A Comparative Study of Image Classification Models on MNIST Dataset

Elias Michael
elias.michael@city.ac.uk

1. Introduction

Image classification is a foundational task in computer vision, with wide-ranging applications across multiple domains. In healthcare, it enables early disease detection by analysing medical scans such as X-rays and MRIs, thus improving diagnostic accuracy and patient outcomes [1]. In autonomous vehicles, image classification supports navigation by recognising pedestrians, traffic signs, and road obstacles [2]. Furthermore, in security systems, facial recognition technologies assist in surveillance and access control. These examples highlight the critical role of image classification in advancing automation, safety, and decision-making.

This study presents a comparative evaluation of two neural network models applied to the task of handwritten digit recognition using the MNIST dataset [3]. While the task itself is well-explored and relatively simple, it serves as a benchmark for evaluating image classification performance. The models under consideration are the Multilayer Perceptron (MLP) and the Convolutional Neural Network (CNN). Multiple configurations are tested with the aim of developing a reliable model that performs consistently well on unseen data.

1.1 Multilayer Perceptron (MLP)

A Multilayer Perceptron is a class of feedforward artificial neural networks consisting of an input layer, one or more hidden layers, and an output layer [4]. Each neuron in the network applies a non-linear activation function, and the model is trained via forward and backward propagation, adjusting weights to minimise prediction error. While MLPs are powerful general-purpose classifiers, they are less efficient for image-related tasks due to their inability to exploit spatial structure within data.

1.2 Convolutional Neural Network (CNN)

A Convolutional Neural Network is specifically designed to process visual information by leveraging spatial hierarchies. It uses convolutional layers to extract local features such as edges and textures, and pooling layers to reduce dimensionality while retaining essential information [4]. Compared to MLPs, CNNs are more efficient and scalable for image classification, as they require fewer parameters and naturally preserve spatial relationships within the input.

1.3 Comparison & Hypothesis

Although both MLPs and CNNs are suitable for pattern recognition, CNNs are expected to outperform MLPs in image classification due to their architectural advantages. CNNs are better equipped to capture spatial hierarchies and are more computationally efficient. Therefore, we hypothesize that CNNs will achieve higher accuracy and generalisation performance on the MNIST classification task compared to MLPs.

2. Dataset

This study employs the MNIST (Modified National Institute of Standards and Technology) dataset, a widely used benchmark in the field of image classification. The dataset comprises greyscale images of handwritten digits ranging from 0 to 9 and is frequently used to evaluate machine learning and computer vision models. It was obtained from the corresponding Kaggle page [3] and loaded according to the provided instructions.

For this experiment, the dataset consists of two separate files: a training set containing 6,000 images and a testing set containing 1,000 images, resulting in a total of 7,000 samples. Each image is represented in greyscale with a resolution of 28×28 pixels, leading to a total of 784 features per image. Each pixel value ranges from 0 to 255, indicating the intensity of the pixel, where 0 is white and 255 is black.

An initial analysis of the training data was conducted by visualising the distribution of class labels using a bar chart. The results show a nearly uniform distribution across all ten-digit classes, suggesting that the dataset is balanced. Furthermore, the dataset contains no missing values, confirming its readiness for model training. To offer a more intuitive understanding of the data, six representative images along with their corresponding digit labels were also plotted.

