University of Liverpool
Department of Computer Science

Applied Artificial Intelligence - COMP534
MLPs and CNNs

Noel Obinna Ogbuagu (201719039), Vasiliki Theofili Nikolaidi (201772172)

April 21, 2024

EMNIST Dataset & Classification Models

The EMNIST (Extended MNIST) dataset derives from the NIST Special Database 19, specifically from the MNIST database subset, and contains a set of handwritten letters and numbers converted to 28x28 pixels. The images are in greyscale, meaning they only contain 1 channel. Six different splits are provided for the dataset: ByClass, ByMerge, Balanced, Letters, Digits and MNIST [3]. For this assignment, the Balanced split has been used, which contains 47 classes that correspond to digits 0-9, all uppercase letters and lowercase letters except 'c', 'i', 'j', 'k', 'l', 'm', 'o', 'p', 's', 'u', 'v', 'w', 'x', 'y' and 'z'. These lowercase letters have been removed from the Balanced dataset to avoid classification errors caused by the misclassification of the lowercase letters with their uppercase counterparts [1].

The dataset was downloaded from Kaggle. Other options available were through the *Torchvision* or *emnist* libraries. Initially, we attempted downloading the dataset from *Torchvision.* However, the class is currently broken and will be fixed in the next released version of the library [2]. The training set contains 112799 images, and the test set contains 18799 images. The labels in the datasets are in the first column and mapped to their respective characters using ASCII. Figures 1(a) and 1(b) show 25 randomly picked images from the training and test set, respectively, along with their true labels.

Two deep learning models have been constructed for the classification of the EMNIST dataset: a Multilayer Perceptron (MLP) and a Convolutional Neural Network. Both these models were created using as parent the *nn.Module* class from the *torch.nn* library, and parameters such as activation function, dropout rate, and batch normalisation were given in the *__init__* method to allow for hyperparameter optimisation.

For the MLP modules, the number of fully connected layers is also given as a parameter *output_layer_sizes* by proving their sizes. In contrast, the input size is always 784 (28x28 pixels), representing the flattened images. If provided, batch normalisation has been applied to all layers, along with the activation function and dropout rate. For the assignment, 4 fully connected layers were implemented. The baseline architecture of the MLP model has been visualised with *Netron* and can be seen in Figure 2(a).

The CNN model follows a similar structure to the MLP model. The number of convolutional layers is given as a combination of parameters (channels and kernels). For assignment 2, convolutional layers were implemented, and two fully connected layers were used at the end to reduce the number of outputs to 47, equal to the number of classes in the Balanced Dataset. After applying the activation function on the convolutional layer, a pooling layer has been used to reduce the feature map's spatial dimensionality for the next layer. Max Pooling has been chosen as it is widely used in image processing to extract the features/values that are highest overall, thus making it more important for training. Dropout and Batch Normalisation layers are added conditionally based on the parameters *use_dropout* and *use_batchnorm*, which are boolean.The baseline architecture of the CNN model can be seen in Figure 2(b).

Hyperparameter Tuning

Various hyperparameters were considered to create low-loss and high-accuracy models. In Table 1, the tested parameters are presented. The tuning was performed using *GridSearhCV* with 5-fold cross-validation, scoring based on model accuracy, and using all available processors to run parallel jobs and reduce computational time. The total candidates for both models were 684, resulting in 3240 fits. For both hyperparameter tuning and training, ten (10) epochs were used and batches of 64 samples. In Tables 2 and 3, the computer specifications are used to optimise and train the models and the elapsed time for hyperparameter tuning and training the final models.

Parameter	**Tested Values**
activation function	ReLU, LeakyReLU, ELU
use_batchnorm	True, False
use_dropout	True, False
optmimiser	Adam, SGD, RMSprop
learning rate	.001, .01, .1
weight decay (L2 regularisation)	.001, .01, .1
exponential learning rate - gamma	.1, .5

Table 1: Parameters to tune for MLP and CNN models

Type	**Specification**
Device	Legion 5 Pro 16ITH6H
Processor (CPU)	11th Gen Intel(R) Core(TM) i7-11800H @ 2.30 GHz
CPU cores	8
Logical Processors	16
RAM	32.0 GB
System Type	64-bit operating system, x64-based processor
GPU	NVIDIA GeForce RTX 3070

Table 2: Computer Hardware Specifications

Model	**Tuning Time**	**Training Time**
MLP	495.36 mins (= 8.2.hours)	10.05 mins
CNN	1067.51 mins (= 17.8 hours)	12.57 mins

Table 3: Tuning and Training times for MLP and CNN models

Optimisers

Three optimisation methods were tested; Adam, SGD, and RMSprop. Adam optimiser was preferred due to its adaptive learning rate capabilities, making it well-suited for datasets with diverse features like EMNIST. SGD randomly selects samples, resulting in faster convergence over epochs, and introducing noise to the training process can help generalise the model better and reduce overfitting. RMSprop was tested to compare its performance in scenarios where adaptive learning rates are beneficial, but momentum may lead to overshooting, especially in smaller datasets. With a learning rate of 0.01, the Adam optimiser delivered the highest accuracy for both MLP and CNN models.

