M1 Machine Learning - Coursework Report

Jacob Tutt (JLT67)

Department of Physics, University of Cambridge

December 18, 2024

Word Count: 2996

1 Introduction

The motivation of this report is to implementation and present the accuracy of various machine learning approaches. Specifically in the classification of a dataset generated from MNIST images, where two digits are vertically stacked (56x28) and labeled by the sum of the constituent digits. The study first evaluates the performance of a fully connected neural network before comparing it to the alternative methods, offering insights into the reasoning for their relative strengths and weaknesses in handling the highly-dimensional dataset.

2 Question 1 - Generation of Data

This section justifies the methods used to generate the (56x28) MNIST dataset used throughout this report, from Tensorflow's original (28x28) images.

2.1 Step 1 - Preprocessing the MNIST Dataset

The MNIST dataset provided is pre-split into training and test sets, and although well documented to be randomly distributed, this project manually shuffles them as good practice. This ensures statistical consistency across the data sets and avoids bias during training. The provided training data was then further split (0.8 : 0.2) into training and validation sets, ensuring all data sets remain completely independent throughout. This allows the validation data to accurately represent the model's performance on unseen data and identify overfitting for hyperparameter tuning. Finally, the data is normalised to the range $[0, 1]$ to maximise the computational efficiency and stability

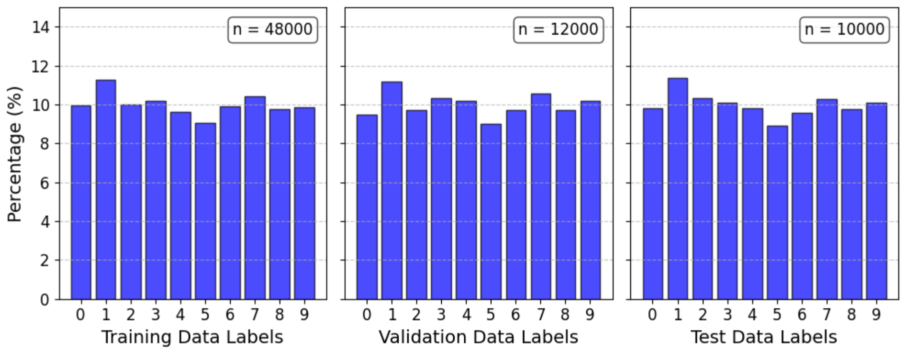


Figure 1: MNIST dataset distribution across the training, validation and test sets.

of the future networks as it ensures all images are equally weighted and avoids the saturation of activation functions. The statistical consistency of the base data sets is verified in Figure 1, showing an almost uniform distribution across all sets. Notably, MNIST does contain slight fluctuations across the digits' distribution.

2.2 Step 2 - Motivation of Classification

Firstly it is important to discuss the overall motivation of the classification, as the distribution of the generated data should mirror that of the future input. This is imperative as the model will inherently minimise the overall loss function and hence bias accuracy towards the most common labels. This report interprets the motivation to classify either the sum of two uniformly distributed single digits or analogously the sum of digits in uniformly distributed two-digit numbers. Both of these result in a triangular distribution for labels from 0 to 18, defined by the probability mass function:

$$P(S=k)=\begin{cases}\frac{k+1}{100}, & \text{if } 0\leq k\leq 9,\\ \frac{19-k}{100}, & \text{if } 10\leq k\leq 18,\end{cases} \tag{1}$$

where:

- k, sum's label.

This distribution makes intuitive sense as it provides an equal number of samples for each unique combination of digits, hence training on each equally. Alternatively the classification could be trained and applied to a uniform distribution across all labels, however, this was determined to have far less practical applications and hence not pursued.

2.3 Step 3 - Generating the (56x28) Dataset

This triangular distribution is implemented using sampling with replacement to select two random digits from the training set: vertically stacking them to form the desired (56x28) image and summing their labels. This was performed independently for the training, validation and test sets from their respective single-digit data sets, ensuring no data leakage. Sampling with replacement is justified due to the comparative sizes of the generated data set and the number of possible unique permutations for double digit numbers, shown in Table 1. The typical ratio between them is never greater than 10^4, hence the probability of repeat samples is negligible. For training sets requiring a comparable number of data points to the unique permutations, a more complex sampling method would be required.

Dataset	**Size(single-digit)**	**Possible Unique Images (double-digit)**	**Samples Generated**
Training	48,000	2,303,928,000	100,000
Validation	12,000	132,000,000	15,000
Test	10,000	90,000,000	15,000

Table 1: Comparison of dataset sizes and unique permutations.

The resultant distributions are plotted to verify their consistency and overall triangular shape, with very slight statistical fluctuations as expected. Notably, this project did not enforce an exact distribution across label classes, as such precision was not deemed sufficiently advantageous to counter its computational cost.

3 Question 2 - Development of Neural Network

This section provides a review of the development of a fully connected neural network to implement the classification, and a review of its performance.

