PathMNIST

Task : Image Classification

1. AIM:

The objective of this study is to perform a comparative evaluation of several machine learning models for a Path MNIST dataset. The dataset contains microscopic images of 9 different tissue types. We are implementing 3 different algorithms: a classical algorithm i.e. Random Forest, a fully connected neural network and a convolutional neural network. Our goal is to recognize the impact of hyper tuning and identify the architecture which is efficient at attaining high performance and handling the challenges of such datasets – inter-class similarity, image quality and noise, class imbalance.

2. IMPORTANCE:

Multiclass image classification is associated with real-world scenarios like medical images, object & face recognition, species identification, quality inspection, satellite image classification. The machine learning pipeline is implemented including data exploration, preprocessing, model development, hyper tuning, and comparative evaluation. This task helps to understand the strengths and weakness of traditional vs modern machine learning algorithms and how they provide trade-offs between complexity, runtime, cost, accuracy in real world use cases. The implementation showed that traditional algorithms were less suitable for such raw image datasets, and neural networks should be preferred for such use cases.

3. Data description and exploration

The dataset used for this study is a subset of the original dataset from Path MNIST - Tissue Nets dataset (https://www.nature.com/articles/s41597-022-01721-8). It consists of 28x28 RGB images of 9 tissue classes representing normal or abnormal body tissues.

The data was provided in form of NumPy arrays segregated based on Train-Test and X-Y (x-test, x-train, y-test, y-train), containing unflattened images of dimension (N,28,28,3). The number of training images were 32000 and testing were 8000.

The provided dataset presents various challenges for multi class image classification. The images are having low resolution (28x28) which would aid in reducing the runtime but might affect in differentiating the details.

To check the class imbalance and understand the class distribution, a histogram of class labels was plotted (Figure 1). There was a slight imbalance, with some classes i.e. class 8 and class 5 are more represented than others i.e. class 6. This could skew model learning in unregularized models or when accuracy of the model is the sole evaluation criteria.

Figure 1: Class distribution in the training

One example of each class was visualised (Figure 2) to determine inter and intra class similarity. Some classes like 0,1 had visually distinct appearance whereas 6,7,8 was visually similar. Some images were noisy, blurred that could skew the model.

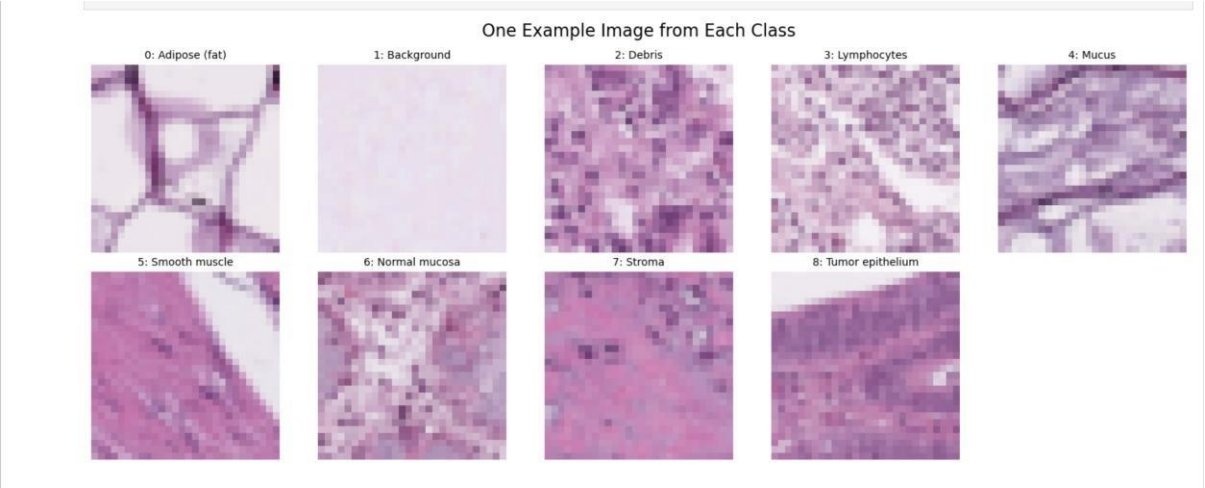

Figure 2: *Visual Image of one example image from each class in the Path MNIST dataset (0–8)*

