

---

# Implementing Neural Networks using Fashion MNIST Image Dataset

---

**Bhakti Jadhav**

Department of Computer Science

University at Buffalo

Buffalo NY, 14214

[bhaktiha@buffalo.edu](mailto:bhaktiha@buffalo.edu)

## Abstract

Fashion MNIST is an image dataset of Zalando's article which comprises of 70,000 images of fashion products from 10 categories so there are 7,000 images from each category. Each component is a  $28 \times 28$  grayscale images. The dataset is divided such that training set consists of 60,000 images and test set consists of 10,000 images. Original MNIST which is overused and can achieve high accuracy easily, using convolution network, accuracy of 99.7% have been achieved on MNIST. So Fashion MNIST is supposed to be a demanding dataset as a drop-in replacement for the original MNIST. Fashion MNIST is similar to original MNIST in terms of data format, image dimensions, training and testing set size and is used for exploring Machine Learning Algorithms. Outcome shows that Convolution performed on data provided, performed well on classification task with a test accuracy of  $\approx 91.5\%$ .

## 1 Introduction

Neural Network is one of the many approaches used for machine learning. It uses network of functions to understand an input data and translates it into another form. This project aims at implementing Neural Networks for the task of classification to recognize an image and identify it as one of the 10 classes.

The learning algorithm will learn by processing 60,000 labeled examples (i.e. data with "answers") during training and use these labels to learn how to construct correct output. After processing sufficient number of examples, the neural network can process unseen inputs and return accurate results. The program learns from experience so more variety of inputs, the more accurate results.

Convolution neural network is a subclass of neural networks. It has convolution layer which captures local information like neighboring pixels in an image and reduces complexity of model. It has advantages like faster training, fewer examples and reduces chances of overfitting.

Neural networks have broad range of applications and can estimate different types of input, including images, videos, files etc. It provides a generalized approach which can be exploited in variety of fields in machine learning. Some of the applications where neural network is used are pattern recognition, email spam filtering, facial recognition, medical diagnosis etc.

Fashion MNIST clothing images is used to train classifiers using different Neural Networks Algorithms. Following are the 3 tasks performed:

1. Built a Neural Network with one hidden layer to be trained and tested on Fashion-MNIST dataset.
2. Built multi-layer Neural Network with open-source neural-network library, Keras on dataset.

3. Built Convolutional Neural Network (CNN) with open-source neural-network library, Keras on dataset.

## 2 Dataset

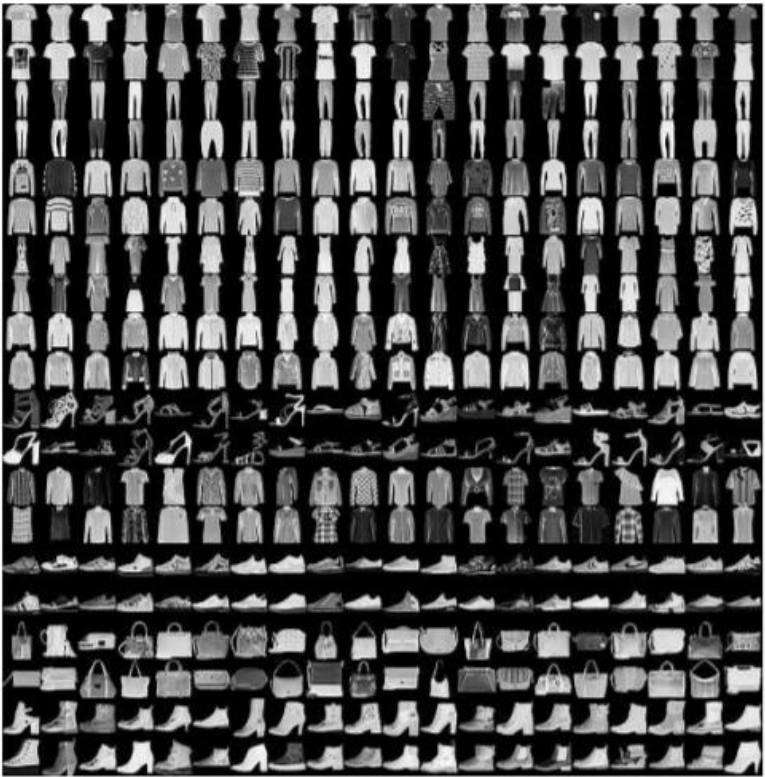
The dataset consists of training set(60,000 examples) and testing set(10,000 examples) along with labels. The images are  $28 \times 28$  grayscale images with pixels between 0 to 255.

