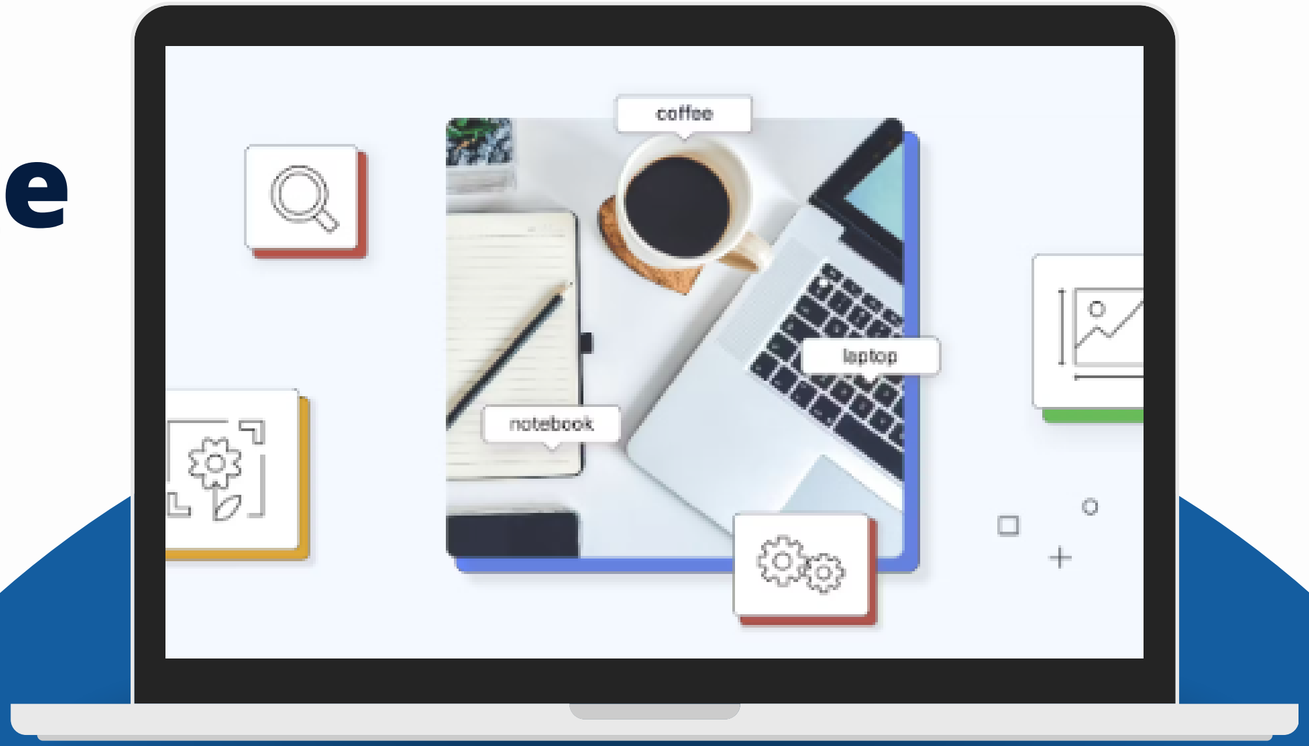
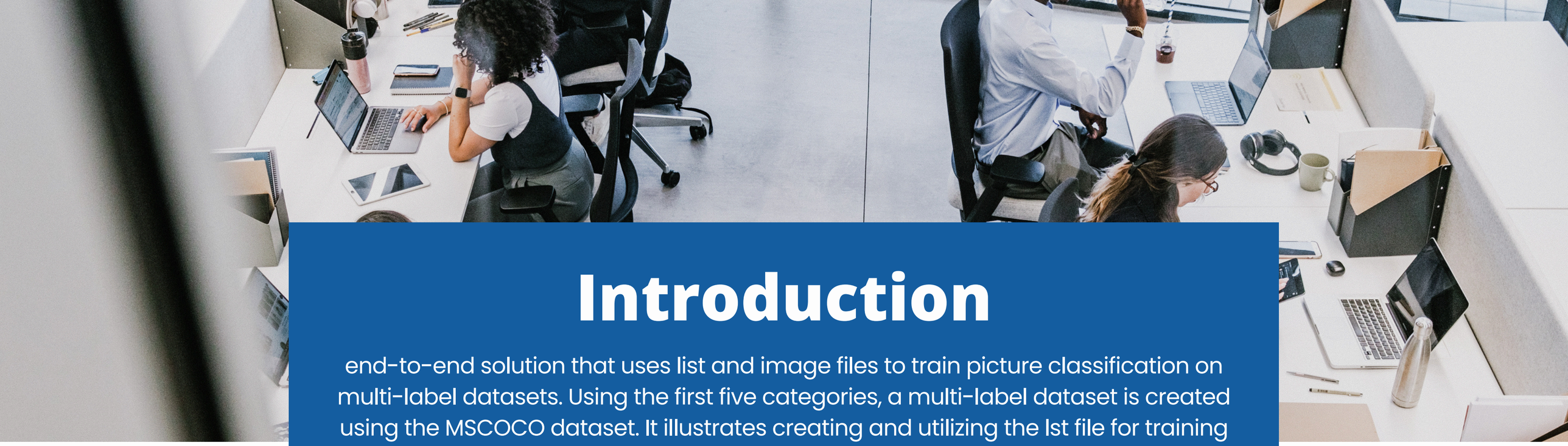
AITS

