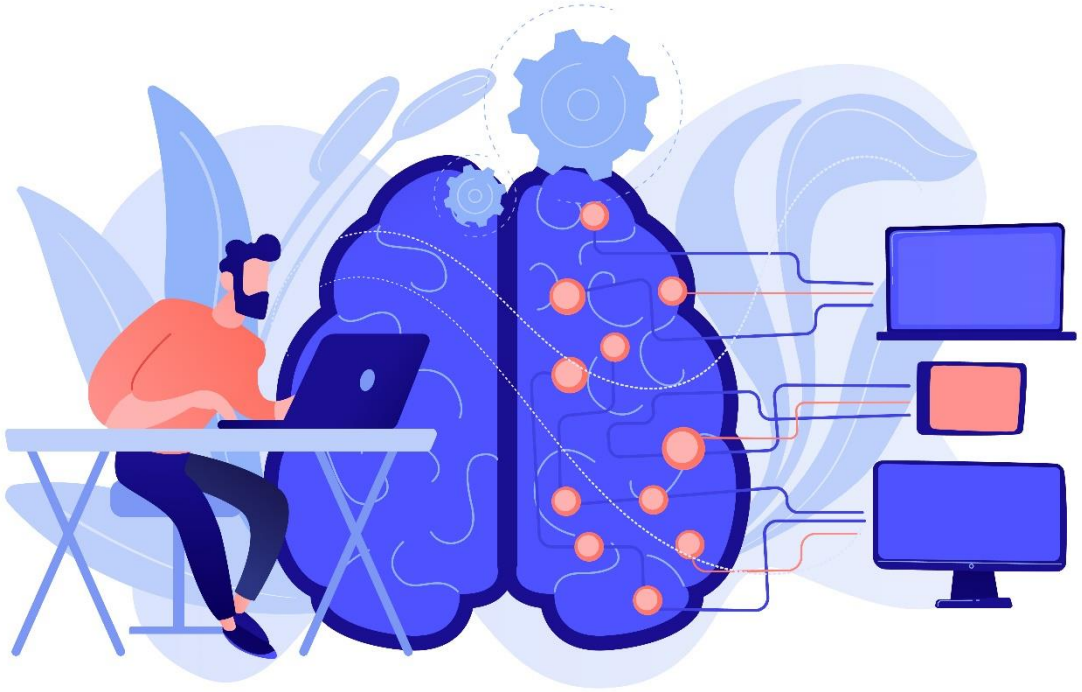
Middlesex University Mauritius



Classification of Handwritten Digits – Machine Learning

CST3170 – Artificial Intelligence

Farhaan Beeharry – M00681483

4th March 2021

Lecturer: Suraj Juddoo

Table of contents

Table of figures

Table of equations

Table of tables

Introduction

This coursework is to build a machine learning system to classify handwritten digits from the UCI digit tasks. The data is from the University of California at Irvine's Machine Learning Repository. The other aim of this project is to run a two-fold test and report the results. Algorithms such as the neural network, nearest neighbour and the k-nearest neighbour were used to be able to compare the results and use the best and most efficient algorithm. The dataset provided by the UCI is a normalised bitmap from a pre-printed form. A two-fold test is used to find the accuracy and efficiency of the system. A two-fold validation is a resampling method. It divides the available set of examples/samples randomly into two parts; one for the training of the neural network and one for the validation of testing of the trained algorithm. The options and variables of the most-fit algorithm will be tweaked and explored to find the optimum solution for this project. The best solution should go beyond 97.72% accuracy which is the accuracy on the testing set provided by the UCI with k-nearest neighbour using Euclidean distance as the metric.

