By: Prashant Kr Mali

AI Assignment

Module - 3 & 4

Tasks to be performed:

Q. 1: What is Tensorflow and how is Keras different from it?

ANSWER:

Tensorflow

Tensorflow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow. Both provide high-level APIs used for easily building and training models, but Keras is more user-friendly because it's built-in Python.

Researchers turn to Tensorflow when working with large datasets and object detection and need excellent functionality and high performance. Tensorflow runs on Linux, macOS, Windows, and Android. The framework was developed by Google Brain and currently used for Google's research and production needs.

- Tensor flow has the advantage that it does support and uses many backend software like GUI and ASIC.
- When it comes to community support tensor flow has the best.
- Tensor flow also helps in debugging the sub-part of the graphs.
- Tensor flow has shown a better performance when compared with other platforms.
- Easy to extend as it gives freedom to add custom blocks to build on new ideas.

Keras

The reader should bear in mind that comparing Tensorflow and keras isn't the best way to approach the question since keras functions as a wrapper to Tensorflow's framework. Thus, you can define a model with keras' interface, which is easier to use, then drop it down into TensorFlow when you need to use a feature that keras doesn't have, or you're looking for specific Tensorflow functionality. Thus, you can place the TensorFlow code directly into the keras training pipeline or model.

Q. 2: What are the different types of models in Keras? Explain the working of Sequential modeling.

ANSWER:

A model is the basic data structure of Keras. Keras models define how to organize layers.

Models in keras are available in two types:

- Keras Sequential Model
- Keras Functional API

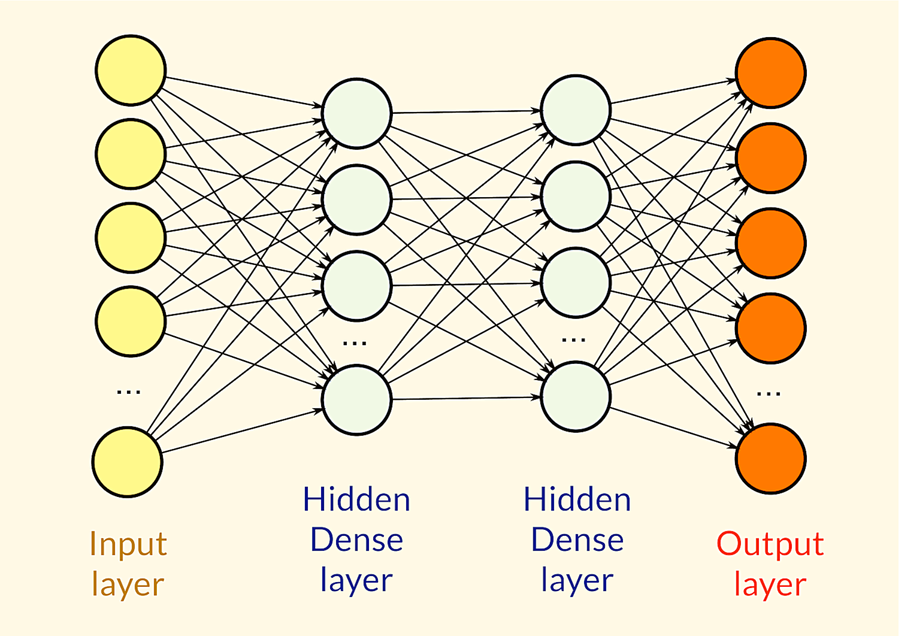
Sequential Model in Keras

It allows us to create models layer by layer in sequential order. But it does not allow us to create models that have multiple inputs or outputs. It is best for a simple stack of layers that have 1 input tensor and 1 output tensor.

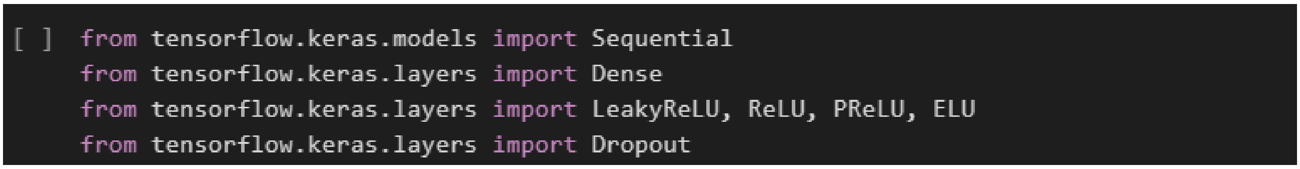
This model is not suited when any of the layers in the stack has multiple inputs or outputs. Even if we want a non-linear topology, it is not suited.

