Multi-Class classification using Artificial Neural Network

Fashion MNIST Data Set

Snigdha Gupta
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
snigdhag@buffalo.edu

Abstract

A Neural Network algorithm which emulates the human brain can classify an image and segregate it into one of the given classes. This is a key algorithm in solving machine learning problems. We use the concept of a connected graph with input and output neurons and weighted edges to classify Fashion MNIST data. Furthermore, we use the concept of Multi-layer and Convolution neural network to solve the same problem using open-source neural network library Keras.

1 Introduction

1.1 Supervised Learning – Neural Network

A neural network is a supervised learning algorithm which means that we provide it the input data containing the independent variables and the output data that contains the dependent variable.[1] In the given data set we have 10 classes and hence this means that the input can belong to any one of the classes. So, given the input – an image of a clothing item, the model should be able to predict which item is present in the image – is it an ankle boot (class 10) or trouser (class 2) or top (class 1) etc.

A multi-layer neural network has multiple layers (Figure 1). Multi-layer neural networks can be set up in numerous ways. MNNs, have at least 1 input layer (ideally more than 2 - 3 layers), that sends weighted inputs to a series of hidden layers and then to an output layer at the end. These more sophisticated setups are also associated with nonlinear builds using sigmoids and other functions to direct the firing or activation of artificial neurons.

1.2 Supervised Learning – Convolutional Neural Network

Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used. The image classification takes an input, processes it and classifies it into a category. An image is a matrix where each pixel value defines the intensity of the pixel that plays the role to create that image. A CNN model ideally undergoes convolution layers with kernels (filters), pooling, fully connected layers and softmax function to classify an image with probabilistic values that range between 0 and 1 (Figure 2).

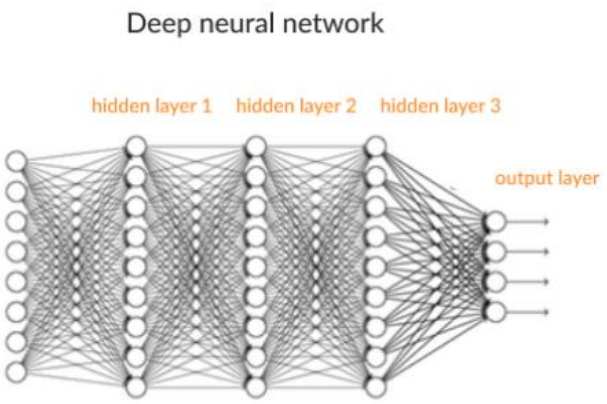

Figure 1 Artificial Neural Network with multiple hidden layers

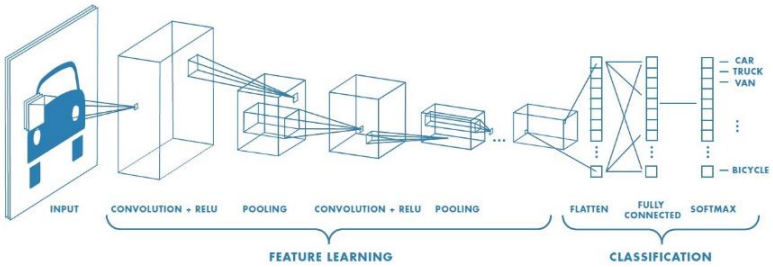

Figure 2 Neural Network with many convolutional layers

Task 1 – Neural Network from scratch:

We first implement neural network from scratch and perform forward pass with one hidden layer. The weights(w_1) and bias(b_1) is used to calculate z_1 and then passed through a sigmoid function to get the activation (a_1). The output of the hidden layer(a_1) then becomes the input for the output layer. We again perform forward pass with new weights (w_2) and bias (b_2). Both weights and biases are randomly initialized at the beginning.

Once forward pass is completed, we calculate the loss using the categorical cost function. Then we perform back propagation between output layer and hidden layer. We update the weights(w_2) and bias(b_2) by calculating the derivative of w_2 w.r.t to the cost function. A similar operation is performed over the weight w_1 biases b_2 and b_1. The entire process narrated is repeated for defined epochs and the training of the model is carried out.

