Introduction to Deep Learning Project 1 Report

### Muhammet Oğuz Özcan

### oguz.ozcan@ozu.edu.tr

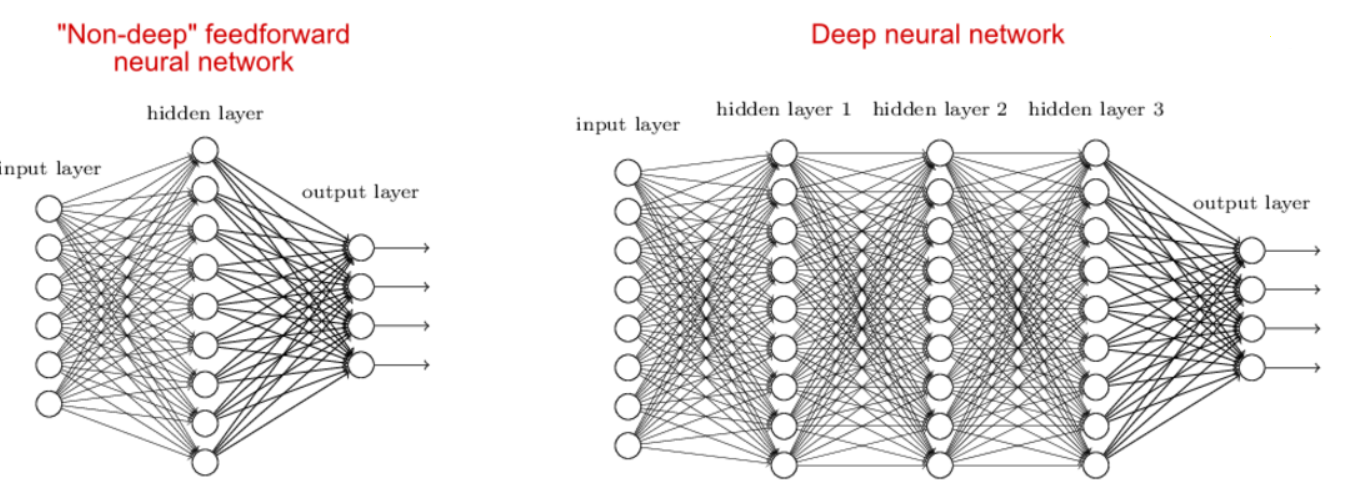
**Abstract**

In this project, Fashion-MNIST data set is used to train and test three different neural networks for a multiclass classification problem. The data consists of 10 different clothing types. Linear classification algorithm is used in this work, while changing the architecture types. Finally, we compare the performance of each of these algorithms.

# Introduction

The focus of this project is to gain experience by solving a classification problem with a new type of data set different that classical MNIST, which is Fashion MNIST. The classical MNIST data set has some problems. First it is too easy that achievement rates can go up to 99.7%. Second MNIST is overused that everyone is using them while start to work on machine learning. Finally, MNIST cannot represent modern computer vision task because it is so simple. For all these reasons, Fashion MNIST data set is used in this project.

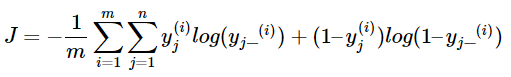
In this project, the shallow and deep neural network topologies will be analyzed. Shallow neural network usually consists of only one hidden layer and deep neural network generally has several hidden layers with various types.



The **activation function** of a node defines the output of that node given an input or set of inputs. Rectified Linear Unit is used as an activation function to calculate the output. ReLu works as the following:



Cost or loss function for the optimization / backpropagation to work on. Here we’ll use the cross entropy cost function, represented by the following formula:



# Problem and Dataset Information

The problem is to make a *classification* on the data set. The data set is available on *https://github.com/zalandoresearch/fashion-mnist* address. Data set includes total of 70,000 examples. Among them 60,000 is used for training and 10,000 is used for testing purposes. Each example is a 28\*28 grayscale image, associated with 10 labels belonging to 10 classes. The labels in the data are:

| **Label** | **Description** |
| --- | --- |
| 0 | T-shirt/top |
| 1 | Trouser |
| 2 | Pullover |
| 3 | Dress |
| 4 | Coat |
| 5 | Sandal |
| 6 | Shirt |
| 7 | Sneaker |
| 8 | Bag |
| 9 | Ankle boot |

To be able to use the data, firstly the data is manually downloaded from the <https://github.com/zalandoresearch/fashion-mnist#get-the-data> link. Then, it must be put in a directory under ‘data/fashion’ to work. Otherwise, Tensorflow will download and use the original MNIST. Since I am using TensorFlow version bigger than 1.5, I used the following command to download the data from Amazon directly.

fashion\_mnist = input\_data.read\_data\_sets('data/fashion', one\_hot=True, validation\_size=0, source\_url='http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/')

# Pre-processing

Since the TensorFlow is thought in this course and also suggested by the instructor, it is chosen for the implementation of the project. It is a widely used open source library for numerical computations. It supports many languages like Java, Python and C++ and Python API is used in this project. It has support for both CPU and GPU and since the computer used for this project does not hold a good GPU, the CPU supported version of the TensorFlow with version 1.5.1 is used.

