AI 620 Emerging Topics in Artificial Intelligence

HOS10A Image Classification Using Amazon SageMaker

03/21/2023 Developed by Yared Shewarade

09/23/2024 Updated by Anh Nguyen

12/3/2024 Reviewed by Jonathan Koerber

School of Technology and Computing (STC) @City University of Seattle (CityU)

Before You Start

- The directory path shown in screenshots may be different from yours.
- Some steps are not explained in the tutorial. If you are not sure what to do:
    1. Consult the resources listed below.
    2. If you cannot solve the problem after a few tries, ask a student worker for help.

Learning Outcomes

Students will be able to learn:

- Introduction to Image Classification
- Image Classification setup using Amazon SageMaker

Resources

- Tripuraneni, S., & Song, C. (2019). *Hands-on artificial intelligence on amazon web services: Decrease the time to market for AI and ML applications with the power of AWS* (1st ed.). Packt.

Introduction to Neural Topic Model (NTM)

Image classification has been one of the leading research fields to successfully classify images and solve many business problems across a variety of industries. For example, the entire autonomous vehicle industry is dependent on the accuracy of these image classification and object detection models. Amazon SageMaker image classification algorithm is a supervised learning algorithm that supports multi-label classification. It takes an image as input and outputs one or more labels assigned to that image. It uses a convolutional neural network that can be trained from scratch or trained using transfer learning when a large number of training images are not available.

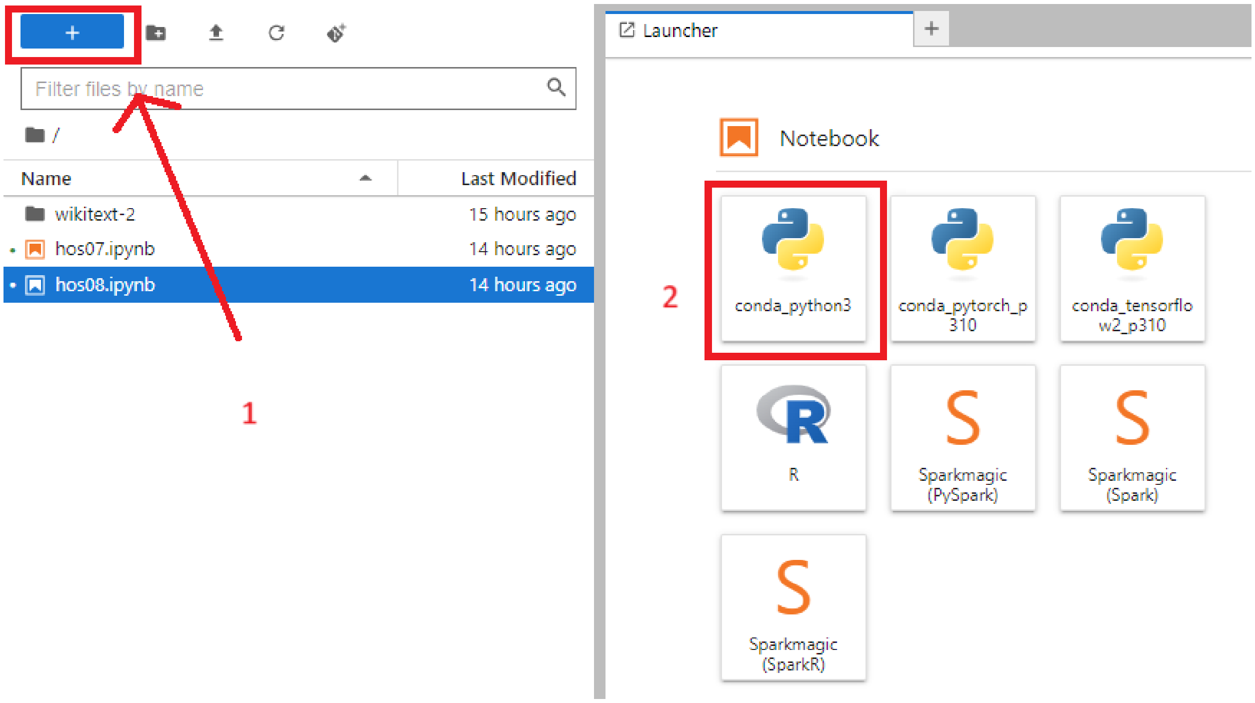
The recommended input format for the Amazon SageMaker image classification algorithms is Apache MXNet RecordIO. However, we can also use raw images in .jpg or .png format.

