A Comparative Study in Digit Recognition Using Multilayer Perceptrons and Convolutional Neural Networks

Daniel Hammocks, Tim Laibacher
City, University of London

November 7, 2018

1 Introduction

Handwritten digit recognition (HDR) is the rudimentary principal behind today's generalised optical character recognition. HDR has been applied throughout numerous domains from the sorting of mail [1], interpreting numerically rated survey responses and reading student candidate numbers on examination scripts [2].

Whilst HDR has already been extensively analysed, handwriting quirks transition over time [3]. This means a constant ongoing analysis is required to ensure accuracy rates in HDR keep pace with a technologically demanding world.

This paper aims to critically evaluate two supervised learning techniques, Multi-Layer Perceptrons and Convolutional Neural Nets, to classify an image into one of the ten numerical digits. We will conduct a systematic hyper-parameter grid search on both forms of neural networks in order to ascertain our leading algorithm.

The remainder of this paper is structured as follows: Section 1 outlines the two chosen supervised learning techniques, Section 2 discloses the finer points of our dataset, Section 3 illustrates the methods used when implementing our algorithms, Section 4 evaluates our results and provides a detailed comparison, and Section 5 outlines our significant concluding points and objectives for future work.

1.1 Multi-Layer Perceptron

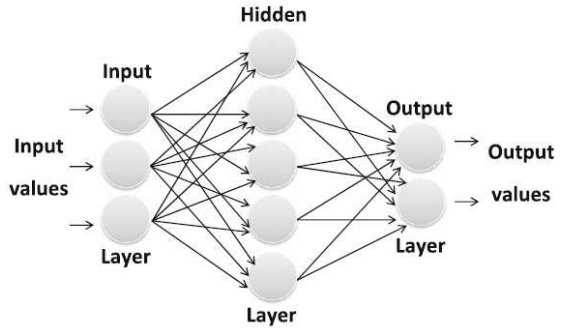

Figure 1: Overview of MLP Structure [4]

A multi-layer perceptron (MLP) is a type of feed-forward artificial neural network (ANN). The MLP consists of three types layers; an input layer, a hidden layer (of which multiple may exist) and an output layer. Shown in Figure 1.
Advantages of MLP's include the ability to learn non-linear and complex relationships; as even a single layer MLP has been proven to successfully model any mathematical function. MLP's also require less formal statistical training than alternative machine learning models [5]. However, an underlying issue with MLP's is the lack of probabilistic foundations, making determining confidence in a classifier difficult.

1.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks can be seen as a special kind of MLP and are a commonly used architecture for computer vision tasks. Originally proposed by Lecun, Bottou, Bengio, *et al.* [6], they have outperformed all previously existing techniques on popular image datasets such as ImageNet and CIFAR10 [7]. A basic CNN for an image classification task, usually consists of an input layer that takes tensors of size $C \times H \times W$ as input, where $C =$number of channels, $H =$height, $W =$width, a middle block that includes successive convolutional layers followed by max pooling layers, and one or more fully connected layers. A convolutional layer consists of D convolutions (aka convolutional layer neurons) where a kernel of size $C \times h \times w$ is applied to the input tensor and consequently outputs D feature maps of size $(H - h + 1) \times (W - w + 1)$ as depicted in Figure 2. Note that the same kernel per channel is swiped across the input rows and columns, this constraint is also referred to as weight-sharing. A max-pooling layer reduces (down-samples) the higher dimensional input. Figure 3 illustrates a 2×2 max pool operations applied on a 4×4 input. A fully connected layer is connected to all neurons in the previous layer.

One of the main advantages of CNN in comparison to MLP is the reduction in the number of free parameters through the aforementioned weight-sharing. Reduced parameters in turn reduce the computational costs and improve the generalization ability [6]. Weight-sharing also enables parallelization to a larger degree than it is the case for MLP. Another advantage is it's superior performance on image classification tasks compared to MLP.

A disadvantage that both CNN and MLP have in common, is that they do not encode the position and orientation of the objects in the image. A recent paper by Sabour, Frosst, and Hinton [8] tries to address this issue by introducing a novel capsule architecture.

