A Comparison of Two Neural Networks Using the MNIST Database

Ashley Mason and Fred Miles

1 Introduction

This study involves working with the renowned MNIST database - a large collection of labelled handwritten digits. This is a supervised learning problem, with the objective of training our two neural network models to recognise these handwritten digits, before testing both models' ability to correctly classify unseen versions. Several papers have been written with the aim of producing the highest accuracy rate and there has been considerable success. While the performance of state-of-the-art models may not be replicated here, the database is ideal for providing comparable results from our two neural network models, which can then be critically evaluated. The two neural network models chosen are the Multi-Layer Perceptron (MLP), and a Convolutional Neural Network (CNN).

2 Dataset Description

The MNIST database consists of 60,000 training images and 10,000 testing images. Each individual digit is a grayscale image and has been normalised so each pixel value is between 0 and 1. Each image is 28x28 pixels and they have 10 possible labels ranging from 0 to 9. Figure 1 shows a small subset and offers a view of the variation the handwritten digits have both across and within each class. Figure 2 is a closer look at an individual image, with the values of pixels displayed.

Transforming the images is an important step, as the MLP and CNN require the images to be inputted in different formats. For the MLP, the pixels for each image need to be squashed into a 1-dimensional vector of length 784 (i.e. 28 x 28). The CNN is designed to accept 3-dimensional images - height x width x colour channels. In this case, each image will need to be transformed into a matrix of size 28x28x1. Only one colour channel is needed due to the images being grayscale.

Figure 1: A subset of MNIST images.

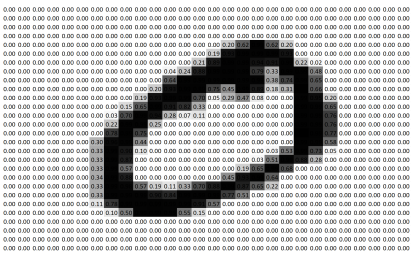

Figure 2: An individual image.

3 Summarisation of the Two Neural Networks

3.1 Multi-Layer Perceptrons

The Multi-Layer Perceptron (MLPs) is the first neural network model to be examined. Generally, the network is designed with one input layer, one output layer, and some number of hidden layers, all of which are fully connected. Except for the inputs, every layer contains neurons that are modelled as perceptrons. These perceptrons receive weighted components from a previous layer, extract features from these components, assign weights based on those features, and then feed these through to the next layer via an activation function.

This is known as the feed forward pass and ends with the model producing a predicted output based on the weights within the network. These predictions are then compared to the true values, and a loss function is evaluated. Backpropogation is the learning engine behind MLPs and is achieved through stochastic gradient descent [1]. Partial derivatives of the loss function are computed with respect to the weights at each layer and are back-propogated through the network. The weights are then updated before being fed forward once again and evaluating the loss function. This process is repeated until convergence and a minimum is achieved. There is no guarantee of a global minimum, however, there are two key hyperparameters that can be tuned to ensure a stronger likelihood of this occurring: momentum and learning rate.

One major advantage of the MLPs is that they require no prior assumptions about the underlying distribution of the data [2], and can handle a wide variety of data types - for example, images. The deep learning aspect also makes MLPs among the most powerful classifiers, with its performance eclipsing many other non-deep learning classifiers. MLP has been successfully applied on a wide variety of real world problems, ranging from predicting the weather [3], identifying breast cancer [4], to forecasting stock market prices [5], demonstrating its versatility. Furthermore, while MLPs' architecture may be simpler than other neural networks, this simplicity means it is computationally cheaper to run compared to other more complex models.

On the other hand, this does result in inferior performance levels, which is one of the criticisms of MLPs. Another drawback of MLPs is that a global optimum is not guaranteed, and convergence to a local minimum can result in poor solutions to a problem. To help avoid this, fine-tuning an MLP's hyperparameters is required, which can be an expensive task, and still might not ensure a global minimum.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is the second deep learning model selected. CNNs are particularly prevalent in computer vision, and since we are working with images, it is a very suitable choice. The design of CNNs is more complex than MLP and where this complexity lies is in the make up of the hidden layers. These tend to be constructed of convolutional, normalisation and pooling layers, before culminating in fully connected layers and finally the output layer.

