Thymic Cancer Classification

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Shruti Sharma

May 2022

The Designated Project Committee Approves the Project Titled

Thymic Cancer Classification

by

Shruti Sharma

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2022

Dr. Marc Stamp Department of Computer Science

ABSTRACT

Thymic Cancer Classification

by Shruti Sharma

Cancer is one of the invasive diseases that humans deal through. Detecting cancers involve various extensive procedures. Thymic cancer is one such rare type of cancer that forms on the outer surface of thymus. Pathologists spend great amount of time to detect cancer and its properties which in turn can be used for patient prognosis. Whole slide imaging (WSI), ever since its first introduction about two decades ago, has been validated for cancer detection and prognosis. The recent approval of US FDA to a WSI system for use in primary surgical pathology diagnosis has opened avenues for wider acceptance and application of this technology in routine practice. Its benefits are innumerable such as ease of access through internet, avoidance of physical storage space, and no risk of deterioration of staining quality or breakage of slides to name a few. Digital scanners, image visualization methods, and abundance of such images has made it possible to provide computer-aided diagnosis using machine-learning techniques. In this review, we introduce the application of Whole Slide Images (WSI) for Thymic Cancer using Machine Learning algorithms to detect cancer type.

ACKNOWLEDGMENTS

I would like to acknowledge and give my thanks to my supervisor Dr Mark Stamp for his continuous inputs and guidance that made this work possible. I would also like to thank Mr Priyam Dhanuka for his inputs and innovative ideas and Dr Anya Arora for her domain guidance.

TABLE OF CONTENTS

CHAPTER

APPENDIX

CHAPTER 1

Introduction

Diagnosis has been performed by a human pathologist observing the stained tissue on a glass slide using a microscope. In the last decade, progress have been made towards digitizing the entire slide with scanners into a digital image (Whole Slide Images, WSI). These WSI images can be analyzed using state-of-the-art machine learning techniques to assist tasks including diagnosis. Since the WSI images have certain unique characteristics, special processing technique are often required. In this review, I aim to describe the use of Thymic whole slide images to determine cancer type. Determining Cancer Type early on can lead to better prognosis for the patient.

1.1 Machine Learning Methods

Usually in case of WSI images, certain pre-processing need to performed prior to applying machine learning algorithms. We break the images in mini patches around 256x256. Following this, we perform feature extraction and classification on each patch based on the problem one is trying to solve. Various local features such as gray level co-occurrence matrix (GLCM) and local binary pattern have been used for WSI Images. Machine Learning Techniques often used in digital pathology are divided into supervised learning and unsupervised learning. The goal of supervised learning is to infer a mapping from the input image to their appropriate labels (in our case cancer type) using the training data. The algorithms for supervised learning includes support vector machines, random forest, logistic regression and CNN. On the other hand the goal of unsupervised learning is to find hidden structure in the input data. The task includes clustering and anomaly detection techniques such ad k-means, auto-encoders and PCA.

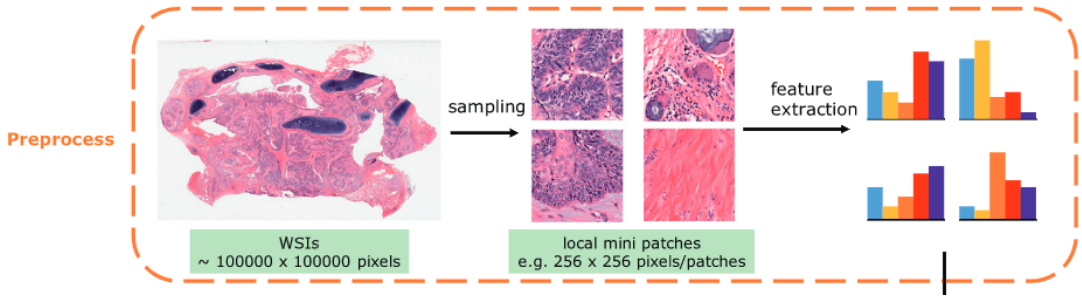


Figure 1: Feature Extraction using patching

1.2 Problems Specific to Whole Slide Images

In this section we describe the unique characteristics of the WSI images and computational methods to treat them. [1]

