COMP5318 Assignment 2

Image Classification

Contents

1 Introduction

1.1 Aim

This study aims to compare traditional machine learning algorithms and deep learning algorithms for classifying datasets from OrganMNIST(ACS), which consists of standardized and labeled images from abdominal CT scans.

1.2 Importance

The importance of this study lies not only in the analysis of the OrganMNIST(ACS) dataset itself but also in the broader implications of comparing algorithms for medical image classification tasks. With the increasing availability of medical imaging data, the need for accurate and efficient classification methods is paramount. Understanding which algorithms perform best for specific tasks can significantly impact medical diagnosis, treatment planning, and patient outcomes. Therefore, this comparative analysis contributes to advancing the field of medical image analysis and has the potential to enhance healthcare practices.

2 Data

2.1 Data description and exploration model

The data is a selection from OrganMNIST(ACS) from MedMNIST, which is a large-scale MNIST-like collection of standardized biomedical images in 2D. All images are pre-processed into 28x28 with the corresponding classification labels, so that no background knowledge is required for users. The resulting dataset, consisting of approximately 708K 2D images and 10K 3D images in total, could support numerous research and educational purposes in biomedical image analysis, computer vision, and machine learning. [1] According to MedMNIST, the dataset features diversity, standardization, and user-friendliness.The sub-dataset used in this work, OrganMNIST(A,C,S), includes pre-processed, labeled abdominal CT scans (axal, coronal, and sagittal) that have been categorized into 11 classes. The Liver Tumor Segmentation Benchmark (LiTS)30 provides the 3D computed tomography (CT) images used in OrganA,C,SMNIST. For simplicity, they are now called OrganMNIST_Axial,Coronal,Sagittal (in MedMNIST v19). To get the organ labels, they utilize bounding box annotations of 11 different body organs from a different study. They crop 2D images in axial, coronal, and sagittal views (planes) from the center slices of the 3D bounding boxes. The viewpoints of OrganA,C,SMNIST are the only things that differ. The photos are resized to 1*28*28 in order to classify 11 bodily organs into multiple classes. The training and validation sets consist of 115 and 16 CT scans from the original training set, respectively. [2]

Here we present some of the sample images from the dataset.

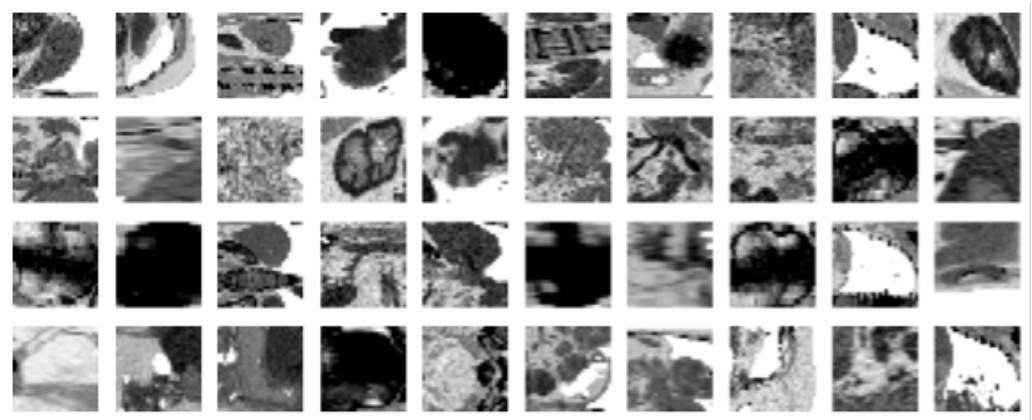


Figure 1: Example Figures

2.2 Pre-processing description and justification

In the pre-processing stage, we did not apply anything sophisticated. We simple apply standard rescaling to the image, changing the value of each pixel from 0 to 255 to 0 to 1 and encode the label from integer to one-hot coding, so that the machine learning algorithm may process this dataset easier. Also, due to the algorithm design, we also change the dataset from matrix into vector in the traditional algorithm and MLP.

3 Methods

Here, we introduce the algorithm we used in this study.

3.1 The traditional algorithm

3.1.1 Theoretical Description

We simply choose KNN algorithm in this section. KNN is a simple and intuitive machine learning algorithm used for classification tasks, based on a simple idea that if two samples are close to each other, they are likely to have similar features like labels. The definition of close are based on the distance function, and the default choice are usually Euclidean distance. Due to the simplicity of this algorithm, KNN is often chosen as a standard for classification tasks, which makes it our choice.

3.1.2 Strengths and Weakness

KNN is one of the simplest classification algorithms, has very little number of parameters, and does not require a training phase, making it very quick on certain datasets. Also, according to this paper[3] KNN may also holds some advantage on image classification tasks, especially when the dataset is small on scale, like the dataset that is used in this study. Now, we talk about the disadvantage about the KNN. From mathematical point of view, the classification task requires to find out the hyperplane that separate different classes. Because the KNN requires to calculate the distance of each pair sample, it suffers from curse of dimensionality, and very sensitive to chaos and outliers, making it not suitable for large scale dataset. Also, due to the nature of this algorithm, the hyperplane founded bears a high level of linearity, where in reality, often not fully satisfy the dataset, resulting it hard to reach a very high accuracy rate.

The KNN does not work well when the dimension of the data is high, which it is in this study. When dimension goes up, the data points becomes sparse, making KNN harder to find the real pattern, because the difference of the distance of the data becomes small.

3.1.3 Architecture and Hyperparameters

KNN algorithm does not have too many hyperparameters. We can only change the number of neighbours and the distance measures. The change of distance measures usually affects running speed and can lead to varying outcomes in terms of classification accuracy and model behaviour. Usually, the Euclidean distance holds the advantage of computation speed, and often resulting fairly high accuracy rate. Other distance measures sometimes would result in higher accuracy rate, yet the difference would not be so significant. The increase in the number of neighbours will certainly increase the computation cost, resulting in longer running time, but, up until a certain degree, it would also increase the tolerance to noise and outliers under the assumption that the noisy point is quite sparse and the amount is quite low.

We use the grid search to find the best hyperparameter combination with cross validation to enhance the robustness of the model.

In the hyperparameter searching, we tested the number of neighbours of 3,5,and 7, and the distance metric where p equals to 1 and 2. We kept the number of neighbours k bigger than 1 to prevent overfitting, especially when the data is sparse as in this case where a small k might lead to strange behaviour, We also want the k to be small enough to prevent underfitting and keep the running time low. The choice of distance metrics were pretty standard, unless there was a special consideration, people often choose 1 and 2.

3.2 Fully connected neural network (MLP)

3.2.1 Theoretical Description

Still, from mathematical point of view, the classification task requires to find out the hyperplane that separate different classes, which means finding a target function based on the data point. It has been proven mathematically, the with the nonlinear activation function, we can approximate any function with any degree of accuracy provided that we have a complicate enough neuro network.

3.2.2 Strengths and Weakness

According to the theory, we can achieve any level of accuracy if we have a complicated enough neuron network, which fundamentally solved the issue of only being able to find linear enough separation caused by KNN. But that would imply the assumption that the dataset was noiseless, which in reality never happens. So, the effort to chase higher accuracy by increasing the complexity of the model would result in more vulnerability to overfitting. Also, because we are dealing with a dataset of images, we would prefer that the model be robust to rather harmless mappings like shifting, rotating, etc. Yet due to the nature of the MLP, we have to apply vectorization to the input; this would cause the input to change dramatically if the image went through those mappings. In theory, we can solve those issues by increasing the model's complexity, but by experience, the overfitting problem would always dominate the result before that issue had been solved. So, the MLP is not very robust to those harmless mappings.

Because the MLP require a training phase, and possess a large number of parameters, it runs slower than KNN. But because throughout the entire process, the computation is very basic and highly vectorized, we can run those in parallel to speed things up, making it less vulnerable to the curse of dimensionality.

3.2.3 Architecture and Hyperparameters

The architecture of the MLP was modified based on the neuron network we used to classify the MNIST on tutorial. After the flatten layer, we built a fully connected layer of 256 neurons, and a fully connected layer of 32 neurons comes after that. Then, the network ended up with a fully connected layer of 11 neurons as an output layer. This architecture had performed reasonably well in the tutorial, which means its model has reached a niche point where we may acquire a good enough hyperplane to separate those classes while not risking too much about overfitting. The MNIST dataset and the dataset we used in this study bears similar feature, including both having class number around 10, both consisting of black and white images, and both having similar image size. With those similarity, we can believe that the model would perform reasonably well on the new dataset.

With MLP, there are many hyperparameters we can tweak. In this study, we put our focus on the optimizer and the activation function. Different choice can affect the network in different ways, generally speaking they typically affects the convergence speed, stability, generalization and the robustness of the model.

We use the grid search to find the best hyperparameter combination with cross validation to enhance the robustness of the model.

The choice of hyperparamters for searching including: optimizer: Adam, SGD, learning rate: 0.001,0.01,0.05,0.1, activation function: ReLU, Sigmoid, and the epoch counts:10,20,50. We made those choices on optimizer and learning rate because on the test run that is not shown in the report or the code, MLP shows a remarkably slow convergence speed, which indicates the learning rate being too small. Yet, a big learing rate might cause overshooting, so those hyperparameters are choosen to find out the best combination. The choice of activation function was there to prevent gradient disapperance, and the epoch was there to give MLP enough time to converge.

3.3 Convolutional neural network (CNN)

3.3.1 Theoretical Description

To solve the weakness of the MLP, people come up with CNN. CNN would perform better on images than MLP, because it can extract feature from images, then classify them based on the extracted features. Generally speaking, the CNN first put the samples into a feature space, then seeks hyperplane in the feature space to separate them. The process of feature extraction would be done by the layers of convolution and pooling. Those features would not be affected so dramatically as the images shifts or rotates, hence proving the robustness to CNN. When human is identifying images, we identify the patterns, or in other term, features. The CNN would perform better by mimicking this human process.

3.3.2 Strengths and Weakness

Because of the forementioned theory the CNN should be robust to those harmless mapping including shifting, rotating and rescaling. But the convolution is a very costly in terms of computation, so we would expect it to run much slower. Also, comparing to MLP, the added layers also increase the complexity of the model, exposing it to a higher risk of over fitting.

3.3.3 Architecture and Hyperparameters

The architecture of the CNN was created based on lecture material, in which it shows an image of the structure of CNN used to classify the image of cars. It has 3 copies of conv-relu-conv-relu-pool before the fully connected layer, and yield pretty good outcome. So, we decide to follow the same architecture. After the feature extraction, we add a flatten layer to connect the MLP. After the flatten layer, we built a fully connected layer of 128 neurons, and a dropout layer comes after that to prevent overfitting. Then, the network ended up with a fully connected layer of 11 neurons as an output layer.

With MLP, there are many hyperparameters we can tweak. In this study, we put our focus on the optimizer, the activation function, and the dropout rate. Different choice can affect the network in different ways, generally speaking they typically affects the convergence speed, stability, generalization and the robustness of the model. The dropout rate affects the complexity of the network, too big would result in underfitting, and too small might risk of overfitting.

We use the grid search to find the best hyperparameter combination with cross validation to enhance the robustness of the model.

The choice of hyperparamters for searching including: optimizer: Adam,SGD, earning rate:0.001,0.01,0.05,0.1, activation function: ReLU, Sigmoid, the epoch counts:10,20,50, and the drop out rate:0.3,0.5,0.7.We made those choices on optimizer and learning rate because on the test run that is not shown in the report or the code, CNN shows a remarkably slow computation speed, which a bigger learning rate might solve the issue. Yet, a big learing rate might cause overshooting or other issues, so those hyperparameters are choosen to find out the best combination. The choice of activation function was there to prevent gradient disapperance, and the epoch was there to give the model enough time to converge when the learning rate is small.

4 Results and discussion

4.1 Hyperparameter tuning results presentation

K-Nearest Neighbors (KNN)

Parameters Tuned:

Number of Neighbors: 3, 5, 7

Distance Metric: Manhattan (p=1), Euclidean (p=2)

Results:

Best Parameters: { 'n_neighbors': 3, 'p': 1 } (Manhattan distance with 3 neighbors)

Best Cross-validation Score: 86%

Multi-Layer Perceptron (MLP)

Parameters Tuned:

Optimizer: Adam, SGD
Learning rate: 0.001, 0.01, 0.05, 0.1
Activation Function: ReLU, Sigmoid
Loss Function: Categorical Crossentropy
Training Epochs: 10, 20, 50

Results:

Best Parameters: { 'activation_function': 'sigmoid', 'epochs': 50, 'loss': 'categorical_crossentropy', 'optimizer': 'sgd', 'learning_rate': 0.05 }
Best Cross-validation Score: 87%

Convolutional neural network (CNN)

Parameters Tuned:

Optimizer: Adam, SGD
Learning rate: 0.001,0.01,0.05,0.1
Activation Function: ReLU, Sigmoid
Loss Function: Categorical Crossentropy
Training Epochs: 5, 10, 20
Dropout: 0.3, 0.5, 0.7

Results:

Best Parameters: { 'activation_function': 'relu', 'dropout': 0.5, 'epochs': 50, 'loss': 'categorical_crossentropy', 'optimizer': 'sgd', 'learning_rate': 0.05 }
Best Cross-validation Score: 96%

4.2 Hyperparameter tuning discussion

K-Nearest Neighbors (KNN)

Trends:

The best performance was achieved with the smallest number of neighbours ($k = 3$) and using the Manhattan distance metric ($p = 1$).
Typically, fewer neighbours can better capture the nuances in data with less noise, leading to higher accuracy in datasets where the boundaries between classes are distinct.

Multi-Layer Perceptron (MLP)

Trends:

The configuration with the ReLU activation function, SGD optimizer with 0.05 learning rate, and categorical cross entropy loss function with the highest accuracy.
ReLU's performance can be used to reduce the vanishing gradient problem which could create a healthy gradient flow through the network.

Convolutional neural network (CNN)

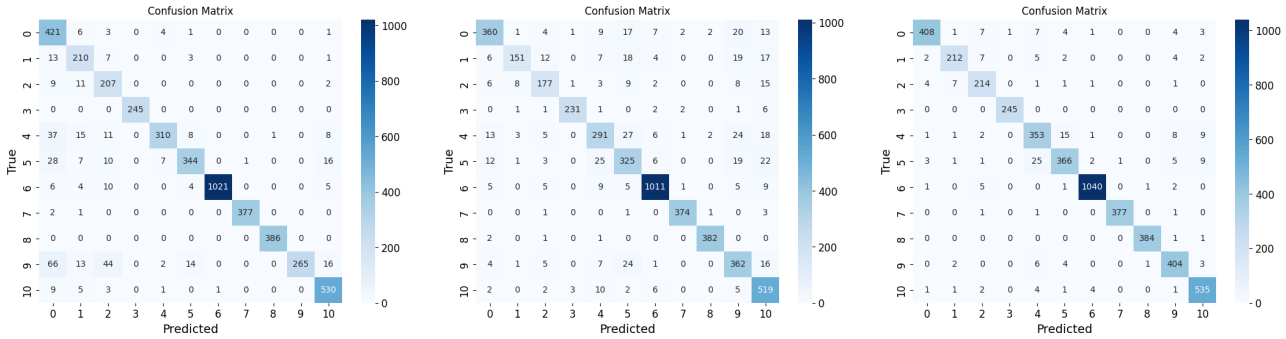
Trends:

The best settings are ReLU activation, SGD optimizer with 0.05 learning rate, dropout rate 0.3, and the categorical crossentropy loss function.
Like the use of ReLU in MLP. Compared with Sigmoid, ReLU's performance can be used to reduce the vanishing gradient problem which could create a healthy gradient flow through the network.
A dropout rate of 0.3 is the optimal rate. Which could provide enough regularization to prevent overfitting without losing too much network capacity.

4.3 Results table

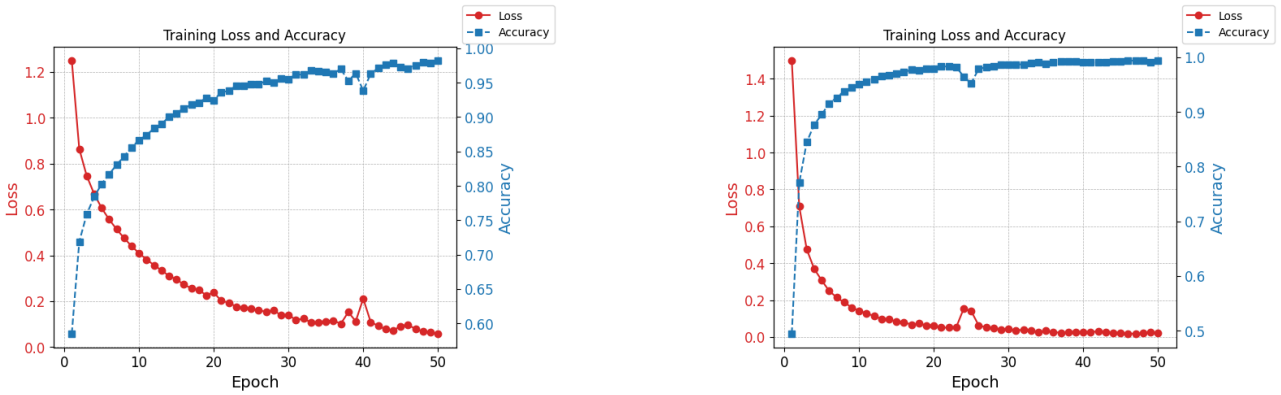
Table 1: Comparison of Classification Models

Model	Evaluation Metrics		
	Precision (Weighted Avg)	Recall (Macro Avg)	F1 Score (Micro Avg)
KNN	0.93	0.91	0.91
MLP	0.89	0.88	0.87
CNN	**0.95**	**0.97**	**0.96**



(a) KNN confusion matrix (b) MLP confusion matrix (c) CNN confusion matrix

Figure 2: Confusion Matrixs(Full size A)



(a) MLP training loss and accuracy (b) CNN training loss and accuracy

Figure 3: Training Loss and Accuracy(Full size A)

4.4 Results discussion/analysis

As we can see table 1. CNN has the highest performance across all the metrics. The reason for that may be due to the ability to extract hierarchical features directly from raw data makes it good for image pattern recognition tasks.

KNN also gets a high score. This may be because the images are not too complex and fit well with the features captured by KNN. An intriguing finding reveals that selecting lower values for k gets higher scores. This phenomenon could be due to overfitting of the dataset.

MLP slightly lags in performance metrics. As shown in Figure 3a, both 'Accuracy' and 'Loss' could potentially improve with additional epochs. Therefore, the primary reason for MLP achieving the lowest score among the three models is likely the insufficient training epochs.

5 Conclusion and future work

5.1 Summary of main findings and identification of study limitations

After comparing K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), and Convolutional Neural Network (CNN) performance by 'Precision', 'Recall', 'F1 Score' and 'Training Loss and Accuracy' we find that:

CNN has the highest precision, recall, and F1 scores, affirming its capability to extract complex patterns and features from data, making it especially suitable for tasks involving high-dimensional data such as images.

KNN performed also pretty well, its effectiveness in scenarios where the relationship between features is more straightforward and can be captured through distance metrics.

MLP has a slightly lower score compared to CNN, which may reflect challenges related to network architecture depth and the vanishing gradient issue.

After this study, we also found some limitations below:

Runtime Efficiency: Although CNN has the highest performance at the same time it needs a large amount of resources to run it. KNN, although quicker at prediction time but in larger data sets its accuracy might be lower.

Interpretability: KNN and MLP generally provide more straightforward interpretative frameworks compared to CNN. Its deep and complex architectures make it difficult to understand how decisions are being made.

5.2 Future work suggestions

For future work we could pruning to remove redundant information in the neural network, decreasing the number of operations required. Which could improve the runtime efficiency and interoperability of the CNN model. Additionally, we can identify an "elbow point" in the training loss and accuracy curve to ascertain whether the training process should be halted or continued.

6 Reflection

6.1 Reflection on most important thing learnt

For KNN, which is simple and efficient compared to MLP, although its accuracy may be slightly lower than CNN, it requires very few computational resources to generate accurate predictions.

For MLP the time needed to finish epoch is less than CNN. However, the convergence is much slower than CNN.

For CNN, its has the highest accuracy. However, it demands significant computational resources, which might not be suitable in resource-constrained environments like our personal computers.

References

[1] Liu, Qian and Shi, Xiang and Wang, Ping and Li, Yan and Su, Hang and Cao, Hongyun and Ye, Bo and Liu, Xiaohua, *Medmnist dataset*, https://medmnist.com/, Accessed on May 9, 2024.

[2] L. Vandepitte, S. Claus, N. De Hauwere, F. Hernandez, and W. Appeltans, "A standardized metadata model for the european marine observation and data network (emodnet) biology data portal," *Scientific Data*, vol. 9, no. 1, pp. 1–13, 2022. DOI: 10.1038/s41597-022-01721-8.

[3] A. N. Zhang, "Title of the article," *Journal Name*, vol. Volume Number, no. Issue Number, Page Numbers, 2019. [Online]. Available: https://www.cqvip.com/qk/93105a/201905/7100113998.html.

A Appendix

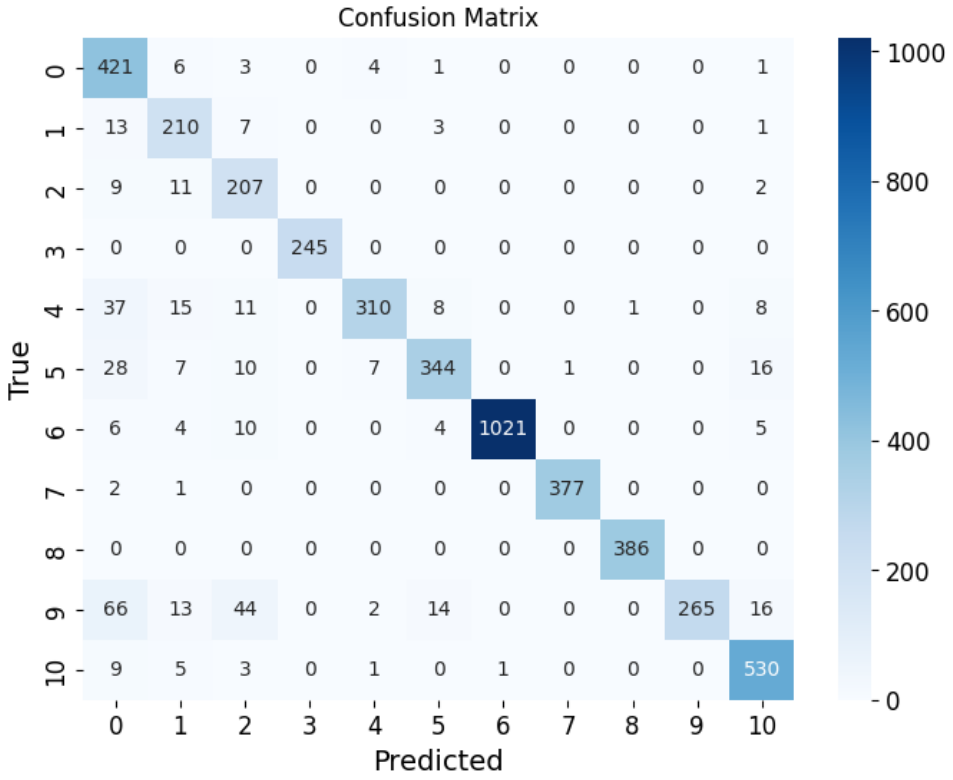


