COMP5318/4318 Machine Learning Report

group: A2 group165
SID1: 540876279, SID2: 540871296, SID3: 540613397
instructor: Dr. Irena Koprinska
May 12, 2024

Content of table

1. Introduction

The study aims to systematically compare and fine-tune multiple predictive models—including Random Forest (RF), Multi-Layer Perceptron (MLP), and Convolutional Neural Networks (CNNs)—for the task of multi-class classification of histopathological medical images. The primary objective is to evaluate each model's performance in terms of classification accuracy, generalization ability, and robustness under varying input conditions.

This task holds significant practical relevance in the field of digital pathology, where automated image-based diagnosis supports early disease detection, reduces clinical workload, and enhances patient outcomes. Precise model selection and tuning are crucial, as even small performance gains can yield significant clinical value. By identifying the most effective model, this study advances the development of reliable computer-aided diagnosis (CAD) systems in medical imaging.

2. Data

2.1 Description and Exploration

The dataset includes 32,000 training and 8,000 test images stored in NumPy arrays, each sized 28×28 with 3 RGB channels. A total of 9 classes are present shown in appendix1, corresponding to tissue types such as adipose, tumor, muscle, glandular, stromal, and background. We visualized representative samples from each class and mapped them to anatomical labels based on domain knowledge. No major class imbalance or corrupted samples were observed, suggesting the dataset is clean and well-structured for classification tasks.

2.2 Pre-processing

1. **Normalization**
   Pixel values ranging from 0 to 255 were scaled to the 0,10,1 range. This standardization ensures consistent feature magnitudes, improving gradient stability during training.
2. **PCA (for ML/MLP models only)**
   Principal Component Analysis was applied prior to training traditional models and MLPs. It reduces dimensionality, suppresses noise, and improves training speed and model generalization. However, PCA was not applied to CNNs, since convolutional layers inherently extract low-dimensional local features.
3. **Missing Values**
   No missing value imputation was needed. In image data, pixel values of 0 (black) or 255 (white) are meaningful and reflect real structures (e.g., background or tissue boundaries). Treating them as missing would distort spatial information.
4. **Class Imbalance**
   Class distributions were checked via histogram plots and found to be roughly balanced across the 9 categories. As a result, no resampling (e.g., oversampling or class weighting) was necessary.

5. **One-Hot Encoding**
   Since the models used `sparse_categorical_crossentropy`, label vectors could be left as integers without one-hot encoding. This reduced memory usage and preprocessing steps while preserving full compatibility with the loss function.

3. Methods

3.1 Random Forest

3.1.1 Theory

Random forest is an ensemble learning method based on decision trees, aiming to improve the accuracy and robustness of the model. Its working principle is to construct a large number of decision trees during the training process and output the patterns of the classes predicted by a single tree. The diversity among trees is achieved through two mechanisms: bootstrap sampling (bagging) of training data and random feature selection at each segmentation, which helps to reduce the correlation among trees.
This algorithm effectively alleviates the overfitting problem and improves the generalization ability by averaging multiple weak learners. Each tree independently learns from a random subset of data and features, enabling the random forest to resist noise and outliers. This feature enables the model to perform complex classification tasks well with limited hyperparameter tuning.

3.1.2 Strengths and Weaknesses

Advantages:
• Robustness against overfitting: Compared with a single decision tree, the ensemble feature of random forests minimizes overfitting and provides powerful generalization performance on various datasets.
• Processing high-dimensional data: Random forests can effectively manage large feature Spaces without the need for extensive feature selection or engineering.
• Interpretability: The feature importance score can be easily extracted, providing insights into the contribution of individual variables to model decision-making.

Disadvantage:
• Computational cost: For large datasets and a large number of trees, training and inference may be relatively slow because each decision tree must be traversed.
• Lack of smooth probability estimation: Compared with probability models, random forests may produce fewer calibrated probability outputs for class prediction.

3.1.3 Architecture and Hyperparameter Tuning

Through preliminary experiments, the grid search method is adopted to determine the optimal configuration of the hyperparameters of the random forest. According to the sequence of the expected impacts, the adjustment process is aimed at the following parameters:
1. Number of trees (n_estimators) : (100,150,200,250)
Increasing the number of trees can continuously improve the accuracy rate, up to 250 trees. Beyond 250 trees, the accuracy rate becomes negligible.
2. The maximum depth of the tree (max_depth) : (None, 20,40)
A depth of 20 is the best trade-off between learning complex patterns and preventing overfitting. Shallower trees do not conform to the data, while deeper trees do not offer obvious additional benefits.
3. The number of features considered in each segmentation (max_features) : ('sqrt', 'log2')
Compared with "log2", the "sqrt" setting offers higher accuracy and more stable results.

Grid search conducted three hierarchical cross-validations on the training data and carried out a total of 72 model evaluations. The optimal configuration consists of n_estimators=250, max_depth=20 and max_features='sqrt', with an average cross-validation accuracy rate of 64.4%. This combination was then applied to train the final random forest model on the complete training set.
This architecture and tuning strategy balance the model performance, stability and computational efficiency, making the random forest a powerful baseline method for image classification tasks.

