Project 2: Neural Networks

Sambo Dutta
University at Buffalo
UBIT Name: sambodut
UB PERSON NUMBER: 50318667
sambodut@buffalo.edu

Abstract

Neural Networks are a set of statistical algorithms which mimics the human brain to determine the relationship between a set of features. Each input has a weight associated to it, which adjust itself based on the training model.It is mostly used in the field of finance and recommendation to predict a target value based on a series of given data.In this project we have implemented three versions of neural networks- Single Hidden Layer Neural Network,Multi-layer Neural Network,Convolution Neural Network.Models were trained based on the Fashion-MNIST Data Set.

1 Introduction

Neural networks are a set of algorithms based on the working of the human brain. These networks are programmed in a way to determine the relationship between the features and find the pattern within the dataset. It is one of the most widely used tools in the field of the machine learning to predict or to classify information.

The basic unit of computation in a neural network is the neuron , often called the node. It receives input from some other nodes, or from an external source and computes an output. These inputs have weights(w) associated with them based on the importance of each of the features. The node applies a function to the weighted sum of its inputs and predicts an output. This output is later matched with the original result to check the correctness of the algorithm and the weights are adjusted based on the output of the model.

In this project we have implemented three variants of neural network, one hidden layer neural network, multi-layer neural network and convolution neural network. The Fashion-MNIST dataset has been used to the test the efficiency of our models and the results are discussed in the later sections.

2 Architectures

2.1 Multi-Layer Neural Network

In the Multi-Layer Network we have the input nodes,where there is no computation. Here the data is taken as input one by one and together they are known as the input layer which represents the feature vector of the training points $X_i = [x_1, x_2, ...x_d]$.

All the input nodes are associated with weights based on the importance of the features.Next we have the hidden nodes where actual computations take place.The output from these nodes will be the sigmoid activation of $X_i * W1_j + b1_j$. Here we have a user defined no. of neurons.

This hidden layer is again associated with a second set of weights $W2_k$ where k ranges

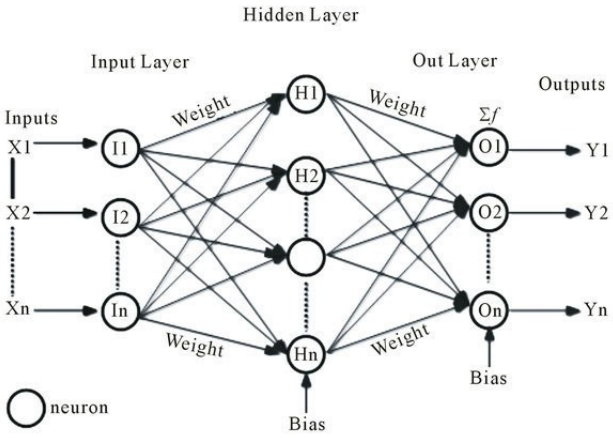

Figure 1: Multi-Layer Neural Network Architecture

from 1 to the number of classes which forms the input of the Output Layer. All the output of this layer is again multiplied with another set of weights $W2_k$. Thus the output from each output node will be the softmax activation of $A_i * W2_k + b2_k$.

2.2 Convolution Neural Network

CNN is mainly used for images where it takes in account all the features of the image and narrows down the set of features to only those features which are more important when compared to others.

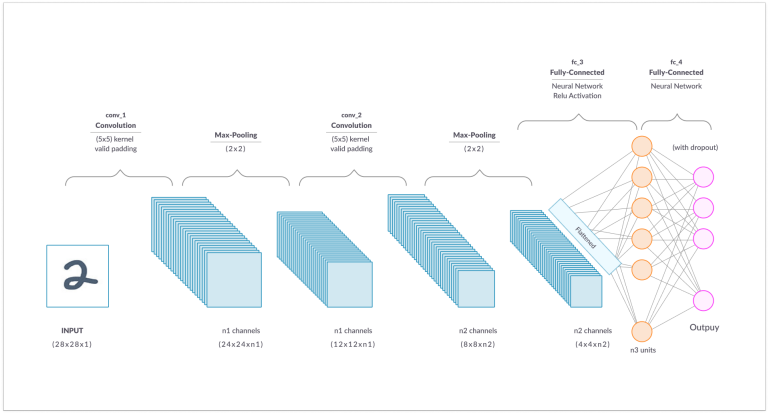

Figure 2: CNN Architecture

There is convolution operation results in a convolved image which is determined by performing an operation with the image matrix and the kernel.Figure 3 illustrates this operation.

