Neural Network and Convolutional Neural Network for the task of classification on g Fashion-MNIST Dataset

Akash Kumar Roy
akashroy@buffalo.edu
Person Number: 50316991
University at Buffalo, State University of New York

Abstract: This is a report for a single layer Neural Network, multi-layer Neural network and convolution neural network model, performed on Fashion-MNIST dataset. Here in the first part we are building a single hidden layer neural network to train our data. This is implemented in python from scratch. In the second part we are building a multi-layer Neural Network with open-source neural-network library, Keras. For the third part we are building Convolutional Neural Network (CNN) with open-source neural-network library, Keras. At the end we are calculating Loss, Accuracy, Confusion Matrix for each classifier and observing the relative strengths and weaknesses.

1. Introduction:

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors. In this project in the first algorithm we will be using a single layer neural network to train our data and in the second project we will be using a multilayer neural network for the same. A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one from the other. In the third part of this project we will built a Convolution Neural Network to train our data and to get a better accuracy than previous two part.

2. Dataset:

The given Dataset for this Project is 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. and test data sets have 785 columns. The first column consists of the class labels and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image

3. Preprocessing:

Preprocessing of Data is generally necessary to remove redundant data, handling Null values and remove multicollinearity. In this Dataset I have performed three operations as a part of Preprocessing

For Single Layer Neural Network

- **A.** I have Normalized the Data by dividing the whole matrix by 255. Because a single pixel has always a value between 0 to 255 and the Maximum value of a pixel can be only 255.
- **B.** Then I implemented K encoding for the training label matrix. So if a matrix has a value of 9 it will be converted to (0,0,0,0,0,0,0,0,0,1).
- **C.** Then I have divided the training set into batches of 120.

For Multi-Layer Neural Network

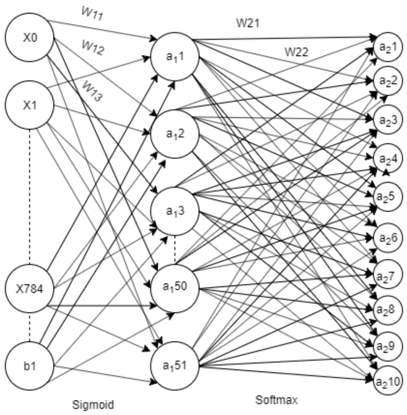
- **A.** I have divided the Training set into Training Data and Validation data. Validation data in 10% of the training data.
- **B.** Rest of the Preprocessing steps are same for Single Layer and Multi-Layer Neural Network.

For Convolution Neural Network

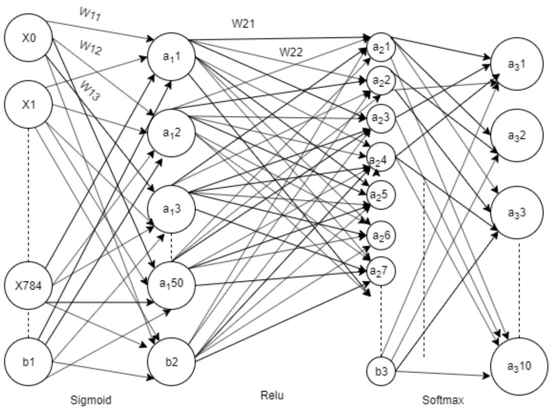
- **A.** Unflatten each row of the Training Data into 28*28 image matrix so we can use proper convolution to the images.
- **B.** Rest of the Preprocessing steps are same for Single Layer and Multi-Layer Neural Network.

4. Architecture:

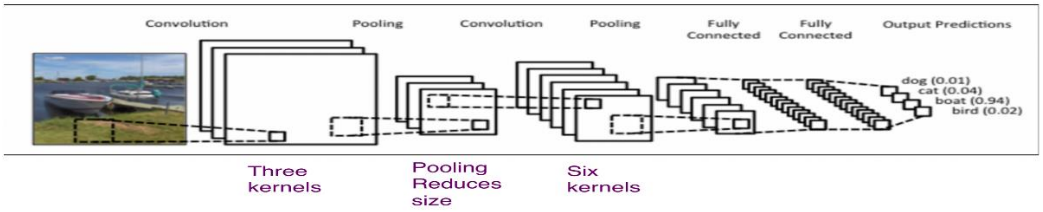
A. Computational Graph for Single Layer Neural Network (10 Class Classification)



B. Computational Graph for Multi-Layer Neural Network (10 Class Classification)



C. Convolution Network Architecture



Equations for Neural Network:

Activation Functions:

Sigmoid Function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Softmax Function:

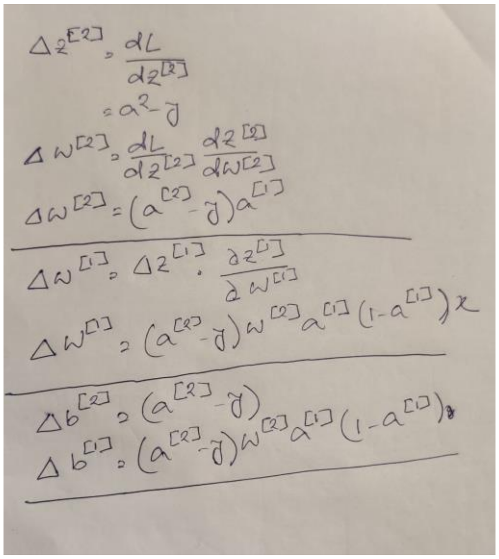
$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

- ***Loss Function: (Categorical_Crossentropy)***

$$L(a^{[2]}, y) = - \Sigma y \log a^{[2]}$$

Back Propagation:

In Back Propagation we are updating W2, W1, B2, B1



$$\Delta z^{[2]} = \frac{dL}{dz^{[2]}}$$

$$= a^2 - y$$

$$\Delta w^{[2]} = \frac{dL}{dz^{[2]}} \frac{dz^{[2]}}{dw^{[2]}}$$

$$\Delta w^{[2]} = (a^{[2]} - y) a^{[1]}$$

$$\Delta w^{[1]} = \Delta z^{[1]} \cdot \frac{\partial z^{[1]}}{\partial w^{[1]}}$$

$$\Delta w^{[1]} = (a^{[2]} - y) w^{[2]} a^{[1]} (1 - a^{[1]}) x$$

$$\Delta b^{[2]} = (a^{[2]} - y)$$

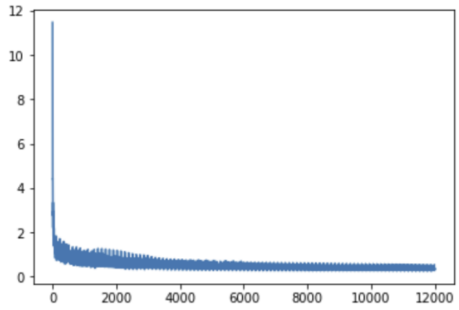
$$\Delta b^{[1]} = (a^{[2]} - y) w^{[2]} a^{[1]} (1 - a^{[1]})$$

5. Result:

- **Single Layer Neural Network:**

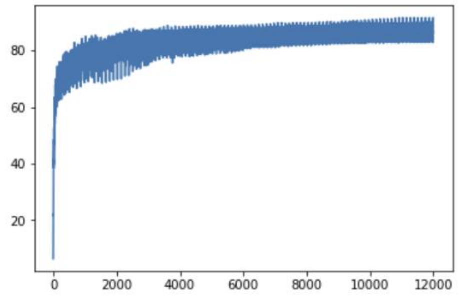
Learning Rate	Number of Hidden Layer	Accuracy	Batch Size
0.4	**85**	**84.5%**	**120**

Loss vs Epoch



X axis: Epcoh, Y axis: Loss

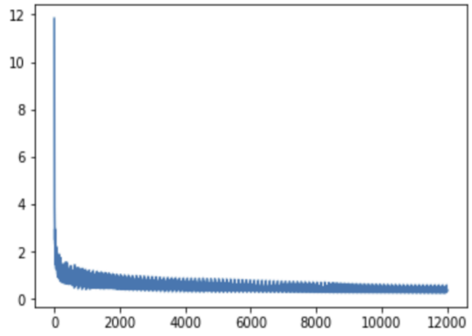
Accuracy vs Epoch



X axis: Epcoh**,** Y axis: Accuracy

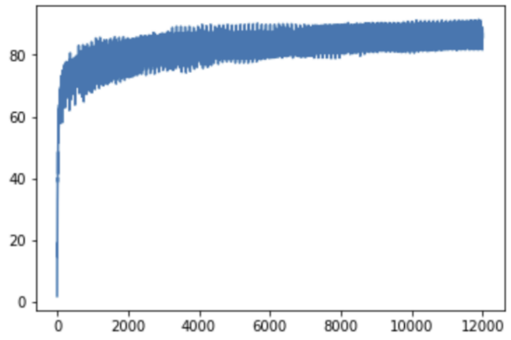
Learning Rate	Number of Hidden Layer	Accuracy	Batch Size
0.5	**90**	**84.7%**	**120**

