Operationalizing ML workflow on AWS-Sagemaker

Training and deployment in a Sagemaker Instance

1. Selected ml.t2.medium as the instance for the jupyter environment. Being familiar as used in previous projects and chapters and this is all around good instance with 2x vCPUs and 4GB of RAM to run some jupyter notebooks.
2. Worked out both single instance training and multi instance training. Multi-instance training took less time than single instance training for completion of training jobs. Models are different with inference scores and the multi instance training seems to be less loss and better accuracy.

   (Please refer image folder for evidences)

Training in an EC2 instance

1. Selected t3.medium as EC2 instance gave expected results, micro ones are too small and can't get the jobs done as per experimentation. 2 vCPUs with 4GB of RAM in t3.medium is right choice considering performance and cost.
2. Followed steps given for training. Method given executing script was so convenient (script - *solution.py*) for job training. For activation I used *source activate pytorch_p38*. The training took about only 8 minutes on this instance and it was succeed, and saved the model in the /TrainedModels path as *model.pth*.
3. Plot of CPU Utilizing during the training time is also in the corresponding image folder.

   (For images please refer EC2 Training Folder.)

Training Job in Sagemaker Vs EC2

1. The difference of codes between the Step 1 and 2 are pretty simple. The sagemaker notebook's code use a different script as entry point for the model training which mean the model is trained in the separate container, that's why we need to save os.environ channel variables specified for sagemaker containers like "SM_CHANNEL_TRAINING, SM_OUTPUT_DIR" etc. and call these in the function files to recognize the paths of the data inputs and outputs. The EC2 training can be said as local training as both the code and the training run in the same machine, so its codes don't need to call the os.environ variables with channel names and the path can be specify by only using os.path function.
2. For that we don't use sagemaker here in EC2 instance, the function calls for the sagemaker don't need to import and use in the EC2 training. The drawback is that we can't see the algorithm metrics easily in CloudWatch console as using sagemaker and

can only see if the model is training and complete successful by viewing the CPU utilization of the instance.

3. As there is no Estimator or Tuner function to call the script and implement, the call for train the model and save it, is in the code already. Model training, creating a fully connected network with pretrained model and creating data loaders are the same in both.
4. Next difference thing is that in EC2 training, all variables like hyperparameters and output locations, etc. are already declare in the script so no argparse is needed. These are the differences I've found out after doing the 2 different training jobs in Sagemaker and EC2. Since the data is processed in lesser time, it is cost-effective for businesses & helps them to move fast. Running a workload in a distributed environment also makes it more scalable.

Lambda Function

1. The lambda function acts as a trigger for invoking the deployed endpoint. It takes the input as JSON format and pass it to the endpoint to make the prediction and then pass out the result.
2. In the lambda code, the main functions are reading the input data, take the data, predicting the data and pass it, after that we can output the result to the Lambda function's output body.
3. Lambda function output the prediction as a list of numbers which represent the each score for the breed of dogs. Input 'url' used from CloudWatch logs "https://s3.amazonaws.com/cdn-origin-etr.akc.org/wp-content/uploads/2017/11/20113314/Carolina-Dog-standing-outdoors.jpg"
4. AWS Lambda functions are designed for traffic going through computing resources

(Please refer the images for this sections in the Lambda Function folder)

Security and Testing

1. Run the previous section's lambda function AS IT will throw an error which is validation error because it doesn't have access to the sagemaker endpoint. So I have added security policy to my lambda's role with AmazonSageMakerFullAccess and it can now access the sagemaker endpoint.
2. The original result is here , an array of 133 values : [[-0.9560320377349854, -1.845695972442627, -1.9934080839157104, 0.21899941563606262, -0.41526561975479126, -2.1272177696228027, 0.10941867530345917, -0.28120625019073486, -1.3089109659194946, 0.1912819892168045, 0.4455648958683014, -1.392216444015503, -1.3182817697525024, -0.09552408754825592, -0.6738944053649902, -0.3438718616962433, -

