CSE 574: Introduction to Machine Learning
PROJECT 2 REPORT

Submitted by:
Vishva Nitin Patel
UB Person Number: 50318625
Email ID: vishvani@buffalo.edu

ABSTRACT:

Presented in this report is the utilization of neural network and convolutional neural network for the task of classification. The classification task will be that of recognizing an image and identify it as one of ten classes. I have implemented the classification program(s) in python from scratch and findings of the same along with the related code snippets are attached herewith. Steps involved were: (1). Build a Neural Network with one hidden layer to be trained and tested on Fashion-MNIST dataset. (2). Build multi-layer Neural Network with open-source neural-network library Keras. (3). Build Convolutional Neural Network (CNN) with open-source neural-network library, Keras. In the end I evaluated the results obtained by each of the classifiers (Single layer Neural Network, Multi-Layer neural network and CNN).

1. Introduction:

Neural networks are a set of algorithms, modeled in replication to the human brain, and are designed to recognize patterns. The patterns they recognize are numerical, contained in vectors, and all real-world data, be it images, sound, text or time series, must be translated into these numerical patterns. Neural networks help us cluster and classify. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on.

Components:

- Neurons - The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image.
- Connections and weights - Each connection provides the output of one neuron as an input to another neuron, where the connections are assigned a weight that represents its relative importance.
- Propagation function - The propagation function computes the input to a neuron from the outputs of its predecessor neurons and their connections as a weighted sum.
- Hyper-parameter - A hyper-parameter is a parameter whose value is set before the learning process begins. The values of parameters are derived via learning. Examples of hyper-parameters include learning rate, the number of hidden layers and batch size.
- Learning - Learning is the adaptation of the network to better handle a task by considering sample observations. Learning involves adjusting the weights (and optional thresholds) of the network to improve the accuracy of the result.

- Learning rate - Size of the corrective steps that the model takes to adjust for errors in each observation. A high learning rate shortens the training time, but with lower ultimate accuracy, while a lower learning rate takes longer, but with the potential for greater accuracy.
- Back-propagation - Method to adjust the connection weights to compensate for each error found during learning. The error amount is effectively divided among the connections. Technically, back-propagation calculates the gradient (the derivative) of the cost function associated with a given state with respect to the weights.

2. Dataset definition:

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 28×28 grayscale image, associated with a label from 10 classes. Fashion-MNIST is intended to serve as a direct drop-in replacement of the original MNIST dataset for bench marking machine learning algorithms.

Dataset Partitioning:

- Training Data: Used for learning and hence making predictions about the testing data. We take 60,000 training examples in this project.
- Testing Data: Used for depicting how accurate our model is and how well it performs. Predictions made about the Testing Data show how well our model has been trained. We take 10,000 testing examples in this project.

3. Pre-Processing:

The dataset is standardized before implementation of Neural networks in following ways:

3.1 One-hidden Layer Neural Network

- Normalize the data to keep the gradient in the range of [0,1].
- One hot encoding on the training and testing labels. The output will be a one-dimensional vector of size with each element representing "probability score".
- Re-shape and shuffle the data set because same category images are placed together and for effective training the data should be scattered.

3.2 Multi-layer Neural Network

- Normalize the data to keep the gradient in the range of [0,1].
- Re-shape and shuffle the data set because same category images are placed together and for effective training the data should be scattered.

3.3 Convolutional Neural Network

- Normalize the data to keep the gradient in the range of [0,1].
- One hot encoding on the training and testing labels. The output will be a one-dimensional vector of size with each element representing "probability score".
- Re-shape and shuffle the data set because same category images are placed together and for effective training the data should be scattered.

4. Architecture:

4.1. One Hidden Layer Neural Network:

A single hidden layer neural network consists of 3 layers: input, hidden and output.

The input layer has all the values form the input, the hidden layer is where most of the calculations happens, every Perceptron unit takes an input from the input layer, multiplies and add it to initially random values. This initial output is not ready yet to exit the perceptron, it has to be activated by a function, in this case a Relu function. The last and third layer is the output layer, it takes all the previous layer Perceptrons as input and multiplies and add their outputs to initially random values, then gets activated by a Sigmoid function.

