Image Classification using AWS SageMaker

# Use AWS Sagemaker to train a pretrained model that can perform image classification by using the Sagemaker profiling, debugger, hyperparameter tuning and other good ML engineering practices. This can be done on either the provided dog breed classification data set or one of your choices.

⇨ We present the code and findings of the project of "Image Classification using Dog Breed Data and AWS Sagemaker" through this notebook. We will start by downloading the dog breed dataset. Then the dog breed data are uploaded to S3 bucket using the AWS Gateway. The next step performed is fine tuning the model by defining the four hyperparameters for tuning. In this project, we will use the pretrained Resnet 50 model. The next task of the project moves head with profiling and debugging new model using Sagemaker and then end the project with deploying the model and further analysis.

## Project Set Up and Installation

Enter AWS through the gateway in the course and open SageMaker Studio.

Download the starter files.

Download/Make the dataset available.

⇨ All of this done following instruction in the train_and_deploy.ipynb

## Dataset

The provided dataset is the dog breed classification dataset which can be found in the classroom.

The project is designed to be dataset independent so if there is a dataset that is more interesting or relevant to your work, you are welcome to use it to complete the project.

⇨ The dog breed data provided by the Udacity course in the project overview page of this part of court is used in this project. The data has images of 133 breeds of dogs which are divided into training test and validation datasets.

### Access

Upload the data to an S3 bucket through the AWS Gateway so that SageMaker has access to the data.

Done.

## Hyperparameter Tuning

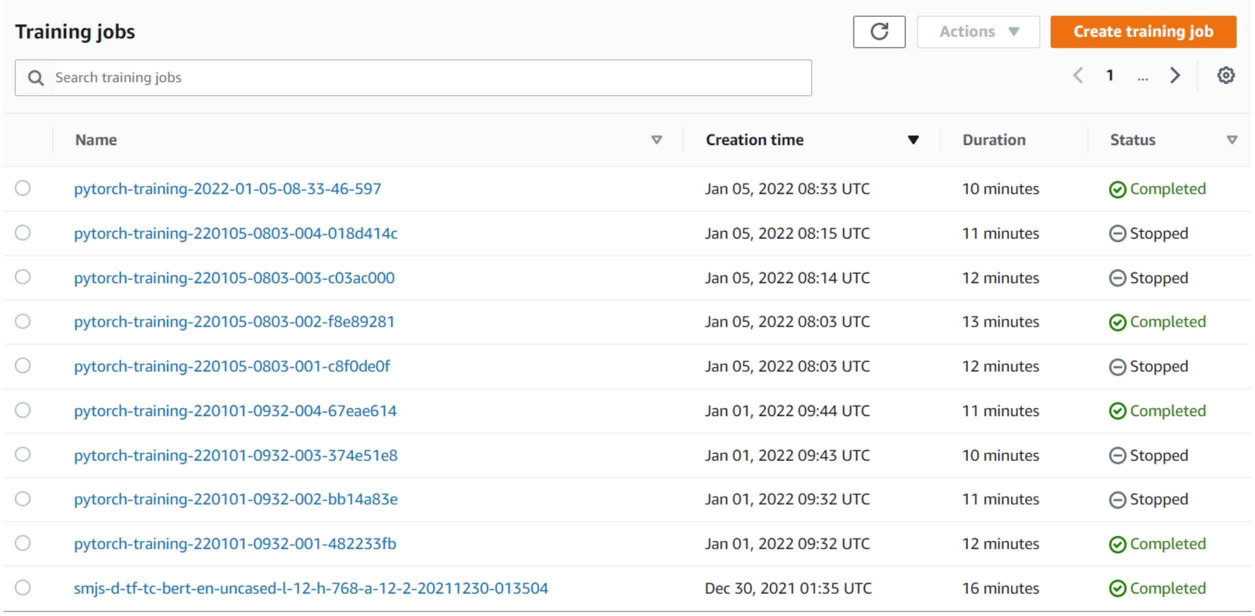
What kind of model did you choose for this experiment and why? Give an overview of the types of parameters and their ranges used for the hyperparameter search

- ⇨ We fine-tuned ResNet model with hyperparameter tuning in this work. We decided to fine tune four hyperparameters. The four hyper parameters fine-tuned are as below: 1)eps which is a continuous hyper parameter is fine tuned in (1e-8, 1e-5) 2)Learning rate, which is also a continuous hyper parameter is fine tuned in (0.0001, 0.05) 3)weight_decay is also an continuous hyper parameter is finetuned in (1e-3, 1e1) 4)Batch size is the categorical hyper parameter and is fine tuned in[64,128,512]

- ⇨ Among the hyperparameters tuned, batch size and learning rate have the highest impact on teh image classification task. The hyperparameter tuned here are in general considered by the researcher for tuning to get the best fit model in the image classification problem.

