CSE 574 – Implementing neural network and convolutional neural network for the task of classification.

Rohit Lalchand Vishwakarma
Department of Computer Science
University at Buffalo – State University of New York
Buffalo, NY 14214
rohitlal@buffalo.edu

Abstract

We are developing a neural network model and convolutional neural network model where we will be recognizing an image of the Fashion-MNSIT dataset in one of the ten classes. The data modelling is done in the python using the Google Colab. Here we are creating three different training models which are with one hidden layer, with multi-layer neural network and with convolutional neural network. We are using various activation function to feed forward through hidden layers and finally back propagate to update the weights. Our model will help to recognize an image of Fashion MNIST and predict it to its most approximated class labels.

1 Introduction

"What we want is machine that can learn from experience."
– **Alan Turing**,1947

1.1 Neural Network and its hidden Layers

In machine learning, we can view the Neural Network as the neurons network of the human body. Every network as neurons which takes the information and passes it to the neurons of the other neural network for human senses. Here, Neural Network can be viewed as a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form.

A hidden layer in an artificial neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function.

Here, we feed forward the input data on multiple hidden layers by applying weights where each hidden layers have their own activation function which is used to generate output from that hidden layers which will become input by applying weights for the next layer.

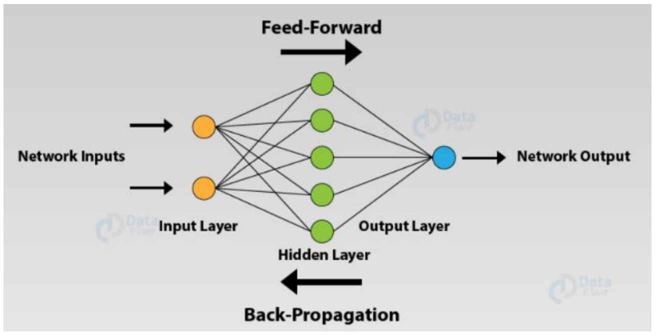


Fig 1.1 – Single Hidden Layer Neural Network

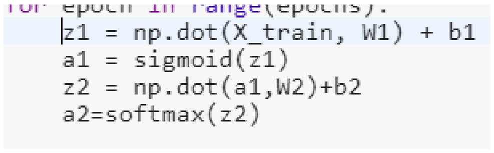
Such types of network are used to predict the output in the classification problem.
Classification problem is the problem when independent variables are continuous and dependent variable is in categorical form

1.2 Using Hidden Layers of the Neural Network and Convolutional Network in our model

Here we have to train model with one hidden layer, multiple layers of neural network and convolutional neural network.

1.2.1 One hidden Layer

Here we are using our training set and after applying weights to it we are passing it through sigmoid function on hidden layer with 100 neurons. These outputs are applied with new weights and given to the last layer i.e the output layer where SoftMax activation function is used to generate a output with 10 class labels. After that we are backpropagating so that we can update the weights and train the model unless we get the good accuracy.



```
for epoch in range(epochs):
    z1 = np.dot(X_train, W1) + b1
    a1 = sigmoid(z1)
    z2 = np.dot(a1,W2)+b2
    a2=softmax(z2)
```

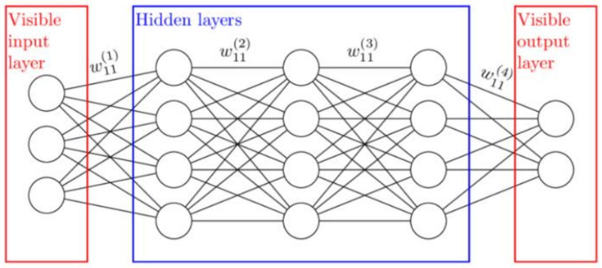
Fig 1.2.1.1-FeedForward

$$\boxed{\Delta W^{[1]} = (a^{[2]}-y)\, W^{[2]} a^{[1]} (1-a^{[1]})\, x} \qquad \boxed{\Delta W^{[2]} = (a^{[2]}-y)\, a^{[1]}}$$

