REPORT FOR ASSIGNMENT 2

Task1. A Section explaining the project.

i. Introduction of the dataset

The EMNIST Balanced dataset indeed has 47 classes. The dataset consists of 10 numeric classes (0-9) and 37 letter classes, which include both uppercase and lowercase letters, but with a distinction only between uppercase and lowercase for the 26 letters that have distinct forms (A-Z, a-z, excluding C, I, J, K, L, M, O, P, S, U, V, W, X, Y, Z).

Here is a brief overview of the EMNIST Balanced dataset:
Number of samples: The dataset contains a total of 131,600 samples, with 47 classes and 2,800 samples per class.

Data split: The dataset is divided into a training set (112,800 samples, approximately 85%) and a test set (18,800 samples, approximately 15%).
Image size: Each image is a 28x28 pixel grayscale image, like the original MNIST dataset. Pixel values range from 0 to 255.

ii. Describe the structure of your MLP and CNNs and discuss how you build up them.

Multilayer Perceptron (MLP):
The MLP consists of five layers, including the input and output layers. The input layer has 784 neurons, which is suitable for processing 28x28 grayscale images. The output layer has 47 neurons, indicating a classification task with 47 classes.

Between the input and output layers, there are three hidden layers with 512, 256, and 128 neurons, respectively. These hidden layers have activation functions, batch normalization, and dropout as optional components. The activation function introduces non-linearity into the network, batch normalization helps with faster convergence and improved performance, and dropout is a regularization technique that reduces overfitting by randomly dropping neurons during training.

Convolutional Neural Network (CNN):
The CNN has two convolutional layers followed by two fully connected layers. The convolutional layers have a kernel size of 3 and padding of 1, meaning they use 3x3 filters and apply padding to keep the spatial dimensions constant. The first convolutional layer has 32 output channels, and the second one has 64 output channels.

After each convolutional layer, there is an optional batch normalization layer, an activation function, and a max-pooling layer with a kernel size of 2 and a stride of 2. The max-pooling layers reduce the spatial dimensions by half while retaining the most important information.

The output of the second max-pooling layer is flattened and passed through two fully connected layers. The first fully connected layer has 256 neurons and is followed by an optional batch normalization layer, an activation function, and an optional dropout layer. The output layer has 47 neurons for classifying input into one of the 47 classes.

Task2. A Section explaining the rationale of your design on MLP and CNNs

i. What kind of hyperparameters or techniques you choose to tune and explore.

We choose all six hyperparameters to tune and explore and we use Cross validation technique.

ii. Why you choose those hyperparameters or techniques.

We want to find the best combination of hyperparameters by using control variable method rather the grid search and we use Cross validation technique to make the train process more reliable.

iii. For each technique, please describe how it affects your model's performance and analyse the reasons why the technique can boost or decrease the performance of your models.

For six hyperparameters:

Scheduler type:
Step decay: Improves convergence by allowing fine-tuning after initial learning.
Exponential decay: Adapts to dataset complexity and accelerates convergence.
In this assignment, we found that exponential decay works better for both models.

Activation functions:
ReLU: Efficient for deeper networks; dead neurons may be an issue.
LeakyReLU: Addresses dead neuron issue, slightly better performance than ReLU.
ELU: Smoother output for negative inputs, better generalization.
In this assignment, we found that ELU works better for both models.

Optimizers:
SGD: Slower convergence, not the best choice for complex datasets.
Adam: Fast convergence and improved generalization.
RMSprop: Faster convergence, better performance than SGD.
In this assignment, we found that Adam works better for MLP and RMSprop works better for CNN.

Batch normalization:
True: Improves performance by reducing internal covariate shift and allowing higher learning rates.
False: Slower convergence and less stable training.
But in this assignment, we found that not adding batch normalization works better for both models.

L1 and L2 regularization options:
None: May lead to overfitting.
L1: Encourages sparsity, improves generalization, and reduces overfitting.
L2: Encourages small, non-zero weights, improves generalization, and reduces overfitting.
In this assignment, we found that not using any regularizations works better for both models.

Dropout options:
True: Helps prevent overfitting but can cause underfitting if too high.
False: May lead to overfitting.
In this assignment, we found that not using drop out works better for both models.

Cross-validation:

Boosts performance by providing better model evaluation, improved hyperparameter tuning, and increased robustness to data distribution. However, it might also increase training time and, in some cases, lead to overfitting the validation set.

In this assignment, we firstly didn't use cross-validation, and the training process didn't last long but the first initial six hyperparameters matters a lot and the result varies a lot so we just polish up the training part by adding cross-validation. Then we use 5-fold cross-validation to train the training part, it took us quite a lot of time to finish this process, but the result shows a great stability when it comes to both MLP and CNN models.

iv. Do you ever have overfitting or underfitting issue while training the models? How do you deal with overfitting or underfitting issue?

Yes, we have overfitting issue while training the models. Not using regularization may causing this issue. Adding batch normalization and drop out can solve the overfitting problem. But the model works well on the train part without using batch normalization and drop out while using Cross validation technique. So, we manually change the hyperparameters to get a better result on test part.

Task3. A section describing and explaining your results:

i. Describe and explain the training loss, testing loss and testing accuracy of MLP and CNNs.

Training loss refers to the error of the model on the training dataset, which is used to optimize the model parameters during the training process. The training loss is usually calculated as the average loss over all the training samples. The goal of the training process is to minimize the training loss by adjusting the model parameters.

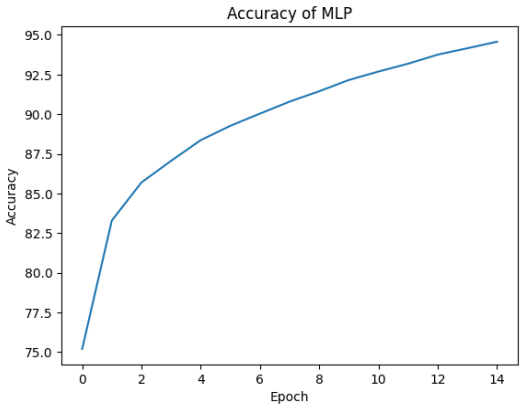
Testing loss refers to the error of the model on the testing dataset, which is used to evaluate the generalization performance of the model. The testing loss is calculated in the same way as the training loss, but it is computed on the testing dataset. The testing loss can help us determine if the model is overfitting or underfitting. If the testing loss is significantly higher than the training loss, it indicates that the model is overfitting.

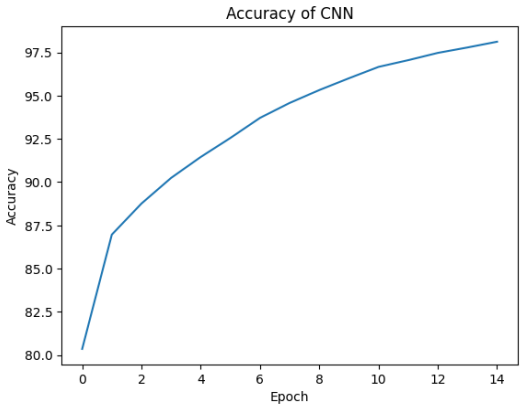
Testing accuracy refers to the percentage of correctly classified samples in the testing dataset. It is a common metric to evaluate the classification performance of the model. The testing accuracy can be calculated as the number of correctly classified samples divided by the total number of samples in the testing dataset.

ii. Describe and explain the predicted results of your models.

1. Accuracy of each model

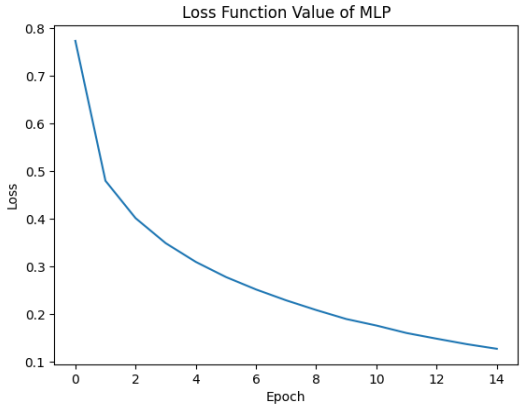
We found that CNN works better than MLP model after 15 epochs.

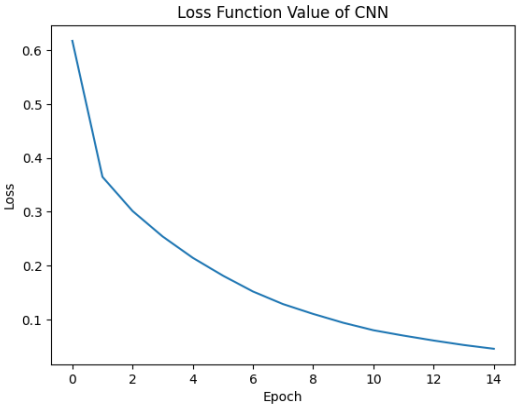




2. Loss of each model

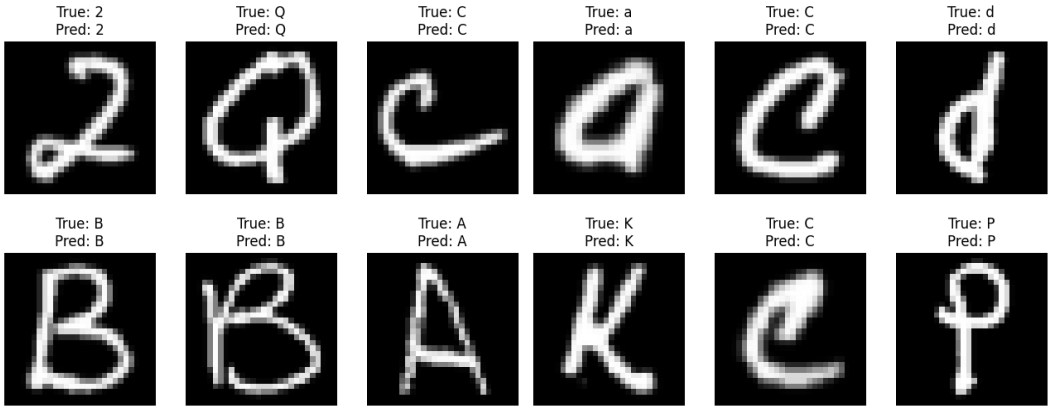
We found that the value of train loss function of each model decreased in the training process, and the value of test loss function of CNN is smaller than MLP, which means CNN performs well than MLP.





3. Prediction result

We can see that both model predict well in their prediction performance.

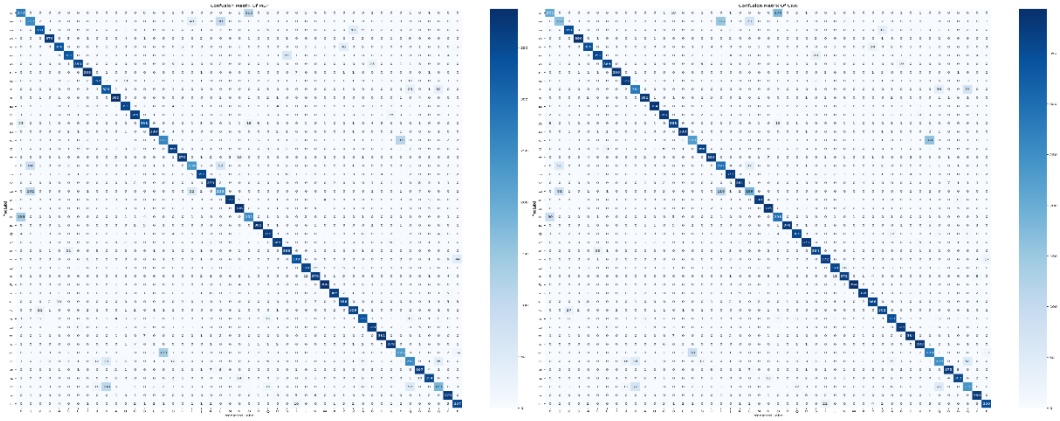


predict result of MLP

predict result of CNN

4. Confusion matrix

From the graph of confusion matrix, we found that both models perform well.



confusion matrix of MLP

confusion matrix of CNN

5. Summarize the performance of two models

	precision	recall	F1-sscore
MLP	**0.85**	**0.85**	**0.85**
CNN	**0.86**	**0.86**	**0.86**

In conclusion, CNN model performs better than MLP.

iii. Your analysis on the performance of MLP and CNNs based on your own AI knowledge. You need to discuss the pros and cons of MLP and CNNs and analyse why a certain model outperformance another model.

CNN: Convolutional Neural Networks have the advantage of local receptive fields and parameter sharing, which allow them to effectively capture local features in images while reducing model complexity. CNNs perform well in image recognition tasks. Given that the EMNIST Balanced dataset is an image classification task, CNNs can achieve good results on this dataset. The strength of CNNs lies in their ability to learn feature representations automatically, avoiding the need for manual feature engineering.

MLP: Multilayer Perceptron is a basic artificial neural network consisting of an input layer, hidden layers, and an output layer. Although MLPs can perform well on certain tasks, their efficiency in processing image data is generally lower than that of CNNs. This is because MLPs do not consider the spatial information in images, and they use flattened image data as input, leading to a loss of spatial information. Additionally, MLPs often have many parameters, which can result in overfitting.

In summary, CNNs are more efficient than MLPs when applied to the EMNIST Balanced dataset, primarily because they can capture local features and retain spatial information in images, while MLPs lose this information.

Task4. The final conclusions:

i. Your own reflection on this project, for example, what you have learned via this project, which part you could do better next time, etc.

In this project, we learned how to build MLP and CNN model and using cross validation to explore the options of different hyperparameters. The most important part is that I learned how to get the best parameters without using GridSearchCV Method, because sometimes the number of the parameters is large, thus we cannot generate the result in limited time. We could Initially, establish a baseline model configuration, explore the best hyperparameters one by one, finally get the best combination. On the other hand, sometimes we found that the accuracy of each model is extremely high, we should learn more about overfitting and underfintting.