Automated Image Tagging System

By: AITS Team



Overview



Introduction

end-to-end solution that uses list and image files to train picture classification on multi-label datasets. Using the first five categories, a multi-label dataset is created using the MSCOCO dataset. It illustrates creating and utilizing the lst file for training with multi-label datasets. The network resets the fully connected layer, which also retrains it and fine-tunes the other levels. The network displays both the use of the use_weighted_loss and multi_label parameters. In the training process, several additional parameters can be changed, including batch size, learning rate, network depth (number of layers), etc. The notebook demonstrates how to host the learned model for inference when training is finished.

Used Tools

S3

Upload the data onto the s3 bucket. The images are uploaded onto train and validation bucket. The lst files are uploaded to train_lst and validation_lst folders

SageMaker

Sagemaker image classification algorithm supports application/x-recordio format which can be used for larger datasets.

Time Line

WEEK 1

Data Collection and Preparation

WEEK 2

Model Training

WEEK 3

Pipeline Integration

WEEK 4

MLOps and Final presentation

Data Collection and Preparation

```python
from pycocotools.coco import COCO
import numpy as np
import os

annFile='./annotations/instances_val2017.json'
coco=COCO(annFile)

catIds = coco.getCatIds()
image_ids_of_cats = []
for cat in catIds:
    image_ids_of_cats.append(coco.getImgIds(catIds=cat))

image_ids = []
labels = []
# use only the first 5 classes
# obtain image ids and labels for images with these 5 classes
cats = [1, 2, 3, 4, 5]
for ind_cat in cats:
    for image_id in image_ids_of_cats[ind_cat-1]:
        if image_id in image_ids:
            labels[image_ids.index(image_id)][ind_cat-1] = 1
        else:
            image_ids.append(image_id)
            labels.append(np.zeros(len(cats), dtype=np.int))
            labels[-1][ind_cat-1] = 1
```

MS COCO is a large-scale dataset for multiple computer vision tasks, including object detection, segmentation, and captioning. In this notebook, we will use the object detection dataset to construct the multi-label dataset for classification. We will use the 2017 validation set from MS-COCO dataset to train multi-label classifier. MS-COCO dataset consist of images from 80 categories. We will choose 5 categories out of 80 and train the model to learn to classify these 5 categories.

Model Training

```
{}/output'.format(bucket, prefix)
mator.Estimator(training_image,
            role,
            train_instance_count=1,
            train_instance_type='ml.p2.xlarge',
            train_volume_size = 50,
            train_max_run = 360000,
            input_mode= 'File',
            output_path=s3_output_location,
            sagemaker_session=sess)
```

Training parameters

There are two kinds of parameters that need to be set for training. The first one are the parameters for the training job. These include:

- **Training instance count**: This is the number of instances on which to run the training. When the number of instances is greater than one, then the image classification algorithm will run in distributed settings.
- **Training instance type**: This indicates the type of machine on which to run the training. Typically, we use GPU instances for these training
- **Output path**: This the s3 folder in which the training output is stored

Pipeline Integration

```
multilabel_ic.set_hyperparameters(num_layers=18,
                                  use_pretrained_model=1,
                                  image_shape = "3,224,224",
                                  num_classes=5,
                                  mini_batch_size=128,
                                  resize=256,
                                  epochs=5,
                                  learning_rate=0.001,
                                  num_training_samples=2500,
                                  use_weighted_loss=1,
                                  augmentation_type = 'crop_col
                                  precision_dtype='float32',
                                  multi_label=1)
```

Apart from the above set of parameters, there are hyperparameters that are specific to the algorithm. These are:

- **num_layers**: The number of layers (depth) for the network. We use 18 in this samples but other values such as 50, 152 can be used.
- **use_pretrained_model**: Set to 1 to use pretrained model for transfer learning.
- **image_shape**: The input image dimensions,'num_channels, height, width', for the network. It should be no larger than the actual image size. The number of channels should be same as the actual image.
- **num_classes**: This is the number of output classes for the dataset. We use 5 classes from MSCOCO and hence it is set to 5