Loss vs Epoch



X axis: Epcoh, Y axis: Loss

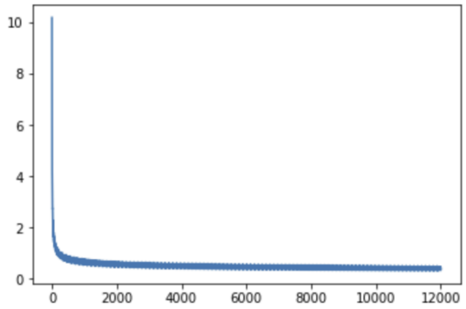
Accuracy vs Epoch



X axis: Epcoh**,** Y axis: Accuracy

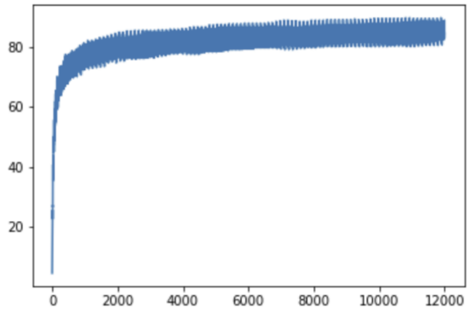
Learning Rate	Number of Hidden Layer	Accuracy	Batch Size
0.6	**100**	**84.9%**	**120**

Loss vs Epoch



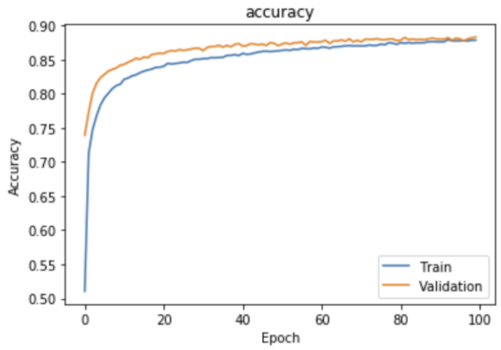
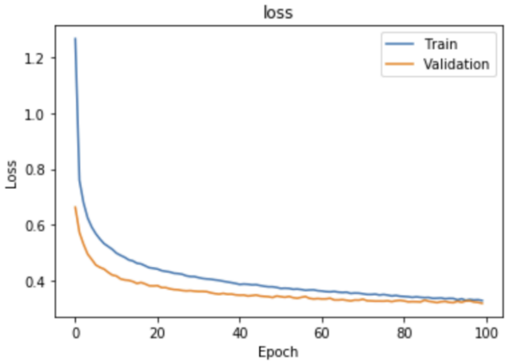
X axis: Epcoh, Y axis: Loss

Accuracy vs Epoch

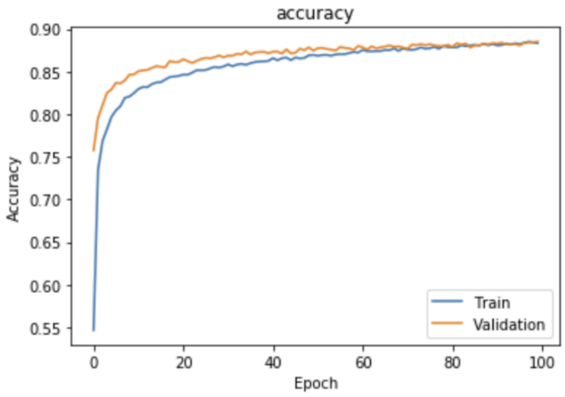
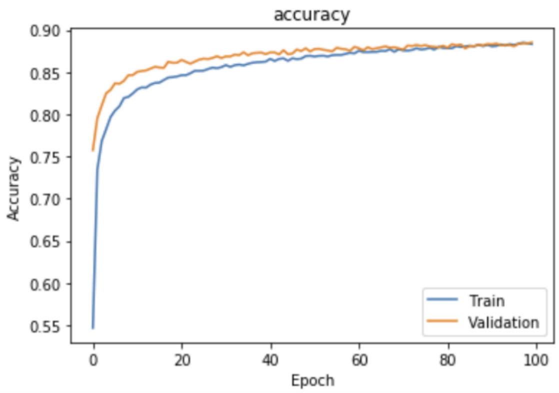


