Configurando seu dispositivo AWS IoT Greengrass Version 2 principal

O Amazon Lookout for Vision AWS IoT Greengrass Version 2 usa para simplificar a implantação do componente de modelo, do componente Amazon Lookout for Vision Edge Agent e do componente de aplicativo cliente em seu dispositivo principal. AWS IoT Greengrass V2 Para obter informações sobre os dispositivos e o hardware que você pode usar, consulteAWS IoT Greengrass Version 2requisitos do dispositivo principal.

Configuração de seu dispositivo principal

Use as informações a seguir para configurar seu dispositivo principal.

Para configurar seu dispositivo principal

1. Configurar suas bibliotecas com GPU. Não execute essa etapa se você não estiver usando a inferência acelerada por GPU.

   a. Verifique se você tem uma GPU compatível com CUDA. Para obter mais informações, consulte Verificar se você tem uma GPU compatível com CUDA.

   b. Configure CUDA, cuDNN e TensorRT no dispositivo seguindo um destes procedimentos:

      - Se você estiver usando um dispositivo Jetson, instale o JetPack versão 4.4 a 4.6.1. Para obter mais informações, consulte JetPack Archive.
      - Se você estiver usando hardware baseado em x86 e sua microarquitetura de GPU NVIDIA for anterior ao Ampere (a capacidade de computação é menor que 8,0), faça o seguinte:
        1. Configure o CUDA versão 10.2 seguindo as instruções no Guia de instalação do NVIDIA CUDA para Linux.
        2. Instale o cuDNN seguindo as instruções na documentação do cuDNN da NVIDIA.
        3. Configure o TensorRT (versão 7.1.3 ou posterior, mas anterior à 8.0.0) seguindo as instruções na DOCUMENTAÇÃO DO NVIDIA TENSORRT.
      - Se você estiver usando hardware baseado em x86 e sua microarquitetura de GPU NVIDIA for Ampere (a capacidade de computação é 8.0), faça o seguinte:
        1. Configure o CUDA (versão 11.2) seguindo as instruções no Guia de instalação do NVIDIA CUDA para Linux.
        2. Instale o cuDNN seguindo as instruções na documentação do cuDNN da NVIDIA.
        3. Configure o TensorRT (versão 8.2.0) seguindo as instruções na DOCUMENTAÇÃO DO NVIDIA TENSORRT.

2. Instale o software AWS IoT Greengrass Version 2 principal em seu dispositivo principal. Para obter mais informações, consulte Instalar o software AWS IoT Greengrass Core no Guia do desenvolvedor do AWS IoT Greengrass Version 2.

3. Para ler do bucket do Amazon S3 que armazena o modelo, anexe permissão à função do IAM (função de troca de tokens) que você cria durante a AWS IoT Greengrass Version 2

configuração. Para obter mais informações, consulte Permitir acesso a buckets do S3 para artefatos de componentes.

4. No prompt de comando, digite o comando a seguir para instalar o Python e um ambiente virtual Python no dispositivo principal.

```
sudo apt install python3.8 python3-venv python3.8-venv
```

5. Use o comando a seguir para adicionar o usuário do Greengrass ao grupo de vídeo. Isso permite que os componentes implantados do Greengrass acessem a GPU:

```
sudo usermod -a -G video ggc_user
```

6. (Opcional) Se você quiser chamar a API do Lookout for Vision Edge Agent de um usuário diferente, adicione o usuário necessário ao`ggc_group`. Isso permite que o usuário se comunique com o Lookout for Vision Edge Agent pelo soquete Unix Domain:

```
sudo usermod -a -G ggc_group $(whoami)
```

Empacotar seu modelo Amazon Lookout for Vision

Um trabalho de empacotamento de modelos empacota um modelo Amazon Lookout for Vision como um componente do modelo.

Para criar uma tarefa de empacotamento de modelo, você escolhe o modelo que deseja empacotar e fornece configurações para o componente de modelo que a tarefa cria. Você só pode empacotar um modelo que tenha sido treinado com sucesso.

Você pode usar o console AWS ou o SDK do Lookout for Vision para criar a tarefa de empacotamento do modelo. Você também pode obter informações sobre os trabalhos de empacotamento de modelos que você cria. Para obter mais informações, consulte Obter informações sobre trabalhos de empacotamento de modelos. Você pode usar o AWS IoT Greengrass V2 console ou o AWS SDK para implantar os componentes no dispositivo AWS IoT Greengrass Version 2 principal.

Tópicos

- Configurações do pacote
- Empacotando seu modelo (console)

- Empacotando seu modelo (SDK)
- Obter informações sobre trabalhos de empacotamento de modelos

Configurações do pacote

Use as informações a seguir para decidir as configurações do pacote para seu trabalho de empacotamento de modelo.

Para criar um trabalho de empacotamento de modelo, consulte Empacotando seu modelo (console) ouEmpacotando seu modelo (SDK).

Tópicos

- Hardware de destino
- Configurações do componente

Hardware de destino

Você pode escolher um dispositivo ou plataforma de destino para seu modelo, mas não ambos. Para obter mais informações, consulte Dispositivos, arquiteturas de chip e sistemas operacionais testados.

Target Device

O dispositivo alvo do modelo, como o NVIDIA® Jetson AGX Xavier. Não é necessário especificar as opções do compilador.

Plataforma de destino

O Amazon Lookout for Vision oferece suporte às seguintes configurações de plataforma:

- Arquiteturas X86_64 (versão de 64 bits do conjunto de instruções x86) e Aarch64 (CPU ARMv8 de 64 bits).
- Sistema operacional Linux
- Inferência usando aceleradores NVIDIA ou CPU.

Você precisa especificar as opções corretas do compilador para sua plataforma de destino.

Opções do compilador

As opções do compilador permitem que você especifique a plataforma de destino para seu dispositivo AWS IoT Greengrass Version 2 principal. Atualmente, você pode especificar as seguintes opções do compilador.

Aceleradora NVIDIA

- `gpu-code`— Especifica o código da gpu do dispositivo principal que executa o componente do modelo.
- `trt-ver`— Especifica a versão do TensorRT no formato x.y.z.
- `cuda-ver`— Especifica a versão CUDA no formato x.y.

Acelerador de CPU

- (Opcional) `mcpu` — especifica o conjunto de instruções. Por exemplo `core-avx2`. Se você não fornecer um valor, o Lookout for Vision usará o `core-avx2` valor.

Você especifica as opções no formato JSON. Por exemplo:

```
{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}
```

Para obter mais exemplos, consulte Dispositivos, arquiteturas de chip e sistemas operacionais testados.

Configurações do componente

O trabalho de empacotamento do modelo cria um componente de modelo que contém seu modelo. O trabalho cria artefatos que são AWS IoT Greengrass V2 usados para implantar o componente do modelo no dispositivo principal.

Você não pode criar um componente de modelo com o mesmo nome e versão de componente de um componente existente.

Nome do componente

Um nome para o componente do modelo que o Lookout for Vision cria durante o empacotamento do modelo. O nome do componente que você especifica é exibido no AWS IoT Greengrass V2 console.

Você usa o nome do componente na receita que você cria para o componente do aplicativo cliente. Para obter mais informações, consulte Criando o componente do aplicativo cliente.

Descrição do componente

(Opcional) Uma descrição do componente do modelo.

Versão do componente.

Um número de versão para o componente do modelo. Você pode aceitar o número de versão padrão ou escolher o seu próprio. O número da versão deve seguir o sistema semântico de números da versão — major.minor.patch. Por exemplo, a versão 1.0.0 representa a primeira versão principal de um componente. Para obter mais informações, consulte Versionamento semântico 2.0.0. Se você não fornecer um valor, o Lookout for Vision usará o número da versão do seu modelo para gerar uma versão para você.

Localização do componente

O local do Amazon S3 em que você deseja que o trabalho de empacotamento do modelo salve os artefatos do componente do modelo. O bucket do Amazon S3 deve estar na mesma região da AWS e conta da AWS em que você usa. AWS IoT Greengrass Version 2 Para criar um bucket do Amazon S3, consulte Criação de um bucket.

Tags

Você pode identificar, organizar, pesquisar e filtrar seus componentes usando tags. Cada tag é um rótulo que consiste em um valor e uma chave definida pelo usuário. As tags são anexadas ao componente do modelo quando o trabalho de empacotamento do modelo cria o componente do modelo no Greengrass. Um componente é um recurso do AWS IoT Greengrass V2. As tags não estão anexadas a nenhum dos seus recursos do Lookout for Vision, como seus modelos. Para obter mais informações, consulte Como rotular recursos da AWS.

Empacotando seu modelo (console)

Você pode criar um trabalho de empacotamento de modelos usando o console Amazon Lookout for Vision.

Para obter informações sobre as configurações do pacote, consulte Configurações do pacote.

Para empacotar um modelo (console)

1. Crie um bucket do Amazon S3 ou reutilize um bucket existente que o Lookout for Vision usa para armazenar os artefatos do trabalho de empacotamento (componente do modelo).
2. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
3. Escolha Como começar.
4. No painel de navegação esquerdo, selecione Projetos.
5. Na seção Projetos, escolha o projeto que contém o modelo que você deseja empacotar.
6. No painel de navegação à esquerda, abaixo do nome do projeto, escolha Pacotes de modelos do Edge.
7. Na seção Modelar trabalhos de empacotamento, escolha Criar trabalho de empacotamento modelo.
8. Insira as configurações do pacote. Para obter mais informações, consulte Configurações do pacote.
9. Escolha Criar pacote de modelos.
10. Espere até que o trabalho de embalagem termine. O trabalho será concluído quando o status do trabalho for Sucesso.
11. Escolha o trabalho de empacotamento na seção Modelar trabalhos de empacotamento.
12. Escolha Continuar implantação no Greengrass para continuar a implantação de seu componente de modelo em. AWS IoT Greengrass Version 2 Para obter mais informações, consulte Implantando seus componentes em um dispositivo.

Empacotando seu modelo (SDK)

Você empacota um modelo como um componente de modelo criando uma tarefa de empacotamento de modelo. Para criar um trabalho de empacotamento de modelo, você chama a API startModelPackagingJob. O trabalho pode demorar um pouco para ser concluído. Para descobrir o status atual, chame DescribeModelPackagingJob e verifique o campo na resposta. `Status`

Para obter informações sobre as configurações do pacote, consulte Configurações do pacote.

O procedimento a seguir mostra como iniciar um trabalho de empacotamento usando a AWS CLI. Você pode empacotar o modelo para uma plataforma de destino ou um dispositivo de destino. Por exemplo, código Java, consulte StartModelPackagingJob.

Para empacotar seu modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs da AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Verifique se o você tem as seguintes permissões para iniciar um trabalho de empacotamento de modelos. Para obter mais informações, consulte StartModelPackagingJob.
3. Use os seguintes comandos da CLI para empacotar seu modelo para um dispositivo de destino ou uma plataforma de destino.

   Target platform

   O comando CLI a seguir mostra como empacotar um modelo para uma plataforma de destino com um acelerador NVIDIA.

   Altere os seguintes valores:

   - `project_name` para o nome do projeto que contém o modelo que você deseja empacotar.
   - `model_version` para a versão do modelo que você deseja empacotar.
   - (Opcional) `description` para uma descrição do seu trabalho de empacotamento de modelo.
   - `architecture` para a arquitetura (`ARM64` ou `X86_64`) do dispositivo AWS IoT Greengrass Version 2 principal em que você executa o componente do modelo.
   - `gpu_code` ao código gpu do dispositivo principal em que você executa o componente do modelo.
   - `trt_ver` para a versão do TensorRT que você instalou no seu dispositivo principal.
   - `cuda_ver` para a versão CUDA que você instalou no seu dispositivo principal.
   - `component_name` a um nome para o componente do modelo no qual você deseja criarAWS IoT Greengrass V2.
   - (Opcional) `component_version` para uma versão do componente de modelo que o trabalho de empacotamento cria. Use o formato `major.minor.patch`. Por exemplo, 1.0.0 representa a primeira versão principal de um componente.
   - `bucket` para o bucket do Amazon S3 no qual o trabalho de empacotamento armazena os artefatos do componente do modelo.
   - `prefix` para o local dentro do bucket do Amazon S3 onde o trabalho de empacotamento armazena os artefatos do componente do modelo.

- (Opcional) `component_description` a uma descrição para o componente do modelo.
- (Opcional) `tag_key1` e `tag_key2` às chaves das tags anexadas ao componente do modelo.
- (Opcional) `tag_value1` e `tag_value2` aos valores-chave das tags anexadas ao componente do modelo.

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='architecture',Accelerator='NVIDIA'},Compi
code\": \"gpu_code\", \"trt-ver\": \"trt_ver\", \"cuda-ver\":
 \"cuda_ver\"}',S3OutputLocation={Bucket='bucket',Prefix='prefix'},ComponentName='Compon
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

Por exemplo:

```
aws lookoutvision start-model-packaging-job \
  --project-name test-project-01 \
  --model-version 1 \
  --description="Model Packaging Job for G4 Instance using TargetPlatform
 Option" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='X86_64',Accelerator='NVIDIA'},CompilerOpt
code\": \"sm_75\", \"trt-ver\": \"7.1.3\", \"cuda-ver\":
 \"10.2\"}',S3OutputLocation={Bucket='bucket',Prefix='test-project-01/
folder'},ComponentName='SampleComponentNameX86TargetPlatform',ComponentVersion='0.1.0',C
 is my component',Tags=[{Key='modelKey0',Value='modelValue'},
 {Key='modelKey1',Value='modelValue'}]}" \
  --profile lookoutvision-access
```

Target Device

Use os seguintes comandos da CLI para empacotar um modelo para um dispositivo de destino.

Altere os seguintes valores:

- `project_name` para o nome do projeto que contém o modelo que você deseja empacotar.
- `model_version` para a versão do modelo que você deseja empacotar.
- (Opcional) `description` para uma descrição do seu trabalho de empacotamento de modelo.
- `component_name` a um nome para o componente do modelo no qual você deseja criarAWS IoT Greengrass V2.
- (Opcional) `component_version` para uma versão do componente de modelo que o trabalho de empacotamento cria. Use o formato `major.minor.patch`. Por exemplo, 1.0.0 representa a primeira versão principal de um componente.
- `bucket` para o bucket do Amazon S3 no qual o trabalho de empacotamento armazena os artefatos do componente do modelo.
- `prefix` para o local dentro do bucket do Amazon S3 onde o trabalho de empacotamento armazena os artefatos do componente do modelo.
- (Opcional) `component_description` a uma descrição para o componente do modelo.
- (Opcional) `tag_key1` e `tag_key2` às chaves das tags anexadas ao componente do modelo.
- (Opcional) `tag_value1` e `tag_value2` aos valores-chave das tags anexadas ao componente do modelo.

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='pre
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

Por exemplo:

```
aws lookoutvision start-model-packaging-job \
  --project-name project_01 \
  --model-version 1 \
```

```
 --description="description" \
 --configuration
"Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='com
model component',Tags=[{Key='tag_key1',Value='tag_value1'},
{Key='tag_key2',Value='tag_value2'}]}" \
 --profile lookoutvision-access
```

4. Observe o valor de JobName na resposta. Você precisa dele na próxima etapa. Por exemplo:

```
{
    "JobName": "6bcfd0ff-90c3-4463-9a89-6b4be3daf972"
}
```

5. Use `DescribeModelPackagingJob` para obter o status atual do trabalho. Altere o seguinte:

   - `project_name` ao nome do projeto que você está usando.
   - `job_name` para o nome do trabalho que você anotou na etapa anterior.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

   O trabalho de empacotamento do modelo será concluído se o valor de `Status` for`SUCCEEDED`. Se o valor for diferente, aguarde um minuto e tente novamente.

6. Continue a implantação usandoAWS IoT Greengrass V2. Para obter mais informações, consulte Implantando seus componentes em um dispositivo.

Obter informações sobre trabalhos de empacotamento de modelos

Você pode usar o console AWS e o SDK do Amazon Lookout for Vision para obter informações sobre os trabalhos de empacotamento de modelos que você cria.

Tópicos

- Obter informações sobre o trabalho de empacotamento do modelo (console)
- Obter informações sobre o trabalho de empacotamento do modelo (SDK)

Obter informações sobre o trabalho de empacotamento do modelo (console)

Para obter informações do trabalho de empacotamento do modelo (console)

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Como começar.
3. No painel de navegação esquerdo, selecione Projetos.
4. Na seção Projetos, escolha o projeto que contém o trabalho de empacotamento do modelo que você deseja visualizar.
5. No painel de navegação à esquerda, abaixo do nome do projeto, escolha Pacotes de modelos do Edge.
6. Na seção Trabalho de empacotamento do modelo, escolha o trabalho de empacotamento do modelo que você deseja visualizar. A página de detalhes do trabalho de empacotamento do modelo é exibida.

Obter informações sobre o trabalho de empacotamento do modelo (SDK)

Você pode usar o AWS SDK para listar os trabalhos de empacotamento de modelos em um projeto e obter informações sobre um trabalho de empacotamento de modelo específico.

Listar trabalhos de embalagem de modelos

Você pode listar os trabalhos de empacotamento de modelos em um projeto chamando a API ListModelPackagingJobs. A resposta inclui uma lista de objetos ModelPackagingJobMetadata que fornece informações sobre cada trabalho de empacotamento de modelo. Também está incluído um token de paginação que você pode usar para obter o próximo conjunto de resultados, se a lista estiver incompleta.

Para listar seus trabalhos de empacotamento de modelos

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs da AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Use o seguinte comando da CLI. Altere o `project_name` para o nome do projeto que você deseja usar.

```
aws lookoutvision list-model-packaging-jobs \
```

```
  --project-name project_name \
  --profile lookoutvision-access
```

Descrever um trabalho de empacotar modelos

Use a API DescribeModelPackagingJob para obter informações sobre um trabalho de empacotamento de modelos. A resposta é um objeto ModelPackagingDescription que inclui o status atual do trabalho e outras informações.

Para descrever um pacote

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs da AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Use o seguinte comando da CLI. Altere o seguinte:

   - `project_name` ao nome do projeto que você está usando.
   - `job_name` para o nome do trabalho. Você recebe o nome do trabalho (`JobName`) ao chamar startModelPackagingJob.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

Escrevendo seu componente de aplicativo cliente

Um componente do aplicativo cliente é um AWS IoT Greengrass Version 2 componente personalizado que você escreve. Ele implementa a lógica de negócios necessária para usar um modelo Amazon Lookout for Vision em AWS IoT Greengrass Version 2 um dispositivo principal.

Para acessar um modelo, seu componente de aplicativo cliente usa o componente Lookout for Vision Edge Agent. O componente Lookout for Vision Edge Agent fornece uma API que você usa para analisar imagens com um modelo e gerenciar os modelos em um dispositivo principal.

A API do Lookout for Vision Edge Agent é implementada usando gRPC, que é um protocolo para fazer chamadas de procedimentos remotos. Para obter mais informações, consulte gRPC. Para escrever seu código, você pode usar qualquer linguagem compatível com o gRPC. Fornecemos

exemplos de código Python. Para obter mais informações, consulte Usando um modelo em seu componente de aplicativo cliente.

> **Note**
>
> O componente Lookout for Vision Edge Agent é uma dependência do componente de modelo que você implanta. Ele é implantado automaticamente no dispositivo principal quando você implanta o componente do modelo no dispositivo principal.

Para escrever um componente de aplicativo cliente, faça o seguinte.

1. Configure seu ambiente para usar o gRPC e instalar bibliotecas de terceiros.
2. Escreva código para usar o modelo.
3. Implante o código como um componente personalizado no dispositivo principal.

Para ver um exemplo de componente de aplicativo cliente que mostra como realizar a detecção de anomalias em um pipeline personalizado do GStreamer, consulte https://github.com/awslabs/aws-greengrass-labs-lookoutvision-gstreamer.

Configuração de seu ambiente

Para escrever o código do cliente, seu ambiente de desenvolvimento se conecta remotamente a um dispositivo AWS IoT Greengrass Version 2 principal no qual você implantou um componente e dependências do modelo Amazon Lookout for Vision. Alternativamente, você pode escrever código em um dispositivo principal. Para obter mais informações, consulte as ferramentas de desenvolvimento do AWS IoT Greengrass e o desenvolvimento de componentes do AWS IoT Greengrass.

Seu código de cliente deve usar o cliente gRPC para acessar o Amazon Lookout for Vision Edge Agent. Esta seção mostra como configurar seu ambiente de desenvolvimento com o gRPC e instalar as dependências de terceiros necessárias para o `DetectAnomalies` código de exemplo.

Depois de terminar de escrever seu código de cliente, você cria um componente personalizado e implanta o componente personalizado em seus dispositivos de borda. Para obter mais informações, consulte Criando o componente do aplicativo cliente.

Tópicos

Configurar o gRPC

Em seu ambiente de desenvolvimento, você precisa de um cliente gRPC que use em seu código para chamar a API do Lookout for Vision Edge Agent. Para fazer isso, você cria um stub de gRPC usando um arquivo de definição de `.proto` serviço para o Lookout for Vision Edge Agent.