Image classification using Amazon SageMaker

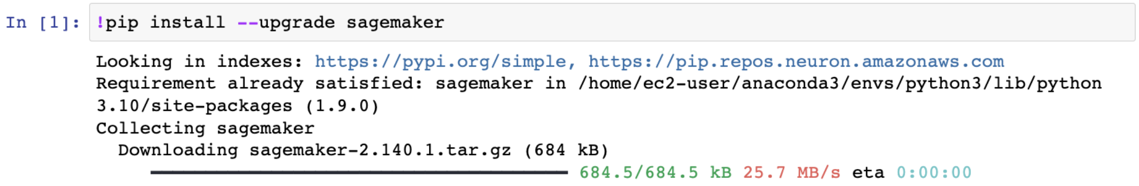
Note: For submission, take the screenshot for all steps and save it in your local repository along with your code.

A. Install the required module and prepare the data

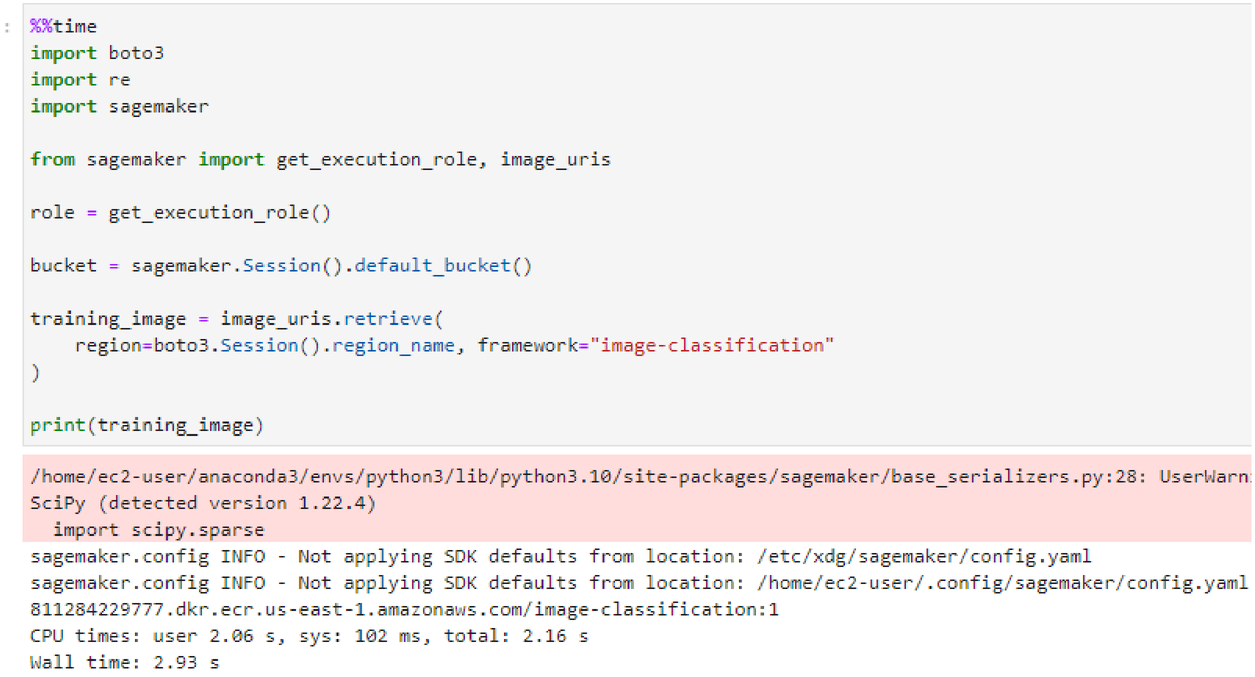
Open the notebook instance you created in HOS07 > Open JupyterLab > New Launcher (+) > conda_python3



Name	Last Modified
wikitext-2	15 hours ago
hos07.ipynb	14 hours ago
hos08.ipynb	14 hours ago



