Group 328 COMP5318 Assignment 2 Report – PathMNIST Classification

1. Introduction

The purpose of this assignment is to compare and evaluate three different machine learning approaches on the PathMNIST dataset. The models are one classical model (KNN), a multilayer perceptron (MLP), and a convolutional neural network (CNN). This involves not only training and testing each model but also tuning their hyperparameters to ensure fair and optimized comparisons. The report emphasizes the importance of selecting the right algorithm or training model in medical imaging applications with higher accuracy.

This task demonstrates a practical medical image classification scenario using histopathology data, reflecting real-world challenges. Choosing a correct algorithm can significantly impact performance, interpretability, and computational efficiency. Deep learning models, especially CNNs, have proven effective in biomedical image analysis due to their capacity to capture spatial features (Yang et al., 2023). By comparing these models, we can see which model and algorithm work better for this type of medical image data and why. Furthermore, it also shows how much of a difference proper hyperparameter tuning can make when accuracy really matters in healthcare tasks.

2. Data

2.1 Dataset Description

The dataset used in this assignment is **PathMNIST**, a subset of the **MedMNIST v2** collection (Yang et al., 2023). It consists of **28×28 RGB images** of medical microscope images. Moreover, each image belonging to one of **9 classes (label of 0 to 8)** representing different tissue types (both normal and abnormal). The dataset includes a **training set, validation set, and test set**. The class labels are provided as integers from 0 to 8.

2.2 Data Exploration

Include visualizations such as class distribution plots and example images.
Discuss any class imbalance, overlapping features between classes, or other notable patterns and characteristics

To prevent any misunderstanding of the dataset, we explored the distribution of samples per class using a bar plot. While the class distribution was relatively balanced, a few classes had a higher number of samples than others. For example, class 8 and 5 have relatively more samples than other classes. (Figure 1) We also visualized the first 10 example images from different classes. (Figure2) The visualization revealed that several classes share visually similar patterns (e.g., subtle color and texture differences). Furthermore, the first 10 example images' visualization reflects a challenge for simple classifiers. Instead, the need for more powerful feature extraction techniques like convolutional layers has been discovered.

Additionally, we examined the pixel intensity distribution and confirmed that values range from 0 to 255. (Figure3) Moreover, we discovered that the values range is most concentrated in the lower half. This observation supports the need for normalization before model training.

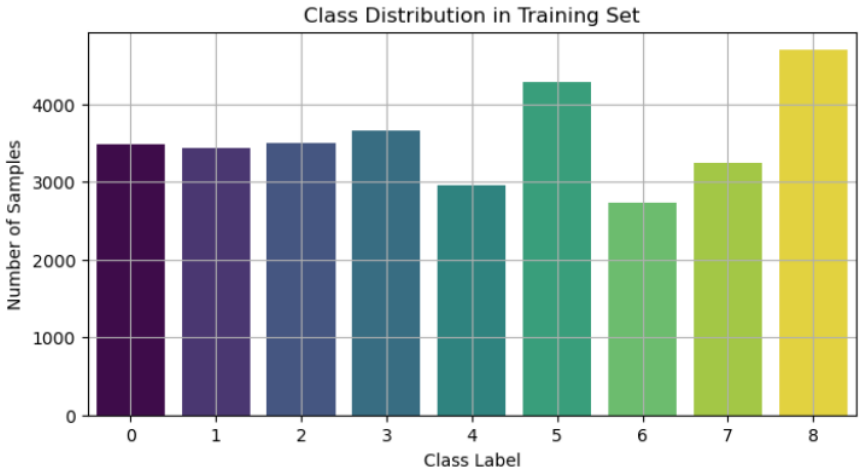


Figure1: Class Distribution in Training Set

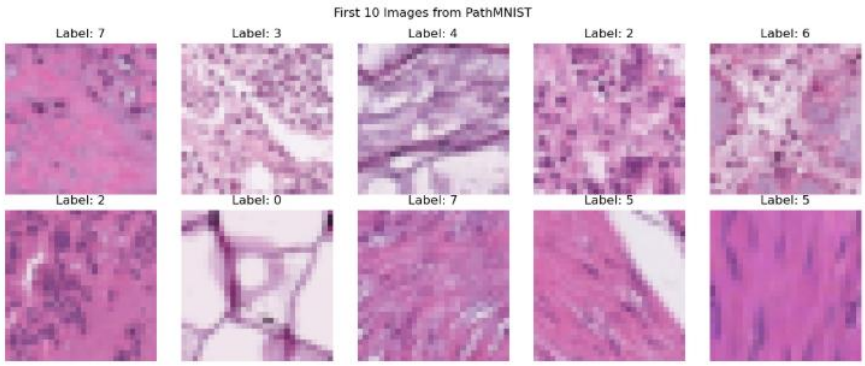


Figure2: First 10 images from PathMNIST

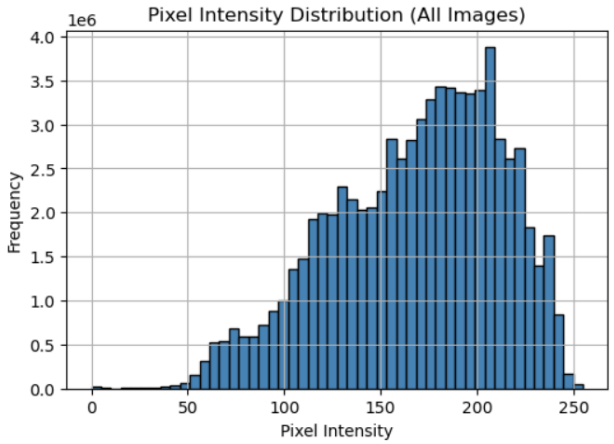


Figure3: Pixel Intensity Distribution for all Images

2.3 Pre-processing

Based on our data exploration and standard practice for image-based machine learning, the following preprocessing steps were applied:

- **Normalization**: All pixel values were scaled from the range [0, 255] to [0, 1] by dividing by 255.0. This ensures that the input features are on a consistent scale. Also, this strategy improves model numerical stability for MLPs and CNNs.
- **Flattening**: For decision tree models and MLPs, images were reshaped from (28, 28, 3) to (2352,) vectors. The numerical value of 2352 vectors are getting from the calculation of 28*28*3. This pre-processing stage enables these models to process the data in tabular format, although it discards the original positional structure.
- **Maintaining shape for CNNs**: CNN models received the original (28, 28, 3) input without flattening, as they rely on spatial relationships between pixels to learn features.

We did not apply data augmentation techniques such as rotation or flipping, as the dataset is already large and well-structured enough. However, implementation of augmentation could be beneficial in future work to improve generalization.

Each preprocessing choice was made based on lecture recommendations and best practices in computer vision. Different preprocessing steps were applied based on the type of model. Each model with each choice is specifically designed to match the model's input requirements and strengths.

3. Methods

3.1 Algorithm Overview

3.1.1 Algorithm of Choice – Decision Tree

A decision tree is a hierarchical model used to solve problems by mapping out decisions and their possible outcomes. (GeeksforGeeks, n.d.) Decision trees have a wide range of practical applications. In healthcare, they assist with diagnoses and treatment planning. In finance, they are used for credit scoring and evaluating risk. They are also applied in marketing strategies, fraud detection and recommendation systems. (Keylabs, n.d.)

A decision tree begins with a **root node**, which raises an initial question based on the dataset's features. The tree branches out by asking a series of **yes/no questions**. Each branch split the data into smaller subsets based on specific attributes. By following the branches, the data is repeatedly divided until no longer useful splits can be made. This process produces **leaf nodes**, which provide the final output. The final output is either a classification (e.g., "normal" or "abnormal") or a prediction (e.g., estimated value). (GeeksforGeeks, n.d.)