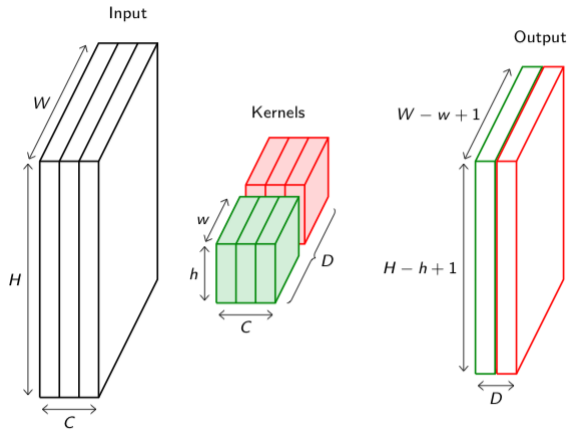

Figure 2: Illustration of a convolutional layer, from [9]

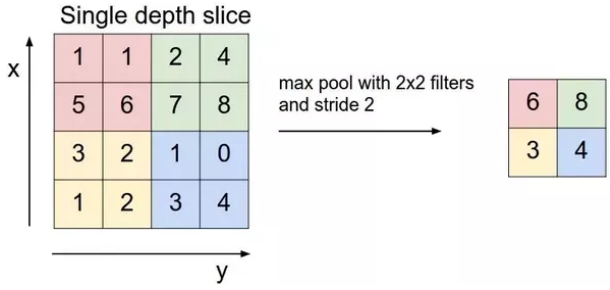

Figure 3: Illustration of a max pooling layer, from [10]

2 Dataset

The data used for our experiments is the MNIST dataset [6]. This is a large dataset consisting of 70,000 images of individually handwritten digits sampled from American Census Bureau employees and American high school students; 25 of which can be seen in Figure 4. The digits were then normalised to fit in 28x28 pixel image and anti-aliased; with the consequence of introducing greyscale. The format of our data is a 784x70000 matrix; each column represents a digit and every 28 rows a column of pixel values in our image. We also had a 1x70000 vector of ground truth labels corresponding to each image.

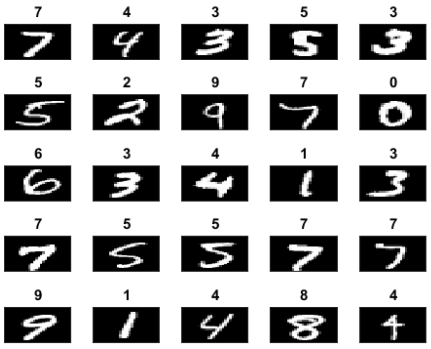

Figure 4: Greyscale Display of 25 Image Files and Their Associated Ground Truths

2.1 Initial Data Analysis

Initial analysis of the data shows that we have an unequal representation of each digit (see Figure 5), this can result in class bias. One solution to address imbalanced classes is to reduce the number of samples from each class to equal the number of samples from the least sampled class. We also calculated heat-maps, Figure 6, displaying the mean pixel value for each digit to identify if certain groups of pixel values were strongly associated to specific digits.

3 Methods

This section outlines our choice of training methodology, model architectures and which hyper-parameters will be optimized in the model selection step. Given the discussion in Section 1, our hypothesis is that the best convolutional neural network will outperform the best Multi-Layer Perceptron model in terms of accuracy.

3.1 Methodology

The dataset was split into a training set, consisting of 50,000 images, a validation set consisting of 10,000 images and a hold-out test set consisting of 10,000 images. The 10,000 test set images are predefined by the authors of the dataset to enable performance comparison amongst different techniques.

