Neural Networks

Soumita Das
University at Buffalo
UBIT Name: soumitad
UB Person Number: 50320170
soumitad@buffalo.edu

Abstract

Neural network is a supervised classification model which mimics the structure of human brain. They are a model of interconnected nodes or neurons where one arrow denotes how the output from one node becomes an input for the next. There are several models of neural networks. The most used one are *feed-forward neural networks* and *recurrent neural networks*. In this project we mainly work with various types of *feed-forward neural networks*. Using the MNIST-Fashion data set we show how the results vary for one-hidden layer NN, multi-layer NN and convolution NN.

1 Introduction

Machine learning problems can be of two types, *supervised* and *unsupervised*. Classification that depends on the attributes and target variables of the training set, and then go on to predict class labels for unknown test samples is a category of *supervised* learning. On the other hand clustering techniques which are *unsupervised* learning groups input samples based on the homogeneity among their features.

Neural networks have been really useful in solving multi-class problems. They are classification models that have been inspired from our biological neural system in the brain. The basic unit of computation in a neural network is the neuron or node. It receives several inputs from previous nodes and computes one output. Each input is combined together using corresponding weights w(trainable hyper parameters) to produce the output.

The idea is that the weights can learn from the training data set and update in the process to give reasonable predictions. In the biological perspective, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon. This firing takes place based on the activation function that we use. Figure 1 shows the basic model of each neuron.

2 Algorithm

For a given data set with n samples and d features, let $X_i = [x_1, x_2, ...x_d]$ denote each input sample where i ranges from 1 to n. A set of parameters(or weights) $W_j = [w_1, w_2....w_d]$ and the bias b_j is tuned based on the input training set for each class j ranging from 1 to c where there are c classes. For a test sample the hypothesis is calculated as $h(X) = W_j * X + b_j$. The hypothesis is then fed into the activation function which returns a probability value between 0 to 1. Based on the probability of each class the point X is assigned to one of them.

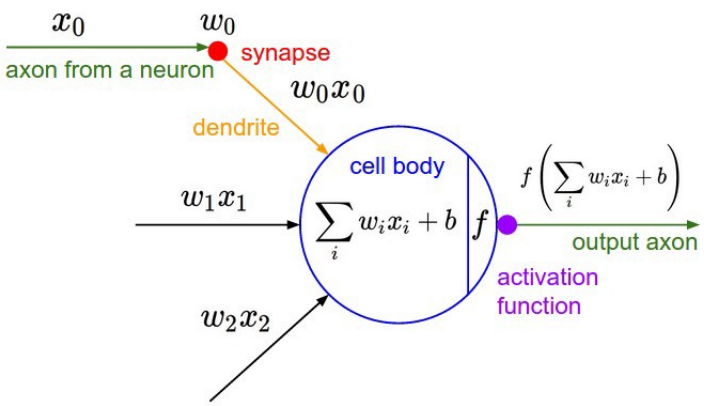

Figure 1: Neuron Architecture

2.1 Types of Activation Functions

2.1.1 Sigmoid Activation Function

Since the output may take an infinite range of values, the sigmoid function helps to range it between 0 to 1. The sigmoid activation function is:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

$\sigma(z)$ is always bounded between 0 to 1. The following figure shows a graph of the function.

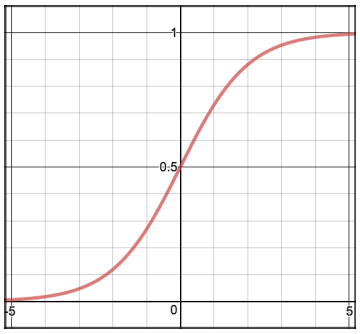

Figure 2: Plot of Sigmoid Activation Function

2.1.2 Softmax Activation Function

In case of multi-class classification often softmax function is preferred. The formula is shown below:

$$\sigma(z_j) = \frac{e^{z_j}}{\sum e^{z_i}} \tag{2}$$

where i ranges from 1 to total number of classes.

