

Operationalizing-an-AWS-ML-Project

Dog Image Classification

The completed project contains code that trains and deploys an image classification model on AWS Sagemaker. Your goal in this project will be to use several important tools and features of AWS to adjust, improve, configure, and prepare the model you started with for production-grade deployment.

In this project, you will complete the following steps:

1. Train and deploy a model on Sagemaker, using the most appropriate instances. Set up multi-instance training in your Sagemaker notebook.
2. Adjust your Sagemaker notebooks to perform training and deployment on EC2.
3. Set up a Lambda function for your deployed model. Set up auto-scaling for your deployed endpoint as well as concurrency for your Lambda function.
4. Ensure that the security on your ML pipeline is set up properly.

Step 1: Training and deployment on Sagemaker

- **Created sagemaker notebook instance** For creating sagemaker notebook, I used ml.t3.medium because it is a low cost instance and would be sufficient to run my notebook.

Amazon SageMaker > Notebook instances

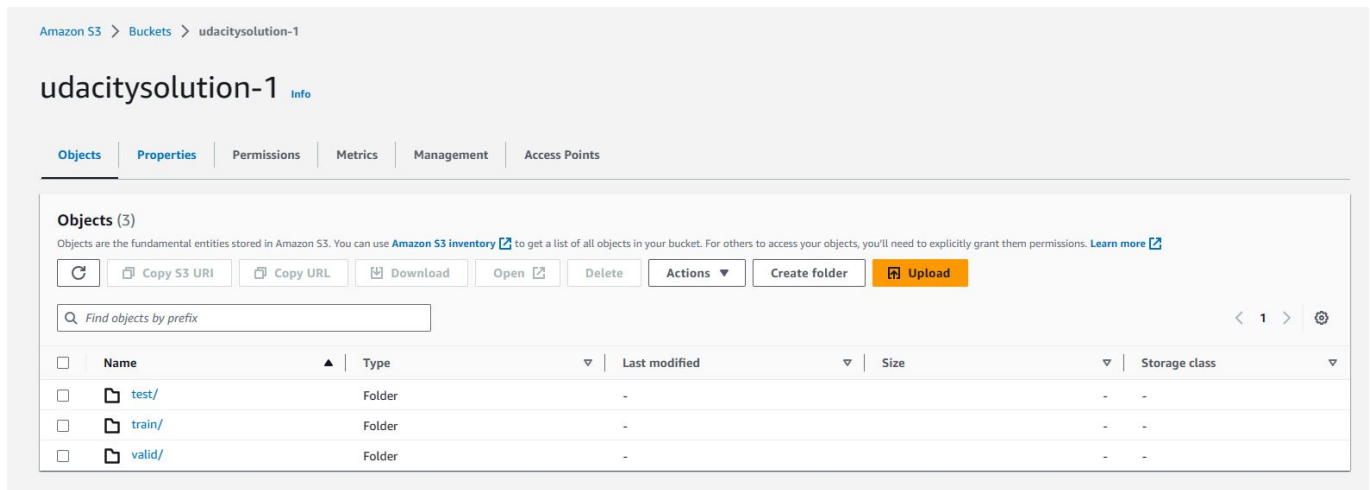
Notebook instances info

Search notebook instances

	Name	Instance	Creation time	Status	Actions
<input checked="" type="radio"/>	solution-1	ml.t3.medium	7/17/2023, 4:05:06 PM	Stopped	Start
<input type="radio"/>	solution	ml.t3.medium	7/17/2023, 4:01:43 PM	InService	Open Jupyter Open JupyterLab

- **S3 bucket for the job** I created a bucket name "udacitysolution-1" and upload data into it using the following code:

```
!wget https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
!unzip dogImages.zip
!aws s3 cp dogImages s3://udacitysolution/ --recursive
```



For this model, there are two hyperparameters: learning rate and batch size.

```
hyperparameter_ranges = {
    "learning_rate": ContinuousParameter(0.001, 0.1),
    "batch_size": CategoricalParameter([32, 64, 128, 256, 512]),
}
```

I used a py script (hpo.py) as entry point to the estimator, this script contains the code need to train model with different hyperparameters values.

```
estimator = PyTorch(
    entry_point="hpo.py",
    base_job_name='pytorch_dog_hpo',
    role=role,
    framework_version="1.4.0",
    instance_count=1,
    instance_type="ml.g4dn.xlarge",
    py_version='py3'
)

tuner = HyperparameterTuner(
    estimator,
    objective_metric_name,
    hyperparameter_ranges,
    metric_definitions,
    max_jobs=2,
    max_parallel_jobs=1, # you once have one ml.g4dn.xlarge instance available
    objective_type=objective_type
)
```