It is worth examining two of the layers mentioned in closer detail. The convolutional layer is the basis of this model. Each layer contains a number of filters, which are designed to detect patterns in the image. The complexity of these filters tends to increase with additional convolutional layers, so initially, it could detect just straight lines within an image, but the latter layers may contain filters able to identify faces or animals for example. Another important concept is the max-pooling layer. This tends to succeed a convolution layer and its purpose is to reduce the dimensionality of the output from the convolution. As a result, computational costs fall due to this parameter reduction and it acts as a control for overfitting.

One advantage of using CNNs is that they are designed to function with little image pre-processing. Many image classification models require significant pre-processing of the images to achieve sufficient accuracy levels, however, due to the design, this step is considerably reduced with CNNs. Furthermore, there is a great deal of flexibility when designing the structure of the hidden layers in a CNN.

The main issue with CNNs is the computer power they require. For the more complex models, it can be very computationally expensive to train. Furthermore, compared to MLPs, there are considerably more hyperparameters that require optimising. Additionally, in order to successfully train a CNN, a large amount of data must be present.

4 Hypothesis Statement

The hypothesis statement is that the CNN will outperform the MLP. This is based on [6] written by LeCun et al., which tested various models' error rate on the MNIST database. Their results demonstrated a clear difference in performance levels. The average error rate for the MLP models was 3.2%, compared to the average error rate of 1.1% for the CNN models tested. This is a significant difference, and therefore we anticipate the CNN will have a superior performance.

5 Training and Evaluation Methodology

As comparisons are to be made between the two neural network models, it is vital to set aside a subset of the data in order to assess which model's performance is greater. Due to the structure of the MNIST database, 10,000 images have already been determined as testing labels, and therefore this subset will be held out for the final testing. The 60,000 images, classed as 'training' in the MNIST database will be used for training and validation.

For validation of each model on an individual basis, the holdout method was used to randomly sample 10% of the training images without replacement. Therefore, this left 54,000 images for the sole purpose of training.

An important step in this process is tuning the hyperparameters, as neural networks can be very sensitive to changes in hyperparameter values [7] This ensures the models are in an optimal position to solve our classification problem. To do this, a grid search coupled with 10 fold cross-validation will be adopted. Despite being computationally expensive, the advantage of this method is that it reduces the danger of overfitting the model, and in addition, accurate estimates of the test performance are obtained for each grid search. To assess which hyperparameters are optimal, the average accuracy level across each cross validation will be taken, with the highest accuracy resulting in those hyperparameters' selection.

With the optimal hyperparameters found, the networks will be re-trained, however, with extended training time frames. To limit the duration of the grid search, the number of epochs will be restricted in order to produce sufficient estimates of which combination of hyperparameters give the best performance, but ensure the grid search is completed in timely fashion. Once re-trained, the two networks will be tested on their individual held out datasets, before producing predictions for the MNIST testing set. For comparison of the two networks, confusion matrices will be constructed for each model and accuracy computed.

6 Architecture of the Neural Networks

Multi-Layer Perceptron

The MLP model will consist of four layers - the input, two hidden layers, and one output layer. The input layer will contain 784 neurons representing each pixel in an image, and 10 neurons in the output layer for each of the 10 labels. Due to the multi-class nature of our problem, it is essential to have a softmax output function. This is commonly used with neural network classifiers and creates a probabilistic distribution over the classes, and the class with the maximum probability is selected. We have opted for the non-linear hyperbolic tangent as our activation function. The number of epochs was set to 10 to ensure sufficient time for the network to train, but maintaining efficiency. The number of batches for each iteration within an epoch was set to 50. This allows for faster training of the network.

To ensure the MLP produces the most accurate predictions possible, it is vital to identify which hyperparameters are optimal for solving the classification problem. Three hyperparameters are to be tuned: momentum, learning rate, and number of neurons within the layers. This combination affects, the likelihood of being trapped in local minima, the speed of which the network learns the weights through the backpropogation process, and finally, the overall performance of the model.