We selected the decision tree as the classical algorithm due to its simplicity and alignment with early COMP5318 tutorial. During the choosing phase of our model, we first planned to choose from the Decision Tree classifier or the k-NN model. After evaluating both options, we proceeded with the Decision Tree as it successfully ran and demonstrated competitive performance during

cross-validation. In addition, k-NN model has more disadvantages than using the decision tree while applying to the dataset in this task. For instance, k-NN is a lazy and simple algorithm, which will be relatively slower computation time than using decision tree classifier. (GeeksforGeeks, n.d.) Another specific example is that the decision tree classifier is more suitable for the dataset that we are applying.

3.1.2 Multilayer Perception (MLP)

An MLP is a feedforward artificial neural network composed of multiple layers of fully connected neurons, each using a nonlinear activation function such as ReLU or tanh. These models are called "feedforward" because the data flows in one direction. It first starts at input x, and flows through the function computations. Lastly, the models end with an output of y. (Goodfellow et al., 2016). Feedforward neural networks are foundational to machine learning and widely used in real-world problems. They are structured by chaining functions together, forming a **directed acyclic graph**. Each function represents a **layer** in the network. The number of layers determines the **depth** of the model, which gives the term **"deep learning".** (Goodfellow et al., 2016).

The **output layer** is trained to match target labels from data, while the behavior of intermediate layers (called **hidden layers**) is learned indirectly. These layers help approximate the true function $f*$. Each of the hidden units can be thought of as a simplified "neuron" that computes an activation. Simple visualized figure is shown in figure 5. Overall, these networks are best seen as **function approximators** designed for generalization, not biological modeling. (Goodfellow et al., 2016).

3.1.3 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a specialized type of artificial neural network designed to automatically extract features from data arranged in a grid-like structure. This makes CNNs especially effective for visual tasks by recognizing spatial patterns. As a result, they are widely used in computer vision applications.

CNN architectures typically include layers such as the input layer, convolutional layers, pooling layers, and fully connected layers, each contributing to feature extraction and classification. (Figure 6 (GeeksforGeeks, n.d.))

In CNN, an image can be viewed as a 3D structure with width, height, and depth, which correspond to color channels like red, green, and blue. CNNs use small filters. Those small filters or kernels scan across local regions of the image to extract features. This process is the "**convolution"** part. CNN transforms the image into new feature maps with reduced dimensions but increased depth. CNNs focus on local patterns, resulting in fewer parameters and better generalization for visual tasks, which is different from standard neural networks that operate on the entire input at once. (GeeksforGeeks, n.d.)

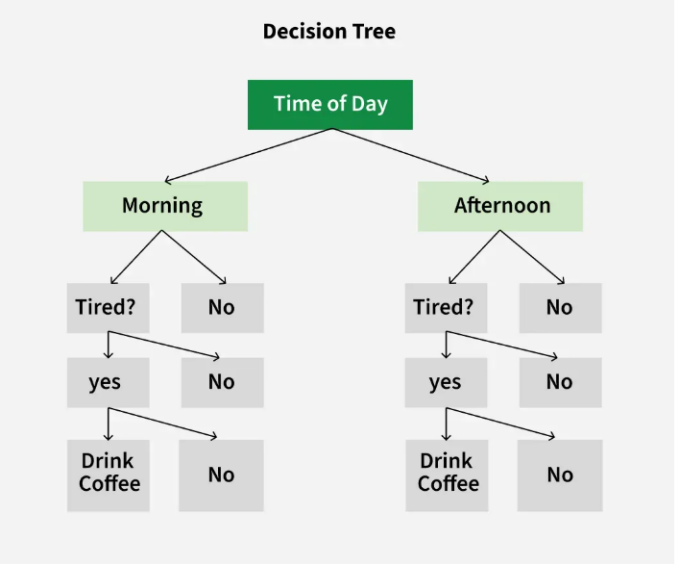


Figure 4: Visualisation of Decision Tree model (GeeksforGeeks, n.d.)

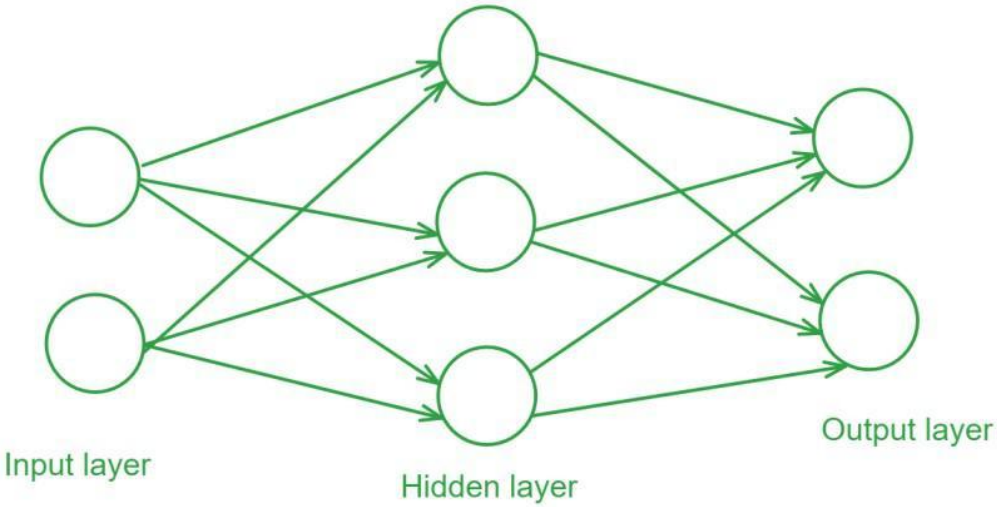


Figure5: Visualisation of Feedforward Neural Networks (GeeksforGeeks, n.d.)

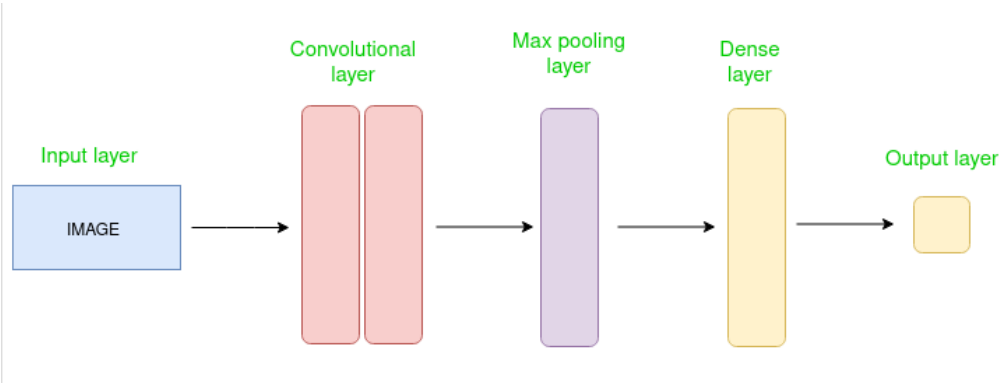


Figure 6: Visualisation of Convolutional Neural Networks (GeeksforGeeks, n.d.)

3.2 Strengths and Weaknesses

Below is the description of theoretical and practical advantages and disadvantages of each model in the context of this assignment's dataset.