The process mentioned above can defined using mathematical equations as explained below:

The Genisis Equation (Eq 1):

$$z_1 = W_1^T * x + b_1$$

Equation 1 is then passed through a sigmoid function and hence activation a_1 is generated.

Activation for input layer and hidden layer (Eq2):

$$a_1 = \sigma(z_1)$$

The sigmoid function is (Eq 3):

$$\sigma = \frac{1}{1 + e^{-z}}$$

The Genisis Equation (Eq 4):

$$z_2 = W_2^T * a_1 + b_2$$

Equation 3 is then passed through a **softmax function** and hence activation a_2 is generated.

Softmax Function: The softmax function is a form of logistic regression that normalizes the input value into a vector of values that follows a probability distribution whose sum is equal to 1. It calculates the probability of each target class over all possible target classes. Since the output is ranged between 0 and 1, we can use softmax to solve multi-class classification problems.

The softmax function is (Eq 5):

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

Where $\theta = w_0x_0 + w_1x_1 + w_2x_2 + ... + w_kx_k$

Activation for hidden layer and output layer (Eq 6):

$$a_2 = \text{softmax}(z_2)$$

The categorical cost function is defined by (Eq 7):

$$Cost = -\text{y} \log(a_2)$$

The derivative of cost w.r.t weights and biases is calculated for w_2, w_1 and b_2 and b_1:

$$\frac{\partial C}{\partial w_2} = \frac{\partial C}{\partial a_2} . \frac{\partial a_2}{\partial z_2} . \frac{\partial z_2}{\partial w_2} \; using \; chain \; rule$$

$$let, \qquad a_2' = \frac{\partial C}{\partial a_2} . \frac{\partial a_2}{\partial z_2}$$

We know value of a_2' is nothing but predicted – actual.

Hence, $a_2' = a_2 - y$, where a_2 is our prediction and y is our actual data.

$\frac{\partial C}{\partial w_2} = (a_2 - y) . \frac{\partial z_2}{\partial w_2}$ and $\frac{\partial z_2}{\partial w_2} = a_1$, therefore,

$$\boldsymbol{\frac{\partial C}{\partial w_2}} = (a_2 - y).a_1$$

Similarly,

$$\boldsymbol{\frac{\partial C}{\partial b_2}} = (a_2 - y), \qquad \boldsymbol{\frac{\partial C}{\partial w_1}} = (a_2 - y).w_2 . sigmoid\_derivative(z_1) . X \; and$$

$$\boldsymbol{\frac{\partial C}{\partial b_1}} = (a_2 - y).w_2 . sigmoid\_derivative(z_1)$$

The weights and biases are then updated and backpropagation is completed..

$$w_2 = w_2 - \alpha \frac{\partial C}{\partial w_2}, \; b_2 = b_2 - \alpha \frac{\partial C}{\partial b_2}, \; w_1 = w_1 - \alpha \frac{\partial C}{\partial w_1}, \; b_1 = b_1 - \alpha \frac{\partial C}{\partial b_1}$$

Task 2 – Multilayer Neural Network and Task 3 – Convolutional Neural Network:

Task 2 and 3 are implemented using Keras – a rich machine learning python library. In Task 2 we include multiple layers by introducing Dense functions. A dense layer represents a matrix vector multiplication. (assuming your batch size is 1) The values in the matrix are the trainable parameters which get updated during backpropagation.[14]. In Task 3 we perform

convolution. It is a mathematical operation "summarizes" a tensor or a matrix or a vector into a smaller one. We summarize a tensor along n dimensions where n is the number of dimensions tensor has. We are using Conv2D to summarize (convolve) since our matrix is 2 dimensional.

2 Data Set

2.1 Zalando's Fashion MNIST

Fashion-MNIST dataset by e-commerce Zalando—consists 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.

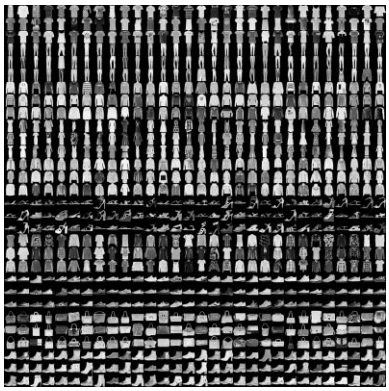

Figure 3 Fashion MNIST data set by Zalando

The Fashion MNIST dataset is identical to the MNIST dataset in terms of training set size, testing set size, number of class labels, and image dimensions:

- 60,000 training examples
- 10,000 testing examples
- 10 classes
- 28×28 grayscale images

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.