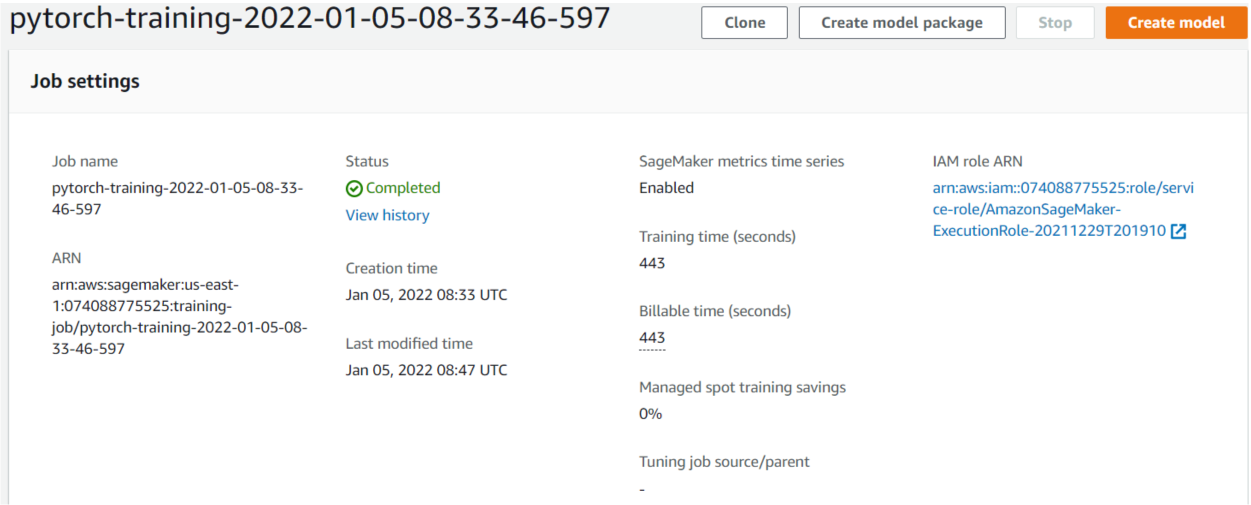
Remember that your README should:

- Include a screenshot of completed training jobs



	Name	Creation time	Duration	Status
○	pytorch-training-2022-01-05-08-33-46-597	Jan 05, 2022 08:33 UTC	10 minutes	✓ Completed
○	pytorch-training-220105-0803-004-018d414c	Jan 05, 2022 08:15 UTC	11 minutes	⊖ Stopped
○	pytorch-training-220105-0803-003-c03ac000	Jan 05, 2022 08:14 UTC	12 minutes	⊖ Stopped
○	pytorch-training-220105-0803-002-f8e89281	Jan 05, 2022 08:03 UTC	13 minutes	✓ Completed
○	pytorch-training-220105-0803-001-c8f0de0f	Jan 05, 2022 08:03 UTC	12 minutes	⊖ Stopped
○	pytorch-training-220101-0932-004-67eae614	Jan 01, 2022 09:44 UTC	11 minutes	✓ Completed
○	pytorch-training-220101-0932-003-374e51e8	Jan 01, 2022 09:43 UTC	10 minutes	⊖ Stopped
○	pytorch-training-220101-0932-002-bb14a83e	Jan 01, 2022 09:32 UTC	11 minutes	⊖ Stopped
○	pytorch-training-220101-0932-001-482233fb	Jan 01, 2022 09:32 UTC	12 minutes	✓ Completed
○	smjs-d-tf-tc-bert-en-uncased-l-12-h-768-a-12-2-20211230-013504	Dec 30, 2021 01:35 UTC	16 minutes	✓ Completed