3.2.1 Algorithm of Choice – Decision Tree

a. Advantages

- **Easy to Understand:** Decision trees are available for this PATHMNIST dataset because they allow clear reasoning for predictions. This interpretability is beneficial when reviewing predictions (GeeksforGeeks, n.d.).
- **Suitable for Tabular Features:** Once the image data is flattened, decision trees can effectively classify based on thresholds in these tabularized features. It is useful when using traditional ML on image datasets like PathMNIST.
- **No Need for Normalization**: Since the pixel values are within a known range (0–255 should normalized to 0–1), decision trees handle these directly without preprocessing complications. This saves time and increases efficiency.
- **Can Model Non-linear Relationships:** In medical images, class boundaries are rarely linearly separable due to complex tissue patterns. Decision trees can capture these nonlinear boundaries by recursive splitting (GeeksforGeeks, n.d.).

b. Disadvantages

- **Overfitting (Especially in small features):** PathMNIST contains rich color and texture difference. A deep decision tree might memorize outliers or subtle pixel-level variations in the training set. As a result, it might cause the poor generalization on unseen images (AIML, n.d.).
- **Not Stable with Small Changes:** Small variations in training images can cause completely different tree structures. This is problematic in medical datasets, which images are different in how the slides are coloured (AIML, n.d.).
- **Biased To Dominant Classes:** If some classes are underrepresented, the decision tree may favor majority classes unless they are explicitly balanced. Therefore, this disadvantage of decision tree risks misclassification process (AIML, n.d.).
- **Scalability:** Although PathMNIST is somehow lightweight, applying the same approach to larger datasets would greatly increase the number of features, making the model possibly harder to manage. In other words, decision trees do not scale efficiently with high-dimensional data.

3.2.2 Multilayer Perception (MLP)

a. Advantages

- **Simpler to Interpret than CNN**: MLPs are relatively easier to design, and train compared to CNN. MLPs offer an understandable alternative, especially when visual transparency is valued (Tahir et al., 2020).
- **Generally Good Performance Different Tasks:** MLPs have been proven to perform generally well in other tasks including classification and regression. In the case of PathMNIST, MLP can still achieve acceptable performance when the image data is flattened into vector form (Tahir et al., 2020).
- **Competitive to RBF Networks:** In some cases, MLPs have been proved to perform better than Radial Basis Function networks in classification accuracy. This offers more efficient and scalable architecture (Tahir et al., 2020).

b. Disadvantages

- **Reduction of Spatial Structure:** The main limitation of using MLPs for image classification is in the flattening process. PathMNIST images are of shape (28, 28, 3), but MLPs require these to be reshaped into 1D vectors (28*28*3 = 2532). Furthermore, this transformation removes the spatial layout, such as edge continuity or textures. Those layout factors are important in differentiating subtle tissue patterns in medical images (Goodfellow et al., 2016).
- **High Dimensionality Leads to More Parameters:** When treating each pixel independently, the model ends up with many weights. This increases the risk of overfitting, particularly when training data is small (Goodfellow et al., 2016).
- **The Need for Large Amount of Dataset:** MLPs need large numbers of labeled image examples to perform well. Although PathMNIST is considered well-balanced, this could become a limitation in other medical imaging contexts where data labeling are possibly limited (Goodfellow et al., 2016).

3.2.3 Convolutional Neural Network

a. Advantages

- **Focus one small region for Image Features**
  In the PathMNIST dataset, key features such as cellular clusters and tissue boundaries are often localised in small parts of the images. CNN focuses on small regions of an image at a time rather than processing the entire image all at a time. This strategy of CNN allows the model to detect important local features in the images and prevent overfitting. (Goodfellow et al., 2016).
- **Parameter Sharing**
  Features like texture or color intensity can appear anywhere in images. CNNs reuse the same filters across the entire image. This strategy increases efficiency and effectiveness at detecting these patterns' position.

- **Hierarchical Feature Learning**
  CNNs combine low-level visual patterns into high-level structures. This strategy enables them to classify between the 9 categories or classes in the PathMNIST dataset.

- **Efficient Learning and Scalability**
  CNNs require fewer parameters than MLPs due to their localised **connectivity and weight sharing**. (Goodfellow et al., 2016). Connectivity refers to the level of the neurons and input data are linked. In CNNs, each filter (neuron) is connected only to small regions of the input rather than the entire image. Another term is weight sharing. In CNNs, it reuses the same weights as it slides within the image. These two factors lead to faster training and less risk of overfitting, since CNNs need fewer numbers of parameters compared to MLPs. Moreover, connectivity and weight sharing allow CNN model detecting the same feature in different locations. (Goodfellow et al., 2016).

b. Disadvantages

- **High Computational Cost**: CNNs require more computational resources compared to traditional models like decision trees or MLPs. For the PathMNIST dataset, training CNNs with multiple convolutional layers and tuning hyperparameters (like filter size, kernel count, learning rate) is time-consuming. Moreover, it demands more GPU support for efficient training process (Goodfellow et al., 2016).

- **Overfitting on Small Datasets**: Although CNNs are powerful, they can overfit if the dataset is small or lacks diversity. Although PathMNIST dataset has over 80,000 samples, images often have only small differences between classes, which can cause the model to overfit. Regularization techniques like data augmentation are helpful to prevent overfitting problems.

- **High Complexity Leads to Reduction of Interpretability**: Unlike decision trees, CNNs operate as black-box models. For medical tasks in PathMNIST, lack of transparency can make it difficult for clinicians to understand why a certain prediction was made. Therefore, it is causing a critical barrier in real-world healthcare deployment (GeeksforGeeks, n.d.).

- **Requires Good Architecture Design**: CNN performance heavily depends on architectural choices like the number of layers, filter size, pooling strategy, and stride. As a result, poorly chosen settings can easily lead to poor performance. (LeCun et al., 2015).

3.3 Architectures and Hyperparameters

3.3.1 Algorithm of Choice from Weeks 1–6 – Decision Tree

a. Architecture

The decision tree model that repeatedly splits data based on limit sets for specific features. The model then maximizes information gain or reduces impurity.

It produces a tree structure consisting of internal decision nodes and leaf nodes which are the predictions.

b. Hyperparameters Tuned

max_depth: The maximum depth of the tree. A deeper tree means more complexity but more risks of overfitting. We tested depths of 5, 10, 15, and 20 to find a balance between underfitting and overfitting.

criterion: The function to measure the quality of a split. This includes:

'gini' for the Gini impurity.

'entropy' for information gain.

c. Search Method

GridSearchCV was used with 5-fold stratified cross-validation.

A grid of hyperparameters was defined for max_depth and criterion. The model was trained and validated across all combinations.

After training through all the combinations, the highest mean cross-validation score was selected.

d. Justification

The hyperparameter max_depth in our decision tree model was tuned over the values [5, 10, 15, 20] to explore how tree complexity affects performance. Trees with depth 5 help prevent overfitting but may underfit complex datasets. On the other hand, deeper trees (depths 15 and 20) detect more feature interactions. This is particularly relevant for high-dimensional image data like PathMNIST. This tuning helps balance between bias and variance (Rokach & Maimon, 2014).

Decision trees have relatively few hyperparameters. This allows GridSearchCV to evaluate all useful combinations without tons of computation. Since the number of possible combinations is limited, grid search provides an effective method for identifying best or optimal parameters after training (Pedregosa et al., 2011). After doing research, our team considered GridSearchCV as an efficient search method throughout this model.

Including criterion helps evaluate the impact of different impurity metrics on classification performance. Gini and entropy are two commonly used metrics to measure the quality of a split in a decision tree. Gini impurity computationally faster and often performs well in practice, so it is

commonly used now for decision tree models. Entropy is based on information gain. Entropy can lead to more balanced trees by considering the distribution of classes. Including criterion especially benefits for imbalanced or multi-class datasets like PathMNIST dataset (Rokach & Maimon, 2014; Quinlan, 1986).