Fig 1.2.1.2-BackPropogation

1.2.2 Multi-Layer Neural Network

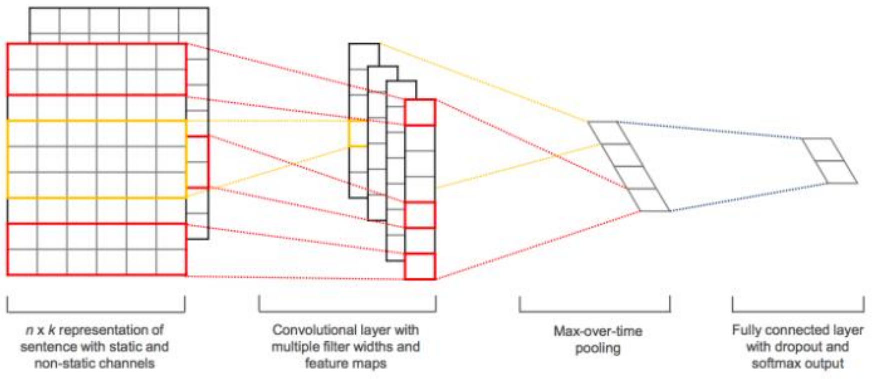
Here we are using our training set and after applying weights to it we are passing it through sigmoid function on hidden layer with 100 neurons. These outputs are applied with new weights and given to the input layer of the 2nd layer with 100 neurons where ReLu function is applied and after applying it with new weights it is passed to the 3rd layer again with 100 neurons and Relu function is applied there.Finally these outputs are applied with new weights and given to the last layer i.e the output layer where SoftMax activation function is used to generate a output with 10 class labels.



1.2.2 Neural Network with 3 hidden layers

1.2.3 Convolutional Neural Network

Here we are using our training set and after applying weights to it we are passing it through sigmoid function on hidden layer with 100 neurons. These outputs are applied with new weights and given to the input layer of the 2nd layer with 100 neurons where ReLu function is applied. Finally, these outputs are applied with new weights and given to the last layer i.e. the output layer where SoftMax activation function is used to generate a output with 10 class labels.


n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

1.2.3 Convolutional Neural Network with one hidden layer

2 Dataset[1]

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 below), and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

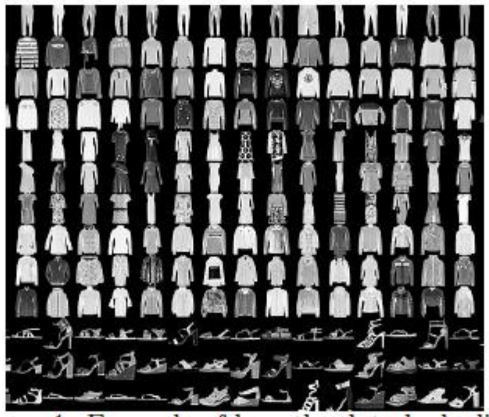


Fig 2.1 – Fashion Data 28X28 for 13 rows

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

3 Preprocessing

3.1 Importing Dataset

The Data that we imported is of matrix form 28X28 having 60000 entries of training. In order to use in the model, we need to first convert the matrix into single dimension. Hence, we multiplied that data and generated 784 rows. So, our final data for training is 60000X784.Similarly we will be doing the same for test data. These data we will be using in One hidden Layer and Multilayer Neural Network. For Convolutional Neural Network the 28X28 is used since the model can use the matrix. i.e. 3-dimensional data.

3.2 Splitting Test Set into Validation and Test

As we need to validate the training data we will be using the test data of 10000X784 (Feature_variable) to be split into 5000X784 (Feature_variable) each for validation and test data. Similarly, 10000X1(Class_Data) into 5000X1 each for validation and testing.

3.3 One hot encoded data

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. In our model we need to convert the Class label data into one hot encoded data. This is because our class label data contains only one value for a row which specifies which class it belongs to. However, we are predicting it for 10 classes, so we will need to convert it from 10000X1 to 10000X10 data where 10 represent number of classes. We will just have to set value 1 for a index value representing class number and rest as 0.