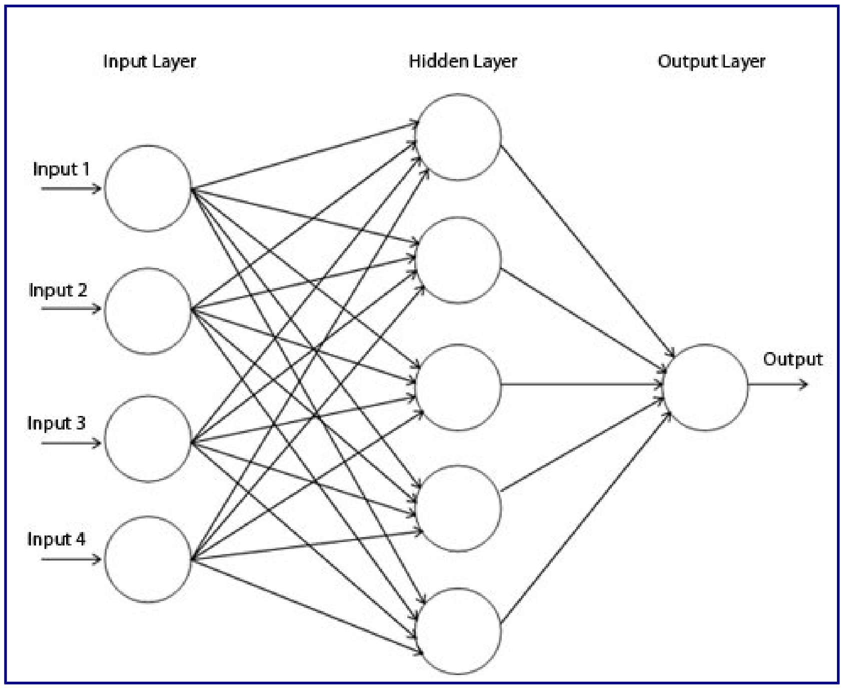


Fig.4.1.1 – 3 Layers of Neural Networks

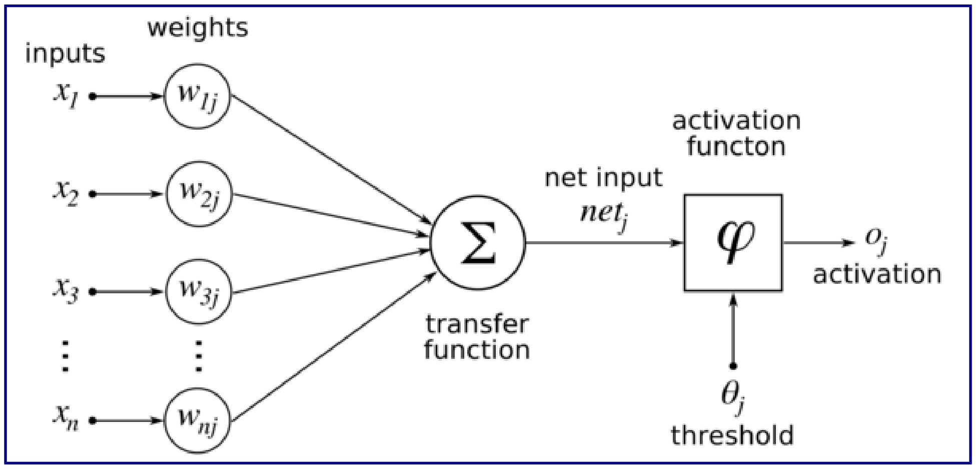


Fig.4.1.2 – Working of Neural Networks

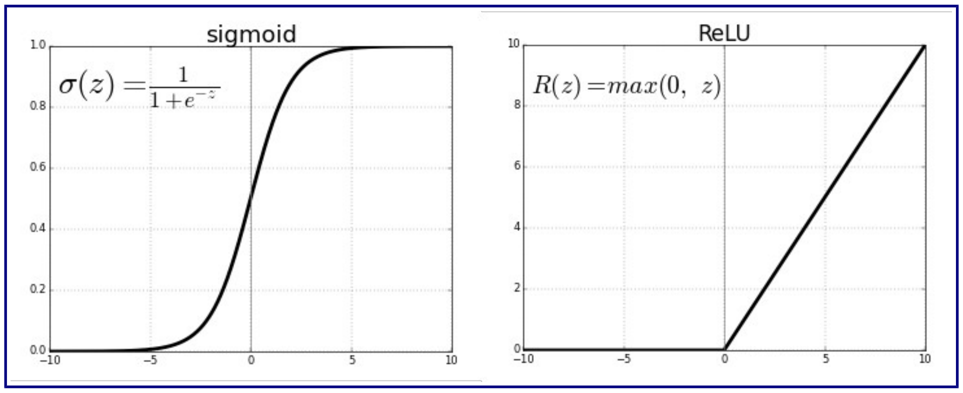


Fig.4.1.3 – Sigmoid and ReLU functions

Tuning Hyper-parameters:

Learning Rate:	**Number of iterations:**	**Cost:**	**Accuracy:**
0.1	500	0.9892	0.68
0.25	600	0.6894	0.75
0.5	600	0.6065	0.78

4.2. Multi-Layer Neural Network:

Multi-layer networks solve the classification problem for non linear sets by employing hidden layers, whose neurons are not directly connected to the output. The additional hidden layers can be interpreted geometrically as additional hyper-planes, which enhance the separation capacity of the network. Figure below shows typical multi-layer network architecture.

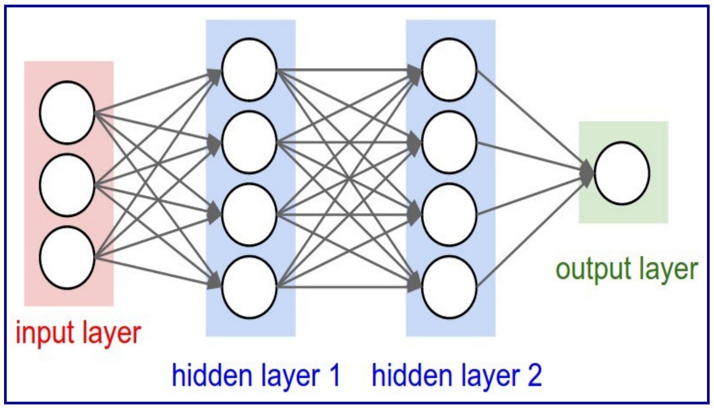


Fig. 4.2.1 – Multi Layer Neural Network

The training occurs in a supervised style. The basic idea is to present the input vector to the network; calculate in the forward direction the output of each layer and the final output of the network. For the output layer the desired values are known and therefore the weights can be adjusted as for a single layer network; in the case of the BP algorithm according to the gradient decent rule.

To calculate the weight changes in the hidden layer the error in the output layer is back-propagated to these layers according to the connecting weights. This process is repeated for each sample in the training set. One cycle through the training set is called an epoch. The number of epochs needed to train the network depends on various parameters, especially on the error calculated in the output layer.

Tuning Hyper-parameters:

Layers:	**Iterations:**	**Cost:**	**Accuracy:**
3	5	29.0767	89.3416
3	20	18.7846	93.5283
6	5	28.6484	89.4466
6	20	19.4242	93.3816

4.2. Convolutional Neural Network:

A Convolutional Neural Network 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 pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these filters/characteristics. A CNN is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and re usability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The role of the CNN is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