After the convolved image is determined we use a pooling operation to reduce the image dimension to the most important features of the image. We can use either of max pooling or average pooling as shown in Figure 4.

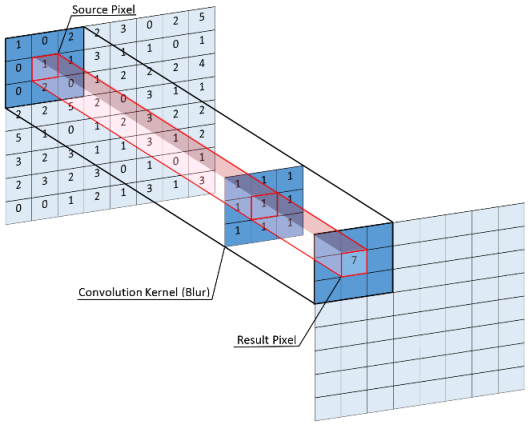

Figure 3: Covolution operation

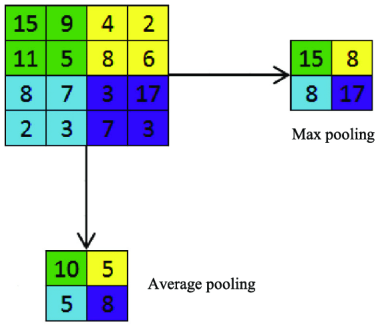

Figure 4: Pooling

3 Activation Functions

3.1 Sigmoid Activation Function

A sigmoid function is a S-curved function which is mainly used to map variables in the range 0 to 1. The following equation is mostly used as the sigmoid activation function:

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

On doing a dot product between the weight vector and the training input vector we might get a very large value, to avoid this we are using the sigmoid activation function to get the values in a smaller sub-range.

3.2 Softmax Activation Function

The softmax function is defined below. It has been expermintally found that this function performs better for multi-class problems.

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

where i ranges from 1 to total number of classes

4 Tuning Parameters using Gradient Descent

Our goal is to minimize the loss function. y is the matrix containing the actual class labels for all the training samples. X is the training data set of $m * d$ size where d is the number of features. a is

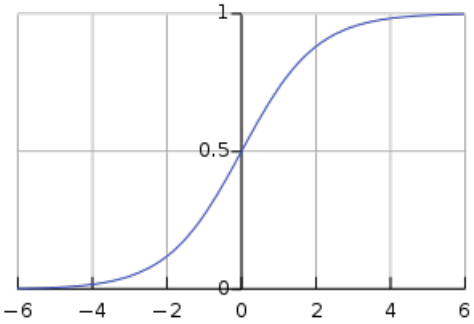

Figure 5: Plot of Sigmoid Activation Function

the matrix containing the predicted class labels for all the training samples. In this case we use cross entropy loss. The equation is given below:

$$L = \frac{-(ylog(a) + (1-y)log(1-a))}{m} \tag{3}$$

where m is the number of training samples.

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 learning rate value should not be very large because then the weight would not get its optimum value. Also if the learning rate is too small there is a high chance we might end up with a local minima. The backpropagation equations for one hidden layer neural networks are shown in the figure below:

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

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

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

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

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

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

5 Experiments

5.1 Dataset

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. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

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. The first column consists of the class labels (see above), and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

5.2 Normalization

Before the splitting the dataset into training,testing and validation,we are normalizing the dataset.The dataset has 784 unique features in which individual features may have in its own ranges.While some of these features may have a very low range, others which have a higher range might overpower its weightage and might become a more dominating feature just because of its higher range. So in order to avoid these kind of situations we are normalizing the data in the range 0 to 1 for each particular feature, so that each features have equal weightage.We have used the equation below to normalize each feature:

$$x_{new} = \frac{x_{old}}{255} \tag{5}$$

since 255 is the maximum value of a pixel.