Neural Network (Multi-Layer Perceptron)

Definition

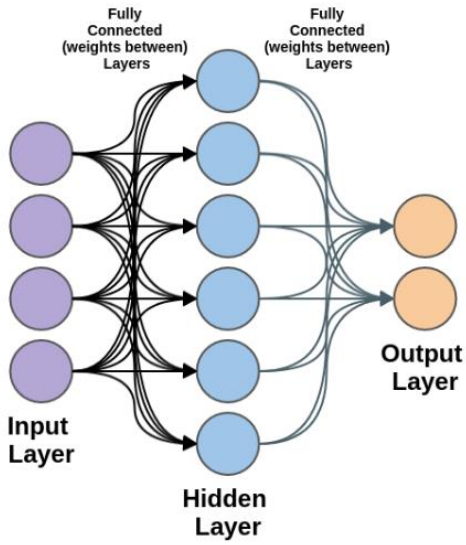


Figure 1 Multi-layer Perceptron

(Walters, 2019)

A multi-layered perceptron is among the most used neural network framework in deep learning. It is also referred to as "vanilla" neural network. It can be used for image identification, stock analysis and spam detection. MLP consists of neurons connected to each other transferring informing among themselves. It imitates the human brain. Each neuron carries a value and the network is broken into three layers namely the input layer, the hidden layer and the output layer.

The input layers

The input layer is the start layer of the network. It picks up an input value to be used to create an output value.

The hidden layers

The hidden layers calculate, compute and operate on the input layer to produce a valid outcome. A neural network should have at least one hidden layer.

The output layers

The output layer only displays the productive outcome of the hidden layer(s).

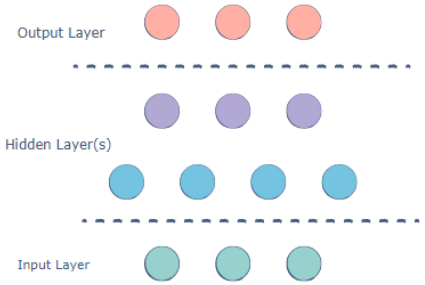


Figure 2 Multi-layer perceptron's layers

(Edpresso Team, 2021)

How does MLP work?

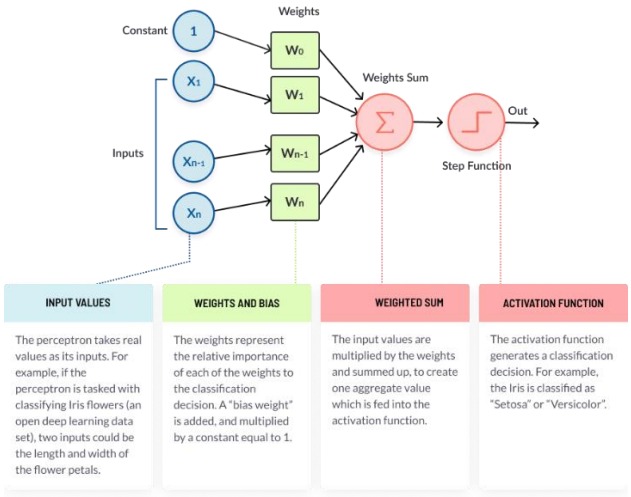


Figure 3 Steps of MLP

(MissingLink.ai, 2021)

A perceptron follows guided steps as follows:

1. Picks up inputs, multiplies the inputs by their weights and calculate their sum.
2. Adds a bias factor; multiply the sum from step 1 by a weight.
3. Feeds the sum through the activation process. Activation function has properties such as its non-linear nature which enables it to train complex neural networks (MissingLink.ai, 2021).
4. The result is the perceptron output.

MLP learns by calculating the correct values for the weights between the input layer and the output layer (Noriega, 2005).