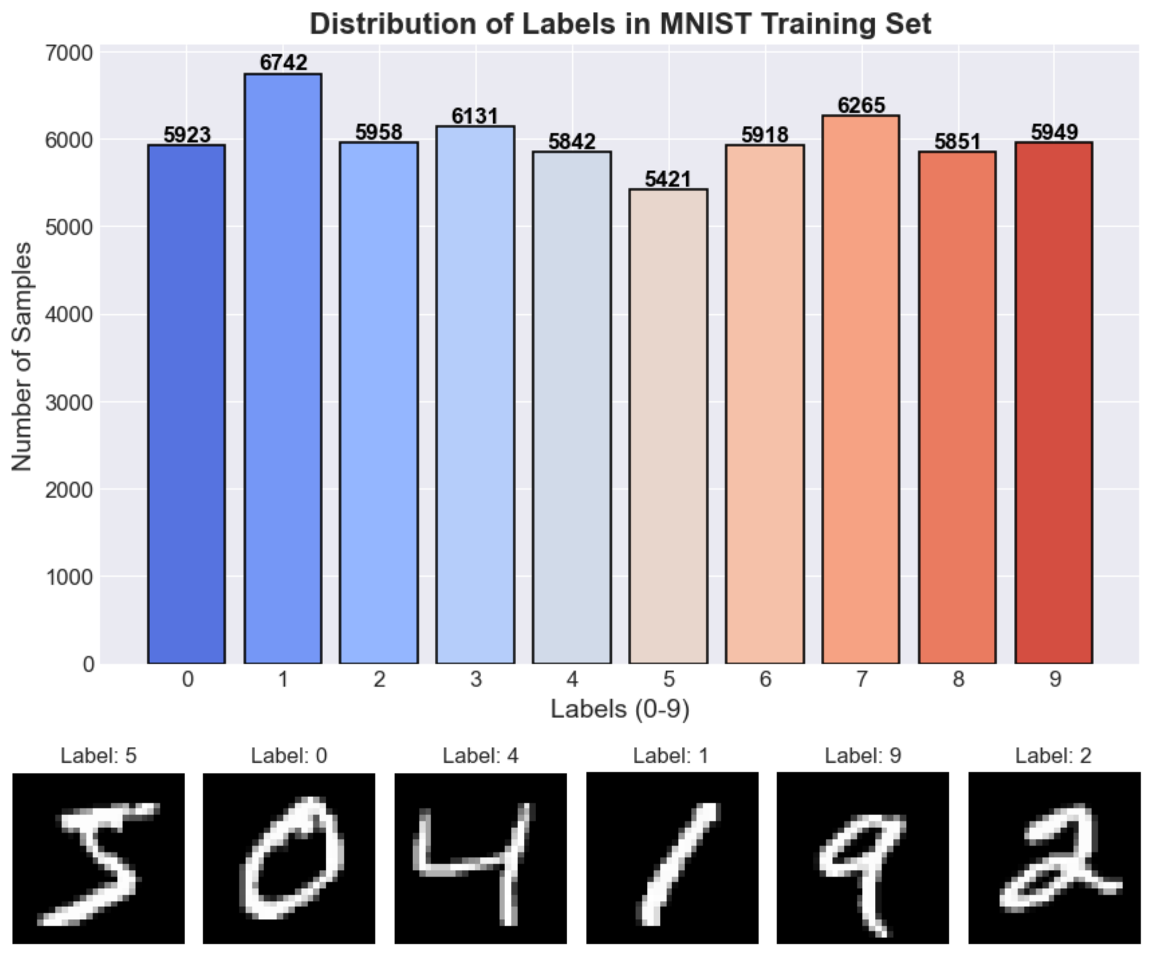


Figure 1: Distribution of digit classes in the training set and six randomly selected sample images with labels.

3. Methods

This section describes the training, validation, and testing procedures used to ensure a fair and reproducible comparison between the two models. It also details the architectural configurations and hyperparameter tuning strategies for both the Multilayer Perceptron (MLP) and the Convolutional Neural Network (CNN).

3.1 Methodology

The dataset was already partitioned into a training set of 6,000 images and a testing set of 1,000 images, corresponding to an approximate 86:14 split. The test set remained strictly reserved for final evaluation and was not used during training or validation.

Model selection for both MLP and CNN was performed using a grid search combined with 5-fold cross-validation [5]. A range of hyperparameters was explored for each architecture, and the average validation error across the folds was used to identify the configuration with the best generalisation performance.

Once optimal hyperparameters were determined, each model was retrained on the full training set using those settings. Although initial trials suggested that training for 10 epochs yielded good cross-validation results, this was based on subsets of the data. Therefore, we further experimented with different epoch values on the full training set to evaluate how dataset size influences the optimal number of training epochs. This allowed us to collect additional metrics on training time and performance, which are later used in evaluating the trade-off between model accuracy and computational efficiency.

