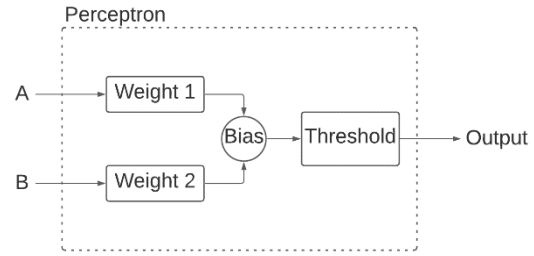
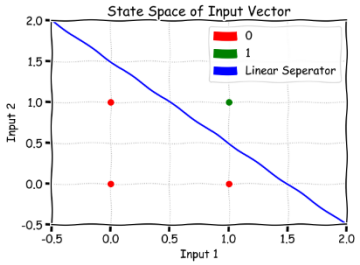
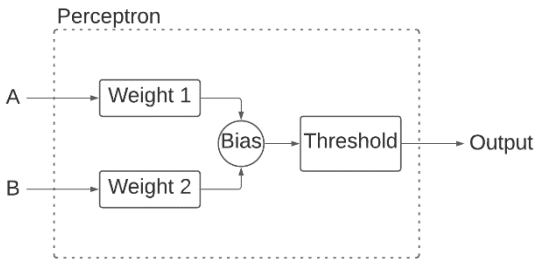
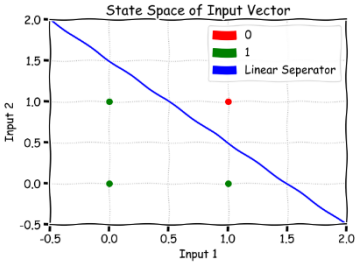
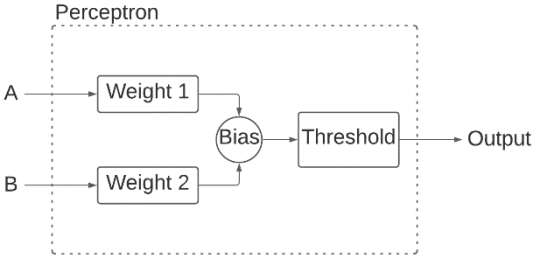
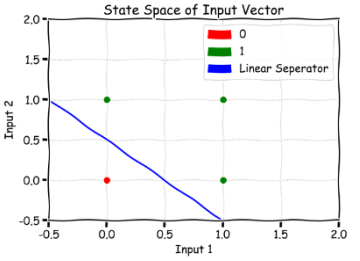
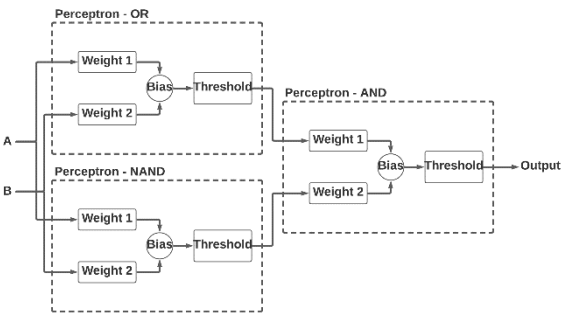
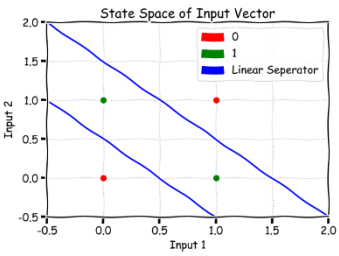
Coursework A – Neural Networks

Exercise One: Multi-Layer Perceptron AND, NAND, OR & XOR representation

Table 1 provides: the truth table, network structure, weights, bias, and graphical state space representation for the perceptrons or MLPs required to perform the following operations most efficiently: AND, NAND, OR & XOR.