Here I passed some paths to our S3 which will be used by the notebook instance to get data, save model and output

```
os.environ['SM_CHANNEL_TRAINING']='s3://udacitysolution-1/'
os.environ['SM_MODEL_DIR']='s3://udacitysolution-1/model/'
os.environ['SM_OUTPUT_DATA_DIR']='s3://udacitysolution-1/output/'
tuner.fit({"training": "s3://udacitysolution-1/"})
```

I started the model training we can see the training job status at SageMaker -> Training -> Training Jobs

Best training job

Training jobs

Training job definitions

Tuning Job configuration

Tags

Training job status counter

Completed 2In Progress 0Stopped 0Failed 0 (Retryable: 0, Non-retryable: 0)

Training jobs

Sorting by objective metric value will display only jobs that have metric values.

View logs

View instance metrics

Stop

Create model

Search training jobs

<1>

	Name	Status	Final objective metric value	Creation time	Training Duration
<input type="radio"/>	pytorch-training-230717-0922-002-f37839c7	Completed	129	7/17/2023, 4:47:09 PM	17 minute(s)
<input type="radio"/>	pytorch-training-230717-0922-001-1a41923c	Completed	474	7/17/2023, 4:22:32 PM	21 minute(s)

Then I got the best model

Best training job hyperparameters

Search

Name	Type	Value
_tuning_objective_metric	FreeText	Test Loss
batch_size	Categorical	"32"
learning_rate	Continuous	0.007770519608421199
sagemaker_container_log_level	FreeText	20
sagemaker_estimator_class_name	FreeText	"PyTorch"
sagemaker_estimator_module	FreeText	"sagemaker.pytorch.estimator"
sagemaker_job_name	FreeText	"pytorch_dog_hpo-2023-07-17-09-22-26-957"
sagemaker_program	FreeText	"hpo.py"
sagemaker_region	FreeText	"us-east-1"
sagemaker_submit_directory	FreeText	"s3://sagemaker-us-east-1-856800247221/pytorch_dog_hpo-2023-07-17-09-22-26-957/source/sourcedir.tar.gz"

- Single instance training with best hyperparameters values

CloudWatch > Log groups > /aws/sagemaker/TrainingJobs

/aws/sagemaker/TrainingJobs

ActionsView in Logs InsightsStart tailingSearch log group

▼ Log group details

ARN
arn:aws:logs:us-east-1:856800247221:log-group:/aws/sagemaker/TrainingJobs:*

Creation time
3 hours ago

Retention
Never expire

Stored bytes
-

Metric filters
0

Subscription filters
0

Contributor Insights rules
-

Data protection
-

Sensitive data count
-

KMS key ID
-

Log streamsMetric filtersSubscription filtersContributor InsightsTagsData protection

Log streams (7)

dog-pytorch-2023-07-17-10-10-25-3891 matchExact matchShow expiredInfo

Log stream▼Last event time▼

dog-pytorch-2023-07-17-10-25-389/algo-1-16895886952023-07-17 17:29:28 (UTC+07:00)

- Multi-instance training with best hyperparameters values (4 instances)

CloudWatch > Log groups > /aws/sagemaker/TrainingJobs

/aws/sagemaker/TrainingJobs

ActionsView in Logs InsightsStart tailingSearch log group

▼ Log group details

ARN
arn:aws:logs:us-east-1:856800247221:log-group:/aws/sagemaker/TrainingJobs:*

Creation time
3 hours ago

Retention
Never expire

Stored bytes
-

Metric filters
0

Subscription filters
0

Contributor Insights rules
-

Data protection
-

Sensitive data count
-

KMS key ID
-

Log streamsMetric filtersSubscription filtersContributor InsightsTagsData protection

Log streams (7)

dog-pytorch-2023-07-17-10-50-15-6504 matchesExact matchShow expiredInfo

Log stream▼Last event time▼

dog-pytorch-2023-07-17-10-50-15-650/algo-1-16895910932023-07-17 18:09:28 (UTC+07:00)

dog-pytorch-2023-07-17-10-50-15-650/algo-3-16895910932023-07-17 18:09:24 (UTC+07:00)

dog-pytorch-2023-07-17-10-50-15-650/algo-2-16895910932023-07-17 18:09:21 (UTC+07:00)

dog-pytorch-2023-07-17-10-50-15-650/algo-4-16895910932023-07-17 18:09:17 (UTC+07:00)

- Deployment

Amazon SageMaker > Endpoints

Endpoints

Update endpointActionsCreate endpoint

Search endpoints

< 1 >

Name▼ARNCreation time▼Status▼Last updated

pytorch-inference-2023-07-17-11-56-14-371arn:aws:sagemaker:us-east-1:856800247221:endpoint/pytorch-inference-2023-07-17-11-56-14-3717/17/2023, 6:56:15 PMInService7/17/2023, 6:58:34 PM

Step 2: EC2 Training

I chose AMI with "Deep Learning AMI GPU PyTorch 1.13.1" and instance type selected was t2.xlarge because it is low cost and sufficient to train model.