5.3 Evaluation Metrics

In order to check the efficiency of the model we have used accuracy,precision,recall and f-1 score.

$$Accuracy = \frac{t_p + t_n}{t_p + f_p + t_n + f_n} \tag{6}$$

$$Precision = \frac{t_p}{t_p + f_p} \tag{7}$$

$$Recall = \frac{t_p}{t_p + f_n} \tag{8}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{9}$$

6 Results

We have evaluated the three models using different hyper-parameters.For the single layered network we have varied the training and validation over a range of epochs ranging from 0 to 1000. The results are shown in the graph below.

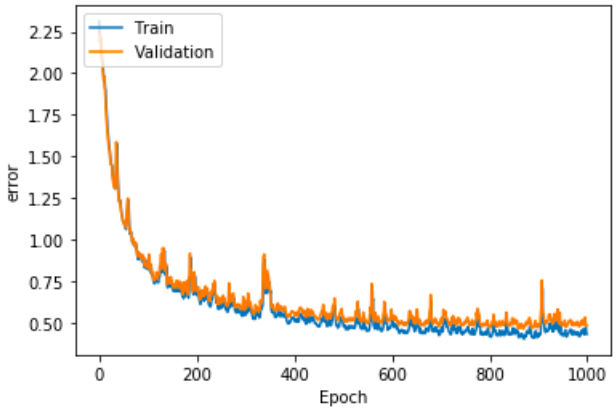

Figure 6: Training Vs Validation for Single Layer

We have also evaluated the model using the metrics precision,recall and f1 score to see performance of the model on increasing the hidden nodes and the epoch range.

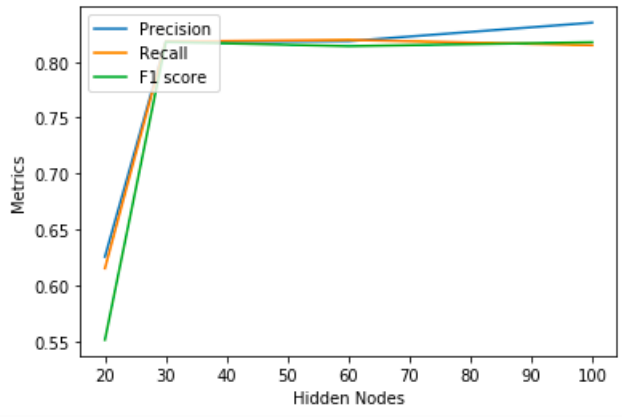

Figure 7: Variation with Hidden Nodes for Single Layer

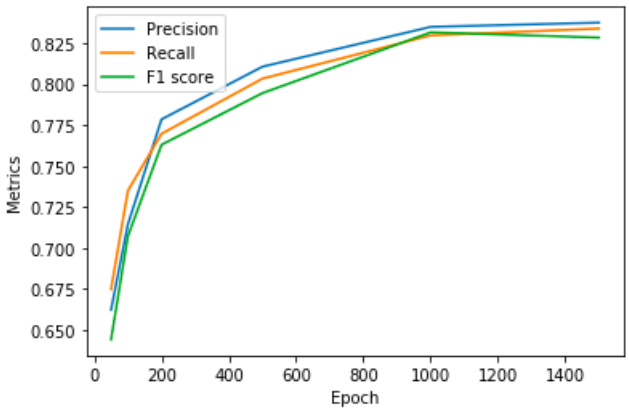

Figure 8: Variation with Epoch for Single Layer

In the next section we have calculated the accuracy on increasing the nodes and epoch for the multi-layered network. We see that the accuracy increases on increasing these two metrics in Figure 9 and Figure 10.