Figure A.1: KNN Confusion Matrix

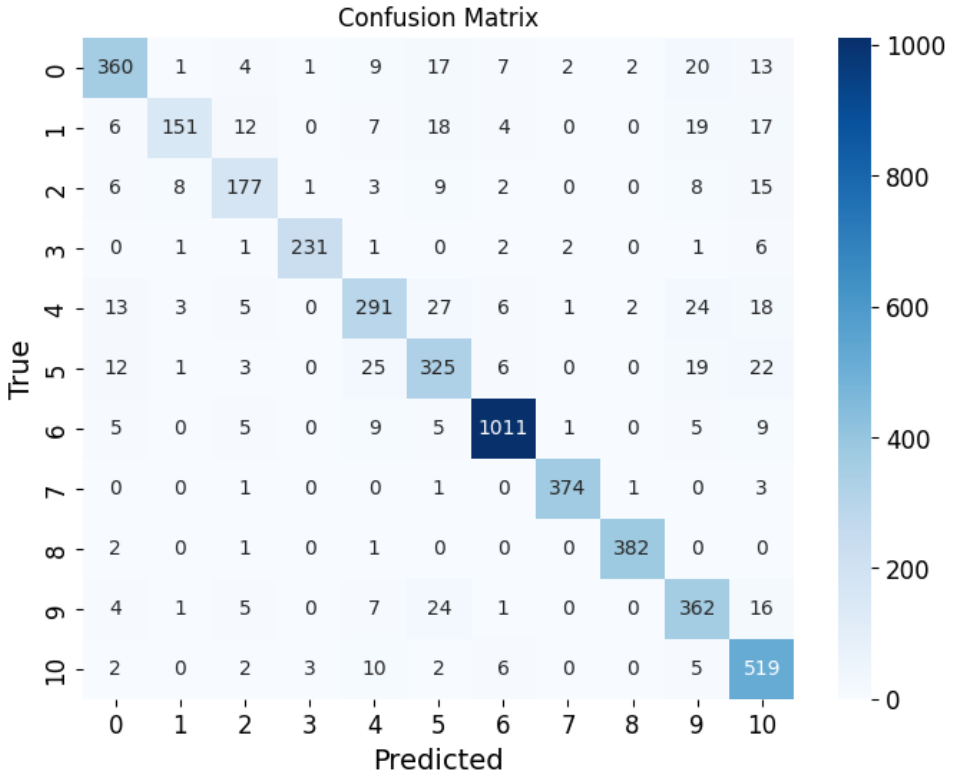


Figure A.2: MLP Confusion Matrix

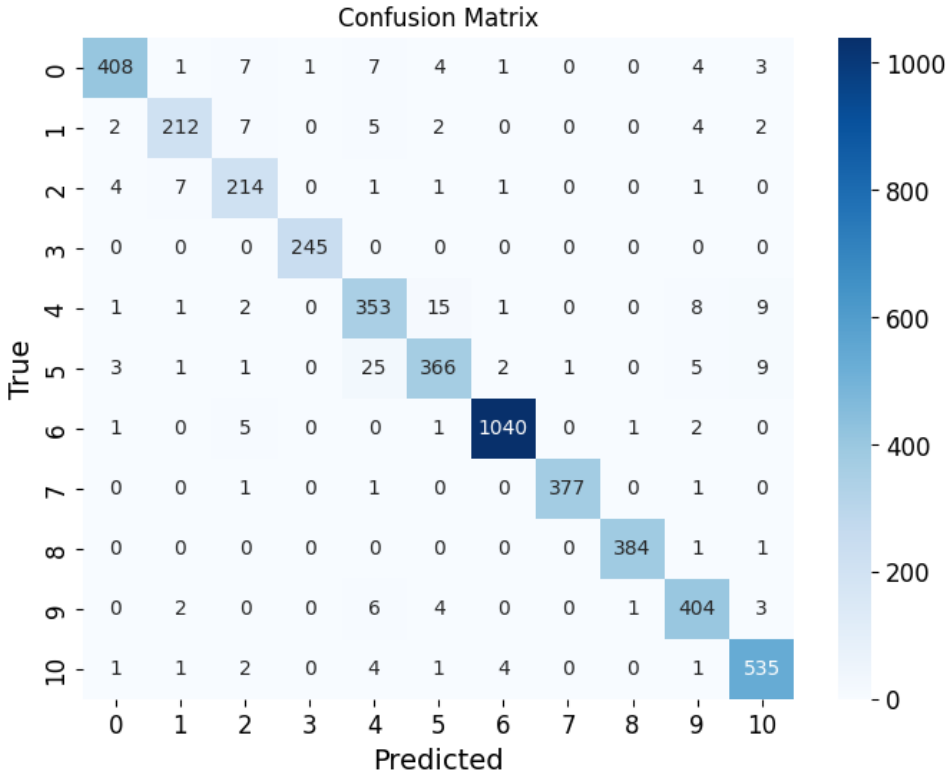


Figure A.3: CNN Confusion Matrix

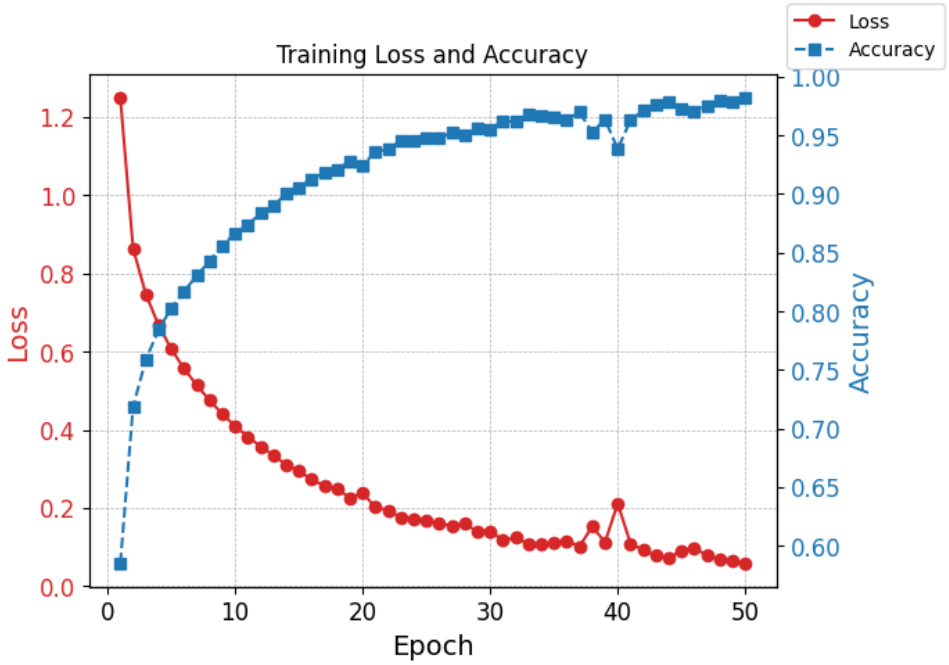


Figure A.4: MLP Training Loss and Accurac

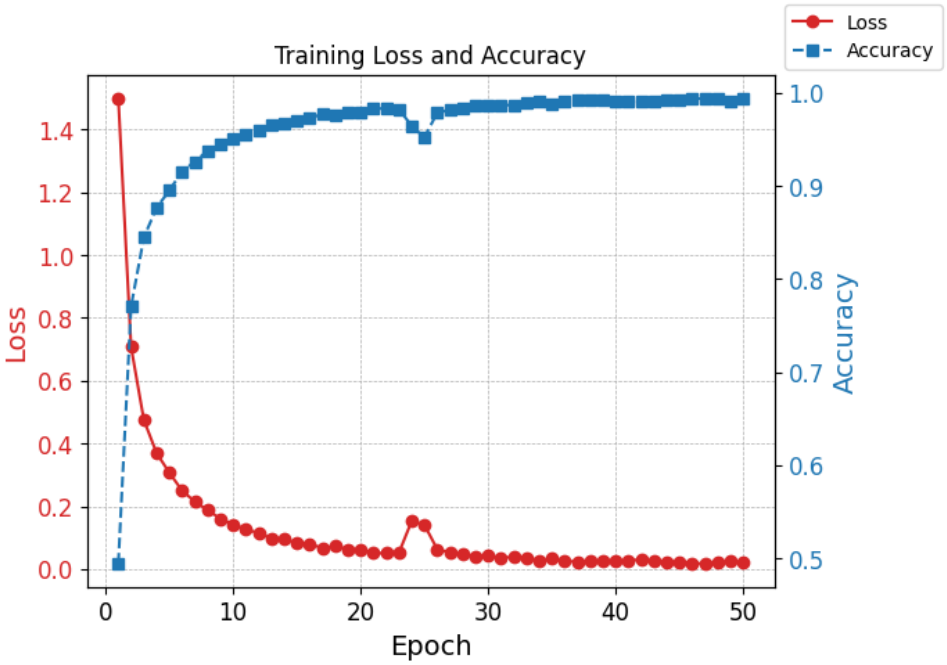


Figure A.5: CNN Training Loss and Accurac