1.2.1 Very Large Image Size

For general image classification small sized images are often used as input. Images with large size often needs to be resized into smaller size, as an increase in the size of the input image increases the number of parameters to be estimated, the required computational power and memory. In contrast, WSI images contain large number of pixels in the order of tens of billions, which is usually hard to analyze. However, resizing the image to a smaller size, say, 256X256 will lead to the loss of information at the cellular level, resulting in lower learning accuracy. WSI images consist of multiple levels of resolution of the same stained specimen based on different zoom levels of the microscope. The multiple level of magnification contain cellular and structural information. It is necessary to perform magnification as the WSI images are difficult to handle at their original resolution. For the purpose of this project we have taken the zoom level half-way between the available zoom levels, so as to obtain a balance between resolution and size. However, for simplicity we have discarded all other zoom levels, and only used the one we selected. We could have broken each WSI image into patches of 256X256, but we do not have cancer labels for each patch. It is a possibility

that one of the patches may not have cancerous cells and so breaking the WSI image into patches may confused the Machine Learning model. Also, WSI images consists of thousands of patches and false positives are highly likely to appear even if individual patches are accurately classified. I decided to resize the images to 4000*2200 pixels for each image.

1.2.2 Insufficient Labeled Images

Probably the biggest problem in WSI image analysis is only a small number of training data is available. For deep learning techniques a large dataset is required. Although label information at the pixel level or patch level is required, most labels in WSI images are the case-level at most. In our case too, we have cancer type labels for every WSI image, but the patches in the large image do not have label annotations. Label information for each patch can be obtained from the internet using crowd sourcing techniques. However, in our case only pathologists can label the WSI images accurately and labelling at the regional label in a WSI requires a lot of labor. As a result, we decided to take only a single zoom level, in a way that it strikes a balance between high resolution and not requiring patch divisions.

1.2.3 Color variation and artifacts

WSI images are create through a vigorous process: the stained tissue specimen is sliced and placed on a glass slide, stained with chemicals and then scanned using a digital scanner. At each of these steps unwanted effects could be introduced. For eg. tissue folding may occur i.e. when the tissue is being placed on the slides they may be bent or wrinkled; lighting conditions may create artifacts during scanning; blur due to different thickness of tissue may appear; sometimes a marker is used to mark tissue regions. Since these artifacts could adversely affect the interpretation, algorithms have been proposed to pre-process the WSI images. Another important issue is color

variation which occurs because of the different types of staining reagents used by pathologists; and different scanning conditions. To address this issue various methods for color normalization have been proposed. Conversion to gray scale is the easiest way to normalize but it causes loss of information regarding color. On the other hand, color augmentation is a kind of data augmentation that is performed by applying random hue, saturation, brightness and contrast. Color augmentation seems to be suitable for WSI(s) with small color variation, since excessive color augmentation can lead to loss of color information.

I have used a library called HistoQC[11] to get rid of artifacts such as tissue folding, image blur and marker stains. It also detects cohort-level outliers (darker or lighter samples as compared to others). The library uses color histogram, edge detection, pen detection and performs color normalization to provide artefact free images. to remove marker stains, extract tissue pixels and resize image to a desired size. It also provides a list of features for each image, which can be used for outlier detection.