3.2 MLP

3.2.1 Theory

A Multi-Layer Perceptron (MLP) is a type of feedforward neural network comprising an input layer, one or more hidden layers, and an output layer. Each neuron applies a weighted linear transformation followed by a non-linear activation (e.g., ReLU or tanh), enabling the model to learn complex decision boundaries. The core computation at each layer follows Z=WX+b, with outputs passed through activation functions.
MLPs are trained by minimizing a loss function (sparse categorical cross-entropy) using optimization methods like Stochastic Gradient Descent (SGD), with gradients computed via back propagation. While MLPs are powerful universal approximators, they lack the ability to capture spatial hierarchies in image data, limiting their performance compared to CNNs.

3.2.2 Strengths and weaknesses

The Multi-Layer Perceptron (MLP) offers several advantages when applied to classification tasks. MLPs are highly flexible universal function approximators capable of learning complex, non-linear relationships between input data and output labels. Due to their simple architecture, they are relatively straightforward to implement and computationally efficient for small-to-medium sized datasets. The fully connected nature of MLPs ensures that every neuron in a layer is connected to all neurons in the next layer, allowing the model to learn global patterns in the data.

However, MLPs also exhibit key limitations, especially in computer vision tasks. They lack the ability to preserve spatial structure and local correlations inherent in images. The flattening of image data prior to input discards spatial locality information, which often leads to sub-optimal performance compared to spatial-aware architectures like Convolutional Neural Networks (CNNs).

Additionally, MLPs are prone to overfitting when applied to high-dimensional datasets or insufficient training data. They require careful tuning of hyperparameters such as the number of hidden layers, number of neurons, learning rate, and dropout rates to balance underfitting and overfitting. In this project, greedy tuning was applied to mitigate this issue and select optimal hyperparameters.

In summary, MLPs serve as a strong baseline model due to their simplicity and versatility, but are often outperformed by CNNs for complex image classification tasks due to their inability to effectively capture local spatial features.

3.2.3 Model Architecture & Hyper-parameters tunning

The MLP model implemented in this study was designed as a fully connected feedforward neural network. The input images from the PathMNIST dataset were flattened and normalized to the range [0, 1]. Principal Component Analysis (PCA) was applied to reduce the feature space from 2352 (28×28×3) to 330 components to enhance computational efficiency and training stability.

In order to determine the optimal architecture, three configurations were evaluated:

- A single hidden layer MLP resulted in underfitting and poor generalization, as the model lacked sufficient capacity to capture the complexity of the data.
- A three hidden layer MLP showed signs of overfitting, with excessive abstraction leading to poor generalization to unseen data.
- A two hidden layer MLP provided the best balance between learning capacity and generalization, and was chosen as the final architecture.

The final MLP architecture was composed of:

- Input layer: 330 nodes (PCA reduced features)
- Dense layer 1: 256 neurons, ReLU activation
- Dropout layer: 0.2 dropout rate
- Dense layer 2: 256 neurons, ReLU activation
- Dropout layer: 0.2 dropout rate
- Output layer: 9 neurons, softmax activation

Hyperparameter optimization was conducted using a greedy search strategy, adjusting one hyperparameter at a time while keeping others fixed:

1. : Larger units improved performance; 256 neurons were selected.
2. : ReLU exhibited faster convergence and superior accuracy.
3. : 0.2 provided the best regularization without loss of learning capacity.
4. : A learning rate of 0.1 led to the fastest convergence with stable training.

The model was compiled using the Stochastic Gradient Descent (SGD) optimizer and trained to minimize sparse categorical cross-entropy loss. Two callback functions, EarlyStopping and

ReduceLROnPlateau, were used to prevent overfitting and adaptively reduce the learning rate when validation performance plateaued.
This systematic tuning procedure resulted in a final test accuracy of 68.59% on the PathMNIST dataset.

3.3 CNN

3.3.1 Theory

Convolutional Neural Networks (CNNs) use convolutional layers to scan images with kernels, detecting localized features such as edges or textures. Pooling layers then downsample these feature maps, reducing spatial resolution while retaining key structural patterns. These operations enable CNNs to learn hierarchical representations, from low-level features in early layers to complex patterns in deeper layers. Fully connected layers near the output aggregate learned representations for classification. The shared weights and spatial locality of CNNs make them especially effective for image tasks.

3.3.2 Strengths and Weaknesses

Strengths:

- High performance on image data: CNNs are well-suited for tasks involving spatial hierarchies, such as medical imaging, due to their ability to learn local and global features.
- Parameter efficiency: Shared weights reduce the number of parameters compared to dense networks, improving generalization and reducing overfitting.

Weaknesses:

- High computational cost: Especially without PCA, CNNs process full-resolution image data, requiring significant training time.
- Interpretability: The depth and complexity of CNN architectures make them difficult to interpret or visualize.
- Overfitting risk: High parameter counts can lead to overfitting, especially on small datasets.
- Hyper-parameter sensitivity: Performance is highly dependent on design choices, requiring careful tuning.

3.3.3 Architecture & Hyper-parameters tunning

After testing multiple structures, the most effective configuration used two convolutional layers followed by one pooling layer. This setup achieved up to ~90% validation accuracy, as it preserved spatial detail and allowed richer hierarchical feature extraction—important for subtle variations in medical tissue.