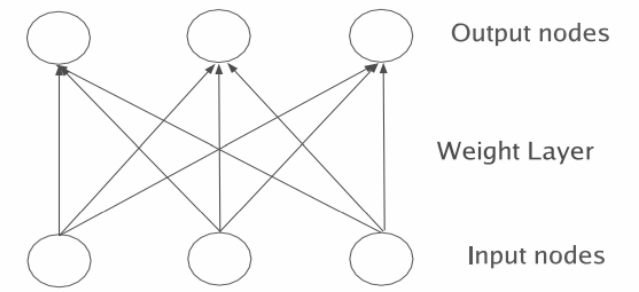
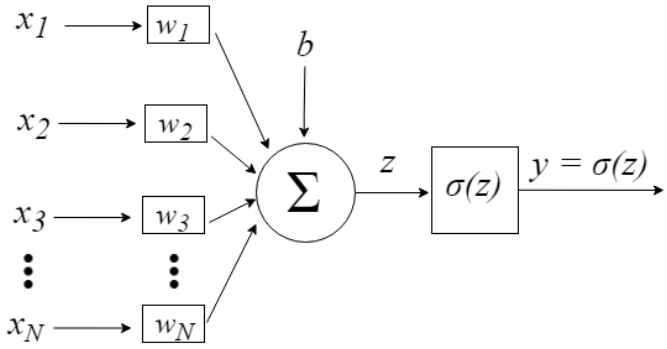


Figure 4 Learning of MLP

(Noriega, 2005)

Multi-layer Perceptron MLP Algorithm described



Equation 1 Input values and weights

(Leite, 2018)

The array $\mathbf{x = [x_1, x_2, x_3, \ldots, x_n]}$ represents the input signals (in this case, representing the handwritten digits). These values are fed to the neuron and are multiplied by their designated

synaptic weights (array **w = [w1, w2, w3, …, wn]**), resulting in the value **z**, which is the activation potential (Leite, 2018) according to the following expression:

$$z = \sum_{i=1}^{N} x_i w_i + b$$

Equation 2 Calculation of the Activation Potential (z)

(Leite, 2018)

The extra value **b** is not affected by the input array and is called the bias value. This biased value provides freedom to the model. The value **z** is passed to the non-linear activation function which is responsible for limiting that value to a certain range, generating the final value **y** in the neuron. As one single neuron cannot do much, a batch of neurons are combined in a structured manner composed of layers (the input layer, the hidden layer(s) and the output layer) with each one having different number of neurons. This structured network forms a neural network and is referred to as the Multi-Layer Perceptron (MLP) (Leite, 2018).

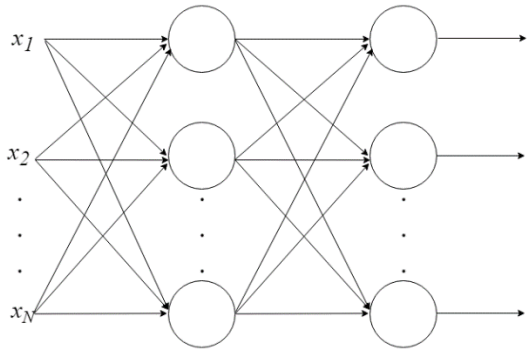


Figure 5 Neural Network

(Leite, 2018)

Activation function (Signum function)

The processing element is the sum of the values followed by a non-linear threshold. Its equation is:

$$y = f(net) = f\left(\sum_i w_i x + b\right)$$

Equation 3 Processing Element equation

Where $\mathbf{w}_i$ are the weights and **b** is the bias term. The activation function *f* is a threshold function defined by:

$$f(net) = \begin{cases} 1 & for \quad net \geq 0 \\ -1 & for \quad net < 0 \end{cases}$$

Equation 4 Signum function

Which is commonly referred to as the signum function (Principe, 2016).

K-Nearest Neighbour vs Multi-layer Perceptron (Neural Network)

What is K-Nearest Neighbour?

K-Nearest Neighbour (K-NN) is among the simplest algorithms used in Machine Learning. It is based on the Supervised Learning technique. K-NN is mostly used for classification and is a non-parametric algorithm (does not make assumptions). K-NN assumes the similarity between 2 cases (new and available) and put the new case into the category which matches or is most similar to with the available categories.