# Learning Algorithms

A linear learning algorithm is used in this project. After the linear classification algorithm, the result is passed to the ReLu (Rectified Linear Unit) activation function. In the output layer, Softmax function is used to create 10 outputs that correspond to 10 labels.

ReLu is one of the most popular activation functions used in Deep Learning community. Activation functions are used in the neurons to decide if the computation made in one neuron should matter or affect the others. Why ReLu is chosen among sigmoid or tanh. Because in big neural networks, which we will have in this project, sigmoid or tanh will process all the inputs and they produce an output which is not zero, in other words, all the output of these activation functions will be processed in the following neurons. This increases the computation complexity and time. However, in ReLu almost half of the network will produce zero result since ReLu creates 0 for negative values. This decreases the computation complexity. Moreover, the gradient of the ReLu function is constant. The constant gradient of ReLUs results in faster learning.

Before the output layer, Softmax function is used. Since we are dealing with classification problem in this project we need to use softmax. What softmax does is, it squashes each unit to be between 0 and 1 and also the total sum of all units adds up to 1. The output of softmax creates a categorical probability distribution. It shows which of the classes that this input belongs to with the highest probability.

## Linear Classification

Linear classification is one of the methods used for classification problems. For a given function f, which is defined as . This is called a score function and it maps the input feature matrix or vector to output label classes.

In the linear classification the weight and bias, W and b respectively, are optimized by training to map input to correct label.

# Evaluation

We now present the results that we obtained on trying the aforementioned algorithms. In order to quantify the performance, we chose to plot Precision-Recall curves for each of the cases.

## One hidden layer with 10 neurons

I started to train my network with one hidden layer with 10 neurons. This is the initial and the basic network that I created. Since I am using CPU based TensorFlow, I cannot use the whole set of data in once, so I am sending the train data in batches of size 1000. The epoch in other words the whole data is used 100 times. The accuracy is 85.80%.

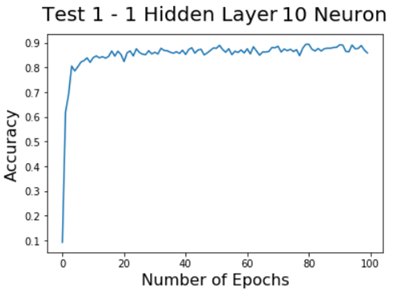


Figure 1: Accuracy of the Network vs. Number of Epochs

Then, I run the network over all the train data, the result of the accuracy was 87.86% and the accuracy of test data is 85.36%. So, this network is not a good one, because I am expecting accuracy around 90-95%. So, I increase the number of training epochs from 100 to 1000, 10 times more training.

After this training, the accuracy was 88.8%, which is also not quite well. Our size of input is 28\*28 and I’ve used only 10 neurons. No matter how much I trained the network, overall accuracy did not go more than 89%. This explains that, if the number of neurons is much smaller than the size of inputs, the network is not a good one.