Hyper-parameter Tuning Strategy
A greedy search method was employed to reduce computational burden. Parameters were tuned in priority order based on expected influence:

1. Filter Number: (16, 32, 64)
    - 32 filters in the first layer captured low-level patterns with minimal computation.

    - 64 filters in the second layer allowed more complex abstractions.
    - This configuration appeared in most of the top-performing models, balancing fine-grain detection with contextual awareness.
2. Kernel Size: (3x3 first layer, 5x5 second layer)
    - The 3×3 kernel in the initial layer efficiently captured fine-grained local features such as edges and textures.
    - the larger 5×5 kernel in the second layer enabled the model to integrate broader contextual information.
    - This progressive increase in receptive field allowed deeper feature abstraction without significantly increasing computation.
3. Dropout Rate: (0.3, 0.4, 0.5)
    - 0.3 provided the best regularization. Higher values led to under-fitting.
4. Learning Rate: (0.01, 0.001, 0.0001)
    - 0.01 with decay achieved stable convergence. Lower rates were too slow or suboptimal.
5. Dense Layer Neurons: (32, 64, 100, 200)
    - 64 neurons performed comparably while reducing risk of overfitting and improving runtime.
6. Activation Function: (ReLU, Tanh)
    - ReLU was selected due to faster convergence and superior empirical results.

Justification for Search/Inference Method (Greedy Search): We chose greedy search over randomized or grid search due to the relatively small dataset and long CNN training time. Greedy search allowed us to incrementally optimize hyper-parameters in order of expected importance while minimizing computation. This strategy helps avoid overfitting and focuses efforts on the most sensitive parameters first. Although greedy search does not guarantee a global optimum, its results were robust across multiple initial configurations, supporting its practical utility.

This configuration balanced accuracy, interpretability, and computational efficiency for the task of multi-class classification on medical images.

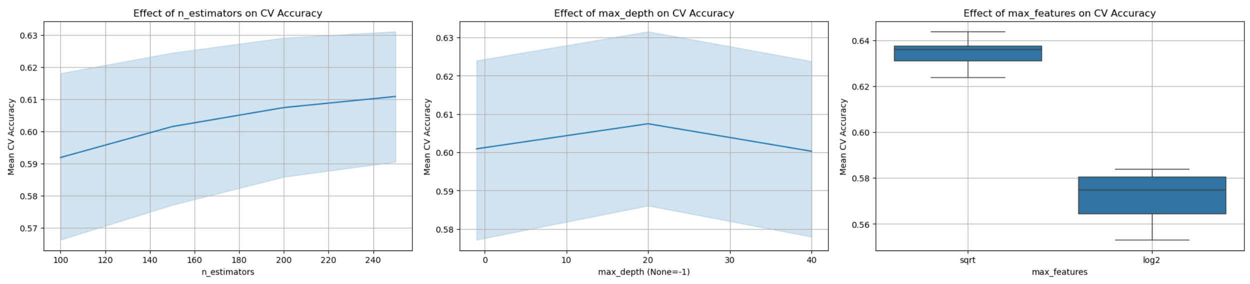
4. Results and Discussion

4.1 Random Forest

4.1.1 Hyperparameter Tuning - Random Forest

Random Forest (RF) is used as the benchmark traditional machine learning method in this study. The model is trained using the pca dimension reduction dataset to reduce the dimension and improve the computational efficiency. Perform grid search to optimize three key hyperparameters: the number of estimators (100,150,200,250), the maximum tree depth (None, 20,40), and the number of features considered at each segmentation ('sqrt', 'log2').

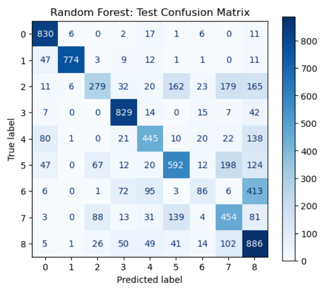
The optimal configuration was determined as n_estimators=250, max_depth=20, and max_features='sqrt', achieving an average cross-validation accuracy of 64.4% in three cross-validations. On the validation set, the accuracy rate of the model is 64.9%, the weighted f1 score is 0.629, the generalization is good, and there is no serious overfitting.



4.1.2 Model Analysis- Random Forest

The finally adjusted RF model was evaluated on the test set, and the result accuracy rate was 64.3%, with a weighted f1 score of 0.614. The classification report shows that it has a strong predictive performance for categories 0, 1 and 3, while there are obvious misclassifications among categories 2, 6 and 7. This pattern indicates that although random forests are robust and easy to train, they lack the representational ability of deep learning models and are unable to effectively separate visually similar tissue types.

The confusion matrix further confirmed this observation, highlighting the severe confusion among categories with overlapping visual or texture features. Although compared with cnn, random forest requires the least preprocessing and less training time, its overall classification performance is relatively low, especially on fuzzy or minority classes.



In conclusion, random forests provide a fast and interpretable baseline, but are limited in their ability to handle the complex spatial features present in histopathological images. These results emphasize the advantages of feature learning methods (such as convolutional neural networks) in the task of medical image classification.

