M1 Coursework: Testing inference pipelines for MNIST addition operations

Word count: 2637

December 18, 2024

1 Introduction

This report covers the different inference pipelines that could be used to sum two handwritten MNIST digits. In particular, it will cover the justification for choosing the architecture and parameters of each method, evaluating the contrasting performances, and discussing the benefits and drawbacks of each method.

1.1 About MNIST

The MNIST database [1] is a balanced database of handwritten digits commonly used for Machine Learning practices. It consists of 60,000 training images and 10,000 test images, such that each image is 28×28 pixels. This project aims to combine two MNIST images side by side, so each image is 28×56 pixels.

2 Justification of methodology: preprocessing and sampling

This study utilizes 50,000, 10,000 and 10,000 concatenated MNIST images for training, validation and testing, which follows the 75:15:15 rule for partitioning machine learning datasets. Scikit-learn was utilized to split the original training data into train and validation sets, in particular using the `train_test_split` function to randomly shuffle and split the MNIST training dataset into 80% training data and 20% validation data. This splitting was performed before combining images to prevent data leakage, ensuring that the model did not encounter validation or test images during training.

To generate side-by-side combinations of MNIST images, two random indices were selected from the 48,000 training samples without replacement, ensuring no two images were placed side by side more than once. A similar process was applied to the validation and test datasets, resulting in paired train, validation, and test images. The original labels of the train, validation, and test data were summed to give a new label, representing the sum of the two handwritten digits. figure 1 illustrates an example of a test image and its summated label.

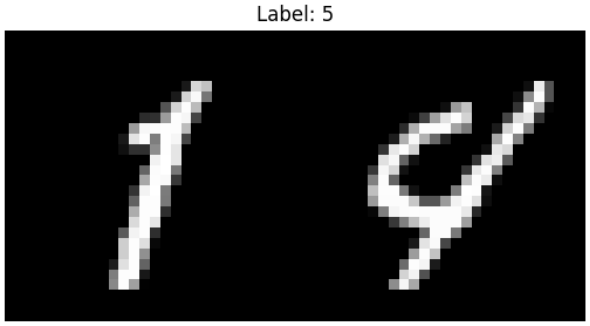

Figure 1: An example of a concatenated MNIST image

Similar to rolling two dies, the nature of summing two MNIST digits together dictates there will be more ways to create combinations of intermediate numbers like number 9, 10 and 11. This leads to more of such intermediate numbers sampled in the test, validation and training dataset as shown in figure 2, forming a binomial distribution.

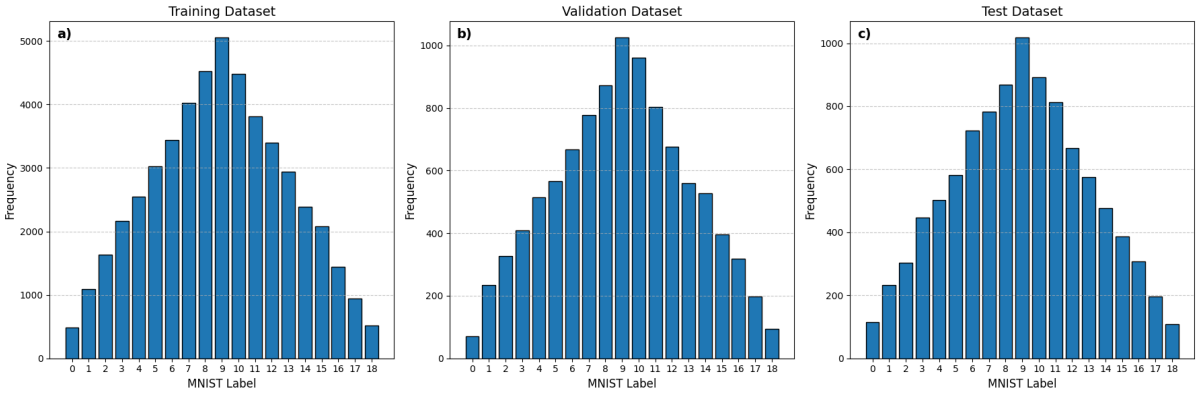

Figure 2: A histogram showing the frequency of MNIST digits in the a) training b) validation and c) test dataset

This report has adopted a random sampling approach from the binomial distribution, rather than uniformily across the 0-18 MNIST digits. This approach was chosen based on the objective of this neural network, which is to train a model to recognise the combinations of digits. Realistically, intermediate values such as 9 can be summed from more combinations than tail values like 0 and 18, thus the model should be exposed to more intermediate values to learn how to effectively categorize such values using different combinations.