3.1 Step 1 - Network Architecture

The architecture of the sequential model was designed using a combination of informed decisions and hyperparameter tuning, aiming to balance accuracy and computational efficiency. The model was optimised using an Optuna study over 35 trials for the 5 hyperparameters summarised in Table 2, which had a typical runtime of 176 minutes. The models were compared through their validation accuracy after 7 epochs. The justification for the structural design is provided below:

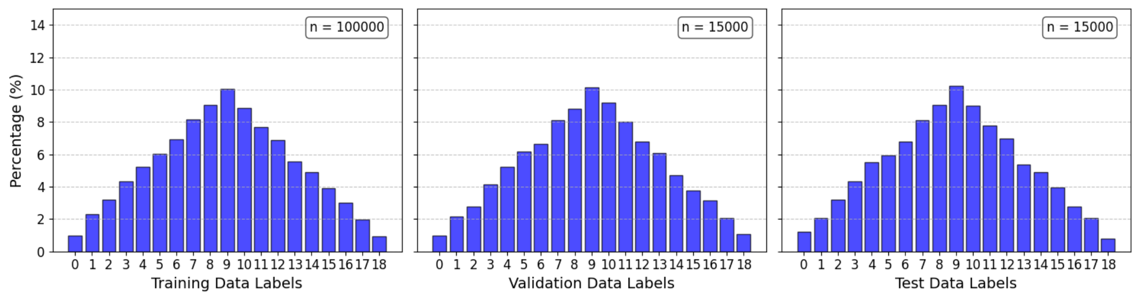


Figure 2: Distribution of the generated double-digit data across all sets.

3.1.1 Network Constants

The network required an input layer of size 1568 to accept the flattened data and an output layer of size 19, corresponding to each label. For the output layer the softmax activation function was used, allowing it to map the outputs to a probability distribution across the labels, which is common for multi-class classification. As a standard for such tasks, the categorical cross-entropy loss function was implemented to quantify the softmax output's accuracy, and the Adam optimiser was used due to its fast and robust convergence from its use of momentum and an adaptive learning rate. The initial learning rate of which was tuned as a hyperparameter.

The classification task's requirement for significant dimensionality reduction lent itself to a funnel-style hidden layer structure, in which the number of neurons decreases sequentially. This was implemented with 3 hidden layers, as previous hyperparameter tuning showed deeper networks did not offer any significant improvements but increased the computational cost significantly. The number of neurons in each layer was determined by tuning the number of neurons in the first layer and the reduction factor between layers. Additionally, a batch size of 64 was chosen to balance training time and stability based on previous hyperparameter tuning. Finally, batch normalisation was performed between each hidden layer to reduce the internal covariate shift [4], and improve the overall stability and training time.

3.1.2 Network Hyperparameters

Table 2 shows the hyperparameter space that the Optuna study was optimised over, from those defining the funnel-like structure described in Section 3.1.1, to the learning rate of the Adam optimiser and the activation function. Notably, this report has not implemented either L1 or L2 regularisation. This is attributed to two factors, firstly the dropout method is exploited to reduce the dependency on any given neurons and acts as its own form of regularisation. Additionally, previous hyperparameter studies showed that, after batch normalisation and dropout rate, L1 and L2's inclusion caused over-regularisation, thus harming performance.

Hyperparameter	**Range/Values**	**Description**
Starting Units	600 to 1000 (log scale)	Number of neurons in first hidden layer
Reduction Factor	0.65 to 0.85	Rate of neuron reduction in successive layers
Learning Rate	10^{-3} to 6×10^{-3} (log scale)	Initial learning rate for Adam optimiser
Activation Function	relu, tanh, sigmoid, swish	Hidden layer's activation function
Dropout Rate	0.25 to 0.5	Fraction of neurons ignored for regularisation

Table 2: Hyperparameters optimised for in Optuna study

3.2 Results of Optuna Study

Figure 3 shows the hyperparameter values for the 6 best trials, evaluated based on the validation accuracy. The study clearly convergences to a narrow range for all hyperparameters, and in real-world applications, further refined hyperparameter searches could be performed for further optimisation. Additionally, none of the parameters approach their bounded limits, showing that an appropriate hyperparameter space was defined.

Of particular interest is the apparent success of the ReLU activation function. Its constant gradient and lack of saturation are often credited for its advantages, unlike Sigmoid and Tanh, which often face vanishing gradients. However, this is less compelling given the network's use

Trial	Value	Dropout Rate	Starting Units	Learning Rate	Reduction Factor	Activation
26	0.9507	0.254	877	0.003	0.709	ReLU
24	0.9478	0.251	853	0.003	0.654	ReLU
19	0.9474	0.259	705	0.002	0.729	ReLU
31	0.9473	0.250	806	0.003	0.699	ReLU
20	0.9468	0.252	838	0.002	0.701	ReLU
33	0.9463	0.294	812	0.003	0.725	ReLU