- Logs metrics during the training process



pytorch-training-2022-01-05-08-33-46-597

Job settings

Job name
pytorch-training-2022-01-05-08-33-46-597

ARN
arn:aws:sagemaker:us-east-1:074088775525:training-job/pytorch-training-2022-01-05-08-33-46-597

Status
Completed
View history

Creation time
Jan 05, 2022 08:33 UTC

Last modified time
Jan 05, 2022 08:47 UTC

SageMaker metrics time series
Enabled

Training time (seconds)
443

Billable time (seconds)
443

Managed spot training savings
0%

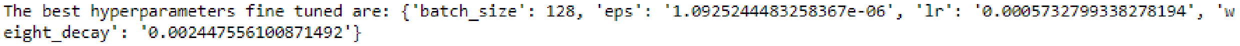
Tuning job source/parent
-

IAM role ARN
arn:aws:iam::074088775525:role/service-role/AmazonSageMaker-ExecutionRole-20211229T201910

- Tune at least two hyperparameters

The four hyper parameters fine-tuned are as below: 1)eps which is a continuous hyper parameter is fine tuned in (1e-8, 1e-5) 2)Learning rate, which is also a continuous hyper parameter is fine tuned in (0.0001, 0.05) 3)weight decay is also an continuous hyper parameter is finetuned in (1e-3, 1e1) 4)Batch size is the categorical hyper parameter and is fine tuned in[64,128,512]

- Retrieve the best hyperparameters from all your training jobs

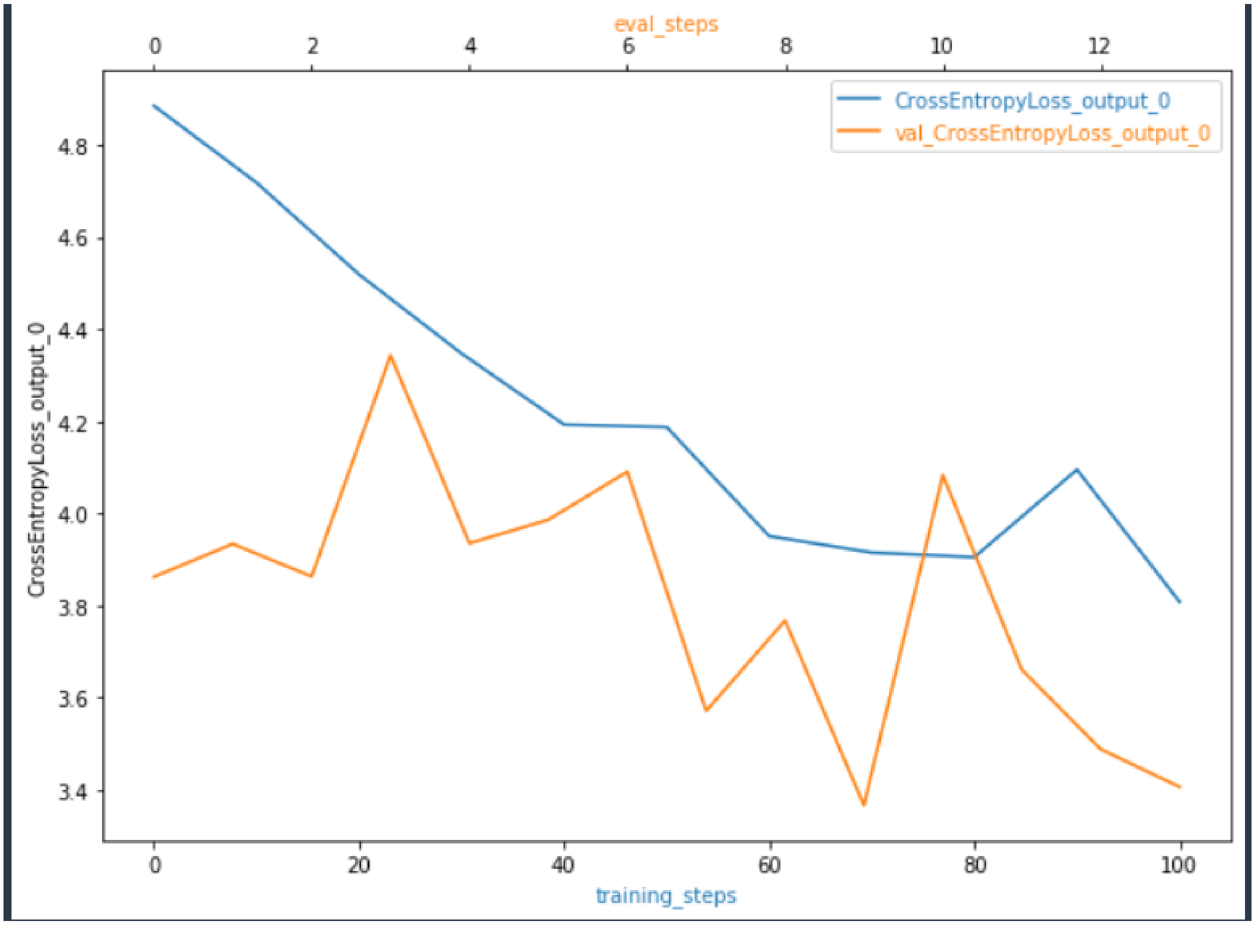


```
The best hyperparameters fine tuned are: {'batch_size': 128, 'eps': '1.0925244483258367e-06', 'lr': '0.0005732799338278194', 'w
eight_decay': '0.002447556100871492'}
```

