CIS 520 Fall 2022, Project Report

Binary Classification of Histopathological Cancer Images

Group Members:

- Pattana Soranasataporn; Email: pattana@seas.upenn.edu
- Andrew Wang; Email: andreww8@seas.upenn.edu
- Renyi Qu; Email: requ@seas.upenn.edu

Team Name: not-awesome-aardvark
home pod: awesome-aardvark

Abstract

Many machine learning problems in the medical domain are data scarce due to limited access to patient records as well as class imbalance. Inspired by this problem, we aimed to analyze how the performance of different machine learning methods changes with access to varying amounts of data. We compared performance across a variety of models, including Logistic Regression, K-Nearest Neighbors, SVM, XGBoost, CNNs, and transfer learning. For our dataset, we used 6 varying supersets (ranging from 1.82% to 50%) of the full Kaggle Histopathological Cancer image dataset with 220,025 images to simulate both data-scarce and data-abundant settings. Existing work uses pretrained models and the entire dataset, and achieves between 93 and 98 percent accuracy in testing [9, 6]. In comparison, we were able to achieve 85 percent accuracy with our Vision Transformer model using only 50 percent of the dataset, 79 percent accuracy with a CNN using 25 percent of the dataset, and up to 77 percent accuracy with XGBoost when using 25 percent of the dataset. These results indicate that transfer learning achieves the highest performance in data-scarce setting but with relatively high computational cost. We suggest that some classical models can still be of great use depending on the situation.

1 Motivation

Machine learning methods have been increasingly utilized and deployed in the medical field with major achievements in health management systems, medical statistics, and robotics [5]. Numerous researchers have attempted to apply machine learning models to medical image classification for disease diagnosis and achieved excellent experimental results. However, in practice, the advancement of machine learning in medical imaging is constrained by the quality and amount of data available, especially in rare and severe diseases such as cancer [12]. As a comparison, the most popular image dataset named ImageNet [4] consists of over 14 million images, while a typical medical dataset may only consist of a few thousand images. Therefore, we find the problem domain of data scarcity in medical imaging interesting because it is a widespread problem that does not have a trivial or obvious solution.

Inspired by the issue of data scarcity in medical imaging, we attempted to explore the performances of different machine learning methods across 6 different training set sizes on the Kaggle histopathological cancer detection dataset. We would expect that models trained on large datasets may perform well, but many images in such datasets may often encode similar or identical information, leading to smaller improvements in performance. Furthermore, we would like to see if we could achieve similar performance on classical machine learning methods in comparison with state-of-the-art methods while using smaller training set sizes, hoping to acquire a general picture of what specific settings different machine learning methods may become applicable in. Additionally, we also think machine learning is suitable to help solve the problem of histopathological cancer image classification in data constrained settings because machine learning methods may be able to learn features and correlations between the input image and the output label that humans otherwise would not notice.

2 Related Work

There is a history of medical image classification using machine learning methods before 2017, which is well summarized by [8]. Previously, the state-of-the-art models were predominantly variations of SVM and CNN for Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). After advancements in gradient tree boosting and neural networks emerged, [7] was able to achieve 97% accuracy with XGBoost in histopathological cancer detection on a different dataset with a specific focus on breast cancer, while [1] would go even further to achieve 99.1% accuracy on a similar dataset. Furthermore, [2] was able to achieve 97.97% F1-score with Vision Transformer. While many of these approaches achieve high accuracies, there are still not many practical applications of these experimental models because of the data shortage problem mentioned by [12].

Previous work on the Kaggle Histopathological Cancer Detection dataset that we use for this project has been performed by both [9] and [6], achieving 98.6% and 95.5% accuracy respectively. However, both cases adopted a pretrained ResNet50, the entire dataset with 220025 samples and the fastAI framework. Their high accuracies may result from these 3 reasons, which may not necessarily imply that they can be applied in real-life occasions where time is urgent, data is scarce, or computational power is unavailable. Unlike what they did, we would like to explore 1) whether we can achieve similar performances using a much smaller subset of the data (i.e., simulation of a data-scarce environment), 2) whether in such settings, prevalent computer vision models such as CNN can still perform better than traditional machine learning models which are not typically used on computer vision tasks, and 3) how much data is necessary for different models to achieve reliable results.