Table 3: Hyperparameter values for top 6 Optuna trials

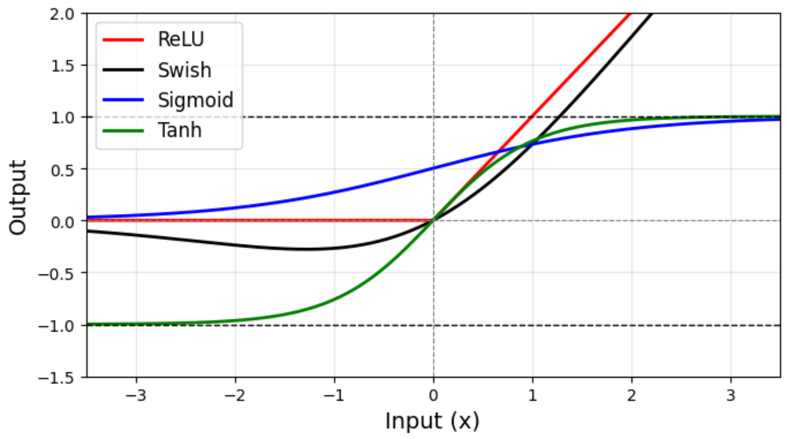


Figure 3: Activation functions used in Optuna study

of input and batch normalisation. Furthermore, this would fail to explain its performance over swish ‘a smooth, non-monotonic [6] generalisation of ReLU. Instead, the consistent performance is likely a result of its truncation of negative values, shown in Figure 3. This feature is particularly beneficial due to the use of the MNIST dataset, where a significant portion of the data consists of zero information (dark pixels) and hence promotes sparsity of the activation functions.

3.3 Best Preforming Network

The best-performing network, summarised in Table 4, was re-trained for up to 20 epochs with early stopping preventing overfitting. The training history in Figure 4 shows the neural network is able to train rapidly, achieving an accuracy of 94% within three epochs. Beyond this, the validation accuracy and loss level off, reaching a maximum accuracy of 94.1%. As the training metrics surpass the validation's, the model begins overfitting and thus the training terminates after just 7 epochs. The network's structure and weights are saved within the project's repository and are reloaded for reproducibility.

Layer Name	No. of Neurons	Activation	Additional Features
Input (Flatten)	1568	None	N/A
Hidden Layer 1 (Dense)	877	ReLU	Batch Normalization, Dropout
Hidden Layer 2 (Dense)	622	ReLU	Batch Normalization, Dropout
Hidden Layer 3 (Dense)	441	ReLU	Batch Normalization, Dropout
Output Layer (Dense)	19	Softmax	N/A

Table 4: Architecture of best neural network

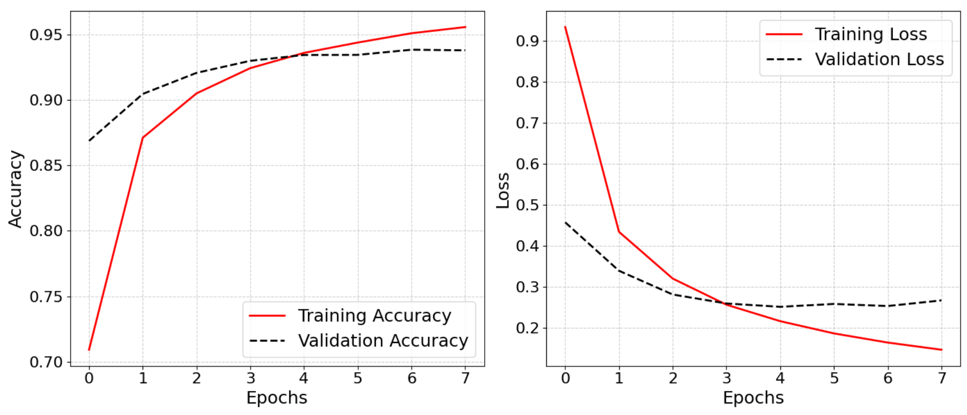


Figure 4: History of Training and Validation Accuracy (left) and Loss (right)

Finally, the performance of the network is evaluated for each class label individually. The accuracy for all classes is seen to fall within the range of 91-99%, showing consistently successful classification, in comparison to random guessing which yields a result of 5.3 %. This consistency across all classes, despite the changeable number of combinations, can be attributed to each unique two-digit pairing being trained equally. Although the inner workings of a neural network are incredibly complex, with this one containing 2,209,150 trainable parameters, this report attempts to justify the variability in success rate. The relative accuracy of each class can be predicted by how easily identifiable its possible constituent pairings are from others. For example, the 98.3 % accuracy of the class labeled 0 may be a result of the only constituent digits being (0,0) which produces a relatively unique shape. Similarly, for the class labeled 1 the only possible combinations are (0,1) and (1,0) and achieves an accuracy of 98.1 %.

