

---

# Fashion MNIST Dataset Classification using Artificial Neural Network and Convolutional Neural Network

---

**Sunnam Ravikiran**

Department of Computer Science

University at buffalo

buffalo, NY 14221

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

## Abstract

ANNs/CCs began as an attempt to exploit the architecture of the human brain to perform tasks that conventional algorithms had had little success. The purpose of the project is to build Artificial Neural network and convolutional neural network the dataset used is the fashion MNIST data set. The data set is split into two portions on which model is trained, validated and tested. It was observed that the model predicted the classes with good accuracy.

## 1 Introduction

An **Artificial Neural Network(ANN)** is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it.

A **Convolutional Neural Network (CNN)** 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 ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

## 2 Dataset

For training and testing of our classifiers, we will use the 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. 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.

1	T-shirt/top
2	Trouser
3	Pullover
4	Dress
5	Coat
6	Sandal
7	Shirt
8	Sneaker
9	Bag
10	Ankle Boot

Table 1: Labels for Fashion-MNIST dataset

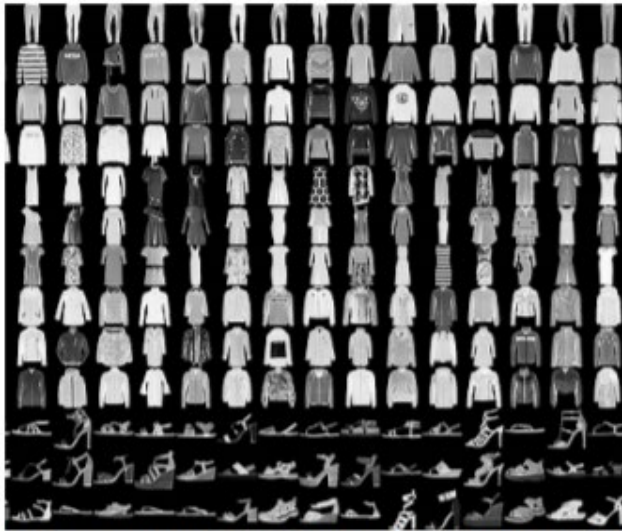


Figure 1: Example of how the data looks like.

The training dataset contains [60000] instances.

The validation dataset contains [10000] instances.

The testing dataset contains [10000] instances.

### 3 Preprocessing the dataset

The given Fashion MNIST dataset has been preprocessed before using logistic regression model as follows.

- *Processing the data*
  1. The dependent variable is identified by reshaped accordingly, matching the size of the out network.
  2. The independent variables which don't contribute the prediction of the dependent variable are dropped from the dataset.
- *Data Partitioning*

In machine learning we usually split our data into three subsets: train, validate and

test dataset, and fit our model on the train data, in order to make predictions on the test data. This is done to avoid the overfitting of the model.

The Given dataset is partitioned into training, validation and testing data. Randomly data is split where training dataset has 60000 instances and test dataset has 10000 instances.

1. The training dataset contains [60000] instances.
2. The validation dataset contains [10000] instances.
3. The testing dataset contains [10000] instances.
- 4.

- **Normalization**

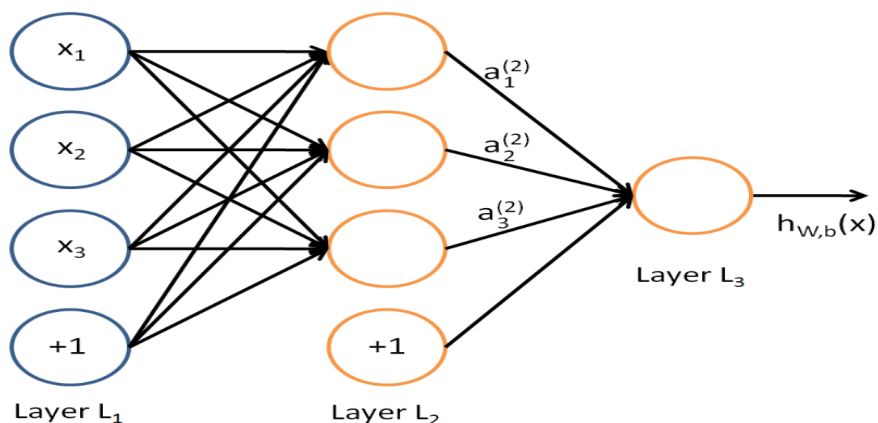
Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. All the datasets (train, validation and test) have been mean normalized.