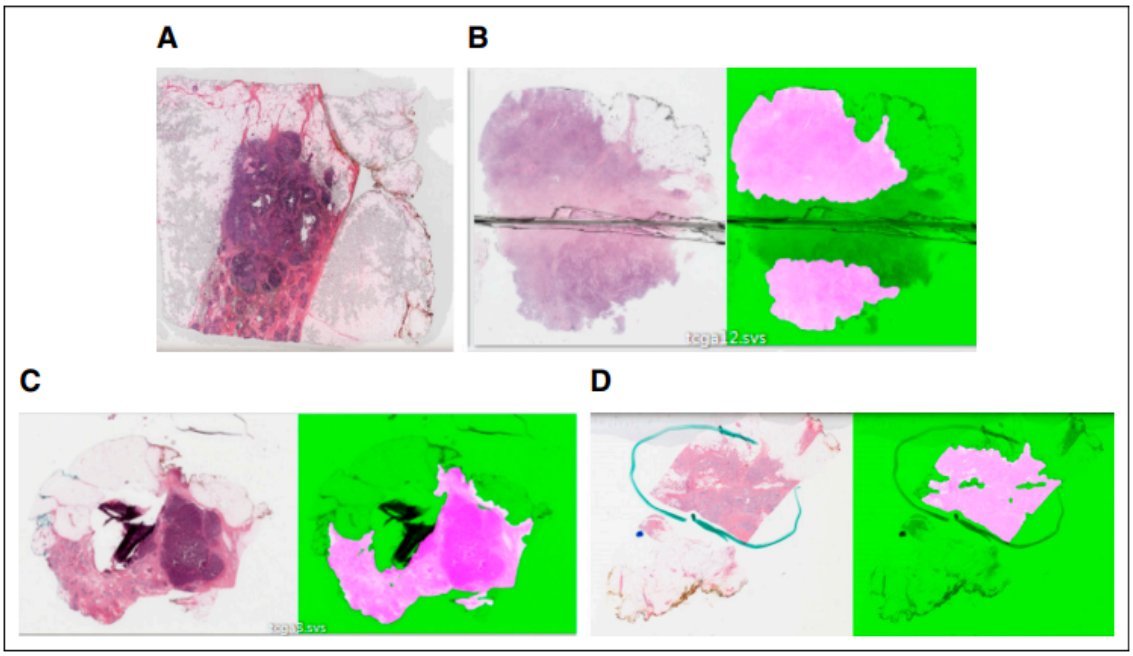


Figure 2: (A) Have a lot of air bubble requiring removal from the training set (B) Glass Slide is cracked causing bluriness near the crack (C) Folded Tissue (D) pen markings correctly identified and avoided