4.2 MLP

4.2.1 Hyperparameter tunning

To optimise the MLP model, a greedy tuning strategy was employed to sequentially adjust one hyperparameter at a time while holding others fixed. This approach significantly reduced the overall search space compared to an exhaustive grid search and allowed for faster convergence towards an optimal configuration. The following four hyperparameters were considered:

- **Number of neurons in hidden layers**: Three configurations (64, 128, 256) were evaluated. Increasing the number of neurons generally improved model capacity and performance, with 256 neurons yielding the best validation accuracy without overfitting.

- **Activation function**: Both ReLU and tanh activation functions were tested. ReLU was selected due to its superior performance and faster convergence, as well as its ability to mitigate vanishing gradient issues.
- **Dropout rate**: Values of 0.2, 0.3, and 0.4 were examined to balance regularisation and learning capacity. A dropout rate of 0.2 was found to provide sufficient regularisation while maintaining strong learning dynamics.
- **Learning rate**: Candidates of 0.1, 0.01, and 0.001 were assessed. A learning rate of 0.1 enabled faster training with stable convergence and did not exhibit instability during training.

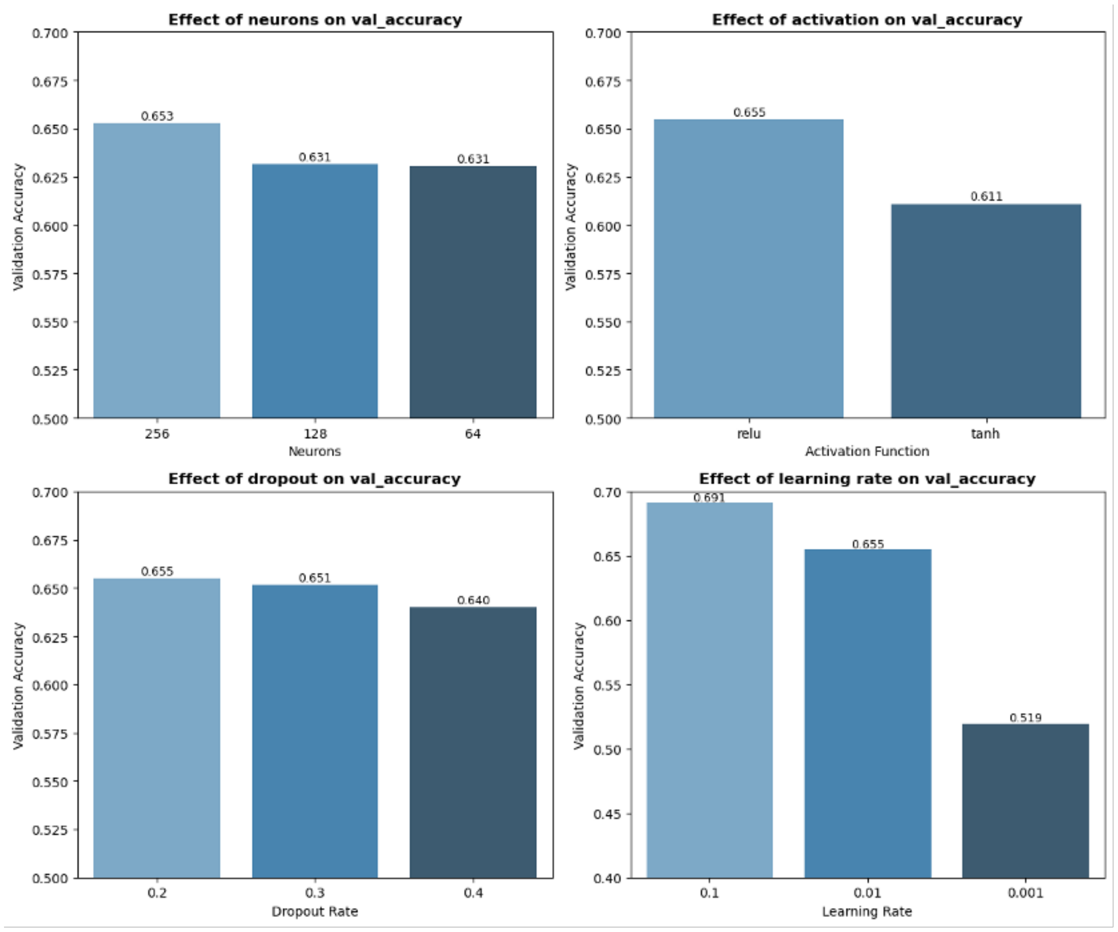


Figure 4.2.1: Effect of hyperparameters on val_accuracy

The tuning process involved training multiple MLP models for 10 epochs on the training dataset and evaluating the validation accuracy at each step. The best hyperparameter combination was selected based on the highest validation performance.