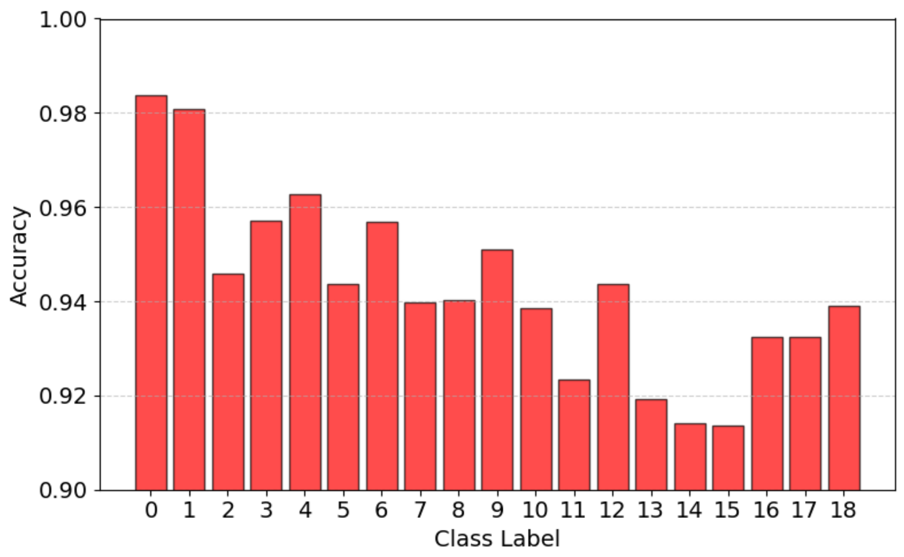
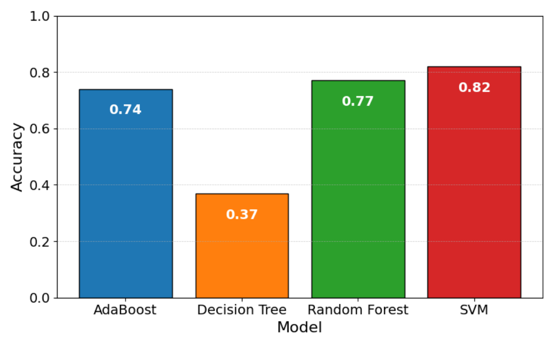


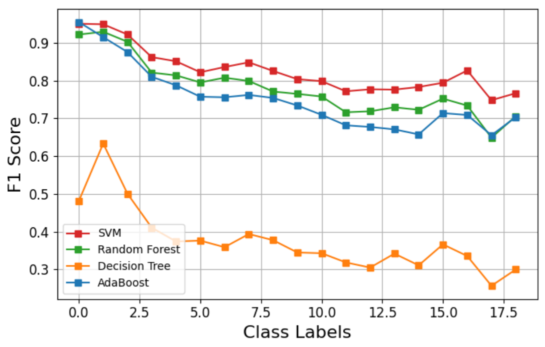
Figure 5: Tests accuracy for each class

4 Question 3 - Alternative Machine Learning Algorithms

This report goes on to investigate other machine learning approaches, specifically a Multi-Class Support Vector Machine (SVM), Adaptive Boosting (AdaBoost), Random Forest and a singular Decision Tree. For each of these, hyperparameter tuning is performed by a short Optuna study, allowing comparability with previous results. However, it is also important to acknowledge the reduced size of the datasets relative to Section 3, with 10,000 generated samples being used during tuning and 50,000 for the subsequent training of the best trials. The reduced volume of data is due to the computational complexity of the SVM classification for N training samples, $O(N^2)$ to $O(N^3)$ [1]. For consistency, all methods use the same sample size as that of the limiting method, SVM, and their performances are presented in Figure 6.



(a) Overall accuracy



(b) F1 Scores over each class size

Figure 6: Comparison of alternative machine learning algorithms

Figure 6a shows the greatest accuracy is achieved by the SVM classifier, whereas a singular decision tree unsurprisingly shows consistently low performance. When considering computational complexity, although SVM offers a 5% accuracy increase over the random forest, this is coupled with $\approx$ 20 times greater training time over the 50,000 samples. The F1 scores of each class are shown for all approaches, a metric that encapsulates both the true positive rate (recall) and the precision of the predictions. The overall trend of the F1 scores is shown to be consistent for all classifiers with higher performance for labels 0 and 1 which slowly decreases until a slight rise at 15. Not only are they consistent with each other but also Figure 5. Although a similar trend is seen, all methods are considerably less accurate than the fully connected neural network previously discussed. A brief overview of each method and its tuned hyperparameters are given below, and confusion matrices for each can be found in the appendices (7).

4.0.1 Support Vector Machine

The SVM in this report uses a 'one vs one' [3] approach with the non-linear kernel, RBF allowing the transformation of input data into higher dimensional space. This approach uses an assortment of binary classifiers to handle the multiple classes. The RBF's kernel coefficient, γ which dictates the influence of each sample, and the SVM's regularisation parameter, C, both underwent tuning [8]. The optimal parameters for γ and C were found to be 0.0128364 and 4.39114 respectively.

4.0.2 Adaptive Boosting

The Adaboost method simply uses an ensemble of weak classifiers to try and classify the more complex MNIST data. The hyperparameters tuned over were the $n_{estimators}$ and the learning rate. The Optuna study found the optimal learning rate to be 0.0255 but its study for the $n_{estimators}$ hit the maximum boundary (300). This is expected due to MNIST data's high variability and complexity.

4.0.3 Decision Tree and Random Forest

The decision tree recursively splits data based on features that minimise the entropy/ Gini index at each node. The random forest classifier builds on this by using an ensemble of decision trees, improving generalisation and overfitting. For both, the hyperparameters tuned over - the max depth, minimium number of splits, minimum samples per leaf, and the number of estimators for the random forest all hit their widely defined boundaries. This implies they struggle to identify high-level features in the dataset.