Activation Functions

The activation functions tested were ReLU, LeakyReLU, and ELU. ReLU was selected for its computational efficiency and ability to reduce the likelihood of vanishing gradients during deep network training. LeakyReLU, a variant of ReLU, was chosen to address some of its limitations, such as the problem of 'dying neurons'. Lastly, ELU was included to allow negative outputs and introduce a level of robustness, which can accelerate learning by helping to converge costs to zero faster and achieve higher classification accuracy. For MLP, ReLU performed the best, while ELU achieved higher overall accuracy in tandem with other parameters for the CNN model.

Adaptive Learning Rate Schedulers

Only the Exponential Learning Rate Scheduler was implemented for the adaptive learning rate with gamma (γ) values of 0.1 and 0.5. The restriction to one scheduler was due to the excessive computational time that would be taken to test more candidate combinations. This scheduler was chosen as it reduces the learning rate by a factor of γ for each epoch. As the epochs progress, the decay becomes the lowest among other schedulers, which helps stabilise the model's convergence by gradually reducing the learning rate and avoiding the risk of overshooting minimal points during training. Both MLP and CNN proved to perform better when using $\gamma = 0.5$.

Batch Normalisation

The impact of batch normalisation was investigated by comparing model performances with and without this technique. Batch normalisation normalises the input layer by adjusting and scaling activations, which has been shown to accelerate neural network training and increase overall accuracy by reducing internal covariate shifts. After tuning, both models used batch normalisation to achieve high prediction accuracy.

Dropout

Dropout was implemented to prevent overfitting, and models were tested with and without this technique. By randomly dropping units (along with their connections) during training, dropout forces the network to not rely heavily on any specific feature, thus enhancing the generalisation capabilities of the model. Initially, dropout was given with a fixed probability of 0.5, as recommended by literature on the optimal choice of dropout rate for Neural Networks [5]. After tuning, both models achieved the best results without dropout.

L1 & L2 normalisation

L1, or Lasso, regularisation was not separately explored due to time constraints and anticipated minimal impact on performance given the complexity of the dataset. Indeed, according to research performed on MLPs for the EMNIST Balanced Dataset [4], the best model was reached on a 3-layered architecture (same as ours) using dropout and L1 regularisation, both techniques utilised to reduce overfitting, achieving 84.83% accuracy on a testing dataset. In contrast, the developed MLP model without L1 regularisation and dropout achieved almost 80% accuracy on the test set, slightly lower than the article's accuracy.

L2 regularisation penalises the square values of the weights, which can lead to more diffuse models with smaller weights, helping reduce overfitting without increasing sparsity, unlike Lasso regularisation. For L2 weight decay, three values were tested (0.001, 0.01, and 0.1), out of which 0.001 gave the best accuracy for both models.

Evaluation & Results

A common methodology was implemented to evaluate the MLP and CNN models. A baseline was trained for both models, where the loss and accuracy of each epoch were measured. Afterwards, both models were tuned using the hyperparameters outlined previously. Next, the models were trained using the best parameters from the previous step. Then, both models were evaluated using the following metrics: accuracy, precision, recall, and the F1 score. Lastly, a confusion matrix was plotted to visualise how the predicted labels measure up against the true labels.

Baseline Loss and Accuracy Results

In comparing the baseline results, the loss and accuracy values indicate clear differences between the multi-layer perceptron (MLP) and convolutional neural network (CNN) models. The MLP model's loss values ranged from 3.852 to 3.850 as shown in Figure 3(a), suggesting a higher prediction error rate. In contrast, the CNN model's loss values started at 1.326 and stabilised around 0.728, reflecting a better learning capability. Correspondingly, the accuracy values for the MLP model were quite low, between 2.176% and 2.128%, indicating it struggled with correct predictions. In contrast, the CNN model's accuracy values began at 59.493% and improved to a stable range of approximately 77.097% (Figure 3(b)), demonstrating a significantly higher learning efficiency. The differences underscore CNN's superior ability to capture patterns and perform more accurately, likely due to its inherent spatial processing capabilities and robust architecture.

