Coursework A: Exercise One

The perceptron is a foundational element of neural networks, which can be used to solve basic logical operations. These perceptrons can stand alone as a single-layer or be combined as a multi-layer perceptron to solve more complex problems. To delve into their capabilities, perceptron based neural networks will be constructed for the AND, NAND, OR, and XOR operations, providing the necessary weights and biases for each. These networks will be represented through truth tables, network structure diagrams and graphical state spaces.

Truth Tables

AND		
X_1	X_2	Output
0	0	0
0	1	0
1	0	0
1	1	1

NAND		
X_1	X_2	Output
0	0	1
0	1	1
1	0	1
1	1	0

OR		
X_1	X_2	Output
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
X_1	X_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Figure 1. *Truth Tables for AND, NAND, OR and XOR*

$$Output = \sigma(w_1 x_1 + w_2 x_2 + b)$$

Equation 1. *Weighted Sum Equation for Perceptron*

Network Structure Diagrams

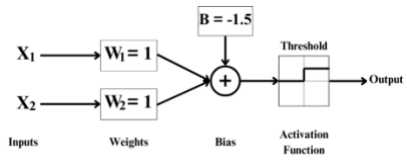

Figure 2. *AND Network Structure Diagram*

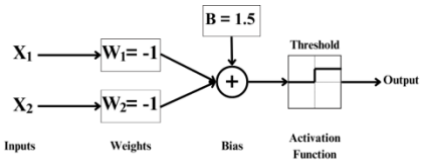

Figure 3. *NAND Network Structure Diagram*

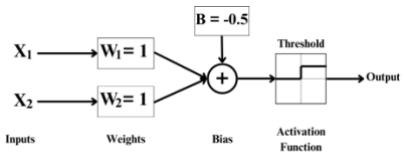

Figure 4. *OR Network Structure Diagram*

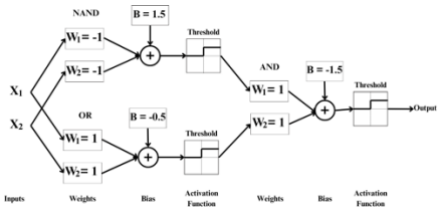

Figure 5. *XOR Network Structure Diagram*

Simple logic gates like AND, NAND, and OR can be implemented using a single-layer perceptron because their classifications are linearly separable. In contrast, XOR is a more complex function and requires a multi-layer perceptron since it cannot be linearly separated. The XOR MLP is built by using NAND and OR perceptrons, whose outputs are then fed into an AND perceptron to produce the final result. The weights and biases for each operation were calculated using Equation 1 (σ = step function).

Graphical State Space Representations

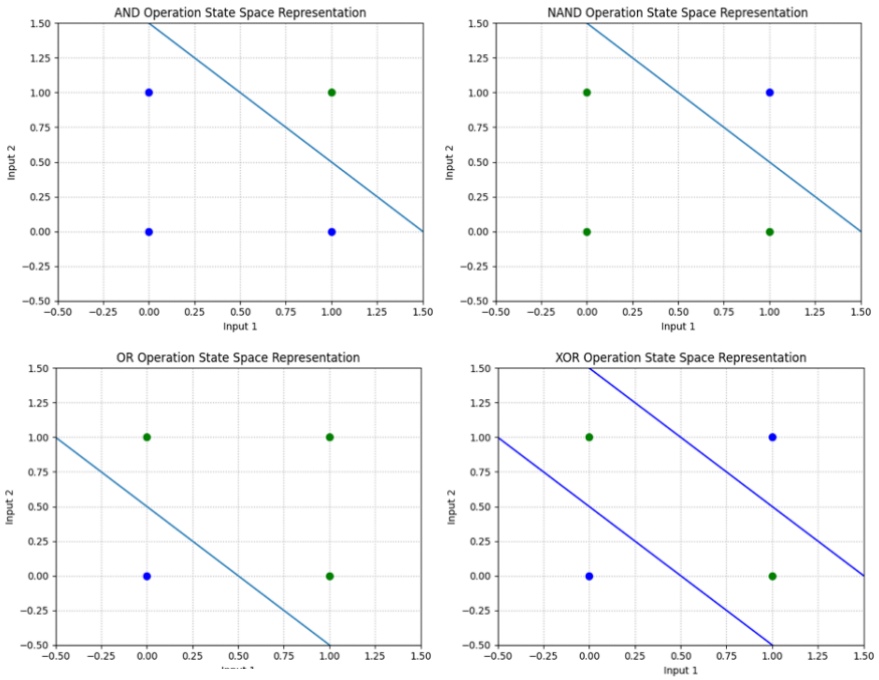

Figure 6. *State Space Representation of AND, NAND, OR and XOR Logic Gates*

The state space representations (Figure 6) show how the input combinations are classified by each network. In the simpler operations a single decision boundary clearly separates the input space into two regions, corresponding to the outputs (green – 1, blue – 0), allowing for a single-layer perceptron due it's linear nature. The XOR operation, however, requires two linear boundaries to classify the inputs, highlighting the non-linearity of the function. This demonstrates the need for a multi-layer perceptron in solving the XOR problem as a single perceptron cannot create this boundary. The XOR state space representation reveals the network's structure, showing that it combines the boundaries of both the NAND and OR gates.