4 Architecture

3.1 Multi-Layer Neural Network

Neural network has multiple layers with neurons at each layer and each layer is responsible for training and learning from data. Each layer has activation function which is applied on input data from previous layer and given as output to next hidden layer. At the output we apply back propagation to updates the weight and sent back in the neural network.

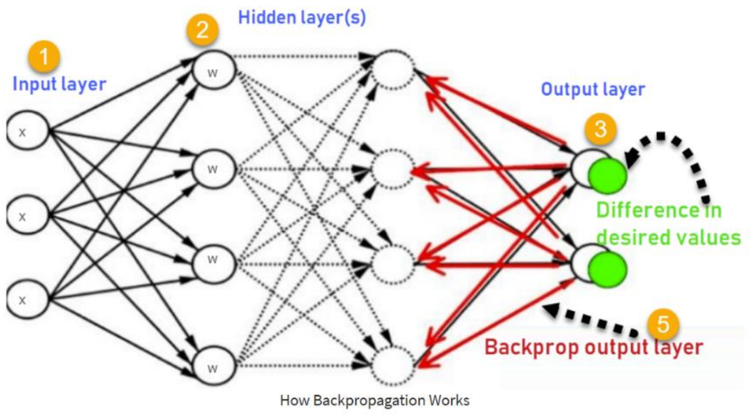


Fig 3.1-Neural Network Model

3.2 Convolutional Neural Network

It 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.

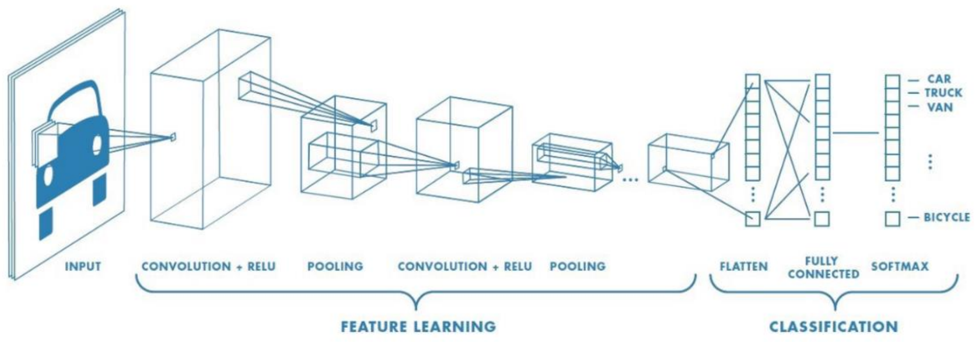


Fig 3.2-Convolutional Neural Network Model

3.2 Cross Entropy

We are using cross entropy function to generate the loss between our predicted and actual value.

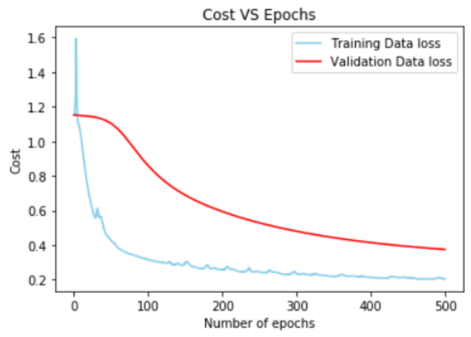
$$L(a^{[2]}, y) = -\Sigma y \log a^{[2]}$$

5 Results

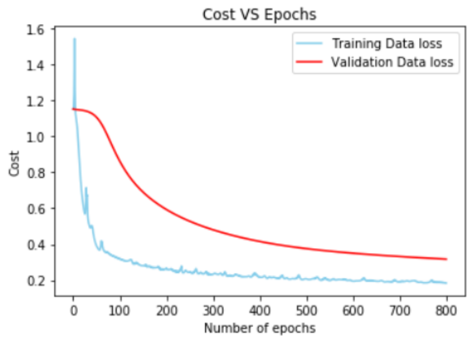
We got the following results for our models