Each training and test example is assigned to one of the labels as shown below:
1 T-shirt/top
2 Trouser
3 Pullover
4 Dress
5 Coat
6 Sandal
7 Shirt
8 Sneaker
9 Bag
10 Ankle Boot

3 Preprocessing

The transformation applied to the data before feeding it to the algorithm is called pre-processing. This is required to convert the raw data into a clean data set, since the raw data is not feasible for the analysis.

The data set comprises of 784 columns. We scale these values to a range of 0 to 1 before feeding them to the neural network model. In order to do so, we divide the values by 255. It's important that the *training set* and the *testing set* be preprocessed in the same way.

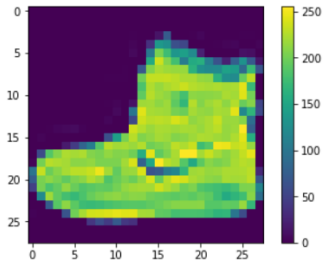

Figure 4 Non-normalized input data

Each image is normalized between a range of 0 to 1 by dividing each pixel value by 255.0 (RGB) to convert the image into gray scale. We do this so that the classes, when assigned weights can be compared.

It is important to verify that the data has been normalized correctly.

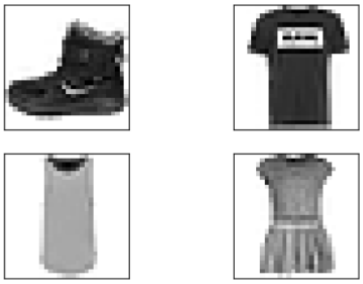

Figure 5 Normalized input data

Furthermore, we use to_catagorical() function present in the machine learning library Keras. This function is used to covert the Y_train/valid/test value to one-hot-vector encoding. This means that every value of Y will be replaced by a vector which will contain 1 at the index equal to the class. Let's say our image belongs to class 7, then all values in the vector will be 0 except the value at the 7^{th} index.

4 Architecture

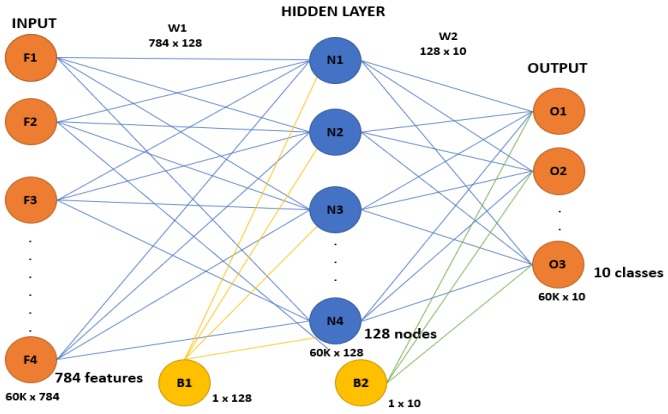

Figure 6 Task1: Architecture of Single Layer Neural Network

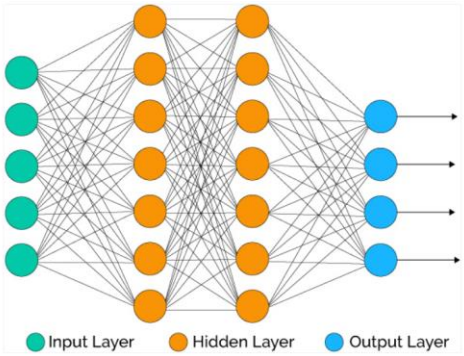

Figure 7 Task 2: Multi-layer neural network architecture

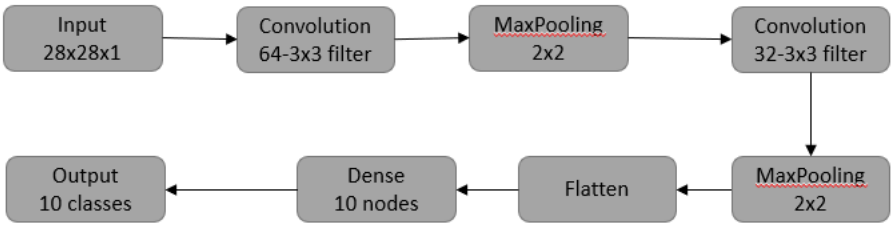

Figure 8 Task 3 Convolutional Neural Network architecture

5 Results

The following results were obtained:

Task 1:

We observe in figure 7 that the loss of training set converges to the optimum minima. The hyperparameters were tried and tested for various values and the following values were concluded:

Number of nodes – 128, Batch size – 500, epochs – 25, learning rate – 0.5. The weights and biases were initialized randomly at the beginning of the training. Furthermore we observe, that with only a slight overfitting, the loss of the validation data reaches the optimum minima and hence we can conclude that the training of the model is occurring as expected.

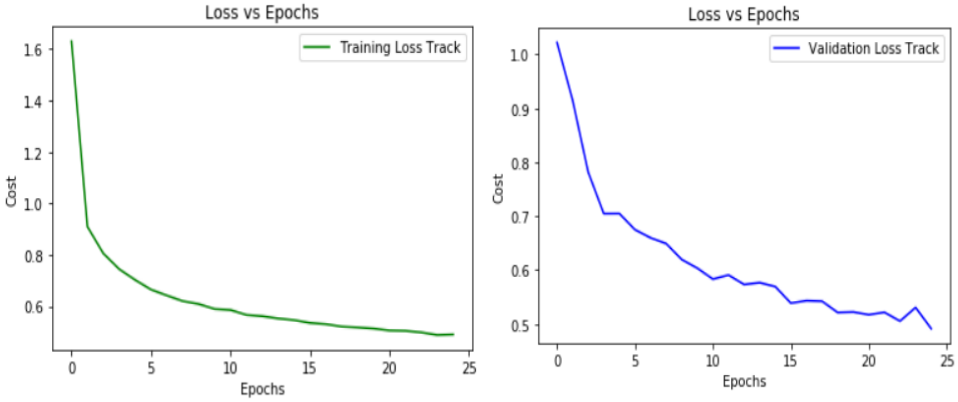

Figure 9 Loss VS Epoch for Training and Validation data set

The evaluation metric is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN}$$

The accuracy of the training and validation data sets were recorded as shown below:

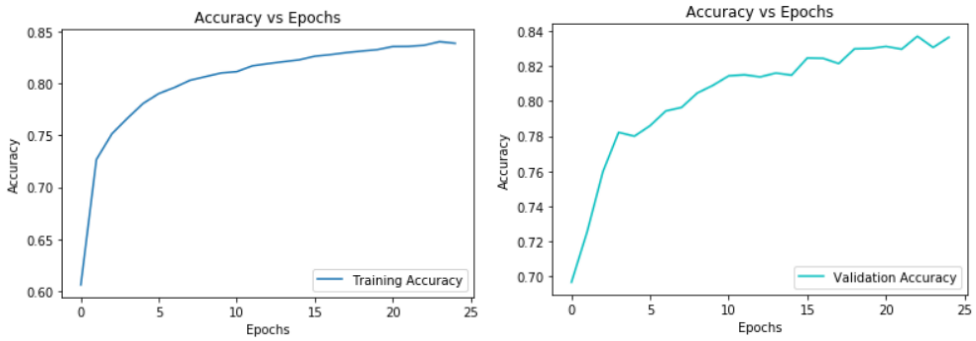

Figure 10 Accuracy VS Epoch for Training and Validation data set

On testing over the test data set, the model correctly predicted 82.6% of the predictions. Thus, given an input the model can predict and classify the image successfully.

```
Accuracy of Model:  82.6

Clasification report:
              precision    recall  f1-score   support

         top       0.78      0.78      0.78      1000
     trouser       0.96      0.94      0.95      1000
    pullover       0.69      0.74      0.71      1000
       dress       0.83      0.85      0.84      1000
        coat       0.73      0.74      0.74      1000
      sandal       0.93      0.90      0.91      1000
       shirt       0.60      0.53      0.56      1000
     sneaker       0.87      0.92      0.90      1000
         bag       0.92      0.94      0.93      1000
  ankle boot       0.93      0.92      0.92      1000

avg / total       0.82      0.83      0.82     10000

Confussion matrix:
 [[781   7  26  46   7   5 114   1  13   0]
 [  5 944   7  32   6   0   5   0   1   0]
 [ 15   3 741  10 125   0  95   0  11   0]
 [ 34  20  14 846  40   1  39   0   5   1]
 [  2   3 123  34 742   1  86   0   9   0]
 [  2   0   1   0   0 895   0  64   6  32]
 [162   4 141  40  90   4 529   0  30   0]
 [  0   0   0   0   0  37   0 924   2  37]
 [  2   1  22   6   4   3  15   9 938   0]
 [  0   1   0   0   0  17   2  60   0 920]]
```

Figure 11 Classification Report and Confusion Matrix for Task 1

Task 2:

In task 2 we use multiple layers to perform the training. A series of Dense() is used which is a fully connected layer in Keras. The following results were obtained:

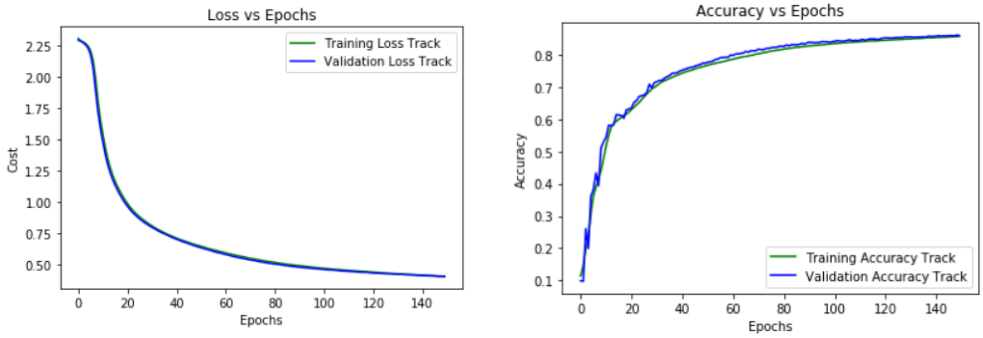

Figure 12 Loss vs Epoch and Accuracy vs Epoch for Task 2

Clasification report:

	precision	recall	f1-score	support
top	0.80	0.79	0.79	1000
trouser	0.98	0.94	0.96	1000
pullover	0.74	0.75	0.75	1000
dress	0.83	0.86	0.84	1000
coat	0.75	0.74	0.75	1000
sandal	0.92	0.93	0.92	1000
shirt	0.59	0.59	0.59	1000
sneaker	0.92	0.91	0.92	1000
bag	0.95	0.95	0.95	1000
ankle boot	0.92	0.94	0.93	1000
accuracy			0.84	10000
macro avg	0.84	0.84	0.84	10000
weighted avg	0.84	0.84	0.84	10000

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	multiple	100480
dense_1 (Dense)	multiple	16512
dense_2 (Dense)	multiple	16512
dense_3 (Dense)	multiple	1290

Total params: 134,794
Trainable params: 134,794
Non-trainable params: 0

Figure 13 Classification Report and Summary for Task 2

Confussion matrix:

```
[[797   1  15  54   1   4 118   0  10   0]
 [  0 943  11  38   4   0   4   0   0   0]
 [ 17   1 759  10 111   0  99   0   3   0]
 [ 29  11   7 861  34   1  53   0   4   0]
 [  0   0 113  30 756   0  98   0   3   0]
 [  0   0   0   1   0 923   0  47   4  25]
 [149   3 142  36  96   1 553   0  20   0]
 [  0   0   0   0   0  36   0 925   0  39]
 [  1   1   1   5   3   9  29   3 947   1]
 [  0   0   0   1   0  22   0  42   1 934]]
```

Figure 14 Confusion Matrix for Task 2

Task 3:

In task 3 we use convolution and max pooling to perform the training. The following results were obtained:

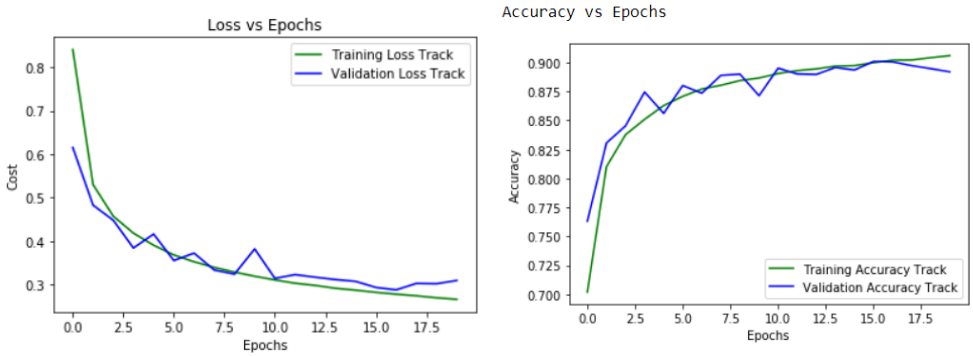

Figure 15 Loss vs Epoch and Accuracy vs Epoch for Task 3

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 64)	640
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_1 (Conv2D)	(None, 14, 14, 32)	18464
max_pooling2d_1 (MaxPooling2	(None, 7, 7, 32)	0
flatten (Flatten)	(None, 1568)	0
dense_4 (Dense)	(None, 10)	15690

Total params: 34,794
Trainable params: 34,794
Non-trainable params: 0

Clasification report:

	precision	recall	f1-score	support
top	0.90	0.71	0.79	1000
trouser	0.98	0.98	0.98	1000
pullover	0.91	0.73	0.81	1000
dress	0.93	0.85	0.89	1000
coat	0.79	0.83	0.81	1000
sandal	0.97	0.97	0.97	1000
shirt	0.58	0.83	0.68	1000
sneaker	0.95	0.94	0.94	1000
bag	0.98	0.97	0.97	1000
ankle boot	0.95	0.96	0.96	1000
accuracy			0.88	10000
macro avg	0.89	0.88	0.88	10000
weighted avg	0.89	0.88	0.88	10000

Figure 16 Summary and Classification Report for Task 3

```
Confussion matrix:
 [[708   2  12  17   5   1 247   0   8   0]
 [  0 982   0   9   4   0   4   0   1   0]
 [ 11   1 732   7  88   0 158   0   3   0]
 [ 13  11   9 849  53   0  62   0   3   0]
 [  0   1  32  10 835   0 120   0   2   0]
 [  0   0   0   0   0 975   0  16   0   9]
 [ 54   2  22  20  62   0 833   0   7   0]
 [  0   0   0   0   0  20   0 939   0  41]
 [  2   2   1   2   5   2   9   4 973   0]
 [  0   0   0   0   0   5   1  29   0 965]]
```

Figure 17 Confusion Matrix for Task 3

6 Conclusion

It can be concluded from our results that neural network can be used to solve multi-classification problem and classify Fashion MNIST data correctly. After training on approximately 60k samples, we were able to create a model that classified images with an accuracy of 82% using single layer neural network, 84% using multi-layer neural network and 88% when training the model using convolutional neural network. The above results clearly depict that CNNs are the most optimum way to train an artificial neural network model.

References

[1] https://stackabuse.com/creating-a-neural-network-from-scratch-in-python/

[2] https://www.techopedia.com/definition/33268/multi-layer-neural-network

[3] https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

[4] https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86

[5] https://www.cancer.org/cancer/breast-cancer/about/what-is-breast-cancer.html

[6] https://www.bcrf.org/breast-cancer-statistics-and-resources

[7] https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

[8] https://d1b10bmlvqabco.cloudfront.net/attach/jzsomflkppr3r4/isamd3soc56z/k0oa54u3szi5/Equation.pdf

[9] Andrew ng Machine Learning Course: https://www.coursera.org/learn/machine-learning/home/welcome

[10] https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/

[11] https://github.com/zalandoresearch/fashion-mnist

[12] https://www.tensorflow.org/tutorials/keras/classification

[13] https://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/

[14] https://www.quora.com/In-Keras-what-is-a-dense-and-a-dropout-layer

[15] https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/