3.2 Architecture and hyperparameters used for the MLP

The Multilayer Perceptron model treats each 28×28 greyscale image as a flattened 784-dimensional input vector. Several architectures were tested, including configurations with two hidden layers of varying sizes, such as [128, 64], [256, 128], and [512, 256]. Each hidden layer uses the ReLU activation function to introduce non-linearity. The final layer consists of 10 neurons with a SoftMax activation, suitable for multi-class classification.

The model is trained using the Adam optimiser and the CrossEntropyLoss function, both standard choices for classification problems. Hyperparameters such as learning rate, batch size, and hidden layer sizes were optimised through grid search. Despite their generality, MLPs do not exploit spatial relationships within image data, which can limit their performance on vision tasks.

3.3 Architecture and hyperparameters used for the CNN

The Convolutional Neural Network is tailored for visual data and processes the 2D image input without flattening it at the outset. It begins with convolutional layers designed to extract local features such as edges and textures using filters of varying sizes (e.g., 3×3 or 5×5) and depths (e.g., 16, 32, or 64 filters). Convolutional layers are followed by ReLU activations and max-pooling layers to reduce spatial dimensionality while preserving salient features.

After the convolutional blocks, the feature maps are flattened and passed through one or more fully connected layers, ultimately producing a 10-class SoftMax output. Like the MLP, the CNN is trained with the Adam optimiser and CrossEntropyLoss criterion. A 5-fold cross-validation strategy is used to validate the model's performance and select optimal hyperparameters. CNNs typically outperform MLPs in image classification due to their ability to preserve spatial hierarchies and reduce parameter counts while extracting informative features.

4. Results, Findings & Evaluation

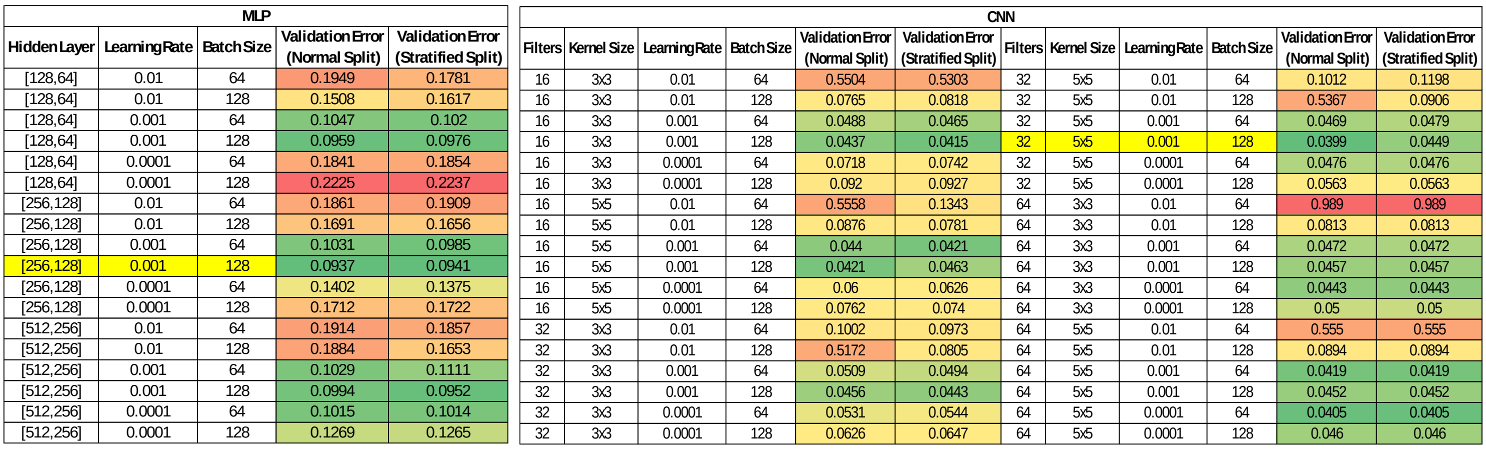
4.1 Model Selection

The grid search results for both models are summarised in Figure 2, illustrating performance across various combinations of hyperparameters. Due to the additional hyperparameter (kernel size or filter count) in the CNN architecture, the number of models evaluated was approximately twice that of the MLP.