Hyperparameter Tuning Results

The hyperparameter results for MLP and CNN models show that the CNN model achieved a higher best score of 83.65% (Figure 4(b)) compared to 77.86% (Figure 4(a)) for the MLP model, indicating superior performance in the CNN architecture. Both models used a similar batch size of 64 and an identical learning rate of 0.01 with Adam optimiser and a weight decay of 0.001. Even though, MLP had 4 fully-coonected layers to process the image features, the CNN model employed a more complex structure with convolutional layers with multiple channels (32 and 64) and kernel sizes of 3x3, pooling and fully-connected layers, which likely contributed to its enhanced performance. The MLP model had larger output layer sizes (256, 128, and 47) and did not use dropout. CNN's advanced architecture and regularisation strategies appear to provide a more robust learning framework than the MLP.

Tuned Model Results

There is a clear trend of performance improvement over the 10 epochs. In the case of the MLP model, the accuracy increased from 55.56% to 78.47%, while the loss decreased from 1.476 to 0.677, indicating a significant reduction in prediction errors as learning progressed. The CNN model showed a similar trend, with an initial accuracy of 54.99% climbing to 85.496%, and a decrease in loss from 1.598 to 0.424. This demonstrates that both models learned from the data over time, enhancing accuracy and reducing error rates.

The higher final accuracy and lower loss in the CNN model suggest that it was more effective at capturing complex patterns and generalising from the training data than the MLP model. The steeper drop in loss for the CNN model might be attributed to its architecture, which was designed to work with spatial data, allowing for better learning efficiency. Ultimately, the results indicate that while both models exhibit successful learning over time, the CNN model achieves higher accuracy and a lower error rate, reflecting its superior performance.

Evaluation Results

The evaluation results reflect a robust performance with low prediction error rates for both models, as seen in Table 4. MLP had a test loss of 0.67, an accuracy of 79.143%, correctly classifying the test data about 79% of the time. CNN achieved a slightly higher accuracy of 84.048% and a lower test loss of 0.48. The precision and recall values of MLP, at 79.4% and 79.1%, respectively, suggest that the model has a good balance between identifying correct positive predictions and capturing most of the true positive instances. For CNN, both of these metrics scored higher; precision was measured at 0.842 and recall at 0.841. The F1-Score, which combines precision and recall, is 83.9% for MLP and 78.9% for CNN, confirming a balanced approach.

	MLP	**CNN**
Test Loss	0.67	0.48
Accuracy	79.143 %	84.048 %
Precision	0.794	0.842
Recall	0.791	0.841
F1-score	0.789	0.839

Table 4: Comparison of prediction metrics between MLP and CNN models

Predicted Results

Figure 5 illustrates the first six samples of the test set along their true and predicted labels. It can be seen that the predictions are the same both for MLP and CNN, a statement that is further supported by the confusion matrices (heatmaps) of the models in Figures 6 and 7, accordingly. The heatmaps for the two models are almost identical, with CNN having slightly fewer misclassifications and revealing the same pitfalls for both models. Uppercase 'O'

(Oh) is often misclassified for zero (0), while the reverse does not happen often. Another common misclassification is that both models mistaken uppercase 'L' (Ell) with one (1). In Figure 5, it can be seen that the models confuse lowercase 'q' for the number 9, also visible in the heatmaps. One last misclassification prevalent in the heatmaps is between uppercase 'F' and lowercase 'f'.

Comparison of MLP and CNN models

When comparing multi-layer perceptrons (MLPs) and convolutional neural networks (CNNs), it's apparent that CNNs are generally better suited for image classification tasks due to their architecture's spatial awareness. CNNs consist of convolutional layers that excel at capturing local features and hierarchies, allowing them to process and analyse images more effectively. This advantage is reflected in the superior accuracy rates achieved by CNNs in the project, which started at 54.99% and reached 85.496%, along with a significant reduction in loss from 1.598 to 0.424. These results demonstrate CNNs' capacity to detect patterns and learn from data efficiently, making them ideal for image recognition and classification tasks.

On the other hand, MLPs are more general-purpose neural networks that lack the spatial awareness inherent in CNNs. As a result, they may struggle with image-related tasks, as observed in their lower accuracy rates, which increased from 55.56% to 78.47%, and loss values that started at 1.476 and decreased to 0.677. The pros of MLPs include simplicity and flexibility, making them suitable for various tasks, but their lack of convolutional layers means they might miss crucial spatial information in image data. Overall, this project demonstrated that while MLPs and CNNs can learn and improve over time, CNNs offer a clear advantage in image classification due to their specialised architecture, suggesting that further refinement and regularisation strategies could bridge the performance gap between these two models.

Conclusion

Through this project, we gained a deeper understanding of the Multilayer Perceptron and Convolutional Neural Networks architecture, as well as how each tested hyperparameter affected the performance of the models. Experimenting with learning rate schedulers, activation functions, dropout, batch normalisation, optimisers and regularisation techniques allowed us to observe the model behaviour under different settings, highlighting the need to balance model complexity and generalisation.