Coursework A: Exercise Two

MNIST Hand-Written Character Model Evaluation

To determine the optimal model configuration, the learning rate was initially tested against varying numbers of hidden neurons to identify the most effective combination, The optimal learning rate was then selected, and the number of hidden nodes was evaluated in relation to the number of training iterations. The data shows lower learning rates (0.01-0.05) led to poor performance due to underfitting, and very high learning rates (0.03) caused model performance to plateau or decrease due to overshooting the weights and potentially causing the model to overfit. Hidden neurons were found to significantly impact performance, with more nodes the network can model more complex representations of the data. Table 1 shows that the best-performing configuration is a learning rate of 0.15 with hidden neurons between 200-1000 with a peak accuracy of 96.4%. This learning rate provides the best balance of weight adjustment and convergence rate.

When looking at training iterations the results shows moderate iterations (3–5) perform suitably, offering a good balance between training time and accuracy. However, very low iterations (1–3) consistently result in poor accuracy, as the model doesn't have enough time to sufficiently learn from the data, even with more hidden nodes. Overall, higher training iterations improved performance, especially when paired with a more complex network structure. The evaluation revealed that the optimal model configuration for classifying digits from the MNIST Handwritten data set, achieving an accuracy of **98.19%**, consists of a 784-1000-10 network architecture, a learning rate of 0.15, and trained over 10 iterations.

Learning Rate

Number of Hidden Neurons	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.15	0.2	0.3
10	0.7172	0.8309	0.8327	0.8804	0.8717	0.8739	0.8614	0.8643	0.863	0.8771	0.8507	0.8433	0.8491
30	0.9051	0.9119	0.9204	0.9148	0.9171	0.9188	0.923	0.9258	0.9292	0.9143	0.9195	0.9122	0.8947
50	0.9115	0.922	0.9276	0.9292	0.9276	0.933	0.9372	0.9265	0.9355	0.9383	0.9362	0.9383	0.9163
100	0.9202	0.9305	0.9361	0.9399	0.9419	0.946	0.9492	0.9469	0.948	0.9514	0.9493	0.9458	0.9423
200	0.926	0.9406	0.9429	0.9495	0.9504	0.9562	0.9554	0.9579	0.9596	0.9559	0.9622	0.9584	0.9537
300	0.9275	0.9395	0.9478	0.9522	0.9553	0.9583	0.9582	0.9614	0.9619	0.9619	0.9632	0.9632	0.9555
400	0.928	0.9437	0.9513	0.9543	0.9587	0.9595	0.9602	0.9604	0.9625	0.9613	0.9613	0.9586	0.9584
500	0.929	0.9454	0.9527	0.9575	0.9572	0.9597	0.9611	0.9602	0.9627	0.9599	0.9644	0.962	0.9566
600	0.9286	0.9445	0.951	0.9546	0.9585	0.958	0.9611	0.9613	0.9624	0.9606	0.9606	0.9627	0.954
700	0.9258	0.9447	0.9518	0.9567	0.9558	0.958	0.9595	0.9573	0.9603	0.962	0.9614	0.9618	0.956
800	0.9239	0.942	0.9491	0.9554	0.9592	0.9583	0.9613	0.9613	0.9614	0.9626	0.9629	0.9621	0.9573
900	0.9158	0.9421	0.9497	0.9551	0.9573	0.9587	0.9604	0.9624	0.9592	0.9614	0.9636	0.9629	0.9564
1000	0.9166	0.9427	0.9519	0.9565	0.9571	0.9597	0.9604	0.9611	0.962	0.9613	0.9644	0.9612	0.95

Table 1. *Accuracy for MNIST Dataset, Learning Rate vs Number of Hidden Neurons (Epochs = 1)*