2.2 Loss Function

Instead of the Squared Error here we use the Cross Entropy for error calculation. Let the output from the activation function be a_i where a_i is a vector containing all probabilities of the X_i belonging to each class and the actual class label of a sample be one hot encoded into y_i for the i_{th} sample in the

training data then the Loss function will be calculated as :

$$L = \frac{-1}{m}\sum_{i=1}^{m}(y_i log(a_i)) \tag{3}$$

where m is the number of training samples.

2.3 Gradient Descent

The weights and the bias are updated based on the gradient descent. The following formula is used to update the weights:

$$w_{new} = w_{old} - \eta \Delta w_{old} \tag{4}$$

where η is the learning rate
The aim is to minimize the loss function. Using gradient descent we aim to find the the global minimum of the loss function. η or the learning rate must be taken care of. If taken too large it may oscillate over the global minimum, while an extremely small value may result in getting stuck at local minimum.
The following equations are used for one-hidden layer neural network:

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

Figure 3: Equations for Backpropagation

2.4 Multi-layer Feed Forward Neural Network

The architecture of the Multi-Layer Neural Network is shown below:

The Input nodes provide information from the outside world to the network and are together referred to as the "Input Layer". They represent the feature vector of the training samples say $X_i = [x_1, x_2, ...x_d]$ denote each input sample where i ranges from 1 to n.

A collection of hidden nodes forms a "Hidden Layer". While a feedforward network will only have a single input layer and a single output layer, it can have one or multiple Hidden Layers. For example from 1 hidden layer X_i vector will be multiplied with a set of weights say $W1_j$ where j ranges from 1 to h (the number of hidden nodes). Thus the output from each of the hidden node will be the sigmoid activation of $X_i * W1_j + b1_j$.

The Output nodes are collectively referred to as the "Output Layer" and are responsible for computations and transferring information from the network to the outside world. The output from the previous hidden layer $A_i = [a_1, a_2...a_j]$ is multiplied with another set of weights $W2_k$ where k ranges from 1 to c (number of classes). Thus the output from each output node will be the softmax activation of $A_i * W2_k + b2_k$.

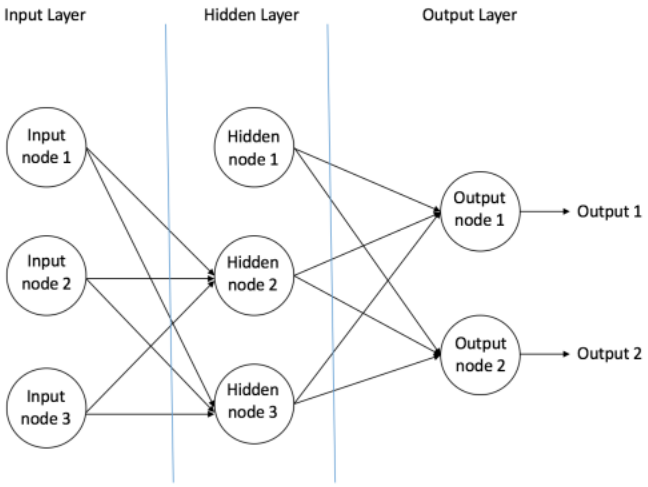

Figure 4: Multi-Layer Neural Network

2.5 Convolution Neural Network

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The role of CNN is to reduce the image to a set of features which are critical from decision making. This is done using two main operations:

2.5.1 Convolution operation

Here in the convolved image each pixel is determined by a convolution operation between the kernel and its neighbouring cells. The convolution operation is shown in the figure below:

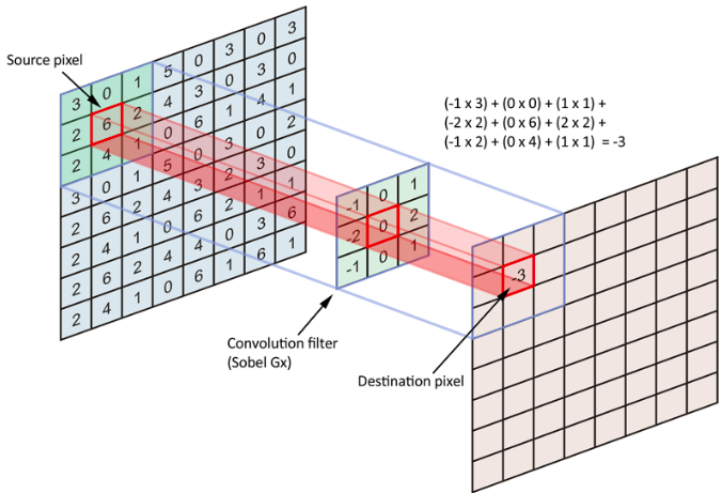

Figure 5: Convolution Operation

2.5.2 Pooling

The pooling operation reduces the dimension of the image to only its critical features. There can be two types of pooling: 1. Max pooling : Take the maximum value of the block of cells. 2. Average Pooling : Take the average value of the block of cells.
The operations are shown below.

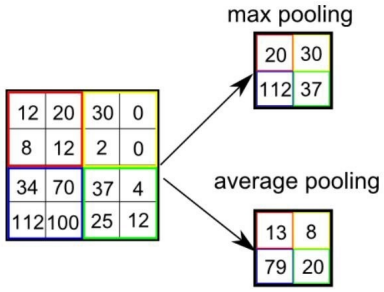

Figure 6: Pooling Operation

The final architecture of convolution neural network is shown in figure 6.

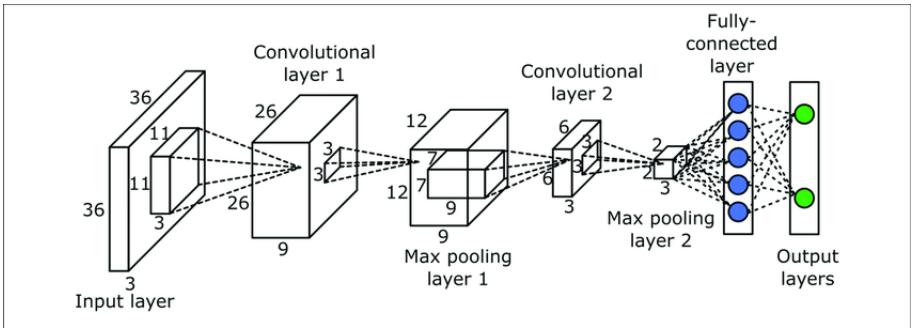

Figure 7: CNN Architecture

3 Experimental Setup

3.1 Data set

For training and testing of our classifiers, we will use the Fashion-MNIST dataset. The Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns.

3.2 Evaluation Metrics

To calculate the effectiveness of the algorithm, we use accuracy, precision and recall.

$$Accuracy = \frac{t_p + t_n}{t_p + f_p + t_n + f_n} \tag{5}$$

$$Precision = \frac{t_p}{t_p + f_p} \tag{6}$$

$$Recall = \frac{t_p}{t_p + f_n} \tag{7}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8}$$

where t_p denotes true positive, t_n denotes true negative, f_p denotes false positive and f_n denotes false negative.

3.3 Results and Discussions

For the first part of the task where the neural network was implemented with one hidden layer, the training and validation loss for every epoch is shown in Figure 8.