A uniform sampling strategy would entail bootstrapping the tail-end digits, so the same frequency of MNIST digits are generated. However, this may lead to more tail-end digits resampled and the undersampling of intermediate combination digits. This may cause the model to overfit the tail-end digits, creating a biased model performance.

However, a potential drawback to using the random sampling approach is the risk of enhanced false positives, particularly for predicting a number as an intermediate value when it is not. This potential limitation of the sampling strategy will be further evaluated in confusion matrices in section 3.

3 Fully Connected Neural Network

3.1 Model structure and fixed hyperparameters

This section aims to build a fully connected neural network to classify the concatenated MNIST image to its summed label. To build an inference pipline that is well-trained, hyperparameterisation is key. The fully connected model consists of an input of layer of 1568 nodes, some hidden layers to be determined in hyperparameter tuning, and output layer of 19 nodes representing the 0-18 classes. Categorical cross entropy is chosen as the loss function as it is fitting for this multiclass classification problem.

3.1.1 Fixed activation functions

Activation functions is a crucial hyperparameter as they introduce non-linearity into a model. For this study, activation functions are needed to learn the non-linearity between the 1568 pixels inputted and the summed label outputted. Common activation function used in machine learning include the sigmoid function and rectified linear unit (relu) function. The majority of studies have found that the relu activation function outperforms the sigmoid functions, as relu avoids the vanishing gradient problem, which allows for more efficient backpropagation [3]. Especially for MNIST classification tasks, Relu is effective at 'switching off' neurons by outputting zeros for negative values, which is effective for the

model in learning MNIST's background pixels with values close to 0. Therefore, this study has chosen to fix relu as one of the model's hyperparameters. The activation function used for the output layer is softmax, a multiclass classification function suited for outputting the probabilities of 0-19 classes.

Optimisers are crucial hyperparameters in minimising loss functions and adjusting the model's weights through backpropagation. One of the most popular optimisers is the ADAM optimiser, which involves the first moment (average of gradients), to determine the direction of gradient descent and the second moment (the average of gradient squared) to compute the magnitude of the step size in gradient descent. As this optimizer is computationally effective and proven to outperform other optimisers in MNIST analysis, it is chosen as a fixed parameter.

3.2 Hyperparameter search

Many other hyperparameters should be tuned depending on the characteristics of the specific dataset, this study used optuna, as the package provides functionalities such as early stopping to avoid overfitting. It also uses bayesian optimisation to search through the hyperparameter space to focus on promising regions. This study searched through the number of hidden layers (1-3), batch size (16, 32, 48 and 60), drop out rate (0 - 50%), number of units (32 - 128) and learning rate ($10^{-5} - 10^{-2}$), as these are critical parameters dictating the computational efficiency, accuracy and bias-variance trade off.

After running 30 trials and 15 epochs each, optuna found the best in Table 1. These parameter values were tuned with validation data, and the best model obtained a validation accuracy of 0.88950.

Hyperparameter	**Value**
Number of Layers	3
Batch Size	64
Dropout Rate	0.0553
Units	96
Learning Rate	0.00127

Table 1: Best hyperparameters for the neural network.

1. Hidden layers: Three hidden layers was optimal in capturing the non-linear relationships in the dataset, without sacrificing computational time and resources. While the original single digit MNIST classification tasks might use two hidden layers, the concatenated MNIST labels may require an extra layer to learn the relationship between digits.

2. The batch size: The largest batch size of 64 samples was chosen for the model, likely because larger batch sizes average the gradients between more samples, allowing for steadier and faster convergence to a minimum.

3. Drop out rate: The optimum drop out rate was 0.0553, suggesting 5.53% of nodes should be dropped randomly from each hidden layer. This effectively introduces noise to the model, which helps to prevent overfitting and improve the model's generalisation capabilities.

4. Number of units: 96 units were used for each hidden layer. This is a reasonable number of nodes that allows the model to capture more details in an image and learn more complex features, while not sacrificing computational efficiency.

5. The learning rate: A learning rate of 0.00127 was optimal for the model. This size ensures the updates are large enough to be computationally efficient, whilst small enough to not overshoot the minimum.

