

# Dog Breed Classifier Project

## - Sancharika Debnath

### Image Classification using AWS SageMaker

This project takes a pretrained CNN model (Resnet18) and fine tunes it for use in classifying dog breeds into 133 classification based on a Dog Breeds dataset.

The steps that the notebook goes through are:

1. Retrieving a dataset
2. Unzipping that dataset.
3. Uploading that dataset to an S3 bucket.
4. Setting up hyperparameter tuning using learning rate, weight decay, eps and batch size using the AdamW optimizer on ResNet18.
5. Starting a hyperparameter tuning job using 20 training jobs (2 at a time) with auto shutdown on the AWS SageMaker Hyperparameter Tuning system.
6. Record the best hyperparameters as discovered from the above.
7. Train a fine-tuned model using ResNet18 and the best hyperparameters over a larger number of epochs, recording profiling and debug data.
8. Examine the output from the profiling and debug of the above.
9. Re-run the training with profiling and debug turned off (due to issues deploying model trained with the above).
10. Deploy that model as an endpoint on AWS.
11. Test that endpoint with new images.

## Hyperparameter Tuning

The ResNet18 model with a single layer Linear NN is used. ResNet18 is a well proven pretrained CNN and the results of a single layer NN was enough to get good results from training. The hyperparameters used to tune were:

1. Learning rate
2. epochs
3. Weight decay
4. Batch size

These hyperparameters gave broad coverage of the tune for the system. Values used between 0.1x and 10x the default as the default values tend to give good results for the first 3, and for batch size -the options of 32 and 64.

