# Multi-Layer Perceptron Learning in Tensorflow

In this article, we will understand the concept of a multi-layer perceptron and its implementation in Python using the TensorFlow library.

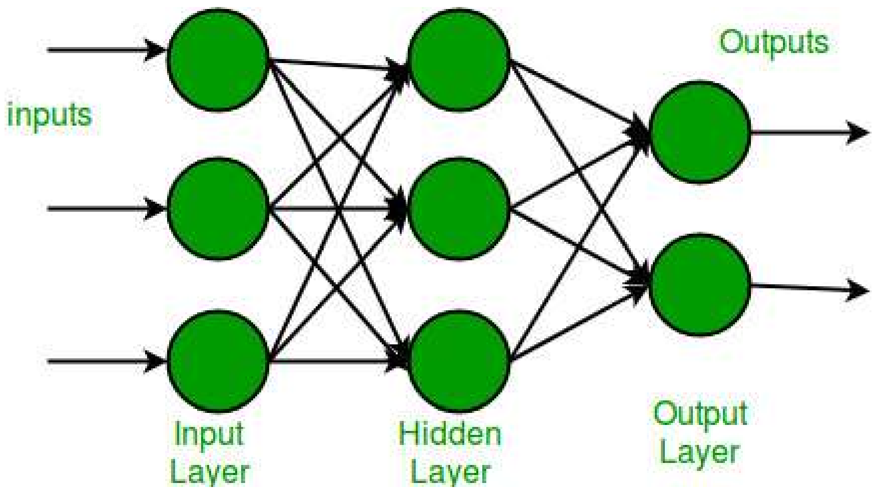
## Multi-layer Perceptron

Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.

A gentle introduction to **neural networks and TensorFlow** can be found here:

- Neural Networks
- Introduction to TensorFlow

A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.



In the multi-layer perceptron diagram above, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.

Every node in the multi-layer perception uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula.

$$(x) = 1/(1 + exp(-x))$$

Now that we are done with the theory part of multi-layer perception, let’s go ahead and implement some code in **python** using the **TensorFlow** library.

## Stepwise Implementation

**Step 1:** Import the necessary libraries.

## Python3

```
import  tensorflow as tf

import  numpy as np

from  tensorflow.keras.models  import  Sequential

from  tensorflow.keras.layers  import  Flatten

from  tensorflow.keras.layers  import  Dense

from  tensorflow.keras.layers  import  Activation

import  matplotlib.pyplot as plt
```

**Step 2:** Download the dataset.

TensorFlow allows us to read the MNIST dataset and we can load it directly in the program as a train and test dataset.

## Python3

```
(x_train, y_train), (x_test, y_test)  =  tf.keras.datasets.mnist.load_data()
```

**Output:**

> Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
>
> 11493376/11490434 [==============================] – 2s 0us/step