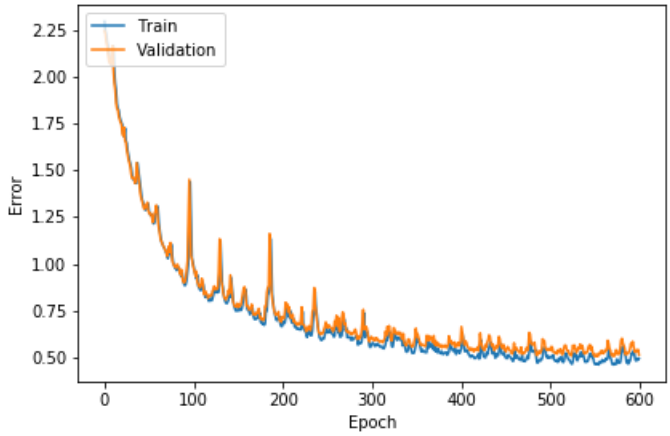

Figure 8: Training and Validation loss for 1 hidden layer NN

Following this hyper-parameters like epochs and hidden nodes were varied. It was observed that initially there was a sharp increase in the values of precision, recall and f-1 score, however on further increase there were not much of a difference. This behaviour has been shown in Figures 9 and 10 respectively.

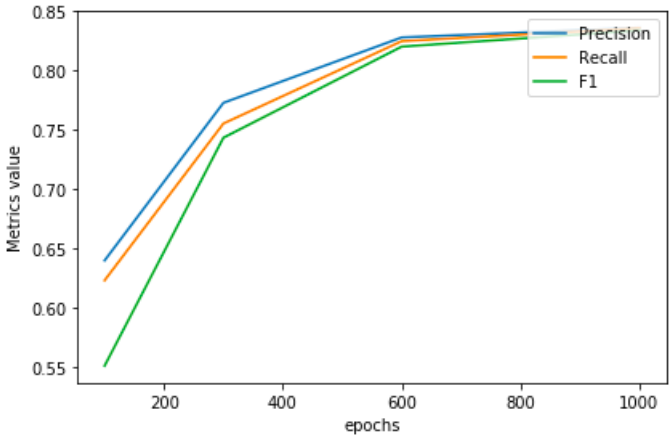

Figure 9: Metrics variation with number of epochs

For task 2, multi-layer neural networks were implemented using Keras. The model performances for each epoch in terms of training and validation loss and accuracy are shown in Figures 11 and 12 respectively. The number of hidden layers were also varied and on comparing Figures 12 and 13 we see that the second model with 3 hidden layers has better validation or test accuracy in the end than that of the model with 2 hidden layers.