## Results

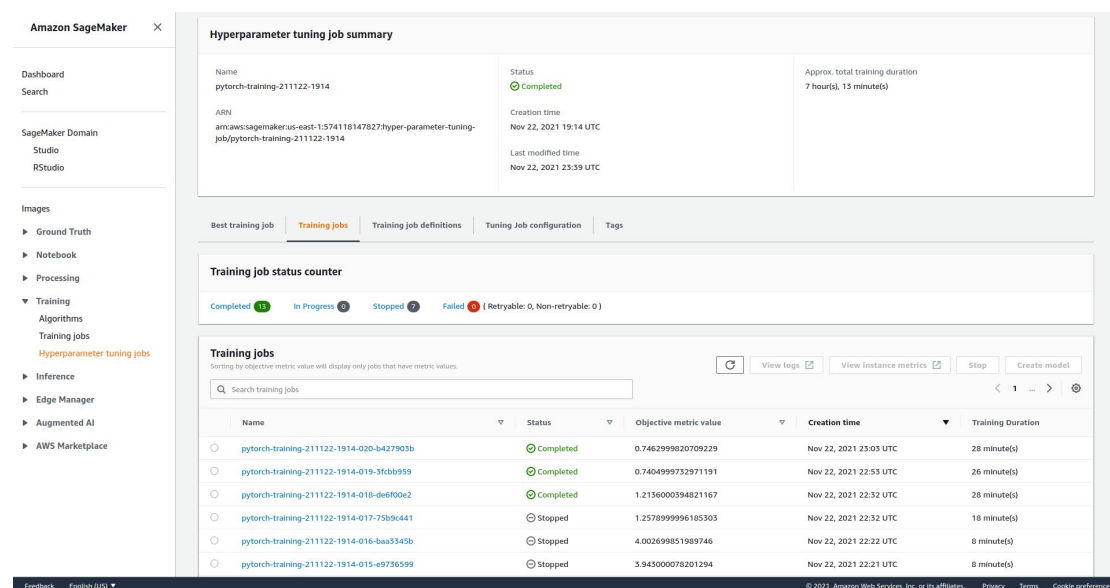


Figure 1: Hyperparameter Training Job

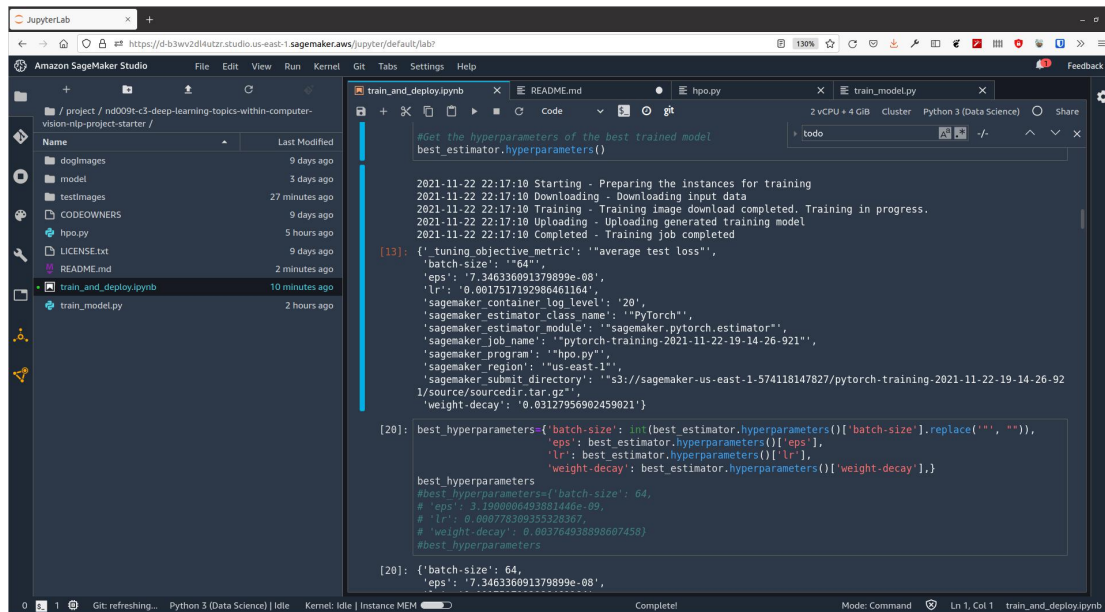


Figure 2: Hyperparameters best results

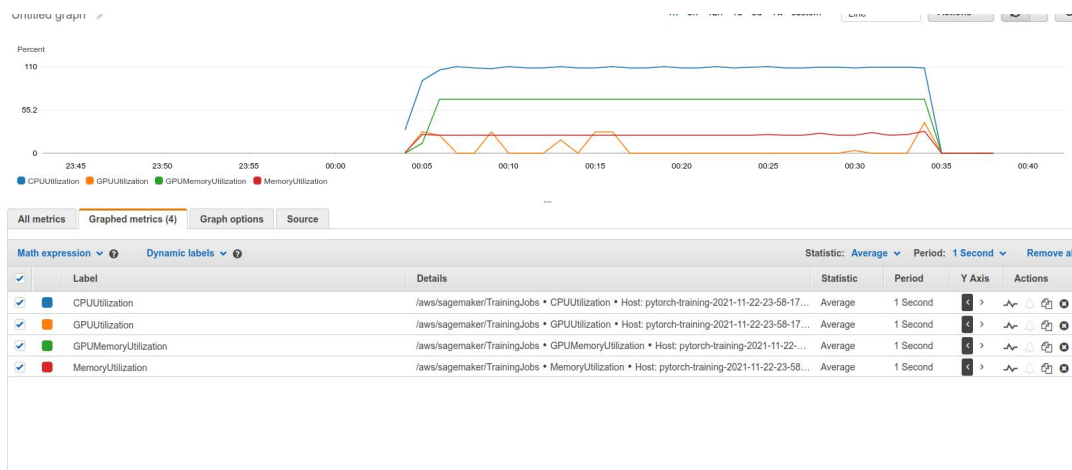


Figure 3: Training job resource use

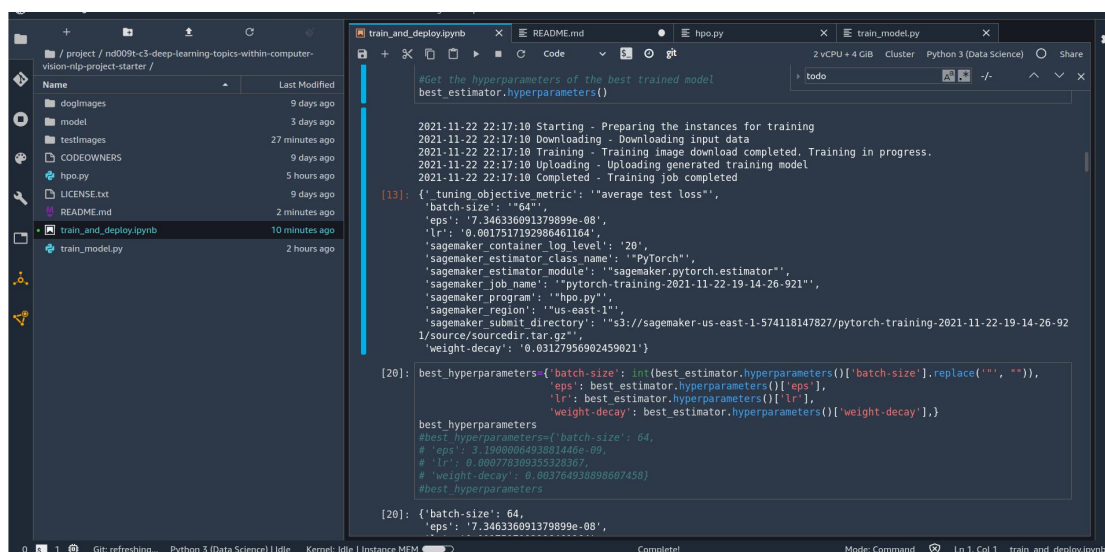
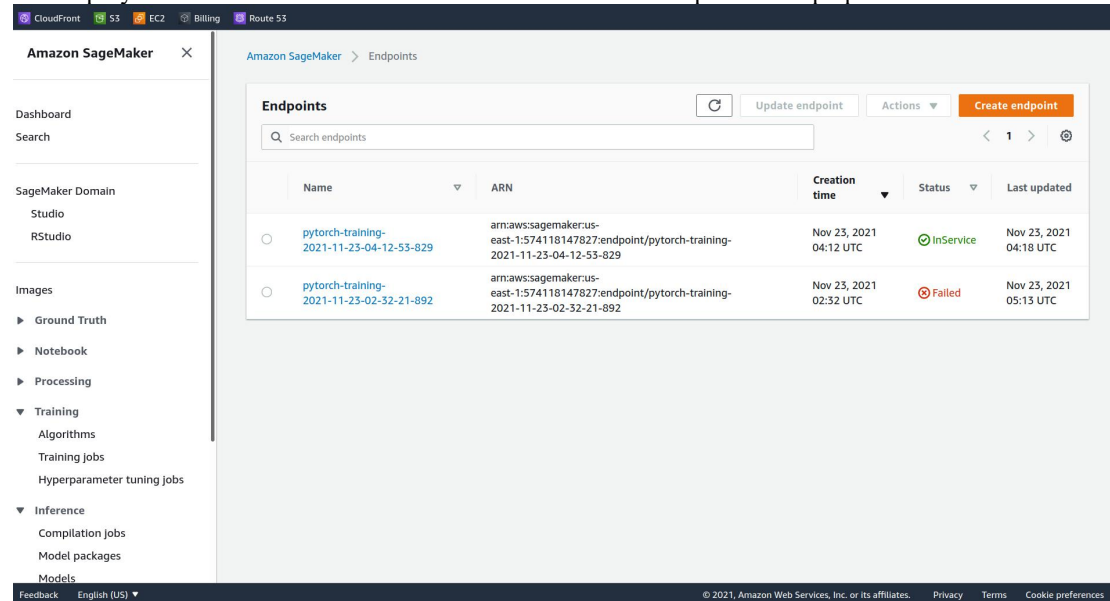


Figure 4: Training job output

## Model Deployment

The deployed model runs a tuned version of ResNet18 and accepts data as preprocessed in the notebook.



Name	ARN	Creation time	Status	Last updated
pytorch-training-2021-11-23-04-12-53-829	arn:aws:sagemaker:us-east-1:574118147827:endpoint/pytorch-training-2021-11-23-04-12-53-829	Nov 23, 2021 04:12 UTC	InService	Nov 23, 2021 04:18 UTC
pytorch-training-2021-11-23-02-32-21-892	arn:aws:sagemaker:us-east-1:574118147827:endpoint/pytorch-training-2021-11-23-02-32-21-892	Nov 23, 2021 02:32 UTC	Failed	Nov 23, 2021 05:13 UTC

Figure 5: Deployed endpoint



Figure 6: Penny for testing

```
import base64 # encode/decode image in base64
import json
import requests
import torch
import torchvision
import PIL
import torchvision.transforms as transforms

transform = transforms.Compose([
    transforms.Resize(256),
    transforms.RandomCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.496], std=[0.229, 0.224, 0.225]),