The investigation of MLP and CNN performance for image classification allowed us to apply the theoretical and practical knowledge we gained during the Applied AI module but also pushed us to further research and experiment with code to create dynamic models that can be reused for other image datasets and optimised to give the best results possible.

In terms of improvements, we would like to have further explored the dynamic architecture of the MLP and CNN models by testing the number of hidden/convolutional layers and also testing more hyperparameters such as different batch sizes, epochs, L1 regularisation and more learning rates schedulers. However, Time constraints in submitting the assignment and the computational time of hyperparameter tuning made this impossible to fit within our assignment. However, it still is a task we would be exploring on our own time out of interest.

References

[1] Gregory Cohen et al. "EMNIST: an extension of MNIST to handwritten letters". In: (Feb. 2017). DOI: 10.48550/arXiv.1702.05373. URL: https://arxiv.org/abs/1702.05373v1 (visited on 04/14/2024).

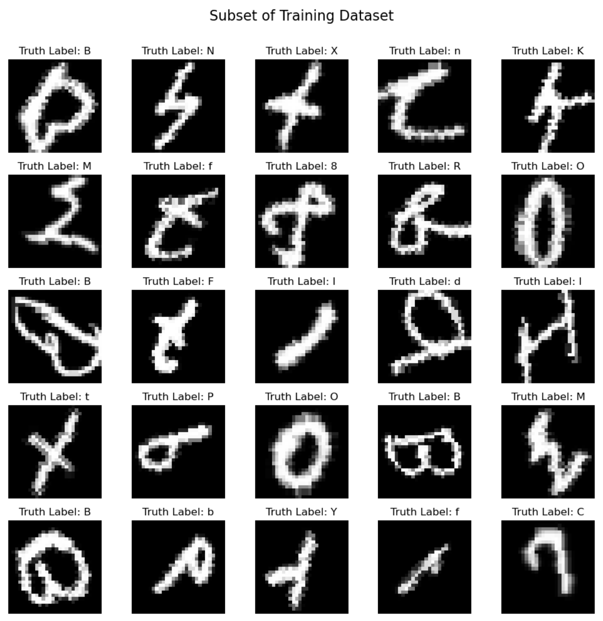
[2] *EMNIST download link is broken · Issue 5662 · pytorch/vision*. GitHub. URL: https://github.com/pytorch/vision/issues/5662 (visited on 04/14/2024).

[3] Nist.gov. *The EMNIST Dataset*. Nist.gov, Apr. 2017. URL: https://www.nist.gov/itl/products-and-services/emnist-dataset (visited on 04/14/2024).

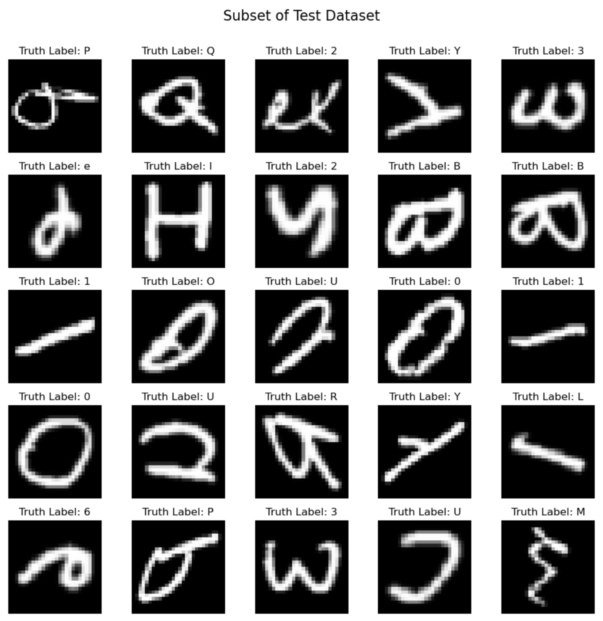
[4] Tidor-Vlad Pricope. "A contextual analysis of multi-layer perceptron models in classifying hand-written digits and letters: limited resources". In: (2021). arXiv: 2107.01782 [cs.LG].

[5] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Appendix

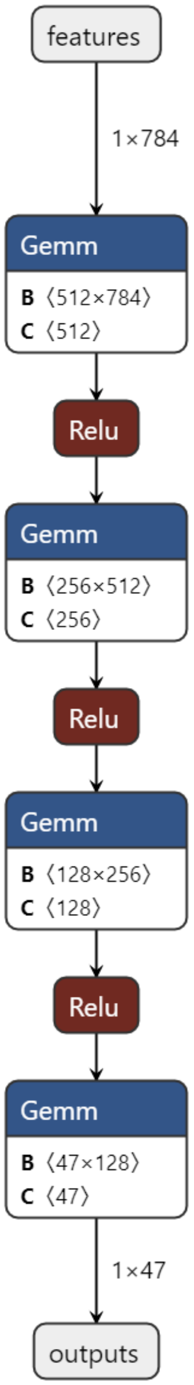


(a)