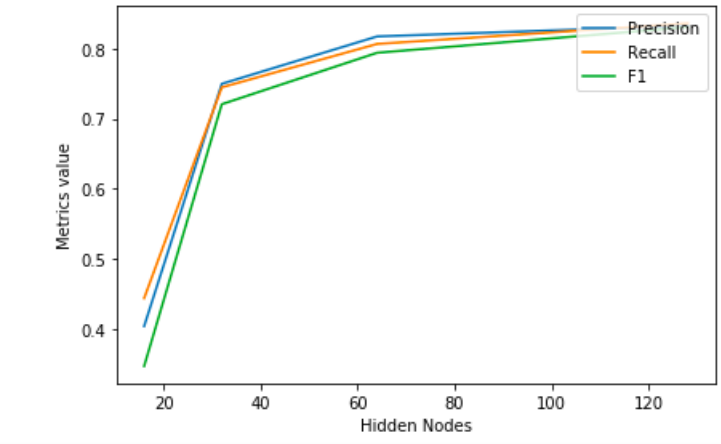

Figure 10: Metrics variation with number of hidden nodes

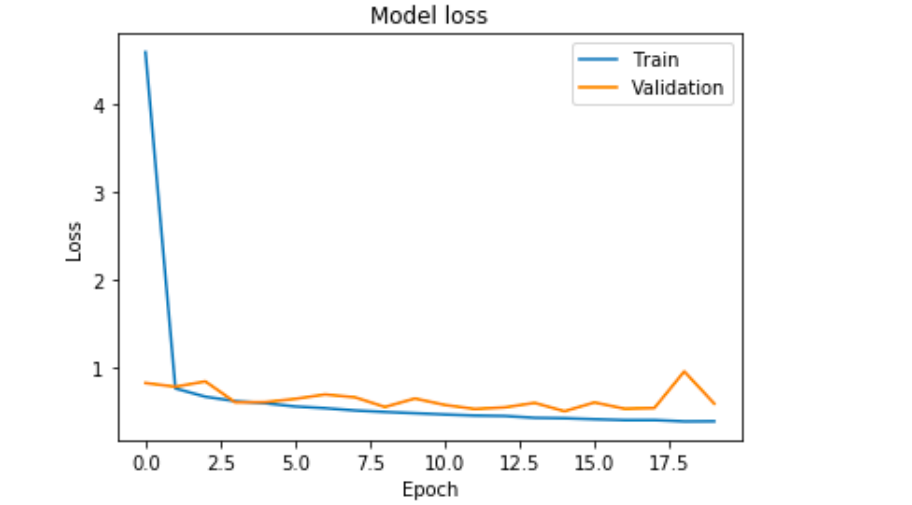

Figure 11: Multilayer NN with 2 hidden layers

Hyper-parameters are varied for the multilayer NN and we see that as the epochs and hidden nodes increases the accuracy increases too. (Figures 14 and 15)

For task 3 we implemented Convolution Neural Network using 2 layers of convolution, 1 max pooling and 1 hidden layer. Following this we varied hyper-parameters like epochs and hidden nodes and the results are shown in Figures 16 and 17 respectively. Figure 18 shows the model development with every epoch for CNN.

4 Conclusion and Future Work

In this work the fashion MNIST dataset was classified using one-hidden layer neural network, multilayer neural network and lastly convolution neural network. The values of epochs and hidden layer nodes and number of hidden layers were varies and results were compared. Keras was used for implementation of the second and third task. In future I plan to automate the process so that we can get the optimum values of the hyper parameters that would give us the best results for the data set. Also, a regularization method can be added to avoid overfitting.