3.3 Performance of Best Performing Neural Network

The training accuracy increased logarithmically as shown in figure 3. The model avoided the issue of overfitting, as shown by 3, such that there is no occurence of validation accuracy falling whilst train accuracy rising in greater epochs. When validating this model against test data, it produced a score of 0.9027.

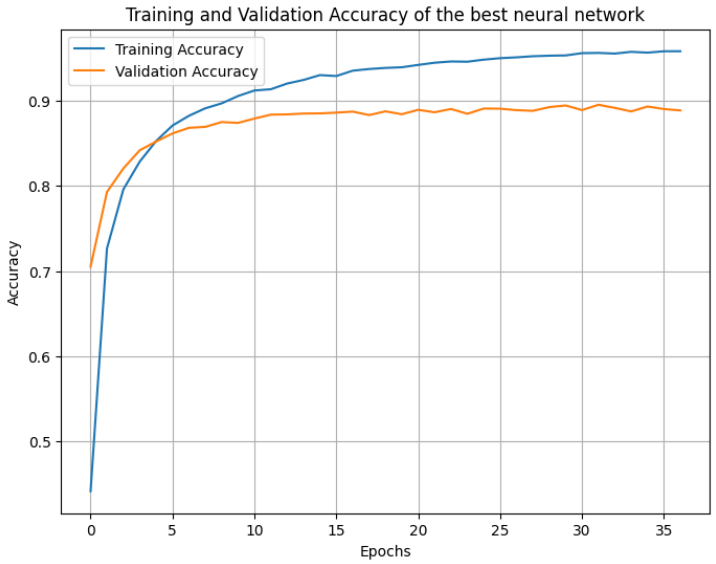

Figure 3: Training and validation accuracy of the best model

The confusion matrix is plotted in Figure 4, showing the true positives along the diagonal in blue, and the misclassified occurences in red.