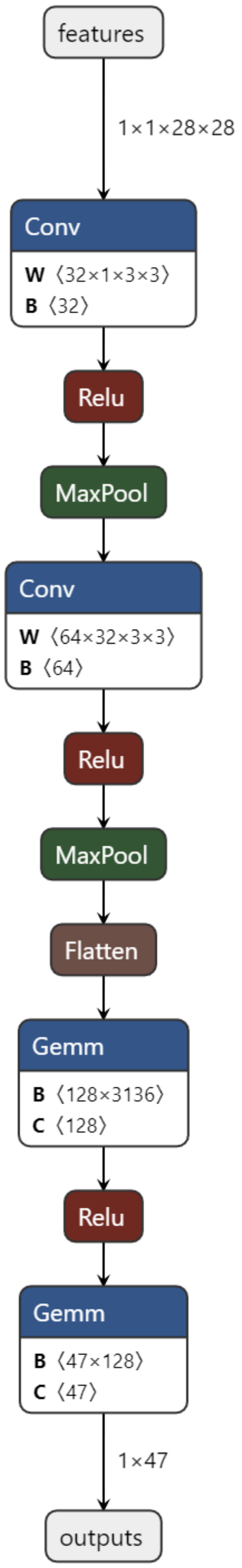


(b)

Figure 1: Visualisation of 25 random images from the training (a) and test (b) datasets after transformation.

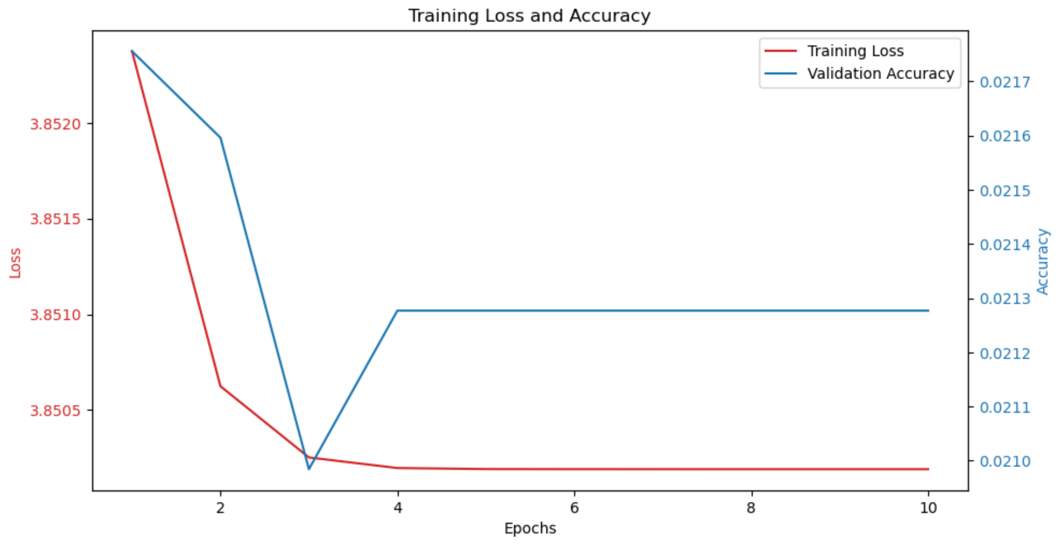


(a)

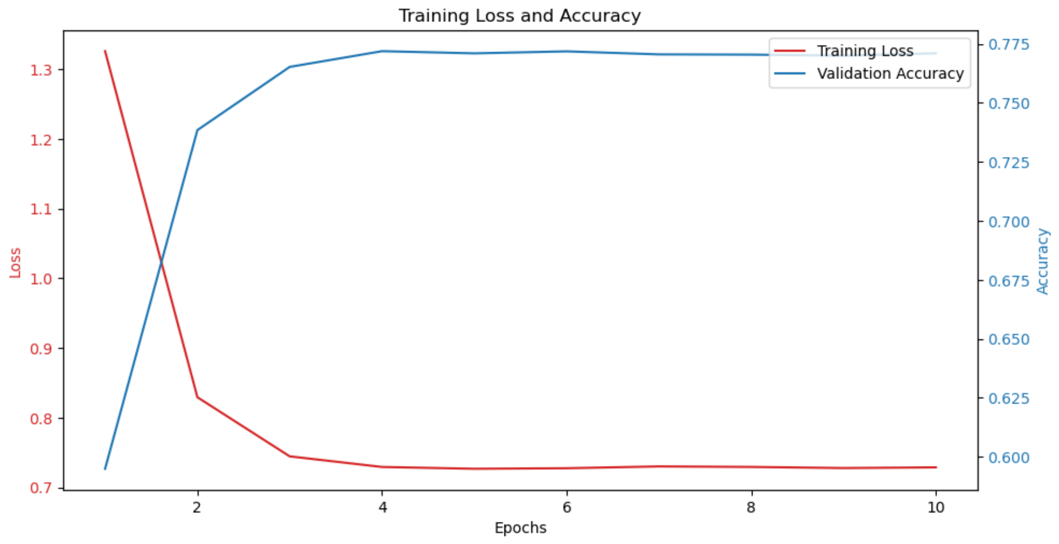


(b)

Figure 2: Model architecture of MLP and CNN (before hyperparameter tuning).

