Please note: all screenshots are within subdirectory screenshots

Step 1: Training and Deployment om Sagemaker

screenshot:
- SageMaker_instance.png
- S3_bucket_creation.png
- Endpoint.png

instance_type=ml.t3.medium

I have selected an instance type of ml.t3.medium. This is a low cost instance $0.05 with 2CPUs and 4 GiB of Memory (https://aws.amazon.com/sagemaker/pricing/)
The notebook will be mainly used for initiating training jobs and creating endpoints. The training itself will run in instance_type defined as part of training and similar an instance_type will be defined as part of endpoint generation for the model serving.

Step 2: EC2 Training

screenshot:
- EC2_training_model.png

instance_type=g4dn.xlarge

I reviewed the estimator setting for the model in the notebook. The model is based on pytorch and instance_type equal to ml.g4dn.xlarge

I did search for similar settings for AMI deeplearning. From the extract below it's stated that these (including g4dn.xlarge) are the lowest cost GPU-based instances offering cost-effective solutions for graphics applications suiting our Dog classifier based on images.

----- Extract From Amazon ---
Amazon EC2 G4dn Instances
G4dn instances, powered by NVIDIA T4 GPUs, are the lowest cost GPU-based instances in the cloud for machine learning inference and small scale training. They also provide high performance and are a cost-effective solution for graphics applications that are optimized for NVIDIA GPUs using NVIDIA libraries such as CUDA, CuDNN, and NVENC. They provide up to 8 NVIDIA T4 GPUs, 96 vCPUs, 100 Gbps networking, and 1.8 TB local NVMe-based SSD storage and are also available as bare metal instances.

For EC2 we needed to define our own instance managed on EC2 (not SageMaker provisioned/managed). After creating the instance we had to manually copy our code. The instance needs to have the

libraries or environment for generating the model
installed, in our case pytorch. The code has been adjusted to
create, train and save using pytorch functionality.
For the notebook we could benefit from the fact that the instance
are created based on SageMaker provisioned instances.
Using SageMakers APIs we could define configuration and setting for
the model/endpoint and sagemaker did take care of the
bulk of the workload. Like creating an training instance using the
defined instance_type, creating, training the model
with specified parameters and finally saving the model to S3 from
which it can be utilised to create an endpoint.

Step 3: Lambda function setup

screenshot:
- Deployed_lambda_function.png
- Lambda_code.png

The purpose of this lambda function is to invoke a model from an
endpoint and return the
model/classifiers results based on a provided event. The event will
capture a picture of a dog for which the model will
return a classification i.e. infer it's dog breed. To communicate
with the endpoint there is a run-time session created
with SageMaker. Via the run-time session the endpoint is invoked
(model called for prediction) and the results are
returned in the results variable. As part of the return a message is
created with statusCode 200 for success and with
the results in the body of the return message.

Step 4: Security and testing

screenshot:
- Lambda_test_results_1.png
- Lambda_test_results_2.png
- SageMakerFullAccess_Lambda.png

Lambda function testing (list of 33 numbers):
[[-10.284646034240723, -7.630772113800049, -0.8699321150779724,
0.9804643392562866, -4.848867893218994, -5.435822010040283,
-0.8905675411224365, -1.3312197923660278, -8.49871826171875,
-0.20553195476531982, -0.07210561633110046, -6.214675426483154,
-4.609841823577881, 2.494947910308838, -6.163872718811035,
-1.487364649772644, -10.233490943908691, -3.119844913482666,
-4.937238693237305, 1.339205026626587, -3.984755754470825,
-1.611011266708374, -9.19676685333252, -6.554681777954102,
-7.34694242477417, -11.732565879821777, -2.5520827770233154,
-6.060281753540039, -9.885061264038086, -2.4542396068573,
-6.600430965423584, -5.319010257720947, -6.658382415771484,
-4.0419182777404785, -12.235913276672363, -9.67834186553955,
-7.546721458435059, -6.893711090087891, -3.812574863433838,
-6.368958473205566, -3.798821449279785, -6.789812088012695,

1.5941994190216064, -3.459785223007202, -0.04283590614795685,
-10.717300415039062, -2.048521041870117, -0.2805102467536926,
-6.473448276519775, -1.425976276397705, -4.320362091064453,
-9.549240112304688, -10.957330703735352, -2.8883426189422607,
-7.660244464874268, -0.8874025940895081, -6.4908127784729,
-9.019164085388184, -4.392540454864502, -3.8623852729797363,
-8.174982070922852, -11.761791229248047, -9.49162769317627,
-9.476940155029297, -4.831858158111572, -12.61843490600586,
1.0447107553482056, -9.017911911010742, -4.60083532333374,
-2.4722020626068115, 0.0769108459353447, -8.705763816833496,
-7.114100933074951, -6.815532684326172, -6.858256816864014,
-4.148754596710205, -11.577467918395996, -3.417882204055786,
-10.362234115600586, -6.070785045623779, 0.3484245538711548,
-9.403095245361328, 1.4810482263565063, -2.3210058212280273,
-10.35000228881836, -10.928654670715332, -1.9997351169586182,
-6.381765365600586, -4.905564785003662, -1.890192985534668,
-9.733583450317383, -5.565062522888184, -6.035923480987549,
-6.912452697753906, -9.550994873046875, -4.201255798339844,
-2.8515937328338623, -8.842897415161133, -9.43519115447998,
-8.145516395568848, -10.88481330871582, 0.31419724225997925,
-3.7690513134002686, -5.94817590713501, -6.026608943939209,
-10.892983436584473, -7.760615348815918, -0.31790822744369507,
-2.043574333190918, -1.5796029567718506, -3.9394760131835938,
-3.8428690433502197, -7.616644859313965, -5.14315128326416,
-9.981572151184082, -2.7746708393096924, -10.5945405960083,
0.5345878601074219, -8.341174125671387, 0.9495958089828491,
-1.4098871946334839, -5.14119291305542, -4.963842868804932,
-4.362910747528076, -7.628365516662598, -7.802095413208008,
-7.050898551940918, -0.3703683316707611, -4.826143741607666,
-10.586912155151367, -10.430638313293457, -1.85893714427948,
-8.298824310302734]]

The SageMakerFullAccess policy is providing too much power and a security risk for a role that only needs to
invoke a SageMaker endpoint. I would remove the SageMakerFullAccess policy in production and look for some privileges
that are limiting the role to the tasks it needs to be able to do. This could also result in creating a complete
separate role with required privileges/policies and assigning to the lambda function. Concrete I would remove the
FullAccess replacing with invoking endpoints only (and/or any other from SageMaker that would be missing after testing).

Step 5: Concurrency and auto-scaling

screenshot:
- Lambda_concurrency.png
- Auto_scaling.png

Concurrency allows Lambda to handle multiple requests concurrently, which can significantly improve the throughput
of our dog classification application. To enable concurrency for our application I have set number of provisioned

concurrency to 2 to start with. There is an extra cost for each provisioned instance, after monitoring the system or getting a better understanding of expected workload of the application I would adjust the number.

With auto-scaling, Lambda automatically scales the number of function instances in response to the incoming workload. When the incoming request rate increases, Lambda adds more instances to handle the additional load. Similar as for concurrency I decided to start conservative and allow up to 3 instances with desired number to only 1.