Epochs

Number of Hidden Neurons	1	3	5	8	10
200	0.9622	0.9719	0.9772	0.9769	0.9797
300	0.9632	0.974	0.9772	0.9793	0.9805
400	0.9613	0.9752	0.979	0.9805	0.981
500	0.9644	0.9734	0.9775	0.9816	0.9805
600	0.9606	0.976	0.9795	0.9801	0.9803
700	0.9614	0.9759	0.9796	0.9808	0.9818
800	0.9629	0.9754	0.9781	0.9801	0.9791
900	0.9636	0.9763	0.9796	0.9796	0.9801
1000	0.9644	0.9747	0.9787	0.9799	0.9819

Table 2. *Accuracy for MNIST Dataset, Epochs vs Number of Hidden Neurons (Learning Rate = 1.5)*

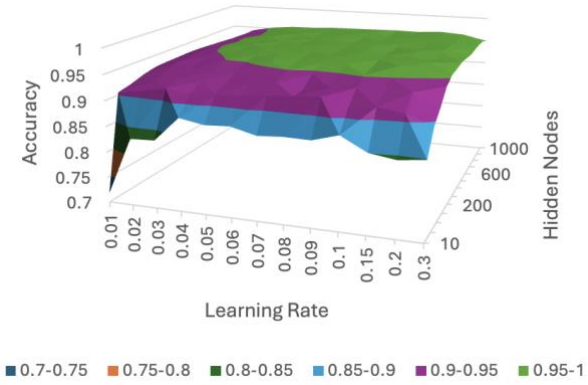

Figure 7. *3D Surface Plot for Table 1 Results*

MNIST Fashion Model Evaluation

Compared to the handwritten dataset, the accuracy results for the Fashion MNIST dataset are noticeably more erratic and less predictable. While the random initialisation of weights could contribute to this, this was present in the model for the handwritten data. As seen in Table 3, higher learning rates (0.09-0.3) result in a decline in accuracy and increased inconsistency, again suggesting that at these rates the model may be adjusting the weights too aggressively. A learning rate of 0.08 was found to produce the best accuracy, with a maximum score of 80.89%. Table 4 shows that, unlike the previous network, accuracy declined significantly at 10 epochs, with 3-8 epochs producing better results, suggesting possible overfitting at 10 epochs. The optimal configuration of epochs and hidden nodes at a learning rate of 0.08 achieved a highest accuracy of **83.51%**. Therefore, the ideal network structure and parameters were identified as a 784-1000-10 node architecture, a learning rate of 0.08, and training over 3 epochs.

Number of Hidden Neurons \ Learning Rate	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.15	0.2	0.3
10	0.6928	0.6874	0.6892	0.7505	0.712	0.7002	0.707	0.6112	0.7197	0.7572	0.7607	0.7082	0.7074
30	0.7983	0.7518	0.7726	0.7726	0.789	0.7948	0.8075	0.7923	0.7469	0.776	0.7786	0.7879	0.7629
50	0.7864	0.7627	0.7771	0.7849	0.791	0.7942	0.7195	0.7961	0.7975	0.7916	0.8005	0.7896	0.7452
100	0.7816	0.805	0.7559	0.7602	0.808	0.787	0.7441	0.7715	0.7409	0.771	0.7959	0.7686	0.7926
200	0.789	0.766	0.788	0.7759	0.791	0.7818	0.7967	0.7869	0.7435	0.744	0.7722	0.7905	0.7819
300	0.7699	0.7931	0.7756	0.7334	0.762	0.7787	0.7719	0.7714	0.7667	0.7934	0.77	0.7682	0.7357
400	0.7542	0.7736	0.7521	0.7867	0.77	0.778	0.7613	0.7865	0.8062	0.7941	0.7979	0.776	0.7847
500	0.736	0.7812	0.7983	0.7869	0.774	0.7825	0.7703	0.7681	0.8	0.7821	0.7949	0.7521	0.7783
600	0.7914	0.7633	0.7697	0.7864	0.746	0.7486	0.7507	0.8052	0.7703	0.7733	0.7616	0.7521	0.7153
700	0.7855	0.7853	0.7573	0.8038	0.785	0.7747	0.7326	0.7829	0.7893	0.7635	0.7953	0.7885	0.7515
800	0.7746	0.8042	0.728	0.7348	0.774	0.7643	0.794	0.7809	0.7814	0.7897	0.7763	0.729	0.7911
900	0.7933	0.7716	0.7716	0.7379	0.77	0.7267	0.7883	0.7974	0.7808	0.7796	0.757	0.802	0.7943
1000	0.7408	0.7987	0.7676	0.7654	0.736	0.761	0.7513	0.8089	0.7256	0.8045	0.7866	0.788	0.7662

Table 3. *Accuracy for Fashion MNIST Learning Rate vs Number of Hidden Neurons (Epochs = 1)*