4 / 9

Instances (1) info

Find instance by attribute or tag (case-sensitive)

Instance ID = i-008b846a83d2b323d Clear filters

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring
<input type="checkbox"/>	MLops	i-008b846a83d2b323d	Pending	t2.xlarge	-	No alarms +	us-east-1d	ec2-3-85-27-248.comp...	3.85.27.248	-	-	disabled

If everything works well we are connected to our instance. The last step is activate pytorch virtual enviroment on terminal by typing:

```
source activate pytorch
```

and train model as usually

```

AWS Deep Learning AMI GPU PyTorch 1.13.1 (Amazon Linux 2)

* Please note that Amazon EC2 F2 Instance is not supported on current DLAMI.
* Supported EC2 instances: G3, F3, F3dn, F4d, G5, G5dn.
* To activate pre-built pytorch environment, run: 'source activate pytorch'
* To activate base conda environment upon login, run: 'conda config --set auto_activate_base true'
* NVIDIA driver version: 525.85.12
* CUDA version: 11.7

AWS Deep Learning AMI Homepage: https://aws.amazon.com/machine-learning/amis/
Release Notes: https://docs.aws.amazon.com/dlami/latest/dg/deguide/appendix-ami-release-notes.html
Support: https://forums.aws.amazon.com/forum.jspa?forumID=263
For a fully managed experience, check out Amazon SageMaker at https://aws.amazon.com/sagemaker
Security scan reports for python packages are located at: /opt/aws/dlami/info/

28 package(s) needed for security, out of 52 available
Run "sudo yum update" to apply all updates.
[root@ip-172-31-26-56 ~]# ls
dogImages dogImages.zip solution.py TrainedModels
[root@ip-172-31-26-56 ~]# source activate pytorch
bash: activate: No such file or directory
[root@ip-172-31-26-56 ~]# source activate pytorch
ERROR: Please note that the Amazon EC2 t2.xlarge instance type is not supported by current Deep Learning AMI.
Please try one of the supported EC2 instances: G3, F3, F3dn, F4d, G5, G5dn.
Please refer the DLAMI release notes https://aws.amazon.com/releases/notes/aws-deep-learning-ami-gpu-pytorch-1-13-1-ami-linux-2/ for more information.
(pytorch) python solution.py
/opt/conda/envs/pytorch/lib/python3.9/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
/opt/conda/envs/pytorch/lib/python3.9/site-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior
is equivalent to passing 'weights=ResNet50_Weights.IMAGENET1K_V1'. You can also use 'weights=ResNet50_Weights.DEFAULT' to get the most up-to-date weights.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
100%
Starting Model Training
saved
(pytorch) ls
dogImages dogImages.zip solution.py TrainedModels
(pytorch) ls TrainedModels/
model.pth
(pytorch)

```

Step 3: Lambda function setup

After training and deploying your model, setting up a Lambda function is an important next step. Lambda functions enable your model and its inferences to be accessed by API's and other programs, so it's a crucial part of production deployment.

Step 4: Lambda security setup and testing

- **Adding endpoints permission to lambda fucntions** Lambda function is going to invoke deployed endpoint. I create policy with permission to only invoke specific endpoint because if we give "Full Access", it has potential to be exploited by malicious actor . Two security policy has been attached to the role :

1. Basic Lambda function execution
2. Sagemaker endpoint invocation permission

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",

```

```

    "Effect": "Allow",
    "Action": "sagemaker:InvokeEndpoint",
    "Resource": "arn:aws:sagemaker:us-east-1:856800247221:endpoint/pytorch-inference-2023-07-17-11-56-14-371"
  }
]
}

```

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.

Policy name	Type	Description
<input type="checkbox"/> AWSLambdaBasicExecutionRole-1d9e5a62-feb5-4577-8ea3-6143bdbb7e64	Customer managed	
<input type="checkbox"/> InvokeEndpoint	Customer inline	

InvokeEndpoint

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": "sagemaker:InvokeEndpoint",
8       "Resource": "arn:aws:sagemaker:us-east-1:856800247221:endpoint/pytorch-inference-2023-07-17-11-56-14-371"
9     }
10  ]
11 }

```

I got the results after running the test with json:

```

{ "url": "https://s3.amazonaws.com/cdn-origin-etr.akc.org/wp-content/uploads/2017/11/20113314/Carolina-Dog-standing-outdoors.jpg" }

