**Building AlexNet with TensorFlow and Running it with AWS SageMaker**

In the last post, we built [AlexNet](https://en.wikipedia.org/wiki/AlexNet) with [Keras](https://keras.io/). This is the second part of AlexNet building. Let’s rewrite the Keras code from the previous post (see [**Building AlexNet with Keras**](https://www.mydatahack.com/building-alexnet-with-keras)) with [TensorFlow](https://www.tensorflow.org/) and run it in [AWS SageMaker](https://aws.amazon.com/sagemaker/) instead of the local machine.

AlexNet is in fact too heavy for a regular commercial laptop to handle it. It only runs on a small dataset and takes for ages. By using the cloud service like AWS, we can access to much better computers without any hardware investment. AWS already has a series of deep learning specialised instances ([P2 Instances](https://aws.amazon.com/ec2/instance-types/p2/)). The smallest with one GPU (p2.xlarge) costs 90 cent per hour. It is much faster than CPU machines. You can experiment on computing capacities as you will be charged only by usage hours. If you are thinking about buying a more expensive GPU laptop for deep learning, the cloud services would be a better option.

AWS recently released SageMaker, which enables you to develop and deploy deep learning code with no hustle. To run Tensorflow code in SageMaker, all you need is to create a notebook instance (check out the getting started video [here](https://www.youtube.com/watch?v=tBRHh_V8vjc)).

**Importing OxfordFlower17 Data**

You can creates a notebook instance with a chosen EC2 instance with SageMaker. Once the instance is created, you can access to the instance through Jupyter notebook for development and deployment. Many deep learning frameworks are already installed. Once you train the model, you can deploy it into the AWS environment without much hustle. The caveat is that you won’t be able to install or update the preinstalled packages as you do not have access to the underlining instance. If you need to have special environmental requirements, you need to bring it in with a Docker container.

In fact, SageMaker does not have tflearn installed. As in the [previous post](https://www.mydatahack.com/building-alexnet-with-keras), we are importing [17 category flower dataset](http://www.robots.ox.ac.uk/~vgg/data/flowers/17/) (OxfordFlower17) from tflearn. If you try to import it in SageMaker, it will give you the module not found error.

The strategy I took here is to upload the dataset as numpy array files to S3 and retrieve them in SageMaker.

1. **Create the numpy files and Upload to S3**

I first created npy files and uploaded to S3 bucket where SageMaker has the access policy.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Import numpy files into the SageMaker instance.**

You can get the file from S3 into the Notebook instance and simply load them as numpy objects.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

**Code**

Strictly speaking, it is slightly different from the original AlexNet. The code is sequential and has no parallel computing components for simplicity. I am doing batch normalisation before every input and doing dropouts in the Dense layer. The network architecture is the same as the [previous post](https://www.mydatahack.com/building-alexnet-with-keras).

With TensorFlow, you really need to be careful about the dimensions. The original dataset is 3-dimentional. After the convolution layers, the dimension is compressed from pooling. So, you need to specify the right dimension (7 x 7 in this case). Otherwise, the code will not run.

In the model, I purposely included the weights and biases with hard-coded values so that it is easy to follow. Apart from the model, the same code used in building Dense Net for Iris works. If you need to understand other part of the codes you should read the previous post ([**Introduction to Dense Net with TensorFlow**](https://www.mydatahack.com/introduction-to-dense-net-with-tensorflow/)).

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]