## 4 Architecture

### ANN

ANNs consist of artificial neurons. Each artificial neuron has a processing node ('body') represented by circles in the figure as well as connections from ('dendrites') and connections to other neurons which are represented as arrows in the figure. In a commonly used ANN architecture, the multilayer perceptron, the neurons are arranged in layers. An ordered set (a vector) of predictor variables is presented to the input layer. Each neuron of the input layer distributes its value to all of the neurons in the middle layer.

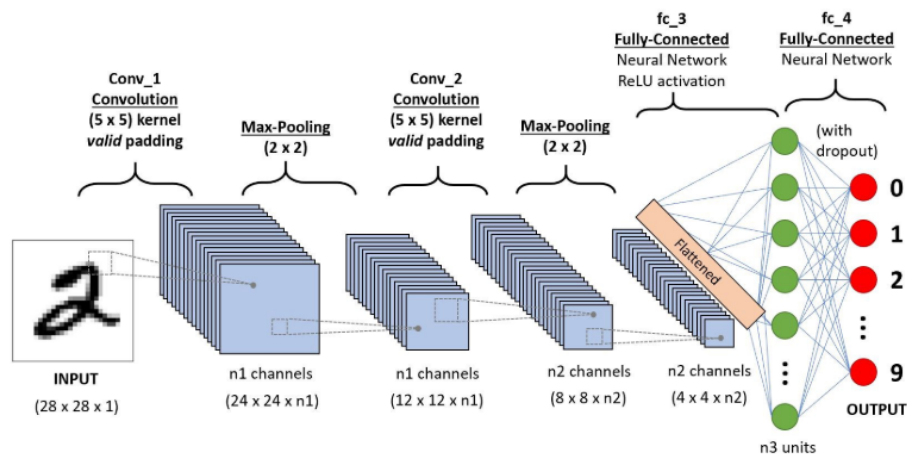
Along each connection between input and middle neurons there is a connection weight so that the middle neuron receives the product of the value from the input neuron and the connection weight. Each neuron in the middle layer takes the sum of its weighted inputs and then applies a non-linear (usually logistic) function to the sum. The result of the function then becomes the output from that particular middle neuron. Each middle neuron is connected to the output neuron. Along each connection between a middle neuron and the output neuron there is a connection weight. In the final step, the output neuron takes the weighted sum of its inputs and applies the non-linear function to the weighted sum.



## CNN

A Convolutional Neural Network (ConvNet/CNN) 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 ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.



A CNN sequence to classify handwritten digits

### General Equations

$$z = xw + b \quad \rightarrow \quad z = \text{function}(w)$$

$$a = \text{activation\_function}(z) \quad \rightarrow \quad a = \text{function}(z)$$