> **Note**
>
> Você também pode obter o arquivo de definição de serviço do pacote de aplicativos Lookout for Vision Edge Agent. O pacote de aplicativos é instalado quando o componente Lookout for Vision Edge Agent é instalado como uma dependência do componente do modelo. O pacote de aplicativos está localizado em`/greengrass/v2/packages/artifacts-unarchived/aws.iot.lookoutvision.EdgeAgent/`*`edge_agent_version`*`/lookoutvision_edge_agent`. Substitua o `edge_agent_version` pela versão do Lookout for Vision Edge Agent que você está usando. Para obter o pacote de aplicativos, você precisa implantar o Lookout for Vision Edge Agent em um dispositivo principal.

Para configurar o gRPC

1. Faça o download do arquivo zip, proto.zip. O arquivo zip contém o arquivo de definição de serviço .proto ()`edge-agent.proto`.
2. Descompacte o conteúdo.
3. Abra um prompt de comando e navegue até a pasta que contém`edge-agent.proto`.
4. Use os seguintes comandos para gerar as seguintes interfaces de cliente do Python.

```
%%bash
python3 -m pip install grpcio
python3 -m pip install grpcio-tools
python3 -m grpc_tools.protoc --proto_path=. --python_out=. --grpc_python_out=.
 edge-agent.proto
```

   Se os comandos forem bem-sucedidos, os stubs `edge_agent_pb2_grpc.py` e `edge_agent_pb2.py` serão criados no diretório de trabalho.

5. Escreva o código do cliente que usa seu modelo. Para obter mais informações, consulte Usando um modelo em seu componente de aplicativo cliente.

Adição de dependências de terceiros

O código de `DetectAnomalies` exemplo usa a biblioteca Pillow para trabalhar com imagens. Para obter mais informações, consulte Detectando anomalias usando bytes de imagem.

Use o seguinte comando para instalar a biblioteca Pillow.

```
python3 -m pip install Pillow
```

Usando um modelo em seu componente de aplicativo cliente

As etapas para usar um modelo de um componente de aplicativo cliente são semelhantes ao uso de um modelo hospedado na nuvem.

1. Comece a executar o modelo.
2. Detecte anomalias nas imagens.
3. Pare o modelo, se não for mais necessário.

O Amazon Lookout for Vision Edge Agent fornece API para iniciar um modelo, detectar anomalias em uma imagem e interromper um modelo. Você também pode usar a API para listar os modelos em um dispositivo e obter informações sobre um modelo implantado. Para obter mais informações, consulte Referência da API do Amazon Lookout for Vision.

Você pode obter informações de erro verificando os códigos de status do gRPC. Para obter mais informações, consulte Obter informações de erro.

Para escrever seu código, você pode usar qualquer linguagem compatível com o gRPC. Fornecemos exemplos de código Python.

Tópicos

- Usando o stub em seu componente de aplicativo cliente
- Iniciar o modelo
- Detecção de anomalias
- Parar o modelo

- Listando modelos em um dispositivo
- Descrever um modelo
- Obter informações de erro

Usando o stub em seu componente de aplicativo cliente

Use o código a seguir para configurar o acesso ao seu modelo por meio do Lookout for Vision Edge Agent.

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
 with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent resources
 leakage
```

Iniciar o modelo

Você inicia um modelo chamando a Modelo inicial API. O modelo pode demorar um pouco para começar. Você pode verificar o status atual ligando paraDescreva o modelo. O modelo está em execução se o valor do `status` campo estiver em execução.

Código de exemplo

Substitua *`component_name`* pelo nome do componente do seu modelo.

```
import time

import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"
```

```
def start_model_if_needed(stub, model_name):
    # Starting model if needed.
    while True:
        model_description_response =
 stub.DescribeModel(pb2.DescribeModelRequest(model_component=model_name))
        print(f"DescribeModel() returned {model_description_response}")
        if model_description_response.model_description.status == pb2.RUNNING:
            print("Model is already running.")
            break
        elif model_description_response.model_description.status == pb2.STOPPED:
            print("Starting the model.")
            stub.StartModel(pb2.StartModelRequest(model_component=model_name))
            continue
        elif model_description_response.model_description.status == pb2.FAILED:
            raise Exception(f"model {model_name} failed to start")
        print(f"Waiting for model to start.")
        if model_description_response.model_description.status != pb2.STARTING:
            break
        time.sleep(1.0)


# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    start_model_if_needed(stub, model_component_name)
```

Detecção de anomalias

Você usa a Detectar anomalias API para detectar anomalias em uma imagem.

A operação `DetectAnomalies` espera que o bitmap da imagem seja passado no formato compactado RGB888. O primeiro byte representa o canal vermelho, o segundo byte representa o canal verde e o terceiro byte representa o canal azul. Se você fornecer a imagem em um formato diferente, como BGR, as previsões de DetectAnomalies estão incorretas.

Por padrão, o OpenCV usa o formato BGR para bitmaps de imagem. Se você estiver usando o OpenCV para capturar imagens para análise, deverá converter `DetectAnomalies` a imagem para o formato RGB888 antes de passá-la para o. `DetectAnomalies`

As imagens que você fornece `DetectAnomalies` devem ter as mesmas dimensões de largura e altura das imagens que você usou para treinar o modelo.

Detectando anomalias usando bytes de imagem

Você pode detectar anomalias em uma imagem fornecendo a imagem como bytes de imagem. No exemplo a seguir, os bytes da imagem são recuperados de uma imagem armazenada no sistema de arquivos local.

Substitua *sample.jpg* pelo nome do arquivo de imagem que você deseja analisar. Substitua *component_name* pelo nome do componente do seu modelo.

```
import time

from PIL import Image
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"

....
# Detecting anomalies.
def detect_anomalies(stub, model_name, image_path):
    image = Image.open(image_path)
    image = image.convert("RGB")
    detect_anomalies_response = stub.DetectAnomalies(
        pb2.DetectAnomaliesRequest(
            model_component=model_name,
            bitmap=pb2.Bitmap(
                width=image.size[0],
                height=image.size[1],
                byte_data=bytes(image.tobytes())
            )
        )
    )
    print(f"Image is anomalous -
 {detect_anomalies_response.detect_anomaly_result.is_anomalous}")
    return detect_anomalies_response.detect_anomaly_result


# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
```

```
    start_model_if_needed(stub, model_component_name)
    detect_anomalies(stub, model_component_name, "sample.jpg")
```

Detectando anomalias usando o segmento de memória compartilhada

Você pode detectar anomalias em uma imagem fornecendo a imagem como bytes de imagem no segmento de memória compartilhada POSIX. Para proporcionar o melhor desempenho, recomendamos usar memória compartilhada para solicitações de DetectAnomalies. Para obter mais informações, consulte Detectar anomalias.

Parar o modelo

Se você não estiver mais usando o modelo, a Parar o modelo API para interromper a execução do modelo.

```
stop_model_response = stub.StopModel(
    pb2.StopModelRequest(
         model_component=model_component_name
    )
)
print(f"New status of the model is {stop_model_response.status}")
```

Listando modelos em um dispositivo

Você pode usar a the section called “ListModels” API para listar os modelos que são implantados em um dispositivo.

```

models_list_response = stub.ListModels(
    pb2.ListModelsRequest()
)
for model in models_list_response.models:
    print(f"Model Details {model}")
```

Descrever um modelo

Você pode obter informações sobre um modelo implantado em um dispositivo chamando a Descreva o modelo API. Usar o `DescribeModel` é útil para obter o status atual de um modelo. Por exemplo, você precisa saber se um modelo está em execução antes de poder ligar `DetectAnomalies`. Para ver um código demonstrativo, consulte Iniciar o modelo.

Obter informações de erro

Os códigos de status do gRPC são usados para relatar os resultados da API.

Você pode obter informações de erro detectando a `RpcError` exceção, conforme mostrado no exemplo a seguir. Para obter informações sobre os códigos de status de erro, consulte o tópico de referência de uma API.

```
# Error handling.
try:
    stub.DetectAnomalies(detect_anomalies_request)
except grpc.RpcError as e:
    print(f"Error code: {e.code()}, Status: {e.details()}")
```

Criando o componente do aplicativo cliente

Você pode criar o componente do aplicativo cliente depois de gerar os stubs do gRPC e ter o código do aplicativo cliente pronto. O componente que você cria é um componente personalizado com o qual você implanta em um dispositivo AWS IoT Greengrass Version 2 principal AWS IoT Greengrass V2. Uma receita que você cria descreve seu componente personalizado. A receita inclui todas as dependências que também precisam ser implantadas. Nesse caso, você especifica o componente de modelo que você cria em Empacotar seu modelo Amazon Lookout for Vision. Para obter mais informações sobre receitas de componentes, consulte Referência de receitas de componentes AWS IoT Greengrass Version 2.

Os procedimentos neste tópico mostram como criar o componente do aplicativo cliente a partir de um arquivo de receita e publicá-lo como um componente AWS IoT Greengrass V2 personalizado. Você pode usar o AWS IoT Greengrass V2 console ou o AWS SDK para publicar o componente.

Para obter informações detalhadas sobre a criação de um componente personalizado, consulte o seguinte na AWS IoT Greengrass V2documentação.

- Desenvolva e teste um componente em seu dispositivo
- Crie AWScomponentes do IoT Greengrass
- Publique componentes para implantar em seus dispositivos principais

Tópicos

- Permissões do IAM para publicar um componente do aplicativo cliente

- Criando a receita
- Publicando o componente do aplicativo cliente (Console)
- Publicação do componente do aplicativo cliente (SDK)

Permissões do IAM para publicar um componente do aplicativo cliente

Para criar e publicar seu componente de aplicativo cliente, você precisa das seguintes permissões do IAM:

- `greengrass:CreateComponentVersion`
- `greengrass:DescribeComponent`
- `s3:PutObject`

Criando a receita

Neste procedimento, você cria a receita para um componente simples do aplicativo cliente. O código em `lookoutvision_edge_agent_example.py` lista os modelos que são implantados no dispositivo e são executados automaticamente após a implantação do componente no dispositivo principal. Para visualizar a saída, verifique o registro do componente depois de implantar o componente. Para obter mais informações, consulte Implantando seus componentes em um dispositivo. Quando estiver pronto, use esse procedimento para criar a receita do código que implementa sua lógica de negócios.

Você cria a receita como um arquivo no formato JSON ou YAML. Você também faz upload do código do aplicativo do cliente para um bucket do Amazon S3.

Para criar a receita do componente do aplicativo cliente

1. Caso ainda não tenha feito isso, crie os arquivos stub do gRPC. Para obter mais informações, consulte Configurar o gRPC.
2. Salve o código a seguir em um arquivo chamado `lookoutvision_edge_agent_example.py`.

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
```

```
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent
 resources leakage

    models_list_response = stub.ListModels(
        pb2.ListModelsRequest()
    )
    for model in models_list_response.models:
        print(f"Model Details {model}")
```

3. Crie um bucket do Amazon S3 (ou use um bucket existente) para armazenar os arquivos de origem do seu componente de aplicativo cliente. O bucket deve estar na sua AWS conta e na mesma AWS região em que você usa AWS IoT Greengrass Version 2 o Amazon Lookout for Vision.
4. Faça upload de `lookoutvision_edge_agent_example.py`, `edge_agent_pb2_grpc.py` and `edge_agent_pb2.py` para o bucket do Amazon S3 que você criou na etapa anterior. Observe o caminho do Amazon S3 de cada arquivo. Você criou `edge_agent_pb2_grpc.py` e `edge_agent_pb2.py` em Configurar o gRPC.
5. Em um editor, crie o seguinte arquivo de receita JSON ou YAML.

   - `model_component` ao nome do componente do seu modelo. Para obter mais informações, consulte Configurações do componente.
   - Altere as entradas do URI para os caminhos S3 de `lookoutvision_edge_agent_example.py`, `edge_agent_pb2_grpc.py` e. `edge_agent_pb2.py`

   JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.lookoutvision.EdgeAgentPythonExample",
  "ComponentVersion": "1.0.0",
  "ComponentType": "aws.greengrass.generic",
  "ComponentDescription": "Lookout for Vision Edge Agent Sample Application",
  "ComponentPublisher": "Sample App Publisher",
  "ComponentDependencies": {
    "model_component": {
      "VersionRequirement": ">=1.0.0",
```

```
      "DependencyType": "HARD"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install grpcio grpcio-tools protobuf Pillow",
        "run": {
          "script": "python3 {artifacts:path}/
lookoutvision_edge_agent_example.py"
        }
      },
      "Artifacts": [
        {
          "Uri": "S3 path to lookoutvision_edge_agent_example.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2_grpc.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2.py"
        }
      ]
    }
  ],
  "Lifecycle": {}
}
```

YAML

```
---
RecipeFormatVersion: 2020-01-25
ComponentName: com.lookoutvison.EdgeAgentPythonExample
ComponentVersion: 1.0.0
ComponentDescription: Lookout for Vision Edge Agent Sample Application
ComponentPublisher: Sample App Publisher
ComponentDependencies:
  model_component:
    VersionRequirement: '>=1.0.0'
    DependencyType: HARD
```

```
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: |-
        pip3 install grpcio
        pip3 install grpcio-tools
        pip3 install protobuf
        pip3 install Pillow
      run:
        script: |-
          python3 {artifacts:path}/lookout_vision_agent_example.py
    Artifacts:
      - URI: S3 path to lookoutvision_edge_agent_example.py
      - URI: S3 path to edge_agent_pb2_grpc.py
      - URI: S3 path to edge_agent_pb2.py
```

6. Salve o arquivo JSON ou YAML em seu computador.
7. Crie o componente do aplicativo cliente seguindo um destes procedimentos:
   - Se você quiser usar o AWS IoT Greengrass console, usePublicando o componente do aplicativo cliente (Console).
   - Se você quiser usar o AWS SDK, use Publicação do componente do aplicativo cliente (SDK).

Publicando o componente do aplicativo cliente (Console)

Você pode usar o AWS IoT Greengrass V2 console para publicar o componente do aplicativo cliente.

Para publicar o componente do aplicativo cliente

1. Caso ainda não tenha feito isso, crie a receita para seu componente de aplicativo de cliente fazendo. Criando a receita
2. Abra o console AWS IoT Greengrass em https://console.aws.amazon.com/iot/
3. No painel de navegação à esquerda, em Greengrass, escolha Components.
4. Em Meus componentes, escolha Criar componente.
5. Na página Criar componente, escolha Inserir receita como JSON se quiser usar uma receita no formato JSON. Escolha Inserir receita como YAML se quiser usar uma receita no formato YAML.
6. Em Receita, substitua a receita existente pela receita JSON ou YAML que você criou em. Criando a receita

7. Escolha Criar componente.
8. Em seguida, implante seu componente de aplicativo cliente.

Publicação do componente do aplicativo cliente (SDK)

Você pode publicar o componente do aplicativo cliente usando a API createComponentVersion.

Para publicar o componente do aplicativo cliente (SDK)

1. Caso ainda não tenha feito isso, crie a receita para seu componente de aplicativo de cliente fazendo. Criando a receita
2. No prompt de comando, insira o comando a seguir para criar o componente do aplicativo cliente. Substitua o `recipe-file` pelo nome do arquivo de receita que você criou em Criando a receita.

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://recipe-file
   ```

   Observe o ARN do componente na resposta. Você precisa dele na próxima etapa.
3. Use o comando a seguir para obter o status do componente do aplicativo do cliente. Substitua o `component-arn` pelo ARN que você anotou na etapa anterior. O componente do aplicativo cliente está pronto se o valor de `componentState` for`DEPLOYABLE`.

   ```
   aws greengrassv2 describe-component --arn component-arn
   ```
4. Em seguida, implante seu componente de aplicativo cliente.

Implantando seus componentes em um dispositivo

Para implantar o componente modelo e o componente do aplicativo cliente em um dispositivo AWS IoT Greengrass Version 2 principal, você usa o AWS IoT Greengrass V2 console ou usa a API CreateDeployment. Para obter mais informações, consulte Criação de implantações no Guia do desenvolvedor do AWS IoT Greengrass Version 2. Para obter informações sobre a atualização de um componente implantado em um dispositivo principal, consulte Revisar implantações.

Tópicos

- Permissões do IAM para implantar componentes
- Implantando seus componentes (console)

- Implantação dos componentes (SDK)

Permissões do IAM para implantar componentes

Para implantar um componente com AWS IoT Greengrass V2 você precisa das seguintes permissões:

- `greengrass:ListComponents`
- `greengrass:ListComponentVersions`
- `greengrass:ListCoreDevices`
- `greengrass:CreateDeployment`
- `greengrass:GetDeployment`
- `greengrass:ListDeployments`

`CreateDeployment`e `GetDeployment` têm ações dependentes. Para obter mais informações, consulte Ações definidas pelo AWS IoT Greengrass V2.

Para obter informações sobre como alterar as permissões do IAM, consulte Alteração das permissões de um usuário.

Implantando seus componentes (console)

Use o procedimento a seguir para implantar o componente do aplicativo cliente em um dispositivo principal. O aplicativo cliente depende do componente do modelo (que, por sua vez, depende do Lookout for Vision Edge Agent). A implantação do componente do aplicativo cliente também inicia a implantação do componente de modelo e do Lookout for Vision Edge Agent.

> **Note**
>
> Você pode adicionar seus componentes a uma implantação existente. Você também pode implantar componentes em um grupo de coisas.

Para executar esse procedimento, você deve ter um dispositivo AWS IoT Greengrass V2 principal configurado. Para obter mais informações, consulte Configurando seu dispositivo AWS IoT Greengrass Version 2 principal.

Para implantar seus componentes em um dispositivo

1. Abra o console do AWS IoT Greengrass em https://console.aws.amazon.com/iot/.
2. No painel de navegação à esquerda, em Greengrass, escolha Implantações.
3. Em Implantações, escolha Criar.
4. Na página Especificar detalhes, faça o seguinte:
    1. Em Informações de implantação, insira ou modifique o nome amigável para sua implantação.
    2. Em Destino de implantação, selecione Dispositivo principal e insira um nome de destino.
    3. Escolha Próximo.
5. Na página Selecionar componentes, faça o seguinte:
    1. Em Meus componentes, escolha o nome do componente do aplicativo cliente (`com.lookoutvison.EdgeAgentPythonExample`).
    2. Escolha Próximo
6. Na página Configurar componentes, mantenha a configuração atual e escolha Próximo.
7. Na página Definir configurações avançadas, mantenha as configurações atuais e escolha Próximo.
8. Na página Revisar, escolha Implantar para começar a implantar seu componente.

Verificando o status da implantação (console)

Você pode verificar o status da implantação no AWS IoT Greengrass V2 console. Se o componente do aplicativo cliente usar a receita e o código de exemplo de the section called “Criando o componente do aplicativo cliente”, visualize o registro do componente do aplicativo cliente após a conclusão da implantação. Se for bem-sucedido, o registro incluirá uma lista dos modelos do Lookout for Vision que são implantados no componente.

Para obter informações sobre como usar o AWS SDK para verificar o status da implantação, consulte Verificar o status da implantação.

Para verificar o status da implantação

1. Abra o console AWS IoT Greengrass em https://console.aws.amazon.com/iot/
2. No painel de navegação à esquerda, escolha Dispositivos principais.
3. Em Dispositivos principais do Greengrass, escolha seu dispositivo principal.

4. Escolha a guia Implantações para ver o status atual da implantação.
5. Depois que as implantações forem bem-sucedidas (o status é Concluído), abra uma janela de terminal no dispositivo principal e visualize o registro do componente do aplicativo cliente em. `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log` Se sua implantação usar a receita e o código de exemplo, o log incluirá a saída `delookoutvision_edge_agent_example.py`. Por exemplo:

```
Model Details model_component:"ModelComponent"
```

Implantação dos componentes (SDK)

Use o procedimento a seguir para implantar o componente do aplicativo cliente, o componente do modelo e o Amazon Lookout for Vision Edge Agent em seu dispositivo principal.

1. Crie um `deployment.json` arquivo para definir a configuração de implantação dos seus componentes. Esse arquivo deve se parecer com o exemplo a seguir.

```
{
  "targetArn":"targetArn",
  "components": {
    "com.lookoutvison.EdgeAgentPythonExample": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
      }
    }
  }
}
```

   - No `targetArn` campo, *`targetArn`*substitua pelo nome de recurso da Amazon (ARN) da coisa ou grupo de itens a ser destinado para a implantação, no seguinte formato:
     - Coisa: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grupo de coisas: `arn:aws:iot:`*`region`*`:`*`account-id:`*`thinggroup/`*`thingGroupName`*
2. Verifique se o destino de implantação tem uma implantação existente que você deseja revisar. Faça o seguinte:

   a. Execute o comando a seguir para listar as implantações para o destino de implantação. `targetArn`Substitua pelo nome de recurso da Amazon (ARN) da coisa ou grupo de coisas

de AWS IoT de destino. Para obter os ARNs das coisas na região atual da AWS, use o comando`aws iot list-things`.

```
aws greengrassv2 list-deployments --target-arn targetArn
```

A resposta contém uma lista com a implantação mais recente do destino. Se a resposta estiver vazia, o destino não tem uma implantação existente e você pode pular para a Etapa 3. Caso contrário, copie o `deploymentId` da resposta para usar na próxima etapa.

b. Execute o comando a seguir para obter os detalhes da implantação. Esses detalhes incluem metadados, componentes e configuração do trabalho. Substitua o `deploymentId` pelo ID da tarefa da etapa anterior.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

c. Copie qualquer um dos seguintes pares de valores-chave da resposta do comando anterior para deployment.json. Você pode alterar esses valores para a nova implantação.