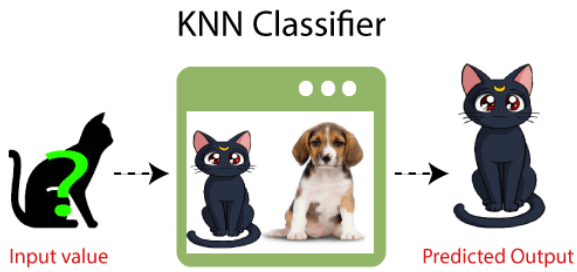


Figure 6 How K-NN works?

(JavaTpoint, 2018)

The equation for calculating the weight is given as **1/(1 + de)** where de is the Euclidean distance between an unknown value and one of its k nearest neighbours (Kaushik, et al., 2005).

Advantages of K-NN

- Simple and easy to implement
- Can easily solve noisy data
- The larger the training data, the higher the effective rate

Disadvantages of K-NN

- Always need to have a value **k** which can be complex at times
- As distance has to be calculated between all training points, the computational cost is high.

Comparison of K-NN and Multi-layer Perceptron (Neural Network)

The training set and data set has both been put to test with the two algorithms (K-NN and MLP). The recognition performance of the two algorithms is recorded and displayed below.

Recognition performance of K-NN

```
---------- Final Results ------------
Test Accuracy for KNN: 98.22
Training Accuracy for KNN: 98.58
BUILD SUCCESSFUL (total time: 1 minute 20 seconds)
```

Figure 7 Application Run No.1 Result

```
---------- Final Results ------------
Test Accuracy for KNN: 98.22
Training Accuracy for KNN: 98.58
BUILD SUCCESSFUL (total time: 1 minute 21 seconds)
```

Figure 8 Application Run No.2 Result

```
---------- Final Results ------------
Test Accuracy for KNN: 98.22
Training Accuracy for KNN: 98.58
BUILD SUCCESSFUL (total time: 1 minute 24 seconds)
```

Figure 9 Application Run No.3 Result

Test Number	Training Accuracy (%)	Test Accuracy (%)
1	98.58	98.22
2	98.58	98.22
3	98.58	98.22
Average	98.58	98.22

Table 1 K-NN Performance Comparison

The K-NN algorithm always outputs the same values for the test accuracy and the training accuracy as it is always going through the same path to predict the handwritten digits.