])

# Test images outside of dataset. The comment after is the result. All tested dogs are not breeds in the actual training data.
PIL_image = PIL.Image.open('testImages/Penny.png') #24 Bichon frise
#PIL_image = PIL.Image.open('testImages/20200912_121320.jpg') #52, 24 Clumber Spaniel, Bichon Frise
#PIL_image = PIL.Image.open('testImages/Rolo.png') #120 Pharaoh Hound
#PIL_image = PIL.Image.open('testImages/DSC08918.JPG') #116 Parson Russell Terrier
#PIL_image = PIL.Image.open('testImages/DSC01822.JPG') #58 Dandie dinmont terrier
#PIL_image = PIL.Image.open('testImages/Lucy.png') #42 Cairn Terrier
#PIL_image = PIL.Image.open('testImages/IMG_20210419-WA0000.jpg') #97 Lakeland Terrier
#PIL_image = PIL.Image.open('testImages/IMG_20210727-WA0000.jpg') #72 German Shorthaired Pointer
#PIL_image = PIL.Image.open('dogImages/dogImages/test/024.Bichon frise/Bichon frise_01707.jpg') #24
#PIL_image = PIL.Image.open('dogImages/dogImages/test/001.Affenpinscher/Affenpinscher_00003.jpg') #1

image = transform(PIL_image)

payload = image.unsqueeze(dim=0)

# TODO: Run an prediction on the endpoint

#image = # TODO: Your code to load and preprocess image to send to endpoint for prediction

response = predictor.predict(payload)
import numpy as np
np.argmax(response) + 1
```

24

Figure 7: Using the endpoint