4. Preprocessing Description and Justification

Several preprocessing steps were applied to ensure the inputs are compatible with the models. Despite the same dataset, preprocessing steps were customized depending on the algorithm's requirements.

a) Normalization: To prevent any feature to dominate the learning process, normalization

keeps input within a specific range. All pixel values were divided by 255 to scale them to [0,1] range.

Neural network models like MLP and CNN require normalisation for the following reasons:

- Speed up convergence: The normalised data is simpler for the optimizer to solve
- Improve numerical stability by preventing the activation function from saturating. The prevents the learning process to slow down or stop.

b) Flattening: For the MLP and random forest, images were flattened to a 1-day array of 2352 (28 x 28 x 3 (rgb) features. This is essential for random forest and MLP as they expect plain vectors i.e. can only process single row of features and cannot understand height x weight x color.

For CNN flattening was not required, and model was trained with the original 3d shaped image (28 x 28 x 3). This supports learning of spatial patterns for images.

3) One-hot encoding: The target class labels (9 classes) were encoded into one hot vectors for MLP and CNN for supporting categorical class entropy loss. They need the value to be spread across 9 positions like 2 = [0, 0, 1, 0, 0, 0, 0, 0, 0]. This helps the model to learn the probability distribution and calculate how wrong the predicted values are. Random forest can work with integer labels.

Preprocessing steps were limited deliberately to ensure fair comparison and evaluation on the provided raw data; the images were not subjected to dimensionality reduction or feature extraction. Only necessary steps were taken that would support the model.

5. Algorithm of Choice: Random Forest (RF)

To begin with, I selected KNN as the model from week 1-6, as it's a relatively simple and non-parametric algorithm. This model failed due to high dimensional data, presence of 2352 features (After flattening), and achieved only 29.6% prediction accuracy. Additionally, KNN struggles with noisy data and imbalanced classes [1]. Thereby, KNN was not a suitable algorithm for this dataset.

In response, I selected random forest, which uses multiple decision trees for attaining high accuracy, prevents overfitting and generalises well to new data [2]. It provides resistance to noise as output values are based on aggregating values from multiple trees or majority voting, thereby reducing the effects of individual noisy data points. [2][3] Each decision tree is trained with subset of features, which reduces the correlation between them and improves generalisation on unseen data.

Random forest requires tabular data as input, so each image was flattened from a 3D to a 1D vector of 2352 (28 x 28 x 3) features. This causes loss of spatial context, it allows the model to process the data.

The model achieved a **prediction accuracy of 64.48% (before hyper tuning)** on the data, which shows that it was able to generalise well despite the lack of spatial structures in the input. This shows the model works well with tabular data, even without image structure.

To optimize performance, grid search was performed to tune the hyperparameters such as the number of trees in the forest (n_estimators), maximum depth of each tree (max_depth), and the number of features to consider per split (max_features).

We chose these parameters as these parameters impact the computational efficiency and the bias-variance tradeoff for the random forest model. More n_estimators improves performance but also increases the time for training the model. Max_Depth controls the depth by limiting the growth of the tree to prevent overfitting. Max_features is responsible for the diversity (#no of features) and the robustness of the model

Best performance i.e. **prediction accuracy of 65.51% (after hypertuning)** was attained for the parameters: n_estimators = 150; max_features = 'sqrt'; max_depth = None (i.e., fully grown trees). This performance was better than KNN and the untuned MLP model. Even though random forest lacks spatial intelligence (like CNN), and input was provided as a long list of numbers (flattened), it still performed well and provided a benchmark for comparison.