3.3.2 Multilayer Perception (MLP)

a. Architecture

In our implementation, the model starts with an input layer that flattens the 28×28×3 image into a 2352-dimensional vector. The input layer is then followed by two hidden dense layers and one output layer for 9-class classification (The dataset includes 9 classes, 0-8 labels). ReLU activation functions were used for non-linearity in the hidden layers, and SoftMax activation was used in the output layer to produce class probabilities. The model was compiled using the **Adam optimiser**, which is widely adopted for its adaptive learning rate and computational efficiency. In addition, Adam optimiser is generally performing better than traditional SGD optimiser (Ahmad, 2021).

b. Hyperparameters Tuned

We tuned the hyperparameters using keras_tuner.RandomSearch:

- units_1 and units_2: Number of neurons in the two hidden layers. We tested two values, **100** and **200**.
- Activation Functions: Activation functions tested include ReLU, sigmoid, and tanh. These control how the network introduces non-linearity and affects overall performance significantly.
- Learning Rate: Learning rates of 0.1, 0.01, and 0.001 were explored using the SGD optimizer to examine efficiency and stability.

c. Search Method

We employed Keras Tuner's RandomSearch method with 24 trials and 15 training epochs per trial. This approach balances search coverage and runtime. The best performing hyperparameters will be used to train a final MLP model for 128 epochs on the full training set.

d. Justification

The choice of neuron counts in each layer reflects a trade-off between model expressiveness and overfitting. Larger layers can model more complex functions but risk overfitting on small datasets like PathMNIST.

The inclusion of multiple activation functions allows us to evaluate which function best suits the distribution of the PathMNIST features. ReLU is often preferred for image data. ReLU activation function has been proved to have computational simplicity and effectiveness to low-density gradients (Goodfellow et al., 2016).

Testing multiple learning rates is extremely significant to maintain stable and efficient training. In specific, when using SGD. SGD is more sensitive to step size than adaptive optimizers.

(Goodfellow, Bengio, & Courville, 2016).It is important to seek out the best combination of learning rate and activation function with SGD to reduce the loss function result.

We used the Adam optimiser for training the MLP model. Adam optimiser adapts learning rates per parameter. Adam optimiser also provides faster and more stable convergence in flattened PathMNIST images, which are considered as high-dimensional data. Compared to standard Keras SGD, Adam is better suited for datasets with complex feature distributions (Kingma & Ba, 2015).

The number of neurons in the two hidden layers was set to either 100 or 200. These values were chosen to improve learning capacity and prevent overfitting. PathMNIST is considered as a moderately sized dataset with small input images (28×28×3). Large layers could lead to overfitting and unnecessary computational cost(Goodfellow, Bengio, & Courville, 2016). A smaller setting like 100 encourages generalization, while 200 adds more representational power without making the model overly deep or slow to converge. (Goodfellow, Bengio, & Courville, 2016).

3.3.3 Convolutional Neural Network

a. Architecture

CNN implemented for this assignment was designed to take advantage of the spatial patterns in the images. The model begins with an input layer that maintains the original image dimensions (28×28×3). Two convolutional layers are used to extract hierarchical features. Each layer is followed by a max pooling operation to reduce spatial dimensions and control overfitting. The number of filters, kernel sizes, and activation functions in both convolutional layers were treated as hyperparameters. The first layer uses tunable filters (32, 64, or 128) and kernel sizes (3 or 5). The first layer uses tunable filters (64, 128, or 256) and kernel sizes (3 or 5).

A dropout layer is added after the convolutional blocks to further reduce the risk of overfitting by randomly deactivating a proportion of neurons during training. The model concludes with a fully connected softmax output layer. This output layer maps the learned information to nine output classes corresponding to the dataset labels.

Hyperparameter tuning was performed using Keras Tuner with a random search strategy. The search space included variations in the number of filters, kernel sizes, activation functions (ReLU and tanh), dropout rate (0.3 to 0.5), and learning rate. The Adam optimizer was selected. Validation accuracy was used as the objective for tuning

b. Hyperparameters Tuned

We tuned the following hyperparameters using keras_tuner.RandomSearch:

filters_1, filters_2: Number of filters for the first and second Conv2D layers.

kernel_size_1, kernel_size_2: Kernel size choices (3 or 5).

activation_1, activation_2: Activation functions used in each Conv2D layer (ReLU or tanh).

dropout_rate: Dropout rate tested between 0.3 and 0.5.

learning_rate: Learning rate for the Adam optimizer (0.1, 0.01, 0.001).

c. Search Method

We used Keras Tuner's RandomSearch method with 36 trials, 15 epochs per trial, and validation accuracy as the objective. After identifying the best-performing hyperparameters, we retrained the CNN for 128 epochs using those optimal and tuned hyperparameters.

d. Justification

Filters: The number of filters in each convolutional layer was selected from 32, 64, 128 (first layer) and 64, 128, 256 (second layer). These values reduce the risk of overfitting situations. Smaller filter counts may not detect enough detail from images, while overlarge filter counts can introduce noise and overfitting. (Kermany et al., 2018).

Kernel Sizes (3x3 and 5x5): Using kernel sizes of 3 and 5 allows the model to capture local patterns of different spatial scales. A 3x3 kernel focuses on finer details, while 5x5 size allow the model to capture broader features. This strategy is important in medical image classification which contains lots of small-scale details and larger structural patterns (Goodfellow, Bengio, & Courville, 2016).

Activation Functions: Both ReLU and Tanh were tested. ReLU is widely used in image-related deep learning. ReLU avoids problems which the gradients become too small to update the weights. Tanh is tested to evaluate potential benefits in stabilizing feature extraction. Therefore, it is necessary to compare these two activation functions' performances in the model, and pick the better one (Arel et al., 2010; Goodfellow et al., 2016).

Dropout Rate: Dropout rates of 0.3 to 0.5 were tested to prevent overfitting. In CNNs, dropout is particularly effective in fully connected layers which memories features while training(Srivastava et al., 2014).

Learning Rate (0.1, 0.01, 0.001): These values were chosen to test different learning speeds. If the learning rate is too high, the model might skip over the best solution. If it's too low, training can become very slow(Goodfellow et al., 2016).

3.3.4 Use of Evaluation Metrics Across All Models in This Assignment

We evaluated all models using confusion matrices and classification reports that include precision, recall, F1-score, and support. These metrics were chosen for the following reasons:

Class-level Analysis:

The dataset is a multi-class classification task with potential class imbalance. Only using accuracy as the evaluation matrix may cause poor performance in minority classes. The confusion matrix reveals which classes are frequently misclassified, helping us understand how model can be improved further.

Precision & Recall:

Precision helps evaluate how reliable a model's positive predictions are for each class. It is important to avoid false positives within classification tasks (Saito & Rehmsmeier, 2015).

Recall indicates the model's ability to detect all actual instances of each class. It is also an important point to use Recall matrix to avoid false positives (Saito & Rehmsmeier, 2015).

F1-Score:

The F1-score combines both precision and recall into a single measure. It is useful when both false positives and false negatives are important to avoid. It is also an indication of the balance between correctly identifying and misclassifying each class (Chinchor, 1992).

Macro and Weighted Averages:

No matter how many samples there are, Macro averaging gives the same importance to every class. Weighted averaging gives more importance to classes that appear more often, which makes it more accurate for this dataset which includes class imbalance (Pedregosa et al., 2011).

By applying the same metrics consistently across all three models, we ensure fair and comparable performance evaluation.

