UNIVERSITY OF CAMBRIDGE

DiS MPHIL

M1 Machine Learning Coursework

Building an Inference Pipeline to Add Two Handwritten Digits

P. Antonopoulos
December 2024

Abstract

In this project, the performance of a neural network, random forest and support vector machine in adding two MNIST digits were compared. It was found that the best models found through hyper-parameter tuning achieved accuracies of 94.85%, 81.66% and 58.36%, respectively. Additionally, the performance of a logistic regression model was investigated when applied on two digits simultaneously or sequentially. Finally, the t-SNE distribution of the classes were visualised and compared when applied in the embedding layer and directly on the input data.

Word count: 2995

Contents

1 Introduction

The field of Machine Learning (ML) continues to rapidly expand. Amongst others, ML can be applied in fields such as finance, engineering and medicine to solve, or aid in solving, a wide range of problems [1]. Classification tasks are standard situations where approaches that use ML can be pursued to find a solution. In tackling such problems, techniques including support vector machines, ensemble methods such as random forests or more complex models, namely neural networks may be applied.

In this project, a supervised learning task of adding two MNIST digits was approached by modifying the MNIST dataset and developing three separate inference pipelines using a neural network, random forest classifier and support vector machine and comparing their performances. In addition, it was investigated whether a weak linear classifier tasked to add two numbers would perform better when trained on two images simultaneously or sequentially. Lastly, the t-SNE method was utilised to visualise data from the neural network to compare it with the raw input dataset to unveil hidden structures.

This report begins by providing a concise overview of the necessary background theory required for the project before presenting a comprehensive overview of the method followed to create the dataset and model pipelines, compare the weak linear classifiers and visualise the data using t-SNE. The inference results from each pipeline is then presented and compared before showcasing the results from the weak linear classifier investigation together with the t-SNE plots. Finally, a critical discussion of the project findings is made before concluding the report.

2 Background

2.1 The MNIST Dataset

The MNIST dataset is perhaps the most well known dataset used to train and evaluate ML models [2]. It is comprised of 70000 28x28 greyscale images of handwritten digits from zero to nine where 60000 are training samples and the remaining reserved for testing. The dataset is used for supervised learning as each image comes with a label corresponding to the digit in the image.

2.2 Neural Networks

A neural network can be thought of as the artificial representation of a human brain. A typical neural network consists of an input layer, hidden layer(s), and an output layer [3] where the output of layer n corresponds to the input of layer $n + 1$. Each layer contains a number of neurons, or units, which like neurons in a brain are connected to many other neurons in the next layer. In the case of a fully connected neural network such as the one used here, each unit is connected to all units in the next layer. Additionally, in the brain, each neuron only fires if the weighted sum of inputs to that neuron exceeds a certain threshold value [4]. Similarly, each unit in a neural network computes the weighted sum of its inputs and applies an activation function to determine whether the unit fires and with what strength [5].

The process of training a neural network is to allow the model to determine the weights which minimise a loss function [5] that quantifies the performance of the model by measuring the deviation of the predicted output to the target output.

2.3 Random Forest and Support Vector Machines (SVMs)

Random forests are an ensemble method which utilise a set of decision trees to classify data [6]. Decision trees themselves are based on a series of questions, or branches which aid classification of input data by repeatedly splitting the data until the final classification is reached at a leaf [5]. On the other hand, Support Vector Machines (SVMs) use outliers in the data (the support vectors) to form hyperplanes which maximise the margin between the two datasets [7].

2.4 Logistic Regression and t-SNE

The weak linear classifier used in this project was the logistic regression model. A logistic regression model can be understood as a perceptron [8] which uses a logistic activation function and a cross entropy loss function in training [5] [9]. The t-Stochastic-Neighbour-Embedding (t-SNE) method is primarily used to visualise the underlying structure of datasets with a large number of features while simultaneously preserving local properties [5]. The approach assigns a probability distribution to the surroundings of each data point and ultimately uses this to separate points [5].