6. Fully Connected Neural Network (Multilayer Perceptron - MLP)

Multi-Layer Perceptron is an artificial neural network that contains multiple layers of connected neurons, capable of learning complex relationships [4]. The output layers contain x neurons, where x is the #classes in the data, each neuron is responsible for calculating the probability of a particular class (unique x label) [5]. They are powerful and can handle nonlinear relationships in data but like random forest they also lack ability for spatial hierarchies.

MLP models expect a vector input [4], so each RGB images needs to be flattened into a 1-D array of 2352 features. This breaks down the spatial relationship between the pixels.

The architecture of the MLP model used in this task is:

- Input layer: 2352 neurons (28 x 28x 3) where 28 x 28 is the original pixels and 3 represents the 3 color channels in RGB. Each neuron represents 1 pixel. [4]
- Hidden layer: 128 neurons. ReLU activation function is used, which allows the model to interpret complex features and introduces non-linearity. This layer calculates the weighted sum of inputs, adds the bias and passes it through the activation function.
- Dropout layer: 30% dropout rate. This randomly deactivates subset of neurons during training to prevent a single neuron from getting too powerful and to reduce overfitting
- Output layer: 9 neurons responsible for the 9 output labels. Softmax activation function is used to convert the score to a probability distribution, which shows the predicted class as well as confidence level.

- Adam optimizer was used to train the model, which performs well on most problems by adapting the learning rate dynamically.
- Loss function: Categorical cross entropy calculates the predicted probability distribution and the one hot encoded value.

Training of the MLP model ran with a batch size of 128 over 10 epochs.

The test accuracy of the MLP model is : 31.20% (Untuned). This low performance of the MLP model is likely due to lack of spatial intelligence, tiny images and no convolutional layers. This shows that a single dense layer would not be sufficient for obtaining insights and features from raw data.

To optimize performance, grid search was performed to tune the hyperparameters such as the neuron units in the hidden layer, dropout rate, batch size and epochs.

These parameters were selected for tuning as they play role in the learning dynamics and generalisation performance. #Neurons directly affect the learning capacity of the model, few neurons could lead to underfitting, and too many could lead to overfitting the data. Batch size controls after how many samples the weights are updated; small values could offer precision and larger values would provide computational speed. Changing epochs helps to check if more iterations could help in convergence of the model.

Best performance i.**e test accuracy of 33.32% (Tuned)** was achieved for the parameters: 128 units, 0.3 dropout, batch size 128. The performance of the MLP model improved but is still underperformed as compared to RF and CNN.

MLP's models struggle when raw image data is fed and lacks ability of spatial awareness. It is just considering the input as random list of numbers. On the other hand, Random Forest handles tabular data well and can identify patterns even when data is not spatially arranged.

7. CNN – Convolutional Neural Network Description

Convolutional neural networks are a deep learning algorithm good at recognizing patterns in images. It automatically learns the features which are crucial for distinguishing different classes. [6]. It uses three-dimensional data as input and filter and feature map from the convolutional layer to learn local patterns [7].

Through convoluted operations, CNN's can learn class boundaries through detection of edges, color gradients, blobs, fine texture and patterns.

The architecture of the CNN model layer by layer for this task is:

- Input shape: RGB Images of (28,28,3)
- Conv2D layer (32 filters, 3×3 kernel): This applies the 32 filters of size 3 x 3, which is used to detect edges, fine-grained textures and blobs.
- ReLU activation: Adds non-linearity to help learn abstract patterns

- MaxPooling2D (2×2): This shrinks the feature map to only include the most important features
- Conv2D layer (64 filters, 3×3 kernel): It applies 64 filters to learn more abstract patterns
- ReLU activation: Adds more non-linearity to help learn abstract patterns
- MaxPooling2D (2×2): Again, reduces dimensionality and reduces computation and overfitting
- Flatten: Converts 2D maps to 1D vector for it to pass through a dense layer
- Dense layer (128 neurons): learns feature combination for decision making
- Dropout: Disables 30% of neurons randomly to prevent overfitting
- Output: final output layer which shows probability distribution over the 9 classes