5 Question 4 - Weak Linear Classifiers

The report moves on to investigate the accuracy that can be achieved using a single-layer logistic regression model when classifying the double-digit images. This is performed twice, firstly classifying the 56x28 images with labels 0-18 directly and secondly independently classifying the 28x28 images with labels 0-9 and convolving their results to form a distribution for their sum. This section predominantly uses the datasets generated in Section 2 for comparability. For the same reasoning as before, both logistic regression models use an Adam optimiser, a softmax activation function, and a categorical cross-entropy loss function. Additionally, both train on the respective data sets for 15 epochs with early stopping enabled. Finally, both models are consistently evaluated for the same 15,000 test images with the only difference being the sequential model is by definition fed the top and bottom images individually.

5.1 Single Classifier

Then single classifier is originally evaluated using the full training set of 100,000 data points (56x28) generated in Section 2. Notably, the model implements early stopping after just 8 epochs with a test score of just 23.54%, suggesting that unsurprisingly the weak linear model is unable to classify the highly dimensional data. Figure 7, which visualises the precision and recall (true positive rate), shows despite this poor overall accuracy, there is much greater variability across the layers with the model still performing relatively very well for labels such as 0, 1 and 18.

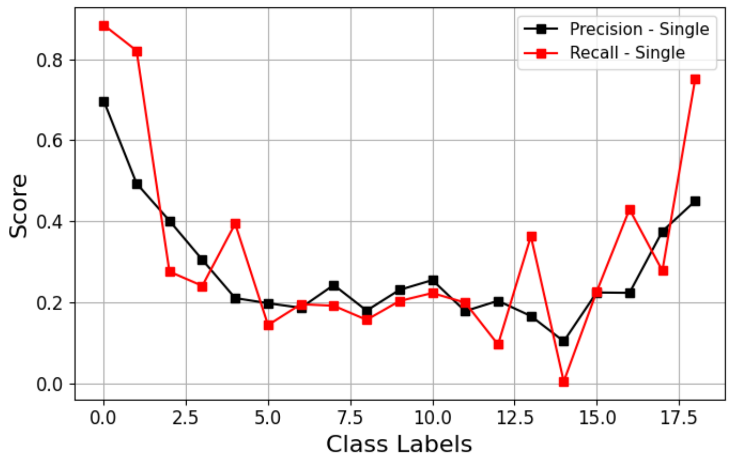


Figure 7: Recall and precision scores a weak linear classifier on double-digit data

5.2 Sequential Classifier

Before presenting the results of the sequential classifier, the training data and the methods used to convolve the probabilities are discussed. Firstly, this report trains the sequential classifier using the original MNIST (single-digit) training set of 48,000 images, ensuring all data points only appear once per epoch. Alternatively, the model could have been trained on the top and bottom images of the generated data, totaling 200,000 data points. However, as there are only 48,000 unique training digits this would result in bias towards the duplicated data points. Secondly, the convolved probabilities from the two individual classifications are calculated using:

$$P_{\text{sum}}[k] = \sum_{i=0}^{k} P_{top}[i] \cdot P_{bottom}[k-i], \tag{2}$$

When trained on the full training set, the model is able to perform with an overall accuracy of 85.69%. This significant increase can be attributed to the greatly reduced input dimensionality and classification options. Overall the task this weak linear classifier is required to perform is far less complex, thus resulting in far greater consistency across all the classes, seen in Figure 8.

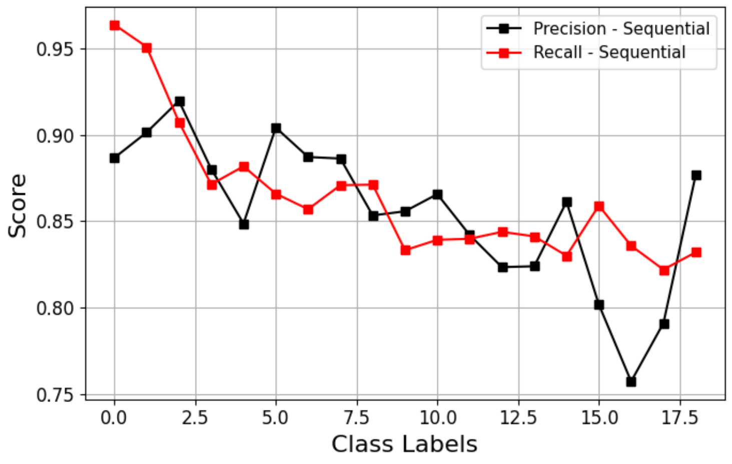


Figure 8: Recall and precision scores for a weak linear classifier applied sequentially on single digits

5.3 Performance with Training Size

The performance of both approaches, when evaluated on datasets of varying size — 50, 100, 500, 1000, 5000, and 10,000 — is illustrated in Figure 9. Both are seen to asymptotically approach their maximum achievable accuracy with diminishing improvements observed beyond a sample size of 1000. However the key distinction is the consistent outperformance of the sequential approach, showing its ability to capture the complex trends in greater detail. This is a result of it transforming a lower-dimensional input into fewer classes, a much simpler classification as previously discussed.