3 Methods

3.1 Constructing the Dataset

To develop models using supervised learning which adds two digits, an appropriate dataset consisting of 56x28 images of two digits with their corresponding labels had to be constructed for training and evaluation. This was done using the standard MNIST dataset where firstly, the images were loaded into a Jupyter notebook and split into train, validation and test data together with one-hot encoding the labels as is standard in classification tasks. Following this, a Python function was written which took input data, its corresponding labels and an argument specifying how many samples to generate to construct a new dataset.

This function generated two random indices to access the corresponding images and labels from the input dataset. The two images were then combined using `np.vstack()` to form a single 56x28 image of two digits and the two original labels were added to produce the label of the sum. These results were then stored and the procedure repeated for however many samples were required. Although the original MNIST dataset contains 60000 training and 10000 test images, it was decided that the number of samples should be increased due to a larger number of classes and features in each image. Therefore, 100000 training, 25000 validation and 25000 test images were generated using this approach in under 2.5 seconds.

It was also necessary to preserve the statistical properties of the dataset. Specifically, when two random integers between zero and nine are summed, a larger number of combinations which sum to nine means that the resulting histogram forms a triangular distribution where the probability is maximum at nine. This statistical property was preserved because ML models should be trained and evaluated on data which is representative of data they might be exposed to if deployed in a real world setting. Therefore, to preserve this property, no requirements that the number of examples in each class had to be equal were applied. It should be noted that differing numbers of examples in each class further justified the need to generate more samples than provided in the original MNIST dataset. Additionally, reproducibility was guaranteed by fixing a random seed before constructing the dataset.

3.2 Developing the Neural Network

To develop the neural network pipeline, hyper-parameter tuning was performed to establish the best model architecture to address the given problem. This was carried out using the `optuna` package which automatically optimises the hyper-parameters and removes the need to do manual grid searches. To create an `optuna` study, an objective function was defined which specifies the hyper-parameters to tune and suggests a range of values to try. The hyper-parameters that were tuned in this study were the activation function, number of hidden layers, the number of units in each dense layer and the learning and dropout rates. After suggesting the hyper-parameters, the network was constructed by creating an instance of a sequential model from `keras` and adding an input layer. Following this, for each layer that `optuna` trialled, a dense layer, batch normalisation layer and dropout layer were added.

A batch normalisation layer was used as the data was split into mini-batches (each containing 128 images) for training and therefore it acts as a regulariser by introducing extra randomness during learning. Additionally, batch normalisation would help decrease training times by suppressing vanishing gradients. To prevent over-fitting, a dropout layer was also added to serve as another regulariser which worked by randomly dropping out units during training to suppress situations where a classification depended too heavily on the output of a few units.
At this point, the output layer with a softmax activation function was added since the softmax function's output represents the probability of the input being in each class. The model was then fit using the ADAM optimiser with a categorical cross-entropy loss function over 20 epochs which was decided to be a good balance between sufficient training time and preventing the model from over-fitting. Finally, the model was evaluated on the validation data before returning the accuracy from the objective function.

The objective function was then optimised over 50 trials by maximising the accuracy together with specifying a seed to `optuna` so the results would be reproducible. On a MacBook Air M2 with 8GB RAM, this took between 4-5 hours which was determined to be suitable. To avoid having to re-run the study, all data was saved to `.csv` files.

The results of the `optuna` study were read into a separate Jupyter notebook which used the optimal hyper-parameters to train the best performing neural network, save the weights to a file and evaluate the model on test data.

3.3 Developing the Random Forest and SVM

Both the random forest classifier and SVM depend on their own hyper-parameters, such as the number of decision trees and maximum depth of a tree in a random forest and the regularisation strength and kernel coefficient for the SVM. Therefore, to make a fair comparison to the best performing neural network, it was decided to tune the hyper-parameters of these models.

The tuning for each model was done in the same way as the neural network with the appropriate hyper-parameters. It was noted that the input data needed to be reformatted by flattening the images and converting the one-hot encoded labels back to their integer representations to work with the `scikit-learn` functions.