Sequential data includes text streams, audio clips, video clips, time-series data and etc. Recurrent Neural Networks (RNNs) is a popular algorithm used in sequence models.

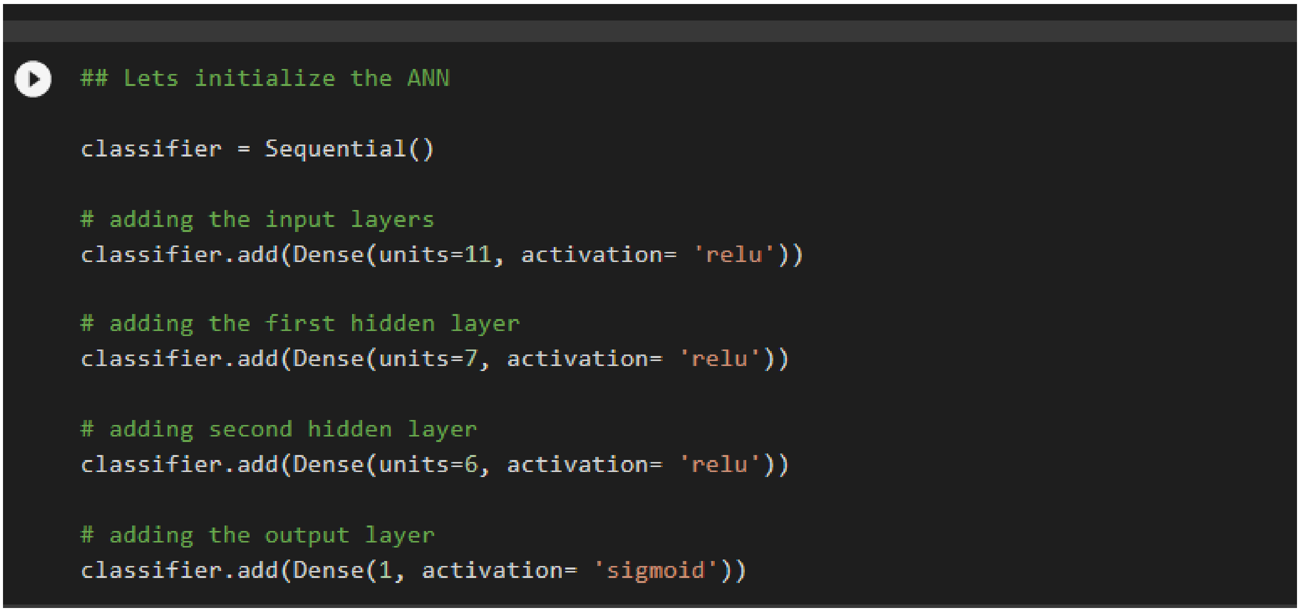
Here is an example of a Sequential model:



Script for an example in python to prepare Sequential model:



```
[ ]  from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense
     from tensorflow.keras.layers import LeakyReLU, ReLU, PReLU, ELU
     from tensorflow.keras.layers import Dropout
```



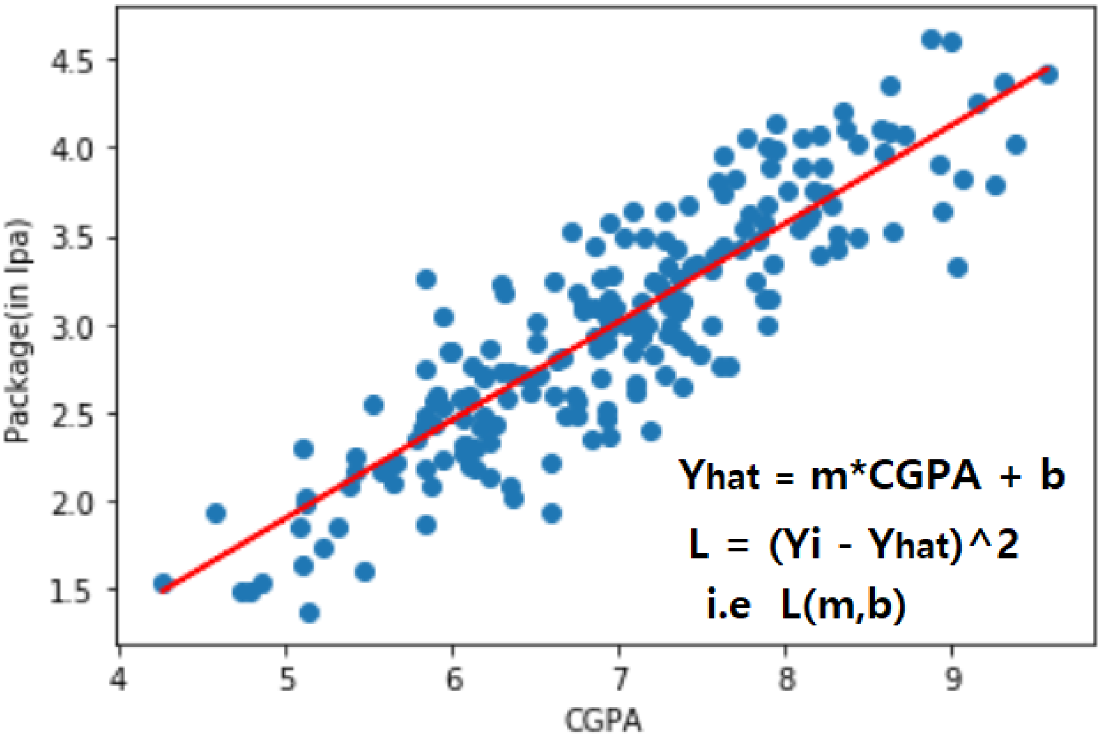
```
## Lets initialize the ANN

classifier = Sequential()

# adding the input layers
classifier.add(Dense(units=11, activation= 'relu'))

# adding the first hidden layer
classifier.add(Dense(units=7, activation= 'relu'))

# adding second hidden layer
classifier.add(Dense(units=6, activation= 'relu'))

# adding the output layer
classifier.add(Dense(1, activation= 'sigmoid'))
```