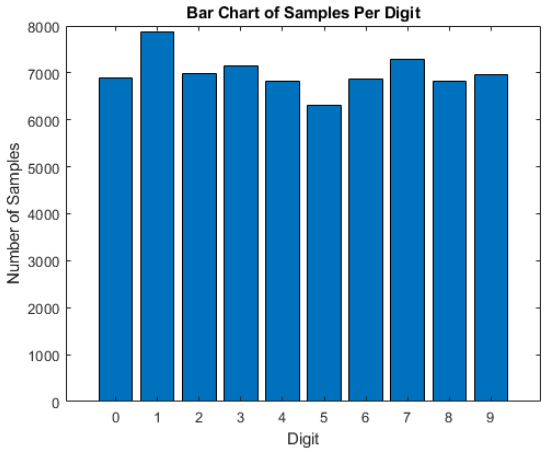

Figure 5: Bar Chart Displaying The Number of Samples Per Digit

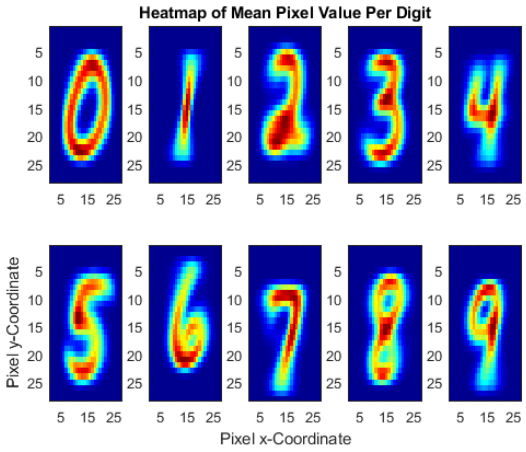

Figure 6: Heat-map Of Mean Pixel Values For Each Digit

3.2 Hyper-Parameters and Architecture Used for MLP

In order to efficiently compare a considerable number of MLPs we performed a grid search on the following hyper-parameters; the learning rate, $LR = \{0.001, 0.01, 0.1\}$, the number of hidden layers, $HL = \{1, 2, 3, 4\}$, and the number of neurons in each hidden layer, $NN = \{10, 50, 100, 250, 500\}$. We chose to alter these parameters with the objective of improving generalisability and performance whilst considering the complexity of the network.

Originally the MLP grid search was completed using gradient descent with momentum (GDM) as the training function but we soon identified a significant variation between the accuracy of our preliminary results and the results of the grid search. On average the accuracy of networks trained with GDM compared to scaled conjugate gradient backpropagation (SCG) was 17.5% and 94.9% respectively. It was therefore decided that we would use SCG as our training function.

During training, the initial weights of the MLPs are randomly assigned, the cross-entropy error of the network is then calculated, SCG proceeds to adjust the weights (a form of backpropagation) and recalculate the error. This process was set to repeat for a maximum of 1000 epochs, or until the validation accuracy had not improved for 6 validations.

3.3 Hyper-Parameters and Architecture Used for CNN

In order to avoid high computational costs and over complexity that can arise with very deep networks and larger layers, we decided to keep the architecture relatively simple. We took inspiration from the VGG architecture introduced by Simonyan and Zisserman [11], where after each block the number of convolutions is increased by a factor of two, while at the same time the spatial resolution is reduced by half through a 2×2 max-pooling operation, creating a "bipyramidal" effect. The resulting first hyper-parameter, referred to as NN in Tables 1-4, determines therefore the number of 3×3 convolutions in the first convolution layer. The second hyper-parameter is the learning rate. Stride and paddings were fixed to 1 for all convolutional layers, similarly the max-pooling layers were set to use a stride of 2.

We conducted a grid search for three model architectures:

1. Three middle blocks, each consisting of a convolutional layer, a ReLu and max pool layer, and one final fully connected layer of size 10 (Table 1).

2. Four middle blocks, each consisting of a convolutional layer, a ReLu and max pool layer, and one final fully connected layer of size 10 (Table 2).

3. Three middle blocks, each consisting of a convolutional layer, a ReLu and max pool layer, and two final fully connected layer of size 100 and 10 (Table 3).

The following sets of hyperparameter were evaluated: $NN = \{8, 16, 24, 32\}$ and $LR = \{0.1, 0.01, 0.001\}$. All models were trained with Stochastic Gradient Descent with Momentum. The momentum and mini-batch size were not part of the grid search and were set to 0.08 and and 128 respectively. Validation accuracy was evaluated every 50 iterations and the training process was stopped if the validation accuracy has not improved for at least 5 validations.