In both architectures, performance was sensitive to the learning rate. For the MLP, model behaviour varied significantly across runs, primarily due to the random initialization of weights using PyTorch's Kaiming Uniform method (with zero biases). This sensitivity made it difficult to replicate exact accuracies on re-runs of the grid search.

CNN models exhibited even greater variability under large learning rates (e.g., 0.01), where training occasionally failed to converge due to overshooting. This instability suggested that parameter updates were too large, causing the model to skip over minima. To mitigate this, a stratified split was introduced in a subsequent grid search, improving balance across folds. While this improved overall stability, convergence issues were still observed in certain folds.

Validation error was colour-coded in the grid search table to facilitate pattern recognition and highlight the best-performing configurations. Models trained with a learning rate of 0.001 consistently achieved the lowest validation errors across both MLP and CNN architectures, suggesting that learning rate was the most influential hyperparameter. The best-performing models for each architecture were highlighted in yellow.



MLP				
Hidden Layer	Learning Rate	Batch Size	Validation Error (Normal Split)	Validation Error (Stratified Split)
[128,64]	0.01	64	0.1949	0.1781
[128,64]	0.01	128	0.1508	0.1617
[128,64]	0.001	64	0.1047	0.102
[128,64]	0.001	128	0.0959	0.0976
[128,64]	0.0001	64	0.1841	0.1854
[128,64]	0.0001	128	0.2225	0.2237
[256,128]	0.01	64	0.1861	0.1909
[256,128]	0.01	128	0.1691	0.1656
[256,128]	0.001	64	0.1031	0.0985
[256,128]	0.001	128	0.0937	0.0941
[256,128]	0.0001	64	0.1402	0.1375
[256,128]	0.0001	128	0.1712	0.1722
[512,256]	0.01	64	0.1914	0.1857
[512,256]	0.01	128	0.1884	0.1653
[512,256]	0.001	64	0.1029	0.1111
[512,256]	0.001	128	0.0994	0.0952
[512,256]	0.0001	64	0.1015	0.1014
[512,256]	0.0001	128	0.1269	0.1265

CNN											
Filters	Kernel Size	Learning Rate	Batch Size	Validation Error (Normal Split)	Validation Error (Stratified Split)	Filters	Kernel Size	Learning Rate	Batch Size	Validation Error (Normal Split)	Validation Error (Stratified Split)
16	3x3	0.01	64	0.5504	0.5303	32	5x5	0.01	64	0.1012	0.1198
16	3x3	0.01	128	0.0765	0.0818	32	5x5	0.01	128	0.5367	0.0906
16	3x3	0.001	64	0.0488	0.0465	32	5x5	0.001	64	0.0469	0.0479
16	3x3	0.001	128	0.0437	0.0415	32	5x5	0.001	128	0.0399	0.0449
16	3x3	0.0001	64	0.0718	0.0742	32	5x5	0.0001	64	0.0476	0.0476
16	3x3	0.0001	128	0.092	0.0927	32	5x5	0.0001	128	0.0563	0.0563
16	5x5	0.01	64	0.5558	0.1343	64	3x3	0.01	64	0.989	0.989
16	5x5	0.01	128	0.0876	0.0781	64	3x3	0.01	128	0.0813	0.0813
16	5x5	0.001	64	0.044	0.0421	64	3x3	0.001	64	0.0472	0.0472
16	5x5	0.001	128	0.0421	0.0463	64	3x3	0.001	128	0.0457	0.0457
16	5x5	0.0001	64	0.06	0.0626	64	3x3	0.0001	64	0.0443	0.0443
16	5x5	0.0001	128	0.0762	0.074	64	3x3	0.0001	128	0.05	0.05
32	3x3	0.01	64	0.1002	0.0973	64	5x5	0.01	64	0.555	0.555
32	3x3	0.01	128	0.5172	0.0805	64	5x5	0.01	128	0.0894	0.0894
32	3x3	0.001	64	0.0509	0.0494	64	5x5	0.001	64	0.0419	0.0419
32	3x3	0.001	128	0.0456	0.0443	64	5x5	0.001	128	0.0452	0.0452
32	3x3	0.0001	64	0.0531	0.0544	64	5x5	0.0001	64	0.0405	0.0405
32	3x3	0.0001	128	0.0626	0.0647	64	5x5	0.0001	128	0.046	0.046

Figure 2: Validation errors for hyperparameter grid search across all MLP and CNN configurations. Yellow highlights indicate the best-performing models.