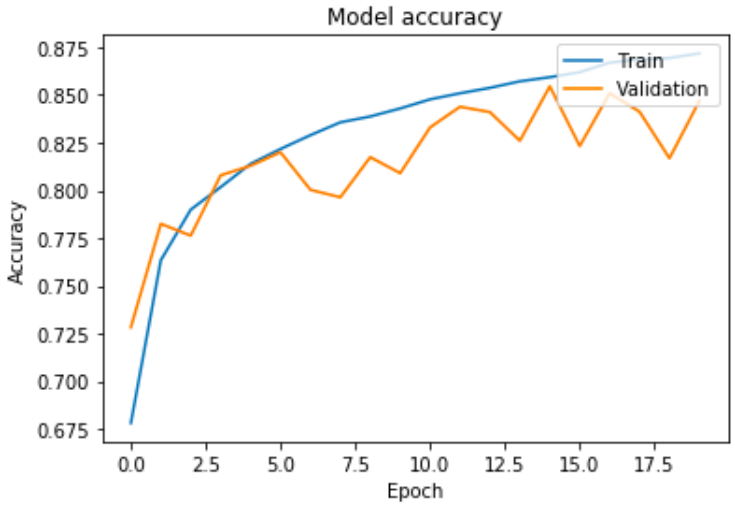

Figure 12: Multilayer NN with 2 hidden layers

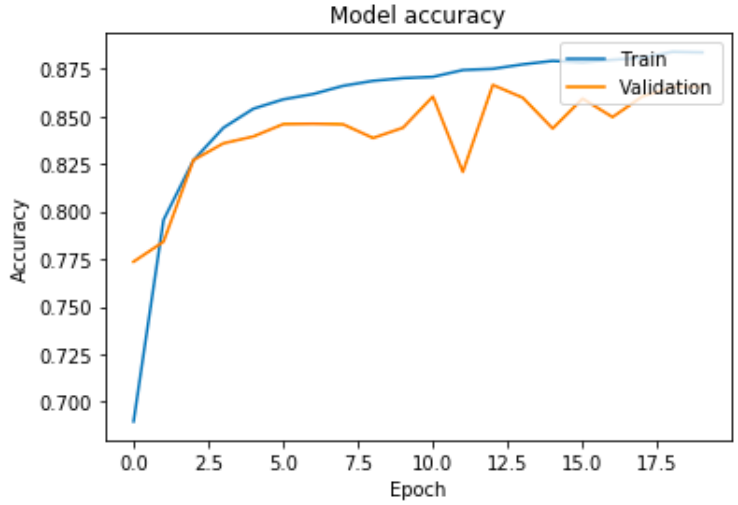

Figure 13: Multilayer NN with 3 hidden layers

5 References

[1] Pattern Recognition and Machine Learning by Christopher M. Bishop
[2] Neural networks and Deep Learning by Andrew Ng at http://cs229.stanford.edu/notes/cs229-notes-deep_learning.pdf