For the training model, we have used Adam optimizer which is used for fast convergence. As it is a multi-class classification problem cross entropy as loss function. Training was conducted for **10 epochs** with a **batch size of 64**. For validation **20% of the training data**. SoftMax function is used in the output layer for aligning the target variables. One hot encoding was applied to the target labels

The basic model for CNN attained 62.77% test accuracy, owing to the advantage of interpreting spatial patterns.

Grid Search CV was used to tune the hyperparameters: dropout rate, batch size, optimizer. The best performance was obtained for Batch Size: 64, Dropout Rate: 0.3, Optimizer: Adam and epochs =10. This model attained test accuracy of 77.04%, much higher than other models. On observing the epochs, we could see both training and validation accuracy improved, which shows that the model was learning without overfitting.

8. Comparison of Strengths and Weaknesses

Model	Strengths	Weaknesses
Random Forest	- Able to handle non-linear relations - Easy implementation and interpretation - Robust to overfitting as it takes aggregate values/majority voting	- Ignores spatial context and treats every pixel as a input with no link to each other - Slow training and computationally expensive (115.52 sec)
MLP	- Fast runtime (13.37 sec) primarily due to minimal layers in this task - Can easily extend by adding more layers or neurons	- Poor test accuracy (33.32%), indicating underfitting - Requires flattened images, unable to capture spatial structures

Model	Strengths	Weaknesses
CNN	- High accuracy (77.04%) within reasonable time (22.01s). Captures spatial features via filters	- Slightly complex to build and tune; presence of multiple layers
	-With proper hyper tuning it generalises well	- Interpretability is complex due to its architecture
	- Good for image datasets; able to identify patterns, edges, fine grained textures	- Sensitive to hyper tuning parameters

9. Results Table

The results shown below portrays performance ranking across the models after hyper tuning : Random Forest, Multi-Layer Perceptron and Convolutional Neural networks.

Model	Test Accuracy (best parameters)	Best Parameters
Random Forest	**65.51%**	n_estimators=150, max_features='sqrt', max_depth=None
MLP (Fully Connected)	**34.36%**	units=128, dropout=0.3, batch_size=64, epochs=15
CNN	**77.04%**	dropout=0.3, optimizer='adam', batch_size=64, epochs=10

10. Discussion:

Convolutional Neural Network (CNN) performed the best, attaining highest test accuracy 77.04%. It is well suited for image datasets, as it can capture spatial features through filters. These convolutional filters are beneficial for RGB images of Path MNIST dataset. The convolutional layers capture spatial local patterns like edges, fine textures, blobs etc which improves its class differentiating ability as compared to random forest or MLP. CNN involves a lot of steps and more layers, yet the training time was just 22.11 seconds. This shows that on fitting the model with best parameters, it is computationally efficient. Figure 2 shows good learning behaviour pattern and with every epoch we can see steady improvement across training and validation. It shows that the model is generalising well to unseen data and isn't overfitting.

Multilayer perceptron (MLP) attained only 34.36% test accuracy. Due to flattening it considers all the pixels independently which hampers it from extracting features from

images. It is not suitable for image classification tasks, unless feature engineering is applied to the data extensively. The accuracy obtained is poor even after hypertuning 33.32% . The training time was only 13.37 seconds, making it very cheap but not useful for this task. Figure 1 shows that the learning behaviour stagnates across epochs as it remained flat, and shows underfitting.

Random forest, a traditional model achieved 65.51% test accuracy. This accuracy was good because random forest like MLP accepts only flattened images as input. This shows random forest high performance on tabular like representation of images. The computation cost for this model is relatively high i.e. 115.52 seconds, which is expected due to creation of multiple decision trees.