Table 1: Files contained in the Fashion-MNIST dataset.

Name	Description	# Examples	Size
train-images-idx3-ubyte.gz	Training set images	60,000	25 MBytes
train-labels-idx1-ubyte.gz	Training set labels	60,000	140 Bytes
t10k-images-idx3-ubyte.gz	Test set images	10,000	4.2 MBytes
t10k-labels-idx1-ubyte.gz	Test set labels	10,000	92 Bytes

Labels are an array of integers 0 to 9 representing 10 different classes of clothing. The training set receives 6000 random images of each category

Table 2: Class names and example images in Fashion-MNIST dataset

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

## 3 Pre-Processing

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

### 3.1 One-hidden Layer Neural Network

1. Normalize the data to keep the gradient manageable in the range of [0,1].
2. One hot encoding the training and testing labels. The output will be a one-dimensional vector of size with each element representing “probability score”, where the score correlates to the class of input example.  
Example :  
“T-shirt/top”: [1,0,0,0,0,0,0,0,0,0]  
“Dress”: [0,0,0,1,0,0,0,0,0,0]
3. Re-shape and shuffle the data set because same category images are placed together and for effective training the data should be scattered and different.

### 3.2 Multi-layer Neural Network

1. Normalize the data to keep the gradient manageable in the range of [0,1].
2. Re-shaping the images since, the first layer of the model expects input to be of the shape (None, 28, 28, 1) whereas the training set has dimensions  $60000 \times 784$ . So, reshaping will convert 784 dimensional into images of size  $28 \times 28 \times 1$ . The “1” corresponds to number of channels.

### 3.3 Convolution Neural Network

1. Normalize the data to keep the gradient manageable in the range of [0,1].
2. Re-shaping the images since, In Convolution classifier the first layer expects input to be of the shape (None, 28, 28, 1) whereas the training set has dimensions  $60000 \times 784$ . So, reshaping will convert 784 dimensional into images of size  $28 \times 28 \times 1$ . The “1” corresponds to number of channels.
3. One hot encoding the training and testing labels. The input label(i.e. y\_train) consists of value corresponding to the image category. To convert it to “One Hot Representation” we convert it to 1-d vector.

## 4 Architecture

### 4.1 One-hidden Layer Neural Network

- One hidden layer neural network consists of 3 layers: input, hidden, output.
- The input layer has all the images. Most of the calculations happen in hidden layer where every perceptron unit takes input, multiplies and add to random values initially and then it is activated. This is a feedforward network in which the input moves from input layer, through hidden layer to output nodes.
- Initially we use random weights. For input in training set neural network is activated and it is observed how loss changes and the error is propagated back to previous layer and weights are adjusted accordingly. In this way the neural network learns from several labeled images.