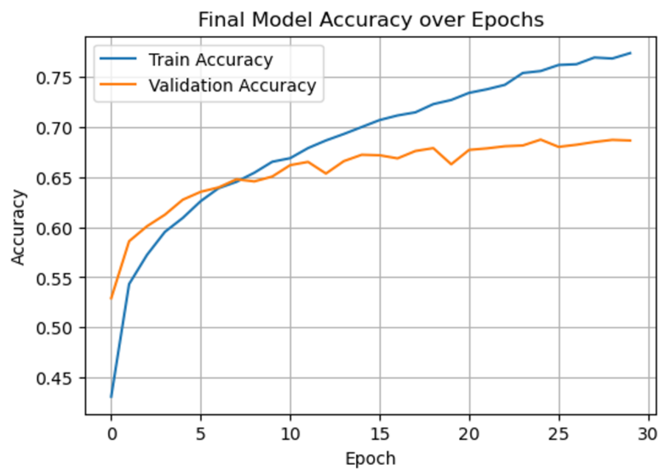
The final optimal hyperparameters obtained after tuning were as follows:

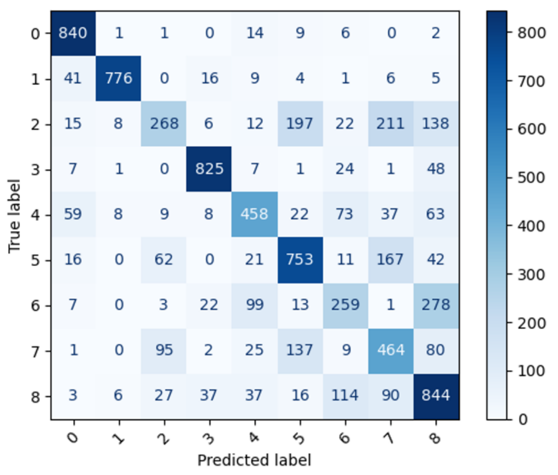
- Hidden units: 256
- Activation function: ReLU
- Dropout rate: 0.2
- Learning rate: 0.1

This iterative tuning procedure effectively enhanced model generalization capability and mitigated overfitting risk, resulting in an improved and stable model for the PathMNIST classification task.

4.2.2 model analysis

The MLP model was trained for 30 epochs using the optimised hyperparameters from Section 4.2.1. The training and validation accuracy over epochs is shown in Figure 4.2.2.1. The model shows a stable upward trend in accuracy without serious overfitting. The final model achieved a test accuracy of **68.65%** on the PathMNIST test set.





True \ Predicted	0	1	2	3	4	5	6	7	8
0	840	1	1	0	14	9	6	0	2
1	41	776	0	16	9	4	1	6	5
2	15	8	268	6	12	197	22	211	138
3	7	1	0	825	7	1	24	1	48
4	59	8	9	8	458	22	73	37	63
5	16	0	62	0	21	753	11	167	42
6	7	0	3	22	99	13	259	1	278
7	1	0	95	2	25	137	9	464	80
8	3	6	27	37	37	16	114	90	844

Figure 4.2.2.1: Accuracy over Epochs Figure 4.2.2.2 : Confusion Matrix

A detailed classification report including precision, recall, and F1-score for each class is presented in Table 4.2.2.1. The model performs well on classes 0 to 4, achieving high F1-scores, but struggles more with classes 5, 6, and 7 due to visual similarities and class imbalance.

Class	Precision	Recall	F1-score	Support
0	0.85	0.96	0.90	873
1	0.97	0.90	0.94	858
2	0.90	0.87	0.87	877
3	0.90	0.90	0.90	914
4	0.67	0.62	0.65	737
5	0.50	0.38	0.43	682
6	0.47	0.57	0.52	813

7	0.56	0.72	0.63	1174
8	0.56	0.72	0.63	1174
Accuracy	-	-	**0.69**	**8000**
Macro avg	0.68	0.67	0.67	8000
Weighted avg	0.69	0.69	0.68	8000