5.1 One Hidden Layer Neural Network (from scratch)

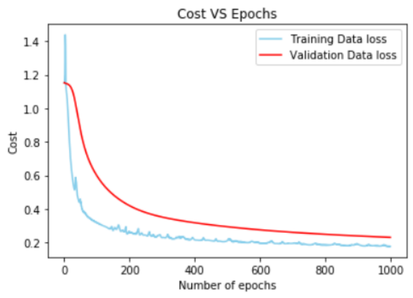
```
Accuracy with training data is:  0.8498

Accuracy with validation data is:  0.7314


<matplotlib.legend.Legend at 0x7ff22e74f978>
```



```
Accuracy with training data is:  0.8670666666666667

Accuracy with validation data is:  0.7684


<matplotlib.legend.Legend at 0x7ff22e733cc0>
```



```
Accuracy with training data is:  0.8710166666666667

Accuracy with validation data is:  0.8378

<matplotlib.legend.Legend at 0x7ff22f174b38>
```



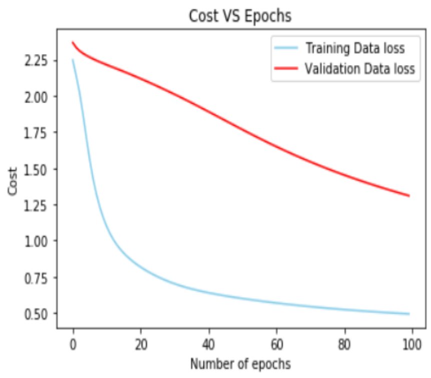
```
confusion_matrix(Y_test,Y_predicted)

array([[441,   1,   6,  25,   2,   2,  15,   0,   5,   0],
       [  3, 473,   1,  13,   2,   0,   1,   0,   0,   0],
       [ 10,   2, 396,   6,  70,   1,   9,   0,   5,   0],
       [ 18,   7,   5, 442,  14,   0,   3,   0,   1,   0],
       [  1,   1,  43,  20, 418,   0,  11,   0,   1,   0],
       [  0,   0,   0,   1,   0, 426,   0,  29,   0,  21],
       [112,   1,  77,  27,  69,   1, 208,   0,  14,   0],
       [  0,   0,   0,   0,   0,  10,   0, 480,   0,  22],
       [  2,   0,   3,   5,   1,   1,   1,   2, 476,   0],
       [  0,   0,   0,   0,   0,   3,   0,  18,   1, 515]])
```

```
              precision    recall  f1-score   support

     class 0       0.75      0.89      0.81       497
     class 1       0.98      0.96      0.97       493
     class 2       0.75      0.79      0.77       499
     class 3       0.82      0.90      0.86       490
     class 4       0.73      0.84      0.78       495
     class 5       0.96      0.89      0.93       477
     class 6       0.84      0.41      0.55       509
     class 7       0.91      0.94      0.92       512
     class 8       0.95      0.97      0.96       491
     class 9       0.92      0.96      0.94       537

    accuracy                           0.85      5000
   macro avg       0.86      0.86      0.85      5000
weighted avg       0.86      0.85      0.85      5000
```

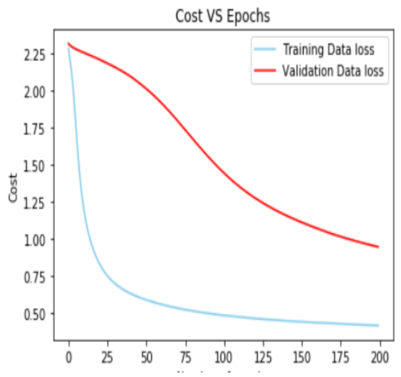
```
Accuracy with test data is:  85.5 %
```

5.2 Multi-Layer Neural Network (using high level libraries eg: keras)

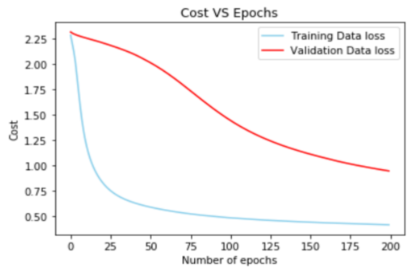
```
Accuracy with training data
 82.51833333333335 %

5000/5000 [==============================] - 1s 118us/step
Accuracy with validation data
 61.260000000000005 %

<matplotlib.legend.Legend at 0x7f6835aec358>
```