Training Time Comparison

- **Random Forest**: 115.52 seconds
- **CNN**: 22.11 seconds
- **MLP**: 13.37 seconds

CNN offered the best trade-off between runtime and performance, achieved high accuracy in reasonable time. CNN is well suited for spatial data. Random forest work decent on flattened inputs. MLP's are not suitable for image spatial data, they are fast but lack the spatial intelligence needed for insights and pattern detection.

Model	**Test Accuracy**	**Training Time (seconds)**
Random Forest	65.51%	115.52 sec
MLP (Fully Connected)	33.32%	13.37 sec
CNN	77.86%	22.66 sec

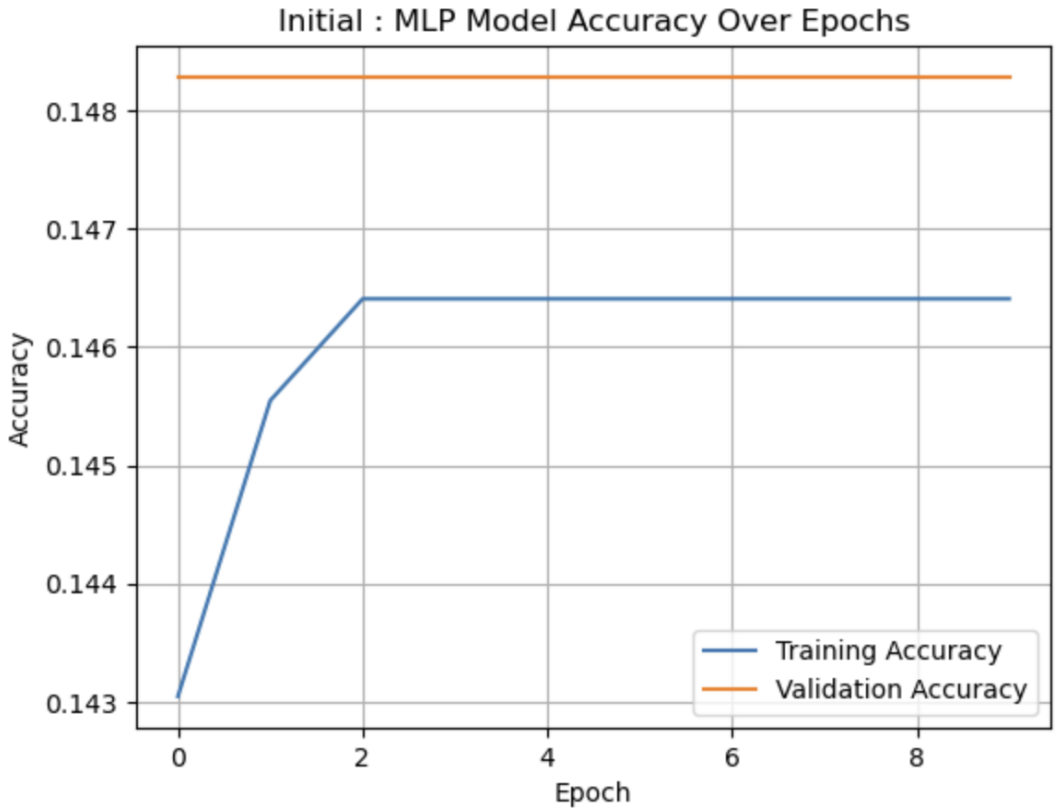

***Figure 3**: MLP model's INITIAL Training curve*

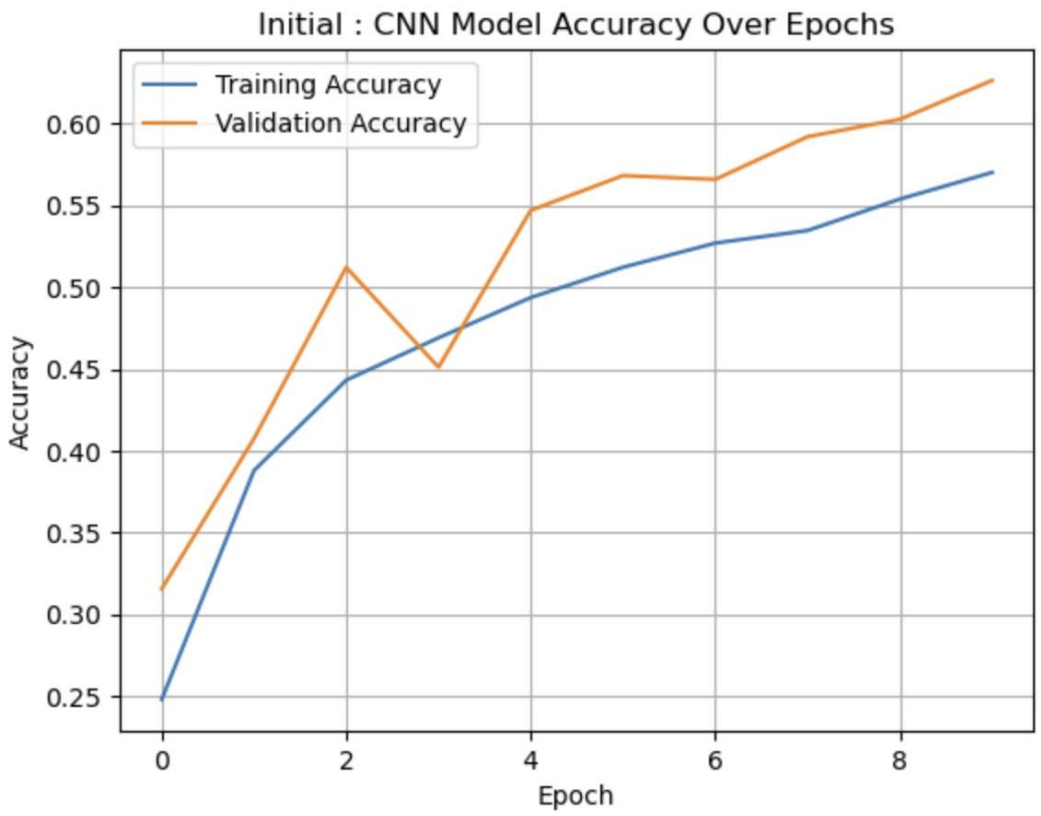

***Figure 4**: CNN model's INITIAL training curve*

The below plots show learning curves for the final models:

The below curve shows the final MLP model learning in both training and validation accuracy over epochs. The higher validation accuracy suggests potential underfitting.