Table 4.2.2.1

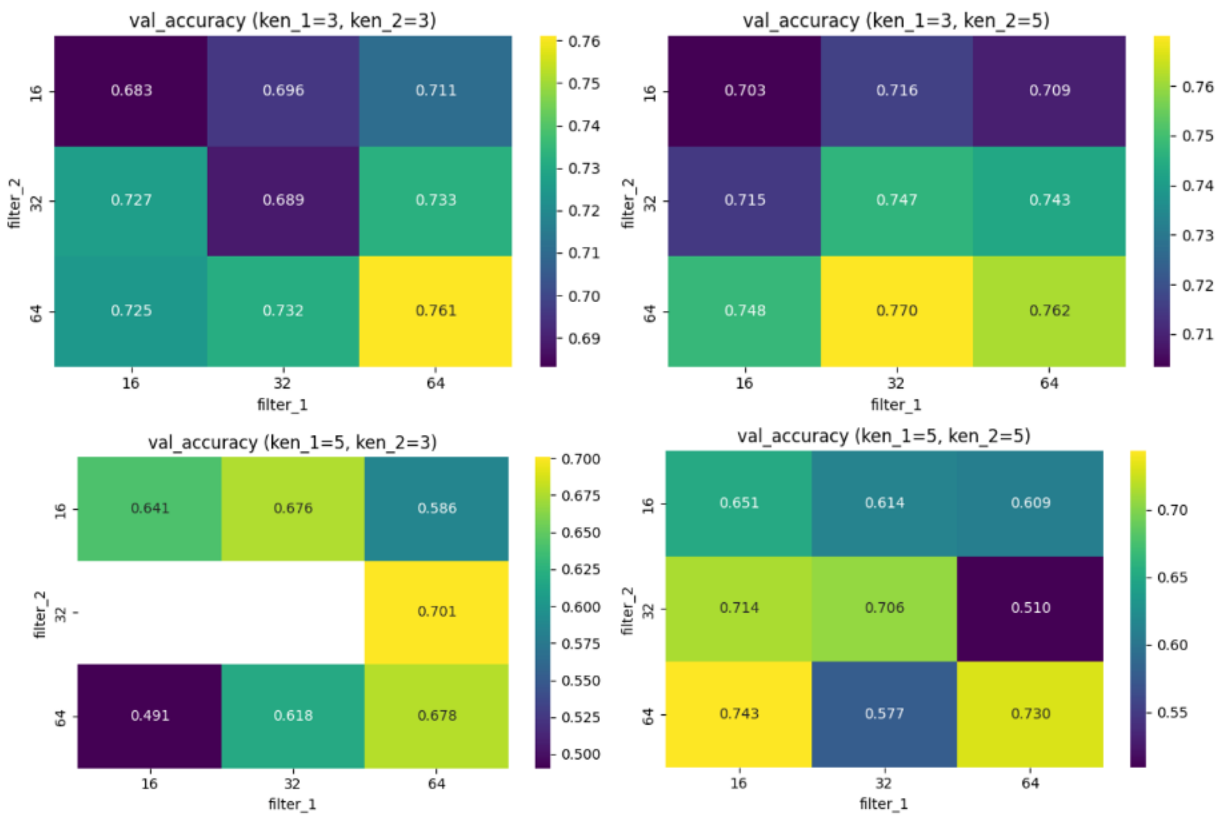
The confusion matrix in Figure 4.2.2.2 shows the distribution of true versus predicted labels. The most common misclassifications occurred between class 4 and class 6, and between class 5 and class 7. This suggests that while the MLP is able to learn global patterns effectively, it lacks the spatial feature extraction capabilities of CNNs.

Overall, the MLP model provides a strong baseline for multi-class classification on this dataset but is likely to be outperformed by spatial-aware architectures like convolutional neural networks.

4.3 CNN

4.3.1 Hyperparameter Tuning - CNN

Due to limitation of computation, we only show one heat map of two key hyper parameter, the rest hyper parameter pair have the similar analysis:



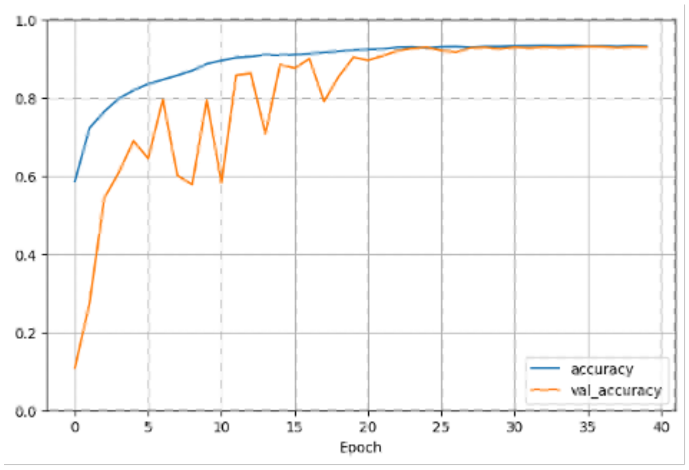
The heat map of show ken_1=3 and ker_2=5 is better than any else kernel combination of kernel size since it has higher value on almost every filter matrix. Since most model perform well in this combination, we used it as first greedy search approach and then choosing the filter size.

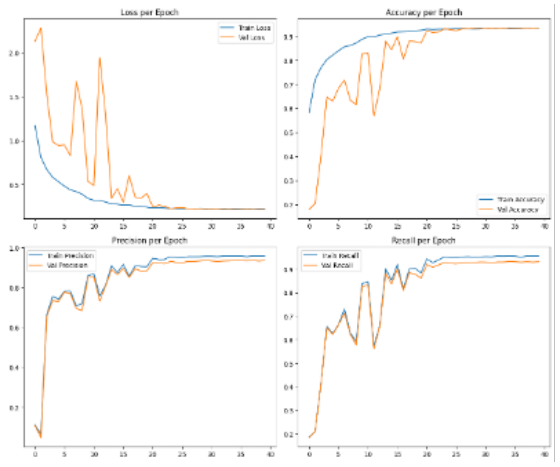
From rule of thumb, second layer usually use larger filter size compare to first one since it need more complex abstraction to extract feature. From the heat map (ken_1=3, ker_2=5), we can see filter_1=32, filter_2=64 achieve highest validation accuracy.

Final Model Configuration (best hp):

- Conv1: 32 filters, 3x3, ReLU
- Conv2: 64 filters, 5x5, ReLU
- Dropout: 0.3
- Dense Layer: 64 neurons, ReLU
- Learning Rate: 0.01 with decay

The final test accuracy reached 93.21, and from validation accuracy, loss, accuracy, precision and recall we call see that the model stop increase it performance from epoch 25. And we may set an early stop for the model at epoch 25.





