Group number: A2 group 66 , SID1: 530311278 , SID2: 500312012

5318 Assigment 2 Report

Introduction

The purpose of this report is to implement and compare the differences of three machine learning algorithms' performance - random forest, MLP and CNN - in the task of histopathology image classification. Different data preprocessing methods are used according to the characteristics of different models and parameters are adjusted. The best parameters are selected to adjust the model. Finally, the classification accuracy and performance differences of the three models on the test set are compared and the reasons are analyzed. Through this process, we explore the advantages and disadvantages of the three models in image classification tasks, which can provide certain guidance and reference for which model will have better performance when encountering similar tasks in the future. The PathMNIST dataset used in this task comes from slice images of different human tissues under a microscope. In practical applications, if pathological images can be efficiently classified, it can assist doctors in normal and abnormal tissue identification, disease screening and diagnosis, etc., which has important application value in biology, medicine and other fields. Deep learning based biomedical image analysis plays an important role in the intersection of artificial intelligence and healthcare(Yang et al., 2023). However, due to the inherent complexity in biomedicine, data modalities, dataset scales and tasks in biomedical image analysis could be highly diverse(Yang et al., 2023). Therefore, choosing an efficient and accurate classification algorithm is of practical significance for improving diagnostic efficiency and decision-making quality.

The PathMNIST dataset has good properties suitable for machine learning research. Its image resolution is small (28×28 pixels), the category labels are clear, and there are no obvious missing values or abnormal distributions, which helps to compare different algorithms fairly. In addition, the nine classification tasks are both difficult and not too complicated, which can effectively support the comparative analysis of models in terms of feature extraction ability, fitting ability, and generalization ability.

Data

Data description and exploration

The dataset used this time is a subset of the MedMNIST v2 dataset, which is a large-scale standardized biomedical image dataset designed specifically for machine learning (Yang et al., 2021). It contains 40,000 color medical images with a resolution of 28x28 pixels, and each image contains

three color channels (RGB) (Table 1). Among them, 32,000 images are used for training and 8,000 images are used for testing.

	Image dimensions	Number of samples
X_train	28 x 28 pixels with 3 channels (RGB)	32,000
X_test	28 x 28 pixels with 3 channels (RGB)	8,000

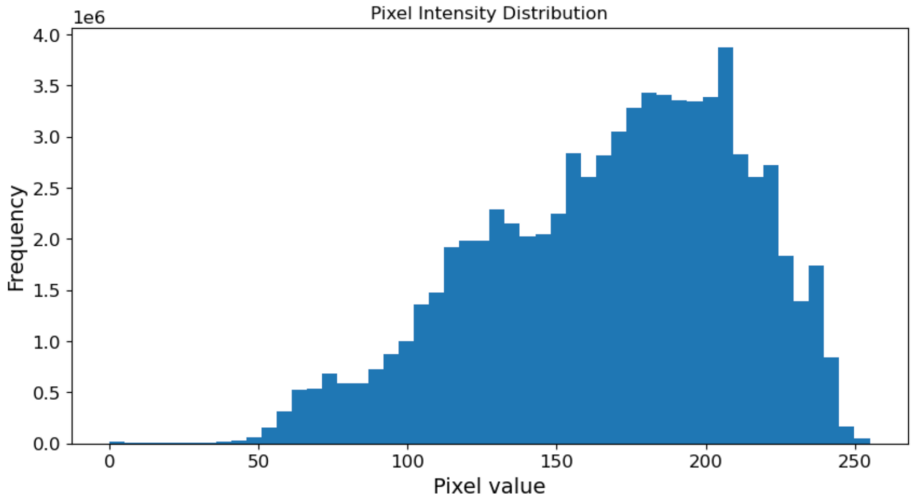
Table 1

The dataset has 9 categories (Class 0-8) and is a multi-classification task. There are some uneven distributions among the categories, for example, Class 6 has the least samples (2728/682 samples) and Class 8 has the most (4697/1174 samples) (Table 2). Uneven sample distribution may affect the classification performance of the model later, so it is necessary to consider the uneven distribution of categories when evaluating the performance of the model later.

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Total
y_train	3490	3431	3505	3656	2950	4290	2728	3253	4697	32,000
y_test	873	858	877	914	737	1072	682	813	1174	8,000

Table 2

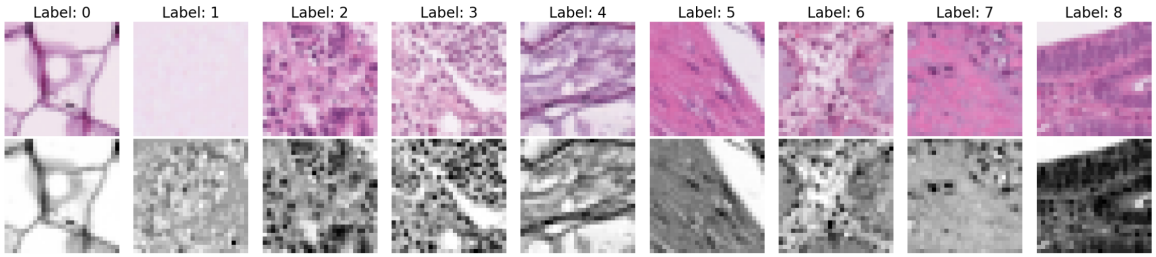
First, we looked at the pixel value distribution of the images in the training set (Picture 1). The pixel values are mainly concentrated between 120 and 230, indicating that the image is relatively bright overall and there are no samples that are too dark. At the same time, extreme values (0 or 255) appear less frequently, which means that there are no large areas of "all black" or "all white" areas in the image, and the overall image quality is high. The pixel distribution graph presents a right-biased but smooth curve, with a roughly single-peak rise and then fall trend from left to right, without obvious jumps or discrete points, indicating that the pixel values of each channel are relatively balanced in overall distribution. Hematoxylin and Eosin (H&E) staining is commonly used in histological images to enhance tissue contrast and assist visual judgment (Madusanka, Jayalath, Fernando, Yasakethu, & Lee, 2023), so this high brightness feature is also consistent with the common characteristics of medical tissue section images.



Picture 1

To further understand the data, we randomly visualized samples of each category of images, displaying them in color (RGB) and grayscale (average grayscale). The results show that most images are relatively centered, that is, the center of the image contains the main structure, and the texture edges and shape structures are relatively clear.

For example, Label 0 has an obvious mesh structure, which helps the model extract effective features. However, the image features of individual categories (such as Class 1 and Class 7) are relatively fuzzy, and the texture contrast is weak, which may make it difficult for the model to distinguish.



Picture 2

Initially, we considered that grayscale images could reduce computational costs and training time, while removing color information could prevent the model from overfitting color features. However, in the comparative experiment, we finally chose to use the original color image as the model input for the following reasons:

	Advantage	Disadvantage
Grayscale	Reducing the input channels from 3 to 1 can speed up training; After removing the color, the model will focus more on the texture and structure of the image; Reduce redundant information	Color is one of the important clues to distinguish categories. Removing it will lead to a decrease in model performance.
Original color	The complete information of the image is retained; The model with color images as input performs better during training; The boundary structure of some tissues in the color image is clearer.	Long training time; Increased model complexity and overfitting risk.

Table 3

Therefore, although grayscale images have the advantage of reducing running time, the complexity of the model built for this task is not high, and the running time is within an acceptable range. Color images provide more discriminative information and can significantly improve model performance. Therefore, we finally chose to use the original color image as the input of the model.

Pre-processing description and justification

When the raw image data is not processed, there may be differences in pixel distribution such as brightness and contrast between different images. Directly using it for training may lead to deviations during subsequent model training and some situations where the convergence speed is deteriorated due to pixel dominance. Studies have shown that data quality has a significant impact on model performance, and poor quality data mainly affects accuracy and leads to incorrect predictions. (Maharana, Mondal & Nemade, 2022). Therefore, we need to perform necessary data preprocessing on the raw image data. At the same time, in order to ensure that the comparison between different models is explanatory and fair, we need to adopt a unified data preprocessing strategy.

Random Forest

The random forest model cannot process image data directly. It needs to flatten each image into a one-dimensional vector so that each pixel corresponds to a feature to meet the input requirements of the model. Studies have shown that it is important to standardize the features of the tree model when dealing with highly heterogeneous feature distributions (Pedregosa et al., 2011). Therefore, we initially used two data preprocessing methods for the random forest: normalizing the original image pixel values [0, 255] to the range [0, 1], and standardization. However, as shown in picture 1, the image pixel value range is not highly skewed and has good balance. Standardization does not bring obvious advantages. Using only Normalization and keeping the same preprocessing method as the other two models is more conducive to the comparison between the models. The results of the baseline (Picture 3) show that the difference in model performance between the two is also minimal, which further verifies the rationality of this judgment.

Classification Report:

	precision	recall	f1-score	support
0	0.82	0.98	0.89	873
1	0.99	0.90	0.94	858
2	0.58	0.32	0.41	877
3	0.74	0.75	0.74	914
4	0.54	0.65	0.59	737
5	0.62	0.73	0.67	1072
6	0.49	0.12	0.19	682
7	0.49	0.46	0.47	813
8	0.52	0.74	0.61	1174
accuracy			0.65	8000
macro avg	0.64	0.63	0.61	8000
weighted avg	0.64	0.65	0.63	8000

Classification Report:

	precision	recall	f1-score	support
0	0.82	0.98	0.89	873
1	0.99	0.90	0.94	858
2	0.58	0.32	0.42	877
3	0.73	0.74	0.74	914
4	0.54	0.65	0.59	737
5	0.62	0.73	0.67	1072
6	0.49	0.12	0.19	682
7	0.49	0.46	0.47	813
8	0.52	0.74	0.61	1174
accuracy			0.65	8000
macro avg	0.64	0.63	0.61	8000
weighted avg	0.64	0.65	0.63	8000

Picture 3 Normalization + Standardization (left) vs Normalization (right)

MLP

Since MLP can only accept one-dimensional vector input, we flatten the two-dimensional image through the Flatten() function of keras.layers when constructing the model, without changing the

image structure in the preprocessing stage like random forest. We need to normalize the image because most neural network weight initialization methods usually assume that the values of the input features are distributed in a small numerical range. If the input value is too large, the gradient will easily disappear when passing through the activation function, thus affecting the training efficiency (University of Sydney, 2025).

CNN

The CNN model can automatically extract local features from a two-dimensional spatial structure. Therefore, we keep the spatial structure unchanged and use the original color image (28, 28, 3) data directly as input. In CNN, due to the presence of a multi-layer stacking structure, input pixel normalization can improve the stability and convergence speed of the training process (Ioffe & Szegedy, 2015).

Validation Set Design for Different Models

Random Forest: For Random Forest, we use GridSearchCV combined with 5-fold cross validation for hyperparameter tuning. This method divides the training data into 5 parts, using 4 parts for training each time and the remaining 1 part as the validation set, and finally taking the average of the entire evaluation result, so there is no need to manually divide the validation set.
MLP and CNN: Due to the high cost of neural network training, it is more efficient to use only a one-time validation set to adjust parameters and monitor the training process rather than performing a full cross-validation for each set of hyperparameters. We independently allocate 10% of the training set to prevent overfitting (e.g. terminate training early with early stopping), select optimal hyperparameters, and monitor loss and accuracy changes. This ensures that the test set is completely independent of the entire training process and is only used for testing the final model to prevent information leakage.

Methods

Theory

Random Forest

Random forest combines multiple decision trees together. Each time the data set is randomly selected with replacement, some features are randomly selected as input, and the majority of classification results are selected as the final result. As an ensemble learning algorithm, it has better

prediction performance than a single estimator. Compared with other algorithms, it is easier to use because it has a more mature and well-encapsulated implementation, does not require complex network structure design or a lot of hyperparameter tuning, and has higher tolerance to overfitting and outliers (Horning, 2010). However, the disadvantage is that if the data set is large, it will lead to longer prediction time. And they are prone to overfitting the training data, which may produce poor results when the model is applied to the full data set. This happens when the tree becomes too large and the terminal nodes only represent a very small subset of the training data. (Horning, 2010). The reason we chose the random forest model is that compared with MLP and CNN, the training process of the random forest model is easier, and it can be trained and obtained quickly. The best model parameters can be determined by using grid search in terms of parameters. In addition, because the random forest model cannot handle spatial information, once the prediction results are not good, it is obvious that the importance of spatial feature extraction can be drawn.

MLP

MLP maps input vectors to output vectors through a series of interconnected layers. It mainly consists of three parts: an input layer, one or more hidden layers, and an output layer. Each neuron (except the input layer) uses a nonlinear activation function (Goodfellow et., 2016). Each neuron calculates the weighted sum of its inputs and applies a nonlinear activation function, such as ReLU. The most common method is to use the backpropagation algorithm. The MLP model first learns, then uses weights to store data, and uses algorithms to adjust weights and reduce bias during training, that is, the error between actual values and predicted values (Argonot, 2020). The main advantage is that MLP has the characteristics of nonlinear mapping, which enables it to handle complex relationships in the data and perform well for large data sets. However, MLP has the problem of easy overfitting and is very sensitive to parameter settings. Factors such as the number of hidden layers, the number of neurons in each layer, the learning rate, and the dropout rate will significantly affect performance. If the network is too deep or lacks regularization, overfitting will occur. Improper selection of learning rate will also lead to slow model convergence or unstable training.

CNN

CNN is mainly used to extract features from grid-like matrix datasets. This is particularly useful for visual datasets such as images or videos. It extracts local features of the input data through convolution operations, and forms complex feature representations through multiple layers of convolution and pooling operations, and finally performs tasks such as classification or regression through fully connected layers. The advantage of CNN is that it is good at detecting patterns and features in images, videos, and audio signals. (Goodfellow, Bengio, & Courville, 2016) It is robust to

translation, rotation, and scaling invariance, and can process large amounts of data and achieve high accuracy without manual feature extraction. Because the CNN model automatically extracts features from the data through the convolution layer and can automatically find patterns in the image that can help classification or recognition, such as edges, shapes, etc., from simple to complex through the convolution kernel layer by layer. Pooling further enhances position invariance because it only cares about the maximum value of the local area. The disadvantage of the CNN model is that the training is computationally expensive and requires a lot of memory, because the convolution layer and pooling layer by layer will form a very deep network, resulting in huge computational complexity. If the data is insufficient or proper regularization is not used, it may be affected by noise and perform poorly on the test set, so Dropout and data augmentation are needed to prevent overfitting. (GeeksforGeeks,2025).

Architecture and hyperparameters

Hyperparameter Turning Strategy

Algorithm	Search Strategy	Validation Mechanism
Random Forest	GridSearchCV	5-fold cross validation
MLP	Keras Tuner – RandomSearch	Validation set (train/valid split) + EarlyStopping
CNN		

Table 4

Random Forest

Initially, we built a Random Forest baseline model with default parameters without any hyperparameter tuning, which served as a control for subsequent parameter tuning models, and also verified whether the data preprocessing process was correct and ensured that the model could run and evaluate smoothly. The accuracy of the model on the test set was 65%.

Hyperparameters Tuning and Design Choice

Hyperparameter	Range	Description
n_estimators	[50, 100, 200]	Determines the number of trees
max_depth	[10, 20, 30]	Limits the maximum depth of each tree
min_samples_split	[2, 5, 10]	Specifies the minimum number of samples required to split an internal node
max_features	['sqrt', 'log2']	Controls the number of features to consider when finding the best split

Table 5

The hyperparameters we choose are shown in the table:

• The larger the number of **n_estimators**, the more stable the model will be, preventing accidental results from affecting the model, but increasing the computational cost.

• **max_depth** prevents the model from overfitting by limiting the tree depth.

• **min_samples_split** can prevent the internal nodes from being split too finely.

• **max_features** is to increase the diversity between trees. The default value for classification tasks is "sqrt" to strike a balance between performance and randomness, while the role of "log2" is to further enhance randomness and tree diversity when there is a risk of overfitting, thereby potentially improving generalization.

MLP

Using the most basic model that only contains 2 layers of Dense, we quickly built a network with fixed structure and fixed parameters as a baseline. We did not adjust the learning rate, add Dropout, or use EarlyStopping, which enabled fast training. After 10 epochs on the training set, this simple baseline model achieved a test accuracy of about 53%, which served as an evaluation benchmark for the advanced models after subsequent hyperparameter tuning.

MLP Architecture

Layer	Description
Input	Input shape (28, 28, 3)
Flatten	Flatten the image into a 1D vector of 2352 dimensions
Dense (units_1)	First hidden layer, number of units is tunable
Dropout	Dropout layer to prevent overfitting
Dense (units_2)	Second hidden layer, number of units is tunable
Dropout	Another Dropout layer
Dense (9)	Output layer for 9-class classification with softmax activation

Table 6

Hyperparameters Tuning and Design Choice

Hyperparameter	**Search Space**
Units in Hidden Layer 1 (units_1)	Start from 256, each time add 64 until 1024
Units in Hidden Layer 2 (units_2)	Start from 128, each time add 64 until 512
Dropout Rate	[0.2 - 0.5], step=0.1
Learning Rate (lr)	[0.0001 - 0.01], Sampling with logarithmic scale
Epoch	Max = 30 (with EarlyStopping)

Table 7

- **Units in Hidden Layer 1** (units_1) is to adjust the network capacity. Larger units are more suitable for complex patterns, but there is a risk of overfitting.
- **Units in Hidden Layer 2** (units_2) controls the feature abstraction depth. Too few means underfitting, and vice versa, there is a risk of overfitting.
- **Dropout Rate** represents the regularization strength. The higher the regularization strength, the more effective it is in preventing overfitting, but if it is too high, it will also lead to underfitting. By adjusting this parameter, you can find the best balance between learning ability and generalization.
- **Learning Rate** (lr) is low, which means the model is stable but the learning speed is slow, and high means fast but easy to diverge.
- **Epoch** is to set a round to leave enough training time, and prevent overfitting through EarlyStopping.

CNN

A baseline model is also set up, which contains two convolutional layers (followed by a max pooling layer), a flattening layer, a fully connected layer, and an output layer. No parameters such as the learning rate are adjusted, and no Dropout or EarlyStopping is used. After training on the training set for 10 epochs, an accuracy of about 77% is achieved on the test set, which serves as a comparison reference for the subsequent hyperparameter tuning and architecture optimization of the CNN model.

CNN Architecture

Layer	Description
Input	Input shape (28, 28, 3)
Conv2D (filters)	First convolutional layer, extract low-level local features in the image
MaxPooling2D	Reduce spatial dimensions
Dropout	Dropout layer to prevent overfitting in the first block
Conv2D (filters *2)	Second convolutional layer, extract mid-level features with increased filters
MaxPooling2D	Further spatial reduction
Dropout	Dropout layer to prevent overfitting in the second block

Conv2D (filters *2)	Third convolutional layer, capture deeper abstract features, no pooling applied, to retain more spatial information for the integration of subsequent fully connected layers
Flatten	Flatten the image into a 1D vector
Dense (filters *2)	Fully connected layer to integrate global spatial features
Dropout	Dropout layer applied after the fully connected layer to reduce overfitting by preventing reliance on specific neuron activations before final output
Dense (9)	Output layer for 9-class classification with softmax activation

Table 8

Hyperparameters Tuning and Design Choice

Hyperparameter	Range
Filters	[32, 64, 128]
Dropout Rate	[0.2 - 0.5], step=0.1
Learning Rate (lr)	[0.0001 - 0.01], sampling=log
Epoch	Max = 30 (with EarlyStopping)

Table 9

• **Filters**: Controls the number of channels in each convolutional layer, affecting the feature extraction capability. Larger filters (such as 128) can extract richer features, but will increase the computational cost.

• **Dropout Rate**: Prevents overfitting. Too low a value cannot effectively regularize the model, while too high a value will result in a decrease in the model's learning capability.

• **Learning Rate**: Controls the optimizer update step size. CNN training is deeper, and lower learning rates perform better in parameter adjustment and make training more stable. Higher learning rates can easily lead to non-convergence or poor verification results.

• **Epochs**: Set to 30 times, with EarlyStopping to ensure that training is not over-trained, and that there are enough rounds to search for the optimal model.

Comparison of Strengths and Weaknesses

Comparison of the relative advantages and disadvantages of the models from a theoretical perspective and in combination with the situation of this data set:

Model	Strengths	Weaknesses
Random Forest	**Easy parameter tuning:** Compared to MLP, RF requires fewer hyperparameter tuning and is therefore easier to implement and optimize (GeeksforGeeks, 2023). **Simple training:** RF models are usually easier to train than CNN, especially on smaller datasets, because they have integrated features and do not require extensive feature engineering (Restackio, 2025). In this task, RF only needs to grid search for optimal parameters and train, while CNN needs to perform multiple rounds of iterative training with a validation set to check model performance.	**Lack of spatial feature extraction:** Unlike CNN, RF cannot capture the spatial hierarchy in the data, which limits its performance on image-related tasks. Therefore, for this image task, we first need to convert the two-dimensional data into one-dimensional, which will lose a lot of information. **Interpretability:** Although RF models are more interpretable than deep neural networks, they are still complex due to the collection of multiple decision trees (GeeksforGeeks, 2023). The algorithm's process is still invisible.
MLP	**Modeling nonlinear relationships:** MLPs are able to model complex nonlinear relationships and have greater flexibility than RFs in capturing complex patterns (CSDN, 2025). Image classification in this task clearly has complex relationships, so MLPs are suitable. **General applicability:** MLPs can be applied to a variety of data types, which makes them more general than CNNs that specialize in grid-like data (CSDN, 2025)	**Overfitting risk:** MLPs are prone to overfitting, especially when data is limited, and require careful regularization (CSDN, 2025). This is not obvious in this task because the dataset is large enough. **Computational complexity, affected by construction:** Training MLPs can require a lot of computation, and compared to RF, they may not handle high-dimensional data well (CSDN, 2025). In this dataset, since we have a simpler MLP construction, the running time is less than RF, but the accuracy is lower than RF.
CNN	**Spatial feature learning:** CNN is good at learning spatial hierarchical structures and performs better than RF and MLP in image classification tasks (Wikipedia, 2025). It is	**Large data requirements:** CNNs usually require a lot of labeled data to achieve optimal performance, more than RF or MLP (Wikipedia, 2025).

	very suitable for the dataset of this image classification**.**	**Long running time:** CNN training is more complex and requires a lot of computing resources, so in most cases it takes longer to run than MLP and RF

Table 10

Overall, although random forest and MLP have advantages in computational efficiency and model complexity and are suitable for scenarios where efficiency is the priority, CNN achieves the best performance in this task by virtue of its stronger feature extraction and classification capabilities in image data, although it has higher computational resource requirements and greater optimization challenges. Therefore, in practical applications, the specific choice of which model to use needs to consider different data, computational costs, and the ultimate goal to be achieved (higher efficiency or higher accuracy, etc.).

Results and Discussion

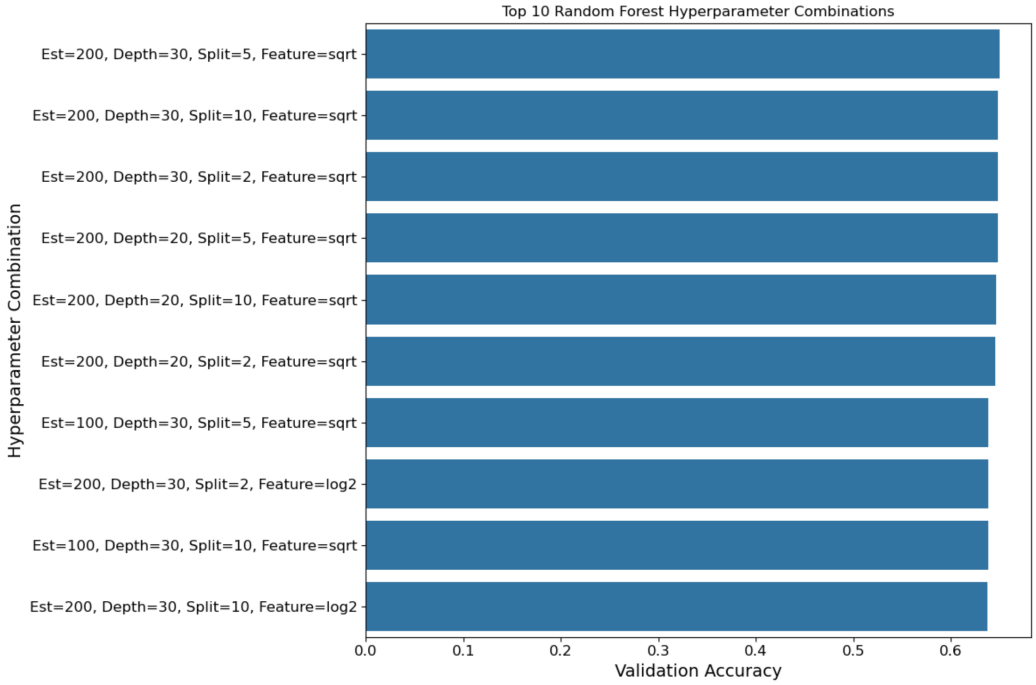
Hyperparameter Tuning Results

Model	Best Hyperparameters	Tuning Time
Random Forest	n_estimators=200, max_depth=30, min_samples_split=5, max_features='sqrt'	100 min
MLP	units_1=448, units_2=320, dropout_rate=0.2, lr=0.0001	22 min
CNN	filters=128, dropout_rate=0.3, lr=0.000136	73 min

Table 11 : The best Hyperparameters of each model

Hyperparameter Tuning Discussion

Random Forest

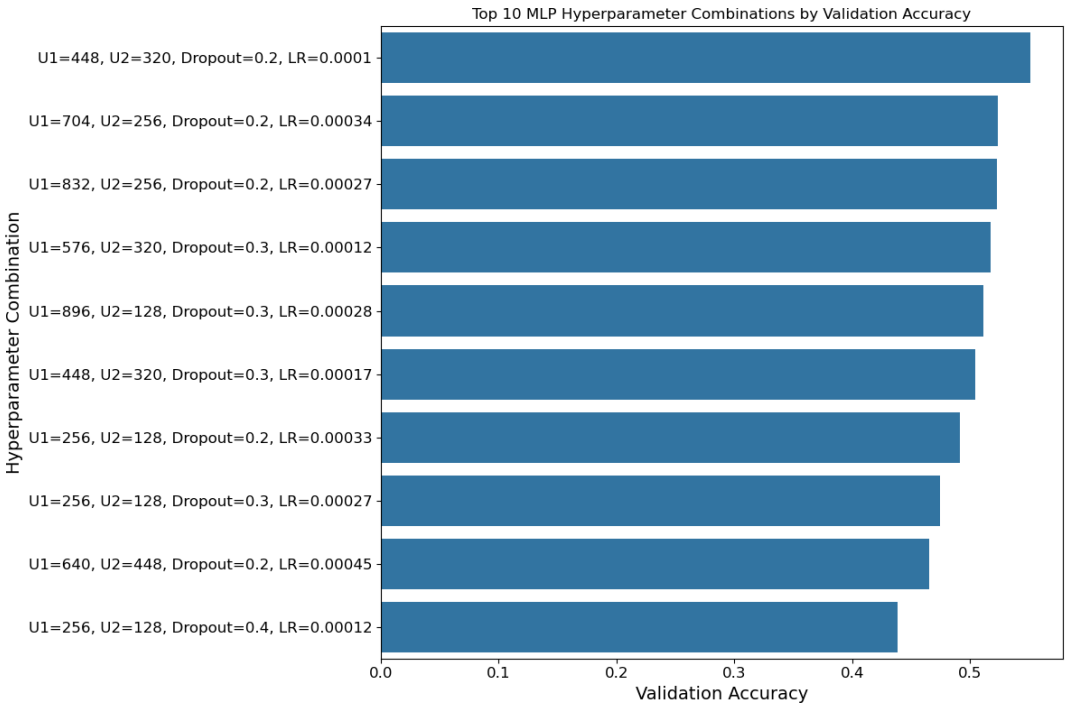


Picture 4

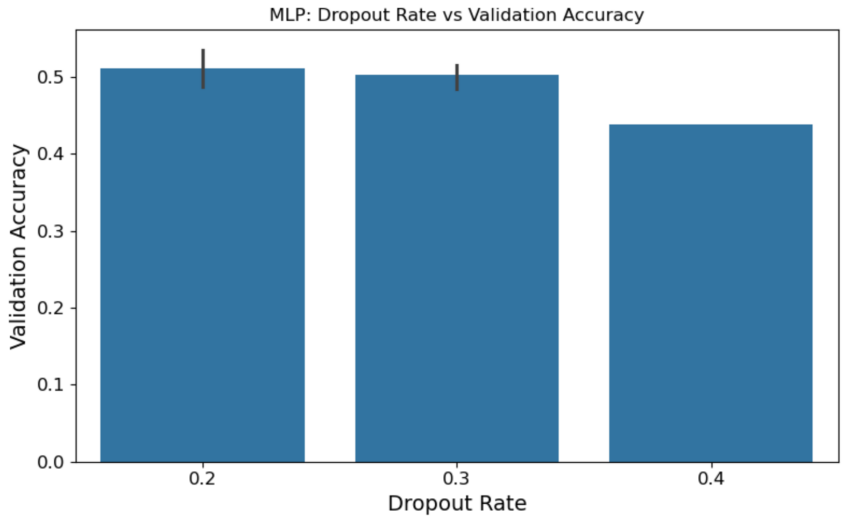
The picture above shows the top 10 parameter combinations ranked by validation accuracy. We found that larger *n_estimators* and deeper *max_depth* improve accuracy, indicating that more trees generally improve model performance and deeper trees can capture more complex features. For *min_samples_split*, all three tested values (2, 5, 10) appear in the top ranking combinations. The best performing model uses 5, but the results for 2 and 10 are also close, indicating that model accuracy is relatively unaffected by this parameter, but slightly biased towards 5. *max_features='sqrt'* consistently outperforms log2, which suggests sqrt is more effective at selecting informative features per split in this context.

Although increasing *n_estimators* and *max_depth* can improve accuracy, the benefits taper off beyond a certain point. Because more complex models take longer to compute, the performance improvement from *n_estimators*=100 to 200 is not significant. Using log2 features can reduce training time, but at the cost of reduced accuracy, so you need to choose whether to convert parameters based on actual needs.

MLP

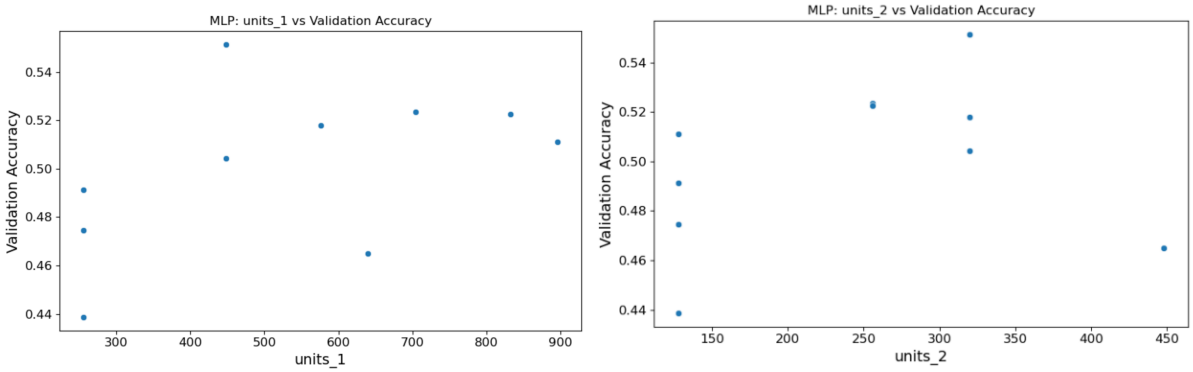


Picture 5

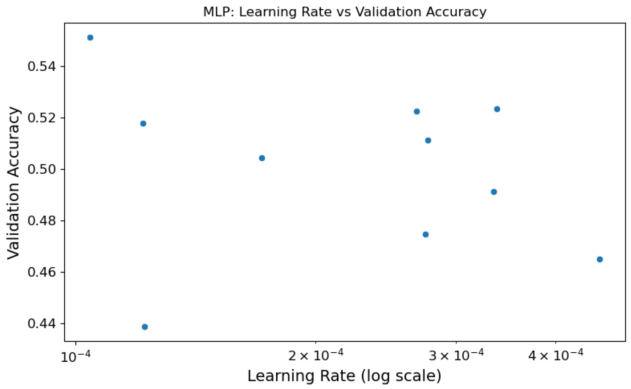


Picture 6

As shown from Picture 6, the performance of the combination with *Dropout=0.4* is significantly worse than that of the combination with *Dropout=0.2 and 0.3*. The model with a smaller *dropout rate (0.2)* achieved the best performance, indicating that the network is not prone to overfitting under the current data and model size, while stronger regularization *(dropout=0.3, 0.4)* will reduce the performance of the model.



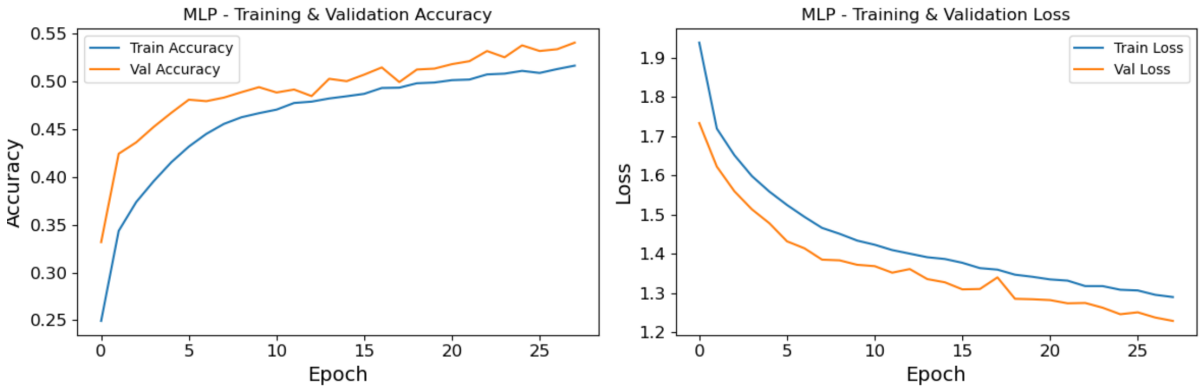
Picture 7 & 8



Picture 9

From Picture 7 and 8, we can see that the model performs best when *units_1 is 448,* and then there are fluctuations but the improvement is not obvious, which shows that the model is sensitive to the number of neurons in the first layer, but the more the better. The trend of *units_2* is relatively unstable, but it can be seen that the performance decreases when the number is too small (approximately less than 130) or too large (450), and the model performs better in the range of 250-330. Therefore, moderately increasing the number of hidden layer neurons can help improve the learning ability of the model, **but too many hidden layer neurons can easily lead to overfitting and high computational overhead.**

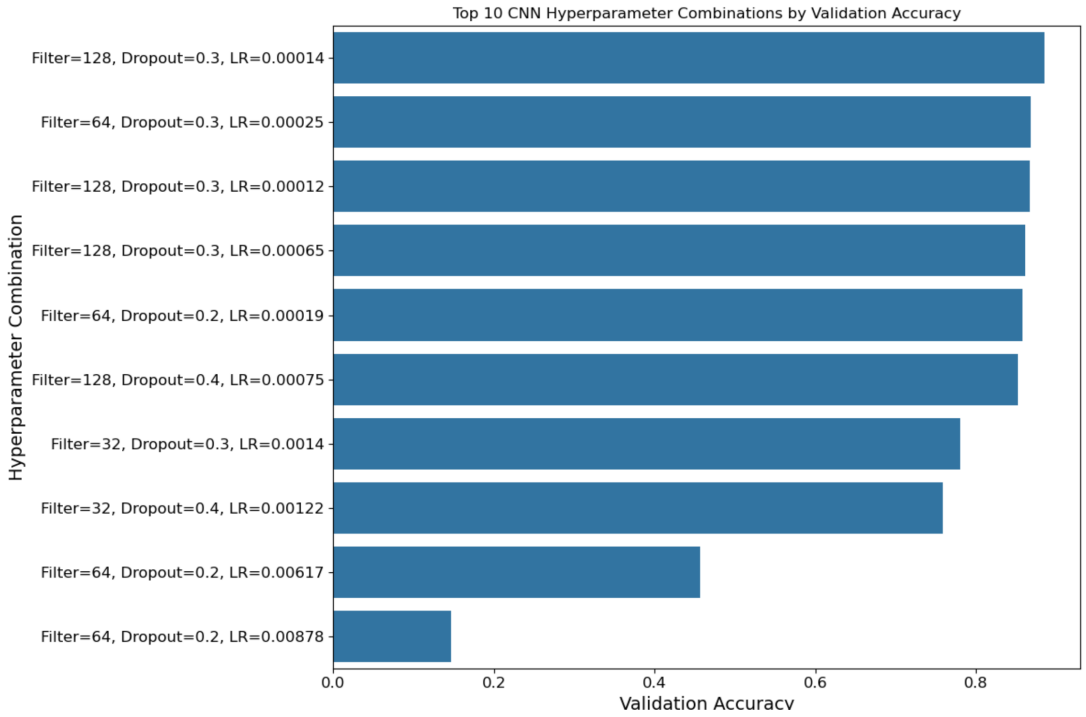
Picture 9 shows that a higher learning rate will lead to unstable model training and reduced accuracy, while a too low learning rate may slow down the convergence of the model or fall into a local optimum. In this task, **the best verification accuracy of the model appears near a smaller learning rate.**



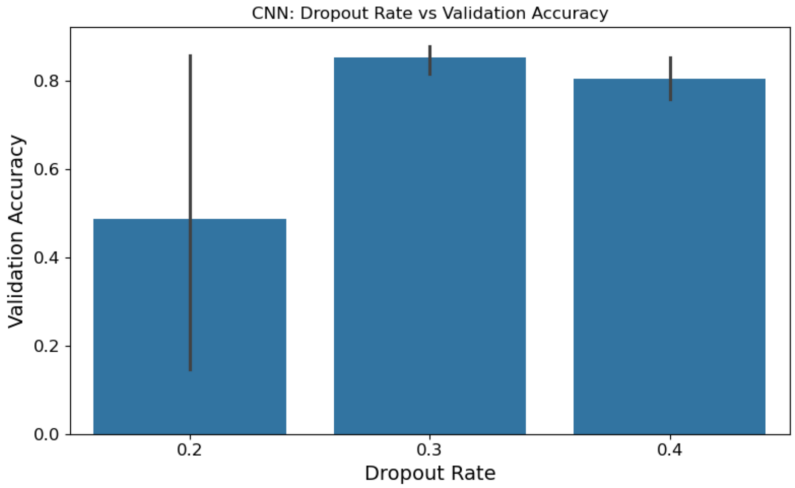
Picture 10

The overall trend of the MLP model's accuracy on the training and validation sets is consistent (Picture 10), and there is **no obvious overfitting**. Both the *training loss* and the *validation loss* continue to decrease, indicating that the model maintains a relatively good optimization state throughout the training process. However, the validation accuracy is slightly higher than the training accuracy most of the time. This may be due to the unstable training caused by *Dropout*, or it may be that the model training depth is not enough and all features are not fully learned. In the future, you can try to expand the parameter search range to improve model performance**.**

CNN

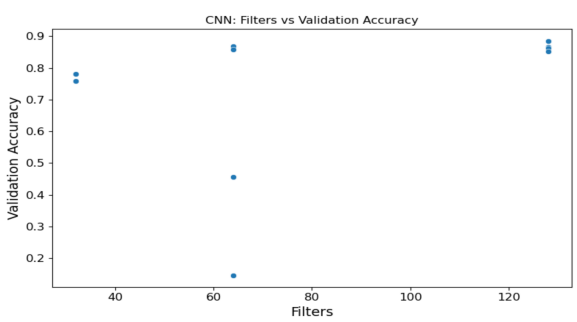


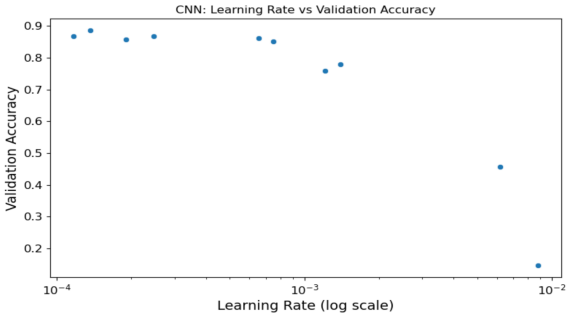
Picture 11



Picture 12

Picture 12 shows that the **CNN model performs best when the *Dropout is 0.3***, indicating that moderate *Dropout* helps improve the generalization ability of the model; too low a *dropout (0.2)* makes the model unstable and the error fluctuates widely; and although good results can be obtained when the Dropout is 0.4, it is slightly lower than that when it is 0.3, which may be due to excessive discarding of neurons, resulting in a decrease in the model's learning ability.



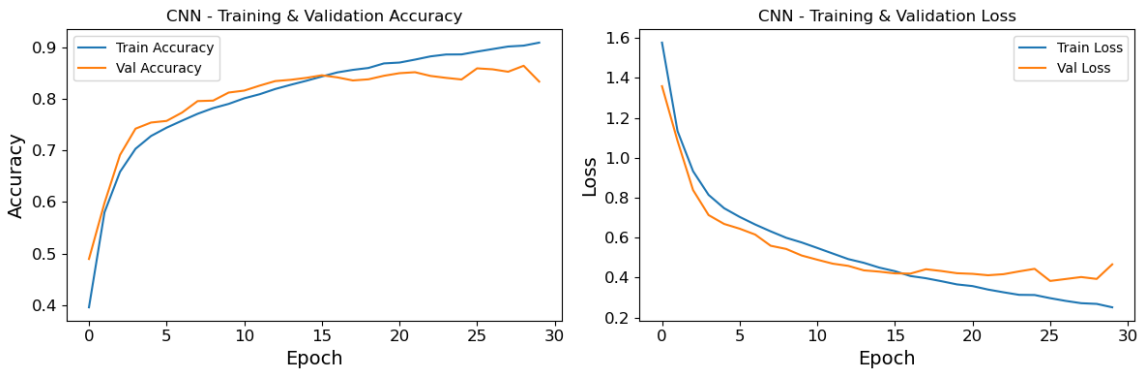


Picture 13 & 14

As can be seen from Picture 13, the validation set accuracy is **the most stable and high when the number of filters is 128**, indicating that a larger number of convolution kernels helps the CNN model extract richer image features. Although 64 filters can also achieve a high accuracy, the fluctuation is large, indicating that the model is more sensitive to initialization or data partitioning at this scale and is not suitable for selection as the optimal parameter. When the number of filters is 32, the overall accuracy is low, which may be due to insufficient extraction capabilities.

The learning rate in Picture 14 shows a very obvious trend. As the learning rate increases, the accuracy continues to decrease, indicating that the model updates too quickly, resulting in unstable training. As expected, a higher learning rate is prone to non-convergence or poor verification results.

Combined with the above analysis, **the CNN model can achieve better performance at high capacity and medium to small learning rates.**



Picture 15

The accuracy of the CNN model increases with the increase of *Epoch*, while the loss decreases (Picture 15). After *Epoch=15*, the accuracy of the model validation set begins to gradually **decrease and is lower than the accuracy of training**, indicating that the performance of the model in the validation set is worse than that in the training set, which is a typical overfitting feature. To solve this problem, we **reduced the epoch from 30 to 18 when building the final model to improve its generalization ability while ensuring the performance of the model.**

Results Discussion

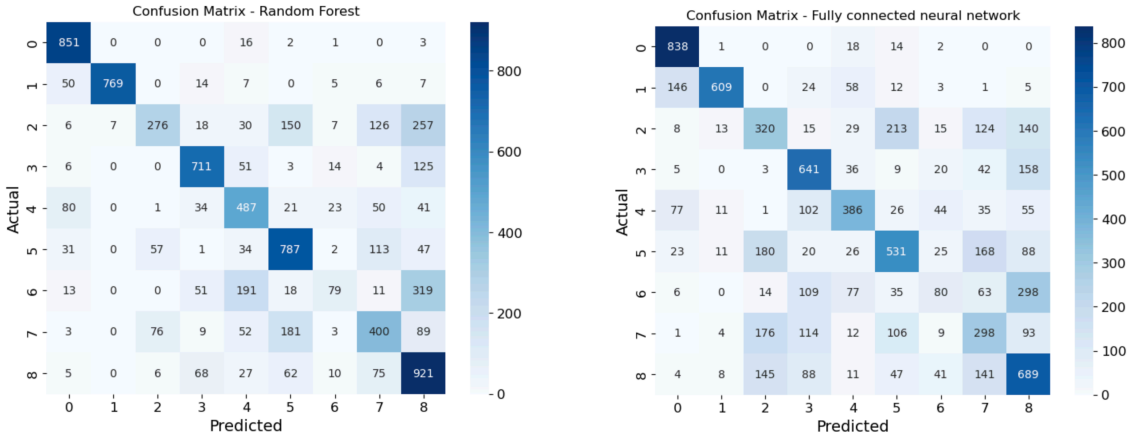
Result Table

Model	Accuracy	Macro F1-score	Runtime	Trainable Parameters
Random Forest	0.66	0.63	128s	Non-parameter model (Based on the tree structure)
MLP	0.55	0.53	115s	1,200,713
CNN	0.85	0.85	571s	1,481,225

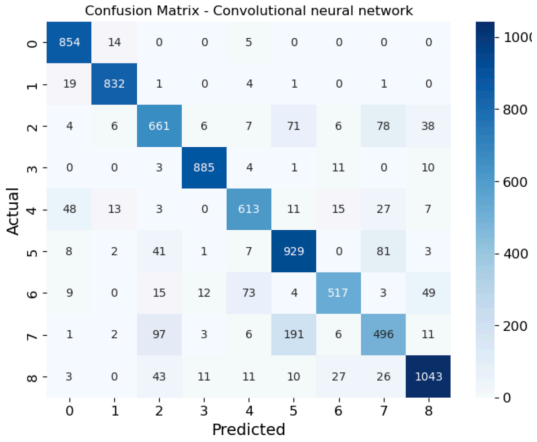
Table 12

The runtime data comes from the time it takes to train the final model with the optimal parameters and the complete training set. Random forest is manually counted, about 128 seconds. MLP has a simple structure and epoch = 28, and the time per epoch is 4~5 seconds, a total of 115 seconds. CNN contains multiple layers of convolution and pooling structures and a large number of parameters. Epoch = 18, the time per epoch fluctuates around 30 seconds, and the training time is 571 seconds, which is much higher than other models.

Model Confusion Matrix Analysis



Picture 16 & 17 Random Forest (left), MLP (right)



Picture 18 CNN

By comparing the confusion matrices of the three models, we can find that there are certain differences and similarities in the recognition of different categories by various models：

- **Class 6 and Class 7 (the most difficult classes to distinguish):** Judging from the image samples, the textures of Class 6 and Class 7 are more complex, the color transition is fuzzy, and they contain irregular light and dark contrasts.
    - **Random Forest**: Class 6 was seriously misclassified as Class 4 (191), and Class 7 was often confused as Class 5 (181), indicating that the model was weak in distinguishing the boundary between the two categories
    - **MLP**: Class 6 had the worst classification ability and was almost not correctly identified (only 79 were correctly classified)
    - **CNN**: Class 7 was easily misclassified as Class 5. Although there were still misclassifications, the recognition was more stable, indicating that CNN was better at processing such complex and easily confused images.

- **Class 0 and Class 1 (easily identifiable categories):** In the sample, Class 0 images have clear structural contours and graphic features that are clearly different from other classes, and Class 1 image patterns are relatively stable, so these features are easy to identify for the image model.
    - **CNN performs best** (854 Class 0 are correctly identified), and Random Forest can also distinguish them relatively accurately;
    - **MLP is slightly weaker in identifying Class 1** (609 are correctly identified), and is easily confused with Class 0, reflecting that MLP is unstable when dealing with classes with fuzzy boundaries and subtle texture differences.

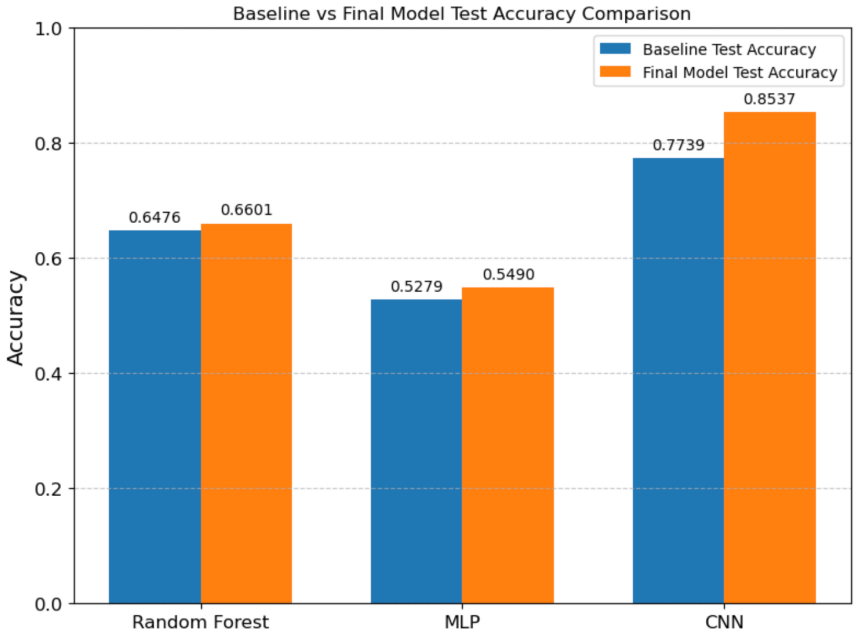
Model Comparison

Model	Strengths	Weaknesses
Random Forest	**Low parameter adjustment complexity:** Compared with CNN and MLP, parameter adjustment only needs to use the grid search method to find the best parameters. **High efficiency:** One-time training, no repeated optimization, the most complex parameter training time is about 100s, which is more efficient than CNN and MLP.	**Unable to extract spatial information:** Compared with CNN, it lacks the ability to extract spatial features and has limited effect in image classification, but it still has certain recognition ability by relying on the integration of a large number of tree models. (Macro F1-score = 0.63, accuracy = 0.66). **Limited room for improvement:** Even if the parameters are adjusted and optimized, the effect is difficult to improve significantly, and the generalization ability is weak.
MLP	**Moderate training efficiency:** A single round of training takes 4–5 seconds, and the total training time is significantly shorter than CNN. A simpler architecture was used in this task, so the efficiency is moderate. **Certain feature learning ability:** Compared with random forest, MLP has certain nonlinear feature learning ability.	**Lack of spatial structure modeling capability:** Compared with CNN, MLP cannot utilize the spatial relationship between image pixels, resulting in low classification accuracy and Macro F1 score (Macro F1-score = 0.52, accuracy = 0.55). **Parameter sensitivity and high tuning requirements:** The model is prone to underfitting or overfitting, and hyperparameters and model structure

		need to be more finely tuned to achieve better performance.
CNN	**Best results:** Using spatial features, the classification results are significantly better (test accuracy 85%, Macro F1-score 0.85) **Automatic feature extraction:** No need to manually design features, and can automatically learn hierarchical patterns.	**The highest training cost:** a single round of training takes 30–40 seconds, the total training time is much longer than RF and MLP, and the tuning architecture design is complex. **High requirements for data and hardware:** a large amount of data and GPU support are required, and the optimization process is time-consuming and labor-intensive.

Table 13

Further Discussion



Picture 19

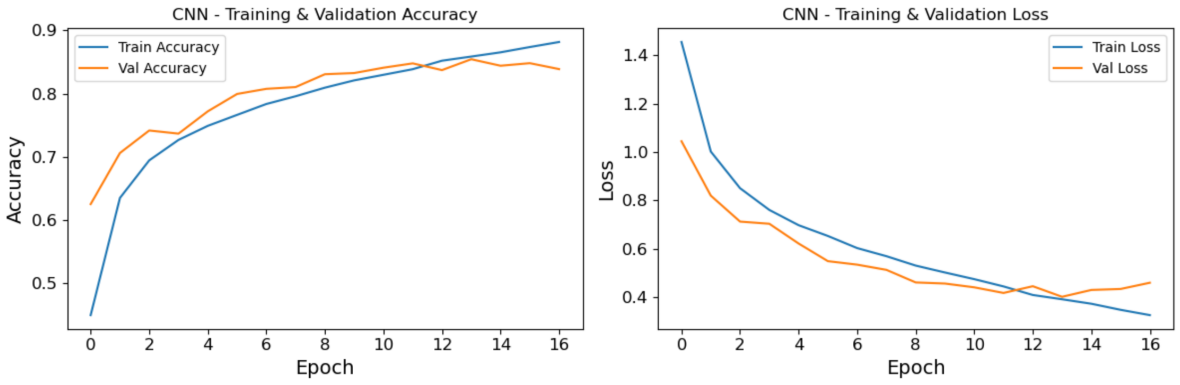
Picture 19 is a **comparison of the accuracy of the baseline and the final model**. It can be seen that the performance of the three models has improved to a certain extent after the parameter adjustment, indicating that the parameter adjustment process is effective. Among them, the performance of the CNN model has improved most significantly, further verifying that CNN is more sensitive to hyperparameters and has stronger expression and learning capabilities in image classification tasks.

Impact of Dropout Placement on CNN Performance

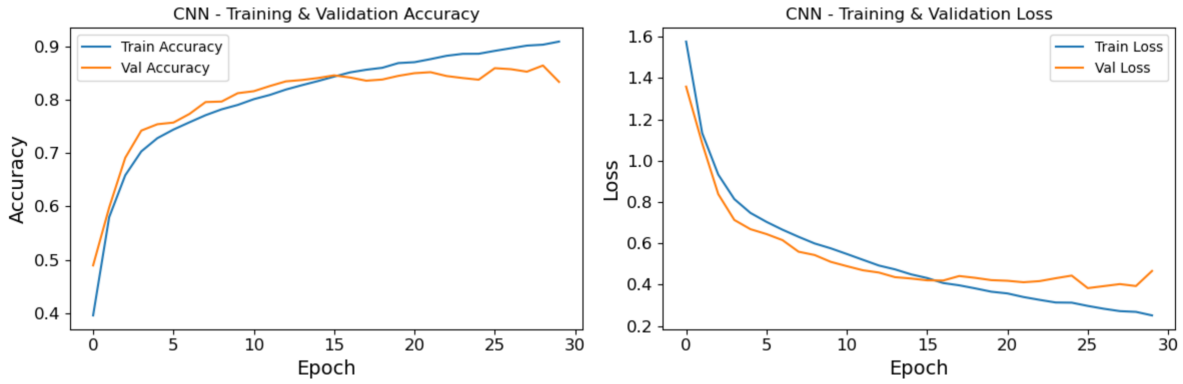
Design Strategy	Dropout Location	Characteristics
1	Add Dropout after each convolutional block	Multi-layer regularization, stronger overfitting prevention; suitable for models prone to overfitting
2	Add only one Dropout before the fully connected layer	Weaker regularization, faster convergence but higher overfitting risk;

Table 14

In this task, we tried two Dropout strategies: **one is to add a Dropout layer after each convolution block**, and the other is to **add Dropout only once before entering the fully connected layer**. The results show that the first method can more evenly regularize the convolution feature extraction process, effectively alleviate the risk of overfitting, and the validation set performance is more stable (Picture 21), because multiple layers of Dropout can prevent the front-layer network from becoming dependent on the training set features too early. The model that only adds Dropout once to the fully connected layer, although it converges faster in the early stage of training, is not as stable as the first model in the validation set performance, and there are obvious fluctuations (Picture 20), indicating that the regularization is insufficient and overfitting occurs. **So in the end we used the first method as the best architecture.**



Picture 20 Dropout applied only before the fully connected layer



Picture 21 Dropout applied after each convolutional block

Conclusion

Summary of Main Findings

Picture 22

The CNN model performs best, giving full play to its advantages in spatial feature extraction and pattern recognition, but it has the highest training cost and has a certain risk of overfitting.

Random Forest is second. Although it does not have the ability to model spatial features, it still achieves good results on non-spatial features with the help of ensemble learning strategies, and is suitable for scenarios with high efficiency requirements.

The MLP model performs relatively poorly. Although it has certain nonlinear modeling capabilities, it cannot fully capture local information in tasks where spatial features are more critical, resulting in the lowest classification accuracy and F1 value, but it has certain selectivity under resource-constrained conditions.

These results are also in line with our theoretical expectations. CNN has a strong ability to extract spatial features and can capture local structures and patterns in images, so it should achieve the best performance in image tasks. MLP cannot model spatial structures in this task, and the input is a flattened vector, which makes it difficult to capture the relationship between adjacent pixels and its performance is limited. Although Random Forest can handle high-dimensional features, it relies on pixel-level tree splitting and is difficult to depict complex image semantic structures, so its performance is medium.

Limitation

Data size and category distribution: Although the PathMNIST dataset has sufficient sample size, the category imbalance in the dataset has not been specifically addressed, which may affect the model's ability to recognize minority classes (e.g., the recognition accuracy of class 6 is generally low).

The CNN model has a certain overfitting phenomenon: Although Dropout and EarlyStopping are added for regularization, the accuracy of the validation set is still observed to decrease in the later stage of training (after Epoch15), indicating that the optimal balance between model complexity and regularization strategy has not yet been achieved.

Future Work Suggestion

In view of the above limitations, future research can be further optimized and expanded from the following directions:

1. **Strengthen the response to the problem of imbalanced categories**: By introducing category weight adjustment, data enhancement or adopting more advanced loss functions (such as Focal Loss) to improve the classification ability of CNN in small sample categories, avoid lowering the overall Macro F1-score due to low recall of minority classes.
2. **Optimize the CNN model architecture and training strategy**: Further explore lightweight network structures (such as MobileNet, EfficientNet) to reduce training costs and prevent overfitting.
3. **Enhance image preprocessing methods**: Use image scaling, enhancement, contrast adjustment and other methods to improve the distinguishability of input features, thereby enhancing the generalization ability of the model.
4. **Expand multi-model fusion methods:** In this project, it was found that CNN performed well in spatial feature extraction, and Random Forest had advantages in global classification and speed. In the future, integrated learning or hybrid model solutions can be tried, such as inputting high-order features extracted by CNN into RF or MLP for classification, which may further improve the comprehensive performance and practicality of the model.

Reflection

Student 1： In this assignment, I designed training and tuning strategies based on the characteristics of different models and used visualizations to understand model structure, performance changes, and overfitting. While building MLP and CNN from scratch, I learned how to design neural networks step by step, including choosing input shape, number of hidden units, learning rate, epochs, and dropout rate. This helped me better understand how neural networks are built. I also realized that data preprocessing (like normalization and channel settings) and hyperparameter choices have a big impact on model performance. By comparing Random Forest, MLP, and CNN on image classification tasks, I learned how to evaluate model performance using accuracy, macro F1-score, and confusion matrix. I also gained a better understanding of the strengths and weaknesses of different models when working with image data.

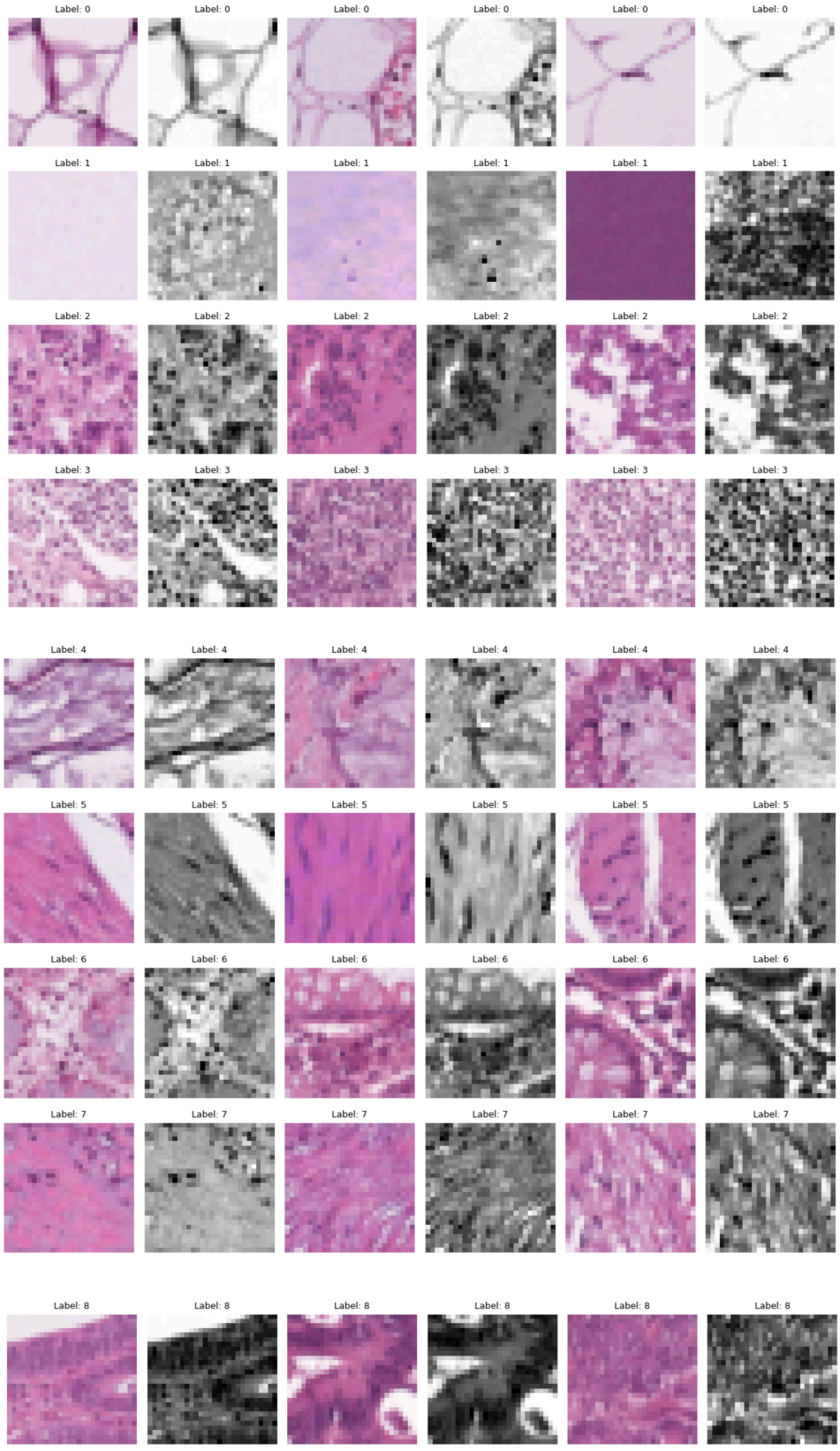
Student 2： Through this assignment, I learned that it is very important to choose the right model when dealing with different types of tasks. For example, in this image classification task, even if we try to adjust the hyperparameters of random forest and MLP, the results are still much different from those of CNN model. This shows that each model has its own type of tasks that it is better at. In practical applications, how to efficiently find a suitable model is also an important step. In the early stage, you need to search for theories, and then preliminarily select a model and test it, so that you can find a suitable model. In terms of model performance, it is not necessarily based on high accuracy, but should be based on the main purpose of the task. If the purpose of the task is to quickly understand the preliminary results, then you should consider the running time of the model more and give up some accuracy of the results.

Reference

1. Argonot. (2020, June 19). *机器学习笔记｜神经网络的反向传播原理及过程（图文并茂+浅显易懂）* [CSDN Blog]. CSDN. https://blog.csdn.net/fsfjdtpzus/article/details/106256925
2. CSDN. (2025, April 7). *多层神经网络（MLP）与卷积神经网络（CNN）的区别.* https://blog.csdn.net/zzq900503/article/details/147040275
3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org/
4. GeeksforGeeks. (2025, April 3). *Introduction to convolution neural network.* https://www.geeksforgeeks.org/introduction-convolution-neural-network/
5. GeeksforGeeks. (2023, February 15). *What are the advantages and disadvantages of Random Forest?* https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-random-forest/
6. Horning, N. . (2010). Random forests : an algorithm for image classification and generation of continuous fields data sets.
7. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)* (Vol. 37, pp. 448–456). JMLR.org.
8. Maharana, K., Mondal, S., & Nemade, B. (2022). A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings, 3*(1), 91–99. https://doi.org/10.1016/j.gltp.2022.04.020
9. Madusanka, N., Jayalath, P., Fernando, D., Yasakethu, L., & Lee, B.-I. (2023). Impact of H&E Stain Normalization on Deep Learning Models in Cancer Image Classification: Performance, Complexity, and Trade-Offs. *Cancers*, *15*(16), 4144-. https://doi.org/10.3390/cancers15164144
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.
11. Shen, D., Wu, G. & Suk, H.-I. Deep learning in medical image analysis. *Annual review of biomedical engineering* **19**, 221–248 (2017).
12. University of Sydney. (2025). *COMP5318/COMP4318 Week 7: Introduction to Keras and Multilayer Perceptrons* [Unpublished tutorial material]. Tutorial notes, COMP5318/COMP4318 Machine Learning and Data Mining, Semester 1.
13. Wikipedia. (2025, May 8). *Convolutional neural network.* https://en.wikipedia.org/wiki/Convolutional_neural_network
14. Yang, J., Shi, R., Wei, D., Chen, L., Zhang, Y., Li, C., ... & Zhou, S. K. (2023). MedMNIST v2 – A large-scale lightweight benchmark for 2D and 3D biomedical image classification. *Scientific Data, 10*(1), 41. https://doi.org/10.1038/s41597-022-01721-8

Appendix

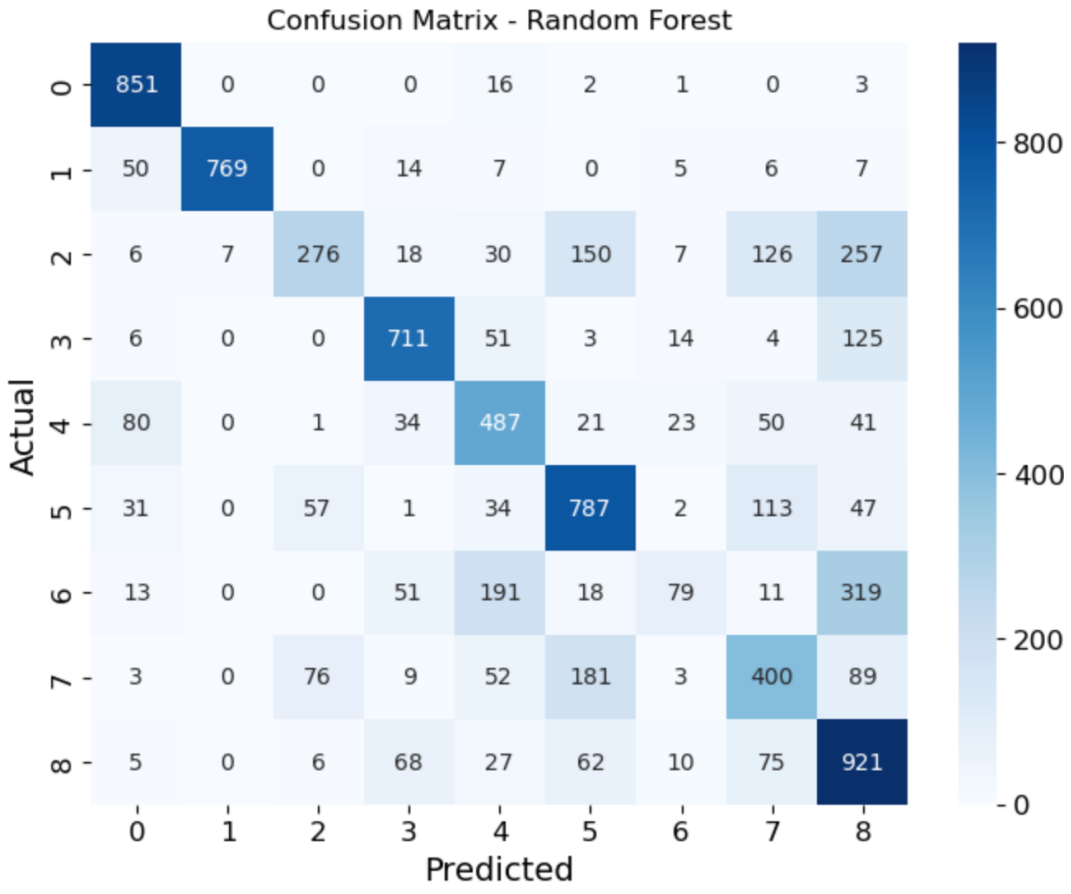
Sample Image



Random Forest – Final model results

Final RF – Test Set Classification Report:

	precision	recall	f1-score	support
0	0.81	0.97	0.89	873
1	0.99	0.90	0.94	858
2	0.66	0.31	0.43	877
3	0.78	0.78	0.78	914
4	0.54	0.66	0.60	737
5	0.64	0.73	0.69	1072
6	0.55	0.12	0.19	682
7	0.51	0.49	0.50	813
8	0.51	0.78	0.62	1174
accuracy			0.66	8000
macro avg	0.67	0.64	0.63	8000
weighted avg	0.67	0.66	0.64	8000



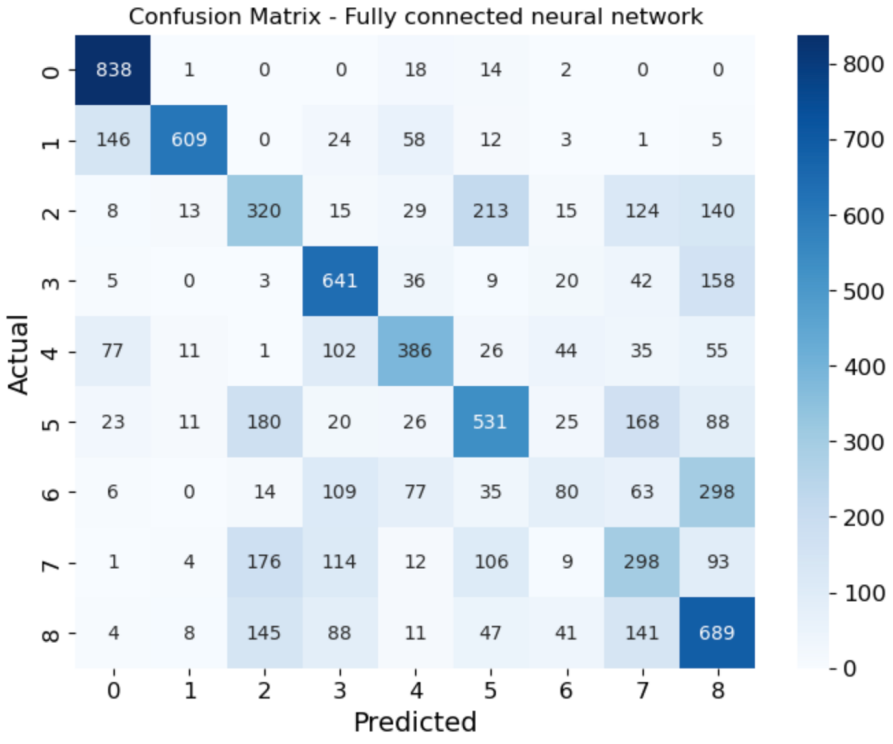
MLP – Final model results

```
Epoch 1/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.2210 - loss: 2.0389
Epoch 2/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.3268 - loss: 1.7498
Epoch 3/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.3718 - loss: 1.6659
Epoch 4/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.3947 - loss: 1.6083
Epoch 5/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4162 - loss: 1.5564
Epoch 6/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4364 - loss: 1.5225
Epoch 7/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4452 - loss: 1.4877
Epoch 8/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4645 - loss: 1.4568
Epoch 9/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4626 - loss: 1.4424
Epoch 10/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4732 - loss: 1.4234
Epoch 11/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4728 - loss: 1.4115
Epoch 12/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4827 - loss: 1.4009
Epoch 13/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4860 - loss: 1.3812
Epoch 14/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4890 - loss: 1.3751
Epoch 15/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4972 - loss: 1.3603
Epoch 16/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4970 - loss: 1.3534
Epoch 17/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5011 - loss: 1.3430
Epoch 18/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5002 - loss: 1.3346
Epoch 19/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5042 - loss: 1.3276
Epoch 20/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - accuracy: 0.5052 - loss: 1.3166
Epoch 21/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - accuracy: 0.5059 - loss: 1.3153
Epoch 22/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5138 - loss: 1.2986
Epoch 23/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5163 - loss: 1.2926
Epoch 24/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5223 - loss: 1.2889
Epoch 25/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5224 - loss: 1.2785
Epoch 26/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5201 - loss: 1.2750

Epoch 27/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5247 - loss: 1.2614
Epoch 28/28
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - accuracy: 0.5268 - loss: 1.2578
```

Layer (type)	Output Shape	Param #
flatten_3 (Flatten)	(None, 2352)	0
dense_8 (Dense)	(None, 448)	1,054,144
dropout_2 (Dropout)	(None, 448)	0
dense_9 (Dense)	(None, 320)	143,680
dropout_3 (Dropout)	(None, 320)	0
dense_10 (Dense)	(None, 9)	2,889

Total params: 3,602,141 (13.74 MB)
Trainable params: 1,200,713 (4.58 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2,401,428 (9.16 MB)

```
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 889us/step
              precision    recall  f1-score   support

           0       0.76      0.96      0.85       873
           1       0.93      0.71      0.80       858
           2       0.38      0.36      0.37       877
           3       0.58      0.70      0.63       914
           4       0.59      0.52      0.56       737
           5       0.53      0.50      0.51      1072
           6       0.33      0.12      0.17       682
           7       0.34      0.37      0.35       813
           8       0.45      0.59      0.51      1174

    accuracy                           0.55      8000
   macro avg       0.54      0.54      0.53      8000
weighted avg       0.55      0.55      0.54      8000
```



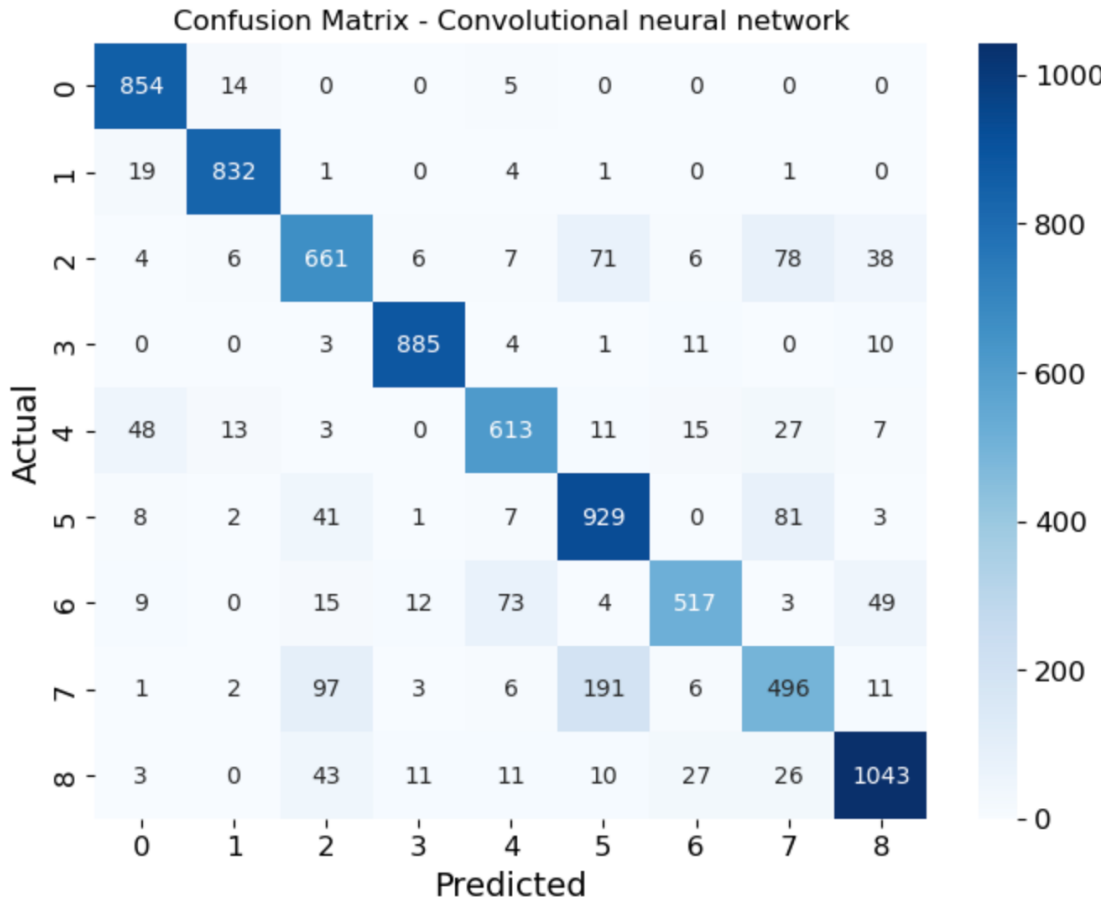
CNN – Final model results

```
Epoch 1/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 30s 29ms/step - accuracy: 0.2993 - loss: 1.8284
Epoch 2/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 32s 32ms/step - accuracy: 0.5642 - loss: 1.1567
Epoch 3/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 32s 32ms/step - accuracy: 0.6474 - loss: 0.9617
Epoch 4/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 33s 33ms/step - accuracy: 0.6934 - loss: 0.8410
Epoch 5/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 35s 35ms/step - accuracy: 0.7240 - loss: 0.7568
Epoch 6/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 34s 34ms/step - accuracy: 0.7448 - loss: 0.6991
Epoch 7/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 33s 33ms/step - accuracy: 0.7631 - loss: 0.6540
Epoch 8/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 34s 34ms/step - accuracy: 0.7736 - loss: 0.6283
Epoch 9/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 35s 35ms/step - accuracy: 0.7846 - loss: 0.5981
Epoch 10/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 34s 34ms/step - accuracy: 0.7966 - loss: 0.5674
Epoch 11/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 33s 33ms/step - accuracy: 0.8024 - loss: 0.5500
Epoch 12/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 31s 31ms/step - accuracy: 0.8151 - loss: 0.5107
Epoch 13/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 29s 29ms/step - accuracy: 0.8208 - loss: 0.4980
Epoch 14/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 29s 29ms/step - accuracy: 0.8312 - loss: 0.4668
Epoch 15/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 29s 29ms/step - accuracy: 0.8325 - loss: 0.4536
Epoch 16/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 29s 29ms/step - accuracy: 0.8445 - loss: 0.4279
Epoch 17/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 30s 30ms/step - accuracy: 0.8484 - loss: 0.4121
Epoch 18/18
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 29s 29ms/step - accuracy: 0.8545 - loss: 0.4039
```