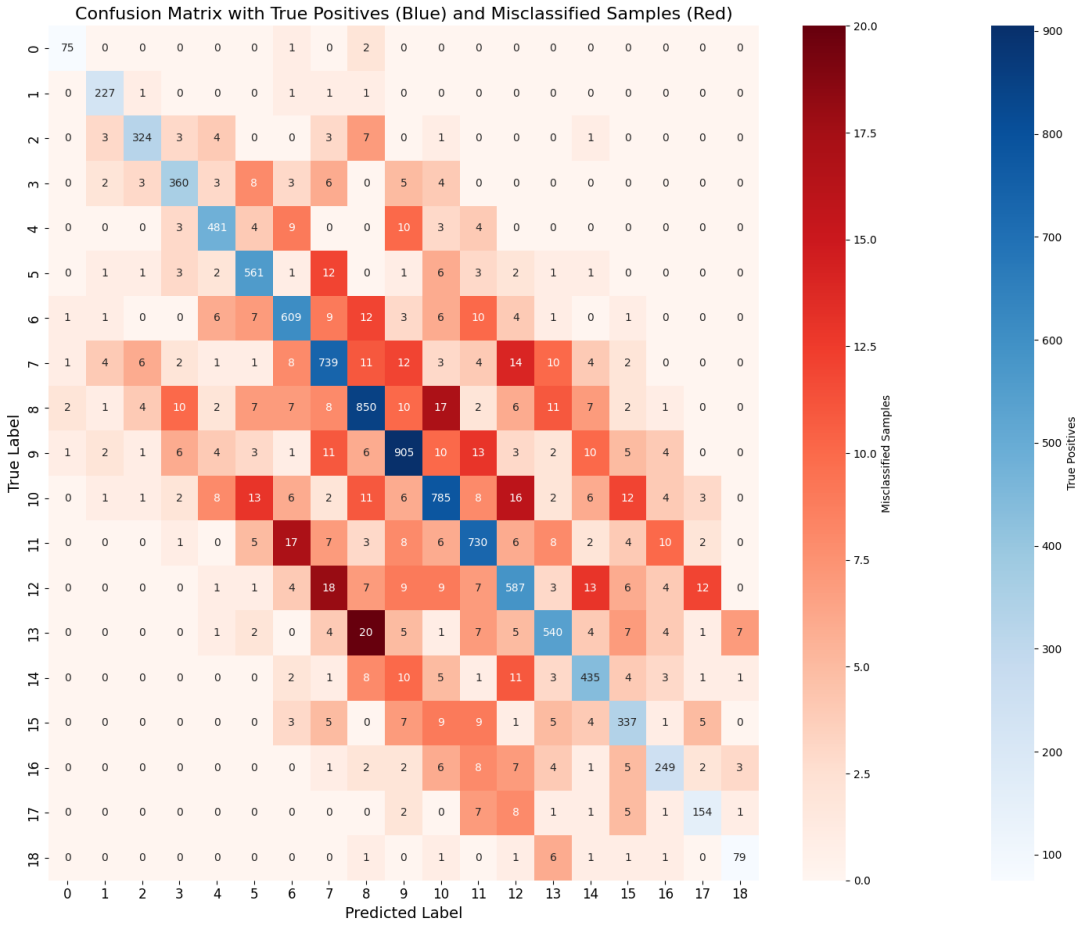

Figure 4: A confusion matrix showing the number of misclassifications (red) and true positives (blue).

Most true positives occur in the intermediate values, which is a direct result of the binomial distribution of the test set. There are also more misclassified intermediate values than tail-end values, suggesting there may be some bias of the model generating false positives for the frequently trained intermediate values. Interestingly, the misclassified samples also appear along a diagonal, implying single digit values are more likely to be misclassified as another single digit, and a double digit number is more likely to be misclassified as a double digit number.

4 Other Inference Algorithms: Support Vector Machine, Random Forest, Decision Tree

This section will compare other inference algorithms such as support vector machine (SVM), random forest and decision tree to compare the performance of the models.

4.1 Hyperparameter Tuning

Figure 5 show hyperparameter tuning of gamma for SVM, and maximum depth of tree for random forest and decision tree. The parameter yielding the highest validation accuracy was used in the model for the training data. Note, only 10% of the training data was used for hyperparameter tuning for SVM, as SVM is computationally expensive.

4.2 Performance Comparison

After training and testing on the full dataset, the test accuracy for the trained neural network, SVM, random forest and decision tree is presented in Figure 6.

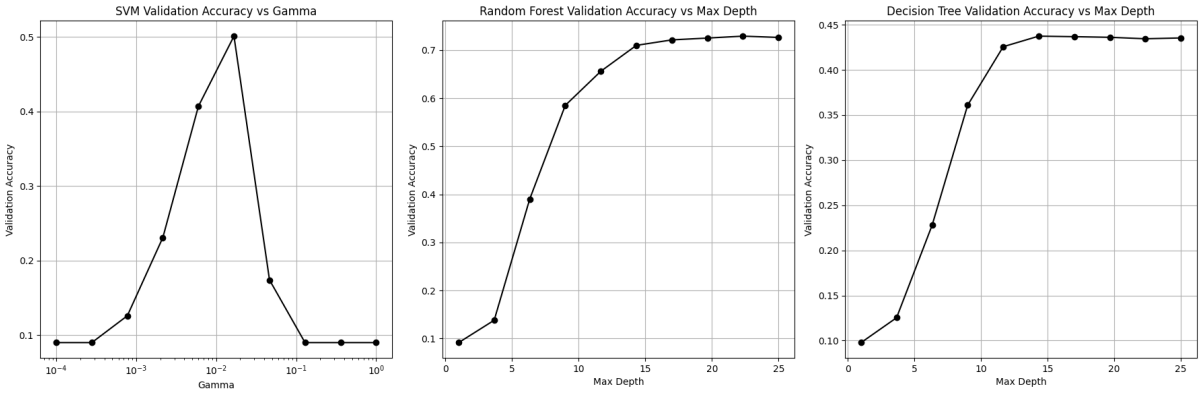

Figure 5: Best hyperparameters obtained for the SVM, Random forest and Decision tree