- `deploymentName`— O nome da implantação.
- `components`— Os componentes da implantação. Para desinstalar um componente, remova-o desse objeto.
- `deploymentPolicies`— As políticas de implantação.
- `tags`— As tags da implantação.

3. Execute o comando a seguir para implantar os componentes no dispositivo. Anote o valor do `deploymentId` na resposta.

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

4. Execute o seguinte comando para obter o status da implantação. Altere o `deployment-id` para o valor que você anotou na etapa anterior. A implantação foi concluída com êxito se o valor de `deploymentStatus` for. `COMPLETED`

```
aws greengrassv2 get-deployment --deployment-id deployment-id
```

5. Depois que as implantações forem bem-sucedidas, abra uma janela de terminal no dispositivo principal e visualize o registro do componente do aplicativo cliente em. `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log`

Se sua implantação usar a receita e o código de exemplo, o log incluirá a saída de `lookoutvision_edge_agent_example.py`. Por exemplo:

```
Model Details model_component:"ModelComponent"
```

Referência da API do Amazon Lookout for Vision

Esta seção é a referência da API para o Amazon Lookout for Vision Edge Agent.

Detectando anomalias com um modelo

Você usa a Detectar anomalias API para detectar anomalias em imagens usando um modelo em execução em um dispositivo AWS IoT Greengrass Version 2 principal.

Obter informações do modelo

APIs que obtêm informações sobre modelos implantados em um dispositivo principal.

- ListModels
- Descreva o modelo

Executando um modelo

APIs para iniciar e interromper um modelo Amazon Lookout for Vision implantado em um dispositivo principal.

- Modelo inicial
- Parar o modelo

Detectar anomalias

Detecta anomalias na imagem fornecida.

A resposta de `DetectAnomalies` inclui uma previsão booleana de que a imagem contém uma ou mais anomalias e um valor de confiança para a previsão. Se o modelo for um modelo de segmentação, a resposta incluirá o seguinte:

- Uma imagem de máscara que cobre cada tipo de anomalia em uma cor exclusiva. Você pode `DetectAnomalies` armazenar a imagem da máscara na memória compartilhada ou devolvê-la como bytes da imagem.
- A área percentual da imagem que um tipo de anomalia cobre.
- A cor hexadecimal de um tipo de anomalia na imagem da máscara.

> **Note**
>
> O modelo que você usa `DetectAnomalies` deve estar em execução. Você pode obter o status atual chamando Descreva o modelo. Para começar a executar um modelo, consulte Modelo inicial.

`DetectAnomalies` suporta bitmaps (imagens) compactados no formato RGB888 intercalado. O primeiro byte representa o canal vermelho, o segundo byte representa o canal verde e o terceiro byte representa o canal azul. Se você fornecer a imagem em um formato diferente, como BGR, as previsões de DetectAnomalies estão incorretas.

Por padrão, o OpenCV usa o formato BGR para bitmaps de imagem. Se você estiver usando o OpenCV para capturar imagens para análise, deverá converter `DetectAnomalies` a imagem para o formato RGB888 antes de passá-la para o. `DetectAnomalies`

A dimensão mínima da imagem suportada é 64x64 pixels. A dimensão máxima da imagem suportada é 4096x4096 pixels.

Você pode enviar a imagem na mensagem protobuf ou por meio de um segmento de memória compartilhada. Serializar imagens grandes na mensagem protobuf pode aumentar significativamente a latência das chamadas para. `DetectAnomalies` Para obter a menor latência, recomendamos que você use memória compartilhada.

```
rpc DetectAnomalies(DetectAnomaliesRequest) returns (DetectAnomaliesResponse);
```

Solicitação de Detectar anomalias

Os parâmetros de entrada para `DetectAnomalies`.

```
message Bitmap {
  int32 width = 1;
  int32 height = 2;
  oneof data {
    bytes byte_data = 3;
    SharedMemoryHandle shared_memory_handle = 4;
  }
}
```

```
message SharedMemoryHandle {
  string name = 1;
  uint64 size = 2;
  uint64 offset = 3;
}
```

```
message AnomalyMaskParams {
SharedMemoryHandle shared_memory_handle = 2;
}
```

```
message DetectAnomaliesRequest {
string model_component = 1;
Bitmap bitmap = 2;
AnomalyMaskParams anomaly_mask_params = 3;
}
```

Bitmap

A imagem com a qual você deseja analisar `DetectAnomalies`.

width

A largura da imagem em pixels.

height

A altura da imagem em pixels.

byte_data

Bytes de imagem passados na mensagem protobuf.

identificador_de_memória compartilhada

Bytes de imagem passados no segmento de memória compartilhada.

Alça de memória compartilhada

Representa um segmento de memória compartilhada POSIX.

name

O nome do segmento de memória POSIX. Para obter informações sobre como criar memória compartilhada, consulte shm_open.

tamanho

O tamanho do buffer da imagem em bytes a partir do deslocamento.

deslocamento

O deslocamento, em bytes, do início do buffer de imagem a partir do início do segmento de memória compartilhada.

Parâmetros da máscara de anomalia

Parâmetros para gerar uma máscara de anomalia. (Modelo de segmentação).

identificador_de_memória compartilhada

Contém os bytes da imagem da máscara, caso `shared_memory_handle` não tenha sido fornecida.

Solicitação de Detectar anomalias

componente_modelo

O nome do AWS IoT Greengrass V2 componente que contém o modelo que você deseja usar.

Bitmap

A imagem com a qual você deseja analisar`DetectAnomalies`.

parâmetros de máscara de anomalia

Parâmetros opcionais para a saída da máscara. (Modelo de segmentação).

Detecte uma resposta de anomalias

A resposta do `DetectAnomalies`

```
message DetectAnomalyResult {
bool is_anomalous = 1;
float confidence = 2;
Bitmap anomaly_mask = 3;
repeated Anomaly anomalies = 4;
float anomaly_score = 5;
float anomaly_threshold = 6;
}
```

```
message Anomaly {
string name = 1;
PixelAnomaly pixel_anomaly = 2;
```

```
message PixelAnomaly {
float total_percentage_area = 1;
string hex_color = 2;
}
```

```
message DetectAnomaliesResponse {
  DetectAnomalyResult detect_anomaly_result = 1;
}
```

Anomalia

Representa uma anomalia encontrada em uma imagem. (Modelo de segmentação).

name

O nome de um tipo de anomalia encontrado em uma imagem. `name`mapeia para um tipo de anomalia no conjunto de dados de treinamento. O serviço insere automaticamente o tipo de anomalia de fundo na resposta de DetectAnomalies.

pixel_anomalia

Informações sobre a máscara de pixels que cobre um tipo de anomalia.

Anomalia de pixels

Informações sobre a máscara de pixels que cobre um tipo de anomalia. (Modelo de segmentação).

área_percentual total

A área percentual da imagem que o tipo de anomalia cobre.

cor hexadecimal

Um valor de cor hexadecimal que representa o tipo de anomalia na imagem. A cor é mapeada para a cor do tipo de anomalia usado no conjunto de dados de treinamento.

Detectar um resultado de anomalia

é_anômalo

Indica se a imagem contém uma anomalia. `true`se a imagem contiver uma anomalia. `false`se a imagem estiver normal.

confidence

A confiança que `DetectAnomalies` se tem na precisão da previsão. `confidence`é um valor de ponto flutuante entre 0 e 1.

máscara de anomalia

se shared_memory_handle não tiver sido fornecido, contém os bytes da imagem da máscara. (Modelo de segmentação).

anomalias

Uma lista de 0 ou mais anomalias encontradas na imagem de entrada. (Modelo de segmentação).

pontuação_anomalia

Um número que quantifica quantas anomalias previstas para uma imagem se desviam de uma imagem sem anomalias. `anomaly_score` é um valor flutuante que varia de `0.0` a (menor desvio de uma imagem normal) a 1,0 (maior desvio de uma imagem normal). O Amazon Lookout for Vision retorna um valor `anomaly_score` para, mesmo que a previsão de uma imagem seja normal.

limite_de_anomalia

Um número (flutuante) que determina quando a classificação prevista para uma imagem é normal ou anômala. Imagens com um `anomaly_score` valor igual ou superior ao valor de `anomaly_threshold` são consideradas anômalas. Um `anomaly_score` valor abaixo

`anomaly_threshold` indica uma imagem normal. O valor `anomaly_threshold` que um modelo usa é calculado pelo Amazon Lookout for Vision quando você treina o modelo. Não é possível definir ou alterar o valor de `anomaly_threshold`

Códigos de status

Código	Número	Descrição
OK	0	`DetectAnomalies` fez uma previsão com sucesso
UNKNOWN	2	Ocorreu um erro desconhecido.
ARGUMENTO_INVÁLIDO	3	Um ou mais parâmetros de entrada são inválidos. Check the the the error for more details.
NÃO_ENCONTRADO	5	Um modelo com o nome especificado não foi encontrado.
RECURSO_ESGOTADO	8	Não há recursos suficientes para realizar essa operação. Por exemplo, o Lookout for Vision Edge Agent não consegue acompanhar o ritmo das chamadas para `DetectAnomalies` Check the the the error for more details.
CONDIÇÃO_PRÉVIA FALHADA	9	`DetectAnomalies` foi chamado para um modelo que não está no estado RUNNING.
INTERNO	13	Ocorreu um erro interno.

Descreva o modelo

Descreve um modelo do Amazon Lookout for Vision que é implantado em AWS IoT Greengrass Version 2 um dispositivo principal.

```
rpc DescribeModel(DescribeModelRequest) returns (DescribeModelResponse);
```

Descreva a solicitação do modelo

```
message DescribeModelRequest {
  string model_component = 1;
}
```

componente_modelo

O nome do AWS IoT Greengrass V2 componente que contém o modelo que você deseja descrever.

Descreva a resposta do modelo

```
message ModelDescription {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message DescribeModelResponse {
  ModelDescription model_description = 1;
}
```

Descrição do modelo

componente_modelo

O nome do AWS IoT Greengrass Version 2 componente que contém o modelo Amazon Lookout for Vision.

lookout_vision_model_arn

O nome de recurso da Amazon ARN do modelo Amazon Lookout for Vision que foi usado para gerar o componente. AWS IoT Greengrass V2

status

O status atual do modelo. Para obter mais informações, consulte Status do modelo.

mensagem_status

A mensagem de status do modelo.

Códigos de status

Código	Número	Descrição
OK	0	A chamada foi bem-sucedida.
UNKNOWN	2	Ocorreu um erro desconhecido.
ARGUMENTO_INVÁLIDO	3	Um ou mais parâmetros de entrada são inválidos. Check the the the error for more details.
NÃO_ENCONTRADO	5	Não foi encontrado um modelo com o nome fornecido.
INTERNO	13	Ocorreu um erro interno.

ListModels

Lista os modelos implantados em um dispositivo AWS IoT Greengrass Version 2 principal.

```
rpc ListModels(ListModelsRequest) returns (ListModelsResponse);
```

Solicitação de modelos de lista

```
message ListModelsRequest {}
```

Resposta de modelos de lista

```
message ModelMetadata {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message ListModelsResponse {
  repeated ModelMetadata models = 1;
}
```

Metadados do modelo

componente_modelo

O nome do AWS IoT Greengrass Version 2 componente que contém um modelo Amazon Lookout for Vision.

lookout_vision_model_arn

O nome do recurso da Amazon (ARN) do modelo do Amazon Lookout for Vision usado para gerar o componente. AWS IoT Greengrass V2

status

O status atual do modelo. Para obter mais informações, consulte Status do modelo.

mensagem_status

A mensagem de status do modelo.

Códigos de status

Código	Número	Descrição
OK	0	A chamada foi bem-sucedida.

Código	Número	Descrição
UNKNOWN	2	Ocorreu um erro desconhecido.
INTERNO	13	Ocorreu um erro interno.

Modelo inicial

Inicia um modelo em execução em um dispositivo AWS IoT Greengrass Version 2 principal. Pode demorar um pouco para que o modelo comece a funcionar. Para verificar o status atual, chame Descreva o modelo. O modelo está em execução se o `Status` campo estiver `RUNNING`.

O número de modelos que você pode executar simultaneamente depende da especificação de hardware do seu dispositivo principal.

```
rpc StartModel(StartModelRequest) returns (StartModelResponse);
```

Iniciar solicitação de modelo

```
message StartModelRequest {
  string model_component = 1;
}
```

componente_modelo

O nome do AWS IoT Greengrass Version 2 componente que contém o modelo que você deseja iniciar.

Iniciar resposta do modelo

```
message StartModelResponse {
  ModelStatus status = 1;
}
```

status

O status atual do modelo. A resposta é `STARTING` se a chamada for bem-sucedida. Para obter mais informações, consulte Status do modelo.

Códigos de status

Código	Número	Descrição
OK	0	O modelo está começando
UNKNOWN	2	Ocorreu um erro desconhec ido.
ARGUMENTO_INVÁLIDO	3	Um ou mais parâmetros de entrada são inválidos. Check the the the error for more details.
NÃO_ENCONTRADO	5	Não foi encontrado um modelo com o nome fornecido.
RECURSO_ESGOTADO	8	Não há recursos suficientes para realizar essa operação. Por exemplo, não há memória suficiente para carregar o modelo. Check the the the error for more details.
CONDIÇÃO_PRÉVIA FALHADA	9	O método foi chamado para um modelo que não está no estado PARADO ou FALHOU.
INTERNO	13	Ocorreu um erro interno.

Parar o modelo

Interrompe a execução de um modelo em um dispositivo AWS IoT Greengrass Version 2 principal. `StopModel` retorna após o modelo ter parado. O modelo foi interrompido com êxito se o campo `Status` na resposta for `STOPPED`.

```
  rpc StopModel(StopModelRequest) returns (StopModelResponse);
```

StopModelRequest

```
message StopModelRequest {
  string model_component = 1;
}
```

model_component

O nome do AWS IoT Greengrass Version 2 componente que contém o modelo que você deseja interromper.

Interromper a resposta do modelo

```
message StopModelResponse {
  ModelStatus status = 1;
}
```

status

O status atual do modelo. A resposta é `STOPPED` se a chamada for bem-sucedida. Para obter mais informações, consulte Status do modelo.

Códigos de status

Código	Número	Descrição
OK	0	O modelo está parando.
UNKNOWN	2	Ocorreu um erro desconhec ido.
ARGUMENTO_INVÁLIDO	3	Um ou mais parâmetros de entrada são inválidos. Check the the the error for more details.

Código	Número	Descrição
NÃO_ENCONTRADO	5	Não foi encontrado um modelo com o nome fornecido.
CONDIÇÃO_PRÉVIA FALHADA	9	O método foi chamado para um modelo que não está no estado RUNNING.
INTERNO	13	Ocorreu um erro interno.

Status do modelo

O status de um modelo implantado em um dispositivo AWS IoT Greengrass Version 2 principal. Para verificar o status atual, chame Descreva o modelo.

```
enum ModelStatus {
  STOPPED = 0;
  STARTING = 1;
  RUNNING = 2;
  FAILED = 3;
  STOPPING = 4;
}
```

Usando o painel do Amazon Lookout for Vision

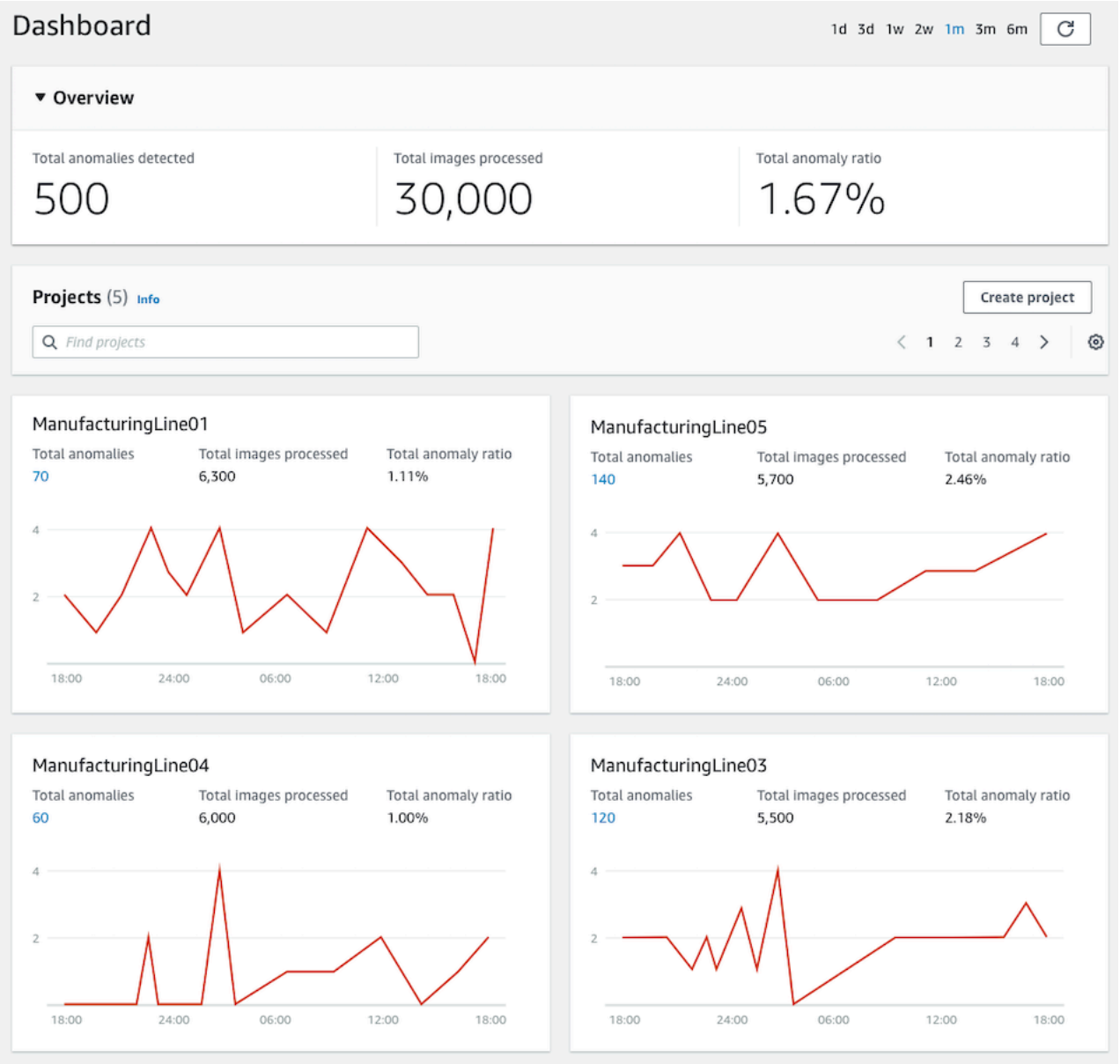
O painel fornece uma visão geral das métricas para seus projetos do Amazon Lookout for Vision, como o número total de anomalias detectadas na última semana. Com o painel, você tem uma visão geral de todos os seus projetos e uma visão geral de cada projeto individual. Você pode escolher a linha do tempo na qual as métricas são mostradas. Também é possível usar o painel para criar um novo projeto.

A seção Visão geral mostra o número total de projetos, o número total de imagens e o número total de imagens detectadas por todos os seus projetos.

A seção Projetos mostra as seguintes informações gerais para projetos individuais:

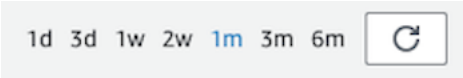
- O número total de anomalias detectadas.
- O número total de imagens processadas.
- A taxa total de anomalias (ou seja, a porcentagem de imagens detectadas com uma anomalia).
- Um gráfico mostra as detecções de anomalias no período de tempo escolhido.

Você também pode obter mais informações sobre um projeto.

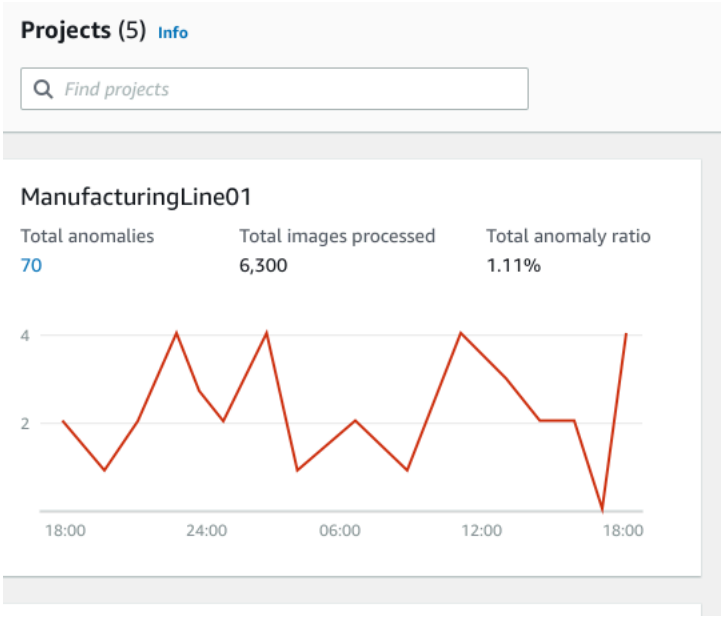


Para usar seu painel

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Como começar.
3. No painel de navegação à esquerda, escolha Painel.
4. Para obter as métricas em um período de tempo específico, faça o seguinte:
   a. Escolha o período de tempo no canto superior direito do painel.
   b. Escolha o botão de atualização para mostrar o painel com o novo cronograma.