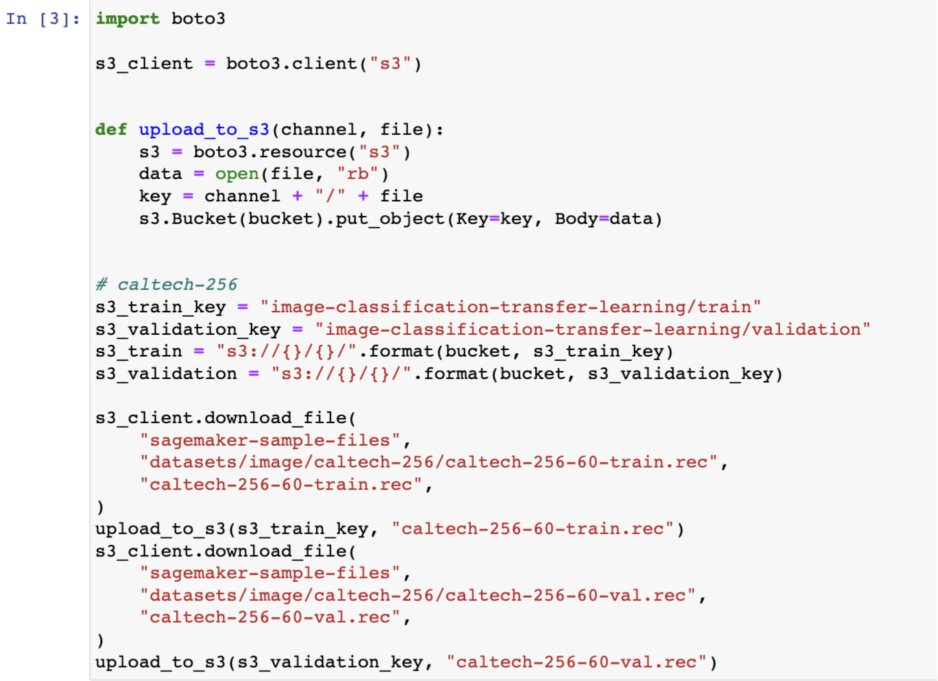
```
In [1]: !pip install --upgrade sagemaker

        Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
        Requirement already satisfied: sagemaker in /home/ec2-user/anaconda3/envs/python3/lib/python
        3.10/site-packages (1.9.0)
        Collecting sagemaker
          Downloading sagemaker-2.140.1.tar.gz (684 kB)
             ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 684.5/684.5 kB 25.7 MB/s eta 0:00:00
```



```
%%time
import boto3
import re
import sagemaker

from sagemaker import get_execution_role, image_uris

role = get_execution_role()

bucket = sagemaker.Session().default_bucket()

training_image = image_uris.retrieve(
    region=boto3.Session().region_name, framework="image-classification"
)

print(training_image)
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagemaker/base_serializers.py:28: UserWarn
SciPy (detected version 1.22.4)
  import scipy.sparse
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
811284229777.dkr.ecr.us-east-1.amazonaws.com/image-classification:1
CPU times: user 2.06 s, sys: 102 ms, total: 2.16 s
Wall time: 2.93 s
```

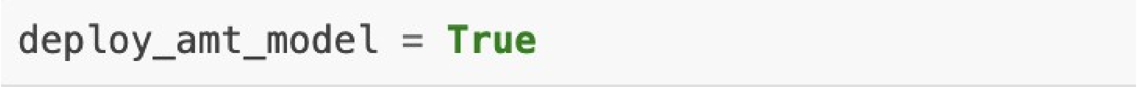
Let’s download the data and transfer them to S3.



```
In [3]: import boto3

        s3_client = boto3.client("s3")


        def upload_to_s3(channel, file):
            s3 = boto3.resource("s3")
            data = open(file, "rb")
            key = channel + "/" + file
            s3.Bucket(bucket).put_object(Key=key, Body=data)


        # caltech-256
        s3_train_key = "image-classification-transfer-learning/train"
        s3_validation_key = "image-classification-transfer-learning/validation"
        s3_train = "s3://{}/{}/".format(bucket, s3_train_key)
        s3_validation = "s3://{}/{}/".format(bucket, s3_validation_key)

        s3_client.download_file(
            "sagemaker-sample-files",
            "datasets/image/caltech-256/caltech-256-60-train.rec",
            "caltech-256-60-train.rec",
        )
        upload_to_s3(s3_train_key, "caltech-256-60-train.rec")
        s3_client.download_file(
            "sagemaker-sample-files",
            "datasets/image/caltech-256/caltech-256-60-val.rec",
            "caltech-256-60-val.rec",
        )
        upload_to_s3(s3_validation_key, "caltech-256-60-val.rec")
```