4.2 Algorithm Comparison

The final selected models were evaluated on the held-out test set. Figure 3 presents the confusion matrices for both the MLP and CNN models, while Figure 4 shows their respective ROC curves with Area Under the Curve (AUC) for each class [7].

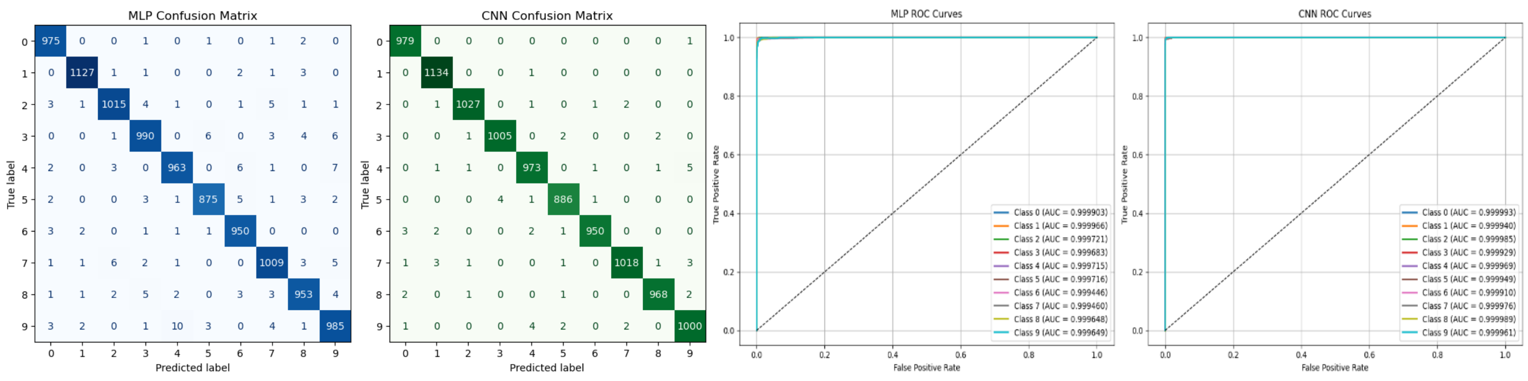


Figure 3 & 4: Confusion matrices (left) and class-wise ROC curves (right) for MLP and CNN models on test data.

Both models demonstrated strong performance, correctly classifying over 98% of the test samples across all classes. As expected, the CNN outperformed the MLP slightly, especially in distinguishing digits such as '0', where it achieved nearly perfect classification (99.9% accuracy). However, this improvement comes at the cost of increased computational complexity.

Initially, additional plots such as accuracy versus epochs and training time versus epochs were generated. However, these plots revealed expected trends without offering novel insights: the CNN consistently outperformed the MLP in accuracy (with differences around 0.3%–0.7%), and both models showed a near-linear increase in training time with epochs. Notably, the CNN had a steeper slope, taking approximately five to six times longer per epoch compared to the MLP.

A more informative comparison is shown in Figure 5, which plots accuracy versus training time for both models. This plot reveals that while the CNN requires more training time per epoch, it achieves higher accuracy than the MLP for comparable training durations. Furthermore, it offers scalability—allowing for additional accuracy improvements with longer training.