Number of Hidden Neurons \ Epochs	1	3	5	8	10
200	0.7435	0.8255	0.8306	0.8093	0.821
300	0.7667	0.831	0.8316	0.8215	0.823
400	0.8062	0.829	0.8136	0.82	0.826
500	0.8	0.8241	0.8256	0.8266	0.82
600	0.7703	0.818	0.7761	0.8159	0.823
700	0.7893	0.8337	0.8233	0.8167	0.809
800	0.7814	0.8193	0.7908	0.7857	0.815
900	0.7808	0.8125	0.8109	0.8318	0.774
1000	0.7256	0.8351	0.8261	0.8343	0.81

Table 4. *Accuracy for Network Epochs vs Number of Hidden Neurons (Learning Rate = 0.8)*

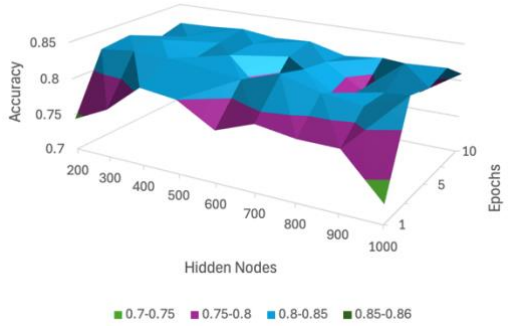

Figure 8. *3D Surface Plot for Table 4 Results*

The MLP network achieved much higher accuracy on the handwritten digit's dataset than on the fashion dataset, largely due to the increased complexity of the fashion images. Clothing items express greater variability in shape compared to digits, making it harder for the network to learn consistent patterns and generalise across classes. Additionally, certain clothing items, such as a coat vs. a shirt or a sneaker vs. an ankle boot, have more subtle differences than digits like 2 vs 9, which are more distinct. This added complexity results in less regularity in the fashion data, whereas handwritten digits follow a more predictable pattern. These findings suggest that while a feed-forward MLP is effective for classifying digits, a more sophisticated model is needed to capture the intricate patterns in the fashion dataset. The optimal solutions for both datasets used 1000 hidden neurons, but the fashion dataset reached its best performance with just 3 epochs, compared to 10 epochs for the handwritten dataset, which is surprising as the fashion data is more complex. Training time for 1000 hidden nodes over 10 epochs averaged **20 minutes**, while only around **5 minutes** were needed for 3 epochs. This shorter training time is beneficial, as it reduces computational cost and time without sacrificing accuracy, making the model more efficient and practical for larger datasets or real-time applications.

Future Improvements: An effective way to improve accuracy on both datasets is by implementing a Convolutional Neural Network (CNN). CNNs are specifically designed for image data, as they create feature maps and detect spatial hierarchies like edges, textures, and shapes. These networks typically consist of multiple convolutional layers for feature extraction, pooling layers to reduce spatial dimensions while preserving key features, and fully connected (dense) layers to integrate the learned representations. CNNs are better at extracting important features compared to MLPs, making them a more suitable solution for improving accuracy in both datasets. For example, a study [1] using an ensemble of three CNNs with varying kernel sizes and data augmentation achieved an error rate of **0.09%** on the handwritten dataset, significantly outperforming the model in this report, which had an error rate of **1.81%**.

Principal Component Analysis (PCA) is a technique to reduce dimensions that transforms data by identifying the principal components, or directions, that capture the most variance in the dataset [2]. This process reduces the number of input features while preserving the most important information. Applying PCA before training the neural network can help eliminate noise and redundant information in the pixel data, allowing the model to focus on key features. The fashion dataset, in particular, may benefit from PCA, as it contains more complex and detailed images. By reducing dimensionality, PCA could help the model learn more generalised features, resulting in better performance.

The evaluation of both models showed that high learning rates cause the network to overshoot the optimal weights, leading to a drop in accuracy, while low learning rates do not adjust the model enough within the given training time. A potential solution is to use adaptive optimisers such as the Adam optimiser, which adjusts the learning rate throughout training [3]. By starting with a higher learning rate and gradually decreasing it as training progresses, this approach can prevent overshooting while ensuring the model makes sufficient changes to the weights early on.

Bibliography

[1] S. An, “An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition,” *arXiv,* 2020.

[2] H. Abdi, “Principal Component Analysis,” *WIREs Computational Statistics,* vol. 2, no. 4, pp. 433-459, 2010.

[3] S. Y. SEN, “Convolutional Neural Network Hyperparameter Tuning with Adam Optimizer for ECG Classification,” *2020 Innovations in Intelligent Systems and Applications Conference (ASYU),* pp. 1-6, 2020.