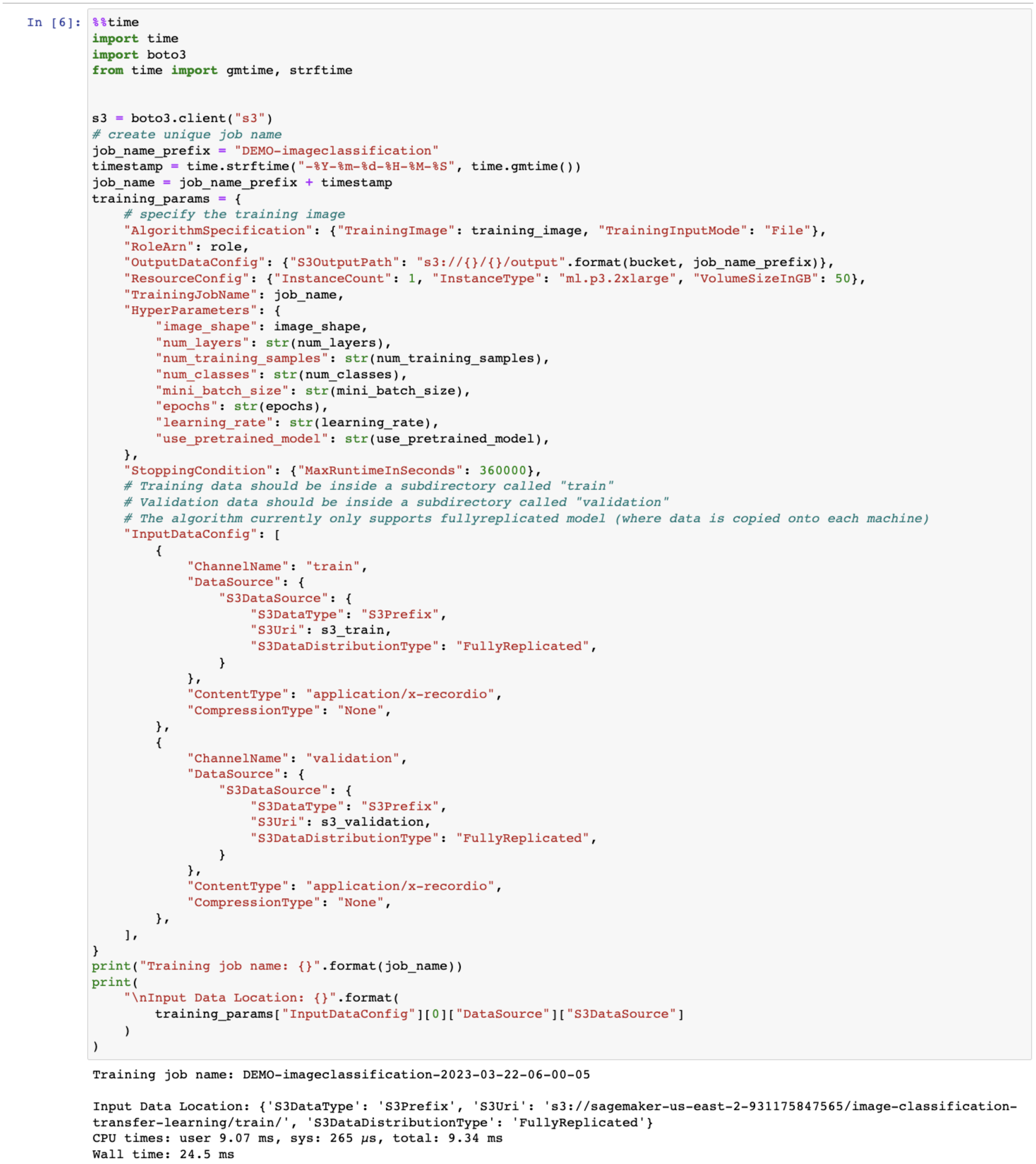
B. Training the model

Let’s train the model once we have the data available in the correct format for training.



```
deploy_amt_model = True
```