5. Para obter mais detalhes sobre um projeto, escolha o nome do projeto na seção Projetos (por exemplo,`ManufacturingLine01`).



6. Para criar um projeto, escolha Criar projeto na seção Projetos.

Gerenciamento de recursos do Amazon Lookout for Vision

Você pode gerenciar seus recursos do Amazon Lookout for Vision usando o console ou o SDK AWS. O Amazon Lookout for Vision tem os seguintes recursos:

- Projetos
- Conjuntos de dados
- Modelos
- Detecções de testes

> **Note**
>
> Não é possível excluir uma tarefa de detecção de testes. Além disso, você não pode gerenciar detecções de teste usando o SDK AWS.

Tópicos

Visualizando seus projetos

Você pode obter uma lista de projetos do Amazon Lookout for Vision e informações sobre projetos individuais no console ou usando AWS o SDK.

> **Note**
>
> Eventualmente, a lista de projetos é consistente. Se você criar ou excluir um projeto, talvez precise esperar um pouco antes que a lista de projetos esteja atualizada.

Visualizar seus projetos (console)

Execute as etapas no procedimento a seguir para visualizar seus projetos no console do.

Para ver seus projetos

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos. A visualização Projetos é exibida.
4. Escolha um nome de projeto para ver os detalhes do projeto.

Visualizando seus projetos (SDK)

Um projeto gerencia os conjuntos de dados e modelos para um único caso de uso. Por exemplo, detectar anomalias em peças de máquinas. O exemplo a seguir chama `ListProjects` para obter uma lista de seus projetos.

Para visualizar seus projetos (SDK)

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

   CLI

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

   ```
   aws lookoutvision list-projects \
     --profile lookoutvision-access
   ```

   Use o comando `describe-project` para obter informações sobre uma frota.

Altere o valor de `project-name` para o ARN do projeto que você deseja descrever.

```
aws lookoutvision describe-project --project-name project_name \
  --profile lookoutvision-access
```

Python

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
    @staticmethod
    def list_projects(lookoutvision_client):
        """
        Lists information about the projects that are in in your AWS account
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("\tARN: " + project["ProjectArn"])
                print("\tCreated: " + str(["CreationTimestamp"]))
                print("Datasets")
                project_description = lookoutvision_client.describe_project(
                    ProjectName=project["ProjectName"]
                )
                if not project_description["ProjectDescription"]["Datasets"]:
                    print("\tNo datasets")
                else:
                    for dataset in project_description["ProjectDescription"][
                        "Datasets"
                    ]:
                        print(f"\ttype: {dataset['DatasetType']}")
                        print(f"\tStatus: {dataset['StatusMessage']}")

                print("Models")
                response_models = lookoutvision_client.list_models(
                    ProjectName=project["ProjectName"]
                )
                if not response_models["Models"]:
```

```
                    print("\tNo models")
                else:
                    for model in response_models["Models"]:
                        Models.describe_model(
                            lookoutvision_client,
                            project["ProjectName"],
                            model["ModelVersion"],
                        )


 print("-----------------------------------------------------------\n")
            print("Done!")
        except ClientError:
            logger.exception("Problem listing projects.")
            raise
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
/**
 * Lists the Amazon Lookout for Vision projects in the current AWS account and
 AWS
 * Region.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return List<ProjectMetadata> Metadata for each project.
 */
public static List<ProjectMetadata> listProjects(LookoutVisionClient lfvClient)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Getting projects:");
        ListProjectsRequest listProjectsRequest = ListProjectsRequest.builder()
                        .maxResults(100)
                        .build();

        List<ProjectMetadata> projectMetadata = new ArrayList<>();
```

```
        ListProjectsIterable projects =
 lfvClient.listProjectsPaginator(listProjectsRequest);

        projects.stream().flatMap(r -> r.projects().stream())
                        .forEach(project -> {
                                projectMetadata.add(project);
                                logger.log(Level.INFO, project.projectName());
                        });

        logger.log(Level.INFO, "Finished getting projects.");

        return projectMetadata;

}
```

Excluir um projeto

É possível excluir um projeto da página de exibição de projetos no console ou usando a operação `DeleteProject`.

As imagens referenciadas pelos conjuntos de dados de um projeto não são excluídas.

Excluir um projeto (console)

Use o procedimento a seguir para excluir um . Se você usar o procedimento do console, as versões do modelo e os conjuntos de dados associados serão excluídos para você.

Para excluir um projeto

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos.
4. Na página Projetos, selecione o projeto que você deseja excluir.
5. Escolha Excluir, no alto da página.
6. Na caixa de diálogo Excluir, digite excluir para confirmar que deseja excluir o projeto.
7. Se necessário, opte por excluir todos os conjuntos de dados e modelos associados.
8. Escolha Excluir projeto.

Como excluir um projeto (SDK)

Você exclui um projeto do Amazon Lookout for Vision chamando DeleteProject e fornecendo o nome do projeto que você deseja excluir.

Antes de excluir um projeto, você deve primeiro excluir todos os modelos do projeto. Para obter mais informações, consulte Exclusão de um modelo (SDK). Você também precisa excluir os conjuntos de dados associados ao modelo. Para obter mais informações, consulte Excluir um conjunto de dados.

Poderá levar alguns instantes para que o projeto seja excluído. Durante esse período, o status do projeto é `DELETING`. O projeto será excluído se uma chamada subsequente para `DeleteProject` não incluir o projeto que você excluiu.

Para excluir um projeto (SDK)

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

   AWS CLI

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

   ```
   aws lookoutvision delete-project --project-name project_name \
     --profile lookoutvision-access
   ```

   Python

   Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

   ```
       @staticmethod
       def delete_project(lookoutvision_client, project_name):
           """
           Deletes a Lookout for Vision Model

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param project_name: The name of the project that you want to delete.
           """
           try:
   ```

```
            logger.info("Deleting project: %s", project_name)
            response =
 lookoutvision_client.delete_project(ProjectName=project_name)
            logger.info("Deleted project ARN: %s ", response["ProjectArn"])
        except ClientError as err:
            logger.exception("Couldn't delete project %s.", project_name)
            raise
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
/**
 * Deletes an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return String The ARN of the deleted project.
 */
public static String deleteProject(LookoutVisionClient lfvClient, String
 projectName)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Deleting project: {0}", projectName);

        DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder()
                        .projectName(projectName)
                        .build();

        DeleteProjectResponse response =
 lfvClient.deleteProject(deleteProjectRequest);

        logger.log(Level.INFO, "Deleted project: {0} ARN: {1}",
                        new Object[] { projectName, response.projectArn() });

        return response.projectArn();

}
```

Visualizar seus conjuntos de dados

Um projeto pode ter um único conjunto de dados usado para treinar e testar seu modelo. Como alternativa, você pode ter conjuntos de dados de treinamento e teste separados. É possível usar o console do para visualizar seus testes de validação. Você também pode usar a operação `DescribeDataset` para obter informações sobre um conjunto de dados (treinamento ou teste).

Visualização dos conjuntos de dados em um projeto (console)

Realize as etapas do procedimento a seguir para exibir os conjuntos de dados do projeto no console do.

Para visualizar seus conjuntos de dados (console)

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos.
4. Na página Projetos, selecione o projeto que contém os conjuntos de dados que você deseja visualizar.
5. No painel de navegação à esquerda, escolha Conjunto de dados para visualizar os detalhes do conjunto de dados. Se você tiver um conjunto de dados de treinamento e de teste, uma guia para cada conjunto de dados será exibida.

Visualização dos conjuntos de dados em um projeto (SDK)

É possível usar a operação `DescribeDataset` para obter informações sobre o conjunto de dados de treinamento ou teste associado a um projeto.

Para visualizar seus conjuntos de dados (SDK)

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

   CLI

   Altere os seguintes valores:

- `project-name` para o nome do projeto que contém o modelo que você deseja visualizar.
- `dataset-type` para o tipo de conjunto de dados que você deseja visualizar (`train` ou `test`).

```
aws lookoutvision describe-dataset --project-name project name\
  --dataset-type train or test \
  --profile lookoutvision-access
```

Python

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
    @staticmethod
    def describe_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Gets information about a Lookout for Vision dataset.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset
 that
                             you want to describe.
        :param dataset_type: The type (train or test) of the dataset that you
 want
                             to describe.
        """
        try:
            response = lookoutvision_client.describe_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            print(f"Name: {response['DatasetDescription']['ProjectName']}")
            print(f"Type: {response['DatasetDescription']['DatasetType']}")
            print(f"Status: {response['DatasetDescription']['Status']}")
            print(f"Message: {response['DatasetDescription']['StatusMessage']}")
            print(f"Images: {response['DatasetDescription']['ImageStats']
['Total']}")
            print(f"Labeled: {response['DatasetDescription']['ImageStats']
['Labeled']}")
            print(f"Normal: {response['DatasetDescription']['ImageStats']
['Normal']}")
```

```
            print(f"Anomaly: {response['DatasetDescription']['ImageStats']
['Anomaly']}")
        except ClientError:
            logger.exception("Service error: problem listing datasets.")
            raise
        print("Done.")
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
/**
 * Gets the description for a Amazon Lookout for Vision dataset.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to describe a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to describe (train
 *                    or test).
 * @return DatasetDescription A description of the dataset.
 */
public static DatasetDescription describeDataset(LookoutVisionClient lfvClient,
                String projectName,
                String datasetType) throws LookoutVisionException {

        logger.log(Level.INFO, "Describing {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .build();

        DescribeDatasetResponse describeDatasetResponse =
 lfvClient.describeDataset(describeDatasetRequest);
        DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();

        logger.log(Level.INFO, "Project: {0}\n"
                        + "Created: {1}\n"
```

```
                        + "Type: {2}\n"
                        + "Total: {3}\n"
                        + "Labeled: {4}\n"
                        + "Normal: {5}\n"
                        + "Anomalous: {6}\n",
                        new Object[] {
                                        datasetDescription.projectName(),
                                        datasetDescription.creationTimestamp(),
                                        datasetDescription.datasetType(),

 datasetDescription.imageStats().total().toString(),

 datasetDescription.imageStats().labeled().toString(),

 datasetDescription.imageStats().normal().toString(),

 datasetDescription.imageStats().anomaly().toString(),
                        });

        return datasetDescription;

}
```

Adicionar imagens ao seu conjunto de dados

Depois de criar um conjunto de dados, você pode adicionar mais imagens a ele. Por exemplo, se a avaliação do modelo indicar um modelo ruim, você poderá aprimorar a qualidade do seu modelo adicionando mais imagens. Se você criou um conjunto de dados de teste, adicionar mais imagens pode aumentar a precisão das métricas de desempenho do seu modelo.

Treine novamente seu modelo depois de atualizar seus conjuntos de dados.

Tópicos

- Adicionando mais imagens
- Adicionar mais imagens (SDK)

Adicionando mais imagens

Você pode adicionar mais imagens aos seus conjuntos de dados fazendo o upload de imagens do seu computador local. Para adicionar mais imagens rotuladas com o SDK, use a operação UpdateDatasetEntries.

Para adicionar mais imagens ao seu conjunto de dados (console)

1. Escolha Ações e selecione o conjunto de dados ao qual você deseja adicionar imagens.
2. Escolha as imagens que você deseja enviar para o conjunto de dados. Você pode arrastar as imagens ou escolher as imagens que deseja carregar do seu computador local. Você pode fazer upload de até 30 imagens por vez.
3. Escolha Fazer upload das imagens.
4. Escolha Salvar alterações.

Quando terminar de adicionar mais imagens, você precisa rotulá-las para que possam ser usadas para treinar o modelo. Para obter mais informações, consulte Classificação de imagens (console).

Adicionar mais imagens (SDK)

Para adicionar mais imagens rotuladas com o SDK, use a operação UpdateDatasetEntries. Você fornece um arquivo de manifesto que contém as imagens que você deseja adicionar. Você também pode atualizar as imagens existentes especificando a imagem no `source-ref` campo da linha JSON no arquivo de manifesto. Para obter mais informações, consulte Criar um arquivo de manifesto.

Para adicionar mais imagens a um conjunto de dados (SDK)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Use o código de exemplo a seguir para adicionar mais imagens a um conjunto de dados.

   CLI

   Altere os seguintes valores:

   - `project-name` ao nome do projeto que contém o conjunto de dados a ser atualizado.

- `dataset-type` para o tipo de conjunto de dados que você deseja atualizar (`train` ou `test`).
- `changes` para o local do arquivo de manifesto que contém atualizações do conjunto de dados.

```
aws lookoutvision update-dataset-entries\
    --project-name project\
    --dataset-type train or test\
    --changes fileb://manifest file \
    --profile lookoutvision-access
```

Python

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
    @staticmethod
    def update_dataset_entries(lookoutvision_client, project_name, dataset_type,
 updates_file):
        """
        Adds dataset entries to an Amazon Lookout for Vision dataset.
        :param lookoutvision_client: The Amazon Rekognition Custom Labels Boto3
 client.
        :param project_name: The project that contains the dataset that you want
 to update.
        :param dataset_type: The type of the dataset that you want to update
 (train or test).
        :param updates_file: The manifest file of JSON Lines that contains the
 updates.
        """

        try:
            status = ""
            status_message = ""
            manifest_file = ""

            # Update dataset entries
            logger.info(f"""Updating {dataset_type} dataset for project
 {project_name}
with entries from {updates_file}.""")
```

```
            with open(updates_file) as f:
                manifest_file = f.read()

            lookoutvision_client.update_dataset_entries(
                ProjectName=project_name,
                DatasetType=dataset_type,
                Changes=manifest_file,
            )

            finished = False
            while finished == False:

                dataset =
lookoutvision_client.describe_dataset(ProjectName=project_name,

DatasetType=dataset_type)

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

                if status == "UPDATE_IN_PROGRESS":
                    logger.info(
                        (f"Updating {dataset_type} dataset for project
{project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_IN_PROGRESS":
                    logger.info(
                        (f"Update failed, rolling back {dataset_type} dataset
for project {project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_COMPLETE":
                    logger.info(
                        f"Dataset updated: {status} : {status_message} :
{dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_COMPLETE":
                    logger.info(
```

```
                        f"Rollback complated after update failure: {status} :
 {status_message} : {dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                logger.exception(
                    f"Failed. Unexpected state for dataset update: {status} :
 {status_message} : {dataset_type} dataset for project {project_name}.")
                raise Exception(
                    f"Failed. Unexpected state for dataset update: {status} :
 {status_message} :{dataset_type} dataset for project {project_name}.")

            logger.info(f"Added entries to dataset.")

            return status, status_message

        except ClientError as err:
            logger.exception(
                f"Couldn't update dataset: {err.response['Error']['Message']}")
            raise
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
/**
 * Updates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision updates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to update a
 *                     dataset.
 * @param datasetType  The type of the dataset that you want to update (train or
 *                     test).
 * @param manifestFile The name and location of a local manifest file that you
 want to
 * use to update the dataset.
 * @return DatasetStatus The status of the updated dataset.
 */

public static DatasetStatus updateDatasetEntries(LookoutVisionClient lfvClient,
 String projectName,
```

```
                String datasetType, String updateFile) throws
FileNotFoundException, LookoutVisionException,
                InterruptedException {

        logger.log(Level.INFO, "Updating {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        InputStream sourceStream = new FileInputStream(updateFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
UpdateDatasetEntriesRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .changes(sourceBytes)
                        .build();

        lfvClient.updateDatasetEntries(updateDatasetEntriesRequest);

        boolean finished = false;
        DatasetStatus status = null;

        // Wait until update completes.

        do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .build();
                DescribeDatasetResponse describeDatasetResponse = lfvClient
                                .describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                status = datasetDescription.status();

                switch (status) {

                        case UPDATE_COMPLETE:
                                logger.log(Level.INFO, "{0} Dataset updated for
project {1}.",
```

```
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;

                        case UPDATE_IN_PROGRESS:

                                logger.log(Level.INFO, "{0} Dataset update for
project {1} in progress.",
                                                new Object[] { datasetType,
projectName });
                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_IN_PROGRESS:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
project {1}. Rolling back",
                                                new Object[] { datasetType,
projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_COMPLETE:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
project {1}. Rollback completed.",
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
project {1}. Unexpected error returned.",
                                                new Object[] { datasetType,
projectName });
                                finished = true;

                }
```

```
        } while (!finished);

        return status;

}
```

3. Repita a etapa anterior e forneça valores para o outro tipo de conjunto de dados.

Removendo imagens do seu conjunto de dados

Você não pode excluir imagens diretamente de um conjunto de dados. Em vez disso, você deve excluir o conjunto de dados existente e criar um novo conjunto de dados sem as imagens que deseja remover. A forma como você remove imagens depende de como você as importou para o conjunto de dados existente (arquivo de manifesto, bucket do Amazon S3 ou computador local).

Você também pode usar o AWS SDK para remover imagens. Isso é útil ao criar um modelo de segmentação de imagem sem um arquivo de manifesto de segmentação de imagem, tornando desnecessário redesenhar as máscaras de imagem com o console Amazon Lookout for Vision.

Tópicos

- Removendo imagens de um conjunto de dados (console)
- Removendo imagens de um conjunto de dados (SDK)

Removendo imagens de um conjunto de dados (console)

Use o procedimento a seguir para remover imagens de um conjunto de dados com o console do Amazon Lookout for Vision.

Para remover imagens de um conjunto de dados (console)

1. Abra a galeria de conjuntos de dados do projeto.
2. Anote o nome de cada imagem que você deseja remover.
3. Exclua o conjunto de dados existente.
4. Faça um dos seguintes procedimentos:

- Se você criou o conjunto de dados com um arquivo de manifesto, faça:

  a. Em um editor de texto, abra o arquivo de manifesto que você usou para criar o conjunto de dados.

  b. Remova a linha JSON de cada imagem que você anotou na etapa 2. Você pode identificar a linha JSON de uma imagem verificando o `source-ref` campo.

  c. é o arquivo manifesto.

  d. Crie um novo conjunto de dados com o arquivo de manifesto atualizado.

- Se você criou o conjunto de dados a partir de imagens importadas de um bucket do Amazon S3, faça o seguinte:

  a. Exclua as imagens que você anotou na etapa 2 do bucket do Amazon S3.

  b. Crie um novo conjunto de dados com as imagens restantes no bucket do Amazon S3. Se você classificar as imagens pelo nome da pasta, não precisará classificar as imagens na próxima etapa.

  c. Faça um dos seguintes procedimentos:

     - Se você estiver criando um modelo de classificação de imagens, classifique cada imagem sem rótulo.
     - Se você estiver criando um modelo de segmentação de imagem, classifique e segmente cada imagem sem rótulo.

- Se você criou o conjunto de dados a partir de imagens importadas de um computador local, faça:

  a. No seu computador, crie uma pasta com as imagens que você deseja usar. Não inclua as imagens que você deseja remover do conjunto de dados. Para obter mais informações, consulte Criação de um conjunto de dados usando imagens armazenadas em seu computador local.

  b. Crie o conjunto de dados com as imagens na pasta que você criou na etapa 4.a.

  c. Faça um dos seguintes procedimentos:

     - Se você estiver criando um modelo de classificação de imagens, classifique cada imagem sem rótulo.
     - Se você estiver criando um modelo de segmentação de imagem, classifique e segmente cada imagem sem rótulo.

5. Treinar o modelo

Removendo imagens de um conjunto de dados (SDK)

Você pode usar o método AWS para remover chaves de um conjunto de dados:

Para remover imagens de um conjunto de dados (SDK)

1. Abra a galeria de conjuntos de dados do projeto.
2. Anote o nome de cada imagem que você deseja remover.
3. Exporte as linhas JSON para o conjunto de dados usando a operação ListDatasetEntries.
4. Crie um arquivo de manifesto com as linhas JSON exportadas.
5. Em um editor de texto, abra o arquivo .
6. Remova a linha JSON de cada imagem que você anotou na etapa 2. Você pode identificar a linha JSON de uma imagem verificando o `source-ref` campo.
7. é o arquivo manifesto.
8. Exclua o conjunto de dados existente.
9. Crie um novo conjunto de dados com o arquivo de manifesto atualizado.
10. Treinar o modelo

Excluir um conjunto de dados

É possível excluir um conjunto de dados de um projeto do usando o console ou a `DeleteDataset` operação. As imagens referenciadas por um conjunto de dados não são excluídas. Se você excluir o conjunto de dados de teste de um projeto que tem um conjunto de dados de treinamento e um conjunto de dados de teste, o projeto reverte para um único projeto de conjunto de dados. O conjunto de dados restante é dividido durante o treinamento para criar um conjunto de dados de treinamento e teste. Se você excluir o conjunto de dados de treinamento, não poderá treinar um modelo no projeto até criar um novo conjunto de dados de treinamento.

Exclusão de conjuntos de dados (console)

Para excluir um serviço, execute o procedimento a seguir. Se você excluir todos os conjuntos de dados em um projeto, a página Criar conjunto de dados será exibida.

Para excluir um segredo (console)

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.

2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos.
4. Na página Projetos, selecione o projeto que contém o conjunto de dados que você deseja excluir.
5. No painel de navegação esquerdo, selecione Conjunto de dados.
6. Escolha Ações e, em seguida, selecione o conjunto de dados que você deseja excluir.
7. Na caixa de diálogo Excluir, digite excluir para confirmar que deseja excluir o conjunto de dados.
8. Escolha Excluir conjunto de dados de treinamento ou Excluir conjunto de dados de teste para excluir o conjunto de dados.

Como excluir um conjunto de dados (SDK)

Use a `DeleteDataset` operação para excluir um conjunto de dados.

Para excluir um conjunto de dados (SDK)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Para excluir uma coleção, use o seguinte código de exemplo.

   CLI

   Altere o valor do seguinte

   - `project-name` para o nome do projeto que contém o modelo a ser excluído.
   - `dataset-type` para um `train` ou `test`, dependendo do conjunto de dados que você deseja excluir. Se você tiver um único projeto de conjunto de dados, especifique `train` a exclusão do conjunto de dados.