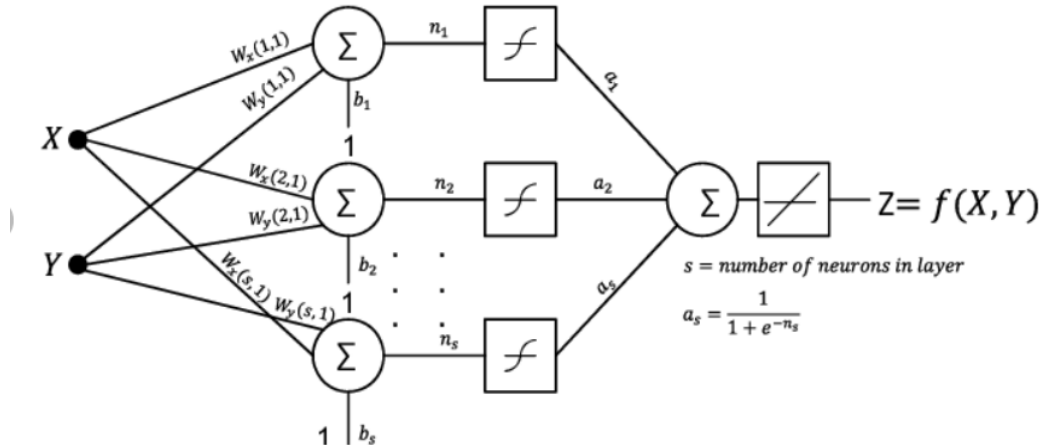


Figure1: One Hidden Layer Neural Network Model

## 4.2 Multi-layer Neural Network

For implementation of Multi-layer Neural Network we have used TensorFlow and Keras which a machine learning algorithm and high level application programming interface to build and train models.

Training the neural network consists of following step:

- Build the model : We have designed 3-layer, 6-layer and 12-layer neural network. In the first step we have flattened the data. The second layer is a dense layer which has ReLU activation function with 128 neurons. The last layer is a dense layer with softmax function which classifies 10 categories of data and has 10 neurons. In the 6- and 12-layer network we have increased the depth of the network by adding more hidden layers with same activation functions.
- Compile the model : We have used sparse\_categorical\_crossentropy loss function. We have used sparse because our labels are not one-hot encodings. We have used Adam optimizer which is an extension to the classic gradient descent. Accuracy metrics is used which measures the fraction of images which are correctly classified.
- Train the model : We train the data be with input images and labels and also validate 10% of the training data.
- Evaluating the model : After the model is trained we can evaluate the performance on testing dataset. Evaluation metrics of the model gives test loss and accuracy.

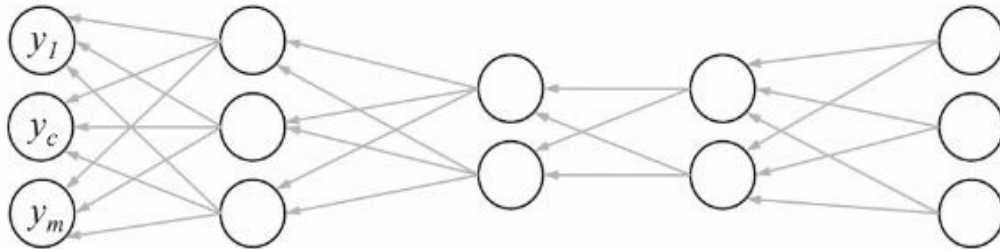


Figure2: Multi-Layer Neural Network Model

### 4.3 Convolution Neural Network

Convolution Neural Network consists of 2 convolution layers, each followed by max pooling layer. The second max-pooling layer is flattened and fed into a fully connected hidden layer which uses ReLU activation function with 1024 neurons. Final output layer uses softmax function to classify 10 categories of data.