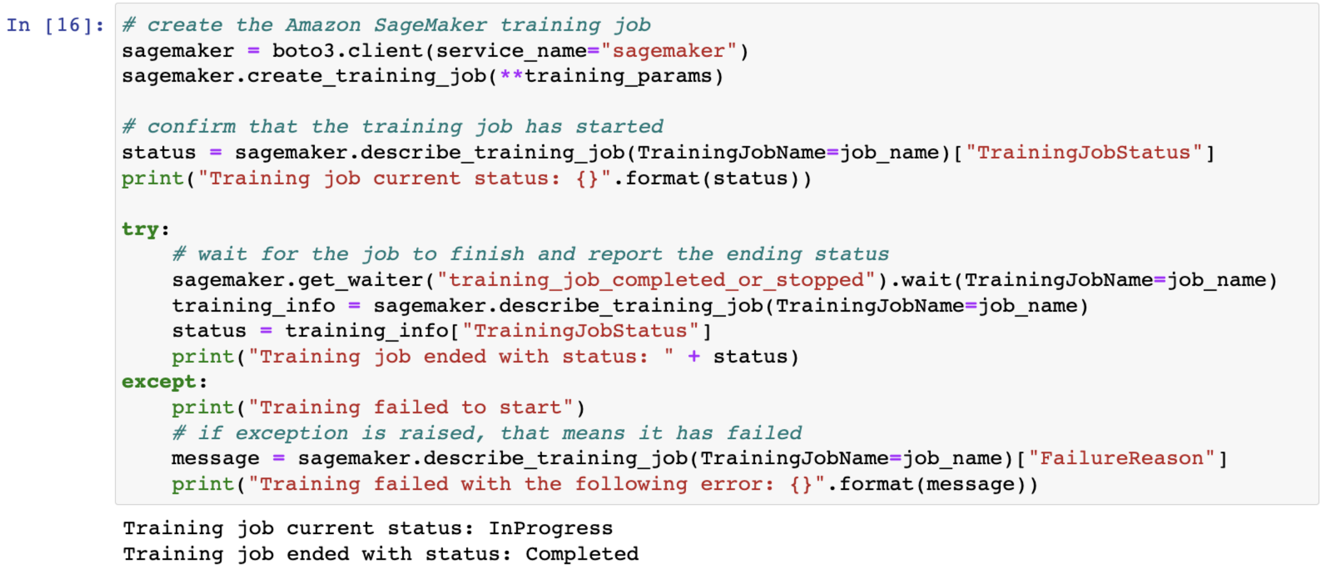
```
In [5]: # The algorithm supports multiple network depth (number of layers). They are 18, 34, 50, 101, 152 and 200
        # For this training, we will use 18 layers
        num_layers = 18
        # we need to specify the input image shape for the training data
        image_shape = "3,224,224"
        # we also need to specify the number of training samples in the training set
        # for caltech it is 15420
        num_training_samples = 15420
        # specify the number of output classes
        num_classes = 257
        # batch size for training
        mini_batch_size = 128
        # number of epochs
        epochs = 2
        # learning rate
        learning_rate = 0.01
        top_k = 2
        # Since we are using transfer learning, we set use_pretrained_model to 1 so that weights can be
        # initialized with pre-trained weights
        use_pretrained_model = 1
```