3.1571264266967773, -0.9911424517631531, -1.229315161705017, 0.9126419425010681, 0.04390105605125427, 0.03597955405712128, -1.6058295965194702, -2.551037073135376, -1.0609209537506104, -2.494464159011841, -0.9864355325698853, -1.5623911619186401, -3.3216047286987305, 0.04928336292505264, -0.48795706033706665, -1.3087732791900635, -2.6978976726531982, -1.1562985181808472, -3.7409775257110596, -3.3125927448272705, -0.3557204604148865, -2.3890140056610107, -0.46460458636283875, -1.0079275369644165, -1.294243335723877, -2.379828929901123, 0.5150794386863708, -0.9428315162658691, -1.709212303161621, -2.22542142868042, -1.5826168060302734, -0.5538403391838074, -2.123223066329956, -0.8225287795066833, -1.6139755249023438, -2.038969039916992, -1.6000288724899292, -2.0004475116729736, -2.143977642059326, 0.595905601978302, -4.069657802581787, -3.013295888900757, -0.450283944606781, -0.2532176971435547, -2.512537956237793, -2.9075136184692383, -2.588399648666382, -3.1245570182800293, -0.7736032009124756, -3.4575462341308594, 0.12425947189331055, -1.7864409685134888, -1.4868929386138916, 0.5813604593276978, 0.41606399416923523, -0.3576119840145111, -2.8170089721679688, -2.7939977645874023, -2.0108160972595215, -1.2022093534469604, -1.270866870880127, -1.2576234340667725, -1.7807035446166992, -1.015439748764038, 0.6646159887313843, -2.273350238800049, -0.731576681137085, -0.17046359181404114, -2.1878719329833984, -2.8903777599334717, -0.9122530817985535, -4.2487897872924805, -1.075931191444397, -0.01767263561487198, -3.653059959411621, -1.0689200162887573, -1.6979835033416748, -2.9535791873931885, -2.0769505500793457, -0.336151123046875, -1.3448779582977295, -2.0190787315368652, -2.205075263977051, -3.00744891166687, -2.8704471588134766, -1.0935817956924438, -1.770317792892456, -4.386178970336914, -2.097766876220703, -2.114828586578369, -2.723649740219116, -1.2167062759399414, -0.4806557595729828, -0.6132817268371582, -0.31669437885284424, -0.5354382991790771, -3.0936765670776367, -1.4011110067367554, -2.76177716255188, -1.0953298807144165, -3.4475953578948975, -0.2085481584072113, -1.966394305229187, -0.21666626632213593, -1.0014561414718628, -3.150832176208496, -3.93908429145813, 0.1824048012495041, -3.2652974128723145, -1.6477389335632324, -1.357935070991516, -0.2648433446884155, -1.1599692106246948, -3.035426139831543, -1.2490164041519165, -1.5336929559707642, -0.6992834806442261]]

3. In IAM dashboard as there is just a few resources, and all of the permissions are set according to the needs of it's usage.
4. But the FullAccess of my Lambda function may create a security vulnerability as it is too much for just using an endpoint yet I can't find the proper permission for just using an sagemaker endpoint. The

AmazonMachineLearningManageRealTime- EndpointOnlyAccess policy is out of the line as it is use to create or delete an endpoint and not using them.

Concurrency and Auto Scaling

1. Resources and functions are ready after execution of lambda. For setting the concurrency and auto scaling to the inference model, selected Provisioned concurrency and created version.
2. For the concurrency of lambda function, I have set up the reserved concurrency as 5 which mean the function can handle up to 5 request at the same time. I have also set the always-ready provisioned concurrency as 2 to reduce the latency.
3. For the endpoint, I have set up maximum 2 endpoints and set the scale in cold down to 30 and scale out cold down to 30 to minimize the cost while auto scaling during traffic and to reduce waiting period.

   (Please refer for the images in the Concurrency and Auto Scaling Folder.)