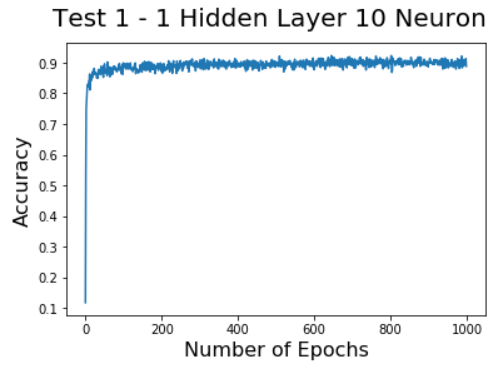


Figure 2: Accuracy of the Network vs. Number of Epochs

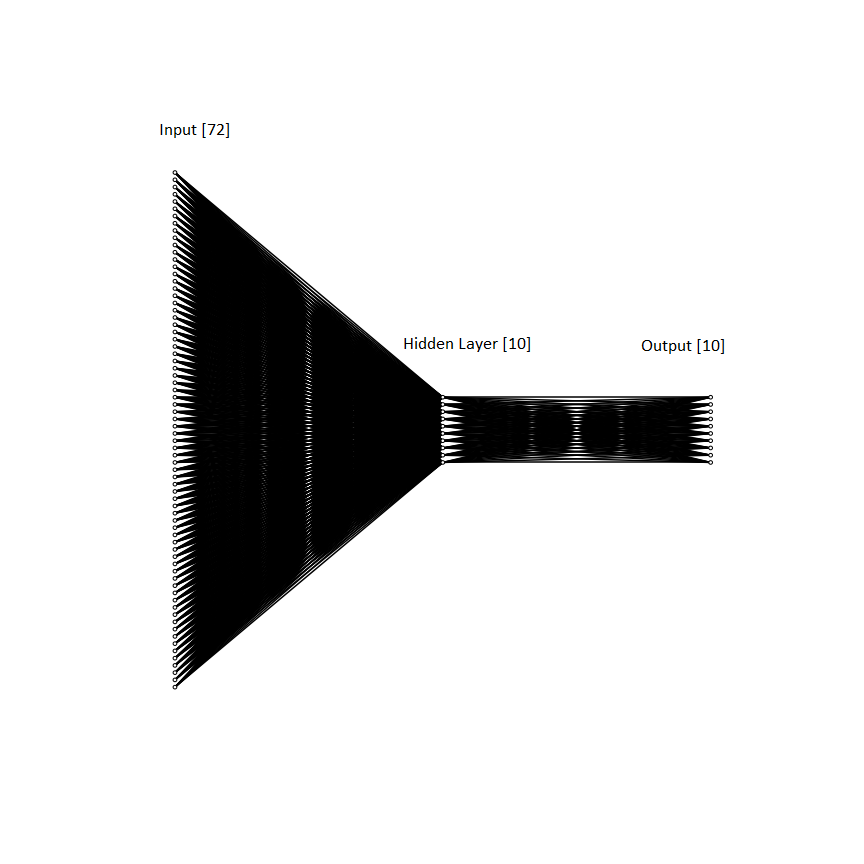


Figure 3: Architecture of the 1st Network

Then, I increase the number of neurons from 10 to 100, which is explained in section 5.2

## One hidden layer with 100 neurons, shallow network

In this architecture, one hidden layer with 100 neurons is created. Since the previous architecture was not very strong in terms of accuracy, this time I increase the number of neurons to 100 and I am creating a shallow network.

I run the whole data set 100 times in 1000 item sized batches. This training last 128.16 seconds and the resulting accuracy was 95.30%. This seems like a very good result. Then I run the network one last time for all data set and the result was 95.29%. Now, it’s time to test our network for test data. I run the network and the accuracy was 88.37%. I was expecting a better result this time, since the accuracy of training dataset is more than 95%. Now, let’s add more hidden layers to the architecture.