Comparison of MLP at different rates

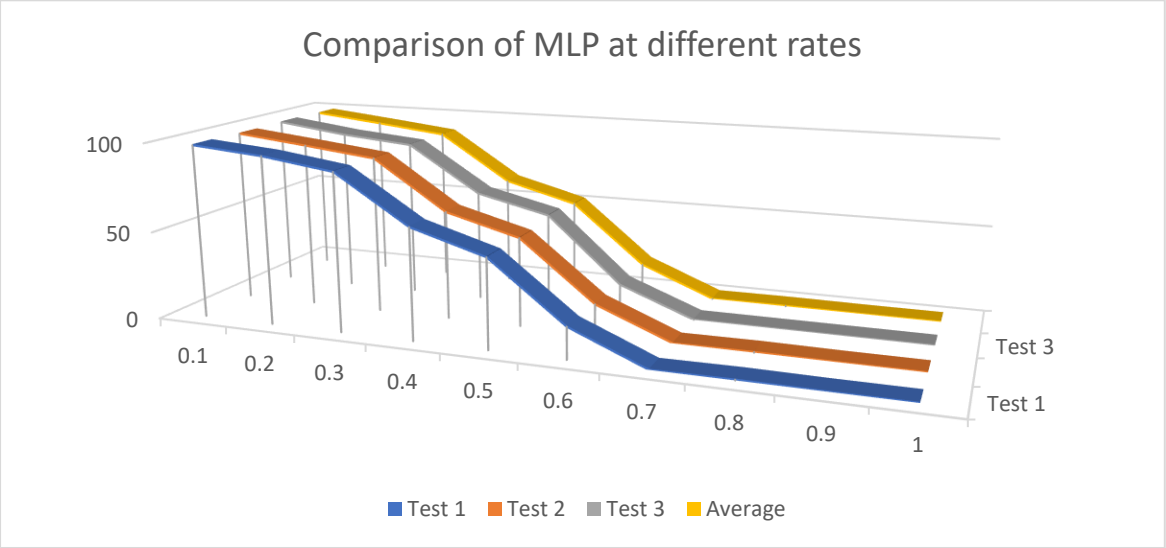


Figure 10 Comparison of MLP at different Learning Rates

Different tests at learning rates at intervals of 0.1 has been run 3 times each and a graph has been plotted. Comparing the lines obtained, a line of best fit (average) has been drawn. It can be observed that the best performance is obtained at learning rate 0.1 and the average value is 98.18%.

Recognition performance of MLP (neural network)

As the best outcomes come from learning rate 0.1, several tests have been done with this value to be able to come to a conclusion. To run the tests, the epochs value has been set to 660, input size to 64, hidden size to 256 and output size to 16. Those tests results are shown below.

```
---------- Final Results ------------
Train Accuracy for Neural Network: 99.96
Test Accuracy for Neural Network: 98.18
BUILD SUCCESSFUL (total time: 1 minute 8 seconds)
```

Figure 11 Application Run No.1 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 98.17
Test Accuracy for Neural Network: 97.92
BUILD SUCCESSFUL (total time: 1 minute 10 seconds)
```

Figure 12 Application Run No.2 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 100.0
Test Accuracy for Neural Network: 98.45
BUILD SUCCESSFUL (total time: 1 minute 13 seconds)
```

Figure 13 Application Run No.3 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 99.86
Test Accuracy for Neural Network: 97.95
BUILD SUCCESSFUL (total time: 1 minute 8 seconds)
```

Figure 14 Application Run No.4 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 99.71
Test Accuracy for Neural Network: 97.90
BUILD SUCCESSFUL (total time: 1 minute 9 seconds)
```

Figure 15 Application Run No.5 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 100.0
Test Accuracy for Neural Network: 98.35
BUILD SUCCESSFUL (total time: 1 minute 13 seconds)
```

Figure 16 Application Run No.6 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 100.0
Test Accuracy for Neural Network: 98.71
BUILD SUCCESSFUL (total time: 1 minute 11 seconds)
```

Figure 17 Application Run No.7 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 100.0
Test Accuracy for Neural Network: 98.44
BUILD SUCCESSFUL (total time: 1 minute 10 seconds)
```

Figure 18 Application Run No.8 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 99.40
Test Accuracy for Neural Network: 97.90
BUILD SUCCESSFUL (total time: 1 minute 7 seconds)
```

Figure 19 Application Run No.9 Result

```
---------- Final Results ------------
Train Accuracy for Neural Network: 100.0
Test Accuracy for Neural Network: 98.20
BUILD SUCCESSFUL (total time: 1 minute 12 seconds)
```

Figure 20 Application Run No.10 Result

Test Number	Training Accuracy (%)	Test Accuracy (%)
1	99.96	98.18
2	98.17	97.92
3	100.0	98.45
4	99.86	97.95
5	99.71	97.90
6	100.0	98.35
7	100.0	98.71
8	100.0	98.44
9	99.40	97.90
10	100.0	98.20
Average	99.71	98.20

Table 2 MLP Performance Comparison

As the MLP algorithm uses the learning technique to generate an outcome, it always outputs different values. On average, the MLP outcome is 98.20% and the training accuracy is 99.71%.

K-NN vs MLP

From the above tests carried out, the training accuracy of the K-NN algorithm is 98.58% and its test accuracy is 98.22%. And the training accuracy of the MLP algorithm is 99.71% and its test accuracy is 98.20%.