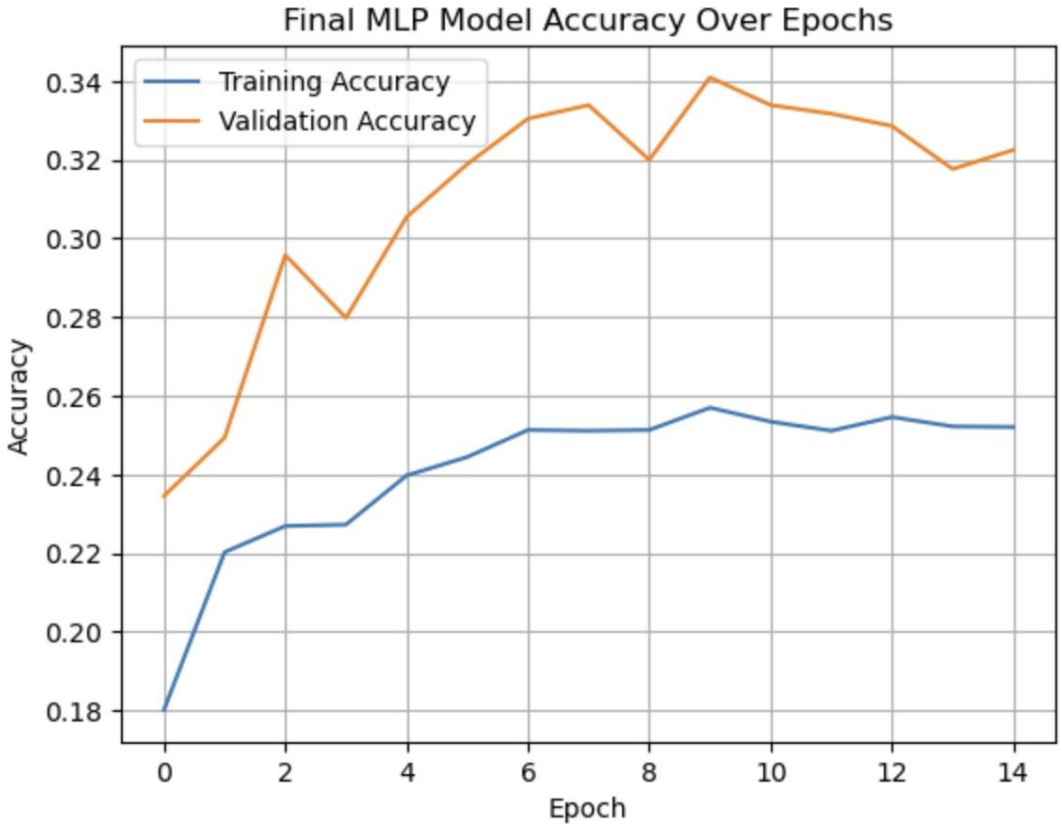

Figure 5: Final Learning curve for MLP

The below plot show the final CNN model's accuracy improves over 10 epochs on both training and validation data.