$$c = -(y * \log(a3)) \quad \rightarrow \quad c = \text{function}(a)$$

### Forward Feed Equations:

$$z1 = x.w1 + b1$$

$$a1 = \text{sigmoid}(z1)$$

$$z2 = a1.w2 + b2$$

$$a3 = \text{softmax}(z2)$$

### Back propagation equations:

$$a3\_delta = a3 - y$$

$$z1\_delta = a3\_delta.w2.T$$

$$a2\_delta = z2\_delta.\text{sigmoid\_derv}(a2)$$

Since there is one hidden layer the equations are as follows

```

z1 = x.w1 + b1
a2 = sigmoid(z1)
c = a1.w2 + b2

dc    dc    da2    dz1
--- = --- . --- . ---
dw1   da2   dz1   dw1

dz1/dw1 = x
da2/dz1 = sigmoid_derv(z1)

dc    dc    da3    dz2
--- = --- . --- . --- => dc/da2 = a3_delta.w2
da2   da3   dz2   da2

w1 = w1 - dc/dw1
and set a1_delta = dc/da1 . da1/dz1
dc    dc    da1    dz1
--- = --- . --- . ---
db1   da1   dz1   db1
dz1/db1 = 1
b1 = b1 - dc/db1 => b1 = b1 - a1_delta

z2 = a1w2 + b2
a2 = sigmoid(z2)
dc    dc    da2    dz2
--- = --- . --- . ---
dw2   da2   dz2   dw2
dz2/dw2 = a1
da2/dz2 = sigmoid_derv(z2)
dc    dc    da3    dz3
--- = --- . --- . --- => dc/da2 = a3_delta.w3
da2   da3   dz3   da2
w2 = w2 - dc/dw2
and set a2_delta = dc/da2 . da2/dz2
dc    dc    da2    dz2
--- = --- . --- . ---
db2   da2   dz2   db2
dz2/db2 = 1
b2 = b2 - dc/db2 => b2 = b2 - a2_delta

```

## 4.1 Sigmoid activation


In order to map predicted values to probabilities, we use the sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

$$P(y = 1|x) = h_{\theta}(x) = \frac{1}{1 + \exp(-\theta^{\top} x)} \equiv \sigma(\theta^{\top} x),$$

$$P(y = 0|x) = 1 - P(y = 1|x) = 1 - h_{\theta}(x).$$

## 4.2 SoftMax activation

softmax function, also known as softargmax[1] or normalized exponential function, function that takes as input a vector of  $K$  real numbers, and normalizes it into a probability distribution consisting of  $K$  probabilities proportional to the exponentials of the input numbers. That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the interval  $(0,1)$ , and the components will add up to 1, so that they can be interpreted as probabilities.

$$P(y=j \mid \theta^{(i)}) = \frac{e^{\theta_j^{(i)}}}{\sum_{k=0}^K e^{\theta_k^{(i)}}}$$


## 4.2 Cost Function [Cross-entropy]

Cross-entropy is commonly used to quantify the difference between two probability distributions. Usually the "true" distribution (the one that your machine learning algorithm is trying to match) is expressed in terms of a one-hot distribution.

The cross entropy between two probability distributions  $p$  and  $q$  over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set if a coding scheme used for the set is optimized for an estimated probability distribution  $q$ , rather than the true distribution  $p$ .

$$H_{y'}(y) := - \sum_i y'_i \log(y_i)$$

## 4.3 Gradient Descent

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, we use gradient descent to update the parameters of our model. Parameters refer to coefficients in Linear Regression and weights in neural networks.

### 4.3.1 Learning rate

The size of these steps is called the *learning rate*. With a high learning rate we can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. With a very low learning rate, we can confidently move in the direction of the negative gradient since we are recalculating it so frequently. A low learning rate is more precise, but calculating the gradient is time-consuming, so it will take us a very long time to get to the bottom.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l),$$