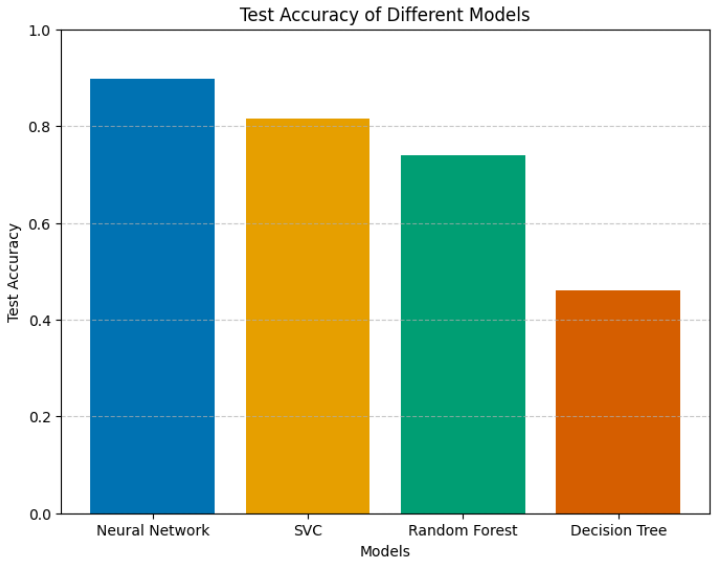

Figure 6: Test accuracy of the different machine learning models

The neural network outperformed other algorithms as it is capable of deep learning and understanding complex, non-linear relationships. Specifically, backpropagation and adjustment of weights in each layer mean that each layer is capable of learning features in an image, such as the edges, lines and loops of a digit.

Decision tree was not able to generalise the relationships of the concatenated MNIST images due to the algorithm architecture. A decision tree is commonly used for classifying simple, low dimensional relationships through splitting at feature thresholds until a leaf is reached. Given each concatenated MNIST image has 1586 (28×56) pixels, the decision tree algorithm is prone to overfitting models, which in turn yields poor test values.

Random forest algorithm, using an ensemble of decision trees, obtained results almost one fold better than that of the decision tree. This is due to the noise introduced through bootstrapping the training data (sampling with replacement) and selecting random subset of features for split at each node. Conceptually, this is similar to adding regularisation in neural networks, as it encourages the model to learn the data pattern without overfitting to the training set.

5 Linear Classifier: Logistic Regression

To showcase the importance of introducing non-linearity to the concatenated MNIST image classification problem, this section will compare the performance of training a single linear classifier with a sequential linear classifier. A sequential classification is defined as classifying the two digits of the concatenated images separately using two linear classifiers. The predicted labels are then added together before validated against the true summed label. The linear classifier chosen for this study is the logistic regression.

5.1 Single and Sequential Linear Classifier Probabilities

To compare the probabilities obtained from the two methods, the single and sequential logistic regressor is trained on 1000 random train concatenated MNIST images. The two models are evaluated against one test image number 9, displayed in Figure 7.

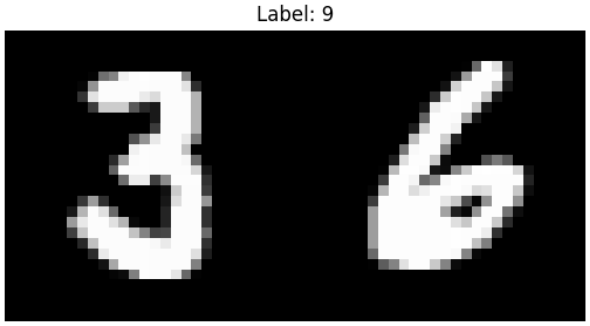

Figure 7: Visualisation of the MNIST test image for testing

The probabilities obtained by both classifiers are plotted in 8. The single logistic regression did not accurately predict a label of 9. However, the two sequential logistic regression predicted the correct numbers for the left and right MNIST digits respectively, thus would accurately produce the summed predictions.