```

Code source [Info](#)

File Edit Find View Go Tools Window

Environment: [inference](#) [lambda_function.py](#)

Execution results Status: **Succeeded** Max memory used: 79 MB Time: 1149.47 ms

Test Event Name

Response

```

{
  "statusCode": 200,
  "headers": {
    "Content-Type": "text/plain",
    "Access-Control-Allow-Origin": "*"
  },
  "type-result": "<class 'str'>",
  "Content-Type-In": "<_main_.LambdaContext object at 0x7fce162d9c70>",
  "body": "[[0.11569850146770477, 0.07335714250802994, -0.004642227198928595, 0.05507751926779747, 0.2541041970252991, 0.1466033096981049, -0.08659107983112335, 0.12284155189990997, -0.27110689878463745, ...]]"
}

```

Function Logs

```

START RequestId: eee2a880-818a-4778-a597-aebc4dda6bb9 Version: $LATEST
Context:: <_main_.LambdaContext object at 0x7fce162d9c70>
Event type: <class 'dict'>
END RequestId: eee2a880-818a-4778-a597-aebc4dda6bb9
REPORT RequestId: eee2a880-818a-4778-a597-aebc4dda6bb9 Duration: 1149.47 ms Billed Duration: 1150 ms Memory Size: 128 MB Max Memory Used: 79 MB

```

Request ID

eee2a880-818a-4778-a597-aebc4dda6bb9

Step 5: Concurrency and auto-scaling

By default a Lambda Function can only respond one request at once. One way to change that is to use concurrency so that the Lambda Function can be responds to multiple requests at once.

To set up concurrency on your Lambda function, you will need to open your Lambda function in the Lambda section of AWS. Next, you should open the Configuration tab. Then, you should configure a Version for your function, in the Version section of the Configuration tab.

After configuring a Version for your Lambda function, navigate to the Concurrency section of the Configuration tab of your function. Use this section to configure concurrency for your Lambda function.

Lambda > Functions > inference > Version: 1 > Configure provisioned concurrency

Configure provisioned concurrency

Provisioned concurrency

Version: 1

Aliases: -

Provisioned concurrency

To enable your function to scale without fluctuations in latency, use provisioned concurrency. You can use Application Auto Scaling to automatically adjust provisioned concurrency to maintain a configured target utilization. Provisioned concurrency runs continually and has separate pricing for concurrency and execution duration. [Learn more](#)

\$2.79 per month in addition to pricing for duration and requests. [Pricing](#)

2

2 available

Cancel

Save

Lambda > Functions > inference > Version: 1

Version: 1

Copy ARN

Actions

Function overview

inference:1

Layers (0)

+ Add trigger

+ Add destination

Description

new_version

Last modified

26 minutes ago

Function ARN

arn:aws:lambda:us-east-1:856800247221:function:inference:1

Code

Test

Monitor

Configuration

General configuration

Triggers

Permissions

Provisioned concurrency

Provisioned concurrency

2

Status

Ready

In addition to setting up concurrency for your Lambda function, you should also set up auto-scaling for your deployed endpoint.

7 / 9

Variant automatic scaling [Learn more](#)

Variant name
AllTraffic

Instance type
ml.m5.large

Elastic Inference
-

Current instance count
1

Current weight
1

Minimum instance count
1

-

Maximum instance count
3

IAM role
Amazon SageMaker uses the following service-linked role for automatic scaling. [Learn more](#)

AWSServiceRoleForApplicationAutoScaling_SageMaker
Endpoint

Built-in scaling policy [Learn more](#)

Policy name
SageMakerEndpointInvocationScalingPolicy

Target metric
[SageMakerVariantInvocationsPerInstance](#)

Target value
20

Scale in cool down (seconds)
- optional
30

Scale out cool down (seconds)
- optional
30

☐ Disable scale in

Select if you don't want automatic scaling to delete instances when traffic decreases. [Learn more](#)

Endpoint runtime settings										Update weights	Update instance count	Configure auto scaling
	Variant name	Current weight	Desired weight	Elastic Inference	Instance type	Current instance count	Desired instance count	Instance min - max	Automatic scaling			
<input type="radio"/>	AllTraffic	1	1	-	ml.m5.large	1	1	1 - 3	Yes			

Endpoint configuration settings

[Change](#)[Clone](#)

Endpoint configuration

Name pytorch-inference-2023-07-17-11-56-14-371	ARN arn:aws:sagemaker:us-east-1:856800247221:endpoint-config/pytorch-inference-2023-07-17-11-56-14-371	Encryption key -	Creation time 7/17/2023, 6:56:14 PM
---	---	---------------------	--

Data capture