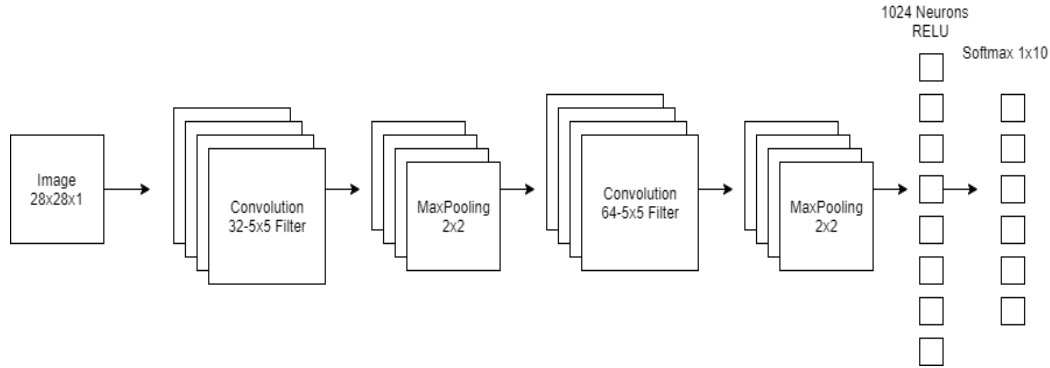


Figure3: Multi-Layer Neural Network Model

Table 3: Convolution Neural Network Model Summary

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 28, 28, 32)	832
max_pooling2d_3 (MaxPooling2)	(None, 14, 14, 32)	0
conv2d_4 (Conv2D)	(None, 14, 14, 64)	51264
max_pooling2d_4 (MaxPooling2)	(None, 7, 7, 64)	0
flatten_2 (Flatten)	(None, 3136)	0
dense_3 (Dense)	(None, 1024)	3212288
dropout_2 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 10)	10250
Total params: 3,274,634		
Trainable params: 3,274,634		
Non-trainable params: 0		

## 5 Result

By tuning hyperparameters, it is observed that the cost function gradually approaches minimum value.

### One Hidden Layer Neural Network:

The best case is with Number of Epochs = 2000 and learning rate = 0.5

Test loss: 49.051

Test accuracy: 81.0

										precision	recall	f1-score	support	
[	789	8	17	33	1	0	174	0	5	0]				
[	10	936	0	19	4	0	2	0	1	0]	0	0.79	0.77	1027
[	15	9	687	23	133	0	121	0	12	1]	1	0.94	0.96	972
[	46	34	16	835	36	1	36	0	9	1]	2	0.69	0.69	1001
[	8	7	150	35	699	1	140	0	5	0]	3	0.83	0.82	1014
[	2	0	0	1	3	880	0	40	12	18]	4	0.70	0.67	1045
[	113	5	112	46	110	0	497	0	28	0]	5	0.88	0.92	956
[	0	0	0	1	0	69	1	885	8	48]	6	0.50	0.55	911
[	0	0	0	1	0	69	1	885	8	48]	7	0.89	0.87	1012
[	17	1	18	7	14	12	29	5	918	1]	8	0.92	0.90	1022
[	0	0	0	0	0	37	0	70	2	931]]	9	0.93	0.90	1040
avg / total										0.81	0.81	0.81	10000	