3.4 Investigating the Weak Linear Classifier Performances

To compare the logistic regression model performances as a function of sample size when applied on the combined images or sequentially, the `scikit-learn` logistic regression class was used to define two models. Additionally, a function was defined to evaluate the accuracy of the second model which takes in a 56x28 image, splits it into two 28x28 images and uses the model to predict the labels of the images. The predictions were then summed and if the result matched the label of the initial 56x28 image then it was judged to be correct. For each sample size, the first model was trained on the 56x28 dataset and the second model on the original MNIST 28x28 images where the number of training samples corresponded to the sample size.

Following this, the performance of the first model was computed using the `score()` method of the logistic regression class while the second model utilised the function to evaluate the accuracy when the images were applied sequentially.

3.5 Implementing t-SNE

To show the t-SNE distribution of the classes in the embedding layer of the neural network, a sub-model was created using `keras` with outputs corresponding to the embedding layer. This sub-model was used to extract embeddings by predicting the classes of images from the 56x28 image dataset. A t-SNE object was then created and the `fit_transform()` method used to visualise the distribution of classes in the embedding layer.

Similarly, to visualise the representation when applied on the raw input, another t-SNE object was used where the input data was passed into the `fit_transform()` method. For efficiency, only a subset of 5000 images were used as t-SNE is computationally expensive. To improve the visualisation of the data structures, the perplexity was optimised by eye through trialling different values and observing which returned appropriate structures.

4 Results

4.1 The Dataset

The new dataset successfully preserved the statistical properties as seen from the figure below.

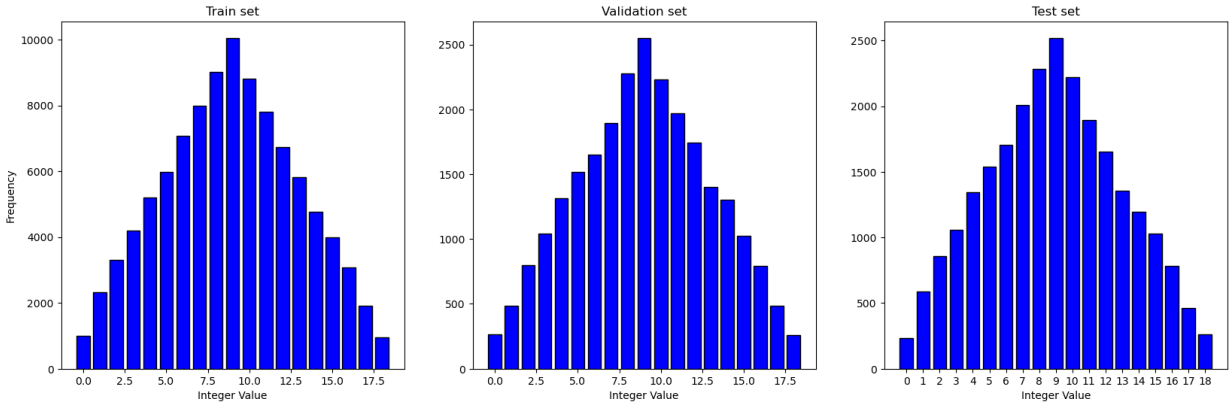

Figure 1: Histogram showing the frequency of each class in the combined 56x28 image train, validation and test datasets.

4.2 Neural Network

A visualisation of the hyper-parameter search conducted using `optuna` was made by plotting the accuracy for each trial and is shown below.