```
aws lookoutvision delete-dataset --project-name project name\
  --dataset-type dataset type \
  --profile lookoutvision-access
```

Python

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
    @staticmethod
    def delete_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Deletes a Lookout for Vision dataset

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset
that
                             you want to delete.
        :param dataset_type: The type (train or test) of the dataset that you
                             want to delete.
        """
        try:
            logger.info(
                "Deleting the %s dataset for project %s.", dataset_type,
project_name
            )
            lookoutvision_client.delete_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            logger.info("Dataset deleted.")
        except ClientError:
            logger.exception("Service error: Couldn't delete dataset.")
            raise
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
/**
 * Deletes the train or test dataset in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to delete a
 *                    dataset.
```

```
     * @param datasetType The type of the dataset that you want to delete (train or
     *                    test).
     * @return Nothing.
     */
    public static void deleteDataset(LookoutVisionClient lfvClient, String
     projectName, String datasetType)
                    throws LookoutVisionException {

            logger.log(Level.INFO, "Deleting {0} dataset for project {1}",
                            new Object[] { datasetType, projectName });

            DeleteDatasetRequest deleteDatasetRequest =
     DeleteDatasetRequest.builder()
                            .projectName(projectName)
                            .datasetType(datasetType)
                            .build();

            lfvClient.deleteDataset(deleteDatasetRequest);

            logger.log(Level.INFO, "Deleted {0} dataset for project {1}",
                            new Object[] { datasetType, projectName });

    }
```

Exportação de conjuntos de dados de um projeto (SDK)

Você pode usar o AWS SDK para exportar conjuntos de dados de um projeto Amazon Lookout for Vision para um local de bucket do Amazon S3.

Ao exportar um conjunto de dados, você pode realizar tarefas como criar um projeto Lookout for Vision com uma cópia dos conjuntos de dados de um projeto de origem. Você também pode criar um instantâneo dos conjuntos de dados usados para uma versão específica de um modelo.

O código Python neste procedimento exporta o conjunto de dados de treinamento (imagens do manifesto e do conjunto de dados) de um projeto para um local de destino do Amazon S3 que você especificar. Se estiver presente no projeto, o código também exporta o manifesto do conjunto de dados de teste e as imagens do conjunto de dados. O destino pode estar no mesmo bucket do Amazon S3 do projeto de origem ou em um bucket diferente do Amazon S3. O código usa a operação ListDatasetEntries para obter os arquivos de manifesto do conjunto de dados. As

operações do Amazon S3 copiam as imagens do conjunto de dados e os arquivos de manifesto atualizados para o local de destino do Amazon S3.

Este procedimento mostra como exportar conjuntos de dados do projeto. Também mostra como criar um novo projeto com conjuntos de dados exportados.

Para exportar os conjuntos de dados de um projeto (SDK)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Determine o caminho de destino do Amazon S3 para a exportação do conjunto de dados. Certifique-se de que o destino esteja em uma região AWS compatível com o Amazon Lookout for Vision. Para criar um novo bucket do Amazon S3, consulte Criação de um bucket.
3. Certifique-se de que o usuário tenha permissões de acesso ao caminho de destino do Amazon S3 para a exportação do conjunto de dados e às localizações do S3 para os arquivos de imagem nos conjuntos de dados do projeto de origem. Você pode usar a política a seguir, que pressupõe que os arquivos de imagens possam estar em qualquer local. Substitua o *bucket/path pelo* bucket e o caminho de destino para a exportação do conjunto de dados.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PutExports",
            "Effect": "Allow",
            "Action": [
                "S3:PutObjectTagging",
                "S3:PutObject"
            ],
            "Resource": "arn:aws:s3:::bucket/path/*"
        },
        {
            "Sid": "GetSourceRefs",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

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

- Usuários e grupos no AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Create a permission set (Criação de um conjunto de permissões) no Guia do usuário do AWS IAM Identity Center.

- Usuários gerenciados no IAM usando um provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criar 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 Creating a role for an IAM user (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.

4. Salve o código a seguir em um arquivo chamado `dataset_export.py`.

```
"""
Purpose

Shows how to export the datasets (manifest files and images)
from an Amazon Lookout for Vision project to a new Amazon
S3 location.
"""

import argparse
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def copy_file(s3_resource, source_file, destination_file):
    """
    Copies a file from a source Amazon S3 folder to a destination
    Amazon S3 folder.
    The destination can be in a different S3 bucket.
    :param s3: An Amazon S3 Boto3 resource.
    :param source_file: The Amazon S3 path to the source file.
    :param destination_file: The destination Amazon S3 path for
    the copy operation.
    """

    source_bucket, source_key = source_file.replace("s3://", "").split("/", 1)
    destination_bucket, destination_key = destination_file.replace("s3://",
 "").split(
        "/", 1
    )

    try:
        bucket = s3_resource.Bucket(destination_bucket)
        dest_object = bucket.Object(destination_key)
        dest_object.copy_from(CopySource={"Bucket": source_bucket, "Key":
 source_key})
        dest_object.wait_until_exists()
        logger.info("Copied %s to %s", source_file, destination_file)
    except ClientError as error:
        if error.response["Error"]["Code"] == "404":
            error_message = (
                f"Failed to copy {source_file} to "
                f"{destination_file}. : {error.response['Error']['Message']}"
            )
            logger.warning(error_message)
            error.response["Error"]["Message"] = error_message
        raise


def upload_manifest_file(s3_resource, manifest_file, destination):
    """
    Uploads a manifest file to a destination Amazon S3 folder.
    :param s3: An Amazon S3 Boto3 resource.
    :param manifest_file: The manifest file that you want to upload.
    :destination: The Amazon S3 folder location to upload the manifest
    file to.
    """
```

```
    destination_bucket, destination_key = destination.replace("s3://",
 "").split("/", 1)

    bucket = s3_resource.Bucket(destination_bucket)

    put_data = open(manifest_file, "rb")
    obj = bucket.Object(destination_key + manifest_file)

    try:
        obj.put(Body=put_data)
        obj.wait_until_exists()
        logger.info("Put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name
        )
        raise
    finally:
        if getattr(put_data, "close", None):
            put_data.close()


def get_dataset_types(lookoutvision_client, project):
    """
    Determines the types of the datasets (train or test) in an
    Amazon Lookout for Vision project.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to check.
    :return: The dataset types in the project.
    """

    try:
        response = lookoutvision_client.describe_project(ProjectName=project)

        datasets = []

        for dataset in response["ProjectDescription"]["Datasets"]:
            if dataset["Status"] in ("CREATE_COMPLETE", "UPDATE_COMPLETE"):
                datasets.append(dataset["DatasetType"])
        return datasets

    except lookoutvision_client.exceptions.ResourceNotFoundException:
```

```
        logger.exception("Project %s not found.", project)
        raise


def process_json_line(s3_resource, entry, dataset_type, destination):
    """
    Creates a JSON line for a new manifest file, copies image and mask to
    destination.
    :param s3_resource: An Amazon S3 Boto3 resource.
    :param entry: A JSON line from the manifest file.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest
    file and dataset images.
    :return: A JSON line with details for the destination location.
    """
    entry_json = json.loads(entry)

    print(f"source: {entry_json['source-ref']}")

    # Use existing folder paths to ensure console added image names don't clash.
    bucket, key = entry_json["source-ref"].replace("s3://", "").split("/", 1)
    logger.info("Source location: %s/%s", bucket, key)

    destination_image_location = destination + dataset_type + "/images/" + key

    copy_file(s3_resource, entry_json["source-ref"], destination_image_location)

    # Update JSON for writing.
    entry_json["source-ref"] = destination_image_location

    if "anomaly-mask-ref" in entry_json:
        source_anomaly_ref = entry_json["anomaly-mask-ref"]
        mask_bucket, mask_key = source_anomaly_ref.replace("s3://", "").split("/",
 1)

        destination_mask_location = destination + dataset_type + "/masks/" +
 mask_key
        entry_json["anomaly-mask-ref"] = destination_mask_location

        copy_file(s3_resource, source_anomaly_ref, entry_json["anomaly-mask-ref"])

    return entry_json
```

```
def write_manifest_file(
    lookoutvision_client, s3_resource, project, dataset_type, destination
):
    """
    Creates a manifest file for a dataset. Copies the manifest file and
    dataset images (and masks, if present) to the specified Amazon S3 destination.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to use.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest file
    and dataset images.
    """

    try:
        # Create a reusable Paginator
        paginator = lookoutvision_client.get_paginator("list_dataset_entries")

        # Create a PageIterator from the Paginator
        page_iterator = paginator.paginate(
            ProjectName=project,
            DatasetType=dataset_type,
            PaginationConfig={"PageSize": 100},
        )

        output_manifest_file = dataset_type + ".manifest"

        # Create manifest file then upload to Amazon S3 with images.
        with open(output_manifest_file, "w", encoding="utf-8") as manifest_file:
            for page in page_iterator:
                for entry in page["DatasetEntries"]:
                    try:
                        entry_json = process_json_line(
                            s3_resource, entry, dataset_type, destination
                        )

                        manifest_file.write(json.dumps(entry_json) + "\n")

                    except ClientError as error:
                        if error.response["Error"]["Code"] == "404":
                            print(error.response["Error"]["Message"])
                            print(f"Excluded JSON line: {entry}")
                        else:
```

```
                            raise
        upload_manifest_file(
            s3_resource, output_manifest_file, destination + "datasets/"
        )

    except ClientError:
        logger.exception("Problem getting dataset_entries")
        raise


def export_datasets(lookoutvision_client, s3_resource, project, destination):
    """
    Exports the datasets from an Amazon Lookout for Vision project to a specified
    Amazon S3 destination.
    :param project: The Lookout for Vision project that you want to use.
    :param destination: The destination Amazon S3 folder for the exported datasets.
    """
    # Add trailing backslash, if missing.
    destination = destination if destination[-1] == "/" else destination + "/"

    print(f"Exporting project {project} datasets to {destination}.")

    # Get each dataset and export to destination.

    dataset_types = get_dataset_types(lookoutvision_client, project)
    for dataset in dataset_types:
        logger.info("Copying %s dataset to %s.", dataset, destination)

        write_manifest_file(
            lookoutvision_client, s3_resource, project, dataset, destination
        )

    print("Exported dataset locations")
    for dataset in dataset_types:
        print(f"   {dataset}: {destination}datasets/{dataset}.manifest")

    print("Done.")


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

```
    parser.add_argument("project", help="The project that contains the dataset.")
    parser.add_argument("destination", help="The destination Amazon S3 folder.")


def main():
    """
    Exports the datasets from an Amazon Lookout for Vision project to a
    destination Amazon S3 location.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    add_arguments(parser)

    args = parser.parse_args()

    try:
        session = boto3.Session(profile_name="lookoutvision-access")
        lookoutvision_client = session.client("lookoutvision")
        s3_resource = session.resource("s3")

        export_datasets(
            lookoutvision_client, s3_resource, args.project, args.destination
        )
    except ClientError as err:
        logger.exception(err)
        print(f"Failed: {format(err)}")


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

5. Execute o código. Forneça os seguintes argumentos de linha de comando:

   - projeto — O nome do projeto de origem que contém os conjuntos de dados que você deseja exportar.
   - destination — O caminho de destino do Amazon S3 para os conjuntos de dados.

   Por exemplo, `python dataset_export.py` *`myproject`* `s3://`*`bucket/path/`*

6. Observe os locais dos arquivos de manifesto que o código exibe. Você vai precisar dele na etapa 3.

7. Crie um novo projeto do Lookout for Vision com o conjunto de dados exportado seguindo as instruções em Criar seu projeto
8. Faça um dos seguintes procedimentos:
   - Use o console do Lookout for Vision para criar conjuntos de dados para seu novo projeto seguindo as instruções em Criação de um conjunto de dados com um arquivo de manifesto (console) Não é necessário executar as etapas 1 a 6.

     Para a etapa 12, faça o seguinte:

     a. Se o projeto de origem tiver um conjunto de dados de teste, escolha Conjunto de dados de treinamento e teste separados; caso contrário, escolha um único conjunto de dados.

     b. Para o local do arquivo manifest, digite o local do arquivo de manifesto apropriado (treinamento ou teste) que você anotou na etapa 6.
   - Use a operação CreateDataset para criar conjuntos de dados para seu novo projeto usando o código em. Criação de um conjunto de dados com um arquivo de manifesto (SDK) Para o `manifest_file` parâmetro, use a localização do arquivo de manifesto que você anotou na etapa 6. Se o projeto de origem tiver um conjunto de dados de teste, use o código novamente para criar o conjunto de dados de teste.
9. Se você estiver pronto, treine o modelo seguindo as instruções emTreinamento de seu modelo.

Visualizar as modelos

Um projeto pode ter várias versões de um modelo. Você pode usar o console para visualizar os modelos em um projeto. Você também pode usar a operação `ListModels`.

> **Note**
>
> A lista de modelos é eventualmente consistente. Se você criar um modelo, talvez seja necessário esperar um pouco para que a lista de modelos seja atualizada.

Visualizar seus modelos (console)

Realize as etapas do procedimento a seguir para exibir os modelos do seu projeto no console do.

Para visualizar seus modelos (console)

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos.
4. Na página Projetos, selecione o projeto que contém os modelos que você deseja visualizar.
5. No painel de navegação à esquerda, escolha Modelos e, em seguida, visualize os detalhes do modelo.

Visualizando seus modelos (SDK)

Para visualizar as versões de um modelo, você usa a operação `ListModels`. Para obter informações sobre uma versão específica do modelo, use a operação `DescribeModel`. O exemplo a seguir lista todas as versões do modelo em um projeto e, em seguida, exibe informações de desempenho e configuração de saída para versões individuais do modelo.

Para visualizar seus modelos (SDK)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Use o código de exemplo a seguir para listar seus modelos e obter informações sobre um modelo.

   CLI

   Use o comando `list-models` para listar os modelos em um projeto.

   Altere os seguintes valores:

   - `project-name` para o nome do projeto que contém o modelo que você deseja visualizar.

   ```
   aws lookoutvision list-models --project-name project name \
     --profile lookoutvision-access
   ```

   Use o comando `describe-model` para obter informações sobre um modelo. Altere os seguintes valores:

- `project-name` para o nome do projeto que contém o modelo que você deseja visualizar.
- `model-version` para a versão do modelo que você deseja descrever.

```
aws lookoutvision describe-model --project-name project name\
    --model-version model version \
   --profile lookoutvision-access
```

Python

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
    @staticmethod
    def describe_models(lookoutvision_client, project_name):
        """
        Gets information about all models in a Lookout for Vision project.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to use.
        """
        try:
            response =
 lookoutvision_client.list_models(ProjectName=project_name)
            print("Project: " + project_name)
            for model in response["Models"]:
                Models.describe_model(
                    lookoutvision_client, project_name, model["ModelVersion"]
                )
                print()
            print("Done...")
        except ClientError:
            logger.exception("Couldn't list models.")
            raise
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
/**
 * Lists the models in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the models that
 *                    you want to list.
 * @return List <Metadata> A list of models in the project.
 */
public static List<ModelMetadata> listModels(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {

        ListModelsRequest listModelsRequest = ListModelsRequest.builder()
                        .projectName(projectName)
                        .build();

        // Get a list of models in the supplied project.
        ListModelsResponse response = lfvClient.listModels(listModelsRequest);

        for (ModelMetadata model : response.models()) {
                logger.log(Level.INFO, "Model ARN: {0}\nVersion: {1}\nStatus:
 {2}\nMessage: {3}", new Object[] {
                                        model.modelArn(),
                                        model.modelVersion(),
                                        model.statusMessage(),
                                        model.statusAsString() });
        }

        return response.models();

}
```

Excluir um modelo

É possível excluir uma versão de um modelo do usando o console ou a operação `DeleteModel`. Você não pode excluir a versão do modelo que está em execução ou sendo treinada.

Se a versão do modelo estiver em execução, primeiro use a operação `StopModel` para interromper a versão do modelo. Para obter mais informações, consulte Parar o modelo do Amazon Lookout for Vision. Se um modelo estiver sendo treinado, espere até que ele termine antes de excluí-lo.

Pode demorar alguns segundos para excluir um modelo. Para determinar se um modelo foi excluído, chame ListProjects e verifique se a versão do modelo (`ModelVersion`) está na `Models` matriz.

Excluir um modelo do (console)

Execute as etapas a seguir para excluir um modelo do console.

Para excluir um canal (console)

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Projetos.
4. Na página Projetos, selecione o projeto que contém o modelo que você deseja excluir.
5. No painel de navegação à esquerda, selecione Modelos.
6. Na exibição de modelos, selecione o botão de opção do modelo que você deseja excluir.
7. Escolha Excluir, no alto da página.
8. Na caixa de diálogo Excluir, digite excluir para confirmar que deseja excluir o modelo.
9. Escolha Excluir) para excluir o modelo.

Exclusão de um modelo (SDK)

Use o procedimento a seguir para excluir o modelo com a operação `DeleteModel`.

Para excluir um modelo (SDK)

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e os SDKs do AWS. Para obter mais informações, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.
2. Para excluir uma coleção, use o seguinte código de exemplo.

   CLI

   Altere os seguintes valores:

   - `project-name` para o nome do projeto que contém o modelo a ser excluído.
   - `model-version` para a versão do modelo que você deseja excluir.

```
aws lookoutvision delete-model --project-name project name\
```

```
  --model-version model version \
  --profile lookoutvision-access
```

Python

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
    @staticmethod
    def delete_model(lookoutvision_client, project_name, model_version):
        """
        Deletes a Lookout for Vision model. The model must first be stopped and
can't
        be in training.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the desired
model.
        :param model_version: The version of the model that you want to delete.
        """
        try:
            logger.info("Deleting model: %s", model_version)
            lookoutvision_client.delete_model(
                ProjectName=project_name, ModelVersion=model_version
            )

            model_exists = True
            while model_exists:
                response =
lookoutvision_client.list_models(ProjectName=project_name)

                model_exists = False
                for model in response["Models"]:
                    if model["ModelVersion"] == model_version:
                        model_exists = True

                if model_exists is False:
                    logger.info("Model deleted")
                else:
                    logger.info("Model is being deleted...")
                    time.sleep(2)

            logger.info("Deleted Model: %s", model_version)
```

```
        except ClientError:
            logger.exception("Couldn't delete model.")
            raise
```

Java V2

Esse código foi retirado do repositório GitHub de exemplos do SDK da documentação AWS. Veja o exemplo completo aqui.

```
/**
* Deletes an Amazon Lookout for Vision model.
*
* @param lfvClient    An Amazon Lookout for Vision client. Returns after the
 model is deleted.
* @param projectName  The name of the project that contains the model that you
 want to delete.
* @param modelVersion The version of the model that you want to delete.
* @return void
*/
public static void deleteModel(LookoutVisionClient lfvClient,
                String projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        DeleteModelRequest deleteModelRequest = DeleteModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        lfvClient.deleteModel(deleteModelRequest);

        boolean deleted = false;

        do {

                ListModelsRequest listModelsRequest =
 ListModelsRequest.builder()
                                .projectName(projectName)
                                .build();

                // Get a list of models in the supplied project.
```

```
                ListModelsResponse response =
lfvClient.listModels(listModelsRequest);

                ModelMetadata modelMetadata = response.models().stream()
                                .filter(model ->
model.modelVersion().equals(modelVersion)).findFirst()
                                .orElse(null);

                if (modelMetadata == null) {
                        deleted = true;
                        logger.log(Level.INFO, "Deleted: Model version {0} of
project {1}.",
                                        new Object[] { modelVersion,
projectName });

                } else {
                        logger.log(Level.INFO, "Not yet deleted: Model version
{0} of project {1}.",
                                        new Object[] { modelVersion,
projectName });
                        TimeUnit.SECONDS.sleep(60);
                }

        } while (!deleted);

}
```

Modelos de marcação

Você pode identificar, organizar, pesquisar e filtrar seus modelos do Amazon Lookout for Vision 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 dos seus modelos, você pode marcá-los com uma chave `Cost center` e adicionar o número do centro de custo apropriado como um 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 Como usar tags de alocação de custos.
- Controle o acesso a um modelo usando o gerenciamento de identidade e acesso (IAM). Para obter mais informações, consulte Controle de acesso aos recursos do AWS usando tags de recursos.

- Automatize o gerenciamento de modelos. Por exemplo, você pode executar scripts de início ou parada automatizados que desativam os modelos de desenvolvimento durante o horário não comercial para reduzir os custos. Para obter mais informações, consulte Executando seu modelo treinado do Amazon Lookout for Vision.

Você pode marcar modelos usando o console Amazon Lookout for Vision ou usando os SDKs AWS.