$$w_{t+1} = w_t - \eta_t \nabla \mathcal{L}$$

$$w_{j,t+1} = w_{j,t} - \eta_t \frac{\partial \mathcal{L}}{\partial w_j}$$

## 4.4 Results

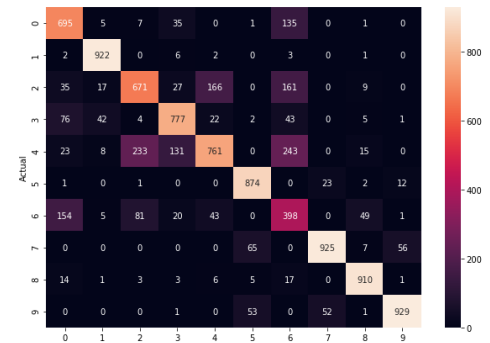
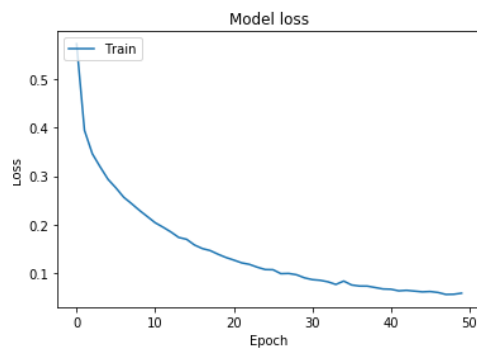
The preprocessing of data along with logistic regression implementation is done in python. the dataset is split into train, validation and test. ANN was build from scratch using numpy lib and CNN/ANN was again built using keras library.

$$Accuracy = \frac{N_{correct}}{N}$$

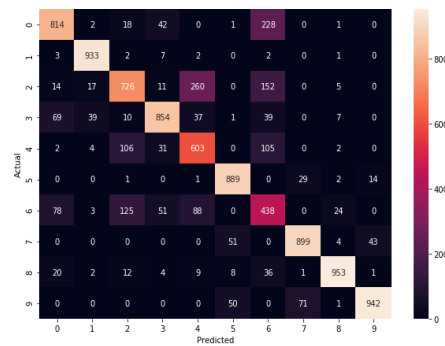
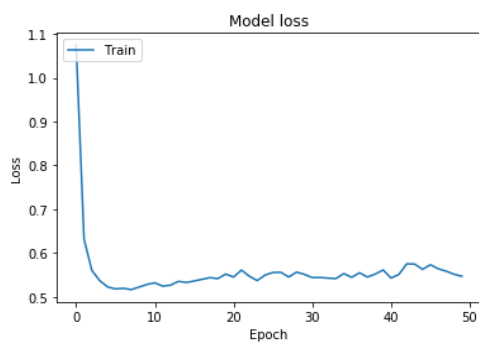
The initial [W1, W2 have been initialized to randomly using normal distribution and the values have been calculated iteratively using gradient decent approach. For each epoch the training loss and validation is calculated and plotted.

### Training Loss Vs Epoch graph

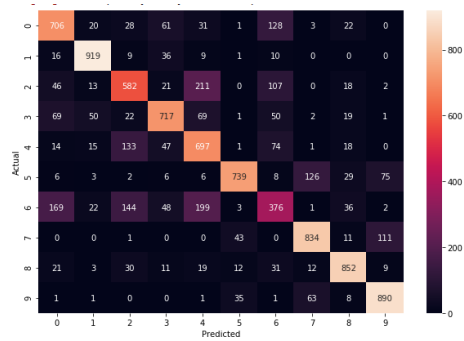
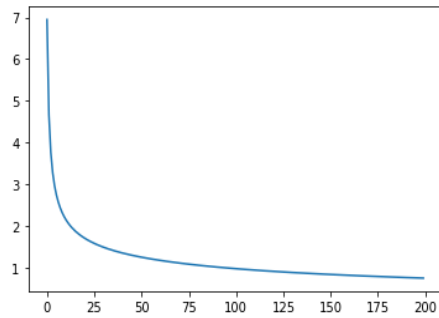
#### CNN Using keras



#### ANN Using Keras



## ANN Implmented from scratch



### Accuracy Results

- ANN (Built from scratch) - 70.8%
- ANN (Using Keras lib) - 78.2%
- CNN (Using Keras lib) - 84.4 %

## 5 Conclusion

From the result above, CNN/ANN have proven so effective that they are the go-to method for any type of prediction problem involving image data as an input. The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image. This allows the model to learn position and scale in variant structures in the data, which is important when working with images. It is also observed that the accuracy increased when the hidden layer were increased.