4 Results, Findings and Evaluation

CNN model training was conducted on the "seaford" gpu server instance, equipped with a NVIDIA Quadro K2000, of City, University of London. The combined training time was approximately 40 minutes. Grid search and training

Table 1: Grid search for CNN with 3 convolutional layer and one fully connected layer

LR	NN	Iter.	Epochs	Time	Batch Acc.%	Val. Acc.%
0.1	8	900	3	30.58	96.88	97.98
0.1	16	1900	6	58.60	99.22	98.29
0.1	24	1400	4	59.15	97.66	97.53
0.1	32	2200	7	120.53	97.66	97.90
0.01	8	1850	6	61.65	99.22	98.66
0.01	16	1050	3	32.88	98.44	**98.84**
0.01	24	1050	3	44.86	98.44	**98.84**
0.01	32	700	2	37.92	100.00	98.67
0.001	8	2250	7	73.45	97.66	98.41
0.001	16	2700	8	83.21	98.44	98.75
0.001	24	1550	5	65.53	98.44	98.62
0.001	32	2150	7	115.36	100.00	98.67

Table 2: Grid search for CNN with 4 convolutional layer and one fully connected layer

LR	NN	Iter.	Epochs	Time	Batch Acc.%	Val. Acc.%
0.1	8	850	3	31.44	100.00	98.71
0.1	16	1000	3	38.62	99.22	98.66
0.1	24	1600	5	86.34	100.00	98.83
0.1	32	1950	6	136.30	99.22	98.72
0.01	8	1450	5	42.78	99.22	98.94
0.01	16	1500	5	47.89	100.00	**99.09**
0.01	24	850	3	32.10	99.22	98.89
0.01	32	1200	4	66.24	100.00	**99.09**
0.001	8	1600	5	47.47	98.44	98.70
0.001	16	1850	6	68.55	100.00	98.93
0.001	24	1450	5	89.56	99.22	98.89
0.001	32	2200	7	152.02	99.22	99.01

Table 3: Grid search for CNN with 3 convolutional layer and two fully connected layer

LR	NN	Iter.	Epochs	Time	Batch Acc.%	Val. Acc.%
0.1	8	950	3	31.33	98.44	98.72
0.1	16	650	2	21.98	99.22	98.61
0.1	24	700	2	31.11	99.22	98.53
0.1	32	700	2	38.80	96.88	98.72
0.01	8	1450	4	47.68	99.22	98.70
0.01	16	1600	5	54.44	100.00	98.88
0.01	24	1150	4	61.56	100.00	**98.94**
0.01	32	1350	4	73.91	99.22	**98.98**
0.001	8	2000	6	64.98	99.22	98.00
0.001	16	3000	9	99.52	98.44	98.61
0.001	24	2000	6	87.53	100.00	98.45
0.001	32	2250	7	127.00	99.22	98.66

Table 4: Test-set results of selected CNN architectures

Conv. Layer	FC Layers	NN	LR	Test Acc.
3	1	24	0.01	98.96
3	1	16	0.01	99.06
4	1	32	0.01	99.17
4	1	16	0.01	**99.21**
3	2	24	0.01	99.01
3	2	32	0.01	99.03

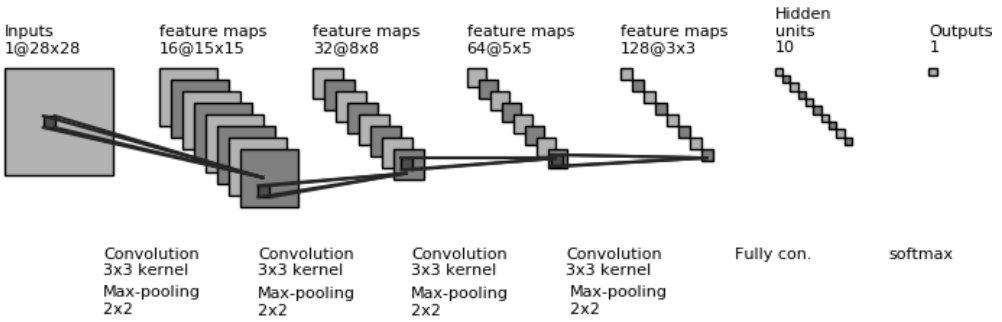

Figure 7: Best performing CNN architecture.

for the evaluated MLP models was conducted on a 16 worker parallel pool (AMD Ryzen 7 1700x CPU) and took approximately 10 hours.

4.1 Model Selection

4.1.1 CNN

Overall all tested CNN variations performed very well with validation accuracies above 97.53% across the board. An apparent sweet-spot for the learning rate hyper-parameter was found to be 0.01. The ideal number of feature maps in the first layer (NN) varied depending on the architecture. The best validation accuracies for model architecture 1 resulted from $NN = 16$ and $NN = 32$, for model architecture 2 from $NN = 16$ and $NN = 32$ and for model architecture 3 from $NN = 24$ and $NN = 32$. The two best performing configurations on the training-set were then evaluated on the hold-out test-set. The best performing architecture is model architecture 2 with $NN = 16$ and $LR = 0.01$, resulting in a test-set accuracy of 99.21% as shown in Table 4. The winning CNN architecture is displayed in detail in Figure 7. Given the relative simplicity of our architecture, the results are surprisingly good, the current state-of-the art result on this dataset is 99.75 % [8].