Tópicos

Como marcar modelos (console)

Você pode usar o console Amazon Lookout for Vision para adicionar tags aos modelos, visualizar as tags anexadas a um modelo e remover tags.

Adicionar ou remover tags (console)

Esse procedimento explica como adicionar ou remover tags de um modelo existente. Você também pode adicionar tags a um novo modelo quando ele é treinado. Para obter mais informações, consulte Treinamento de seu modelo.

Para adicionar tags ou remover tags de um modelo existente (console)

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação, escolha 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 esta etapa.
12. (Opcional) Para remover uma tag, escolha Remover ao lado da tag que você deseja remover. Se você estiver removendo uma tag salva anteriormente, ela será removida quando você salvar suas alterações.
13. Escolha Salvar alterações para salvar suas alterações.

Exibir tags de modelos (console)

Você pode usar o console do Amazon Lookout for Vision 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 Listar tags de modelo (SDK).

Para visualizar as tags anexadas a um modelo

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação, escolha Projetos.
4. Na página Recursos de projetos, escolha 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 cuja tag você deseja visualizar.
7. Na página de detalhes do modelo, escolha a guia Tags. As tags são mostradas na seção Tags.

Modelos de marcação (SDK)

Você pode usar o SDK da AWS para:

- Adicionar tags a um novo modelo
- Adicionar tags a um modelo existente
- Liste as tags anexadas a um cofre.

- Remover tags de um modelo

Esta seção inclui exemplos AWS CLI. Se ainda não tiver instalado a AWS CLI, consulte Etapa 4: configurar os AWS SDKs AWS CLI e.

Adicionar tags a um novo modelo (SDK)

É possível adicionar tags a um modelo quando você o cria usando a operação CreateModel. Especifique uma ou mais tags no parâmetro `Tags` de entrada da matriz.

```
aws lookoutvision create-model --project-name "project name"\
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":  "output
 folder" } }'\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Para obter mais informações sobre como criar um trabalho de treinamento, consulte Treinando um modelo (SDK).

Adicionar tags a um modelo existente (SDK)

Para adicionar uma ou mais tags a um modelo existente, use a operação TagResource. Especifique o Nome de recurso da Amazon (ARN) do modelo (`ResourceArn`) e as tags (`Tags`) que você deseja adicionar.

```
aws lookoutvision tag-resource --resource-arn "resource-arn"\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Por exemplo, código Java, consulte TagModel.

Listar tags de modelo (SDK)

Para listar as tags anexadas a um modelo, use a operação ListTagsForResource e especifique o Amazon Resource Name (ARN) do modelo, o (`ResourceArn`). A resposta é um mapa de chaves e valores de tag anexados ao modelo especificado.

```
aws lookoutvision list-tags-for-resource --resource-arn resource-arn \
  --profile lookoutvision-access
```

Para ver quais modelos em um projeto têm uma tag específica, chame `ListModels` para obter uma lista de modelos. Em seguida, ligue `ListTagsForResource` para cada modelo no formulário de respostaListModels. Inspecione a resposta de `ListTagsForResource` para ver se a tag necessária está presente.

Por exemplo, código Java, consulte ListModelTags. Por exemplo, código Python que pesquisa um valor de tag em todos os projetos, consulte find_tag.py.

Remoção de tags de um modelo (SDK)

Para remover uma ou mais tags de um modelo, use a operação UntagResource. Especifique o Amazon Resource Name (ARN) do modelo (`ResourceArn`) e as chaves de tag (`Tag-Keys`) que você deseja remover.

```
aws lookoutvision untag-resource --resource-arn resource-arn\
  --tag-keys '["Key"]' \
  --profile lookoutvision-access
```

Por exemplo, código Java, consulte UntagModel.

Visualizando suas tarefas de detecção de testes

Você pode visualizar suas detecções de testes usando o console do. Você não pode usar o SDK da AWS para visualizar tarefas de detecção de testes.

> **Note**
>
> A lista de detecções de ensaios clínicos acaba sendo consistente. Se você criar uma detecção de teste, talvez seja necessário esperar um pouco antes que a lista de detecções de teste seja atualizada.

Visualizando suas tarefas de detecção de testes (console)

Use os procedimentos a seguir para visualizar suas detecções de testes.

Para visualizar suas tarefas de detecção de testes

1. Abra o console Amazon Lookout for Vision em https://console.aws.amazon.com/lookoutvision/.
2. Escolha Comece a usar.
3. No painel de navegação à esquerda, escolha Detecções de teste.
4. Na página de detecções de testes, escolha uma tarefa de detecção de testes para ver seus detalhes.

Exemplos de códigos e conjuntos de dados

Veja a seguir exemplos de código e conjuntos de dados que você pode usar com o Amazon Lookout for Vision.

Tópicos

Código de exemplo

Os exemplos de códigos a seguir do Amazon Lookout for Vision estão disponíveis.

Exemplo	Descrição
GitHub	Exemplo de código Python que treina e hospeda um modelo Amazon Lookout for Vision.
Amazon Lookout for Vision Lab	Um notebook Python que você pode usar para criar um modelo com as imagens de exemplo da placa de circuito.
Código de exemplo em Python	Exemplos de Python usados na documentação do Amazon Lookout for Vision.
Código de exemplo Java	Exemplos de Java usados na documentação do Amazon Lookout for Vision.

Exemplos de conjuntos de dados

Veja a seguir exemplos de conjuntos de dados que você pode usar com o Amazon Lookout for Vision.

Tópicos

- conjuntos de dados de segmentação de imagens
- Conjunto de dados de classificação de imagens

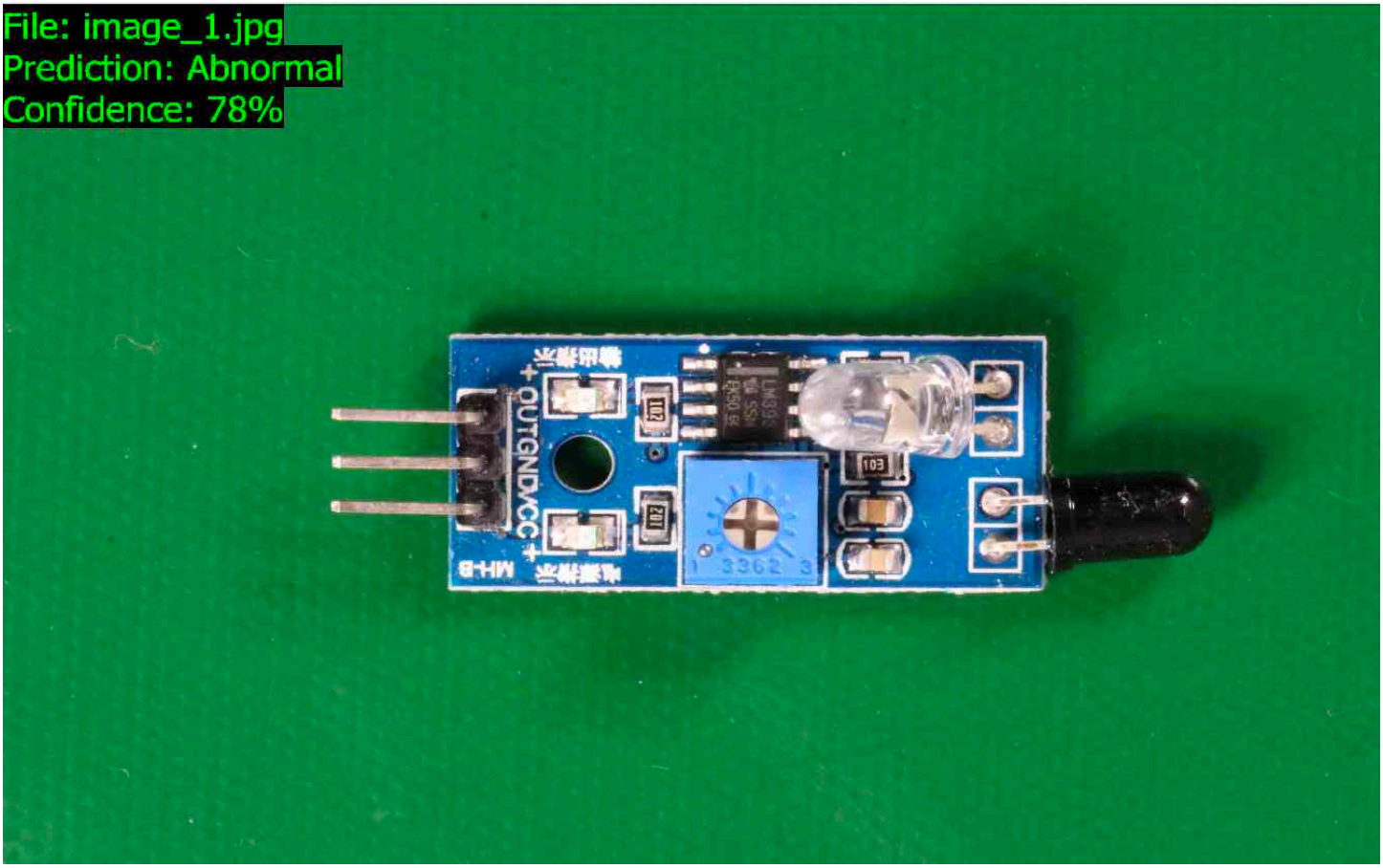
conjuntos de dados de segmentação de imagens

Conceitos básicos do Amazon Lookout for Vision fornece um conjunto de dados de cookies quebrados que você pode usar para criar um modelo de segmentação de imagens.

Para outro conjunto de dados que cria um modelo de segmentação de imagens, consulte Identificar a localização de anomalias usando o Amazon Lookout for Vision na borda sem usar uma GPU.

Conjunto de dados de classificação de imagens

O Amazon Lookout for Vision fornece exemplos de imagens de placas de circuito que você pode usar para criar um modelo de classificação de imagens.



Você pode copiar as imagens do repositório https://github.com/aws-samples/amazon-lookout-for-vision do GitHub. As imagens estão na `circuitboard` pasta.

A `circuitboard` pasta tem as seguintes pastas.

- `train`— Imagens que você pode usar em um conjunto de dados de treinamento.
- `test`— Imagens que você pode usar em um conjunto de dados de teste.
- `extra_images`— Imagens que você pode usar para executar um teste de detecção ou testar seu modelo treinado com a operação DetectAnomalies.

As pastas `train` e `test` têm, cada uma, uma subpasta denominada `normal` (contém imagens normais) e uma subpasta denominada `anomaly` (contém imagens com anomalias).

> **Note**
>
> Posteriormente, ao criar um conjunto de dados com o console, o Amazon Lookout for Vision pode usar os nomes das pastas (`normal` e `anomaly`) para rotular as imagens automaticamente. Para obter mais informações, consulte the section called “Bucket do Amazon S3”.

Para preparar as imagens do conjunto de dados

1. Clone o repositório https://github.com/aws-samples/amazon-lookout-for-vision em seu computador. Para obter mais informações, consulte Clonagem de um repositório.
2. Crie um bucket do Amazon S3. Para obter mais informações, consulte Como faço para criar um bucket S3?.
3. No prompt de comando, insira o seguinte comando para copiar as imagens do conjunto de dados do seu computador para o bucket do Amazon S3.

   ```
   aws s3 cp --recursive your-repository-folder/circuitboard s3://your-bucket/
   circuitboard
   ```

Depois de fazer o upload das imagens, você pode criar um modelo. Você pode classificar automaticamente as imagens adicionando as imagens do local do Amazon S3 para o qual você carregou anteriormente as imagens da placa de circuito. Lembre-se de que você é cobrado por cada treinamento bem-sucedido de um modelo e pela quantidade de tempo em que o modelo está sendo executado (hospedado).

Como criar um modelo de classificação

1. Faça Criar um projeto (console)
2. Faça Criar um conjunto de dados usando imagens armazenadas em um bucket do Amazon S3

   - Para a etapa 6, escolha a guia Separar conjuntos de dados de treinamento e teste.
   - Para a etapa 8a, insira o URI do S3 para as imagens de treinamento que você carregou em Para preparar as imagens do conjunto de dados. Por exemplo `s3://your-bucket/circuitboard/train`. Para a etapa 8b, insira o URI do S3 do conjunto de dados de teste. Por exemplo, `s3://your-bucket/circuitboard/test`.
   - Certifique-se de executar a etapa 9.
3. Faça Treinando um modelo (console)
4. Faça Iniciar seu modelo (console)
5. Faça Detectar as anomalias de uma imagem Você pode usar imagens da pasta `test_images`.
6. Quando terminar de usar o modelo, façaParar o modelo (console).

Segurança no Amazon Lookout for Vision

A segurança para com a nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você se beneficiará de datacenters e arquiteturas de rede criados para atender aos requisitos das empresas com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa produtos da AWS na Nuvem AWS. A AWS também fornece serviços que podem ser usados com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon Lookout for Vision, consulte Serviços do AWS no escopo por programa de conformidade.
- Segurança na nuvem: sua responsabilidade é determinada pelo serviço da AWS que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua empresa e as leis e regulamentos aplicáveis.

Esta documentação o ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Lookout for Vision. Os tópicos a seguir mostram como configurar o Lookout for Vision para atender aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros serviços do AWS que o ajudam a monitorar e proteger seus recursos do Lookout for Vision.

Tópicos

Proteção de dados no Amazon Lookout for Vision

O modelo de responsabilidade compartilhada da AWS aplica-se à proteção de dados no Amazon Lookout for Vision. Conforme descrito nesse modelo, a AWS é responsável por proteger a

infraestrutura global que executa toda a Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para mais informações sobre a proteção de dados na Europa, consulte o artigo AWS Shared Responsibility Model and GDPR no Blog de segurança da AWS.

Para fins de proteção de dados, recomendamos que você proteja asConta da AWS credenciais da e configure as contas de usuário individuais com o AWS IAM Identity Center ou o AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA [multi-factor authentication]) com cada conta.
- Use SSL/TLS para se comunicar com os atributos da AWS. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure o registro em log das atividades da API e do usuário com o .AWS CloudTrail
- Use AWS as soluções de criptografia da , juntamente com todos os controles de segurança padrão dos Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar a AWS por meio de uma interface de linha de comandos ou uma API, use um endpoint do FIPS. Para ter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Name (Nome). Isso inclui quando você trabalha com o Lookout for Vision ou outros Serviços da AWS usando o console, a API, AWS CLI ou AWS SDKs. Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Criptografia de dados

As informações a seguir explicam onde o Amazon Lookout for Vision usa a criptografia de dados para proteger seus dados.

Criptografia inativa

Imagens

Para treinar seu modelo, o Amazon Lookout for Vision faz uma cópia de suas imagens originais de treinamento e teste. As imagens copiadas são criptografadas em repouso no Amazon Simple Storage Service (S3) usando criptografia do lado do servidor com uma ou uma chave fornecida por você. Chave pertencente à AWS As chaves são armazenadas usando o AWS Key Management Service (SSE-KMS). Suas imagens de origem não são afetadas. Para obter mais informações, consulte Treinamento de seu modelo.

Amazon Lookout para modelos Vision

Por padrão, modelos treinados e arquivos de manifesto são criptografados no Amazon S3 usando a criptografia no lado do servidor com chaves do KMS armazenadas no AWS Key Management Service (SSE-KMS). O Lookout for Vision usa Chave pertencente à AWS um. Para obter mais informações, consulte Proteger dados usando a criptografia no lado do servidor. Os resultados do treinamento são gravados no bucket especificado no parâmetro `output_bucket` de entrada para`CreateModel`. Os resultados do treinamento são criptografados usando as configurações de criptografia definidas para o bucket (`output_bucket`).

Bucket de console do Amazon Lookout for Vision

O console Amazon Lookout for Vision cria um bucket Amazon S3 (bucket de console) que você pode usar para gerenciar seus projetos. O bucket do console é criptografado usando a criptografia padrão do Amazon S3. Para obter mais informações, consulte Criptografia padrão do Amazon Simple Storage Service para buckets do S3. Se você estiver usando sua própria chave KMS, configure o bucket do console depois que ele for criado. Para obter mais informações, consulte Proteger dados usando a criptografia no lado do servidor. O Amazon Lookout for Vision bloqueia o acesso público ao bucket do console.

Criptografia em trânsito

Os pontos de extremidade da API do Amazon Lookout for Vision suportam apenas conexões seguras por HTTPS. Toda a comunicação é criptografada com Transport Layer Security (TLS).

Gerenciamento de chaves

É possível usar o AWS Key Management Service (KMS) para gerenciar a criptografia das imagens de entrada que você armazena nos buckets do Amazon S3. Para obter mais informações, consulte Etapa 5: (Opcional) Usando sua própria chave do AWS Key Management Service.

Por padrão, suas imagens são criptografadas com uma chave que a AWS possui e gerencia. Você também pode optar por usar sua própria AWS Key Management Service (KMS). Para obter mais informações, consulte Conceitos do AWS Key Management Service.

Privacidade do tráfego entre redes

Um ponto de extremidade do Amazon Virtual Private Cloud (Amazon VPC) para o Amazon Lookout for Vision é uma entidade lógica dentro de um VPC que permite conectividade apenas com o Amazon Lookout for Vision. O Amazon VPC encaminha as solicitações ao Amazon Lookout for Vision e encaminha as respostas de volta ao VPC. Para obter mais informações, consulte Endpoints da VPC no Guia do usuário da Amazon VPC. Para obter informações sobre o uso de endpoints do Amazon VPC com o Amazon Lookout for Vision, consulte. Acesse o Amazon Lookout for Vision usando um endpoint de interface (AWS PrivateLink)

Gerenciamento de identidade e acesso para o Amazon Lookout for Vision

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores de IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar os recursos do Lookout for Vision. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Políticas gerenciadas da AWS para o Amazon Lookout for Vision
- Solução de problemas de identidade e acesso do Amazon Lookout for Vision

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no Lookout for Vision.

Usuário do serviço: se você usar o serviço Lookout for Vision para fazer seu trabalho, o administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usar mais recursos do Lookout for Vision para fazer seu trabalho, poderá precisar de permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se não conseguir acessar um recurso no Lookout for Vision, consulte Solução de problemas de identidade e acesso do Amazon Lookout for Vision.

Administrador de serviços: se você for o responsável pelos recursos do Lookout for Vision em sua empresa, provavelmente terá acesso total ao Lookout for Vision. É sua função determinar quais recursos e funcionalidades do Lookout for Vision os usuários do serviço devem acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como sua empresa pode usar o IAM com o Lookout for Vision, consulte Como o Amazon Lookout for Vision funciona com o IAM.

Administrador de IAM: se você for um administrador de IAM, talvez queira saber detalhes sobre como escrever políticas para gerenciar o acesso ao Lookout for Vision. Para ver exemplos de políticas baseadas em identidade do Lookout for Vision que você pode usar no IAM, consulte Exemplos de políticas baseadas em identidade do Amazon Lookout for Vision.

Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já

configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinatura de solicitações de AWS API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia AWS IAM Identity Center do usuário e Utilizar a autenticação multifator (MFA) na AWS no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do Usuário do IAM.

Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade.

Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas da AWS aplicativos. Para obter mais informações sobre o Centro de Identidade do IAM, consulte "O que é o Centro de Identidade do IAM?" no Guia do usuário AWS IAM Identity Center .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma função do IAM no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para o uso de perfis, consulte Utilizar perfis do IAM no Guia do usuário do IAM.

Funções do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do Usuário do IAM. Se você usar o Centro de identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no Guia do usuário AWS IAM Identity Center .
- Permissões temporárias para usuários do IAM –- um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas –- é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.
  - Função de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de

serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para funções vinculadas ao serviço.

- Aplicativos em execução no Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. AWS É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Utilizar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciando acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do Usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

As políticas do IAM definem permissões para uma ação independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do Usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. AWS WAF Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do Desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um atributo avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do Usuário do IAM.
- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em. AWS Organizations AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizações e SCPs, consulte How SCPs work (Como os SCPs funcionam) no Guia do usuário do AWS Organizations .
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do Usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como o Amazon Lookout for Vision funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Lookout for Vision, saiba quais recursos do IAM estão disponíveis para uso com o Lookout for Vision.

Recursos do IAM que você pode usar com o Amazon Lookout for Vision

Atributo do IAM	Lookout for Vision
Políticas baseadas em identidade	Sim
Políticas baseadas em recursos	Não
Ações das políticas	Sim
Atributos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
ABAC (tags em políticas)	Parcial
Credenciais temporárias	Sim
Sessões de acesso direto (FAS)	Sim
Perfis de serviço	Não
Funções vinculadas ao serviço	Não

Para ter uma visão geral de como o Lookout for Vision e AWS outros serviços funcionam com a maioria dos recursos do IAM, AWS consulte os serviços que funcionam com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade para o Lookout for Vision

Suporta políticas baseadas em identidade	Sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil ao qual ela está anexada. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elementos da política JSON do IAM no Guia do Usuário do IAM.

Exemplos de políticas baseadas em identidade para o Lookout for Vision

Para ver exemplos de políticas baseadas em identidade do Lookout for Vision, consulte Exemplos de políticas baseadas em identidade do Amazon Lookout for Vision.

Políticas baseadas em recursos no Lookout for Vision

Oferece compatibilidade com políticas baseadas em recursos	Não

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para

o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, você pode especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em atributo. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso são diferentes Contas da AWS, um administrador do IAM na conta confiável também deve conceder permissão à entidade principal (usuário ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Para obter mais informações, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Ações de políticas para o Lookout for Vision

Oferece compatibilidade com ações de políticas	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de ações do Lookout for Vision, consulte Ações definidas pelo Amazon Lookout for Vision na Referência de Autorização de Serviço.