3 Data Set

We used the Kaggle Histopathological Cancer Detection dataset [3], a modified version of the Patch-Camelyon dataset (PCam) [10] where duplicate samples were removed. Each sample is an image of size $96 \times 96 \times 3$ (width, height, the number of channels (RGB) respectively). However, only the center part of size $32 \times 32 \times 3$ was relevant for labeling these images. Each image is labeled either 1, if the center 32 by 32 pixel did contain cancerous feature, and 0 otherwise. Hence, we only used the center 32 by 32 pixel region as input to our models for this project, resulting in a total feature size of 3,072 (3 color channels x 32 pixels x 32 pixels) per image. Some sample images are included below.

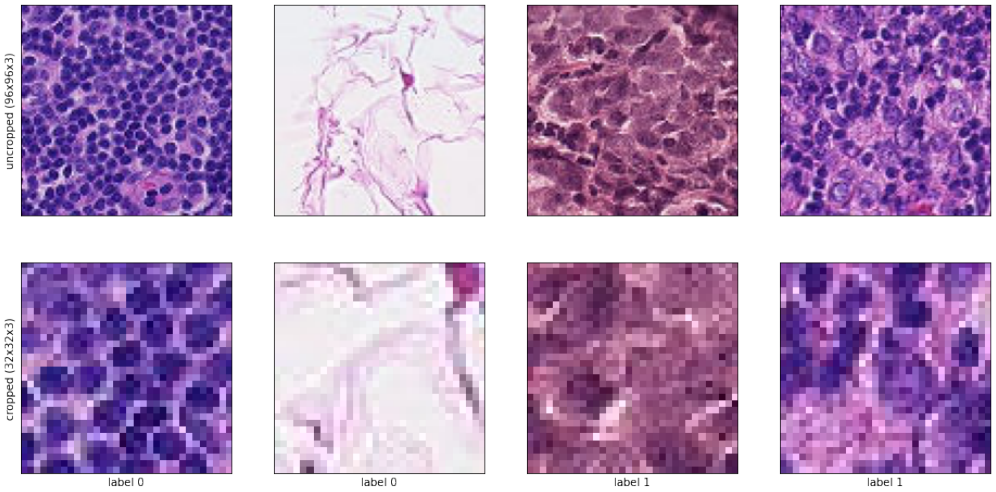

Figure 1: Visualization of uncropped (top) and cropped (bottom) images

The total size of our dataset was 220,025 images; however, in our project, we selected 6 different training set sizes to simulate both data scarce and data rich scenarios. Each successive training set is a superset of the images in the previous training set. The sample sizes are shown in Table 1. In the first 3 steps which

represent the data-scarce setting, we incremented the sample size linearly. In the last 3 steps which represent the data-abundant setting, we incremented the sample size exponentially.

Step 1 (1.82%)	Step 2 (3.18%)	Step 3 (5%)	Step 4 (10%)	Step 5 (25%)	Step 6 (50%)
4,000	7,000	10,488	20,976	52,440	104,880

Table 1: Sample sizes for 6 different training subsets

The original dataset has a class ratio of $0.6 : 0.4$ between label 0 and label 1; however, we created each of the 6 training sets such that the ratio of class 0 to 1 in each training set was 50/50. In addition, we ensured that every subset contained its preceding smaller subset. For example, Step 3 included all images from Step 2, while Step 2 included all images from Step 1. We took this approach because we wanted models trained on each successive "step size" to have access to the same (but greater) amount of information in the previous step size, and because different images of the same class can provide different amounts of information. The test dataset was the same for all 6 partitions, consisting of 20,000 images randomly sampled from the rest of the original dataset excluded from the 6 training subsets. No class balancing was done on the test set (8179 images labeled 0 and 11821 images labeled 1) because we wanted to compare our model's performance to the performance of state of the art methods which didn't use a class balanced test set.

Because we used a wide range of machine learning models and conducted experiments individually, the data pre-processing part was different for each team member as explained in the following subsections.Note that since the data are images and we have full information, no imputation is required for all methods.

For classical models, the image was flattened to $32 * 32 * 3 = 3072$ data points for each image. The data was then normalized by dividing 255 (maximum value for each RGB color).

For CNN, no special preprocessing was done outside of only using the center 32 by 32 pixel region of each image as input to the CNN.

For transfer learning, a major issue is the mismatch in input sizes between pretrained models ($224 \times 224 \times 3$) and image data ($32 \times 32 \times 3$). A $1 : 7$ upscaling was necessary in order to make use of the pretrained models. The original plan was to use the Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) model from [11] to upscale all images in clear resolution without information loss, but this plan came to a halt because it was extremely slow and power-consuming to upscale these images. Due to the limited duration of this project, it was simply not realistic to use ESRGAN. We looked into the related work on the same dataset, but they did not mention how they upscaled their images. Therefore, *torchvision.transforms.Resize()* was used in the end to generate sparse training features without modifying too much of the original information encoded in each image data. No normalization was used to increase the training speed.

4 Problem Formulation

We formulate our task as a binary classification problem under the supervised learning branch: given input image, predict whether it implies histopathological cancer (label 1) or not (label 0). We use the binary cross entropy loss function defined as $\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m}(y_i \log \hat{p}(y_i) + (1-y_i)\log(1-\hat{p}(y_i))$, where y_i is the actual label for the ith example while $\hat{p}(y_i)$ is the estimated probability of "label=y_i" by the model. We use the cross entropy loss function (as opposed to, say the mean square error loss function) for our deep learning (CNN and transfer learning) models because it allows us to train our model so it outputs a probability score of whether an image is label 1 or 0. For our classical models done in scikit-learn, we used the built in loss function. Since the loss functions used may not be the same across our classical models in scikit-learn and our deep learning models in PyTorch, we used accuracy, recall, and F1-score on the test set as shared metrics by which to compare model performance.

5 Methods

We used a total of 7 models divided into 3 branches: classical models, convolutional neural network, and transfer learning. Logistic Regression and K-Nearest Neighbors were selected as our 2 baseline models because they are the simplest classification models, very easy to interpret and implementable with few

hyperparameters to tune. The other 5 methods (SVM, XGBoost, CNN, ResNet, ViT) were chosen because of their great performance in medical image classification as aforementioned in Section 2

5.1 Logistic Regression (Baseline 1)

Logistic Regression was chosen as our baseline model due to its simplicity. We will be using the cross-entropy loss and the flattened data to train the model. Here, different regularization techniques (L1, L2 and elastic net) were applied to see which is most effective.

Packages/Models: *sklearn.linear_model.LogisticRegression*

5.2 K-Nearest Neighbors (Baseline 2)

K-Nearest Neighbors was chosen as our second baseline model because it is the simplest non-parametric classification method. It uses k-nearest existing data points to classify new data via majority vote. For our model, we will use the euclidean distance for the distance calculation.

Packages/Models: *sklearn.neighbors.KNeighborsClassifier*

5.3 Support Vector Machine

We select Support Vector Machine as our first model because it accounts for a higher level of complexity compared to the baseline models. SVM models create a line/ multidimensional plane to classify the data. Some kernel might be applied first for non-linear classification to be effective. Since the image dataset has a very high dimension and we do not expect a simple classification plane to exist, we used an RBF kernel to see how it performs. The gamma value was set to 1/n_features where n_features $= 32 * 32 * 3$.

Packages/Models: *sklearn.svm.svc*

5.4 XGBoost

Gradient Boosting, especially XGBoost, has achieved significant performance in the practical machine learning field and thus, it was chosen as our second model. XGBoost is a type of gradient-boosted decision tree which attempts to classify data by assembling set of weaker learners. It continuously builds decision trees to learn from leftover residuals. We will be trying to tune the number of weak learners, as well as the maximum depth.

Packages/Models: *XGBoost*

5.5 Convolutional Neural Network

Convolutional neural networks use filters that convolve across the input image to learn features of input images. We chose CNNs as one of our models because they are commonly used on image classification tasks. Furthermore, we also did so because we hope to explore how much of an advantage (or disadvantage) that their ability to learn spatially invariant features (due to their use of convolutions) provides under data constrained settings, in comparison to standard machine learning methods such as logistic regression, KNN's, SVMs, and XGBoost.

Packages/Models: *torch.nn.Conv2d, torch.nn.MaxPool2d, torch.nn.ReLU, torch.nn.Linear*

5.6 Residual Network

ResNet is part of the transfer learning branch. ResNet is an enhanced structure of traditional neural networks, in particular CNNs, where the input of the layer block is directly appended to the output at the end of the layer block in order to effectively diminish the vanishing gradient descent issue in dense neural networks. For this project specifically, we used a pretrained ResNet-50 model trained on ImageNet-1k with 1,200,000 images of 1,000 classes. We added 2 fully-connected layers on top of it to generate binary output. We chose ResNet as one of our models so we can compare performance under data constrained settings in our experiments, to performance in data rich settings in previous work that used ResNet with the full dataset [6, 9].

Packages/Models: *torchvision.models.resnet50*

5.7 Vision Transformer

ViT is the other part of the transfer learning branch. It utilizes the transformer architecture invented originally for the NLP field. The vanilla transformer model has a parallel attention-based encoder-decoder structure, but Vision Transformer from Google Research uses only the encoder and adds a multi-layer perceptron head on top of it for classification tasks. The transformer encoder takes a sequence of patches of the input image from top-left to bottom-right as input and encodes these patches. For this project specifically, we used a pretrained ViT model trained on ImageNet-21k with 14,000,000 images of 21,000 classes. We added 2 fully-connected layers on top of it to generate binary output. We also chose to use ViT because we wanted to explore how much of a difference in performance that different pre-trained models can provide when given the same amount of data.

Packages/Models: *google/vit-base-patch16-224 (Hugging Face)*

6 Experiments and Results

6.1 Experiment

For each model, hyperparameter tuning was conducted when the model only has access to the step 1 dataset. We did so because using a large amount of data to determine the optimal hyperparameters for a model trained on a smaller amount of data provides unrealistic access to data that would otherwise not be available under data-constrained settings. Furthermore, for each model we chose not to change its complexity (Ex: number of layers) as we evaluated across different dataset sizes because more complex models should naturally be able to perform better on larger dataset sizes, and using more complex models would not localize the effect in the change in performance to the amount of data available. For each of our step sizes, a new model with the selected hyperparameters at Step 1 was trained on the corresponding superset of data. All models were evaluated on the common test set.

6.1.1 Classical models

For **logistic regression**, all types of regularization result in similar test accuracy on step size 1 of around **0.62**. We ended up choosing L2 regularization due to its popularity. We realized later that the model is under-fitting after seeing low training accuracy. Thus, using no regularization should achieve similar results; however, we didn't retrain the model due to time constraints.

For **K-nearest neighbors**, k values of [1,3,5,10,20] were tested to see which value will generate the best results. Here, we use 5-fold cross validation to chose the k value. After performing the k-fold cross validation, the value of k=3 was chosen.

For **SVM**, we believe the model is underfitting because other models (Ex: ResNet) are able to achieve higher accuracies at step 1 with the same amount of data. This might mean the model could not capture the complexity of the data.

For **XGBoost**, maximum depth of [3,6] and number of weak learners of [250,500,1000] was searched with 5-fold cross-validation. From the results, the validation accuracy does not differ much and can actually be caused by randomness. The best-performing model is when max_depth = 6 and number_of_estimators = 600. However, for evaluation we will be using the max_depth = 3 and number_of_estimators = 250 since the validation accuracy are quite similar and the model are much more simpler. Since our training accuracy was considerably higher than our validation accuracy, the model seems to be over-fitting. Early-stopping at 60 epochs was applied to see the effects after observing the training and validation loss. Note that the early stopping did not increase the model test accuracy in our case but it will make the model less complex.

6.1.2 Convolutional Neural Network

For our CNNs, we used the cross entropy loss function, as well as the stochastic gradient descent optimizer with a learning rate of 0.001 and a momentum of 0.9. We also used a batch size of 4 and always trained over

10 epochs. Our model architecture consists of 2 convolutional layers, each followed by a ReLU activation and max pooling. Afterwards, we use 3 fully connected layers where the first 2 fully connected layers are each followed by a ReLU activation and the third fully connected layer is not followed by a ReLU activation. Our first convolutional layer uses a kernel size of 5 with 6 output channels. Our first max pooling layer uses a kernel size of 2. Our second convolutional layer uses a kernel size of 5 with 16 output channels. Our second max pooling layer uses a kernel size of 2. Our first fully connected layer has an input dimension of 400 and output dimension of 120. Our second fully connected layer has an input dimension of 120 and output dimension of 84. Lastly, our third fully connected layer has an input dimension of 84 and an output dimension of 2 (representing the activations/scores for the 2 classes).

6.1.3 Transfer learning

For transfer learning, all pretrained weights were frozen because updating them at Step 1 did not lead to better accuracy and took significantly longer to train. 2-layer structure was used for the classification head because it achieved far better results than 1-layer structure. With further tuning, the best results were achieved with the first linear layer of an output size of 32 activated by ReLU and the second linear layer with an output size of 2 to match the logits for the 2 labels. Adam was chosen as the optimizer for weight updates, and Early Stopping with a patience of 2 was used as regularization to prevent overfitting.

6.2 Evaluations

For evaluation, we mainly focused on 3 metrics: accuracy, recall, and F1-score. Accuracy directly indicated how well our models distinguished cancerous and non-cancerous images, and recall was included because it is a key metric in the medical field. Recall measures the fraction of predicted positive labels among true positive labels. If a patient has cancer and our model fails to predict positive, the patient will probably miss the best opportunity to cure cancer at the early stage. Hence, recall is of high importance among all metrics. However, a high recall accompanied by low precision would be troublesome in real-life situations. In such cases, our model misclassified many negative examples as positive, which would bring patients unnecessary fears and troubles. Low precision would also mean more cost for the patient and hospital, as we would be classifying most patients as cancerous. Therefore, F1-score was also included to keep track of such behavior. F1-score is sufficient to indicate low precision if it is lower than recall, and it offers a general insight of model performance, so we chose to report F1-score instead of precision. ROC curves for all models were provided at Step 1 and Step 6 to visualize the difference in the overall performance between data-scarce (Step 1) and data-abundant (Step 6) settings.

6.3 Results

In Table 2-4, **bold** means the best result of all models at each step (i.e., best column-wise), while red means the best result of all steps for each model (i.e., best row-wise).

	Step 1 (4,000)	Step 2 (7,000)	Step 3 (10,488)	Step 4 (20,976)	Step 5 (52,440)	Step 6 (104,880)
LogReg (baseline)	0.62	0.63	0.64	0.67	0.69	0.70
KNN (baseline)	0.62	0.63	0.64	0.60	0.62	0.61
SVM	0.71	0.72	0.73	0.73	0.75	0.75
XGBoost	0.74	0.75	0.75	0.76	0.77	0.77
CNN	0.73	0.57	0.75	0.70	0.79	0.78
ResNet	0.77	0.78	0.79	0.80	0.81	0.81
ViT	**0.80**	**0.82**	**0.83**	**0.84**	**0.85**	**0.85**

Table 2: Test Accuracy for all models trained on 6 different subsets

	Step 1 (4,000)	Step 2 (7,000)	Step 3 (10,488)	Step 4 (20,976)	Step 5 (52,440)	Step 6 (104,880)
LogReg (baseline)	0.65	0.65	0.67	0.67	0.68	0.67
KNN (baseline)	0.79	0.77	0.76	0.76	0.61	0.51
SVM	0.80	0.78	0.77	0.76	0.76	0.76
XGBoost	0.76	0.76	0.77	0.77	0.78	0.78
CNN	0.68	**0.92**	0.63	**0.89**	0.74	0.70
ResNet	0.83	0.78	0.75	0.77	0.77	0.79
ViT	**0.83**	0.78	**0.78**	0.81	**0.82**	**0.82**

Table 3: Test Recall for all models trained on 6 different subsets

	Step 1 (4,000)	Step 2 (7,000)	Step 3 (10,488)	Step 4 (20,976)	Step 5 (52,440)	Step 6 (104,880)
LogReg (baseline)	0.58	0.59	0.60	0.62	0.64	0.64
KNN (baseline)	0.63	0.63	0.63	0.61	0.57	0.52
SVM	0.69	0.70	0.70	0.70	0.71	0.72
XGBoost	0.70	0.71	0.71	0.72	0.73	0.74
CNN	0.67	0.64	0.67	0.71	0.74	0.72
ResNet	0.74	**0.81**	0.74	0.76	0.77	0.78
ViT	**0.77**	0.78	**0.79**	**0.80**	**0.81**	**0.82**

Table 4: Test F1-score for all models trained on 6 different subsets

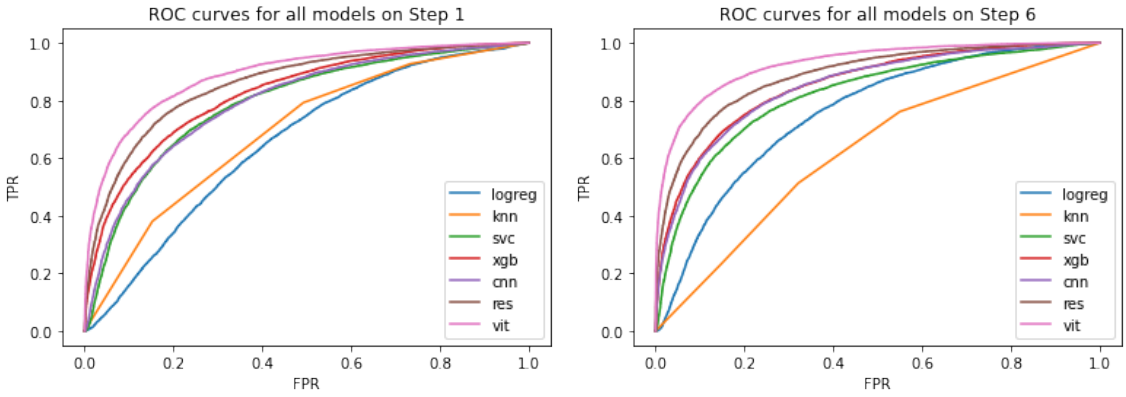

Figure 2: ROC curves of different models

7 Conclusion and Discussion

7.1 Discussion

For model comparison, transfer learning generally performed the best in all 3 metrics regardless of sample size. ViT performed the best with consistent accuracy, recall, and F1-score around or above 0.80 at all steps, while the two baseline models had lower performances with accuracies around 0.70. The performance of XGBoost was consistently better than SVM at each step except for recall with scarce data. While the results for CNN fluctuated, it eventually became more accurate than all the classical ML models as sample size increased. Further analysis of CNN is included in section 7.2.2. In general, regardless of data availability, transfer learning achieved better performance than other methods.

For our sample size comparison, most models reached their best accuracies at Step 5 with 52,440 images while KNN reached its best accuracy at Step 3 with 10,488 images. Of importance, doubling the sample size from 52,440 to 104,880 did not alter the accuracy significantly. This is counterintuitive because we would

expect the accuracy to rise as sample size increases. One potential reason behind this could be that our models are not complex enough to learn more when given greater amounts of data; however, when we tried increasing the complexity of our CNN by adding convolutional layers our test accuracy decreased instead of increasing. This may be due to the additional convolutions and max pooling reducing the dimensionality of the image too much and potentially also blurring the image too much. If we had more time, we would try increasing the number of linear layers instead to validate whether a more complex model can learn from more data in this scenario. Another potential reason behind the lack of a significant change in accuracy when our models are given more data could be random weight initializations causing small fluctuations in test accuracy between our models trained at step 1 and our models trained at step 6. If we had more time, we would repeatedly train our models at step 1, step 2, and so on, and average their test accuracies to validate whether randomness in weight initialization could have caused the fluctuations in performance (as opposed to model complexity causing the differences in accuracy that we saw).

For our recall comparison, 5 out of 7 models achieved the best recall at the earlier stages, with 4 of them achieving the best at Step 1. At first glance this may seem beneficial since we were able to achieve high recall even with a lack of training data. However, the F1-score table may be more meaningful. When the models achieved their best recall, they also had a lower F1-score, implying that they had much lower precision as the price for high recall. When data was scarce, most of the models were inclined to give the input image a label of 1 regardless of what the true label was even though the training subsets were fully class-balanced. One potential reason behind this could be that hyperparameter tuning was done at Step 1, meaning that our model was optimized for Step 1. Even though this optimization was monitored on accuracy, it might have also made the model become more prone to identify underlying patterns of label-1 images for the 4,000 images which may not be representative of the larger dataset, thus leading to an unusually high recall at the very first step. After all, the highest F1-scores were still achieved at data-abundant stages. Depending on the need of medical staff, it may be acceptable to misclassify some 0s as 1s in order to predict the true cancerous cases correctly since data is scarce.

According to the ROC curves, most models achieved better performance with a data-abundant setting except ResNet, XGBoost, and KNN. While KNN's accuracy was consistently lower than other models at each step size, XGBoost and pretrained ResNet may be more desirable methods to use in the data-scarce setting since they had consistent overall performance with little improvement even when data is vastly available. However, pretrained ViT was still the best in the data-scarce setting.

Nevertheless, there was a tradeoff between performance and consumption of time and power. Even in the data-scarce setting at Step 1, the training time for ResNet was roughly 30 seconds per epoch, while that for ViT was roughly 120 seconds per epoch. The inference time on 20,000 images for ResNet was around 45 seconds while that for ViT was around 2 minutes. On top of time, these two models were trained on the Nvidia RTX 3070 GPU, while the other models were mostly trained on CPUs. If hospitals choose not to invest money in GPU setups, then they could take potentially a much longer time to train and make inferences. Also, they would occupy computational resources that may be of better use for other medical tasks. If time and computational resources are problematic for medical staff in certain situations, we would not suggest transfer learning. CNN, XGBoost, and SVM should still be considered in the data-scarce setting.

7.2 Individual Model Analyses

7.2.1 Classical Models

Logistic regression performs poorly, achieving only **0.62** test accuracy with step size 1. Although the accuracy increases with more data, it did not increase significantly. This is probably due to the nature of the data being images that were flattened. The corresponding training accuracy is only about **0.77**. This represents that the model could not quite fit the data efficiently.

KNN performs similarly to logistic regression. However, as we increase the training data, the accuracy did not increase. This may be because with a small value of k, more data may lead to more noise. With more data, a higher k value should be used for the model to reduce noise and perform better.

SVM performed considerably well compared to our baseline models with a minimum test accuracy of **0.71**. The accuracy also increases with number of data points. Note that it is interesting to see that the recall decreases with higher data point even though F1 value increases. This seems to suggest that the model is less prone to predicting one values and are being less biased. The disadvantages of SVM is that the

computation time increases significantly as the number of data point increases and might not be suitable when both data sizes and dimension is high.

XGboost performed very well with limited data with a minimum test accuracy of **0.74**. The accuracy, F1 score and recall all increased with more data. The model also trained considerably faster due to boosting. With small amounts of data, hyperparameter tuning and early-stopping did not yield good results. If we performed hyperparameter tuning on a larger dataset, we may have achieved better results.

7.2.2 Convolutional Neural Network

In comparison to previous work that used pretrained models, our CNNs did not achieve similar accuracy. This is likely because our CNNs are less complex and use less training data. As mentioned previously, if we had more time, we could consider seeing whether performance improves with more linear layers (as opposed to more convolutional layers), as well as repeating the trials for each dataset size and averaging the results to reduce the chance of random weight initialization causing us to see the results we see. Furthermore, we could also look into creating a modified loss function as well as survey a variety of different optimization methods (Ex: using the Adam optimizer instead of SGD) to understand and see their impact on performance and training time.

7.2.3 Transfer Learning

In comparison to previous work that used pretrained ResNet models on the entire dataset and achieved between 95.5 and 98.6 percent test accuracy, our transfer learning methods did not achieve the expected performance. One major reason was the freezing of pretrained weights. The pretrained weights dictated the influence on the final outcome, and they were trained on a completely irrelevant dataset with a significantly larger sample size. On top of that, our histopathological images looked completely different from the pictures in ImageNet. Therefore, freezing these weights has set an upper bound to our maximal performance, especially with our relatively tiny sample size. In addition, our images were upscaled suboptimally to match the high resolution for ImageNet. Therefore, we did not observe significant improvement in the performances of ResNet and ViT even as the training set size increased.

7.3 Conclusion

In this project, we investigated the performance of various machine learning methods in binary classification of histopathological cancer images in different data-scarce and data-abundant settings. We have 3 major findings. First, transfer learning with ResNet and ViT achieved the best overall performances (with a maximum accuracy of 81 and 85 percent, respectively) in all data settings but at the cost of time and computational resources. If there is a shortage of time and computational resources in practice, we would suggest using XGBoost, CNN and SVM as they still achieve 77, 79, and 75 percent accuracy (respectively) at much faster speed with lower computational resources needed. Second, most models were prone to misclassify negative samples as positive despite having high recalls. If this is acceptable in certain situations, then medical staff may not need to worry too much about data scarcity when using machine learning methods. Third, too much data may not necessarily improve the performance of the models in histopathological cancer detection. Best performances for most models were attained at Step 5 with half of the sample size as Step 6. Since data availability and data cost is a problem in the medical field, medical staff could consider using when their model performance plateaus as an indicator for when to stop collecting data. The overall project process taught us how to run a complete machine learning procedure on any arbitrary problem: data preprocessing, model selection and architecture, training, evaluation, and analyses. Most of the options we tried worked very well, but some parts such as ESRGAN for upscaling images did not. It would be interesting to investigate how to reduce the computational cost of transfer learning while maintaining high performance in medical imaging so that they can be deployed in practice and also what is the most suitable sample size considering the tradeoff of both data scarcity and predictive power of cancerous images. In addition, it could be interesting to expand the exact same pipeline used in this project onto other medical classification issues such as brain tumor, lung cancer, etc.

Acknowledgments

We appreciate the guidance from Professor Lyle Ungar and TA Parth Sheth on this project. We are also grateful to all staff members of CIS 520 for their efforts throughout this semester.

References

[1] Qasem Abu Al-Haija and Adeola Adebanjo. Breast cancer diagnosis in histopathological images using resnet-50 convolutional neural network. In *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–7. IEEE, 2020.

[2] Haoyuan Chen, Chen Li, Ge Wang, Xiaoyan Li, Md Mamunur Rahaman, Hongzan Sun, Weiming Hu, Yixin Li, Wanli Liu, Changhao Sun, et al. Gashis-transformer: A multi-scale visual transformer approach for gastric histopathological image detection. *Pattern Recognition*, 130:108827, 2022.

[3] Will Cukierski. Histopathologic cancer detection, 2018.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Pavel Hamet and Johanne Tremblay. Artificial intelligence in medicine. *Metabolism*, 69:S36–S40, 2017.

[6] Jithin James. Histopathologic cancer detection.

[7] Xin Yu Liew, Nazia Hameed, and Jeremie Clos. An investigation of xgboost-based algorithm for breast cancer classification. *Machine Learning with Applications*, 6:100154, 2021.

[8] Eka Miranda, Mediana Aryuni, and E Irwansyah. A survey of medical image classification techniques. In *2016 international conference on information management and technology (ICIMTech)*, pages 56–61. IEEE, 2016.

[9] Towards Data Science. Histopathological cancer detection with deep neural networks, 2019.

[10] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. June 2018.

[11] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

[12] S Kevin Zhou, Hayit Greenspan, Christos Davatzikos, James S Duncan, Bram Van Ginneken, Anant Madabhushi, Jerry L Prince, Daniel Rueckert, and Ronald M Summers. A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises. *Proceedings of the IEEE*, 109(5):820–838, 2021.