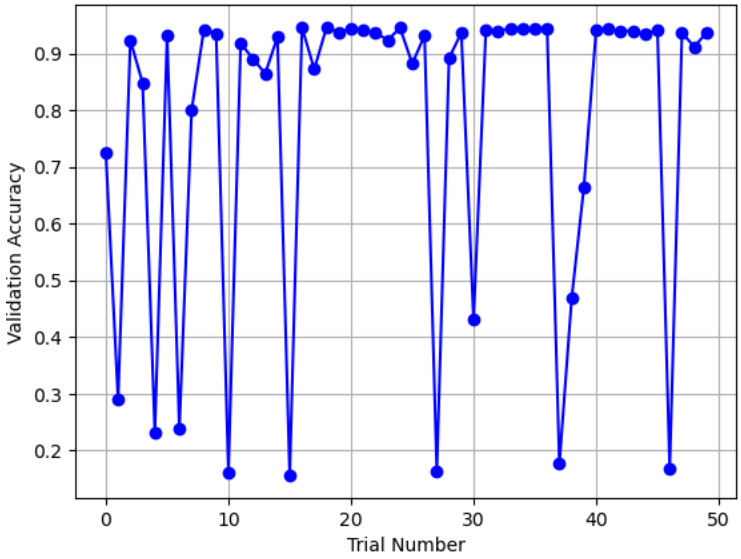

Figure 2: Validation accuracy across the `optuna` trials.

The optimum hyper-parameters which resulted from the `optuna` study are summarised in the table below.

Hyper-parameter	Optimal Value
Activation function	Sigmoid
Number of units	418
Learning rate	0.000757
Dropout rate	0.298
Number of layers	5

Table 1: Summary of optimum neural network hyper-parameters found through `optuna`.

A plot of the validation accuracies as a function of epochs could then be made when the best performing neural network was trained and is shown below.

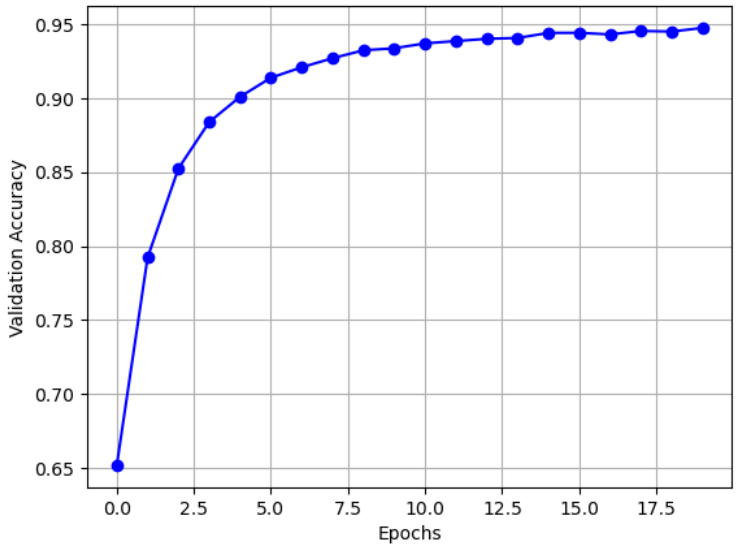

Figure 3: Best neural network validation accuracy across training epochs.

Finally, this neural network (with 1,372,731 parameters) had an accuracy of 94.85% when evaluated on the test data.

4.3 Random Forest and SVM

Using the optimal hyper-parameters for the random forest and SVM to train these models, the accuracy on the test data was evaluated and compared to the performance of the neural network as summarised below.

Model	Test accuracy (%)
Neural network	94.85
Random forest	81.66
SVM	58.36

Table 2: Comparison of the best neural network, random forest and SVM test accuracies.

It should be noted that the optimal hyper-parameters were a random forest with 285 decision trees, each with a maximum depth of 196 and a SVM with a regularisation parameter of 27.6 and a kernel coefficient of 0.0131.

4.4 Logistic Regression Performance

The accuracies of the two logistic regression models as a function of sample size is presented below.