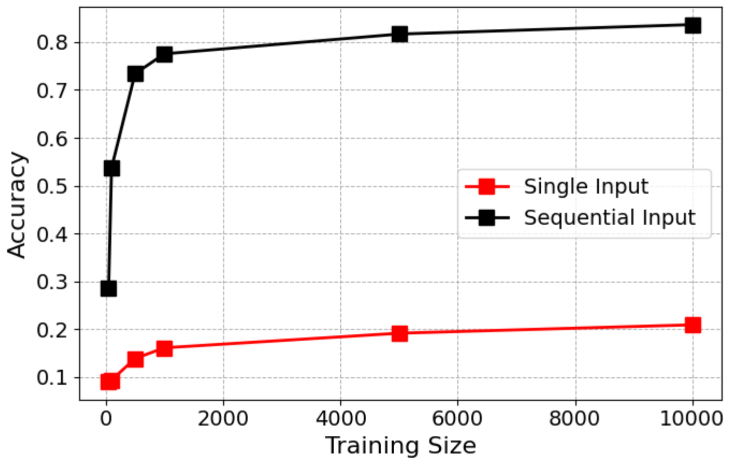


Figure 9: Accuracy of both weak linear classifiers with variable training sizes

6 Question 5 - t-SNE of Input and Embedding Layer

The final section of this report aims to visualise the inner workings of the fully connected neural network developed in Section 3 and demonstrate its ability to transform the highly variable and complex MNIST dataset into clear and well-separated groupings. A comparison is made between the spatial and temporal distribution of the data in both the input and embedding layers through cluster analysis [5]. As both layers are highly dimensional, t-distributed Stochastic Neighbour Embedding (t-SNE), is employed to visualise these clusters within two-dimensional space through nonlinear dimensionality reduction [2].

Firstly to produce the best visualisation of the multidimensional clusters, its associated hyperparameter, perlexity is optimised. Perplexity aims to balance the emphasis of global and local relationships in an attempt to display meaningful clusters in 2D and its typical values are in the range of 5-50 [2]. It is often considered to show ‘randomness’ with slightly different values producing greatly varying results as well as being highly unique for each dataset [9].

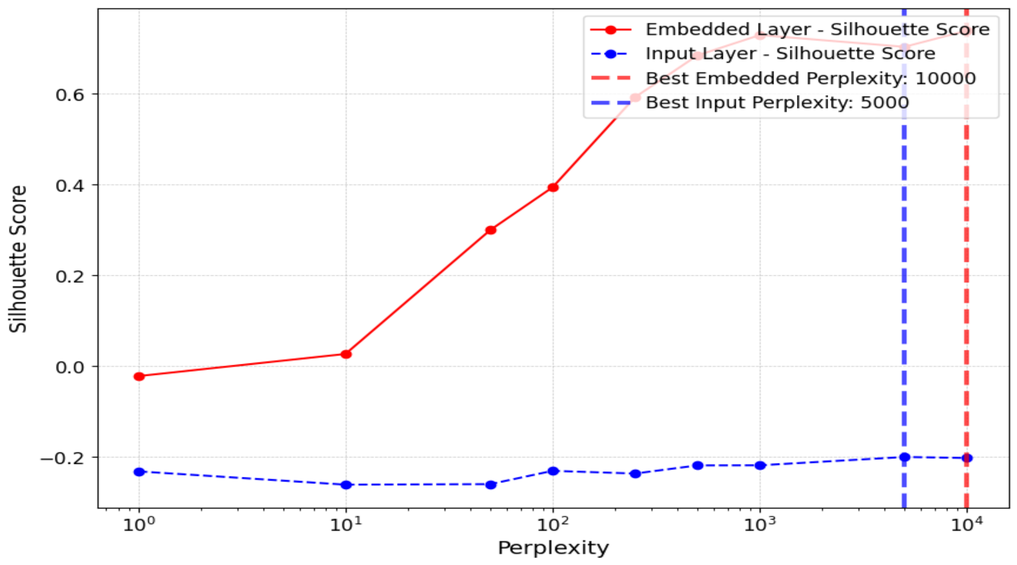


Figure 10: Variation in silhouette score with perplexity

This report first attempts to exploit the silhouette score as an optimisation metric, which offers a quantitative value for the quality of clusters [7] . The results of this investigation are seen in Figure 10, and although the embedding layer appears to have reached a plateau with maximum value ≈ 0.75, it corresponds to a perplexity 10,000 far outside the expected range. The failure of the silhouette score to produce valid results is a result of it taking Euclidean distance as a factor in its evaluation whereas t-SNE aims to preserve local relationships rather than global separation. Consequently, the silhouette score has incorrectly prioritised a t-SNE with high separation over cohesion as shown by the poor visual performance in Figure 11.