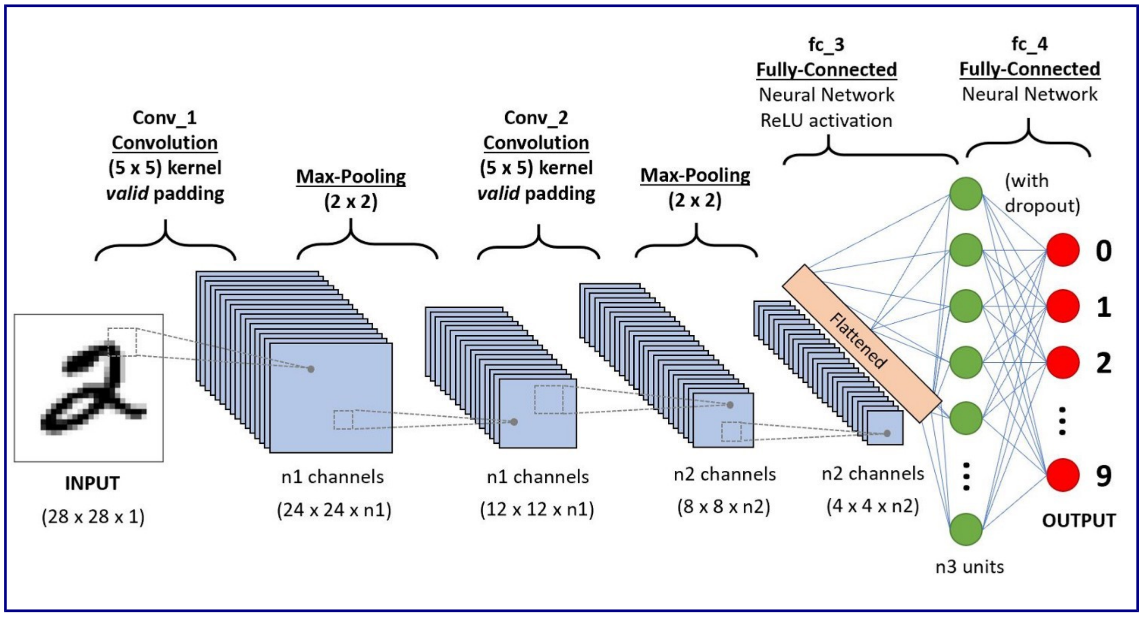


Fig. 4.2.1 – Working of Convolutional Neural Networks

Tuning Hyper-parameters:

Number of Iterations:	**Loss:**	**Accuracy:**
2	0.4761	0.8316
5	0.35	0.8742
7	0.3089	0.8897

Here I have used **keras optimizer Adam** which incorporates the Deep Learning Adam Optimization Algorithm. Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights based on training data.

5. Results:

The best choice of hyper-parameters (learning rate, number of iterations,number of layers) give the highest accuracy and the least cost. The values for best hyper-parameters and their respective cost and accuracy for each classifier is as follows:

- <u>One Hidden Layer Neural Network:</u> With Learning Rate = 0.5, and 600 iterations, cost comes up to 60.65% and accuracy is 78%. I have kept a higher learning rate because the one hidden layer approach is not as effective and gives higher cost with low accuracy when the learning rate is low.

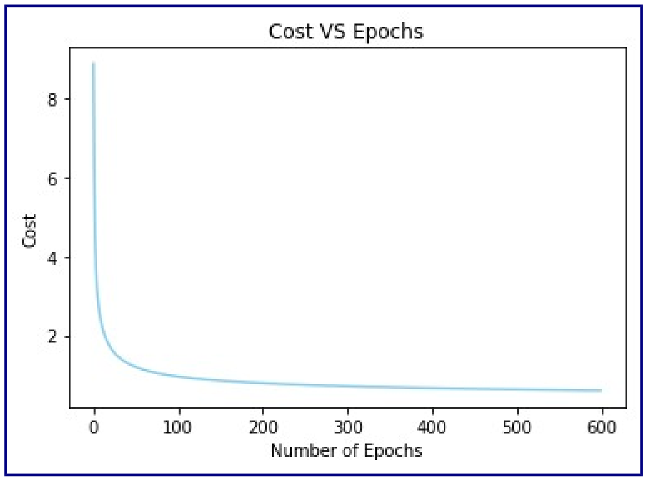
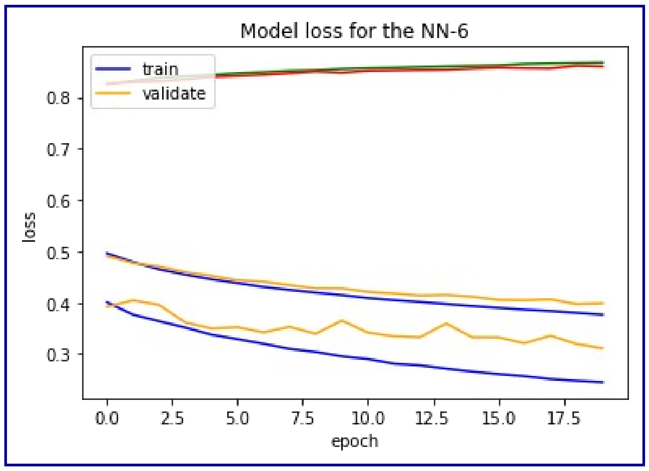