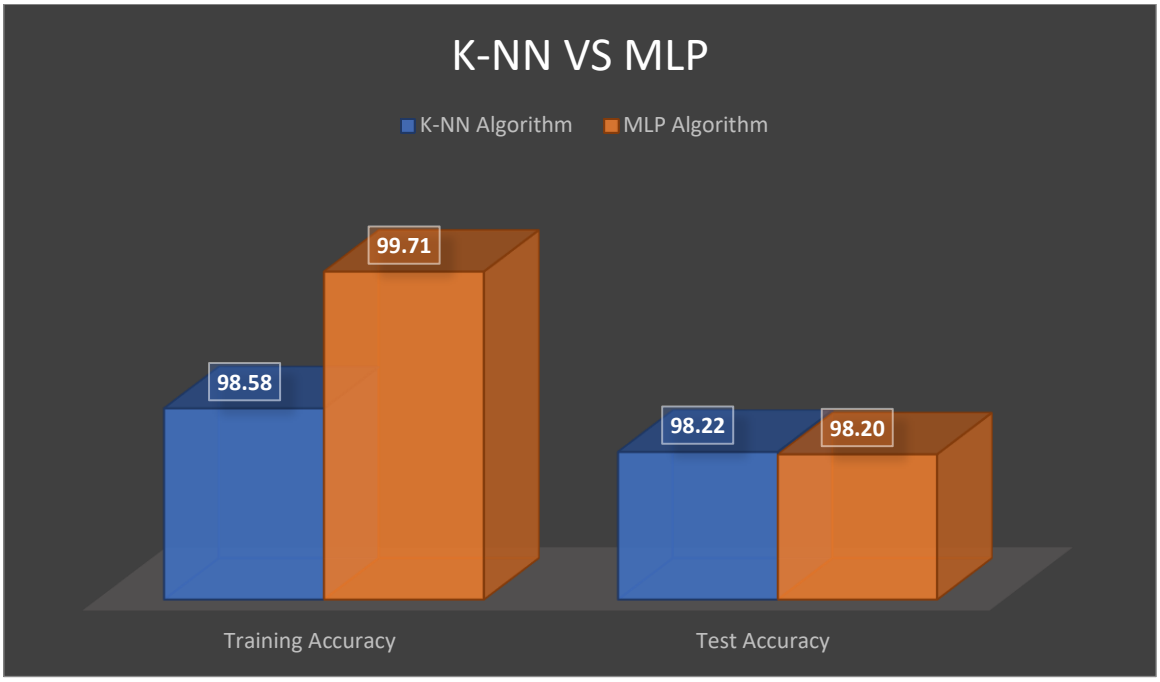


Figure 21 K-NN algorithm vs MLP algorithm

Comparing the training accuracy of the two algorithms used and tested, MLP surpasses K-NN by 1.13% but considering the test accuracy, K-NN surpasses MLP by 0.02%.

As the test accuracy of both algorithms are tantamount, a conclusion can be reached by comparing the training accuracy. In this case, while comparing handwritten digits, the MLP is more effective and accurate while training the algorithm. Therefore, with larger dataset of values, MLP will be more efficient and will produce more accurate results compared to K-NN.

Conclusion

The above results suggest that, for handwritten digits recognition and classification, a multi-layer perceptron can be implemented to produce a remarkable performance that is comparable to that of the K-NN classifier. With larger sets of data, MLP produces a way-ahead performance of both training and testing accuracy. The possible reason for the better performance of the MLP can be explained as follows. As the K-Nearest Neighbour algorithm depends on the comparing set of values, the value of k and the Euclidean distance, the freedom of tweaking to improve its performance is limited. It depends on the similarity of the value compared to the categories defined by the reference set. However, a tweakable number of hidden layers, weights, neurons and learning rate of the Multi-layer Perceptron algorithm gives the chance of optimising its performance. Therefore, the classification of handwritten digits using Multi-layer Perceptron can be enhanced and be more productive and effective compared to the K-Nearest Neighbour classifier. Also, MLP has greater efficiency rate with lower computational cost (Mustapha, et al., 2014).