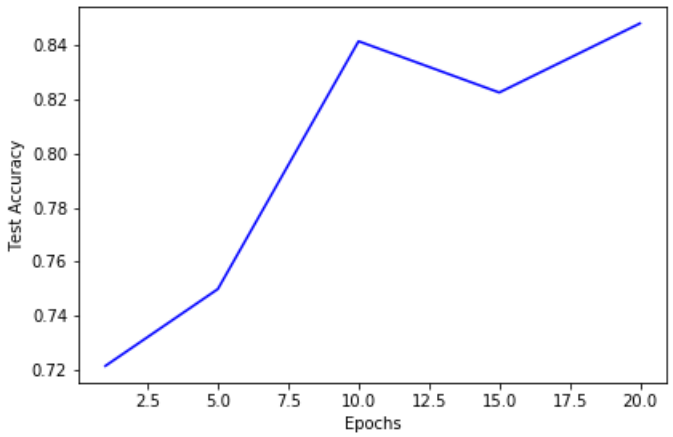

Figure 14: Accuracy variation with number of epochs for NN

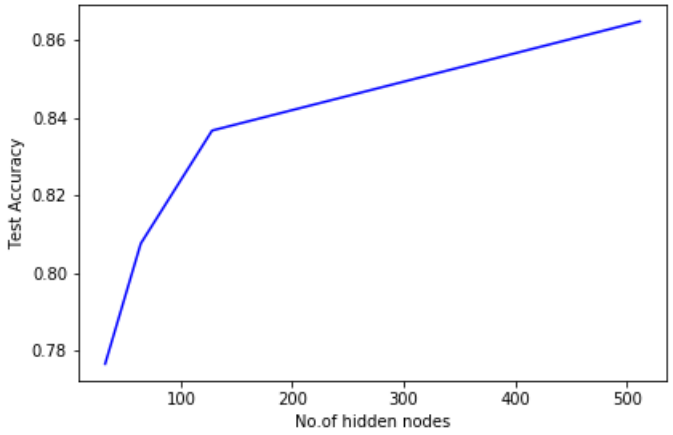

Figure 15: Accuracy variation with number of hidden nodes for NN

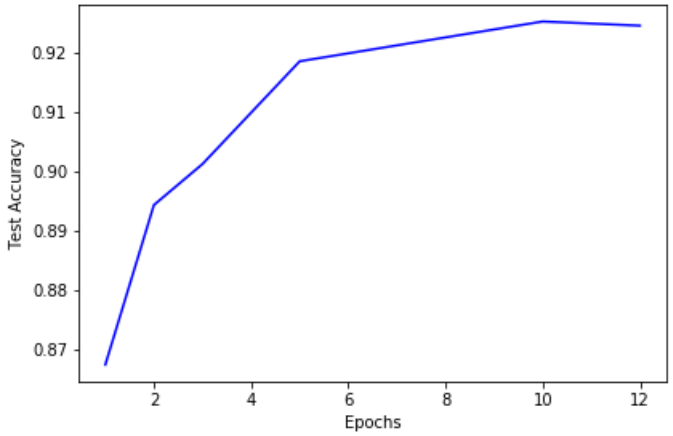

Figure 16: Accuracy variation with number of epochs for CNN

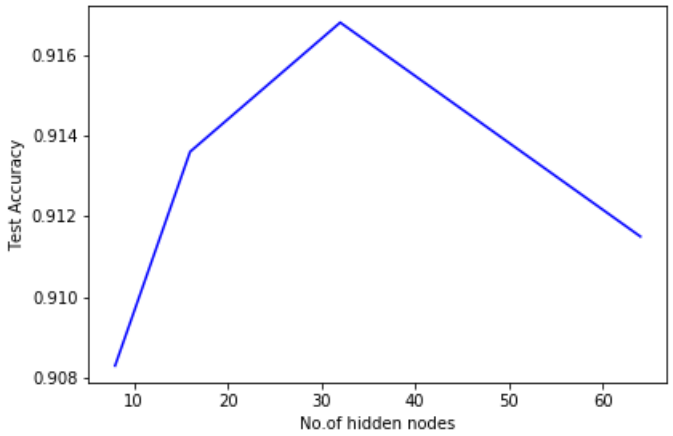

Figure 17: Accuracy variation with number of hidden nodes for CNN

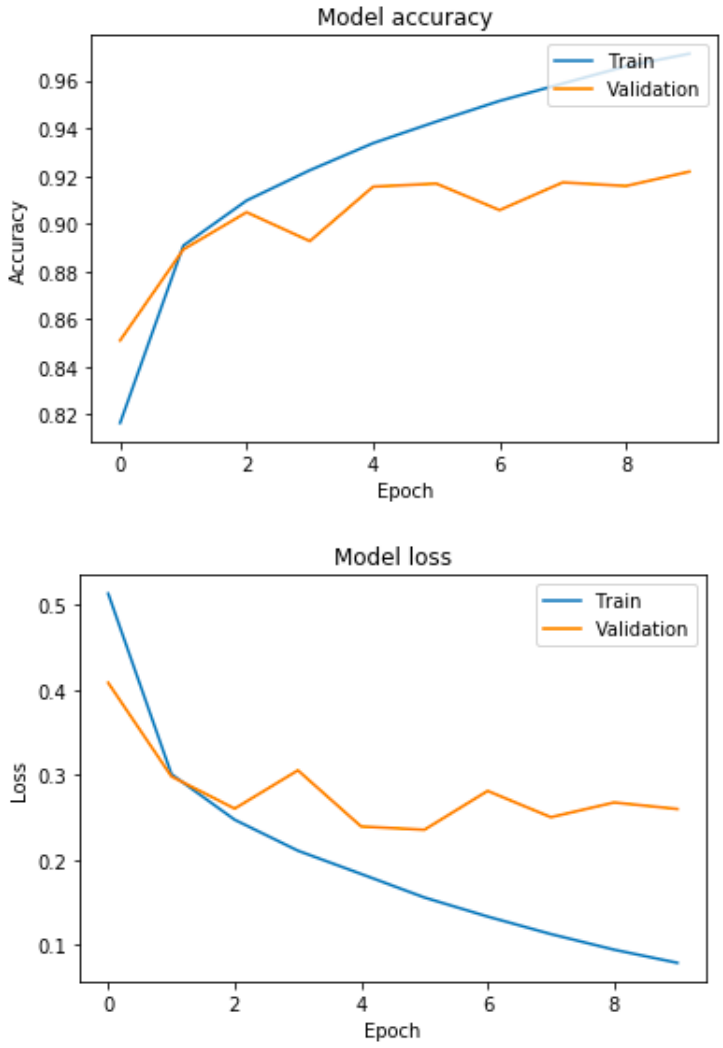

Figure 18: CNN model