X axis: Epcoh**,** Y axis: Accuracy

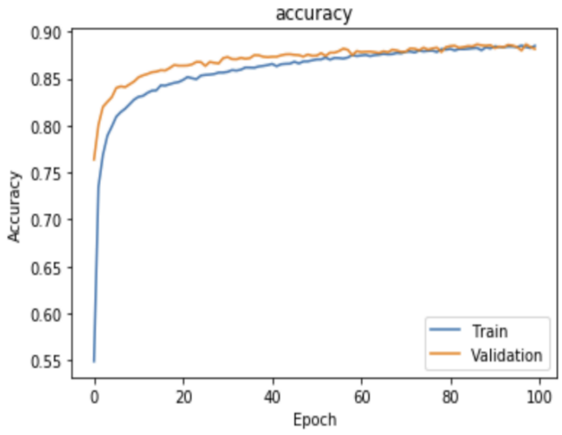
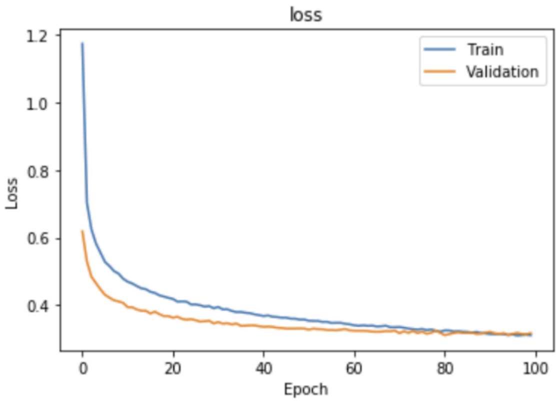
- **Multi-Layer Neural Network:**
  - **Achieved Accuracy=88%**
  - Layer 1=64, Layer 2=128, Optimizer: SGD Learning Rate=0.07,Batch Size=500
  - **Activation Functions:** Sigmoid, Relu, SoftMax

Epoch vs Accuracy	Epoch vs Accuracy
	

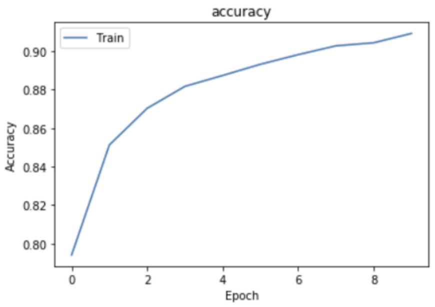
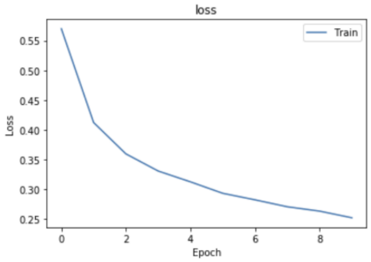
- **Achieved Accuracy=88.6%**
- Layer 1=128, Layer 2=256, Optimizer: SGD Learning Rate=0.09
- **Activation Functions:** Sigmoid, Relu, SoftMax
-

Epoch vs Accuracy	**Epoch vs Accuracy**
	

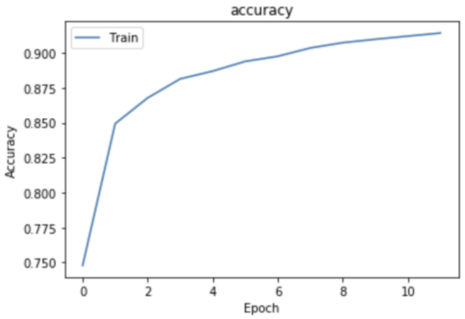
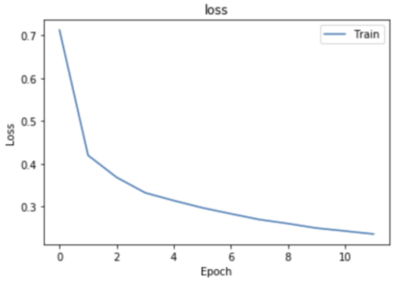
- **Achieved Accuracy=89%**
- Layer 1=128, Layer 2=256, Optimizer: SGD Learning Rate=0.1
- **Activation Functions:** Sigmoid, Relu, SoftMax

Epoch vs Accuracy	**Epoch vs Accuracy**
	

- **Convolution Neural Network:**
  - **Achieved Accuracy=89.8%**
  - Layer 1=64, Layer 2=64, Optimizer: Adam, Epoch=10
  - **Activation Functions:** Relu, Relu, SoftMax

Epoch vs Accuracy	**Epoch vs Accuracy**
	

  - **Achieved Accuracy=91%**
  - Layer 1=64, Layer 2=64, Optimizer: Adam, Epoch=12
  - **Activation Functions:** Relu, Relu, SoftMax

Epoch vs Accuracy	**Epoch vs Accuracy**
	

6. Conclusion:

After Observing the output of the three classifier we can see that we are getting maximum accuracy from Convolution Neural Network Model. Also, we can see that by modifying the hyper-parameters we can get better result from our neural networks model.

7. References

- Lecture Slides
- Bishop - Pattern Recognition And Machine Learning - Springer 2006
- Wikipedia