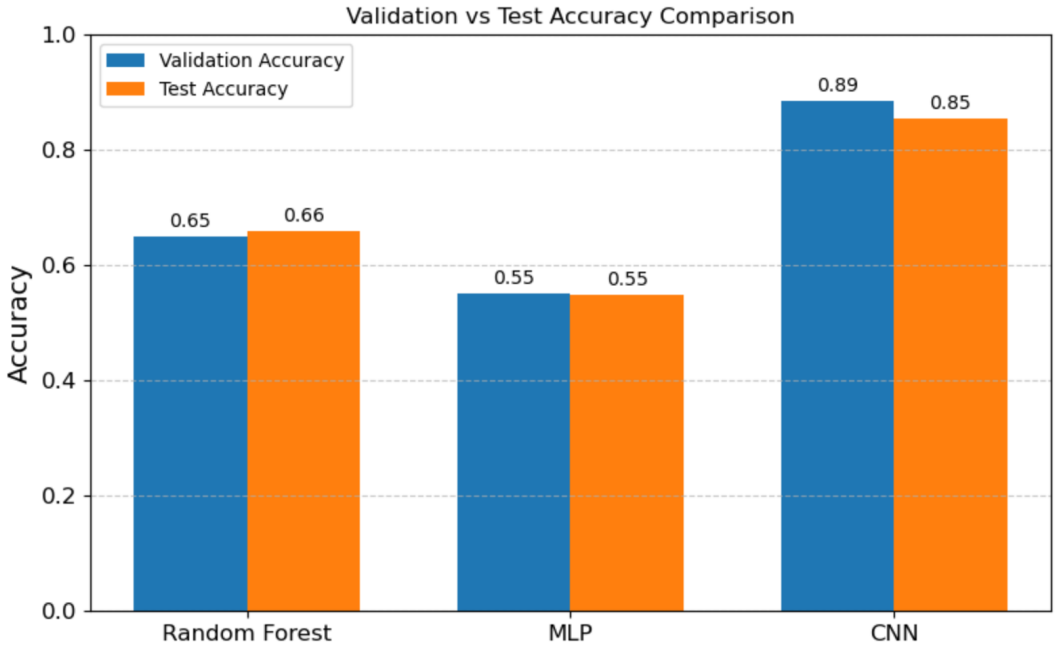
```
250/250 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step
              precision    recall  f1-score   support

           0       0.90      0.98      0.94       873
           1       0.96      0.97      0.96       858
           2       0.77      0.75      0.76       877
           3       0.96      0.97      0.97       914
           4       0.84      0.83      0.84       737
           5       0.76      0.87      0.81      1072
           6       0.89      0.76      0.82       682
           7       0.70      0.61      0.65       813
           8       0.90      0.89      0.89      1174

    accuracy                           0.85      8000
   macro avg       0.85      0.85      0.85      8000
weighted avg       0.85      0.85      0.85      8000
```

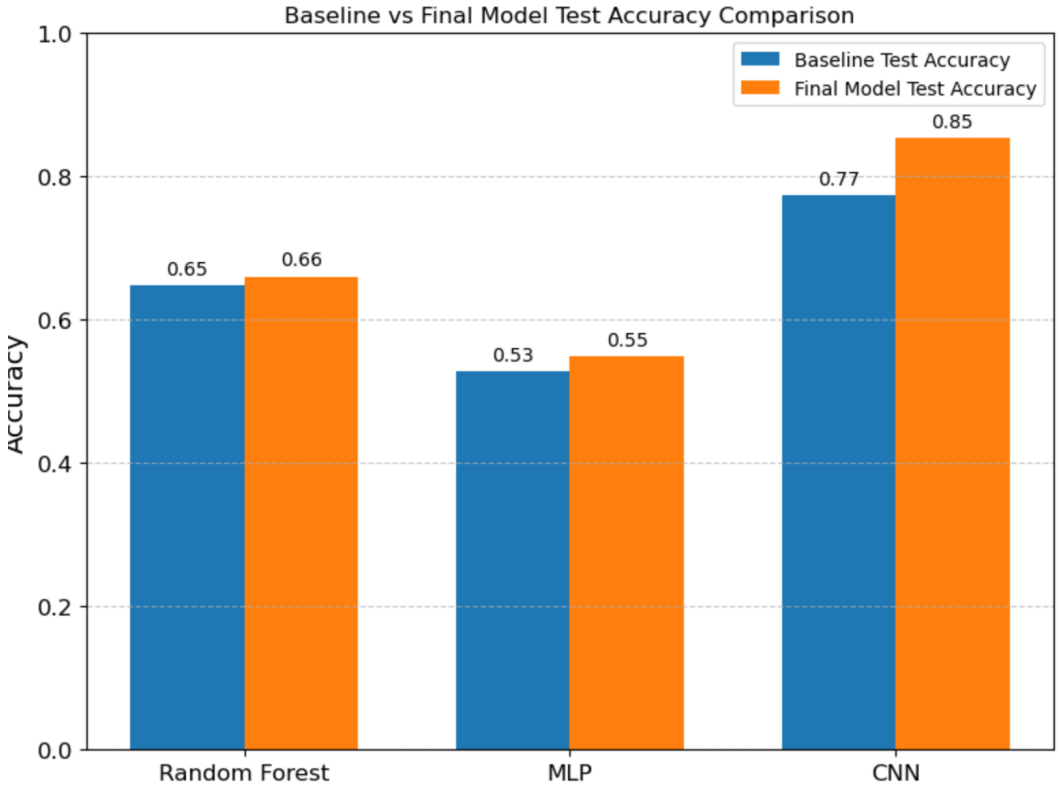
Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 128)	3,584
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 128)	0
dropout_4 (Dropout)	(None, 13, 13, 128)	0
conv2d_3 (Conv2D)	(None, 11, 11, 256)	295,168
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 256)	0
dropout_5 (Dropout)	(None, 5, 5, 256)	0
conv2d_4 (Conv2D)	(None, 3, 3, 256)	590,080
flatten_4 (Flatten)	(None, 2304)	0
dense_11 (Dense)	(None, 256)	590,080
dropout_6 (Dropout)	(None, 256)	0
dense_12 (Dense)	(None, 9)	2,313

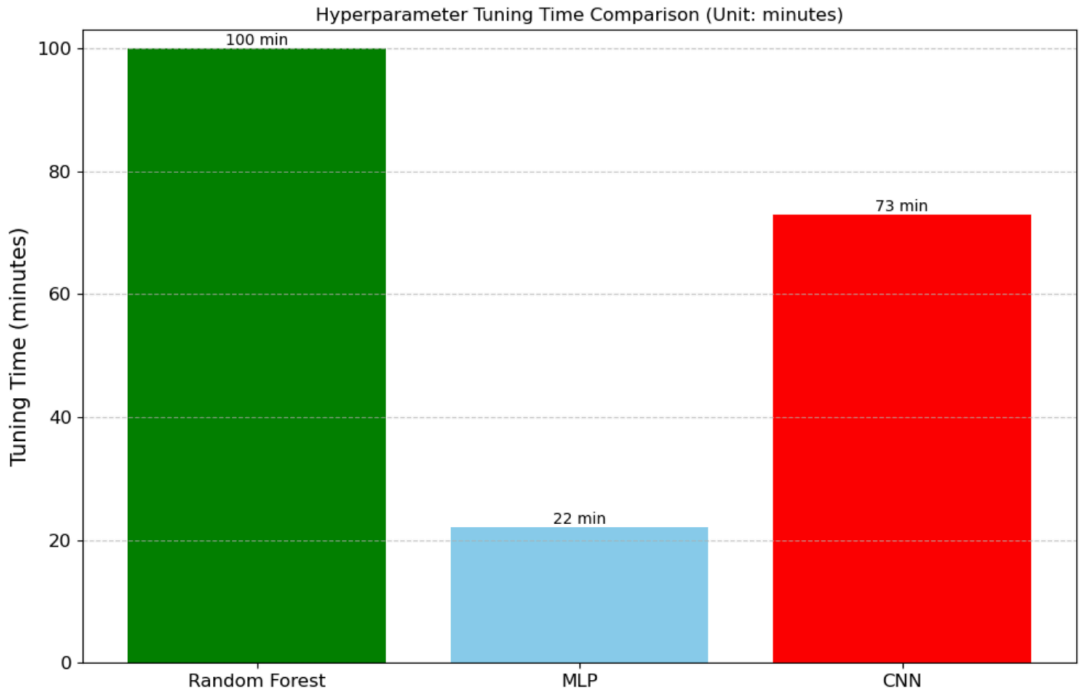
Total params: 4,443,677 (16.95 MB)
Trainable params: 1,481,225 (5.65 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2,962,452 (11.30 MB)



Model comparison






Hyperparameter Tuning Time Comparison (Unit: minutes)
Tuning Time (minutes)
0
20
40
60
80
100
100 min
22 min
73 min
Random Forest
MLP
CNN