Table 1 – AND, NAND, OR & XOR Perceptron representation

<table>
<tr><th>Truth Table</th><th>Network Structure</th><th>Weights & Bias</th><th>Graphical State space</th></tr>
<tr><td colspan="4">AND Perceptron</td></tr>
<tr>
<td>
<table>
<tr><th>A</th><th>B</th><th>OUT</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
<td> 
</td>
<td>W1=1
W2=1
Bias= -1.5</td>
<td> 
</td>
</tr>
<tr><td colspan="4">NAND Perceptron</td></tr>
<tr>
<td>
<table>
<tr><th>A</th><th>B</th><th>OUT</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
<td> 
</td>
<td>W1=-1
W2=-1
Bias= 1.5</td>
<td> 
</td>
</tr>
<tr><td colspan="4">OR Perceptron</td></tr>
<tr>
<td>
<table>
<tr><th>A</th><th>B</th><th>OUT</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
<td> 
</td>
<td>W1=2
W2=2
Bias= -1</td>
<td> 
</td>
</tr>
<tr><td colspan="4">XOR Multi-Layer Perceptron</td></tr>
<tr>
<td>
<table>
<tr><th>A</th><th>B</th><th>OUT</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
<td> 
</td>
<td>Weights used in the three perceptrons required to produce the non-linearly separable XOR were the same as detailed above</td>
<td> 

Two lines needed to show decision boundary for non-linearly separable XOR state space</td>
</tr>
</table>

Exercise Two: MNIST Neural Network

a) **MNIST hand-written dataset:** network structure optimisation

The non-optimised network using parameters in table 2 was trained for hand-written MNIST data yielding performance ratings shown in table 3. These Performance ratings acted as a baseline to improve on.

Table 2 - Initial Neural Network Parameters

Parameter	Value
Input Nodes	784
Hidden Nodes	100
Output Nodes	10
Learning Rate	0.3

Table 3 – Performance of non-optimised neural network for hand-written MNIST classification

Training Set	Test Set	Performance
MNIST_TRAIN_100	MNIST_TEST_10	60.00%
MNIST_TRAIN	MNIST_TEST	94.40%

The network structure was then optimised to achieve the highest possible performance scores. The number of hidden nodes was changed to the mean of the combined number of input and output nodes, this increase in hidden neurons allowed more decision boundaries to be formed, without being so high that over-fitting or excessive training times become an issue. The learning rate was lowered to prevent large oscillations resulting in converging on a suboptimal solution. The number of training iterations was increased to compensate for the lower learning rate. The final parameters are given in table 4. Performance ratings of the optimised neural network showed an increase of 10% in the small data set and a 3.3% in the full dataset as shown in table 5.

Table 4 - Optimised Neural Network Parameters

Parameter	Value
Input Nodes	784
Hidden Nodes	397
Output Nodes	10
Learning Rate	0.1
Training Iterations	4

Table 5 – Performance of optimised neural network for hand-written MINIST classification

Training Set	Test Set	Performance
MNIST_TRAIN_100	MNIST_TEST_10	70.00%
MNIST_TRAIN	MNIST_TEST	97.70%

b) **MNIST fashion dataset:** network structure optimisation

This process was then repeated for the MNIST fashion dataset, firstly with the non-optimised parameters in table 2, yielding performance ratings shown in table 6.

Table 6 – Performance of non-optimised neural network for fashion MNIST classification

Training Set	Test Set	Performance
FASHION_MNIST_TRAIN_1000	FASHION_MNIST_TEST_10	80.00%
FASHION_MNIST_TRAIN	FASHION_MNIST_TEST	69.94%

The network structure was then optimised to achieve the highest possible performance scores. The number of hidden nodes changed to 2/3 of the combined number of input and output nodes, this is higher than in the optimised MNIST network, as the more complex fashion data requires an increased number of decision boundaries. The learning rate was lowered further than the MNIST dataset as the differences between the more complex datasets could easily be missed when training if the volatility were too high. The number of training iterations was increased to compensate for the lower learning rate. The final parameters are given in table 7. Performance ratings of the optimised neural network showed an increase of 20% in the small data set and a 16.81% in the full dataset as shown in table 8.