4.3.2 Model Analysis - CNN

We evaluated a Convolutional Neural Network (CNN) trained over 40 epochs, using kernel sizes of (3×3)-(5×5), filter numbers of 32 and 64 in two convolutional layers, and dropout regularization. Learning rate scheduling via `ReduceLROnPlateau` helped stabilize training. The final test accuracy reached **93.21%**, with a macro F1-score of **92.99%** and weighted F1-score of **93.19%**, indicating strong generalization across classes.

Best Hyperparameters and Performance Summary

Model	Kernel Sizes	Filter Numbers	Dropout	LR Scheduler	Epochs	Test Accuracy	F1 Score (macro)	Time per Epoch	Total Time
CNN	(3,3)-(5,5)	32-64	0.3	Yes	40	93.21%	92.99%	~7s	~5 min

Classification Report Insights

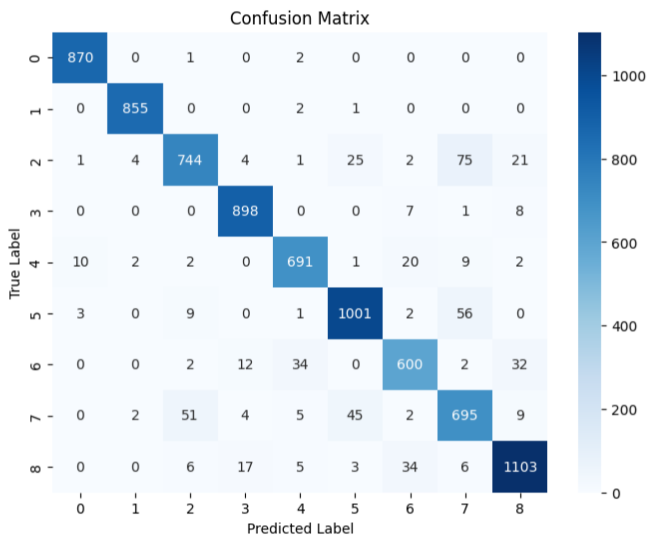
From the classification report and confusion matrix:

- Classes **0, 1, 3** were predicted with the highest accuracy and F1-scores (≥0.97), reflecting their clear feature patterns.
- Class **7** had the lowest precision (0.8235) and F1-score (0.8389), often being confused with class 5 or 6, as visible in the confusion matrix.
- The macro average F1-score (0.9299) confirms balanced performance, not overly biased toward high-support classes.

Confusion Matrix Analysis

The confusion matrix reveals:

- **Class 2** shows substantial misclassification into class 7 and 8 (~75 and 21 samples respectively), likely due to visual similarity or feature overlap.
- **Class 5** is confused with class 7 (56 samples), possibly due to shared patterns in pixel intensity or shape.
- **Class 6** had some confusion with class 3 and 4 — highlighting room for improvement in capturing fine-grained features.



Training Dynamics and Runtime Factors

As shown in the learning curves:

- Training and validation accuracy steadily increased, with minor overfitting spikes around epoch 8 and epoch 12, which were handled by LR reduction.
- The loss converged well, dropping from 1.39 to ~0.22, with smoother validation loss after epoch 15.
- Learning rate schedule (starting at 0.001, reducing down to 1.95e-6) was crucial in fine-tuning and stabilizing training after initial volatility.

- Time per epoch ranged between 5–11 seconds, averaging ~7 seconds. The full training time was under 5 minutes, thanks to moderate model complexity and input resolution.

Interpretation in Light of CNN Theory

This performance aligns with CNN's theoretical strengths:

- **Convolutional layers** effectively extracted local features, especially for well-structured digits (like class 1 and 3).
- **Weight sharing and pooling layers** provided translation invariance and reduced overfitting.
- **Dropout and LR scheduling** helped control model complexity and optimize learning dynamics.

However, **CNNs' limitations in fine-grained class distinction** (e.g., between class 2, 5, 7, 8) suggest that incorporating **multi-scale filters**, **attention mechanisms**, or **residual connections** may further boost performance, particularly for ambiguous or visually similar classes.

4.4 Model Comparison

Model	Accuracy	Runtime (s)	Best Hyperparameters
PCA+RF	0.64	Short	{'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 250}
PCA+MLP	0.69	Short	{'units': 256, 'activation': 'relu', 'dropout': 0.2, 'lr': 0.1}
CNN	0.93	High	{'filter': (32, 64), 'kernel': (3,5), 'lr': 0.01, 'dropout': 0.3, 'neuron': 64, 'activation': relu}

Among the three evaluated models, CNN achieved the highest classification accuracy (0.93), demonstrating strong capability in capturing spatial patterns from raw image inputs. However, this came at the cost of increased training time due to higher model complexity.

PCA+MLP showed moderate performance (accuracy: 0.69) and retained low computational cost. Its fully connected architecture allowed for non-linear decision boundaries, but it lacked spatial awareness and required careful regularization to avoid overfitting.

PCA+RF was the most interpretable and fastest to train, with an accuracy of 0.64. While suitable for baseline comparisons, RF struggled with high-dimensional image features even after PCA dimensionality reduction.