Q. 3: Explain the losses in Deep Learning and how are they responsible for model learning.

ANSWER: In mathematical optimization and decision theory, a loss or cost function (sometimes also called an error function) is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event.

In simple terms, the Loss function is a method of evaluating how well the algorithm is modeling your dataset. It is a mathematical function of the parameters of the machine learning algorithm.

In simple linear regression, prediction is calculated using slope(m) and intercept(b). the loss function for this is the (Yi – Yihat)^2 i.e loss function is the function of slope and intercept.



Importance of Loss function

- **Loss function vs Cost function**

Cost function and Loss function are synonymous and used interchangeably but they are different.

- **Loss Function:**

A loss function/error function is for a single training example/input.

- **Cost Function:**

On the other hand, a cost function is the average loss over the entire training dataset.

Loss function in Deep Learning

1. Regression

- MSE (Mean Squared Error)
- MAE (Mean Absolute Error)
- Hubber loss

2. Classification

- Binary cross-entropy
- Categorical cross-entropy

3. Auto Encoder

- KL Divergence

4. GAN

- Discriminator loss
- Minmax GAN loss

5. Object detection

- Focal loss

6. Word embeddings

- Triplet loss

Q. 4: MNIST has been the toy dataset for Deep Learning, but today you will need to design a Neural Network for the Fashion MNIST dataset (which can be sourced from keras.datasets.fashion_mnist).

a. **How many hidden layers are required for this dataset?**
b. **How many neurons per layer work best for the model, explain and generate insights across it.**

ANSWER: Kindly refer to the file name: 'FashionMNIST_ANN.ipynb' for the source code.

4_a. How many hidden layers are required for this dataset?

By following a small set of clear rules, one can programmatically set a competent network architecture and can efficiently decide the number of hidden layers and nodes in a neural network.

Choosing Hidden Layers

1. Well, if the data is linearly separable then we don't need any hidden layers at all.
2. If data is less complex and is having fewer dimensions or features, then neural networks with 1 to 2 hidden layers would work.
3. If data is having large dimensions or features then to get an optimum solution, 3 to 5 hidden layers can be used.

**It should be kept in mind that increasing hidden layers would also increase the complexity of the model and choosing hidden layers such as 8, 9, or in two digits may sometimes lead to overfitting.

4_b. How many neurons per layer work best for the model, explain and generate insights across it.

Choosing Nodes in Hidden Layers

Once hidden layers have been decided the next task is to choose the number of nodes in each hidden layer.

1. The number of hidden neurons should be between the size of the input layer and the output layer.
2. The most appropriate number of hidden neurons is,

sqrt (input layer nodes * output layer nodes)

The number of hidden neurons should keep on decreasing in subsequent layers to get more and more close to pattern and feature extraction and to identify the target class.

Sometimes the number of nodes in hidden layers can increase also in subsequent layers and the number of hidden layers can also be more than the ideal case.