Table 7 - Optimised Neural Network Parameters

Parameter	Value
Input Nodes	784
Hidden Nodes	595
Output Nodes	10
Learning Rate	0.02
Training Iterations	6

Table 8 – Performance of optimised neural network for fashion MINIST classification

Training Set	Test Set	Performance
MNIST_TRAIN_100	MNIST_TEST_10	100.0%*
MNIST_TRAIN	MNIST_TEST	86.61%

*slight variation due to random start weights means 90% performance rating sometimes obtained

Differences in performance between hand-written and fashion datasets:

Both the MNIST and fashion MNIST datasets contain 28x28 greyscale images, totalling 784-dimensions, with 60,000 training images and 10,000 test images. However, the fashion MNIST dataset resulted in a lower performance than the original MNIST dataset by roughly 11%. This is due to the fashion MNIST data being more complex, meaning the 10 classes experience a larger overlap in the 784 dimensions than the simpler handwritten MNIST data. Larger overlap in dimensions means forming a decision boundary between classes is much harder with more outliers. Drastically increasing the number of hidden neurons to counteract this would lead to overfitting, or excessively long training times. As a result, a lower performance rating is achieved when using this simple two-layer Neural Network.

c) Proposal for improving overall score on both MNIST and fashion MNIST

- **Convolutional Neural Networks**

One of the limiting factors preventing our two-layer neural network from obtaining higher performance is the curse of dimensionality. The 784 dimensions and large number of hidden neurons required to achieve enough boundary resolution, result in a very large network with an impractical number of weights causing unacceptably long training times preventing better performance from being met. Convolutional neural networks work by convolving a filter window over the entire image to produce a feature map, this feature map is then passed through a subsampling or pooling layer to further reduce the number of dimensions and remove noise. Max pooling is often used where the maximum value in each pre-specified window size is taken, this decreases the size of the feature map whilst keeping significant information. After multiple layers of convolution and subsampling, the final dimensions are then passed into a fully connected MLP for classification. The constant reduction in dimensionality by the convolution and pooling layers also helps to control overfitting by limiting dimensions passed to the classifying MPL (overfitting still possible from too many filters). As the training stage in the convolutional neural network determines weights for the single convolutional filter window being passed over the entire input image, the total number of weights needing training is greatly reduced (reduction depending on size of filter window). Because of this, much more advanced feature recognition may be applied to the MNIST datasets without running into the same limiting curse of dimensionality and so greater classification performance can be achieved on both MNIST datasets.

- **Image Pre-Processing**

The MNIST datasets can be pre-processed with image enhancement techniques to improve the performance of the neural network's classification. The first technique is 'deskewing', this is where the image is straitened [1]. By straitening all the images such that images for each class are presented to the neural network in the same orientation, the same pixels will be used more frequently within individual classes (probability a pixel is use for that class of image won't be impacted by image rotation). The neural network can now more easily identify the key pixels to use for classifying each class and produce better decision boundaries. The second way in which the dataset could be pre-processed is by reducing noise in the images. Image enhancement uses blurring filters to reduce this noise, a weighted filter window where the sum of all weights is 1, and the weight of each pixel is 1/(number of pixels in windows), is convoluted with the original image. This results in a new image where noise has been reduced, but the DC level of the image is unchanged. When the neural network is trained on the smoothed MNIST dataset, it will be exposed to less noise and so better extract the important features for classifying the images, improving its performance.

- **PCA – Principal component analysis**

PCA creates a set of linearly uncorrelated variables called principal components, the number of principle components is less than or equal to the original meaning it can reduce the number of dimensions being passed into the neural network. The network structure can therefore be simplified, facilitating experimentation with lower learning rates and higher iteration numbers, without unacceptably long training times. The first principal component has the largest variance, so contains the most information, meaning many bottom principal components can also be discarded to further reduce the number of dimensions in the classification problem if needed.

References:

[1] Marek Gagolewski, "*Machine Learning in R from scratch*", [Online], 28/10/2012, Available: https://lmlcr.gagolewski.com/shallow-and-deep-neural-networks.html