Overall, CNN is the preferred choice for accuracy-critical applications, while RF and MLP may be considered when interpretability or efficiency is a primary concern.

5. Conclusion

This study compared Random Forest (RF), Multi-Layer Perceptron (MLP), and Convolutional Neural Networks (CNNs) for multi-class classification of histopathological images. CNNs outperformed the other models in accuracy, achieving strong performance due to their ability to exploit spatial hierarchies in image data. MLPs showed moderate accuracy but suffered from longer training time and weaker generalization, while RF offered high interpretability and fast runtime but lacked deep representational power.

A key limitation was the restricted image resolution and dataset size, which constrained the performance ceiling for all models. Additionally, tuning CNNs involved longer computation time and was less interpretable than tree-based methods.

For future work, we suggest incorporating **attention mechanisms** or **residual connections** into the CNN architecture to better capture subtle texture differences and improve gradient flow, particularly for visually similar classes. This approach balances performance gains with manageable complexity and aligns with current trends in medical image analysis. Additionally, integrating **model interpretability tools** such as Grad-CAM would enhance the clinical transparency of CNN predictions, making them more suitable for real-world deployment.

6. Reflection

zliu0364: I learned how architectural tuning (e.g. Conv–Conv–Pool vs. Conv–Pool–Conv) affects performance, and that even small design changes (e.g. filter size, dropout) can significantly shift results. Greedy search gave me hands-on insight into structured hyperparameter exploration and balancing model complexity with performance.

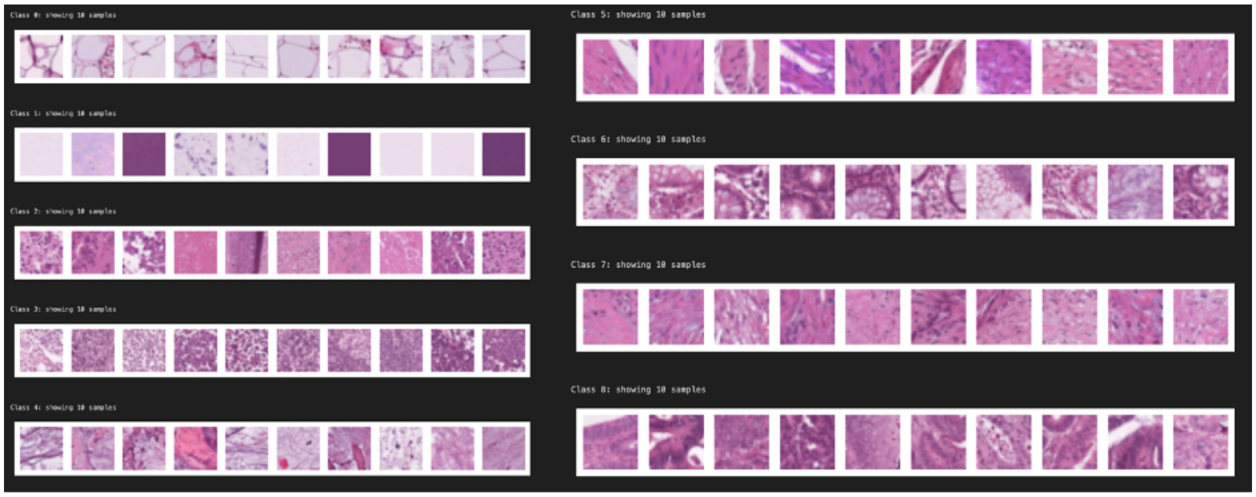
clyu0346: Working with Random Forest models gave me a valuable perspective on how traditional machine learning differs from deep learning approaches. I appreciated the simplicity of the pipeline, where minimal data preprocessing was required compared to neural networks. Through experimenting with hyperparameters like the number of estimators and tree depth, I learned how ensemble methods balance variance and bias to improve model stability. This experience reinforced my understanding of the importance of model interpretability and efficiency, especially when working with smaller datasets or limited computational resources.

cliu0758: Completing this assignment helped me significantly deepen my understanding of neural network model design and optimization. I learned how to apply practical strategies such as greedy hyperparameter tuning to efficiently explore a large search space. One major challenge was balancing model capacity and overfitting, especially when tuning hidden layer size and learning rate. Additionally, using callbacks like EarlyStopping and ReduceLROnPlateau greatly stabilized the training process. I also gained experience in interpreting learning curves and tuning visualization plots to support model evaluation. Going forward, I would explore more advanced regularization techniques and potentially use Bayesian optimization for tuning to further improve performance and efficiency. Overall, this project provided valuable hands-on experience with real-world medical image data and deep learning pipelines.

7. Reference

Keras Team. (n.d.). *Keras API reference*. Keras. https://keras.io/api/
Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444. https://doi.org/10.1038/nature14539
Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Appendix 1


Class 0: showing 10 samples
Class 1: showing 10 samples
Class 2: showing 10 samples
Class 3: showing 10 samples
Class 4: showing 10 samples
Class 5: showing 10 samples
Class 6: showing 10 samples
Class 7: showing 10 samples
Class 8: showing 10 samples