FAQ - Optimizing Neural Network

#### **Q1. The various methods for creating Neural Networks models are listed below. What is the difference between these codes? What effect do these codes have on the model?**

***model = tf.keras.Sequential()***

***model = keras.Sequential()***

***model = Sequential()***

The only difference between the three codes mentioned above is in calling the Sequential function, and all three codes will work and produce the same model output.

* **model = tf.keras.Sequential()**

In the above code, the Sequential function is called using the TensorFlow and Keras library.

* **model = keras.Sequential()**In the above code, the Sequential function is called using the Keras library.
* **from tensorflow.keras.models import Sequential  
  model = Sequential()**In the above code, the Sequential function is called using Tensorflow and Keras libraries after importing the libraries.

#### **Q2. Is it always necessary to use batch normalization when building models with ANN? What is its application?**

No, Batch Normalization is not always used in the model unless the model is overfitting. Batch Normalization is one of the techniques used in Neural Networks to prevent the model from overfitting.

#### **Q3. How do we decide how many neurons to include in each of the hidden layers? Why 256, 64, 32, etc.?**

There are a few hyperparameters in Neural Networks that we will pass to the model, and the number of neurons in the model is one of them, so there is no rule of thumb for what number to use in the model.

As we know, computers work only with 0's and 1's, which are only two digits. This is why all the memory addressing, resolutions (in games), and size of storage devices are **powers of two**. And this is the reason for the number of neurons to be powers of two, too. Neural networks require GPUs for faster processing. And there is no rule that we should only use the power of two numbers as the neurons. We can use the non-power of two numbers as well.

#### **Q4. What is the function of 'units' in the code below?**

**model2.add(Dense(activation = 'relu', input\_dim = 11, units = 128))**

Units in dense layers are the number of neurons present in the dense layer. You can specify the number of neurons directly or define units = 128, which is shown in the below example.

**model2.add(Dense(activation = 'relu', input\_dim = 11, units = 128))**

**model2.add(Dense(128, activation = 'relu', input\_dim = 11))**

#### **Q5. Let’s say there is a dataset with 11 columns and 10000 rows. So the input dim should be equal to the number of columns, and the units should be equal to the number of rows?**

Yes, if there are 11 columns in the dataset then the input \_dim will be 11, but the units in the input layer will not be equal to the number of rows in the dataset. The number of units that should be present in the input/hidden layer is a hyperparameter that will be passed to the model. The number of units in the hidden layer can be 16,32,64,128,512 and 1024 etc.

#### **Q6. Why is it necessary to apply to\_categorical on the target column? Is this due to the Multi-Class classification problem?**

Yes, for a multi-class classification problem, we must encode the target column by to\_categorical to remove the weightage of the number.

In the week-1 hands-on, we used a multi-class classification with 10 classes to predict between 0 and 9. When we use this target variable in its current form, the model interprets the highest number, 9, as having more weight than other numbers, and the model is biased toward the highest number. To remove the weightage of the number, we will encode the target variable with to\_categorical.