Convolutional Neural Network

Our CNN is structured as follows: the first layer is the initial convolution and square filters will be used of size 5x5 pixels. The next layer is defined as the Rectifier Activation Linear Unit (ReLU). This normalisation process reduces the computational cost by replacing all negative values created from the filtering with zero. A max pooling layer then shrinks the feature stacks by taking the maximum value within a 2x2 grid placed on each stack. This helps generalise the model. An additional convolution layer follows; this time square filters of size 3x3 will be used. The next two layers are a repetition of the ReLU and max-pooling layers, and result in the input values for the first fully connected layer. This is followed by one more ReLU layer before the last fully connected layer, which contains 10 neurons representing the 10 classes. In a similar way to our MLP, a softmax output function is required, and a classification layer evaluates the cross-entropy loss function to complete the CNN.

There are a number of hyperparameters that can be optimised within a CNN, however, due to limited

computer power, it was decided to select three. To make comparisons with the MLP, a grid search was conducted to find the optimal momentum and learning rate values, as well as the number of filters within the convolutional layers. The number of filters for a convolutional layer tends to increase with the depth of the network, so it was important to account for this. However, adding a fourth hyperparameter to the grid search would have been computationally difficult. Therefore, the number of filters for the second layer was inputted as a function of the first to avoid finding an additional hyperparameter. For efficiency, the maximum number of epochs was set to five.

7 Experimental Results

The grid search was conducted for 96 combinations of hyperparameters for each model with Figure 3 the results. For each fold of the cross validation, 48,600 images were used for training, followed by testing how successfully they classify the remaining 5,400 images with comparison to the true class labels. The accuracies were then averaged over the 10 iterations for each combination of the hyperparameters.

The MLP was far more sensitive to changes in momentum than the CNN, reaching its highest accuracy with momentum at 0.9. This was to be expected, since it is more likely for the network to locate a global minimum rather than a local one with higher momentum levels. Increasing the number of neurons within each layer led to an improvement initially, but this effect diminished with minimal differences between 15 and 20 neurons. The effect of increasing the learning rate was far greater with a low momentum value, however, the MLP became more insensitive to changes in the learning rate as momentum increased.

		CNN - No. of Filters				MLP - No. of Neurons			
Momentum	Learning Rate	12	15	18	21	5	10	15	20
0.25	0.01	0.1124	0.1124	0.1124	0.1124	0.3696	0.3462	0.2971	0.1918
	0.025	0.1124	0.1124	0.1404	0.1560	0.5189	0.4716	0.4330	0.4159
	0.05	0.9275	0.9329	0.9412	0.9438	0.5916	0.6328	0.6062	0.5427
	0.075	0.9607	0.9654	0.9699	0.9680	0.6302	0.6977	0.7032	0.6855
	0.1	0.9748	0.9761	0.9754	0.9782	0.6469	0.7359	0.7574	0.7334
	0.5	0.8012	0.7181	0.6307	0.8971	0.6808	0.7863	0.7443	0.7275
0.5	0.01	0.1124	0.1124	0.1124	0.1124	0.4167	0.4163	0.3960	0.2906
	0.025	0.6410	0.7816	0.8891	0.9065	0.5370	0.5713	0.4859	0.4282
	0.05	0.9590	0.9606	0.9686	0.9688	0.6060	0.6888	0.7061	0.6579
	0.075	0.9754	0.9772	0.9784	0.9798	0.6495	0.7413	0.7512	0.7387
	0.1	0.9804	0.9820	0.9813	0.9814	0.6649	0.7451	0.7956	0.7929
	0.5	0.5381	0.4565	0.6239	0.2815	0.6640	0.8795	0.7814	0.6566
0.75	0.01	0.1130	0.2022	0.2888	0.4962	0.5145	0.5143	0.4261	0.4053
	0.025	0.9586	0.9648	0.9679	0.9680	0.6490	0.6936	0.6857	0.6817
	0.05	0.9786	0.9806	0.9824	0.9828	0.6210	0.7505	0.7917	0.7775
	0.075	0.9832	0.9839	0.9849	0.9839	0.6417	0.7955	0.8550	0.8652
	0.1	0.9842	0.9846	0.9844	0.9851	0.6549	0.8505	0.8694	0.8841
	0.5	0.1929	0.3638	0.3641	0.3601	0.6337	0.8859	0.8021	0.7165
0.9	0.01	0.9578	0.9643	0.9661	0.9661	0.6166	0.6939	0.6773	0.6657
	0.025	0.9822	0.9831	0.9833	0.9843	0.6455	0.7764	0.7983	0.7997
	0.05	0.9848	0.9852	0.9850	0.9858	0.6825	0.8419	0.8848	0.8928
	0.075	0.9845	0.9846	0.9844	0.9860	0.6558	0.8779	0.8945	0.8965
	0.1	0.9821	0.9841	0.9842	0.9859	0.6470	0.8671	0.8953	0.8989
	0.5	0.1006	0.1014	0.1028	0.1072	0.6073	0.8121	0.7476	0.6295