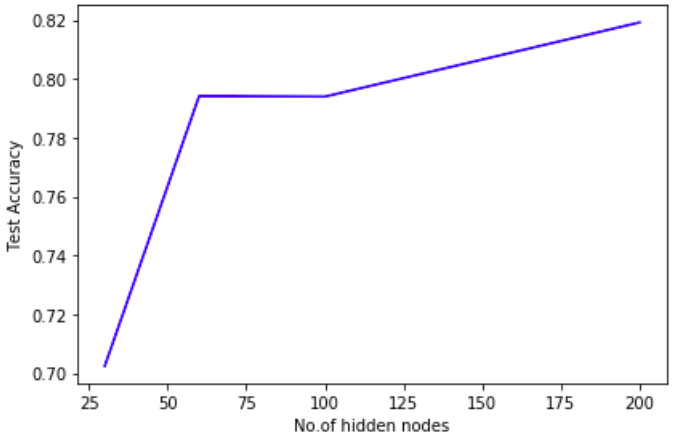

Figure 9: Variation of Accuracy with nodes for Multi-Layered Network

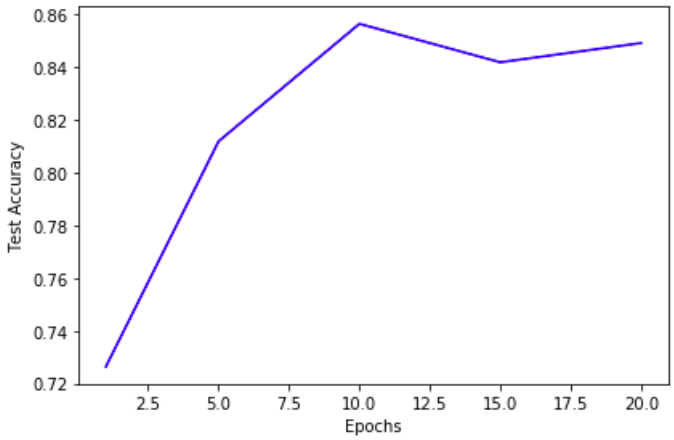

Figure 10: Variation of Accuracy with Epoch for Multi-Layered Network

The performance of the Multi-Layered Network are depicted with the graphs Figure 11 Figure 12.

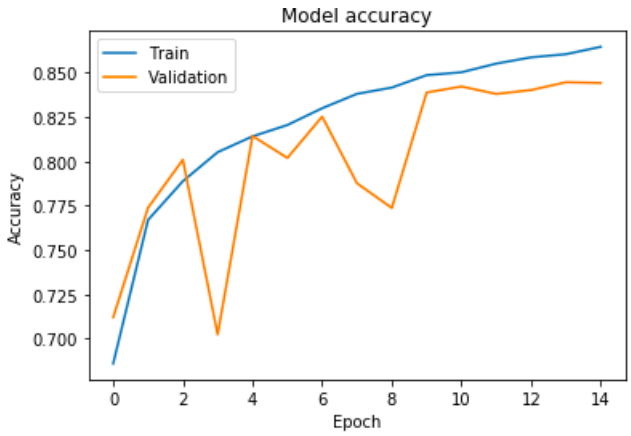

Figure 11: Model Accuracy of Multi-Layered Network

In the last section we have evaluated the CNN model.The performance of the model is depicted in the figures 13 and 14. Figure 15 and 16 on the other hand shows how the accuracy varies with the no of hidden nodes and epoch.

7 Conclusion

The experiment above demonstrated a detailed analysis of three different models- Single-Layered Neural Network, Multi-Layered NN and CNN on the Fashion-MNIST dataset.The models were to classify an object in either of the 10 given classes.
The results show the performance of the three models based on the hyper-parameters. We can make an conclusion that the accuracy of all the three models gradually increase with the hidden nodes and the epoch range.
Finally, it also proves that a more complex model like CNN outperforms a simple neural network with 1 hidden layer. This comparison has been evident in the experiments.