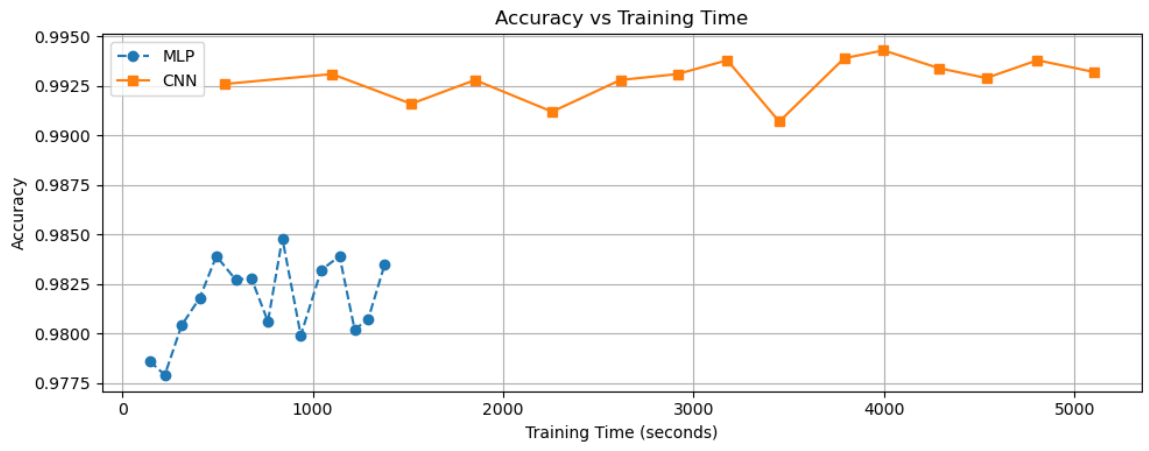


Figure 5: Accuracy versus training time for MLP (blue dots) and CNN (orange squares).

5. Conclusions

This study set out to compare the performance of a Multilayer Perceptron (MLP) and a Convolutional Neural Network (CNN) on the task of handwritten digit classification using the MNIST dataset. The results confirmed the initial hypothesis: the CNN model consistently outperformed the MLP in both accuracy and generalisation, albeit at a higher computational cost.

Several important insights emerged throughout the experimentation process. First, weight initialisation proved to be a critical factor in training stability and reproducibility, particularly for the MLP model. In this study, we adopted PyTorch's default Kaiming Uniform initialisation [6], which is known to perform well with ReLU activation functions. However, repeated runs of the grid search still produced varying results. While stratified splits introduced some stability, they did not fully resolve the inconsistencies. Future work could explore alternative combinations of initialisation strategies and activation functions—such as pairing Xavier (Glorot) initialisation with tanh or sigmoid activations—to assess their impact on convergence behaviour and performance consistency.

Additionally, the learning rate was identified as the most influential hyperparameter in both models. All best-performing configurations converged at a learning rate of 0.001, suggesting that a finer-grained exploration of this parameter could further optimise performance. Future studies may benefit from narrowing the search intervals or using adaptive learning rate strategies.

Overall, this comparison highlights the importance of architecture-specific tuning and supports the CNN model as a more robust solution for image classification tasks, especially when accuracy and spatial awareness are priorities.

6. References

[1] Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Ghafoorian, M., van der Laak, J.A.W.M., van Ginneken, B. and Sánchez, C.I. (2017). A Survey on Deep Learning in Medical Image Analysis. Medical Image Analysis, 42(1), pp.60–88. doi:https://doi.org/10.1016/j.media.2017.07.005.

[2] Ramakrishnan, P., K Dhanavel, K Deepak and R Dhinakaran (2023). Autonomous Vehicle Image Classification using Deep Learning. doi:https://doi.org/10.1109/icscds56580.2023.10104830.

[3] Khodabakhsh, H. (n.d.). MNIST Dataset. [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/hojjatk/mnist-dataset.

[4] Goodfellow, I., Bengio, Y. and Courville, A. (2016a). Deep Learning. Cambridge, Massachusetts: The MIT Press, pp.164–167, pp.326–339.

[5] Gorriz, J.M., Segovia, F., Ramirez, J., Ortiz, A. and Suckling, J. (2024). Is K-fold cross validation the best model selection method for Machine Learning? [online] arXiv.org. Available at: https://arxiv.org/abs/2401.16407.

[6] GeeksforGeeks. (2023). Kaiming Initialization in Deep Learning. [online] Available at: https://www.geeksforgeeks.org/kaiming-initialization-in-deep-learning/.

[7] Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, [online] 27(8), pp.861–874. doi:https://doi.org/10.1016/j.patrec.2005.10.010.

[8] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. [online] Available at: http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf.