1. Introduction

In this project, we used three different machine learning models--Support Vector Machines (SVM), Multilayer Perceptrons (MLP), and Convolutional Neural Networks (CNN)--on a medical image classification task using the PathMNIST dataset. Our goal is to find the most suitable model for this type of data. Considering factors such as accuracy, efficiency, and interpretability, comparing machine learning algorithms on medical imaging problems is crucial (Raghu et al., 2019). This research is significant for understanding the different models on challenging medical image classification tasks, which can provide insights for related research and applications.

2. Data

2.1 Data Description and Exploration

The dataset we used in this project is PathMNIST, which is a biomedical image classification dataset containing 107,180 RGB microscope. Each images is 28x28 pixels and distributed across 9 tissue classes. These images come from real medical samples. Our initial data exploration noticed that some classes looked very similar. For example,the images of stomach and intestine had only small differences in color and texture, which made the classification task more difficult. There was not a big class imbalance, but the small visual differences between classes made the classification task more difficult. This means the model has to learn very fine details to achieve better performance.

2.2 Pre-processing

Before training, we did some basic data preprocessing for the images. First of all, we scaled the pixel values to between 0 and 1 for all models. Specifically, for SVM and MLP, we used PCA to reduce the number of features. We kept 99% of the variance and reduced the input size to 719 features, which made the data smaller and easier to work. For the CNN, we trained it on the original images directly instead of using PCA. Because CNNs can automatically learn spatial features. These choices we made were based on what we learned in class and what worked better during testing. Th preprocessing steps helped reduce noise and made it easier for the models to focus on useful patterns.

3. Methods

3.1 Theory

We compared three algorithms: SVM, MLP, and CNN. SVMs find the optimal hyperplane to separate classes in high-dimensional space (Schmidhuber, 2015). MLPs learn hierarchical feature representations through hidden layers (LeCun et al., 2015). CNNs leverage spatial structure by learning local features and aggregating them to make predictions (Khan et al., 2018). The three models have different strengths in learning data representations and decision boundaries. Comparing them helps understand the role of different inductive biases in medical image classification tasks.

3.2 Strengths and Weakness

Model	Strengths	Weakness
SVM	Simple, effective, interpretable	Slow inference, poor scalability to large or high-dim data
MLP	Learns flexible and rich feature representations	Prone to overfitting without enough data or regularization
CNN	Captures spatial features, excels in image tasks	Requires more compute, may overfit on small datasets

3.3 Architecture and hyperparameters

For each model, we designed an architecture and selected relevant hyperparameters for tuning (Table 1). The SVM used an RBF kernel to generate nonlinear decision boundaries, with C and gamma controlling regularization strength and kernel bandwidth, respectively. The MLP used two hidden layers with ReLU activations to learn rich nonlinear feature transformations, balancing model complexity by tuning the number of units in each layer. The CNN used two convolutional layers to learn hierarchical spatial features, with the number of filters and kernel size determining the model capacity and receptive field size.

When we chose hyperparameters, we focused on the ones that really matter. These are the parameters that significantly influence model performance. We picked them based on what we learned in class and also what we saw from practice. For example, in SVM, the C and gamma are very important. C controls how much the model wants to avoid mistakes on the training set. Gamma changes how the decision boundary looks. If gamma is too high, the boundary gets weird.In MLP, the number of hidden units is a big deal. If we use too few, the model can’t learn enough. If we use too many, it might overfit or take too long to train. For CNN, we looked at the number of filters and the kernel size. These control how the model looks at the image. Small kernels are good for local features. Too many filters might be too much and make the model heavy. So by tuning these parameters, we tried to find a good balance. Not too simple, not too complex. Just enough to work well on this task.

Table 1. Model architectures and hyperparameters

Model	Architecture	Hyperparameters Tuned
SVM	RBF kernel	kernel=rbf, C=1, gamma=scale
MLP	2 hidden ReLU layers	units_1=576, units_2=448, learning_rate=0.001
CNN	2 conv (3×3), 1 dense	conv1_filters=96, conv2_filters=64, dense=640, learning_rate=0.0005

We employed stratified 3-fold cross-validation to evaluate SVM's hyperparameters, while using a validation set approach for MLP and CNN. The goal was to strike a balance between underfitting and overfitting while

ensuring training efficiency. Cross-validation helps obtain an unbiased estimate of the model's generalization performance, while the validation set method enables faster evaluation of the model's sensitivity to hyperparameter changes.

4. Results

4.1 Hyperparameter Tuning Results

We did grid search for each model to find the best hyperparameters. Here is what we found:

MLP: The best validation accuracy (51.3%) came from using two hidden layers with 576 and 448 units and a learning rate of 0.001. When the network was too shallow (like 128-64), it couldn't learn enough features. But when it was too deep, the performance did not improve and sometimes became unstable. This shows that the model needs enough capacity, but not too much, to work well.

CNN: The highest accuracy (74.3%) was achieved using smaller kernel sizes (3x3), filters of size 96 and 64, and a dense layer with 640 units. Using larger kernels (like 5x5) or too many filters actually made the model worse. This means that focusing on small local patterns in the image works better, and a model that is too complex might overfit.

SVM: The RBF kernel did much better than the linear one. The best result (59.1%) came from using C = 1 instead of C = 10, which shows that moderate regularization helps the model generalize better.

In short, tuning the model well—choosing the right size, learning rate, and regularization—makes a big difference. A model that's too simple or too complex won't perform well. Finding a good balance is key.

4.2 Hyperparameter Tuning Discussion

The results above clearly show that different hyperparameter settings can greatly affect model performance.

For the MLP, wider hidden layers (like 576 and 448 units) helped the model learn better, but making the model too deep didn't help and sometimes caused problems. A learning rate of 0.001 worked best—it wasn't too slow or too unstable.

For the CNN, small kernel sizes (3x3) were the best. They helped the model capture useful details in the images. Larger kernels (like 5x5) didn't work as well. Using filters of size 96 and 64 was enough, and going bigger didn't help. The learning rate of 0.0005 gave stable training, while larger values made the training unstable.

As for SVM, the RBF kernel was much better than the linear one. This makes sense because the data is not linearly separable. Using a C value of 1 gave

better generalization, while larger values overfit.

Overall, this tuning process shows how important it is to adjust each model carefully to match the dataset and task.

4.3 Final Results and Model Comparison

We compared the three models—SVM, MLP, and CNN—in terms of both test accuracy and training time. Their results are shown in Table 2. Each model had different features depending on its structure and learning mechanism.

The CNN showed the best performance, achieving the highest test accuracy at 84.6%, which was much higher than other models. This is expected because CNN are designed specifically for image-related tasks. However, this advantage comes at a cost. The CNN also spent the longest time to train, with 1810.31 seconds, almost 10 times longer than the MLP. This is mainly because its deep architecture, the large number of parameters and the use of operations like convolution and pooling, which require more computation. So, while CNN provide great accuracy, it is not ideal when we need fast training.

The SVM provided the second high test accuracy at 66.0%, which also provided a relatively fast training time with 237.36 seconds. SVMs are not designed for image data, but after we used PCA to reduce the number of features, the model still worked well. It was faster than CNNs, but its training time was still longer than MLP. This is because kernel methods usually needs to take more time to compute, especially when there are many training samples or features. Overall, SVM found a balance between speed and accuracy, it will be a good option when you want better results than MLP but do not have the resources to train a CNN.

The MLP was the fastest to train, completing in only 137.65 seconds. This is because its relatively simple architecture, which just contain two hidden layers, especially since we used a moderate learning rate and batch size. However, its test accuracy was the lowest, only 58.3%.MLPs are flexible, but they are not very good at learning spatial features in images. This limits their ability to make accurate predictions on visual tasks like PathMNIST.

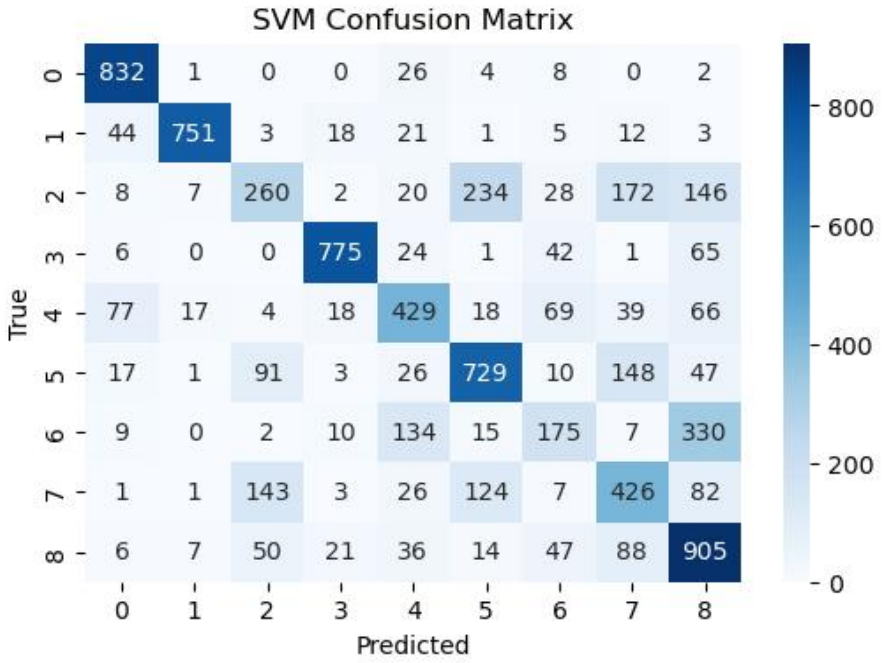
In summary, CNNs offer the best accuracy but require the most time. MLPs are fast but less accurate. SVMs are in the middle area. These differences indicate that the choice of model should depend on the specific needs of the task.

Table 2. Test performance

Model	**Best Hyperparameters**	**Test Accuracy**	**Training Time**
SVM	C = 1, gamma = ‘scale’	0.660	237.36s
MLP	hidden = [576, 448], lr = 0.001	0.583	137.65s
CNN	filters = [96, 64], kernel = 3, dense = 640, lr =0.0005	0.846	1810.31s

To gain further insights into model behavior, we plotted confusion matrices for all three classifiers (Figures 1-3). These visualizations reveal which classes were most frequently confused, helping us understand the strengths and limitations of each model.

Figure 1. Confusion Matrix – SVM

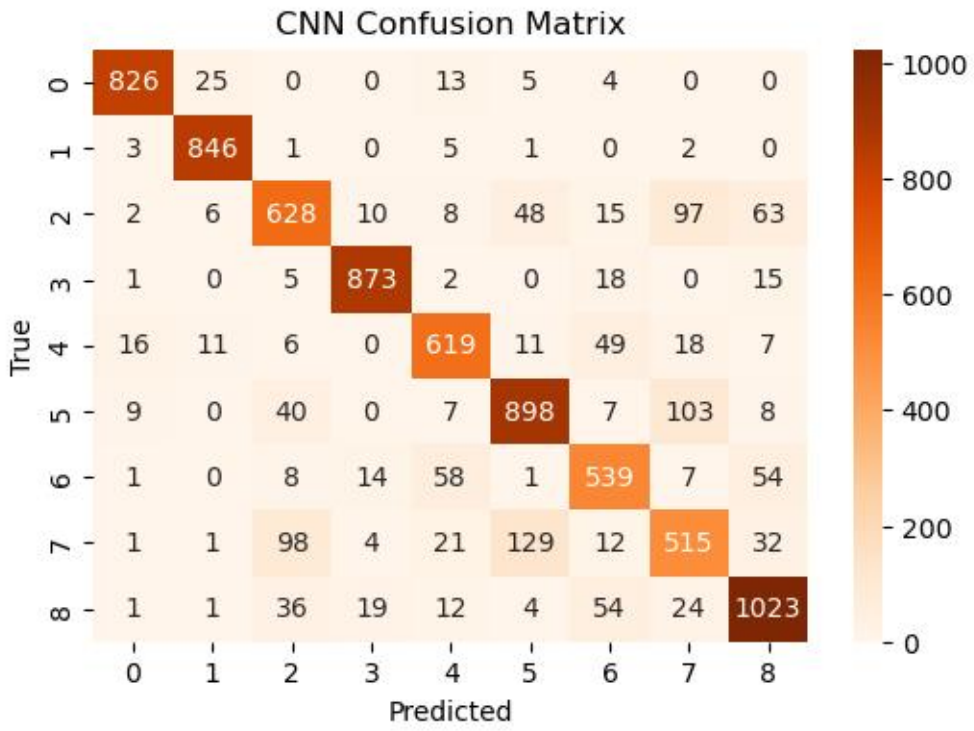


The SVM model performed well on classes 0, 1, and 3, with strong precision and recall. However, it showed poor performance in classes 2, 6, and 7, where it wrongly classified many samples into neighboring classes which look similar. The absence of spatial feature learning in SVM limits its ability to distinguish between visually complex classes.

Figure 2. Confusion Matrix – MLP

The MLP improved upon the SVM in several categories, particularly classes 2, 4, and 6. While still facing challenges in distinguishing visually similar classes, the MLP's nonlinear representation capacity helped reduce some confusion. The results indicate better overall balance but limited spatial awareness.

Figure 3. Confusion Matrix – CNN



The CNN achieved the most concentrated predictions along the diagonal, indicating strong performance across nearly all classes. It significantly reduced confusion in visually similar pairs and effectively handled complex patterns due to its spatial feature extraction, justifying its superior overall accuracy.

The confusion matrices revealed that all models struggled most with distinguishing between the visually similar "stomach" and "intestine" classes. Further analysis could explore techniques for improving performance on these challenging cases, such as data augmentation, transfer learning, or attention mechanisms.

5. Conclusion

This project compared SVM, MLP, and CNN for classifying tissue images in the PathMNIST dataset. CNN had the best accuracy. This shows that CNNs work well for finding important patterns in images. The MLP also did a good job, but the SVM did worse, probably because it cannot handle complex data as well.

There are some limits to our study. The dataset is not very big, and the task is quite simple. In the future, we can try using transfer learning, adding more training data with augmentation, and testing stronger models like ResNets. We could also use tools to explain how the models make their decisions. This

would help build trust.

In conclusion, CNNs are strong tools for medical image tasks, but MLPs also have promise. Choosing the right model and tuning it well can make a big difference in how well it works.

6. Reflection

Student 1 (540282735): I learned a lot from this project. I now understand better how different models work on real problems. I used to think theory was the most important, but now I know that results can change a lot depending on the data and task. Tuning the model's settings was very important. It took time, testing, and knowledge. This helped me get a better feel for which model to use and showed me that machine learning needs many trials and changes.

Student 2 (540114883): I saw how helpful domain knowledge is when choosing a model. CNN did better because it used the image's spatial structure. This shows that using what we know about the data can help us make better models.

References

Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). A Guide to Convolutional Neural Networks for Computer Vision. *Synthesis Lectures on Computer Vision*, *8*(1), 1–207. https://doi.org/10.2200/s00822ed1v01y201712cov015

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

Raghu, M., Zhang, C., Kleinberg, J., & Bengio, S. (2019). Transfusion: Understanding Transfer Learning for Medical Imaging. *Advances in Neural Information Processing Systems*, *32*. https://proceedings.neurips.cc/paper/2019/hash/eb1e78328c46506b46a4ac4a1e378b91-Abstract.html

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*(61), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Tizhoosh, H., & Pantanowitz, L. (2018). Artificial intelligence and digital pathology: Challenges and opportunities. *Journal of Pathology Informatics*, *9*(1), 38. https://doi.org/10.4103/jpi.jpi_53_18