Fig.5.1 – Cost Vs Iterations

- <u>Multi-Layer Neural Network:</u> With 3 layers, and 20 iterations, cost comes down to 18.7846% and Accuracy comes up to 93.5283. This is the best accuracy and cost pair.



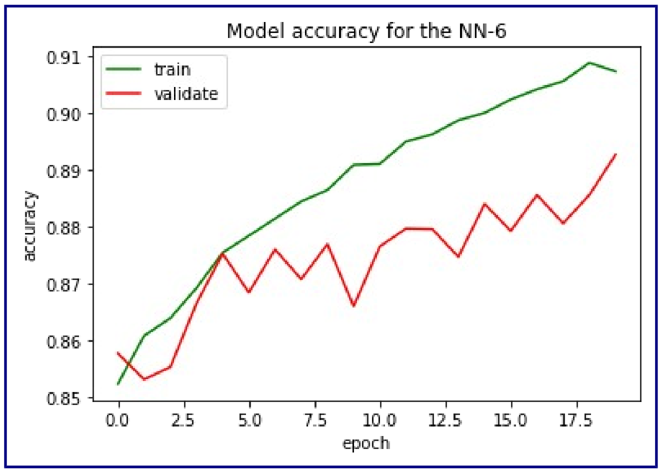


Fig. 5.2. – Loss Vs. Iterations (Layers = 6)

Fig. 5.3. - Accuracy Vs. Iterations (Layers =6)

- Convolutional Neural Network: Since I have used keras Optimizer – Adam, it auto – optimizes the learning rate to give the best accuracy, and hence with an increase in iteration a drop in cost and improved accuracy is obtained. In 7 iterations the cost comes down to 0.3089 and the accuracy comes up to 0.8897, which is fair enough and in case we increase the iterations, the accuracy will not dramatically increase.

6. Conclusion:

After tuning the hyper-parameters as depicted in the hyper-parameter tuning tables associated to each classifier, the results shown in the Result section were classified as the best outcomes from the set of hyper-parameters I tested.
CNN was able to provide highest accuracy in the least number of iterations but it's time complexity is quite significant.

7. References:

(1) https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6
(2) https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
(3) https://medium.com/@ipylypenko/exploring-neural-networks-with-fashion-mnist-b0a8214b7b7b
(4) https://stackabuse.com/creating-a-neural-network-from-scratch-in-python-adding-hidden-layers/
(5) https://gist.github.com/cristiandima/6382bd099eb64a30929b87784bdfa2b3
(6) https://www.kaggle.com/accepteddoge/fashion-mnist-with-numpy-neural-networks
(7) https://www.tensorflow.org/tutorials/keras/classification
(8) https://github.com/justanothergirlwhocodes/TensorFlow-NNs/blob/master/NN_MNIST_depth.ipynb
(9) https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/
(10) https://towardsdatascience.com/deep-neural-networks-from-scratch-in-python-451f07999373
(11) https://www.pyimagesearch.com/almost-finished-code/
(12) https://jonathanweisberg.org/post/A%20Neural%20Network%20from%20Scratch%20-%20Part%201/
(13) https://skymind.ai/wiki/neural-network
(14) https://en.wikipedia.org/wiki/Artificial_neural_network
(15) https://www.nicolamanzini.com/single-hidden-layer-neural-network/
(16) https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/
(17) https://www.teco.edu/~albrecht/neuro/html/node18.html
(18) https://towardsdatascience.com/neural-networks-from-scratch-easy-vs-hard-b26ddc2e89c7