## Debugging and Profiling

Give an overview of how you performed model debugging and profiling in Sagemaker

- ⇨ SageMaker Debugger utilizes the built-in profiling rules to find the unwanted scenario which can develop during the training. We utilized the trained model, the best hyperparameters along with the profiler and debugger hoods for profiling and debugging in sagemaker



### Results

TODO: What are the results/insights did you get by profiling/debugging your model?

- ⇨ The output of the debugger plot is not as smooth as expected. Adding the layers in the pretrained model can help to make it smooth.

TODO Remember to provide the profiler html/pdf file in your submission.

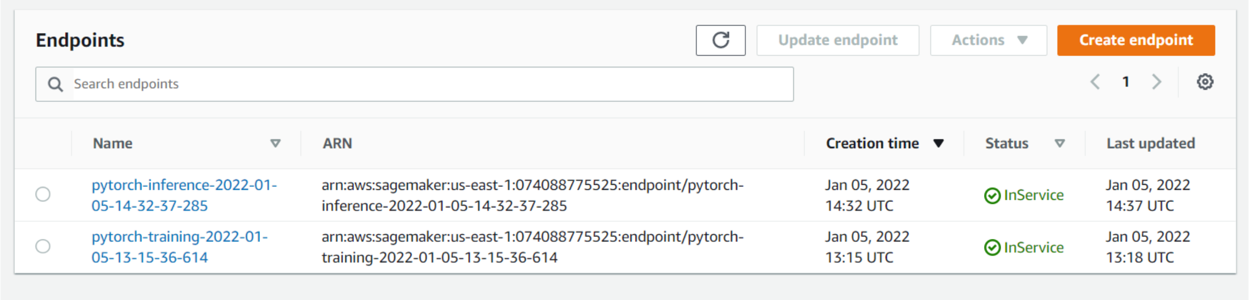
Html submitted through the github.

## Model Deployment

TODO: Give an overview of the deployed model and instructions on how to query the endpoint with a sample input.

⇨ Model was deployed in “ml.t2.medium" instance and were queried with the images as input to the endpoint created as a result of model deployment. Endpoint is store in S3 object after the model is deployed.

TODO Remember to provide a screenshot of the deployed active endpoint in Sagemaker.



Endpoints

	Name	ARN	Creation time	Status	Last updated
○	pytorch-inference-2022-01-05-14-32-37-285	arn:aws:sagemaker:us-east-1:074088775525:endpoint/pytorch-inference-2022-01-05-14-32-37-285	Jan 05, 2022 14:32 UTC	✓ InService	Jan 05, 2022 14:37 UTC
○	pytorch-training-2022-01-05-13-15-36-614	arn:aws:sagemaker:us-east-1:074088775525:endpoint/pytorch-training-2022-01-05-13-15-36-614	Jan 05, 2022 13:15 UTC	✓ InService	Jan 05, 2022 13:18 UTC