4.1.2 MLP

The results of our MLP grid search reveal a selection of trends amongst our hyperparameters. Firstly, it can be seen from Figure 10 that as the number of neurons increase the accuracy drastically increases but the accuracy

then begins to plateau after 100 neurons per layer. When we compare the learning rate to the number of neurons (Figure 10) we see a very weak correlation in the accuracy increasing as the learning rate decreases. Further comparing the learning rate with the number of hidden layers makes inferring a general trend difficult as results are sporadic. Finally, the increase from one hidden layer to two hidden layers (Figure 11) accounts for the greatest accuracy increase but the accuracy does continue to increase as we increase the number of hidden layers albeit at a slower rate.

Considering the results of our best performing MLP architectures (Figure 9), although it is clear in terms of accuracy which network performs best, it is important to take a holistic approach to model selection. For a 0.03% decrease in accuracy we can decrease our training time by a factor of five whilst simplifying our model by halving the NN, decreasing the number of HL by 1 and increasing the learning rate.

Figure 8: Top 20 MLP Grid Search Results

NN	HL	LR	Time (mins)	Epochs	Train. Acc.	Val. Acc.
500	3	0.1	56	165	0.9962	0.9768
500	3	0.001	54	161	0.9980	0.9759
250	4	0.001	10	140	0.9942	0.9744
250	2	0.001	20	104	0.9925	0.9744
500	3	0.01	49	145	0.9953	0.9740
250	2	0.1	10	104	0.9926	0.9733
250	4	0.01	17	117	0.9970	0.9731
500	4	0.001	71	159	0.9929	0.9729
250	3	0.001	13	113	0.9886	0.9723
500	2	0.1	9	121	0.9920	0.9722
500	1	0.1	16	130	0.9990	0.9722
250	2	0.01	28	95	0.9902	0.9722
500	1	0.01	14	114	0.9973	0.9721
500	1	0.001	15	123	0.9968	0.9718
250	1	0.001	6	98	0.9934	0.9717
250	1	0.01	7	101	0.9940	0.9712
250	4	0.1	18	129	0.9921	0.9710
100	3	0.01	3	96	0.9894	0.9708
250	1	0.1	3	109	0.9946	0.9707
100	2	0.1	7	86	0.9873	0.9707

Figure 9: Test-set Results of Selected MLP Architectures

NN	HL	LR	Time (mins)	Epochs	Test Acc.
500	3	0.001	54	161	0.9753
250	**2**	**0.1**	**10**	**104**	**0.9750**
500	1	0.001	15	123	0.9745
250	1	0.1	7	109	0.9736
500	3	0.1	56	165	0.9735
500	3	0.01	49	145	0.9734

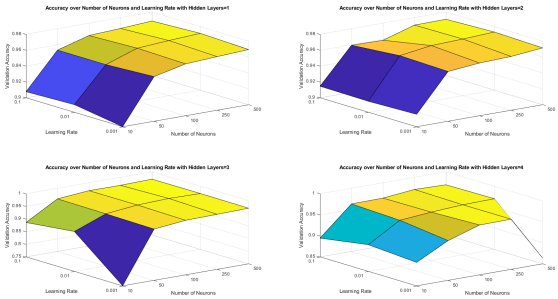

Figure 10: MLP NN and LR Versus Accuracy

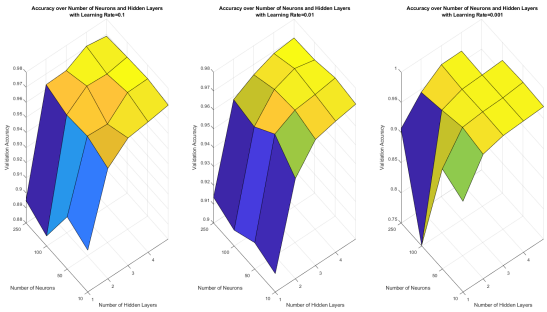

Figure 11: MLP NN and HL Versus Accuracy

4.2 Comparison of Algorithms

Both our MLP and CNN models performed well with test-set accuracies of 97.53% and 99.21% respectively. Interestingly, the worst performing convolution architecture on the validation set, with 3 convolutional layers, 24 feature maps in the first layer, a learning rate of 0.1 and a validation accuracy of 97.53% , is still better than 18 out of the top 20 MLP architectures evaluated on the validation dataset.