In [6]:

```
%%time
import time
import boto3
from time import gmtime, strftime


s3 = boto3.client("s3")
# create unique job name
job_name_prefix = "DEMO-imageclassification"
timestamp = time.strftime("-%Y-%m-%d-%H-%M-%S", time.gmtime())
job_name = job_name_prefix + timestamp
training_params = {
    # specify the training image
    "AlgorithmSpecification": {"TrainingImage": training_image, "TrainingInputMode": "File"},
    "RoleArn": role,
    "OutputDataConfig": {"S3OutputPath": "s3://{}/{}/output".format(bucket, job_name_prefix)},
    "ResourceConfig": {"InstanceCount": 1, "InstanceType": "ml.p3.2xlarge", "VolumeSizeInGB": 50},
    "TrainingJobName": job_name,
    "HyperParameters": {
        "image_shape": image_shape,
        "num_layers": str(num_layers),
        "num_training_samples": str(num_training_samples),
        "num_classes": str(num_classes),
        "mini_batch_size": str(mini_batch_size),
        "epochs": str(epochs),
        "learning_rate": str(learning_rate),
        "use_pretrained_model": str(use_pretrained_model),
    },
    "StoppingCondition": {"MaxRuntimeInSeconds": 360000},
    # Training data should be inside a subdirectory called "train"
    # Validation data should be inside a subdirectory called "validation"
    # The algorithm currently only supports fullyreplicated model (where data is copied onto each machine)
    "InputDataConfig": [
        {
            "ChannelName": "train",
            "DataSource": {
                "S3DataSource": {
                    "S3DataType": "S3Prefix",
                    "S3Uri": s3_train,
                    "S3DataDistributionType": "FullyReplicated",
                }
            },
            "ContentType": "application/x-recordio",
            "CompressionType": "None",
        },
        {
            "ChannelName": "validation",
            "DataSource": {
                "S3DataSource": {
                    "S3DataType": "S3Prefix",
                    "S3Uri": s3_validation,
                    "S3DataDistributionType": "FullyReplicated",
                }
            },
            "ContentType": "application/x-recordio",
            "CompressionType": "None",
        },
    ],
}
print("Training job name: {}".format(job_name))
print(
    "\nInput Data Location: {}".format(
        training_params["InputDataConfig"][0]["DataSource"]["S3DataSource"]
    )
)
```

```
Training job name: DEMO-imageclassification-2023-03-22-06-00-05

Input Data Location: {'S3DataType': 'S3Prefix', 'S3Uri': 's3://sagemaker-us-east-2-931175847565/image-classification-
transfer-learning/train/', 'S3DataDistributionType': 'FullyReplicated'}
CPU times: user 9.07 ms, sys: 265 µs, total: 9.34 ms
Wall time: 24.5 ms
```



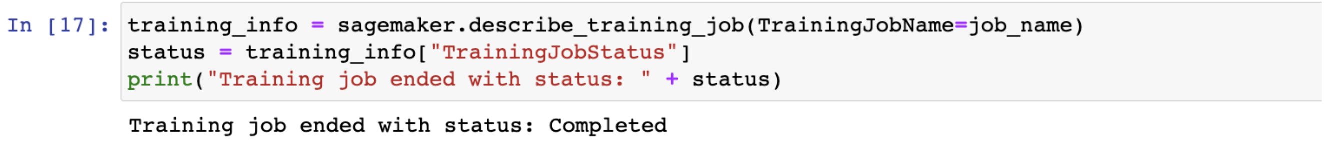
In [16]:

```
# create the Amazon SageMaker training job
sagemaker = boto3.client(service_name="sagemaker")
sagemaker.create_training_job(**training_params)

# confirm that the training job has started
status = sagemaker.describe_training_job(TrainingJobName=job_name)["TrainingJobStatus"]
print("Training job current status: {}".format(status))

try:
    # wait for the job to finish and report the ending status
    sagemaker.get_waiter("training_job_completed_or_stopped").wait(TrainingJobName=job_name)
    training_info = sagemaker.describe_training_job(TrainingJobName=job_name)
    status = training_info["TrainingJobStatus"]
    print("Training job ended with status: " + status)
except:
    print("Training failed to start")
    # if exception is raised, that means it has failed
    message = sagemaker.describe_training_job(TrainingJobName=job_name)["FailureReason"]
    print("Training failed with the following error: {}".format(message))
```

```
Training job current status: InProgress
Training job ended with status: Completed
```

If we see the message “Training job current status: Completed”, it indicates that the training is successfully completed.

Let’s print out the status of the training



In [17]:

```
training_info = sagemaker.describe_training_job(TrainingJobName=job_name)
status = training_info["TrainingJobStatus"]
print("Training job ended with status: " + status)
```

```
Training job ended with status: Completed
```

HOS submission instructions:

1. Please install the GitHub Desktop: https://cityuseattle.github.io/docs/git/github_desktop/

2. Clone, organize, and submit your work through GitHub Desktop: https://cityuseattle.github.io/docs/hoporhos