Figure 3: Grid Search Results.

For the CNN, there was far less sensitivity across the three hyperparameters. The majority of the combinations produced accuracy levels above 95%, and only extremities yielded poor results. This consistency is likely to occur because of the complex nature of the layers in CNNs. Additionally, momentum and learning rate's potency is reduced in CNNs compared to the MLP, as they only affect the model in the fully connected layers. Increasing the number of filters in the convolutional layers was expected to have more of an effect, however, this was only really visible when the learning rate was sufficiently low. The reason for this is possibly due to the fact that 12 filters is enough for the CNN to operate successfully. One commonality both models shared is that they both experienced overfitting when the learning rate was 0.5.

As mentioned earlier, the epochs were previously limited in order to reduce the duration of the grid search. For training with these hyperparameters, it was therefore important to allow the networks to train for an extended period of time, allowing the models to reach close to their full potential given their design. Following this training, they both would be tested on firstly, each model's held out data for validation, and finally, the 10,000 images classed as 'testing' by MNIST.

8 Analysis

The two models' classification was successful for the majority of unseen images in both validation and testing datasets, with the results shown in Figure 4. However, as anticipated in our hypothesis statement, the CNN was clearly superior. This mirrored the findings of [6]. Accuracy values surpassing 98% are impressively precise and demonstrates the power CNNs can possess. Although the MLP was less successful at solving the classification problem, it still produced an accuracy level for the testing images of above 92%, also highlighting the strength of neural networks. Furthermore, only 20 neurons were present at each hidden layer, and there is potential for this to increase considerably with a far larger number of neurons. One of the more recent studies [8] into MLPs used six layers consisting of over 7,500 neurons, which contributed to an accuracy of 99.65%.

Model	Epochs	Training Time (secs)	Validation Acc.	Testing Acc.
CNN	10	64.3	98.62%	98.92%
MLP	500	59.5	92.47%	92.70%

Figure 4: Testing Results.

However, it is unsurprising that the CNN's performance was superior here. As mentioned previously, CNNs have long been used for computer vision tasks because of their ability to analyse images so successfully, and this is seen here. This is predominantly due to the convolutional operations that are applied to the images within the network, and the lack of these layers in the MLP result in the difference of prediction quality.

There is one area where the CNN struggled to match the MLP. Only 10 epochs were used in training the CNN, compared to 500 epochs for the MLP. Despite this, training time was longer for the CNN. This highlights one of the main problems with CNNs. Although they are incredibly powerful, the trade off is the computer power required to train them. The example here is a relatively simple CNN, but if the complexity of the network was increased, the training time would grow exponentially.

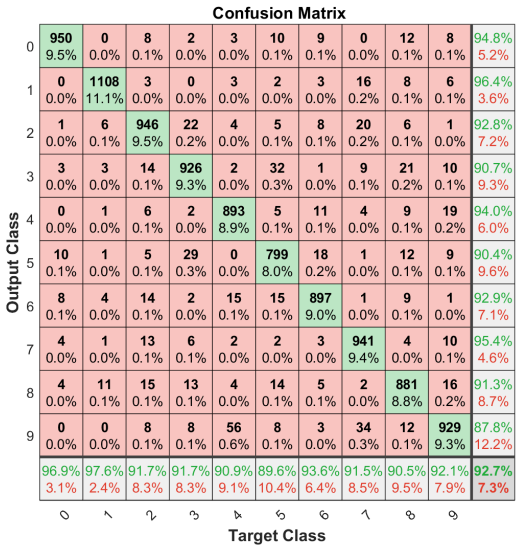

Figure 5: MLP Confusion Matrix.