As ações de política no Lookout for Vision usam o seguinte prefixo antes da ação:

```
lookoutvision
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "lookoutvision:action1",
      "lookoutvision:action2"
         ]
```

Para ver exemplos de políticas baseadas em identidade do Lookout for Vision, consulte Exemplos de políticas baseadas em identidade do Amazon Lookout for Vision.

Recursos de políticas para o Lookout for Vision

Oferece compatibilidade com recursos de políticas	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Lookout for Vision e seus ARNs, consulte Recursos definidos pelo Amazon Lookout for Vision na Referência de Autorização de Serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon Lookout for Vision.

Para ver exemplos de políticas baseadas em identidade do Lookout for Vision, consulte Exemplos de políticas baseadas em identidade do Amazon Lookout for Vision.

Chaves de condição de política para o Lookout for Vision

Suporta chaves de condição de política específicas de serviço	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único `Condition` elemento, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um atributo somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos da política do IAM: variáveis e tags no Guia do usuário do IAM.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Para ver uma lista das chaves de condição do Lookout for Vision, consulte Chaves de condição do Amazon Lookout for Vision na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas pelo Amazon Lookout for Vision.

Para ver exemplos de políticas baseadas em identidade do Lookout for Vision, consulte Exemplos de políticas baseadas em identidade do Amazon Lookout for Vision.

ACLs no Lookout for Vision

Oferece compatibilidade com ACLs	Não

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com Lookout for Vision

Oferece compatibilidade com ABAC (tags em políticas)	Parcial

O controle de acesso baseado em recurso (ABAC) é uma estratégia de autorização que define permissões com base em recursos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a entidades do IAM (usuários ou funções) e a vários AWS recursos. A marcação de entidades e atributos é a primeira etapa do ABAC. Em seguida, você cria políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag do recurso que ela estiver tentando acessar.

O ABAC é útil em ambientes que estão crescendo rapidamente e ajuda em situações onde o gerenciamento de políticas se torna um problema.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobre o ABAC, consulte O que é ABAC? no Guia do Usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Utilizar controle de acesso baseado em atributos (ABAC) no Guia do usuário do IAM.

Usar credenciais temporárias com o Lookout for Vision

Oferece compatibilidade com credenciais temporárias	Sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS "Trabalhe com o IAM" no Guia do usuário do IAM.

Você está usando credenciais temporárias se fizer login AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre como alternar funções, consulte Alternar para um perfil (console) no Guia do usuário do IAM.

Você pode criar manualmente credenciais temporárias usando a AWS API AWS CLI ou. Em seguida, você pode usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para mais informações, consulte Credenciais de segurança temporárias no IAM.

Sessões de acesso direto para o Lookout for Vision

Suporte para o recurso Encaminhamento de sessões de acesso (FAS)	Sim

Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.

Perfis de serviço para o Lookout for Vision

Oferece suporte a perfis de serviço	Não

Um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> Alterar as permissões de um perfil de serviço pode interromper a funcionalidade do Lookout for Vision. Edite os perfis de serviço somente quando o Lookout for Vision instruir.

Funções vinculadas ao serviço para o Lookout for Vision

Oferece suporte a perfis vinculados ao serviço	Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Função vinculada ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Exemplos de políticas baseadas em identidade do Amazon Lookout for Vision

Por padrão, os usuários e as funções não têm permissão para criar ou modificar recursos do Lookout for Vision. Eles também não podem realizar tarefas usando a AWS API AWS Management Console, AWS Command Line Interface (AWS CLI) ou. Para conceder aos usuários permissão para executar

ações nos recursos de que precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documento de política JSON, consulte Criação de políticas do IAM no Guia do Usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Lookout for Vision, incluindo o formato dos ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição para o Amazon Lookout for Vision na Referência de Autorização de Serviço.

Tópicos

- Melhores práticas de política
- Acessar um único projeto Amazon Lookout for Vision
- Exemplo de política baseada em tags

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Lookout for Vision em sua conta. Essas ações podem incorrer em custos para seus Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do Usuário do IAM.
- Aplique permissões de privilégio mínimo –- ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do Usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso –- você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode

gravar uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS CloudFormation. Para obter mais informações, consulte Elementos da política JSON do IAM: Condição no Guia do usuário do IAM.

- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais –- o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de 100 verificações de política e recomendações acionáveis para ajudá-lo a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas Recomendadas de Segurança no IAM no Guia do Usuário do IAM.

Acessar um único projeto Amazon Lookout for Vision

Neste exemplo, você deseja conceder a um usuário da sua AWS conta acesso a um dos seus projetos do Amazon Lookout for Vision.

```
{
    "Sid": "SpecificProjectOnly",
    "Effect": "Allow",
    "Action": [
        "lookoutvision:DetectAnomalies"
    ],
    "Resource": "arn:aws:lookoutvision:us-east-1:123456789012:model/myproject/*"
}
```

Exemplo de política baseada em tags

As políticas baseadas em tags são documentos de políticas JSON que especificam as ações que uma entidade principal pode executar em recursos marcados.

Usar uma tag para acessar um recurso

Este exemplo de política concede a um usuário ou função da sua conta AWS permissão para usar a operação `DetectAnomalies` com qualquer modelo marcado com a chave `stage` e o valor `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "LookoutVision:DetectAnomalies"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
   ]
}
```

Use uma tag para negar acesso a operações específicas do Amazon Lookout for Vision

Este exemplo de política nega permissão a um usuário ou função em sua conta da AWS para chamar as operações `DeleteModel` ou `StopModel` com qualquer modelo marcado com a chave `stage` e o valor `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "LookoutVision:DeleteModel",
                "LookoutVision:StopModel"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
```

```
                }
            }
        }
    ]
}
```

Políticas gerenciadas da AWS para o Amazon Lookout for Vision

Uma política gerenciada da AWS é uma política autônoma criada e administrada pela AWS. As políticas gerenciadas da AWS foram projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas gerenciadas da AWS podem não conceder permissões de menor privilégio para seus casos de uso específicos, pois elas estão disponíveis para todos os clientes da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente específicas para seus casos de uso.

Você não pode alterar as permissões definidas em políticas gerenciadas pela AWS. Se a AWS atualiza as permissões definidas em um política gerenciada pela AWS, a atualização afeta todas as identidades de entidades principais (usuários, grupos e perfis) às quais a política está vinculada. É mais provável que a AWS atualize uma política gerenciada pela AWS quando um novo AWS service (Serviço da AWS) é lançado ou novas operações de API são disponibilizadas para os serviços existentes.

Para obter mais informações, consulte Políticas gerenciadas pela AWS no Manual do usuário do IAM.

Política gerenciada pela: AmazonLookoutVisionReadOnlyAccess

Use a política `AmazonLookoutVisionReadOnlyAccess` para permitir que os usuários acessem somente leitura o Amazon Lookout for Vision (e suas dependências) com as seguintes ações do Amazon Lookout for Vision (operações do SDK). Por exemplo, você pode usar `DescribeModel` para obter informações sobre um modelo existente.

- DescribeDataset
- Descreva o modelo

- DescribeModelPackagingJob
- DescribeProject
- Listar entradas do conjunto de dados
- ListModelPackagingJobs
- ListModels
- ListProjects
- ListTagsForResource

Para chamar ações somente para leitura, os usuários não precisam de permissões de bucket do Amazon S3. No entanto, as respostas da operação podem incluir referências aos buckets do Amazon S3. Por exemplo, a entrada `source-ref` na resposta de `ListDatasetEntries` é uma referência a uma imagem em um bucket do Amazon S3. Adicione permissões de bucket do Amazon S3 se seus usuários precisarem acessar os buckets referenciados. Por exemplo, um usuário pode querer baixar uma imagem referenciada por um campo `source-ref`. Para obter mais informações, consulte Concessão de permissões do Amazon S3 Bucket.

É possível anexar a política `AmazonLookoutVisionReadOnlyAccess` a suas identidades do IAM.

Detalhes da permissão

Esta política inclui as seguintes permissões.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
```

```
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gerenciada pela: AmazonLookoutVisionFullAccess

Use a política `AmazonLookoutVisionFullAccess` para permitir que os usuários tenham acesso total ao Amazon Lookout for Vision (e suas dependências) com ações do Amazon Lookout for Vision (operações do SDK). Por exemplo, você pode treinar um modelo sem precisar usar o console Amazon Lookout for Vision. Para obter mais informações, consulte Ações.

Para criar um conjunto de dados (`CreateDataset`) ou criar um modelo (`CreateModel`), seus usuários devem ter permissões de acesso total ao bucket do Amazon S3 que armazena imagens do conjunto de dados, arquivos de manifesto do Amazon SageMaker Ground Truth e resultados de treinamento. Para obter mais informações, consulte Etapa 2: Configurar permissões.

Você também pode dar permissão às ações do SDK do Amazon Lookout for Vision usando a política `AmazonLookoutVisionConsoleFullAccess`.

É possível anexar a política `AmazonLookoutVisionFullAccess` a suas identidades do IAM.

Detalhes da permissão

Esta política inclui as seguintes permissões.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Política gerenciada pela: AmazonLookoutVisionConsoleFullAccess

Use a política `AmazonLookoutVisionFullAccess` para permitir que os usuários tenham acesso total ao console do Amazon Lookout for Vision, às ações (operações do SDK) e a quaisquer dependências que o serviço tenha. Para obter mais informações, consulte Conceitos básicos do Amazon Lookout for Vision.

A política `LookoutVisionConsoleFullAccess` inclui permissões para o bucket do console Amazon Lookout for Vision. Para obter informações sobre o bucket do console, consulte Etapa 3: Criar um bucket do console. Para armazenar conjuntos de dados, imagens e arquivos de manifesto do Amazon SageMaker Ground Truth em um bucket diferente do Amazon S3, seus usuários precisam de permissões adicionais. Para obter mais informações, consulte the section called "Configuração das permissões do bucket do Amazon S3".

É possível anexar a política `AmazonLookoutVisionConsoleFullAccess` a suas identidades do IAM.

Agrupamentos de permissões

Essa política é agrupada em declarações com base no conjunto de permissões fornecidas:

- `LookoutVisionFullAccess` — Permite o acesso para realizar todas as ações do Lookout for Vision.
- `LookoutVisionConsoleS3BucketSearchAccess` — Permite a listagem de todos os buckets do Amazon S3 de propriedade do chamador. O Lookout for Vision usa essa ação para identificar o bucket do console Lookout for Vision específico da região da AWS, se houver um na conta do chamador.
- `LookoutVisionConsoleS3BucketFirstUseSetupAccessPermissions` — Permite criar e configurar buckets do Amazon S3 que correspondam ao padrão de nome do bucket do console Lookout for Vision. O Lookout for Vision usa essas ações para criar e configurar um bucket de console Lookout for Vision específico da região quando não consegue encontrar um.
- `LookoutVisionConsoleS3BucketAccess` — Permite ações dependentes do Amazon S3 em buckets que correspondam ao padrão de nome de bucket do console Lookout for Vision. O Lookout for Vision é `s3:ListBucket` usado para pesquisar objetos de imagem

ao criar um conjunto de dados a partir de um bucket do Amazon S3 e ao iniciar uma tarefa de detecção experimental. O Lookout for Vision usa `s3:GetBucketLocation` e `s3:GetBucketVersioning` para validar a região da AWS, o proprietário e a configuração do bucket como parte do seguinte:

- Criação de um conjunto de dados
- Treinamento de um modelo
- Iniciando uma tarefa de detecção de testes
- Realizando feedback de detecção de testes

`LookoutVisionConsoleS3ObjectAccess` — Permite a leitura e gravação de objetos do Amazon S3 dentro de buckets que correspondem ao padrão de nome de bucket do Lookout for Vision Console. O Lookout for Vision usa essas ações para exibir imagens nas visualizações da galeria do console e fazer upload de novas imagens para uso em conjuntos de dados. Além disso, essas permissões permitem que o Lookout for Vision grave metadados enquanto cria um conjunto de dados, treina um modelo, inicia uma tarefa de detecção de teste e realiza feedback de detecção de teste.

- `LookoutVisionConsoleDatasetLabelingToolsAccess` — Permite ações de rotulagem dependentes do Amazon SageMaker GroundTruth. O Lookout for Vision usa essas ações para escanear buckets do S3 em busca de imagens, criar arquivos de manifesto do GroundTruth e anotar os resultados da tarefa de detecção de testes com rótulos de validação.
- `LookoutVisionConsoleDashboardAccess` - Permite a leitura das métricas do Amazon CloudWatch. O Lookout for Vision usa essas ações para preencher os gráficos do painel e as estatísticas detectadas de anomalias.
- `LookoutVisionConsoleTagSelectorAccess` — Permite a leitura de sugestões de chave e valor de tag específicas da conta. O Lookout for Vision usa essas permissões para fornecer recomendações para chaves e valores de tags nas páginas do console Manage tags.
- `LookoutVisionConsoleKmsKeySelectorAccess` — Permite listar chaves e aliases AWS Key Management Service (KMS). O Amazon Lookout for Vision usa essa permissão para preencher as chaves KMS na seleção de Tags sugeridas em determinadas ações do Lookout for Vision que oferecem suporte a chaves KMS gerenciadas pelo cliente para criptografia.

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

```
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketFirstUseSetupAccess",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketVersioning",
                "s3:PutLifecycleConfiguration",
                "s3:PutEncryptionConfiguration",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketAcl",
                "s3:GetBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
```

```
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDatasetLabelingToolsAccess",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:RunGenerateManifestByCrawlingJob",
                "groundtruthlabeling:AssociatePatchToManifestJob",
                "groundtruthlabeling:DescribeConsoleJob"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleTagSelectorAccess",
            "Effect": "Allow",
            "Action": [
                "tag:GetTagKeys",
                "tag:GetTagValues"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleKmsKeySelectorAccess",
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Política gerenciada pela: AmazonLookoutVisionConsoleReadOnlyAccess

Use a política `AmazonLookoutVisionConsoleReadOnlyAccess` para permitir que os usuários tenham acesso somente para leitura ao console do Amazon Lookout for Vision, às ações (operações do SDK) e a quaisquer dependências que o serviço tenha.

A política `AmazonLookoutVisionConsoleReadOnlyAccess` inclui permissões do Amazon S3 para o bucket do console do Amazon Lookout for Vision. Se as imagens do conjunto de dados ou os arquivos de manifesto do Amazon SageMaker Ground Truth estiverem em um bucket diferente do Amazon S3, seus usuários precisarão de permissões adicionais. Para obter mais informações, consulte the section called “Configuração das permissões do bucket do Amazon S3”.

É possível anexar a política `AmazonLookoutVisionConsoleReadOnlyAccess` a suas identidades do IAM.

Agrupamentos de permissões

Essa política é agrupada em declarações com base no conjunto de permissões fornecidas:

- `LookoutVisionReadOnlyAccess` — Permite o acesso para realizar ações do Lookout for Vision somente para leitura.
- `LookoutVisionConsoleS3BucketSearchAccess` — Permite a listagem de todos os buckets S3 de propriedade do chamador. O Lookout for Vision usa essa ação para identificar o bucket do console Lookout for Vision específico da região da AWS, se houver um na conta do chamador.
- `LookoutVisionConsoleS3ObjectReadAccess` — Permite ler objetos do Amazon S3 e versões de objetos do Amazon S3 nos buckets do console Lookout for Vision. O Lookout for Vision usa essas ações para exibir as imagens em conjuntos de dados, modelos e detecções de testes.
- `LookoutVisionConsoleDashboardAccess` — Permite ler as métricas do Amazon CloudWatch. O Lookout for Vision usa essas ações para preencher as estatísticas dos gráficos do painel e das anomalias detectadas.

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

```
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeTrialDetection",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListTrialDetections",
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectReadAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Fique atento às atualizações do Vision para as políticas gerenciadas da AWS

Veja detalhes sobre as atualizações das políticas gerenciadas da AWS para o Lookout for Vision desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações nesta página, assine o feed RSS na página de histórico do documento Lookout for Vision.

modelo adicionadas	adicionou as seguintes operações de empacotamento de modelos às políticas AmazonLookoutVisionFullAccess e AmazonLookoutVisionConsoleFullAccess: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob O Amazon Lookout for Vision adicionou as seguintes operações de empacotamento de modelos às políticas AmazonLookoutVisionReadOnlyAccess e AmazonLookoutVisionConsoleReadOnlyAccess: • DescribeModelPackagingJob • ListModelPackagingJobs	
Novas políticas adicionadas	O Amazon Lookout for Vision adicionou as seguintes políticas. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess	11 de maio de 2021
Lookout for Vision começou a monitorar as alterações	O Amazon Lookout for Vision começou a rastrear alterações	1º de março de 2021

Alteração	Descrição	Data
Operações de embalagem do modelo adicionadas	O Amazon Lookout for Vision adicionou as seguintes operações de empacotamento de modelos às políticas AmazonLookoutVisionFullAccess e AmazonLookoutVisionConsoleFullAccess: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob O Amazon Lookout for Vision adicionou as seguintes operações de empacotamento de modelos às políticas AmazonLookoutVisionReadOnlyAccess e AmazonLookoutVisionConsoleReadOnlyAccess: • DescribeModelPackagingJob • ListModelPackagingJobs	7 de dezembro de 2021
Novas políticas adicionadas	O Amazon Lookout for Vision adicionou as seguintes políticas. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess em suas políticas gerenciadas	11 de maio de 2021

Solução de problemas de identidade e acesso do Amazon Lookout for Vision

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Lookout for Vision e o IAM.

Tópicos

Não estou autorizado a realizar uma ação no Lookout for Vision

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `lookoutvision:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lookoutvision:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso *`my-example-widget`* usando a ação `lookoutvision:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Se você receber um erro informando que não está autorizado a executar a ação `iam:PassRole`, suas políticas deverão ser atualizadas para permitir que você passe uma função para o Lookout for Vision.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no Lookout for Vision. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas de fora da minha acessem meus Conta da AWS recursos do Lookout for Vision

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Lookout for Vision oferece suporte a esses recursos, consulte Como o Amazon Lookout for Vision funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Validação de conformidade do Amazon Lookout for Vision

Auditores terceirizados avaliam a segurança e a conformidade do Amazon Lookout for Vision como parte de AWS vários programas de conformidade. O Amazon Lookout for Vision é compatível com a Regulamentação Geral de Proteção de Dados (RGPD)

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos qualificados para a HIPAA.

> **Note**
>
> Nem todos Serviços da AWS são elegíveis para a HIPAA. Para obter mais informações, consulte a Referência dos serviços qualificados pela HIPAA.

- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Guias de conformidade do cliente — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).

- Avaliação de recursos com regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os atributos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.
- Amazon GuardDuty — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- AWS Audit Manager— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência no Amazon Lookout for Vision

A infraestrutura global da AWS é criada com base em regiões da AWS e zonas de disponibilidade. As regiões da AWS As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, que são conectadas com baixa latência, throughput elevada e redes altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre regiões e zonas de disponibilidade da AWS, consulte Infraestrutura global da AWS.

Segurança da infraestrutura no Amazon Lookout for Vision

Como um serviço gerenciado, o Amazon Lookout for Vision é protegido pela segurança de rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte Segurança na Nuvem AWS. Para projetar seu ambiente da

AWS usando as práticas recomendadas de segurança de infraestrutura, consulte Proteção de infraestrutura em Pilar segurança: AWS Well-Architected Framework.

Você usa as chamadas de API publicadas pela AWS para acessar o Lookout for Vision por meio da rede. Os clientes devem oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Monitoramento do Amazon Lookout for Vision

O monitoramento é uma parte importante da manutenção da confiabilidade, da disponibilidade e do desempenho do Amazon Lookout for Vision e de suas outras soluções AWS. A AWS fornece as seguintes ferramentas de monitoramento para observar o Lookout for Vision, informar quando algo está errado e tomar medidas automáticas quando apropriado:

- O Amazon CloudWatch monitora os recursos da AWS e os aplicativos que você executa na AWS em tempo real. É possível coletar e rastrear métricas, criar painéis personalizados e definir alarmes que o notificam ou que realizam ações quando uma métrica especificada atinge um limite definido. Por exemplo, você pode fazer o CloudWatch acompanhar o uso da CPU ou outras métricas das instâncias do Amazon EC2 e iniciar automaticamente novas instâncias quando necessário. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch.
- O Amazon CloudWatch Logs permite monitorar, armazenar e acessar seus arquivos de registro das instâncias do Amazon EC2, do CloudTrail e de outras fontes. O CloudWatch Logs pode monitorar informações nos arquivos de log e notificar você quando determinados limites forem atingidos. Você também pode arquivar seus dados de log em armazenamento resiliente. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch Logs.
- O Amazon EventBridge pode ser usado para automatizar seus serviços da AWS e responder automaticamente a eventos do sistema, como problemas de disponibilidade de aplicações ou alterações de recursos. Os eventos dos serviços da AWS são entregues ao EventBridge quase em tempo real. Você pode escrever regras simples para indicar quais eventos são do seu interesse, e as ações automatizadas a serem tomadas quando um evento corresponder à regra. Para obter mais informações, consulte o Guia do Usuário do Amazon EventBridge.
- O AWS CloudTrail captura chamadas de API e eventos relacionados feitos por ou em nome de sua conta da AWS e entrega os arquivos de log a um bucket do Amazon S3 que você especificar. Você pode identificar quais usuários e contas chamaram a AWS, o endereço IP de origem do qual as chamadas foram feitas e quando elas ocorreram. Para obter mais informações, consulte o Guia do usuário do AWS CloudTrail.