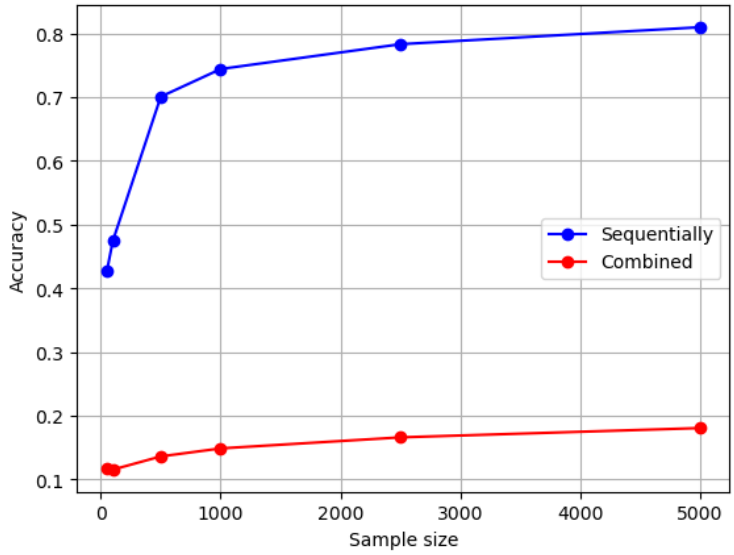

Figure 4: Logistic regression model performance when applied on the combined dataset or when the images are applied sequentially as a function of sample size.

4.5 t-SNE Visualisation

A comparison of the t-SNE distribution of the classes within the embedding layer and the representation obtained from the input layer is shown below.

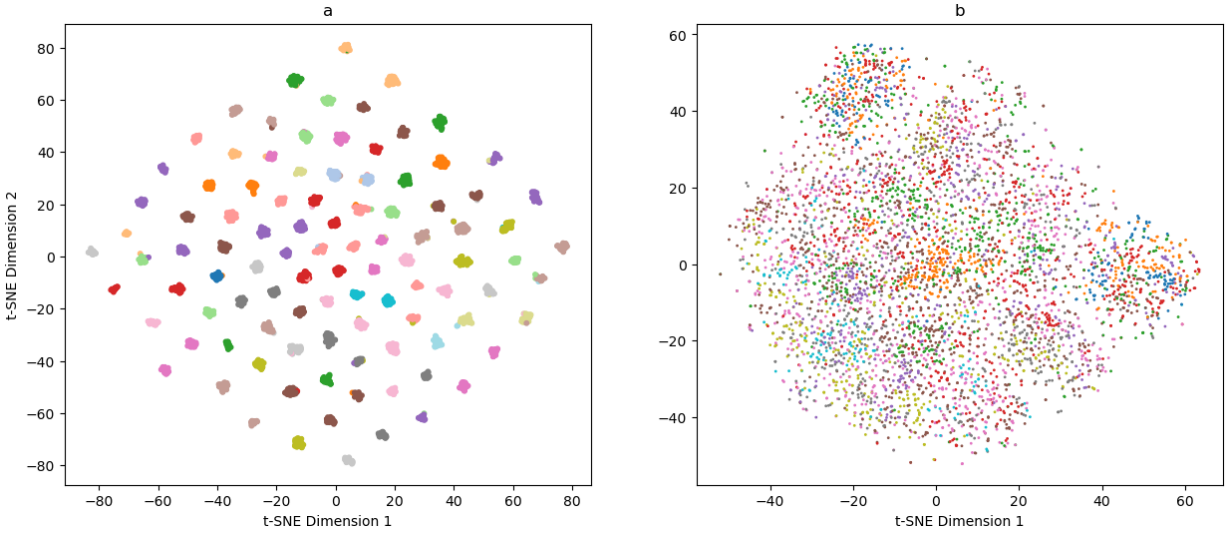

Figure 5: (a) t-SNE distribution in the embedding layer. (b) the representation when directly applied on the input data.

The optimum perplexities used to obtain these plots were 30 and 55 for the embedding layer and input dataset, respectively.

5 Discussion

Figure 2 illustrates the progression of the `optuna` study where different combinations of hyper-parameters yield different validation accuracies. Ultimately, the study resulted in the optimum hyper-parameters as presented in Table 1. It can be seen that the network is relatively complex with 5 hidden layers and 418 units per layer which suggests it could be prone to over-fitting. However, the accuracy of 94.85% when evaluated on the test data being approximately equal to the validation accuracy demonstrates that the network did not over-fit. Additionally, Figure 3 does not show any decrease in validation accuracy over the training epochs which further confirms that no over-fitting occurred. Lastly, after inspecting the weights matrix from each layer there were no weights equal to zero which would have meant some units would never be activated.