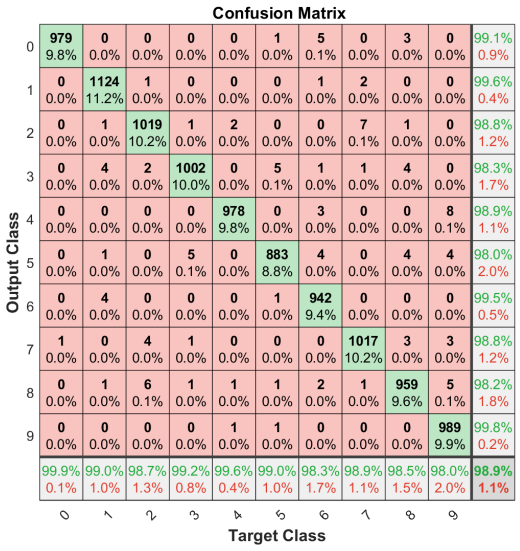

Figure 6: CNN Confusion Matrix.

The confusion matrices in Figures 5 and 6 present how accurate the models were on a label-by-label basis. Given the high accuracy of the CNN, there was little fluctuation between the digits, however, variance was greater for the MLP. The lack of variation in CNN's label-by-label accuracies can be explained by CNN's ability to accept whole images as inputs into the network before extracting features on the image in its entirety.

For the MLP, where variance is more prominent, it is worth analysing the confusion matrix further. Straight-edged digits appeared to have more success with classes $1, 4, 7$ producing high accuracies. Numbers with more curvature struggled to match them with 9 being the only digit with less than 90% accuracy. The reason behind this occurrence is likely to be due to the initial vectorisation of the images. With the number

1, the majority of the digit's shape is maintained within a vector, as its form is essentially a straight line, whereas a more curved digit is likely to appear more randomised within a vector.

9 Conclusion

Overall, the main conclusion of the study were that CNNs are a more powerful learning tool. This can be drawn due to the overwhelming superiority of the CNN results compared to the MLP on the MNIST database. However, it is certainly worth highlighting that working with images favours the CNN considerably. MLPs are very versatile when it comes to solving problems, whereas the applications of CNNs are more restricted.

Limitations affected both models. Tuning the hyperparameters was a particularly arduous task, taking well over 10 hours to complete. Due to this longevity, several hyperparameters remained at a default value and had they been tuned, which could have boosted performance. To maintain consistency and comparability, momentum and learning rates were tuned for both models. However, it would have been worth tinkering with the parameters within the convolution layers, such as the filter size, to view the effect. Visually, it would also have been pleasing to display the features stacks within the CNN, however, spatial restrictions prevented this.

In the future, it would be interesting to work with different models. There is an incredible flexibility when it comes to designing the architecture of neural networks, and investigating how variation in the structure affects results would be an intriguing task. Furthermore, while MNIST was suitable for this study, working with a more varied dataset, such as Caltech 101, would no doubt provide more excitement.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," vol. 521, pp. 436–44, 05 2015.

[2] A. Michael, "Neural networks and deep learning," 2015.

[3] K. Abhishek, M. Singh, S. Ghosh, and A. Anand, "Weather forecasting model using artificial neural network," *Procedia Technology*, vol. 4, pp. 311 – 318, 2012. 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[4] S. A. Mojarad, S. S. Dlay, W. L. Woo, and G. V. Sherbet, "Cross validation evaluation for breast cancer prediction using multilayer perceptron neural networks," *American Journal of Engineering and Applied Sciences*, vol. 4, no. 4, 2011.

[5] A. H. Moghaddam, M. H. Moghaddam, and M. Esfandyari, "Stock market index prediction using artificial neural network," *Journal of Economics, Finance and Administrative Science*, vol. 21, no. 41, pp. 89 – 93, 2016.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pp. 3460–3468, AAAI Press, 2015.

[8] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *CoRR*, vol. abs/1003.0358, 2010.

[9] F. Giannini, V. Laveglia, A. Rossi, D. Zanca, and A. Zugarini, "Neural networks for beginners. a fast implementation in matlab, torch, tensorflow," 2017.

[10] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 689–692, ACM, 2015.