4. Results and Discussion

4.1 Hyperparameter Tuning Results

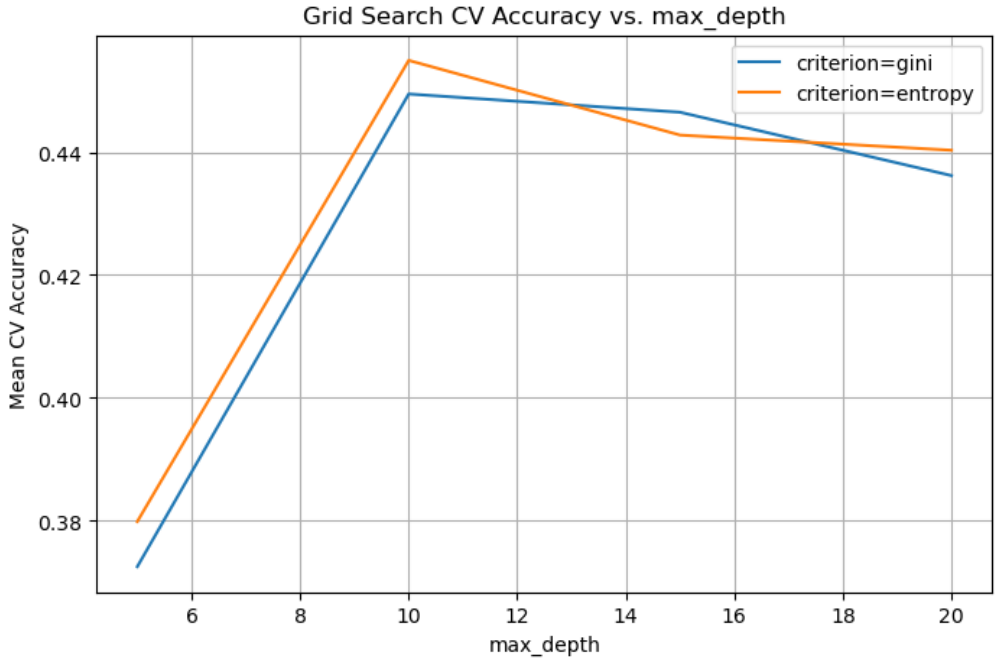
To achieve the best performance for each model, we conducted hyperparameter tuning for each model separately.

Decision Tree:

From the graph, we can see that regardless of whether using gini or entropy, when max_depth = 10, the effect is the best. Then, the performance of entropy is slightly better than that of gini. At max_depth = 10, the Best Validation Accuracy reaches 0.4549. However, after max_depth = 10, both criteria have declined. We suspect this is due to overfitting.

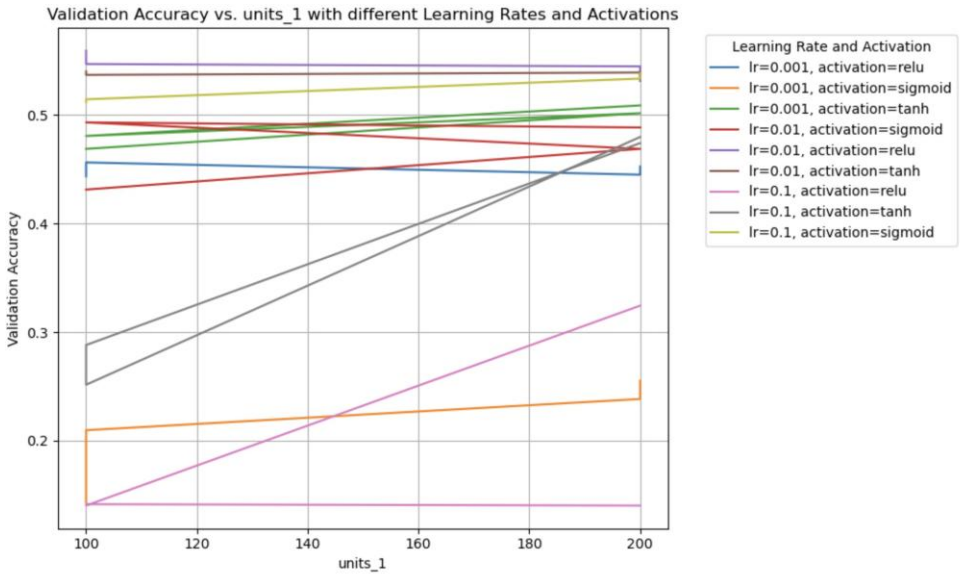
In terms of running time, the running time of Decision Tree is not very short. We suspect this is because We used grid_search. It will perform cross-validation, with 4 options for max_depth and 2 options for criterion. There are a total of 8 models × 5-fold cross-validation = 40 fits. The running time for each combination is approximately 4 seconds. Compared to the subsequent MLP and CNN models, Decision Tree still has an advantage in speed, but its Best Validation Accuracy is not very high. It reached the best at max_depth = 10, but it was only 0.4549.

	param_max_depth	param_criterion	mean_test_score
0	5	gini	0.372531
1	10	gini	0.449469
2	15	gini	0.446500
3	20	gini	0.436188
4	5	entropy	0.379875
5	10	entropy	0.454937
6	15	entropy	0.442781
7	20	entropy	0.440312



MLP:

From the graph, we can tell that the running results of MLP are generally low, but they are still higher compared to Decision Tree. When learning rate = 0.01 and activation = Relu, the Validation Accuracy is the highest, approximately 0.5591. At the beginning of the run, we noticed that the running time of MLP was generally shorter than that of Decision Tree. We think this is illogical. In my subsequent research, we found that max_trials saved all the data during the first run and did not perform repeated runs afterwards. Therefore, this code runs particularly fast in subsequent runs, but during the first run, the running time of each combination was approximately 20 seconds, which is indeed longer than that of Decision Tree, the total running time shown in the output is also very long. Due to the design of the table, we were unable to include units_2, but by combining the two tables, we could find the corresponding units_2 for the parameter combination. From the graph, it can be seen that when lr = 0.01, activation = relu, units_1 = 100, and units_2 = 200, the parameter combination has the best effect.

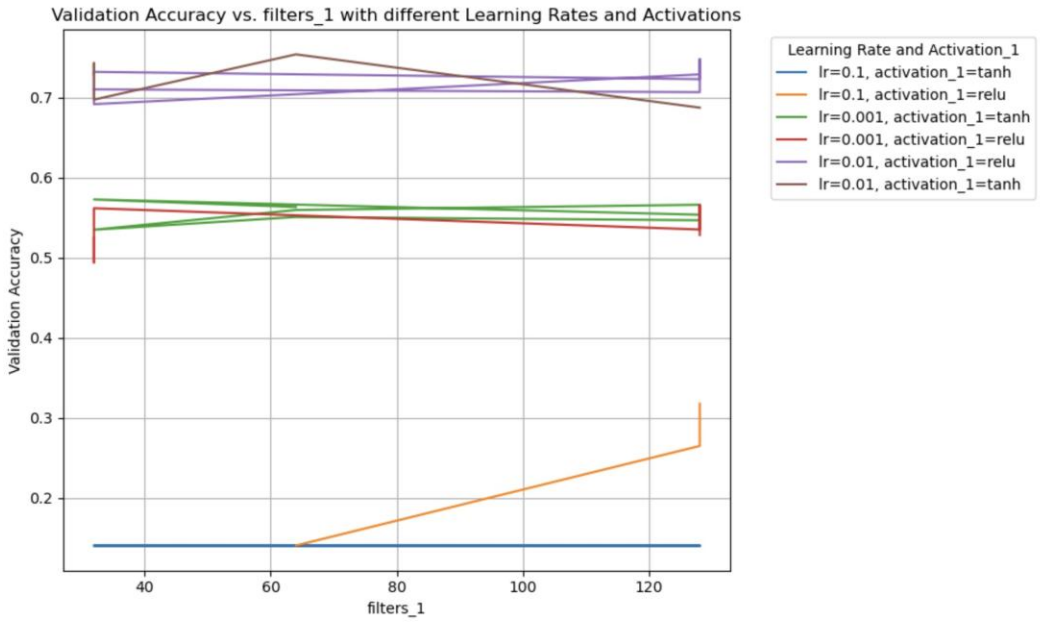



```
Trial 36 Complete [00h 00m 10s]
val_accuracy: 0.14031249284744263

Best val_accuracy So Far: 0.5590624809265137
Total elapsed time: 17h 59m 36s
```