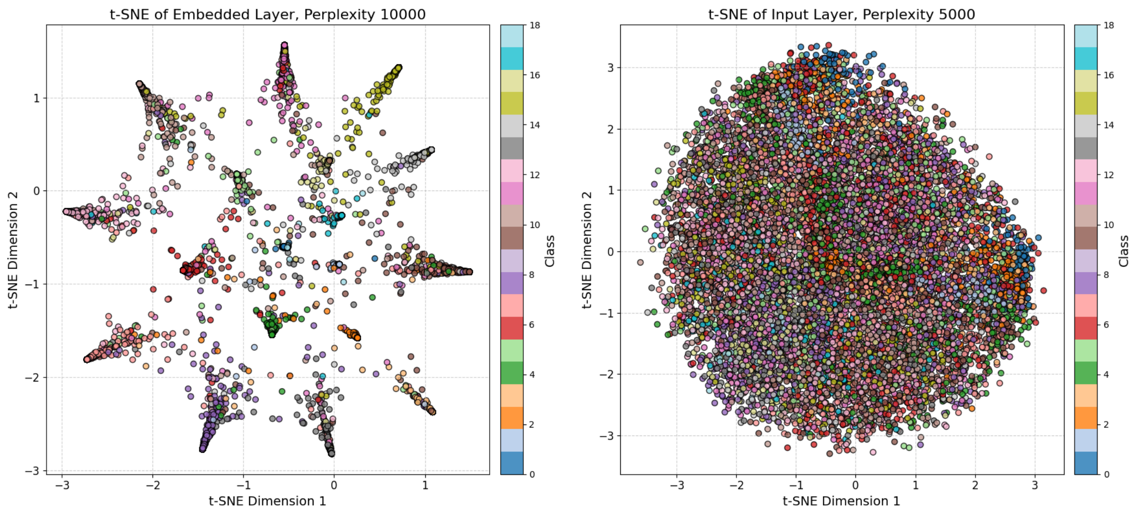


Figure 11: Resultant t-SNE distribution of embedding layer (perplexity, 10000) and input layer (perplexity, 5000) with best silhouette score (non-optimal)

As a result, the report resorts to optimising perplexity visually over the traditional range of 5-50. It finds no perplexity value offers any meaningful structuring for the input layer, a result of MNIST's initial high variability and complexity. This further justifies the inaccuracy of a weak linear classifier in Section 5.1. However, t-SNE is able to produce distinct clusters from the embedding layer, showing the neural network is successful in transforming the highly dimensional data to a clear structure within feature space, as suggested by its 94% accuracy. The plot for the embedding layer at its 'optimal' perplexity of 30 shows the surface area of clusters can be seen to be correlated to the number of unique combinations that result in that label. For example, the labels 0 and 18 both have 1 unique permutation and hence are contained within very small regions with the plot. Alternatively, the label 9, which has 10 unique permutations is approximately the largest.

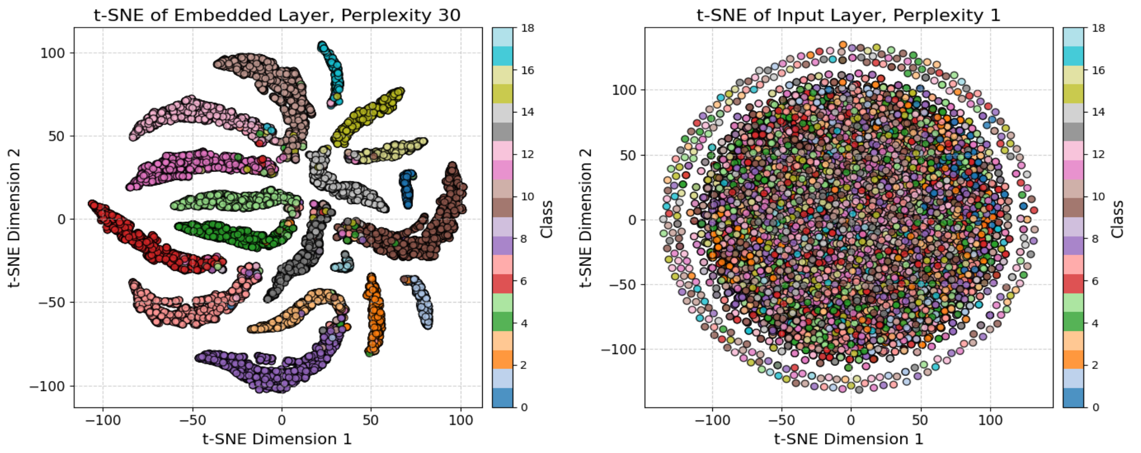
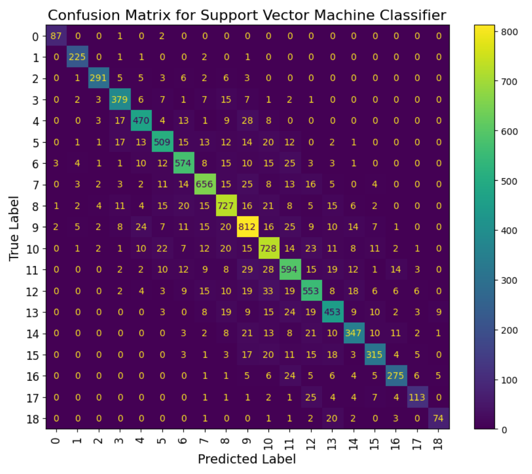


Figure 12: Visually optimised t-SNE distribution of embedding layer (perplexity, 30) and input layer (perplexity, 1)

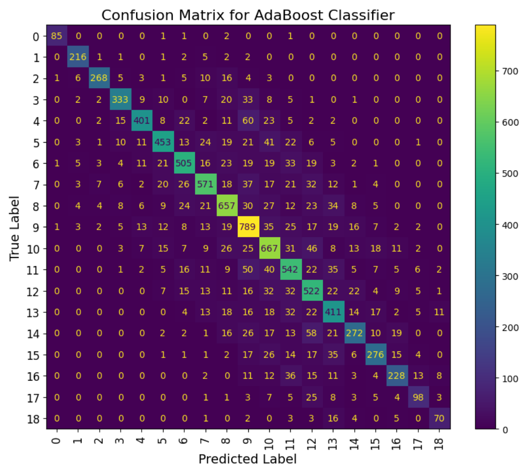
7 Appendicies

Most networks trained during this report have been saved within the repository's Results directory for reproducibility and transparency of results to others.

