**Operationalizing an AWS ML Project**

Table of Contents

[Notebook Setup 1](#_Toc119627811)

[Sagemaker Training and Deployment 2](#_Toc119627812)

[EC2 Training 3](#_Toc119627813)

[Lambda functions 4](#_Toc119627814)

[Security and Testing 5](#_Toc119627815)

[Concurrency and Auto-scaling 7](#_Toc119627816)

# Notebook Setup

I have chosen the ‘ml.t2.medium’ instance type for the following reasons. The execution of the code does not require a very computationally powerful CPU and high RAM, hence we should look at smaller instances**.** To avoid high costs, we should select a notebook that is low in per hour cost while meeting the require CPU and RAM needs. Comparing ‘ml.t2.medium’ and ‘ml.t3.medium’, the former is cheaper due to slower boot time while having the same 2 vCPU and 4GB memory. Since boot time speed is not important, the former was chosen.

Graphical user interface, application

Description automatically generated

**Figure 1. Sagemaker Notebook Instance**

The dog breed dataset has been uploaded to the S3 bucket using sagemaker.

Graphical user interface, text, application

Description automatically generated

**Figure 2. S3 Bucket**

# Sagemaker Training and Deployment

For hyperparameter tuning, the ‘ml.m5.2xlarge’ instance type, which has 8 vCPU and 32 GB of RAM at a cost of $0.461 per hour, was used. Training was done using a multi-instance format, with max jobs of 6 and max parallel jobs of 3.

Table

Description automatically generated with medium confidence

**Figure 3. Hyperparameter Tuning Job**

For training, the ‘ml.m5.2xlarge’ instance type, which has 8 vCPU and 32 GB of RAM at a cost of $0.461 per hour, was used. Training was done using a multi-instance format, with max jobs of 6 and max parallel jobs of 3.

***Graphical user interface, application, email

Description automatically generated***

**Figure 4. Multi-Instance Training Job**

Multi instance deployed endpoint: pytorch-inference-2022-11-17-17-22-57-546

Graphical user interface, text, application, email

Description automatically generated

**Figure 5. Sagemaker Endpoints**

# EC2 Training

The **t2.2xlarge** instance and the **Deep Learning AMI (Amazon Linux 2)** was used. Given that t2 instances can sustain high CPU performance for long periods without incurring extraordinary costs, this instance is a good mix of performance and affordability.

Graphical user interface, text, application

Description automatically generated

**Figure 6. EC2 Instance Info**

Text

Description automatically generated

**Figure 7. EC2 Training Saved - model.pth**

|  |  |  |
| --- | --- | --- |
| Item | EC2 | Sagemaker |
| Dataset/Model | Takes from local path | Takes from S3 bucket |
| Hyperparameter Tuning | Internal script | External script |
| Instances | Script and training job on the same instance | Script and training job on separate instance |

**Figure 8. Comparison of EC2 and Sagemaker scripts**

# Lambda functions

The lambda function will be used to invoke the deploy endpoint for the multi-instance trained model.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Figure 9. Lambda Function**

# Security and Testing

Test event was executed in lambda function.

Graphical user interface, text, application, email

Description automatically generated

**Figure 10. Lambda Function Test Event**

Upon execution of test evet, we got ‘AccessDeniedException’as the lambda function does not have access to Sagemaker.

Graphical user interface, text

Description automatically generated with medium confidence

**Figure 11. Lambda function test event failure response**

The ‘SageMakerFullAccess**’** policy was added to lambda function’s role.

Graphical user interface, text, application, email

Description automatically generated

**Figure 12. Lambda function role IAM permissions**

Test event was successfully executed, and 33 dog breeds was indicated in the result.

Graphical user interface, application, Word

Description automatically generated

**Figure 13. Lambda function test event success response**

While we have given the lambda function full access to sagemaker as well as the sagemaker notebook full access to S3, it is possible to add further granular permissions to allow access to a specific notebook or s3 bucket.

Graphical user interface, text, application, email

Description automatically generated

**Figure 14. IAM Roles**

# Concurrency and Auto-scaling

Version configuration for our lambda function was created.

Concurrency is set to 5, which means that the function can handle up to 5 requests at the same time. We will only be using provisioned concurrency, as we are not given the maximum number of requests.

Graphical user interface

Description automatically generated with medium confidence

**Figure 15. Lambda Function Provision Concurrency**

Auto-scaling is set to a maximum of 3 instances, which means in time of high number of requests, 2 additional instances will be deployed. In addition, a scale-in and scale-out cooldown time of 30 seconds was used to ensure that users are not experiencing high latency.

Graphical user interface, application

Description automatically generated

**Figure 16. Endpoint Auto-scaling Config**