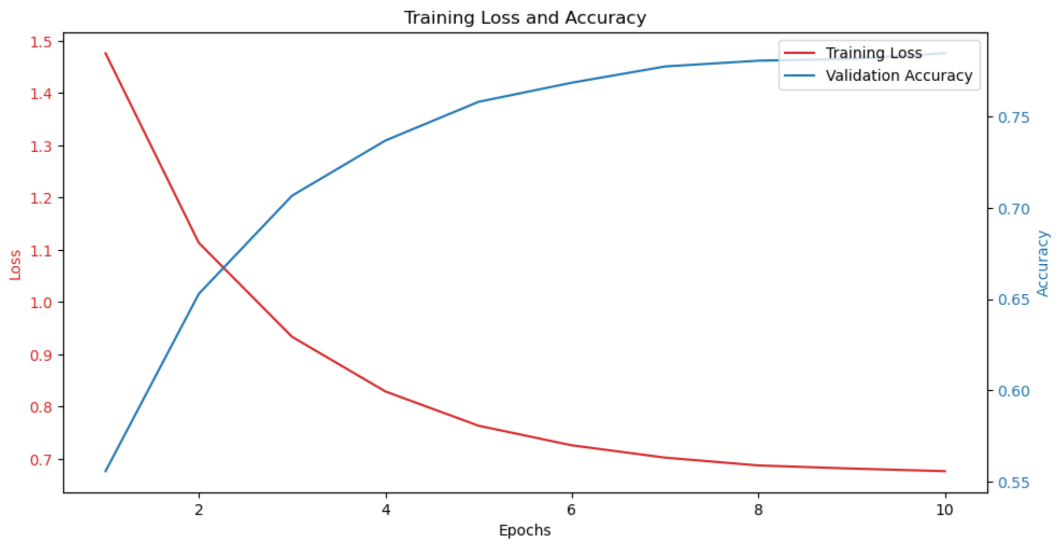


(a)

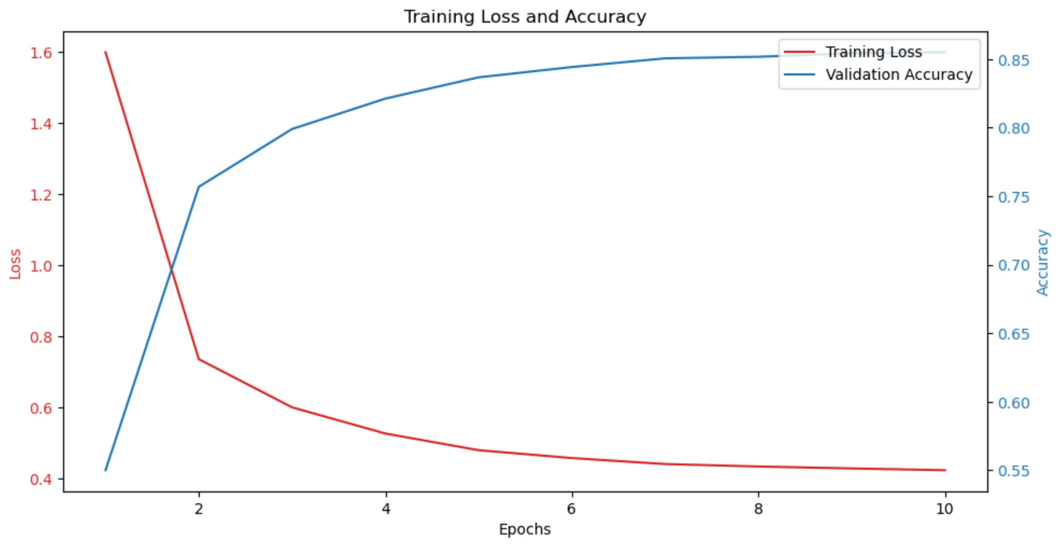


(b)

Figure 3: Training loss and accuracy for the baseline MLP (a) and CNN (b) models.