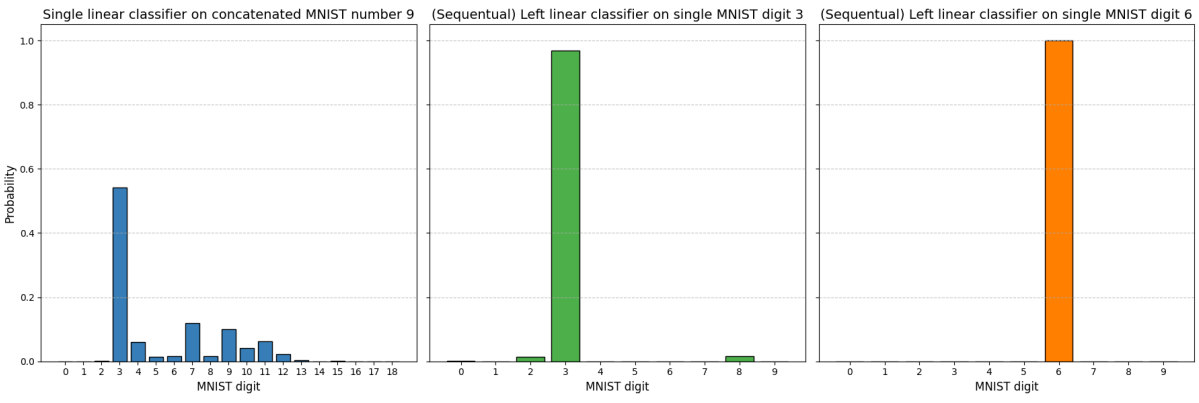

Figure 8: The probabilities of the predictions for the single linear classifer (blue), and the probabilities for the sequential classifier for the left digit (green) and right digit (orange)

This exercise highlights that a complex exercise such as identifying the sum of numbers cannot be easily learnt by a weak linear classifier. Similar to the XOR problem, understanding the interaction of two digits summing to a label is a complex tasks that require non-linear classification techniques.

5.2 Performance of Single and Sequential Linear Classifiers with Different Sample Sizes

When using increasing training sample sizes, test accuracy of fitting the concatenated MNIST images through a single linear classifer is consistently poor, never exceeding an accuracy of 20% as shown in Table 2. As mentioned above, the linear classifier lacks the incapacity to understand the relationships within a high-dimensional dataset such as the concatenated MNIST images. This is a limiting factor for the test accuracy, such that increasing the size of training data does not improve the classifier predictions.

Sample Size	Single Linear Classifier Test Accuracy	Sequential Linear Classifier Test Accuracy
50	0.1235	0.3922
100	0.1256	0.5280
500	0.1364	0.7380
1000	0.1390	0.7713

Table 2: Comparison of Test Accuracies for Single and Sequential Linear Classifiers with Varying Sample Sizes

In comparison, the test accuracy of the sequential linear classifier increased significantly with more samples. When the classification task is handled with a non-linear classification technique, the ability to make predictions is no longer limited by the choice of model. Thus, as sample size increases, the sequential linear classifier is able to train on more data to improve the accuracy of predictions.

6 t-SNE Distribution

To assess the neural network's ability to learn the features of the concatenated MNIST images, t-SNE is employed to visualise how the input layer and embedding layer of the neural network differ in clustering features. t-SNE is a non-linear dimensionality reduction technique that aims to present the high-dimensional data in an intepretable 2D plot, without sacrificing the relationships between points. This is done through minimizing the Kullback-Leibler divergence.

An important hyperparameter that will be optimised is the perplexity, which dictates the balance of preserving and presenting local and global structures in the plots. A perplexity of 5, 30, 50 will be tested as instructed by the t-SNE developer's notes [2]. To observe and analyse t-SNE performance at very high perplexities, a perplexity of 100 is also studied in this section. Due to computational constraints, 5000 points are randomly sampled from the input and embedding layer for t-SNE analysis.

6.1 Input Layer

As shown in Figure 9, no clear clustering of classes can be observed when applying t-SNE to the input layer. This is as expected, as the input layer is a high dimensional vector of 1568 features (28 pixels x 56 pixels). These raw inputs carry lots of information across the 1568 dimensions, which are not easily to summarized into a 2D plot. Thus, the clustering of the 5000 images is poor, and no clear classes can be identified.

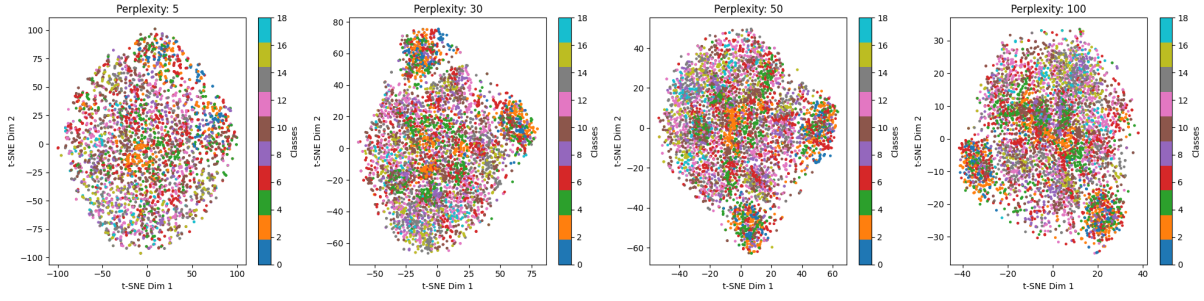

Figure 9: t-SNE plots with varying perplexities value for the input layer

Due to the complexity of the input data, this outweights the ability for t-SNE to optimising the preservation of local and global structures, such that changing perplexity values do not lead to drastic improvements in clustering. It is also worth noting that as there are 5000 inputs plotted in the graphs, this crowds the plots, rendering it difficult to draw meaningful conclusions.