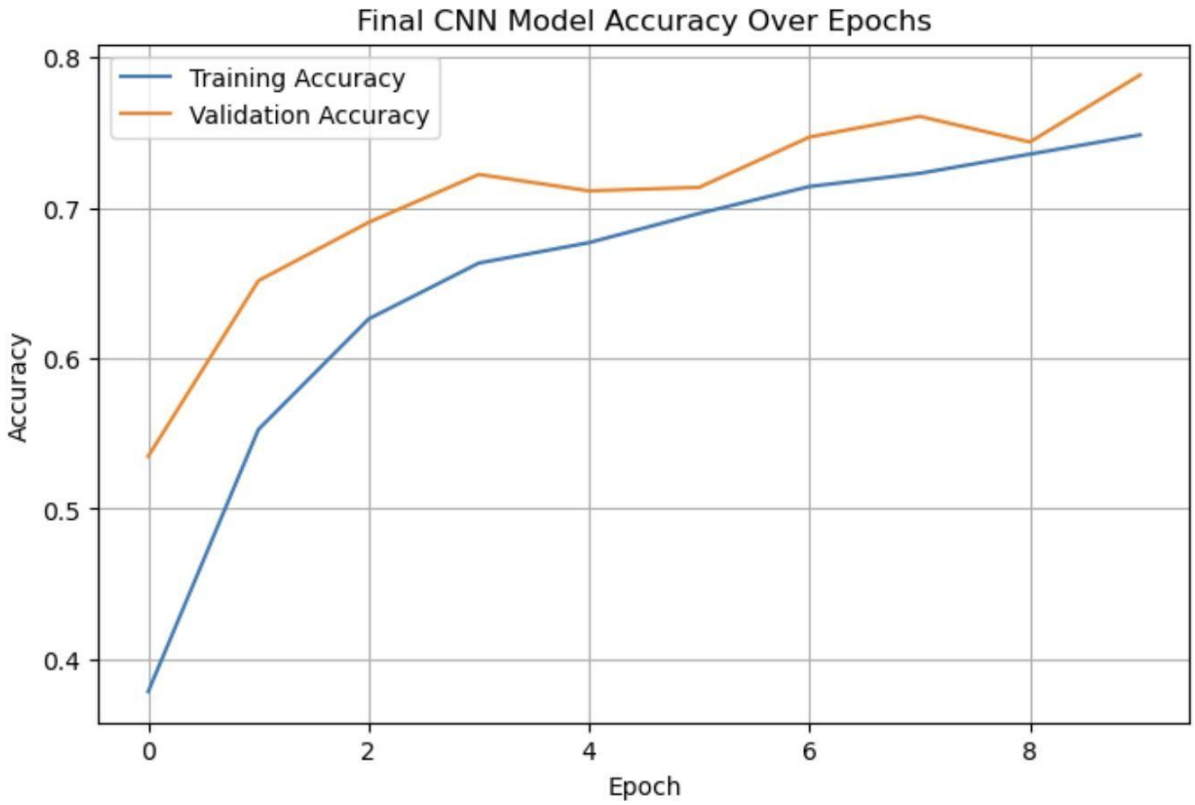

Figure 5: Final Learning curve CNN

11. Limitations of This Study:

- The speed of the models was dependent on our GPU systems. It could vary depending on the system. The runtime of different models could differ leading to different results
- For CNN and MLP models, the initial weight is randomly initialised and data is randomly shufled. Even though a random seed was set, results might vary due to other factors.
- In this study, we did not observe misclassification based on tissue type and if the models performed very poorly on some specific class labels.
- Limited hypertuning was performed due to resource and time constraints. More exhaustive work on hypertuning could have resulted in better results.
- CNN had a structural advantage over MLP and RF, as image data was fed without flattening. Image specific features were not preserved for MLP and RF.
- The resolution of images 28 x 28 enhanced computation, but in real world microscopic images are of higher resolution and more complex.

12. Scope of Future Work:

To enhance the performance of the model, several steps could be taken in future:

- Extensive hyperparameter tuning could be performed including different activation functions, learning rates, additional layers.
- To improve generalisation to unseen data, data augmentation could be applied to the images captured by microscope. This could include zoom, rotation, flipping or stain normalisation.
- Conducting error analysis for class labels to understand if a model performed badly on specific class labels could be used to improve performance of the model. Confusion matrices, model bias could help in evaluating this.
- Instead of replying the whole study on a single train test split, using stratified k fold cross validation could help analyse the performance of model better and reduce evaluation bias.
- Using high resolution images could help the model to discover more patterns and to better understand complex structures, thereby enhancing the model's performance.

REFERENCES

1. Wang, J. (n.d.). *What are KNN's advantages and disadvantages compared to the other popular machine learning models?* Medium. https://medium.com/@jing.wang.ds/what-are-knns-advantages-and-disadvantages-compared-to-the-other-popular-machine-learning-models-b171c48dd817

2. GeeksforGeeks. (n.d.). *Random Forest for image classification using OpenCV.* https://www.geeksforgeeks.org/random-forest-for-image-classification-using-opencv/

3. Analytics Vidhya. (2021, June). *Understanding Random Forest.* https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#Random_Forest_Algorithm_Use_Cases

4. Eincode. (n.d.). *Mastering the Multi-Layer Perceptron (MLP) for image classification.* Medium. https://medium.com/eincode/mastering-the-multi-layer-perceptron-mlp-for-image-classification-a0272baf1e29

5. GeeksforGeeks. (n.d.). *Classification using Sklearn Multi-Layer Perceptron.* https://www.geeksforgeeks.org/classification-using-sklearn-multi-layer-perceptron/

6. Huang, M. (n.d.). *Image recognition with CNNs: Improving accuracy and efficiency.* Medium. https://medium.com/@mitchhuang777/image-recognition-with-cnns-improving-accuracy-and-efficiency-dd347b636e0c

7. IBM. (n.d.). *Convolutional Neural Networks.* https://www.ibm.com/think/topics/convolutional-neural-networks