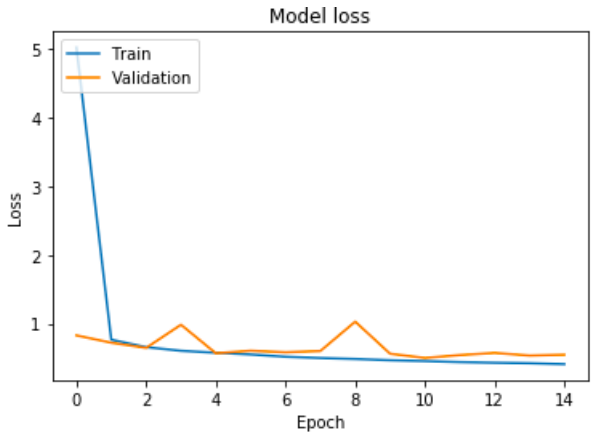

Figure 12: Model Loss of Multi-Layered Network

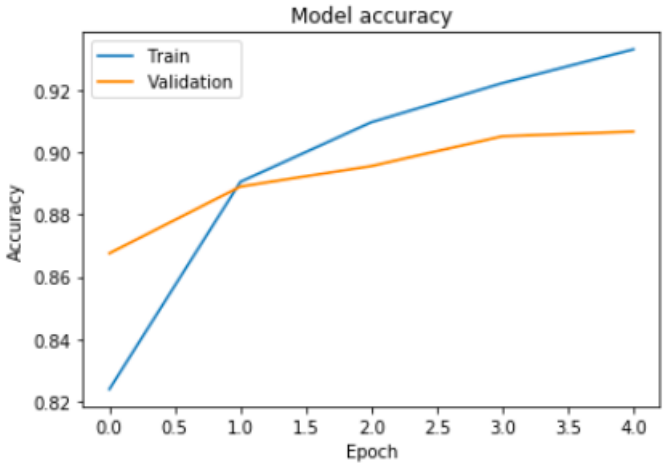

Figure 13: Model Accuracy of CNN

8 References

1. Fashion MNIST Dataset
https://github.com/zalandoresearch/fashion-mnist
2. Pattern Recognition and Machine Learning (Information Science and Statistics) by Christopher M. Bisho3
3.https://www.investopedia.com/terms/n/neuralnetwork.asp
4.https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

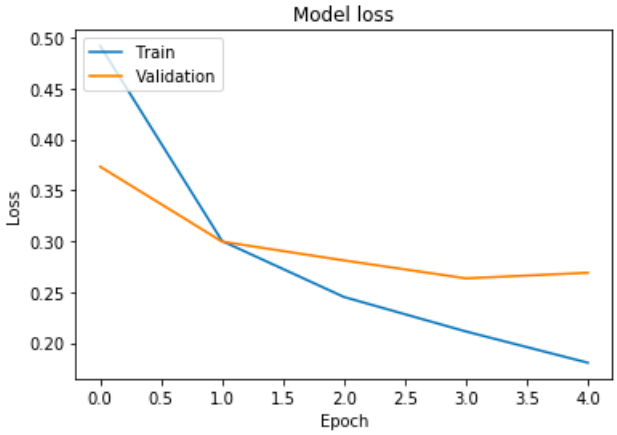

Figure 14: Model Loss of CNN

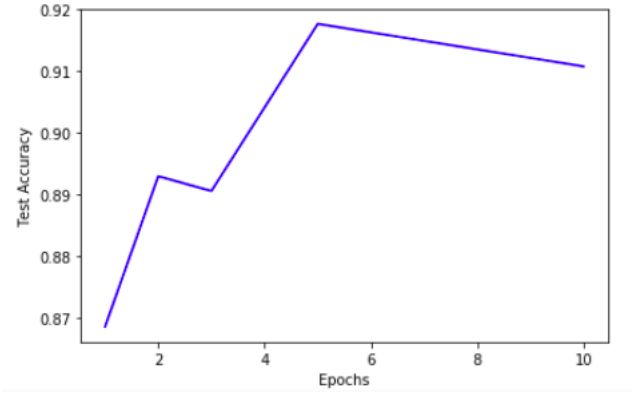

Figure 15: Accuracy vs Epoch for CNN

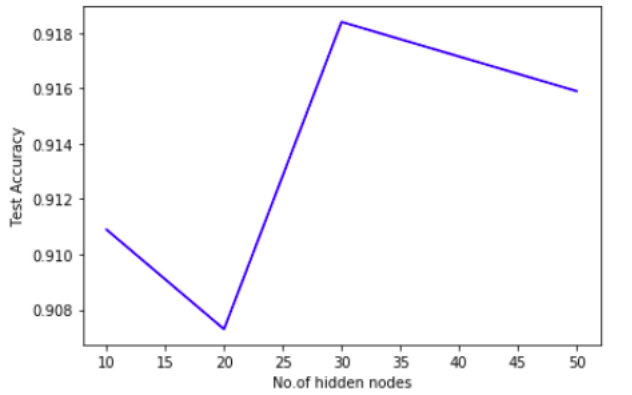

Figure 16: Accuracy vs Hidden Nodes for CNN