"Going deeper", that is increasing the number of hidden layers, resulted in an increased test-set performance for the evaluated CNN and MLP configurations. As outlined in Section 4.1.2, an increase in the number of neurons per layer led to an increase in validation accuracy, with four out of the six MLP architectures evaluated on the test-set having the highest number of neurons per layer that was considered during the grid search. The same can not be said for the CNN models, as both architecture variation 1 and 2 achieved the best test-set results with the lowest number of neurons in the first convolutional layer considered during the grid search (16).
Both algorithms were found to be sensitive to the learning rate. Whereas for CNN an optimal learning rate of 0.01 was found for all three tested architectures, for MLP the optimal learning rate varied depending on the number of hidden layers and neurons per layer.

When scrutinising false negatives in the MLP model we notice a recurring 'pairing' pattern between different digits. This can be interpreted from the confusion matrix (Figure 12). This pairing pattern is the result of a target class being identified as another class and vice versa; such pairs that exist include (3,5) and (4,9). This phenomena was anticipated as when comparing the heat-maps (Figure 6) we can see significant correlations between the hotspots in these digits. When we compare this observation with the results from the CNN (Figure 13) we note that the (3,5) pairing is less prominent, illustrating the CNN architectures superior feature selection techniques. Like for the MLP, the (4,9) pair is also the most commonly misclassified pair for the CNN, albeit to a far lesser degree with only 6 misclassification's vs 46 misclassification's for the MLP model.