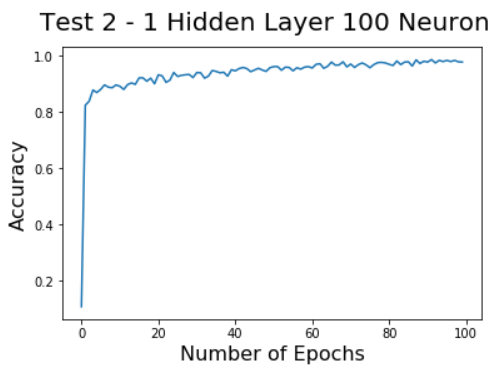


Figure 4: Accuracy of the Network vs. Number of epochs

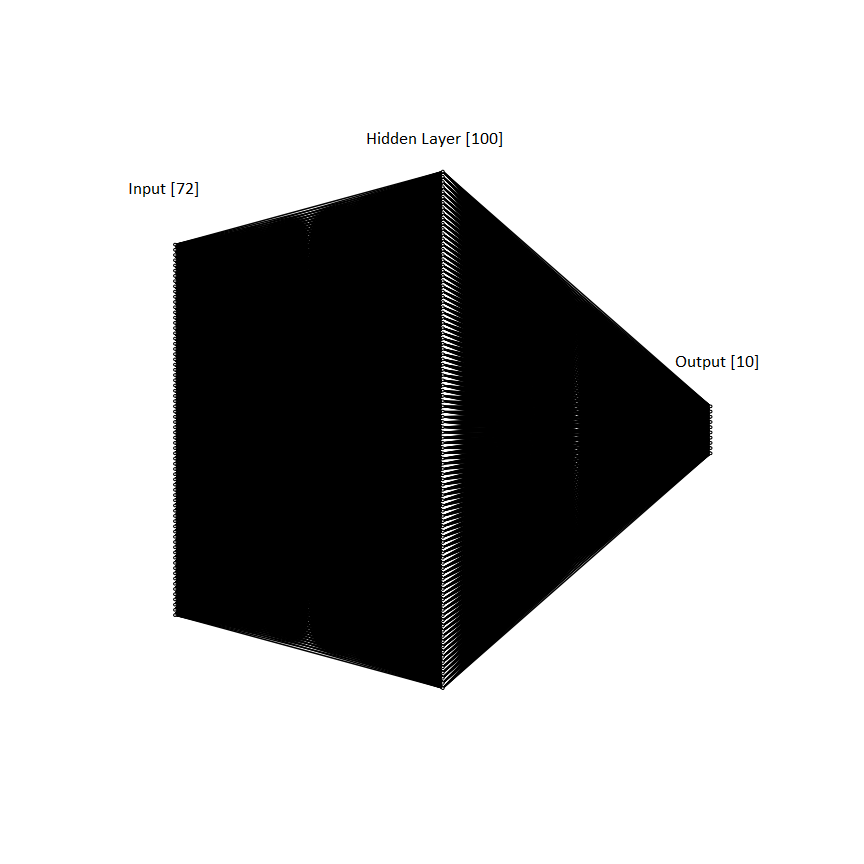


Figure 5: Architecture of the 2nd Network

## Three hidden layer with 100 neurons each, deep network

In this architecture, two more hidden layers added to the network with same number of neurons in each layer. This architecture is called as deep neural network architecture.

I trained the network with again 1000 batch item size and for 100 times. The time of the training was 164.49 seconds, which is more that the previous one as expected, since this network is much more complex than the previous one. The accuracy was 98.10%, which is very good and better than the shallow network. After that I run the test for train and test data set and the accuracy for train set is 97.85% and accuracy of test data set is 87.86%. The accuracy of the test data did not pass over the 90%.

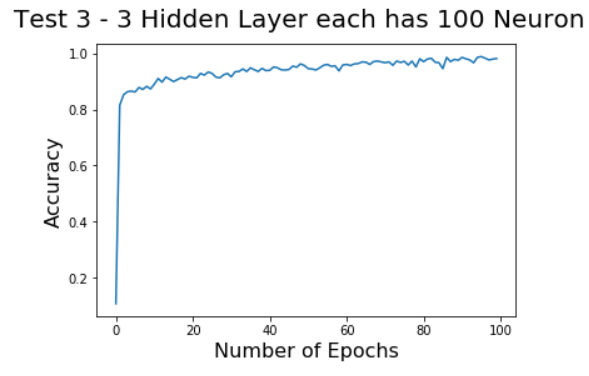


Figure 6: Accuracy of the Network 3 vs. Number of epochs

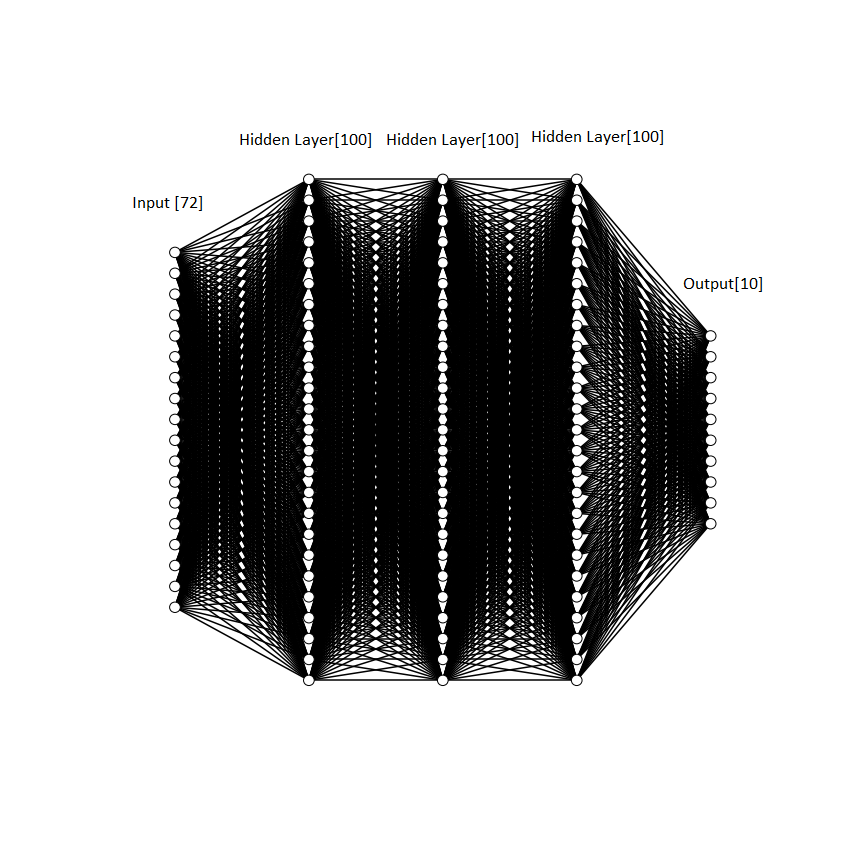


Figure 8: Architecture of the 3rd network

# Conclusion

As mentioned earlier, the aim in this project is to get hands-on experience working on a deep learning neural networks problem and to understand the effects of different architectures, specifically deep and shallow networks.

# References

* Fashion MNIST data repository - https://github.com/zalandoresearch/fashion-mnist