Monitoramento do Lookout for Metrics com o Amazon CloudWatch

Você pode monitorar o Lookout for Vision usando o CloudWatch, que coleta dados brutos e os processa em métricas legíveis e quase em tempo real. Essas estatísticas são mantidas por 15 meses, de maneira que você possa acessar informações históricas e ter uma perspectiva melhor de

como o aplicativo web ou o serviço está se saindo. Você também pode definir alarmes que observam determinados limites e enviam notificações ou realizam ações quando esses limites são atingidos. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch.

O serviço Lookout for Vision relata as métricas a seguir no namespace`AWS/LookoutVision`.

Métrica	Descrição
`DetectedAnomalyCount`	O número de anomalias detectadas em um projeto Estatística válida: `Sum,Average` Unidade: contagem
`ProcessedImageCount`	O número total de imagens executadas por meio da detecção de anomalias Estatística válida: `Sum,Average` Unidade: contagem
`InvalidImageCount`	O número de imagens que eram inválidas e não puderam retornar um resultado Estatística válida: `Sum,Average` Unidade: contagem
`SuccessfulRequestCount`	O número de chamadas bem-sucedidas da API . Estatística válida: `Sum,Average` Unidade: contagem
`ErrorCount`	A contagem do número de erros de API Estatística válida: `Sum,Average` Unidade: contagem
`ThrottledCount`	O número de erros de API causados pela limitação

Métrica	Descrição
	Estatística válida: `Sum,Average` Unidade: contagem
`Time`	O tempo em milissegundos para o Lookout for Vision calcular a detecção de anomalias Estatística válida: `Data Samples,Average` Unidades: milissegundos para `Average` estatísticas e contagem para `Data Samples` estatísticas
`MinInferenceUnits`	O número mínimo de unidades de inferência especificado durante a `StartModel` solicitação. Estatística válida: `Average` Unidade: contagem
`MaxInferenceUnits`	O número máximo de unidades de inferência especificadas durante a `StartModel` solicitação. Estatística válida: `Average` Unidade: contagem
`DesiredInferenceUnits`	O número de unidades de inferência para as quais o Lookout for Vision está aumentando ou diminuindo. Estatística válida: `Average` Unidade: contagem

Métrica	Descrição
`InServiceInferenceUnits`	O número de unidades de inferência que o modelo está usando. Estatística válida: `Average` É recomendável usar a estatística Média para obter a média de 1 minuto de quantas instâncias são usadas. Unidade: contagem

Há suporte para as dimensões a seguir para as métricas do Lookout for Vision.

Dimensão	Descrição
`ProjectName`	Você pode dividir as métricas por projeto para ver quais projetos estão com problemas ou precisam ser atualizados.
`ModelVersion`	Você pode dividir as métricas por versão do modelo para ver quais modelos estão com problemas ou precisam ser atualizados.

Registro de chamadas à API do Lookout for Vision com AWS CloudTrail

O Amazon Lookout for Vision está integrado ao AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou um serviço AWS no Lookout for Vision. O CloudTrail captura todas as chamadas de API do Lookout for Vision como eventos. As chamadas capturadas incluem chamadas do console do Lookout for Vision e chamadas de código para as operações da API do Lookout for Vision. Se você criar uma trilha, poderá ativar a entrega contínua de eventos do CloudTrail em um bucket do Amazon S3, incluindo eventos do Lookout for Vision. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Event history (Histórico de eventos). Usando as informações coletadas pelo CloudTrail, você

pode determinar a solicitação que foi feita ao Lookout for Vision, o endereço IP a partir do qual a solicitação foi feita, quem fez a solicitação, quando foi feita e outros detalhes.

Para saber mais sobre o CloudTrail, consulte o Guia do usuário do AWS CloudTrail.

Informações do AthloudTrail

O CloudTrail é habilitado em sua conta da AWS quando ela é criada. Quando ocorre uma atividade no Lookout for Vision, essa atividade é registrada em um evento do CloudTrail junto com outros eventos de serviço da AWS no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua conta da AWS. Para obter mais informações, consulte Como visualizar eventos com o histórico de eventos do CloudTrail.

Para obter um registro contínuo de eventos na sua conta AWS, incluindo eventos do Lookout for Vision, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra em log eventos de todas as regiões na partição da AWS e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, é possível configurar outros serviços da AWS para analisar mais ainda mais e agir com base nos dados de eventos coletados nos logs do CloudTrail. Para obter mais informações, consulte:

- Visão geral da criação de uma trilha
- Serviços e integrações compatíveis com o CloudTrail
- Configurar notificações do Amazon SNS para o CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e receber arquivos de log do CloudTrail de várias contas

Todas as ações do Athenas e documentadas na Referência de API do. Por exemplo, as chamadas para as ações `CreateProject`, `DetectAnomalies` e `StartModel` geram entradas nos arquivos de log do CloudTrail.

Se você monitorar o Amazon Lookout para chamadas de API do Vision, poderá ver chamadas para as seguintes APIs.

- LookoutVision: inicie a detecção experimental
- LookoutVision: Listar detecção de testes
- LookoutVision: descreva a detecção de ensaios

Essas chamadas especiais são usadas pelo Amazon Lookout for Vision para apoiar várias operações relacionadas à detecção de testes. Para obter mais informações, consulte Verificando seu modelo com uma tarefa de detecção de teste.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário da raiz ou do AWS Identity and Access Management.
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Para obter mais informações, consulte o Elemento userIdentity do CloudTrail.

Entendendo as entradas do arquivo de log do Lookout for Vision

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log a um bucket do Amazon S3 especificado. Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros de solicitação e assim por diante. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API pública. Dessa forma, eles não são exibidos em uma ordem específica.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra a ação `CreateDataset`.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAYN4CJAYDEXAMPLE:user",
    "arn": "arn:aws:sts::123456789012:assumed-role/Admin/MyUser",
    "accountId": "123456789012",
    "accessKeyId": "ASIAYN4CJAYEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAYN4CJAYDCGEXAMPLE",
```

```
        "arn": "arn:aws:iam::123456789012:role/Admin",
        "accountId": "123456789012",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-20T13:15:09Z"
      }
    }
  },
  "eventTime": "2020-11-20T13:15:43Z",
  "eventSource": "lookoutvision.amazonaws.com",
  "eventName": "CreateDataset",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "128.0.0.1",
  "userAgent": "aws-cli/3",
  "requestParameters": {
    "projectName": "P1",
    "datasetType": "train",
    "datasetSource": {
      "groundTruthManifest": {
        "s3Object": {
          "bucket": "myuser-bucketname",
          "key": "training.manifest"
        }
      }
    },
    "clientToken": "EXAMPLE-0526-47dd-a5d3-2ca975820a34"
  },
  "responseElements": {
    "status": "CREATE_IN_PROGRESS"
  },
  "requestID": "EXAMPLE-15e1-4bc9-be38-cda2537c75bf",
  "eventID": "EXAMPLE-c5e7-43e0-8449-8d9b87e15acb",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012"
}
```

Criação de recursos do Amazon Lookout for Vision com AWS CloudFormation

O Amazon Lookout for Vision é integrado ao AWS CloudFormation, um serviço que ajuda você a modelar e configurar seus recursos AWS para que você possa gastar menos tempo criando e gerenciando seus recursos e infraestrutura. Você cria um modelo que descreve todos os recursos AWS que deseja, e o AWS CloudFormation se encarrega de provisionar e configurar esses recursos para você.

Você pode usar AWS CloudFormation para provisionar e configurar projetos do Amazon Lookout for Vision.

Ao usar AWS CloudFormation, você pode reutilizar seu modelo para configurar seus projetos do Lookout for Vision de forma consistente e repetida. Basta descrever seus projetos uma vez e, em seguida, provisionar os mesmos projetos repetidamente em várias contas e regiões da AWS.

Registro de AWS CloudFormation

Para provisionar e configurar projetos para o Lookout for Vision e serviços relacionados, você deve entender os modelos AWS CloudFormation. Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os recursos que você deseja provisionar nas suas pilhas do AWS CloudFormation. Se você não estiver familiarizado com JSON ou YAML, poderá usar o AWS CloudFormation Designer para ajudá-lo a começar a usar os modelos do AWS CloudFormation. Para obter mais informações, consulte O que é o Designer? (O que é o AWS CloudFormation Designer) no Manual do usuário do AWS CloudFormation.

Para obter informações de referência sobre projetos do Lookout for Vision, incluindo exemplos de modelos JSON e YAML, consulte a referência do tipo de recurso LookoutVision.

Saiba mais sobre o AWS CloudFormation

Para saber mais sobre o AWS CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- Manual do usuário do AWS CloudFormation
- AWS CloudFormation Referência da API

- Guia do usuário da interface de linha de comando do AWS CloudFormation

Acesse o Amazon Lookout for Vision usando um endpoint de interface (AWS PrivateLink)

Você pode usar AWS PrivateLink para criar uma conexão privada entre seu VPC e o Amazon Lookout for Vision. Você pode acessar o Lookout for Vision como se ele estivesse em sua VPC, sem o uso de um gateway de Internet, dispositivo NAT, conexão VPN ou conexão AWS Direct Connect. As instâncias em sua VPC não precisam de endereços IP públicos para acessar o Lookout for Vision.

Você estabelece essa conexão privada criando um endpoint de interface, alimentado pelo AWS PrivateLink. Criaremos uma interface de rede de endpoint em cada sub-rede que você habilitar para o endpoint de interface. Essas são interfaces de rede gerenciadas pelo solicitante que servem como ponto de entrada para o tráfego destinado ao Lookout for Vision.

Para obter mais informações, consulte Acessar os Serviços da AWS pelo AWS PrivateLink no Guia do AWS PrivateLink.

Considerações sobre os pontos de extremidade do Lookout for Vision VPC

Antes de configurar um ponto de extremidade de interface para o Lookout for Vision, analise as Considerações Guia do AWS PrivateLink.

O Lookout for Vision oferece suporte a chamadas para todas as ações de API da interface.

As políticas de ponto de extremidade VPC não são compatíveis com o Lookout for Vision. Por padrão, o acesso total ao Lookout for Vision é permitido por meio do ponto de extremidade da interface. Como alternativa, você pode associar um grupo de segurança às interfaces de rede do endpoint para controlar o tráfego para o Lookout for Vision por meio do endpoint da interface.

Criação de um ponto de extremidade de VPC de interface para o Lookout for Vision

Você pode criar um ponto de extremidade de interface para o Lookout for Vision usando o console do Amazon VPC ou a AWS Command Line Interface (AWS CLI). Para mais informações, consulte Criar um endpoint de interface no Guia do usuário do AWS PrivateLink.

Crie um endpoint de interface para o Lookout for Vision usando o seguinte nome de serviço:

```
com.amazonaws.region.lookoutvision
```

Se você ativar o DNS privado para o ponto de extremidade da interface, poderá fazer solicitações de API ao Lookout for Vision usando seu nome DNS regional padrão. Por exemplo, `lookoutvision.us-east-1.amazonaws.com`.

Criar uma política de VPC endpoint para o

Uma política de endpoint é um recurso do IAM que você pode anexar a um endpoint de interface. A política de endpoint padrão permite acesso total ao Lookout for Vision por meio do endpoint da interface. Para controlar o acesso permitido ao Lookout for Vision a partir de sua VPC, anexe uma política de endpoint personalizada ao endpoint da interface.

Uma política de endpoint especifica as seguintes informações:

- Os diretores que podem realizar ações (Contas da AWSusuários do IAM e funções do IAM).
- As ações que podem ser executadas.
- O recurso no qual as ações podem ser executadas.

Para obter mais informações, consulte Control access to services using endpoint policies (Controlar o acesso a serviços usando políticas de endpoint) no Guia do AWS PrivateLink.

Exemplo: Política de endpoint VPC para ações do Lookout for Vision

A seguir, um exemplo de uma política de endpoint personalizada para o Lookout for Vision. Quando você anexa essa política ao seu endpoint de interface, ela especifica que todos os usuários que têm acesso ao endpoint da interface VPC têm permissão para chamar a operação de `DetectAnomalies` API para o modelo Lookout for Vision associado ao projeto. `myModel` `myProject`

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
```

```
            "lookoutvision:DetectAnomalies"
         ],
         "Resource": "arn:aws:lookoutvision:us-west-2:123456789012:model/myProject/
myModel"
      }
   ]
}
```

Cotas na Amazônia em busca de visão

As tabelas a seguir descrevem as cotas atuais no Amazon Lookout for Vision. Para obter mais informações sobre cotas que podem ser alteradas, consulte Cotas de serviço da AWS.

Cotas modelo

As cotas a seguir se aplicam ao teste, treinamento e funcionalidade de um modelo.

Recurso	Quota
Formato de arquivo compatível	Formatos de imagem PNG e JPEG
Dimensão mínima de imagem do arquivo de imagem em um bucket do Amazon S3	64 pixels x 64 pixels
Dimensão máxima de imagem do arquivo de imagem em um bucket do Amazon S3	4096 pixels X 4096 pixels é o máximo. Dimensões menores podem ser carregadas mais rapidamente.
Diferentes dimensões de imagem dos arquivos de imagem usados em um projeto	Todas as imagens no conjunto de dados devem ter as mesmas dimensões
Tamanho máximo de arquivo para uma imagem em um bucket do Amazon S3	8 MB
Falta de rótulos	As imagens devem ser rotuladas como normais ou anomalias antes do treinamento. Imagens sem rótulos são ignoradas durante o treinamento.
Número mínimo de imagens rotuladas como normais no conjunto de dados de treinamento	10 para um projeto com conjuntos de dados de treinamento e teste separados. 20 para um projeto com um único conjunto de dados.
Número mínimo de imagens rotuladas como anomalia em um conjunto de dados de treinamento	0 para um projeto com conjuntos de dados de treinamento e teste separados. 10 para um projeto com um único conjunto de dados.

Recurso	Quota
Número máximo de imagens no conjunto de dados de treinamento de classificação	16.000
Número máximo de imagens em um conjunto de dados de teste de classificação	4.000
Número mínimo de imagens rotuladas como normais no conjunto de dados de teste	10
Número mínimo de imagens rotuladas como anomalia no conjunto de dados de teste	10
Número máximo de imagens em um conjunto de dados de treinamento de localização de anomalias	8000
Número máximo de imagens em um conjunto de dados de teste de localização de anomalias	800
Número máximo de imagens no conjunto de dados de detecção de ensaios	2.000
Tamanho máximo do arquivo de manifesto do conjunto de dados	1 GB
Número máximo de conjuntos de dados de treinamento em um modelo	1
Tempo máximo de treinamento	24 horas
Tempo máximo de teste	24 horas
Número máximo de rótulos de anomalias em um projeto	100
Número máximo de rótulos de anomalia em uma imagem de máscara	20

Recurso	Quota
Número mínimo de imagens para uma etiqueta de anomalia. Para contar, a imagem deve conter somente um tipo de rótulo de anomalia.	20 para um único projeto de conjunto de dados. 10 para cada conjunto de dados em um projeto com conjuntos de dados de treinamento e teste separados.

Histórico do documento do Amazon Lookout for Vision

A tabela a seguir descreve alterações importantes em cada versão do Guia do Desenvolvedor do Amazon Lookout for Vision. Para receber notificações sobre atualizações dessa documentação, você pode se inscrever em um feed RSS.

- Última atualização da documentação: 20 de fevereiro de 2023

Alteração	Descrição	Data
Foi adicionado um exemplo de função Lambda	Exemplo mostrando como encontrar anomalias com uma AWS Lambda função. Para obter mais informações, consulte Encontrar anomalias com uma função do AWS Lambda.	20 de fevereiro de 2023
Atualização da orientação para IAMAWS WAF	Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM.	8 de fevereiro de 2023
Exemplo de exportação de conjunto de dados adicionado	Foi adicionado um exemplo em Python mostrando como usar a `ListDatasetEntries` operação para exportar os conjuntos de dados de um projeto Amazon Lookout for Vision. Para obter mais informações, consulte Exportação de conjuntos de dados de um projeto (SDK).	2 de dezembro de 2022

Tópico de conceitos básicos atualizado	Introdução atualizada para mostrar a criação de um modelo de segmentação de imagens com um exemplo de conjunto de dados. Para obter mais informações, consulte Introdução ao Amazon Lookout for Vision.	20 de outubro de 2022
Adição do tópico de solução de problemas	Foi adicionado o tópico de solução de problemas do treinamento de modelos. Para obter mais informaçõ es, consulte Solução de problemas de modelos de.	17 de outubro de 2022
Tópico adicionado sobre o uso de trabalhos do Amazon SageMaker Ground Truth	Em vez de rotular imagens você mesmo, você pode usar as tarefas do Amazon SageMaker Ground Truth para rotular imagens para modelos de classificação e segmentaç ão de imagens. Para obter mais informações, consulte Usar um trabalho do Amazon SageMaker Ground Truth.	19 de agosto de 2022

O Amazon Lookout for Vision agora fornece localização de anomalias.	Você pode criar um modelo de segmentação que encontre os locais em uma imagem em que diferentes tipos de anomalias (como arranhões, amassados ou rasgos) estão presentes, o rótulo da anomalia e o tamanho da anomalia. Para obter mais informações, consulte Executando seu modelo treinado do Amazon Lookout for Vision.	16 de agosto de 2022
O Amazon Lookout for Vision agora fornece inferência de CPU em dispositivos periféricos.	Agora, os modelos do Amazon Lookout for Vision podem ser implantados para executar inferência localmente em uma plataforma computacional x86 executando Linux com apenas uma CPU, sem precisar de um acelerador de GPU. Para obter mais informações, consulte Acelerador de CPU.	16 de agosto de 2022
Agora, o Amazon Lookout for Vision pode escalar automaticamente as unidades de inferência.	Para ajudar com picos na demanda, o Amazon Lookout for Vision agora pode escalar o número de unidades de inferência que seu modelo usa. Para obter mais informações, consulte Executando seu modelo treinado do Amazon Lookout for Vision.	16 de agosto de 2022

Exemplos de Java adicionados	O guia do desenvolvedor do Amazon Lookout for Vision agora inclui exemplos de Java. Para obter mais informações, consulte Introdução ao SDK AWS.	2 de maio de 2022
Disponibilidade geral da implantação do modelo em um dispositivo de ponta	A implantação do modelo em um dispositivo de ponta gerenciado por agora AWS IoT Greengrass Version 2 está disponível ao público em geral. Para obter mais informações, consulte Usando seu modelo Amazon Lookout for Vision em um dispositivo de ponta.	14 de março de 2022
Informações atualizadas do bucket do console	Informações atualizadas sobre o conteúdo do bucket do console e abordagens alternativas para criar o bucket do console. Para obter mais informações, consulte Etapa 4: criar o bucket do console.	7 de março de 2022
Criar um arquivo de manifesto a partir de um arquivo CSV	Agora você pode simplificar a criação de um arquivo de manifesto usando um script que lê as informações de classificação de um arquivo CSV. Para obter mais informações, consulte Como criar um arquivo de manifesto de um arquivo CSV.	10 de fevereiro de 2022

Versão prévia da implantação do modelo em um dispositivo de ponta	A versão prévia da implantação do modelo em um dispositivo de ponta gerenciado por já AWS IoT Greengrass Version 2 está disponível. Para obter mais informações, consulte Usando seu modelo Amazon Lookout for Vision em um dispositivo de ponta.	7 de dezembro de 2021
Novos exemplos de Python e Java 2 adicionados	Foram adicionados exemplos de Python e Java 2 para analisar imagens com. `DetectAnomalies` Para obter mais informações, consulte Detectar anomalias em uma imagem.	7 de setembro de 2021
Novas políticas gerenciadas da AWS adicionadas.	O Amazon Lookout for Vision adiciona suporte AWS para políticas gerenciadas. Para obter mais informações, consulte as políticas AWS gerenciadas do Amazon Lookout for Vision.	11 de maio de 2021
Informações atualizadas da unidade de inferência.	Foram adicionadas informações que descrevem as unidades de inferência e como elas são carregadas. Para obter mais informações, consulte Executando seu modelo treinado do Amazon Lookout for Vision.	15 de março de 2021

Disponibilidade geral do Amazon Lookout for Vision.	O Amazon Lookout for Vision agora está disponível ao público em geral. Exemplos de código Python atualizados para lidar com tarefas assíncronas, como treinar um modelo.	17 de fevereiro de 2021
Marcação e AWS CloudFormation suporte adicionados.	Agora você pode marcar modelos do Amazon Lookout for Vision e criar projetos AWS CloudFormation com. Para obter mais informações, consulte Como marcar modelos e Criar projetos do Amazon Lookout for Vision com o AWS CloudFormation.	31 de janeiro de 2021
Novo recurso e guia	Este é o lançamento inicial do serviço Amazon Lookout for Vision. Guia do desenvolvedor do Amazon Lookout for Vision.	1º de dezembro de 2020

Glossário do AWS

Para obter a terminologia mais recente da AWS, consulte o glossário da AWS na Referência do Glossário da AWS.

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.