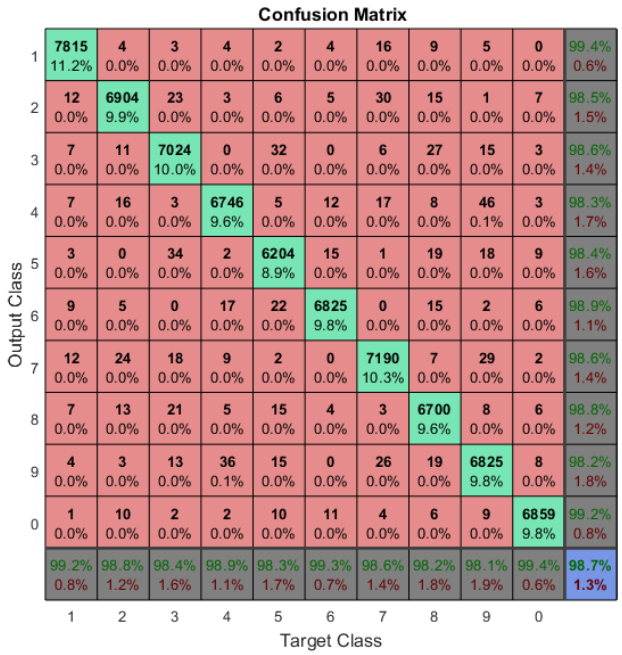

Figure 12: Confusion Matrix of MLP

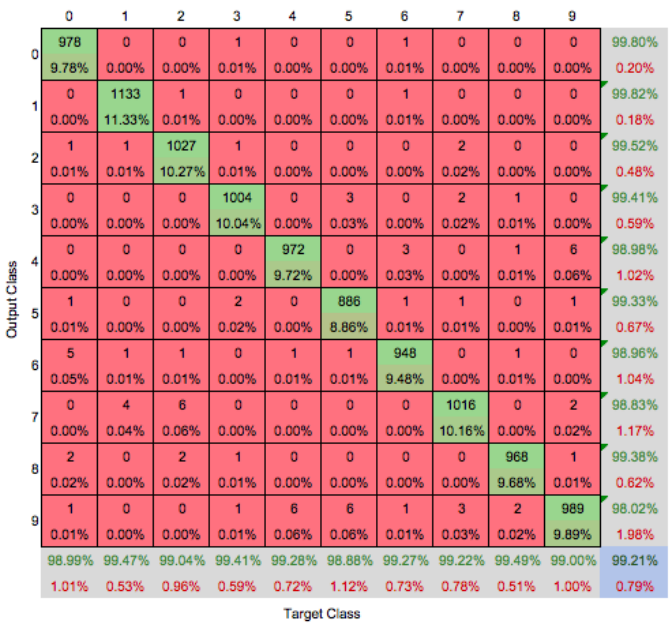

Figure 13: Confusion Matrix of CNN

5 Conclusion

Our study successfully compares MLP and CNN architectures for the purpose of digit recognition, with both reporting a high degree of accuracy. It is clear from the comparisons exhibited that the CNN is the superior of the two architectures for digit recognition. The reduction in the number of weights to train through weight-sharing in CNN, further decreases their computational cost and improves their generalization ability. While our results confirm the latter, we are not able to confirm the former statement since training was conducted on different hardware. We therefore suggest running the training and evaluation on identical machines in future work. Another important finding is that an exceedingly more complex MLP is required to mimic the accuracy rates of a much simpler CNN.

We have also exemplified how confusion matrices can be an insightful resource for multi-class classification problems as they provide an intuitive visual for identifying any erroneous classifications.

Whilst our study was successful and highlighted some important trends and results, there is still additional work that could be pursued. Our study was focused on comparing the accuracy of the two architecture types but in real life applications, of computer vision, time constraints are often a significant factor in model selection. Another comparison technique we wished to employ was the creation of a set of digits, containing noisy pixels based on our heat-map information, to identify which model performed better with a set of prefabricated digits. For the MLP we considered only a fixed number of fully connected layers with an identical number of neurons. We believe the MLP could be further refined by further investigating the optimal number of neurons and varying the number of neurons across the hidden layers. As previously briefly discussed, we noticed a radical variation when altering the training function of our MLP model and this is something that could be explored further. In regards to the CNN architectures we would be interested in experimenting with even deeper network architectures to explore if performance will eventually plateau or degenerate and at what point this might take place.

References

[1] Y. L. Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson, “Advances in Neural Information Processing Systems 2”, in, D. S. Touretzky, Ed., San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 396–404, ISBN: 1-55860-100-7. [Online]. Available: `http://dl.acm.org/citation.cfm?id=109230.109279`.

[2] N. Yoshida, K. Koyama, K. Ng, W. Tsukahara, and M. Nakagawa, “New Features for a Pen and Paper-based Exam Scripts Marking System”, in *Proceedings of E-Learn: WORLD Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2009*, T. Bastiaens, J. Dron, and C. Xin, Eds., Vancouver, Canada: Association for the Advancement of Computing in Education (AACE), Oct. 2009, pp. 3758–3765. [Online]. Available: `https://www.learntechlib.org/p/33026`.

[3] E. L. Grigorenko, E. Mambrino, and D. D. Preiss, *Writing: A Mosaic of New Perspectives*, en. Psychology Press, May 2012, Google-Books-ID: iJpuAFblpCoC, ISBN: 978-1-136-66891-3.

[4] G. Al-Naymat, M. Alkasassbeh, N. Abu-Samhadanh, and S. Sakr, “Classification of VoIP and non-VoIP traffic using machine learning approaches”, *Journal of Theoretical and Applied Information Technology*, vol. 3192, 2016.

[5] J. V. Tu, “Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes”, *Journal of Clinical Epidemiology*, vol. 49, no. 11, pp. 1225–1231, 1996, ISSN: 0895-4356. DOI: https://doi.org/10.1016/S0895-4356(96)00002-9. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895435696000029.

[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, Nov. 1998, ISSN: 0018-9219. DOI: 10.1109/5.726791.

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”, *ArXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[8] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic Routing Between Capsules”, *ArXiv:1710.09829 [cs]*, Oct. 2017, arXiv: 1710.09829. [Online]. Available: http://arxiv.org/abs/1710.09829.

[9] Francais Fleuret, *Deep Learning Course*. [Online]. Available: https://documents.epfl.ch/users/f/fl/fleuret/www/dlc/.

[10] *Convolutional neural network*, en, Page Version ID: 827834032, Feb. 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=827834032.

[11] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *ArXiv:1409.1556 [cs]*, Sep. 2014, arXiv: 1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556.