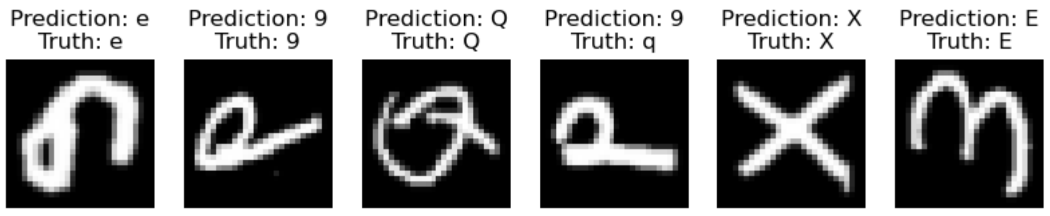


(a)

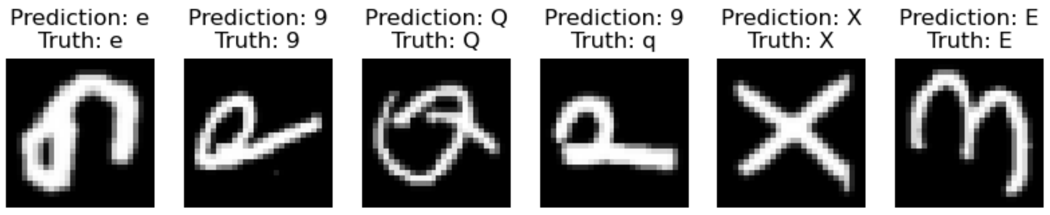


(b)

Figure 4: Training loss and accuracy for the tuned MLP (a) and CNN (b) models.



(a)



(b)

Figure 5: First six (6) images from the test set with their true and predicted labels for the tuned MLP (a) and CNN (b) models.

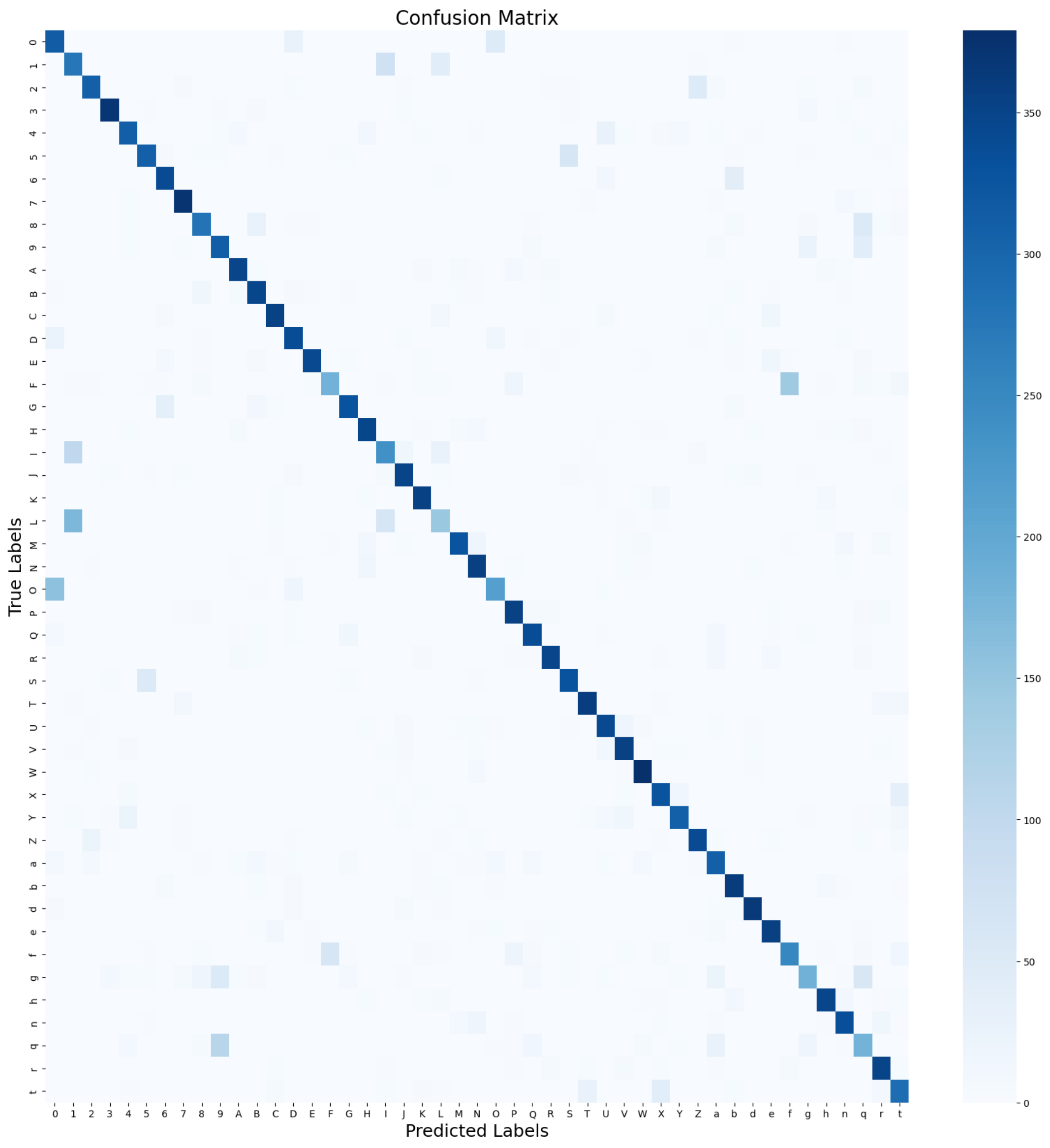


Figure 6: Heatmap for the MLP model showcasing predictions

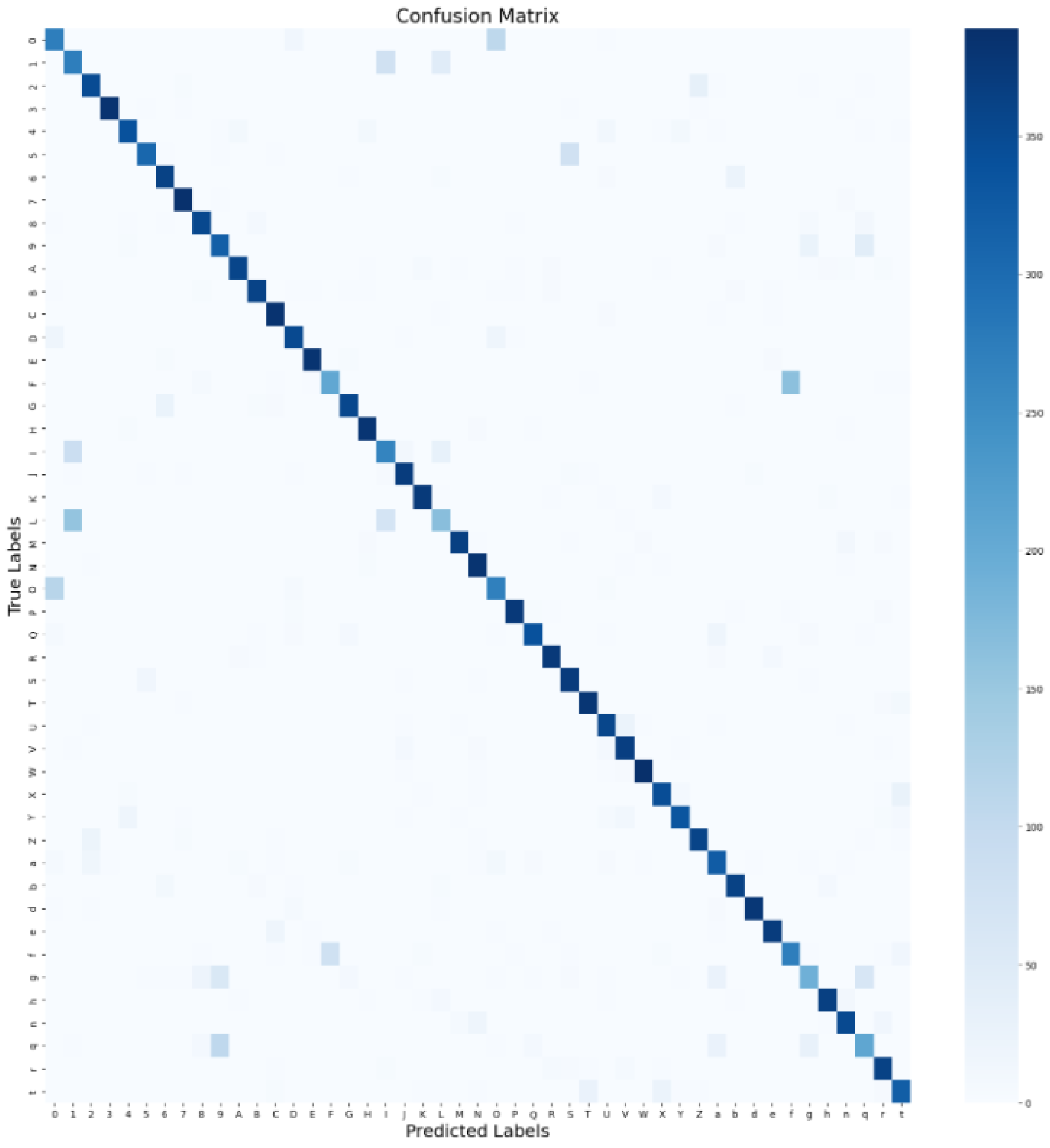


Figure 7: Heatmap for the CNN model showcasing predictions