	units_1	units_2	activation	learning_rate	val_accuracy	accuracy	\
0	200	100	relu	0.001	0.452500	0.559063	
1	100	200	sigmoid	0.010	0.431250	0.559063	
2	200	100	relu	0.100	0.324375	0.559063	
3	200	100	sigmoid	0.001	0.255313	0.559063	
4	200	200	relu	0.001	0.445000	0.559063	
5	100	100	relu	0.100	0.140312	0.559063	
6	100	200	relu	0.010	0.559062	0.559063	
7	100	100	tanh	0.001	0.468750	0.559063	
8	200	100	tanh	0.001	0.501562	0.559063	
9	100	200	relu	0.100	0.141563	0.559063	
10	100	100	tanh	0.010	0.540000	0.559063	
11	200	200	sigmoid	0.010	0.468750	0.559063	
12	100	200	tanh	0.010	0.536875	0.559063	
13	100	100	relu	0.010	0.546875	0.559063	
14	200	100	tanh	0.100	0.474063	0.559063	
15	100	200	sigmoid	0.100	0.511875	0.559063	
16	100	200	tanh	0.100	0.288125	0.559063	
17	200	100	tanh	0.010	0.539062	0.559063	
18	100	100	sigmoid	0.010	0.493125	0.559063	
19	200	200	tanh	0.010	0.536250	0.559063	
20	100	100	sigmoid	0.100	0.514375	0.559063	
21	200	200	relu	0.010	0.544688	0.559063	
22	200	100	sigmoid	0.100	0.533437	0.559063	
23	100	100	relu	0.001	0.456250	0.559063	
24	100	200	relu	0.001	0.443437	0.559063	
25	100	200	tanh	0.001	0.480625	0.559063	
26	200	200	sigmoid	0.001	0.238438	0.559063	
27	200	100	sigmoid	0.010	0.488438	0.559063	
28	100	200	sigmoid	0.001	0.209688	0.559063	
29	200	200	sigmoid	0.100	0.538437	0.559063	
30	100	100	sigmoid	0.001	0.143438	0.559063	
31	200	100	relu	0.010	0.531250	0.559063	
32	100	100	tanh	0.100	0.251563	0.559063	
33	200	200	tanh	0.001	0.508750	0.559063	
34	200	200	tanh	0.100	0.479688	0.559063	
35	200	200	relu	0.100	0.140312	0.559063	

CNN：

From the graph, we can see that the running results of CNN have significantly improved compared to MLP and Decision Tree. The Validation Accuracy is generally above 0.5, with the highest reaching 0.7541. However, its running time is much longer. It takes nearly 4 minutes for a single combination to run, and the total running time is also very long. We selected max_trials as 36, which is different from max_trials in MLP. In MLP, there are a total of 36 combinations after arranging different hyperparameters. Therefore, we trained all of them. But for CNN, after arranging the max_trials, there are 3888 combinations. We couldn't train all of them, so I only listed 36 combinations. However, the results are very good. The training Validation Accuracy is generally high, and some slightly lower ones might be overfitting.




```
Trial 36 Complete [00h 03m 48s]
val_accuracy: 0.5284374952316284

Best val_accuracy So Far: 0.7540624737739563
Total elapsed time: 18h 36m 46s
```

	filters_1	kernel_size_1	activation_1	filters_2	kernel_size_2	\		activation_2	dropout_rate	learning_rate	val_accuracy	accuracy
0	128	5	tanh	256	3		0	tanh	0.3	0.100	0.140312	0.754062
1	64	3	tanh	128	5		1	tanh	0.3	0.001	0.563438	0.754062
2	32	3	tanh	256	3		2	relu	0.3	0.100	0.140625	0.754062
3	32	5	tanh	256	3		3	relu	0.4	0.001	0.572812	0.754062
4	32	5	relu	64	5		4	tanh	0.4	0.001	0.524688	0.754062
5	32	5	relu	64	3		5	relu	0.3	0.010	0.710312	0.754062
6	128	5	tanh	128	5		6	tanh	0.3	0.001	0.553750	0.754062
7	128	5	tanh	64	5		7	relu	0.3	0.001	0.566250	0.754062
8	64	3	tanh	128	3		8	relu	0.4	0.001	0.559687	0.754062
9	128	5	tanh	64	5		9	tanh	0.3	0.100	0.140312	0.754062
10	128	3	relu	256	5		10	relu	0.4	0.100	0.317813	0.754062
11	32	5	tanh	128	3		11	tanh	0.5	0.001	0.535000	0.754062
12	128	3	tanh	128	5		12	tanh	0.3	0.010	0.687500	0.754062
13	128	3	relu	64	3		13	tanh	0.5	0.010	0.706875	0.754062
14	64	3	tanh	256	5		14	relu	0.5	0.010	0.754062	0.754062
15	32	3	tanh	256	5		15	tanh	0.3	0.010	0.697500	0.754062
16	128	3	relu	256	5		16	tanh	0.4	0.100	0.265000	0.754062
17	128	3	relu	256	5		17	relu	0.5	0.010	0.748125	0.754062
18	128	5	relu	128	5		18	relu	0.5	0.010	0.723125	0.754062
19	32	3	tanh	256	5		19	relu	0.3	0.010	0.735937	0.754062
20	64	3	tanh	128	5		20	tanh	0.4	0.001	0.550937	0.754062
21	32	5	tanh	128	5		21	tanh	0.5	0.100	0.140312	0.754062
22	32	5	tanh	64	5		22	relu	0.5	0.010	0.743125	0.754062
23	32	3	relu	128	3		23	tanh	0.5	0.001	0.494063	0.754062
24	32	3	tanh	64	5		24	relu	0.5	0.010	0.729062	0.754062
25	32	3	relu	256	5		25	relu	0.3	0.001	0.561875	0.754062
26	32	5	relu	128	5		26	relu	0.4	0.010	0.732188	0.754062
27	32	5	relu	256	5		27	tanh	0.3	0.010	0.691875	0.754062
28	128	5	relu	128	5		28	relu	0.4	0.010	0.729062	0.754062
29	128	5	tanh	64	5		29	relu	0.4	0.001	0.546875	0.754062
30	128	5	relu	256	3		30	tanh	0.3	0.001	0.535312	0.754062
31	128	3	relu	64	3		31	relu	0.3	0.010	0.735625	0.754062
32	128	5	relu	64	3		32	relu	0.4	0.001	0.565625	0.754062
33	64	5	relu	64	5		33	relu	0.3	0.100	0.140625	0.754062
34	128	3	relu	256	5		34	relu	0.4	0.001	0.544688	0.754062
35	128	3	relu	64	5		35	relu	0.5	0.001	0.528437	0.754062

4.2 Final Model Performance

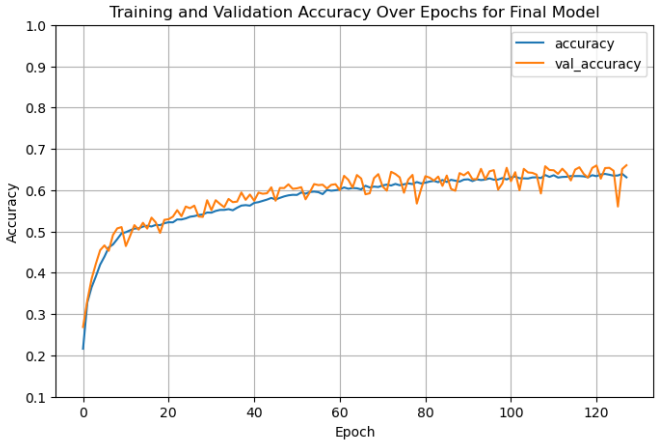
Decision Tree:

The accuracy of the best model from each algorithm in the Decision Tree is 0.46. Among them, the Precision of category 1 is 0.88, the Recall is 0.84, and the F1 score is 0.86, indicating that this model performs very well in class recognition. However, overall, the Accuracy, Precision, Recall, and F1 Score of Decision Tree are all around 0.46, and the training time is about 20 seconds. This shows the distinct characteristics of Decision Tree, namely its speed and relatively low accuracy.

```
   max_depth criterion  accuracy  precision    recall  f1_score  \
0         10   entropy  0.462625   0.459779  0.462625  0.455008

                                  confusion_matrix  training_time
0  [[698, 34, 7, 1, 84, 19, 11, 9, 10], [46, 720,...      20.112729

Confusion Matrix:
[[698  34   7   1  84  19  11   9  10]
 [ 46 720   4  30  24   4   2  14  14]
 [ 12   7 221  55  49 168  29 154 181]
 [  8  12  14 477  84   9  43  40 227]
 [ 62  23  17  93 294  38  29  93  88]
 [ 25   8 171  25  73 396  21 240 114]
 [ 25   5  26 127 138  31  59  78 193]
 [  3   3  90  45  99 132  18 303 120]
 [ 16  10  61 243  77  78  60  96 533]]

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.80      0.79       873
           1       0.88      0.84      0.86       858
           2       0.36      0.25      0.30       876
           3       0.44      0.52      0.47       914
           4       0.32      0.40      0.35       737
           5       0.45      0.37      0.41      1073
           6       0.22      0.09      0.12       682
           7       0.30      0.37      0.33       813
           8       0.36      0.45      0.40      1174

    accuracy                           0.46      8000
   macro avg       0.46      0.45      0.45      8000
weighted avg       0.46      0.46      0.46      8000
```

MLP:

The accuracy of the best model from each algorithm in the MLP is 0.6603. From the learning curve graph, it is clearly visible that as the value of epoch gradually increases, both the accuracy and val_accuracy of the image increase, but gradually level off, and the fluctuation of val_accuracy is significant. The Accuracy, Precision, Recall, and F1 Score of MLP are all around 0.63 and the training time is around 96 seconds. This shows that MLP is slower than Decision Tree and has a relatively higher accuracy rate.




```
Best Epoch: 128
Best Validation Accuracy: 0.6603
```

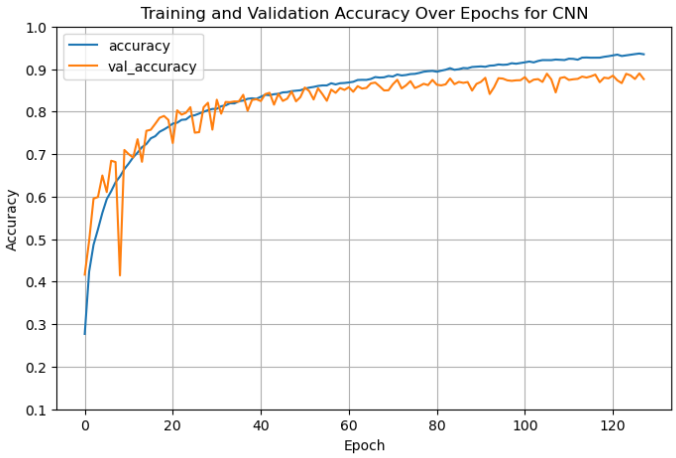
```
Confusion Matrix:
[[824  15   0  12   4   3  15   0   0]
 [ 56 721   2  20  36   6   2   6   9]
 [  7   9 284   6  11 265  43 125 126]
 [  2   1   0 772   7   2  42   0  88]
 [ 68  24   5  39 406  15  78  30  72]
 [ 32   3  89   0  10 721  31 156  31]
 [ 13   0   1  66  71  13 212  11 295]
 [  0   1 116   1  25 214  16 358  82]
 [  3   4  59  44  12  31 134  81 806]]
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.94      0.88       873
           1       0.93      0.84      0.88       858
           2       0.51      0.32      0.40       876
           3       0.80      0.84      0.82       914
           4       0.70      0.55      0.62       737
           5       0.57      0.67      0.62      1073
           6       0.37      0.31      0.34       682
           7       0.47      0.44      0.45       813
           8       0.53      0.69      0.60      1174

    accuracy                           0.64      8000
   macro avg       0.63      0.62      0.62      8000
weighted avg       0.63      0.64      0.63      8000
```

```
Model Evaluation Results:
   units1  units2  accuracy  precision  recall  f1_score  \
0      64      32     0.638   0.634441   0.638  0.630143

                                  confusion_matrix  training_time
0  [[824, 15, 0, 12, 4, 3, 15, 0, 0], [56, 721, 2...      95.735808
```

CNN:

The accuracy of the best model from each algorithm in the CNN is 0.8897. As can be seen from the learning curve graph, as the value of epoch gradually increases, both the accuracy and val_accuracy of the images increase, but they also gradually level off in the same way as MLP. Moreover, the fluctuation of val_accuracy is significant, and it is significantly lower than accuracy when epoch > 40. The Accuracy, Precision, Recall, and F1 Score of MLP are all around 0.88, and the training time is around 2264 seconds. It can reflect the obvious characteristics of CNN, but the speed is very slow, yet the accuracy is extremely high.




```
Best Epoch: 127
Best Validation Accuracy: 0.8897
```

```
Confusion Matrix:
[[365   1   0   0   8   1   1   2   0]
 [  3 351   0   0   6   0   0   0   0]
 [  0   0 267   0   4  11   2  27  17]
 [  0   0   3 339   0   0   6   0   7]
 [  3   6   2   0 286   5  10   6   2]
 [  5   0  17   1   5 343   1  58   4]
 [  1   0   5   2  12   0 211   2  22]
 [  0   0  35   0   8  24   1 249   4]
 [  1   0  16   3   9   0  13  14 393]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       378
           1       0.98      0.97      0.98       360
           2       0.77      0.81      0.79       328
           3       0.98      0.95      0.97       355
           4       0.85      0.89      0.87       320
           5       0.89      0.79      0.84       434
           6       0.86      0.83      0.84       255
           7       0.70      0.78      0.73       321
           8       0.88      0.88      0.88       449

    accuracy                           0.88      3200
   macro avg       0.87      0.87      0.87      3200
weighted avg       0.88      0.88      0.88      3200


Model Evaluation Results:
   units1  units2  accuracy  precision   recall  f1_score  \
0      64      32   0.87625   0.879668  0.87625  0.877147

                                  confusion_matrix  training_time
0  [[365, 1, 0, 0, 8, 1, 1, 2, 0], [3, 351, 0, 0,...    2263.855921
```

4.3 Analysis and Discussion

For these three types of models, we can clearly perceive the differences among them.

Decision Tree:

Its advantages lie in "**No Need for Normalization**" and "**Easy to Understand**", which can be clearly felt during the training process. It can fit non-linear decision boundaries and is suitable for structured, simple image features. However, its disadvantages are also obvious. It is clearly felt in the subsequent training that "**Overfitting**" occurs. When the depth increases, its accuracy significantly decreases. Its accuracy initially increases, reaches the highest at a depth of 10, and then gradually decreases. From the previous confusion matrix, it can be seen that the classification effect of categories 0 and 1 is very good, with a recall rate of 0.80. However, the performance of categories 2, 5, 6, 7, and 8 is poor, with a very low recall rate and a high confusion rate. There are a large number of misclassifications. The model's handling of these data is not particularly good.