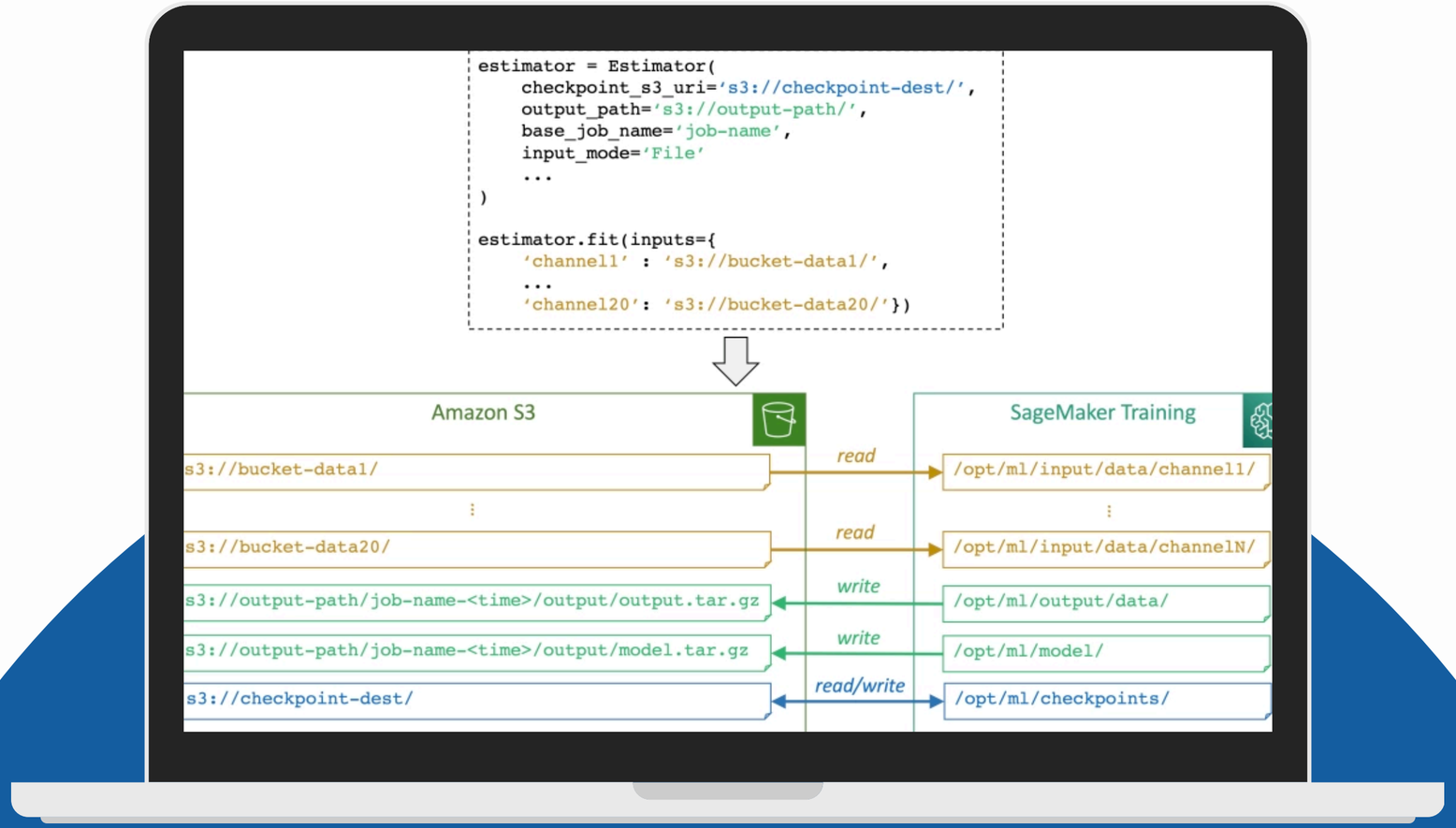
MLOps and Final presentation

```
import json

with open(file_name, 'rb') as image:
    f = image.read()
    b = bytearray(f)
ic_classifier.content_type = 'application/x-imag
results = ic_classifier.predict(b)
prob = json.loads(results)
classes = ['Person', 'Bicycle', 'Car', 'Motorcyc
for idx, val in enumerate(classes):
    print('%s:%f '%(classes[idx], prob[idx]), en
```

Evaluate the image through the network for inference. The network outputs class probabilities for all the classes. As can be seen from this example, the network output is pretty good even with training for only 5 epochs

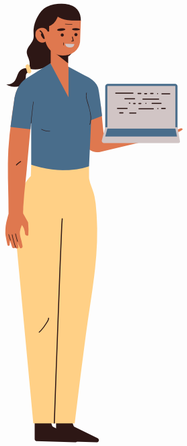
S3 And SageMaker Illustration



Our Team

Faris Hesham
Developer

Kareem Mohamed
Developer



Radwa Sayed
Developer

Noorhan Hamed
Developer

Michel Youssef
Developer

THANK YOU!

AITS Team