6.2 Embedding Layer

The embedding layer t-SNE plots in Figure 10 show drastically clearer classifications of the classes compared to the input layer. This improvement can be attributed to the feature learning in the previous two layers, such that each layer has selectively extracted meaningful features in the concatenated MNIST images before passing down to the next. Upon reaching the embedding layer, the learned features allow the network to better classify the MNIST digits. Additionally, the embedding layer only has 96 nodes, which is a 16 fold decrease in dimensionality compared to the input layer. This allows for the local and global relationships to be better preserved on the t-SNE plot.

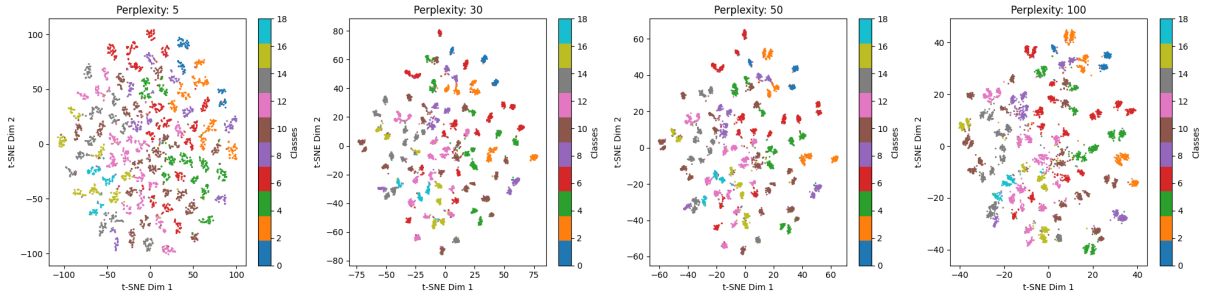

Figure 10: t-SNE plots with varying perplexities value for the embedding layer

Contrasting to the t-SNE plots in the input layer, the clustering at different perplexity values for the embedding layer are distinctly different as shown in 10. Smaller perplexities like 5 emphasises the preservation of local neighbourhoods, rather than the global relationships. For example, within the digit class of 4, each point is loosely clustered. This implies that nuanced differences between the handwritten digit 4s are preserved. Compared to larger perplexities like 50, t-SNE emphasises less on the local relationships, as seen from the tight clustering of the classes. In contrast, the distance between each class is larger, implying more emphasis on the global differences between classes. However, at very large perplexities like 100, as global relationships are greatly emphasised, some local relationships may be lost, causing the splitting of clusters in each class.

7 Conclusion

For the task of learning concatenated MNIST images, it was found that the neural network, after hyperparameter tuning using optuna, was the best at predicting summed MNIST digits. It's performance exceeds that of random forest, decision trees, SVM and linear classifiers. The neural network's strong performance can be attributed to the layers of successful feature learning in the network, as shown in the t-SNE plots. For future applications, this suggests neural networks are superior to other algorithms in learning complex relationships.

8 Appendix

Generation tools such as ChatGPT and Claude.ai are used to debug and refine the python code associated to this report. Such tools were used to help debug the confusion matrix to display the misclassified samples in red. It was also used to help index individual digits of the concatenated MNIST digits, and split the images before processing with the sequential classifier in part 5. The tools were used to facilitate LaTeX fomratting for figures and tables used in the report.

References

[1] Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[2] Laurens van der Maaten. t-distributed stochastic neighbor embedding (t-sne). `https://lvdmaaten.github.io/tsne/`, 2008. Accessed: 2024-06-18.

[3] Shin-Hao Wang and Eric Sakk. The effect of activation function choice on the performance of convolutional neural networks. *Journal of Emerging Investigators*, 2024. Kang Chiao International School, Taipei City, Taiwan; Morgan State University, Baltimore, Maryland.