MLP:

The advantage of MLP lies in being **simpler to interpret than CNN** and **generally good performing different tasks**. During training, it indeed met the basic classification requirements, with an accuracy rate remaining around 0.638. Overall, the performance was good. However, its drawbacks are also obvious. It **lacks a convolutional structure**, making it difficult to extract local image features, which can **easily lead to overfitting**, especially when the training data is small. From its learning curve, we can see that its accuracy gradually levels off and does not show overfitting, but there is a risk of overfitting. It reached the best validation performance at Epoch = 128, and after that, the performance may decline, indicating that the training is slow and accompanied by the risk of overfitting. If the training iterations are increased further, overfitting may occur. Its confusion matrix shows that class 0, class 1, and class 3 have good performance, while class 2 performs poorly. The recall is only 0.32. For class 5, 7, and 8, the classification errors are large, indicating that MLP cannot effectively distinguish the details between these classes. Class 6 and class 8 also have similar problems.

CNN:

The advantage of CNN lies in **Hierarchical Feature Learning**, focusing on a small region for Image Features to prevent overfitting, as well as **Efficient Learning and Scalability**. From the training results, it is clearly seen that its accuracy is very high and its learning ability is very strong. In the parts of filters=hp.Choice("filters_1", [32, 64, 128]) and kernel_size = hp.Choice("kernel_size_1", [3, 5]), Hierarchical Feature Learning is also used. Its disadvantages are **High Computational Cost** and **Requires Good Architecture Design**. It requires a very long training time and more effort is needed in the basic model design, such as in the arrangement and combination of hyperparameters, we have 3888 kinds, but due to the consideration of time cost, we only conducted training for 36 kinds of combinations. Its confusion matrix does not have obvious weaknesses. The weakest category, class 7, has a recall of 0.78. Its training should be very successful.

5. Conclusion and Future Work

In conclusion, we believe that CNN model has the best performance. Firstly, its performance has the best performance compared to other models we trained in this assignment. The accuracy, precision, recall, and F1-score are all quite high. The number of misclassifications in the confusion matrix is also much lower than the other two. The recall for each category remains at a relatively high level, and the lowest recall is as high as 0.78. Although the interpretability of CNN is relatively more difficult to explain compared to Decision Tree, because it learns features and mixes multiple filters for data training and interpretation, it is still possible to understand the model's focus area through visualization tools (such as Grad-CAM). In terms of training time, the training time of CNN is indeed much longer than the other two (approximately 2264 seconds), but considering its excellent performance, this cost is not unacceptable. On better equipment and better GPUs, the training time should be faster. Considering the number of errors, we are more inclined to have a slightly longer training time. Regarding the equipment, as we mentioned earlier, because the total number of hyperparameter combinations for CNN is 3888, due to hardware and time limitations, we could not train all 3888 combinations. For future work directions, we think if there is an opportunity to use better equipment, we should train all combinations and expand the search space, such as increasing the number of layers. At the same time, the number of training times should be increased, and mechanisms such as cross-validation should be introduced to study whether it has overfitting. We also need to further explore the generalization ability of this model in other multi-classification tasks to see if it can be applied to most classification problems.

6. Reflection

Student A:

I learned a lot about how to choose and tune machine learning models depending on the type of data throughout this assignment. At first, I didn't realize how much impact the model architecture and hyperparameters had on performance. I struggled a bit with understanding why CNNs worked so much better on image data compared to MLP or Decision Trees. By doing this assignment, I have read through a good amount of papers related to the use of deep learning models in images' classification applications. By reading through those papers, I gained a better understanding of the architecture and the working flow in these models we have used in this assignment. After experimenting and seeing the results, it became clear how important spatial features are.

One of the most valuable lessons was learning how to properly evaluate models. Evaluating models is not just about accuracy, but metrics like precision, recall, and F1-score helped me see where the model was failing. This assignment also helped me feel more confident using Keras Tuner. Moreover, I understand how to balance overfitting and underfitting through validation. It wasn't easy at times, but working through those challenges helped me gain a much deeper understanding of practical machine learning.

Student B:

Throughout the entire assignment, my study of the code became more in-depth. Previously, my understanding of max_trials was merely that it involved randomly selecting a few numbers for testing, and the more numbers, the better. However, during subsequent exploration, I discovered that the selection of max_trials should be related to the number of my combinations. For MLP, there are only 36 combinations, and if the equipment allows, I should run all of them to check their training results. In contrast, for CNN, there are 3888 combinations. I think the optimal number should be 200. However, CNN has a large number of combinations, and each combination takes a long training time. Based on the current equipment conditions, selecting 36 times can also slightly explore the tip of the iceberg of CNN, but this data should be used as a reference only because the data volume is indeed too small compared to 3888. Regarding how to correctly evaluate the model, I found that in addition to accuracy, there are many other parameters that can be referred to. For example, confusion matrices, precision, recall, F1-score, and training time in a comparison table. This really makes me not be limited to just one accuracy, but should also consider a model's quality from other more aspects.

7. References

1. Yang, J., Shi, R., Wei, D., Zhao, L., Wang, Z., Wang, Y., Zhang, Y., & Zhou, J. (2023). *MedMNIST v2: A large-scale lightweight benchmark for 2D and 3D biomedical image classification*. Scientific Data, 10, 41. https://doi.org/10.1038/s41597-022-01721-8

2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

3. GeeksforGeeks. (n.d.). *Decision tree*. https://www.geeksforgeeks.org/decision-tree/

4. GeeksforGeeks. (n.d.). *K-Nearest Neighbours*. https://www.geeksforgeeks.org/k-nearest-neighbours/

5. GeeksforGeeks. (n.d.). *Feedforward neural network*. https://www.geeksforgeeks.org/feedforward-neural-network/

6. GeeksforGeeks. (n.d.). *Introduction to Convolutional Neural Network*. https://www.geeksforgeeks.org/introduction-convolution-neural-network/

7. Keylabs. (n.d.). *Decision trees: How they work and practical examples*. https://keylabs.ai/blog/decision-trees-how-they-work-and-practical-examples/

8. AIML. (n.d.). *What are the advantages and disadvantages of using a decision tree?* AIML.com. Retrieved May 8, 2025, from https://aiml.com/what-are-the-advantages-and-disadvantages-of-using-a-decision-tree/

9. Tahir, F., Akram, M. U., Siddiqui, A. M., & Yousaf, A. (2020). Principle of neural network and its main types: Review. *ResearchGate*. https://www.researchgate.net/publication/343837591_Principle_of_Neural_Network_and_Its_Main_Types_Review

10. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

11. GeeksforGeeks. (n.d.). *Why deep learning is black box?* https://www.geeksforgeeks.org/why-deep-learning-is-black-box/

12. Rokach, L., & Maimon, O. (2014). *Data mining with decision trees: Theory and applications* (2nd ed.). World Scientific.

13. 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.

14. Ahmad, F. (2021, January 5). *Stochastic Gradient Descent (SGD) and Adam*. Medium. https://medium.com/data-scientists-diary/stochastic-gradient-descent-sgd-and-adam-4fe496ef1bbf

15. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1412.6980

16. Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., ... & Zhang, K. (2018). *Identifying medical diagnoses and treatable diseases by image-based deep learning*. *Cell*, 172(5), 1122-1131.e9. https://doi.org/10.1016/j.cell.2018.02.010

17. Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE, 10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

18. Chinchor, N. (1992). MUC-4 Evaluation Metrics. In *Proceedings of the 4th Message Understanding Conference* (pp. 22–29)..

19. 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.

20. Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

21. Chollet, F. (2021). *Simple MNIST convnet* [Keras code example]. Keras. https://keras.io/examples/vision/mnist_convnet/

22. Chollet, F. (2021). *Simple MNIST convnet* [Keras code example]. Keras. https://keras.io/examples/vision/mnist_convnet/

23. Mueller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: A guide for data scientists*. O'Reilly Media.