**This whole depends upon the use case and problem statement that we are dealing with.

Q. 5: Explain your understanding of overfitting and underfitting in detail. How do you identify and measure overfitting?

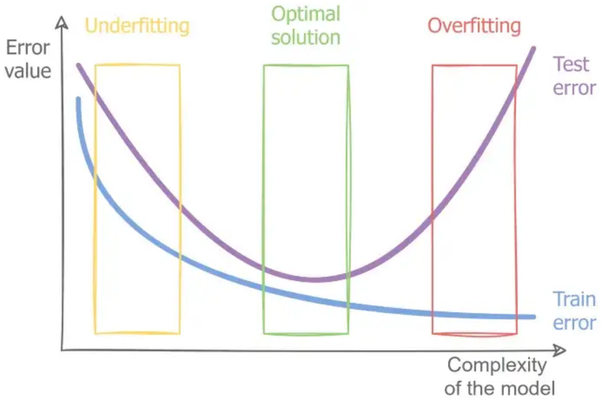
ANSWER:

Underfitting is a situation when the model is *too simple* for data. More formally, the hypothesis about data distribution is wrong and too simple — for example, the data is quadratic, and the model is linear. This situation is also called *high bias*. This means that the required algorithm can do accurate predictions, but the initial assumption about the data is incorrect.

Opposite, **overfitting** is a situation when the model is *too complex* for data. More formally, the hypothesis about data distribution is wrong and too complex — for example, the data is linear, and the required model is a high-degree polynomial. This situation is also called *high variance*. This means that the algorithm can't do accurate predictions — changing the input data only a little, the model output changes very much.

- low bias, low variance — is a good result, just right.
- low bias, *high variance* — *overfitting* — the algorithm outputs very different predictions for similar data.
- *high bias*, low variance — *underfitting* — the algorithm outputs similar predictions for similar data, but predictions are wrong (algorithm "miss").
- high bias, high variance — very bad algorithm. You will most likely never see this.

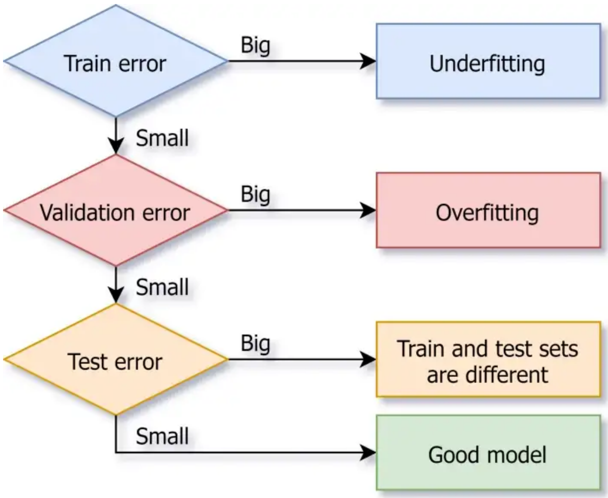
How to Detect Underfitting and Overfitting



Underfitting means that the model makes accurate, but initially incorrect predictions. In this case, *train error is large* and *val/test error is large* too.

Overfitting means that the model makes not accurate predictions. In this case, *train error is very small* and *val/test error is large*.

When we find a *good model*, *train error is small* (but larger than in the case of overfitting), and *val/test error is small* too.



- **underfitting** occurs when the model is **too simple** for data and to fix **underfitting**, you should **complicate** the model.

- **overfitting** occurs when the model is **too complex** for data and to fix **overfitting**, you should **simplify** the model.

Q. 6: What is the use of batch size and why is it used in Deep Learning?

ANSWER:

The batch size is the number of samples that will be passed through to the network at one time. Note that a batch is also commonly referred to as a mini - batch.

An epoch is one single pass over the entire training set to the network. The batch size and an epoch are not the same thing.

Number of epochs can be decided with the help of decided batch - size:

batches in epoch = training set size / batch_size

Why Use Batches?

Generally, the larger the batch size, the quicker our model will complete each epoch during training. This is because, depending on our computational resources, our machine may be able to process much more than one single sample at a time.

However, is that even if our machine can handle very large batches, the quality of the model may degrade as we set our batch larger and may ultimately cause the model to be unable to generalize well on data it hasn't seen before.

In general, the batch size is another one of the *hyperparameters* that we must test, and tune based on how our specific model is performing during training. This parameter will also have to be tested in regard to how our machine is performing in terms of its resource utilization when using different batch sizes.

Q. 7: Using the above problem statement of Fashion MNIST build a NN with different batch sizes and report the accuracy metrics (acc, val acc, loss, val loss) in a sheet?

ANSWER:

Kindly refer to the file name: 'FashionMNIST_ANN.ipynb' for the source code & 'Fashion_MNIST_Report.xlsx' for the accuracy metrics report with different batch sizes (32, 100, 200, 300, 400) in the folder.