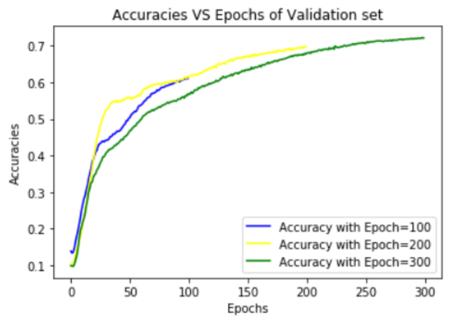


```
Accuracy with training data
 85.16666666666667 %

5000/5000 [============================] - 1s 180us/step
Accuracy with validation data
 69.64 %

<matplotlib.legend.Legend at 0x7f68358ad940>
```



```
Accuracy with training data
 86.33166666666666 %


5000/5000 [==============================] - 1s 193us/step
Accuracy with validation data
 72.04 %


<matplotlib.legend.Legend at 0x7f68356b9390>
```





```
confusion_matrix(Y_test,model_keras.predict_classes(X_test)) #to
```

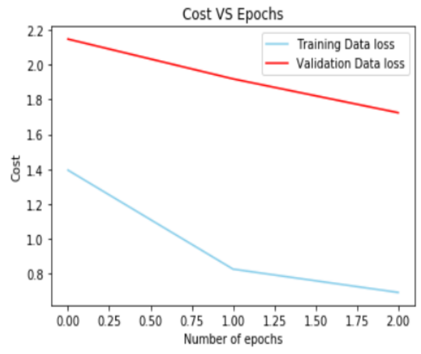
```
array([[405,   1,   4,  28,   2,   1,  52,   0,   4,   0],
       [  3, 466,   5,  15,   3,   0,   1,   0,   0,   0],
       [  7,   2, 364,   6,  72,   0,  45,   0,   3,   0],
       [ 13,   7,   3, 437,  12,   0,  18,   0,   0,   0],
       [  0,   1,  41,  21, 385,   0,  45,   0,   2,   0],
       [  0,   0,   0,   1,   0, 432,   0,  26,   0,  18],
       [ 65,   1,  54,  23,  42,   1, 308,   0,  15,   0],
       [  0,   0,   0,   0,   0,  16,   0, 471,   0,  25],
       [  1,   0,   2,   6,   2,   3,   9,   2, 466,   0],
       [  0,   0,   0,   0,   0,   9,   0,  22,   1, 505]])
```

```
              precision    recall  f1-score   support

     class 0       0.75      0.89      0.81       497
     class 1       0.98      0.96      0.97       493
     class 2       0.75      0.79      0.77       499
     class 3       0.82      0.90      0.86       490
     class 4       0.73      0.84      0.78       495
     class 5       0.96      0.89      0.93       477
     class 6       0.84      0.41      0.55       509
     class 7       0.91      0.94      0.92       512
     class 8       0.95      0.97      0.96       491
     class 9       0.92      0.96      0.94       537

    accuracy                           0.85      5000
   macro avg       0.86      0.86      0.85      5000
weighted avg       0.86      0.85      0.85      5000
```

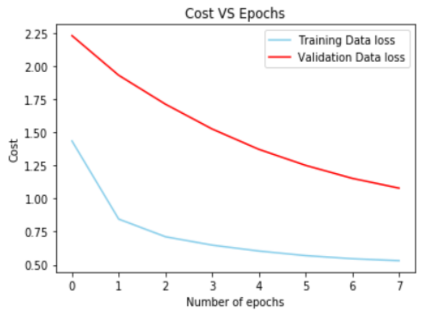
```
Accuracy with test data :
 85.02 %
```