Figure5: Confusion Matrix and Classification Report for Simple NN

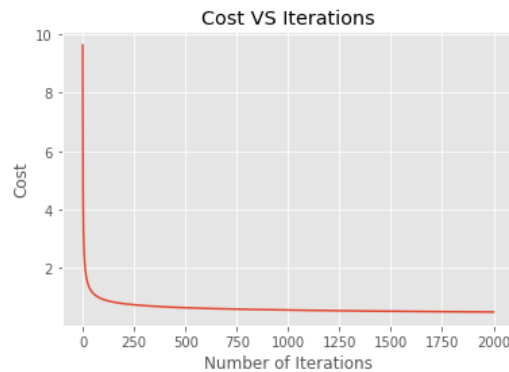


Figure6: Training Loss Plot for 2000 Epochs

### Multi-layer Neural Network:

The best case is with Number of epochs = 10

Test loss: 39.550

Test loss: 35.660

Test loss: 34.991

Test accuracy: 86.119

Test accuracy: 87.1

Test accuracy: 87.56

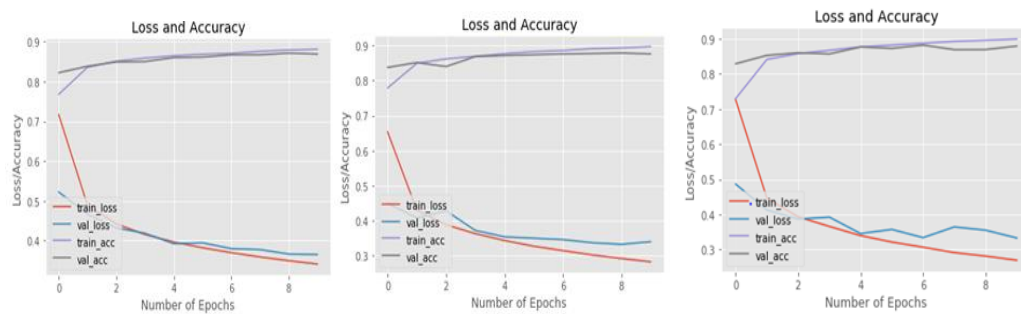


Figure7: NN-3, NN-6 and NN-12 Graph for 10 Epochs

### Convolution Neural Network:

The best case is with Number of epochs = 20

Test loss: 23.886

Test accuracy: 91.51

[[883 0 19 12 2 0 110 0 7 0]											precision	recall	f1-score	support
[ 0 981 1 4 1 0 1 0 1 0]										0	0.88	0.85	0.87	1033
[ 16 0 882 10 55 0 71 0 2 0]										1	0.98	0.99	0.99	989
[ 12 13 9 931 32 0 27 0 4 0]										2	0.88	0.85	0.87	1036
[ 6 2 50 18 868 0 66 0 4 0]										3	0.93	0.91	0.92	1028
[ 1 0 0 0 0 977 0 6 2 5]										4	0.87	0.86	0.86	1014
[ 79 2 39 21 41 0 719 0 1 1]										5	0.98	0.99	0.98	991
[ 0 0 0 0 0 15 0 973 5 31]										6	0.72	0.80	0.76	903
[ 3 2 0 3 1 1 6 0 974 0]										7	0.97	0.95	0.96	1024
[ 0 0 0 1 0 7 0 21 0 963]]										8	0.97	0.98	0.98	990
										9	0.96	0.97	0.97	992
										avg / total	0.92	0.92	0.92	10000

Figure8: Confusion Matrix and Classification Report for CNN

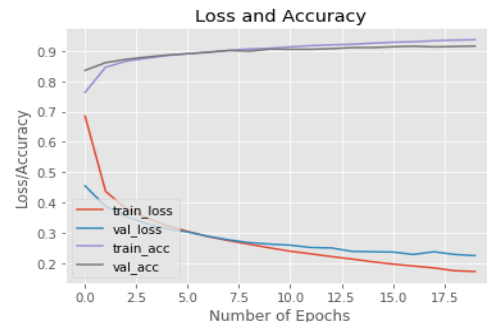


Figure9: CNN Graph for 20 Epochs

## 6 Conclusion

Simple Neural Network with one hidden layer gives the least test accuracy among the 3 implemented neural networks. In Multi-layer Neural Network it is observed that deeper the model, higher is the accuracy. Also, the loss is decreasing with increase in hidden layers. The Convolution-layer Neural Network gives the best model with highest accuracy and least loss as expected.

Performance : SNN < MNN < CNN

## References

- [1] <https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/>
- [2] <https://arxiv.org/pdf/1708.07747.pdf>
- [3] <https://medium.com/@ipylpenko/exploring-neural-networks-with-fashion-mnist-b0a8214b7b7b>
- [4] <https://deeptai.org/machine-learning-glossary-and-terms/neural-network>
- [5] <https://jonathanweisberg.org/post/A%20Neural%20Network%20from%20Scratch%20-%20Part%201/>
- [6] [https://github.com/justanothergirlwhocodes/TensorFlow-NNs/blob/master/NN\\_MNIST\\_depth.ipynb](https://github.com/justanothergirlwhocodes/TensorFlow-NNs/blob/master/NN_MNIST_depth.ipynb)
- [7] <https://pravarmahajan.github.io/fashion/>
- [8] <https://www.pyimagesearch.com/2019/02/11/fashion-mnist-with-keras-and-deep-learning/>