While it was expected that the optimal activation function would be the ReLU function due to its popularity in classification problems as it avoids vanishing gradients [5], it was noted that the batch normalisation layer would have taken care of that problem. Therefore, the model would be free to explore other activation functions such as the sigmoid which would have otherwise suffered from vanishing gradients.

Table 2 presents a performance comparison between the best neural network, random forest and SVM. It can be seen that the neural network outperforms the other models with the SVM yielding the lowest accuracy. The neural network performing better was according to expectation because it utilises hidden layers and therefore has a large representational power. This stems from the universal approximation theorem which states that any continuous multivariable function can be approximated with arbitrary accuracy using a neural network with just a single hidden layer [5]. In this case, the neural network has five hidden layers and will correspondingly have a high expressivity that it can use to fit to the data.

In comparing the random forest and SVM, it was noted that for efficiency, the SVM was only trained on 5000 images whereas the random forest and neural network could be trained on the full dataset. Therefore, an insufficient training sample could be a reason as to why the SVM performed worst. To further explore this, the SVM may be trained on google colab or a supercomputer where advanced GPUs could speed up training.

From Figure 4, it can be seen that the logistic regression model applied sequentially on two digits yielded a higher performance then when applied on the combined images. This was expected because the logistic regression model is a weak linear classifier where its linear properties inhibit the model's potential to learn complicated patterns within handwritten digits [10]. It can be understood that 28x28 images of single digits would have simpler patterns thus making it more straightforward to learn than 56x28 images containing two digits. Therefore, when trained on the former data and applied on images of two digits sequentially, the model would be able to perform better compared to being trained and evaluated on the more complex dataset.

It was also noted that the method to determine the sequential accuracy could be improved. This is because the prediction could be judged as correct if the model made an incorrect prediction on both numbers but which still happened to sum to the true label. However, the probability of this occurring was determined to be low and hence would negligibly contribute to the accuracy.

Finally, the t-SNE distributions look as expected. The representation of classes in the embedding layer (Figure 5 (a)) shows clear structure as the data would have passed through the whole network with all features being appropriately weighted. On the other hand, the representation when applied on the input dataset (Figure 5 (b)) contains some clusters however it is evident there is less structure than in the embedding layer.

6 Conclusion

In conclusion, inference pipelines to add two handwritten digits were developed using a neural network, random forest and SVM. Hyper-parameter tuning was performed to determine the best models for accurate and fair comparisons. Additionally, the performance of a logistic regression model was investigated when sequentially applied on the images or when they were combined. Finally, the t-SNE distribution of the data was visualised and compared when used in the embedding layer or on the input dataset.

References

[1] I. El Naqa and M. J. Murphy, *What is machine learning?* Springer, 2015.

[2] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[3] M. Uzair and N. Jamil, "Effects of hidden layers on the efficiency of neural networks," in *2020 IEEE 23rd international multitopic conference (INMIC)*. IEEE, 2020, pp. 1–6.

[4] C. M. Bishop, "Neural networks and their applications," *Review of scientific instruments*, vol. 65, no. 6, pp. 1803–1832, 1994.

[5] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists," *Physics reports*, vol. 810, pp. 1–124, 2019.

[6] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[7] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[8] G. Tzougas and K. Kutzkov, "Enhancing logistic regression using neural networks for classification in actuarial learning," *Algorithms*, vol. 16, no. 2, p. 99, 2023.

[9] A. Zaidi and A. S. M. Al Luhayb, "Two statistical approaches to justify the use of the logistic function in binary logistic regression," *Mathematical Problems in Engineering*, vol. 2023, no. 1, p. 5525675, 2023.

[10] D. Stilinski and A. Robert, "Comparing error rates of mnist datasets using various machine learning models," EasyChair, Tech. Rep., 2024.