5.3 Convolutional Neural Network (high level libraries eg: keras)

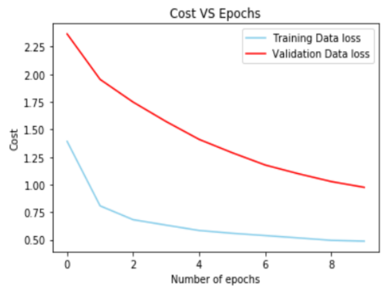
```
Accuracy with training data
 75.78166666666667 %


5000/5000 [==============================] - 3s 603us/step
Accuracy with validation data
 62.739999999999995 %


<matplotlib.legend.Legend at 0x7fe881742908>
```

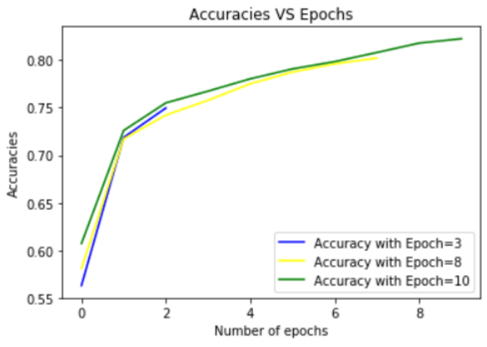


```
Accuracy with training data
 81.15333333333334 %


5000/5000 [==============================] - 3s 607us/step
Accuracy with validation data
 71.32 %


<matplotlib.legend.Legend at 0x7fe881505a90>
```



```
Accuracy with training data
 83.235 %


5000/5000 [==============================] - 4s 701us/step
Accuracy with validation data
 70.66 %


<matplotlib.legend.Legend at 0x7fe8812cf588>
```





```
confusion_matrix(Y_test,Y_predicted)

array([[441,   1,   6,  25,   2,   2,  15,   0,   5,   0],
       [  3, 473,   1,  13,   2,   0,   1,   0,   0,   0],
       [ 10,   2, 396,   6,  70,   1,   9,   0,   5,   0],
       [ 18,   7,   5, 442,  14,   0,   3,   0,   1,   0],
       [  1,   1,  43,  20, 418,   0,  11,   0,   1,   0],
       [  0,   0,   0,   1,   0, 426,   0,  29,   0,  21],
       [112,   1,  77,  27,  69,   1, 208,   0,  14,   0],
       [  0,   0,   0,   0,   0,  10,   0, 480,   0,  22],
       [  2,   0,   3,   5,   1,   1,   1,   2, 476,   0],
       [  0,   0,   0,   0,   0,   3,   0,  18,   1, 515]])
```

```
              precision    recall  f1-score   support

     class 0       0.75      0.89      0.81       497
     class 1       0.98      0.96      0.97       493
     class 2       0.75      0.79      0.77       499
     class 3       0.82      0.90      0.86       490
     class 4       0.73      0.84      0.78       495
     class 5       0.96      0.89      0.93       477
     class 6       0.84      0.41      0.55       509
     class 7       0.91      0.94      0.92       512
     class 8       0.95      0.97      0.96       491
     class 9       0.92      0.96      0.94       537

    accuracy                           0.85      5000
   macro avg       0.86      0.86      0.85      5000
weighted avg       0.86      0.85      0.85      5000
```

```
Accuracy with test data :
 82.56 %
```

6 Conclusion

We can conclude that with one hidden layer prediction accuracy for the test data 85.5 % with multi-layer neural network using keras prediction accuracy for the test data is 85.02 % and with convolutional neural network is 82.56%.

Thus by increasing the nodes of our neurons and running for more epochs and choosing the appropriate learning rate we can make our model to learn better and predict with more better accuracy.

7 References

1] iml_project2.pdf (Document for Project 2)

2] Equations2.pdf (Recitation pdf or feedforward, Cross entropy and back propagation).

3]https://www.techopedia.com/definition/33264/hidden-layer-neural-networks#targetText=A%20hidden%20layer%20in%20an,output%20through%20an%20activation%20function.

4] https://deepai.org/machine-learning-glossary-and-terms/neural-network#targetText=%2C%20is%20a%20computational%20learning%20system,output%2C%20usually%20in%20another%20form.&targetText=Neural%20networks%20are%20just%20one,used%20in%20machine%20learning%20algorithms.

5] https://medium.com/machine-learning-for-humans/neural-networks-deep-learning-cdad8aeae49b