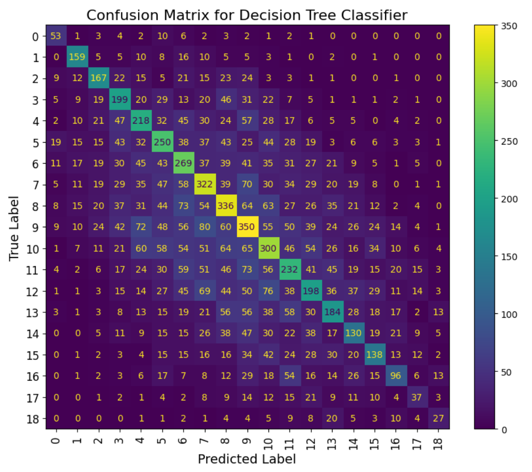
7.1 Confusion Matricies



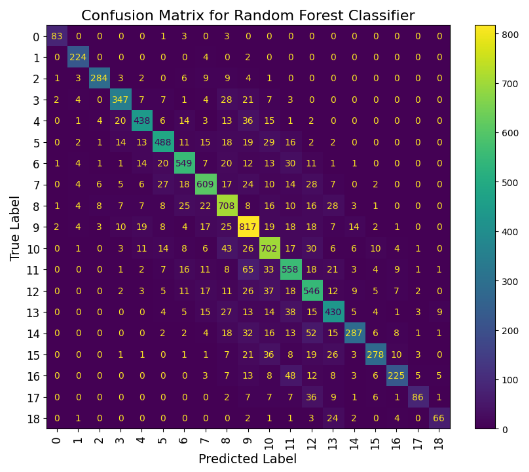
(a) SVM Classifier



(b) Adaboost Classifier



(c) Decision Tree Classifier



(d) Random Forest

Figure 13: Confusion Matrix Section 4 Classifier evaluated on test data

7.2 Declaration of Use of Autogeneration Tools

This report made use of Large Language Models (LLMs), primarily ChatGPT and Co-Pilot, to assist in the development of the project. These tools have been employed for:

- Formatting plots to enhance presentation quality.
- Performing iterative changes to already defined code.
- Debugging code and identifying issues in implementation.
- Helping with Latex formatting for the report.
- Identifying grammar and punctuation inconsistencies within the report.

References

[1] Retantyo Wardoyo Abdiansah Abdiansah. "Time Complexity Analysis of Support Vector Machines (SVM) in LibSVM". In: *International Journal of Computer Applications* 128.3 (Oct. 2015), pp. 28–34. ISSN: 0975-8887. DOI: 10.5120/ijca2015906480. URL: https://ijcaonline.org/archives/volume128/number3/22854-2015906480/.

[2] Yanshuai Cao and Luyu Wang. "Automatic Selection of t-SNE Perplexity". In: *CoRR* abs/1708.03229 (2017). arXiv: 1708.03229. URL: http://arxiv.org/abs/1708.03229.

[3] Phoenix X. Huang and Robert B. Fisher. "Individual feature selection in each One-versus-One classifier improves multi-class SVM performance". In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*. 2014. URL: https://homepages.inf.ed.ac.uk/rbf/PAPERS/phicpr14.pdf.

[4] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[5] Honghua Liu et al. "Using t-distributed Stochastic Neighbor Embedding (t-SNE) for cluster analysis and spatial zone delineation of groundwater geochemistry data". In: *Journal of Hydrology* 597 (2021), p. 126146. ISSN: 0022-1694. DOI: 10.1016/j.jhydrol.2021.126146. URL: https://www.sciencedirect.com/science/article/pii/S0022169421001931.

[6] Prajit Ramachandran, Barret Zoph, and Quoc Le. "Swish: a Self-Gated Activation Function". In: (Oct. 2017). DOI: 10.48550/arXiv.1710.05941.

[7] Ketan Rajshekhar Shahapure and Charles Nicholas. "Cluster Quality Analysis Using Silhouette Score". In: *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. 2020, pp. 747–748. DOI: 10.1109/DSAA49011.2020.00096.

[8] Karl Thurnhofer-Hemsi et al. "Radial basis function kernel optimization for Support Vector Machine classifiers". In: *CoRR* abs/2007.08233 (2020). arXiv: 2007.08233. URL: https://arxiv.org/abs/2007.08233.

[9] C. Xiao, S. Hong, and W. Huang. "Optimizing graph layout by t-SNE perplexity estimation". In: *International Journal of Data Science and Analytics* 15 (2023), pp. 159–171. DOI: 10.1007/s41060-022-00348-7. URL: https://doi.org/10.1007/s41060-022-00348-7.