Self-Marking Sheet

Available Points	Area	Self-Allocated Points	Justification
20	Report	18	All required topics have been covered in the report and algorithm well described
20	Running Code	20	There are no errors in the code + validation for input has been included
20	Quality of Code	18	Code has been well indented and proper use of methods/functions and arguments + appropriate comments where needed
20	Quality of Algorithm	17	Good implementation of several algorithms (K-NN and MLP) and tested to compare results
20	Quality of Results	19	Results are clear, well explained and well compared using graphs and tables + relevant conclusion on the comparison

References

Edpresso Team, 2021. *Edpresso a shot of dev knowledge.* [Online]
Available at: https://www.educative.io/edpresso/what-is-a-multi-layered-perceptron
[Accessed 23 02 2021].

Gessa, D., 2012. *mlp-java.* [Online]
Available at: https://github.com/dakk/mlp-java/blob/master/src/multilayersperceptronlib/MultiLayerPerceptron.java
[Accessed 8 02 2021].

JavaTpoint, 2018. *K-Nearest Neighbor(KNN) Algorithm for Machine Learning.* [Online]
Available at: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning#:~:text=K%2DNearest%20Neighbour%20is%20one,similar%20to%20the%20available%20categories.
[Accessed 26 02 2021].

Kaushik, R. et al., 2005. Comparison of the Multi Layer Perceptron and the Nearest Neighbor Classifier for Handwritten Numeral Recognition. *Journal of Information Science and Engineering,* Volume 21-6.

Leite, T. M., 2018. *Neural Networks, Multilayer Perceptron and the Backpropagation Algorithm.* [Online]
Available at: https://medium.com/@tiago.tmleite/neural-networks-multilayer-perceptron-and-the-backpropagation-algorithm-a5cd5b904fde
[Accessed 26 02 2021].

MissingLink.ai, 2021. *Perceptrons and Multi-Layer Perceptrons: The Artificial Neuron at the Core of Deep Learning.* [Online]
Available at: https://missinglink.ai/guides/neural-network-concepts/perceptrons-and-multi-layer-perceptrons-the-artificial-neuron-at-the-core-of-deep-learning/
[Accessed 23 02 2021].

Mustapha, D., Boubaker, D., Amir, N. & Patrick, S., 2014. *Multi-Layer Perceptron Neural Network and Nearest Neighbor Approaches for Indoor Localization.* San Diego, CA, USA, IEEE.

Noriega, L., 2005. *Multilayer Perceptron Tutorial.* Beaconside Staffordshire ST18 0DG, School of Computing.

Patidar, P., 2019. *Optical-recognition-of-handwritten-digits-dataset.* [Online]
Available at: https://github.com/patidarparas13/Optical-recognition-of-handwritten-digits-dataset/blob/master/Optical-recognition-of-handwritten-digits-dataset.ipynb
[Accessed 8 02 2021].

Principe, D. J., 2016. *Neural Networks for Signal Processing.* [Online]
Available at: http://www.cnel.ufl.edu/courses/EEL6814/chapter3.pdf
[Accessed 27 02 2021].

Tiwari, K., 2016. *Optical-Recognition-of-Handwritten-Digits-Dataset-in-Java.* [Online]
Available at: https://github.com/khushi4tiwari/Optical-Recognition-of-Handwritten-Digits-Dataset-in-Java/blob/master/src/handwritten/optdigits.java
[Accessed 10 02 2021].

Walters, A. G., 2019. *Classify Sentences via a Multilayer Perceptron (MLP).* [Online]
Available at: https://austingwalters.com/classify-sentences-via-a-multilayer-perceptron-mlp/
[Accessed 27 02 2021].