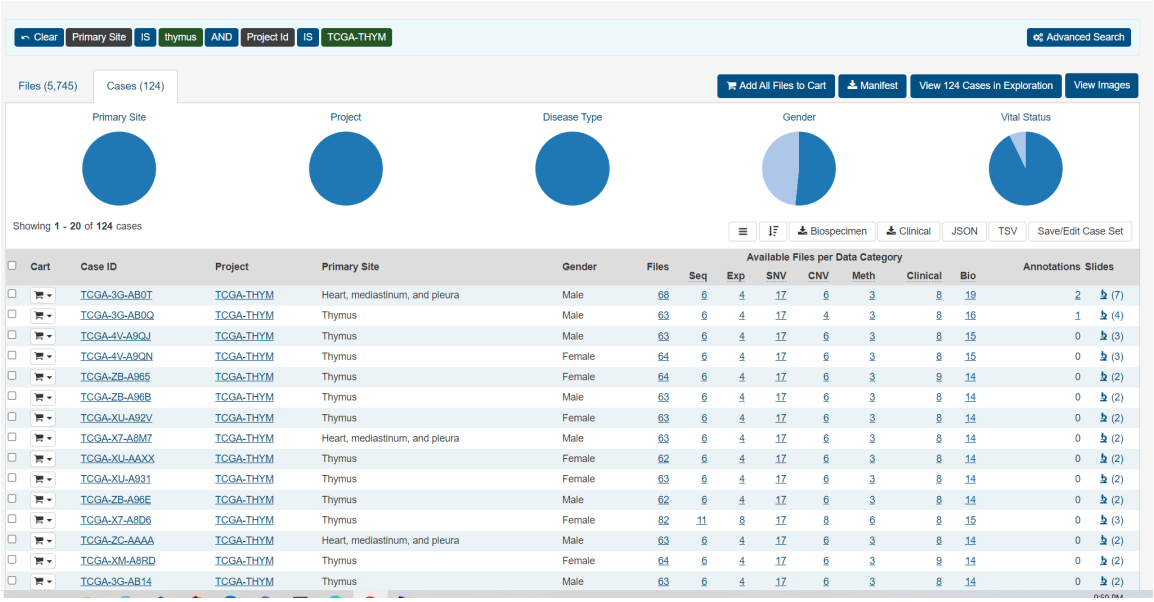
CHAPTER 2

Implementation

This chapter explains the implementation details for the project. It explains the steps taken towards data collection, data cleaning/analysis and data modelling. I attempt to explain the steps I have taken and the rationale behind it.

2.1 Data Collection

The data has been collected from this portal.[2] There were a total of 124 patients with 320 WSI images (some patients had more than 1 WSI image reference)



Cart	Case ID	Project	Primary Site	Gender	Files	Available Files per Data Category							Annotations	Slides
						Seq	Exp	SNV	CNV	Meth	Clinical	Bio		
	TCGA-3G-AB0T	TCGA-THYM	Heart, mediastinum, and pleura	Male	68	6	4	17	6	3	8	19	2	(7)
	TCGA-3G-AB0Q	TCGA-THYM	Thymus	Male	63	6	4	17	4	3	8	16	1	(4)
	TCGA-4V-A9QJ	TCGA-THYM	Thymus	Male	63	6	4	17	6	3	8	15	0	(3)
	TCGA-4V-A9QN	TCGA-THYM	Thymus	Female	64	6	4	17	6	3	8	15	0	(3)
	TCGA-ZB-A965	TCGA-THYM	Thymus	Female	64	6	4	17	6	3	9	14	0	(2)
	TCGA-ZB-A96B	TCGA-THYM	Thymus	Male	63	6	4	17	6	3	8	14	0	(2)
	TCGA-XU-A92V	TCGA-THYM	Thymus	Female	63	6	4	17	6	3	8	14	0	(2)
	TCGA-X7-A8M7	TCGA-THYM	Heart, mediastinum, and pleura	Male	63	6	4	17	6	3	8	14	0	(2)
	TCGA-XU-AAXX	TCGA-THYM	Thymus	Female	62	6	4	17	6	3	8	14	0	(2)
	TCGA-XU-A931	TCGA-THYM	Thymus	Female	63	6	4	17	6	3	8	14	0	(2)
	TCGA-ZB-A96E	TCGA-THYM	Thymus	Male	62	6	4	17	6	3	8	14	0	(2)
	TCGA-X7-A8D6	TCGA-THYM	Thymus	Female	82	11	8	17	8	6	8	15	0	(3)
	TCGA-ZC-AAAA	TCGA-THYM	Heart, mediastinum, and pleura	Male	63	6	4	17	6	3	8	14	0	(2)
	TCGA-XM-A8RD	TCGA-THYM	Thymus	Female	64	6	4	17	6	3	9	14	0	(2)
	TCGA-3G-AB14	TCGA-THYM	Thymus	Male	63	6	4	17	6	3	8	14	0	(2)

Figure 3: Encrypted virus before and after decryption.

The total size of the data was close to 298 Gigabytes. The data collection took a period of 3 days over the network. Each sample had an image and a metadata file attached to it.

2.2 Process

Once the data is collected and available, we use the tool provided by HistoQC [3] to do the following :-

2.2.1 Data Cleaning

We load all the SVS files onto the tool and analyze image attributes. The tool, through a series of algorithms such as color histogram, pen marking and edge detection get rid of blurriness, marker stains, folding artefacts and also provide image features that can help us filter outliers. Given are some outliers I processed as a part of this effort:

- Given below is a chart displaying a line for every image and a column for every feature related to the image. The highlighted boxes show the outliers that I wish to address in the next steps.

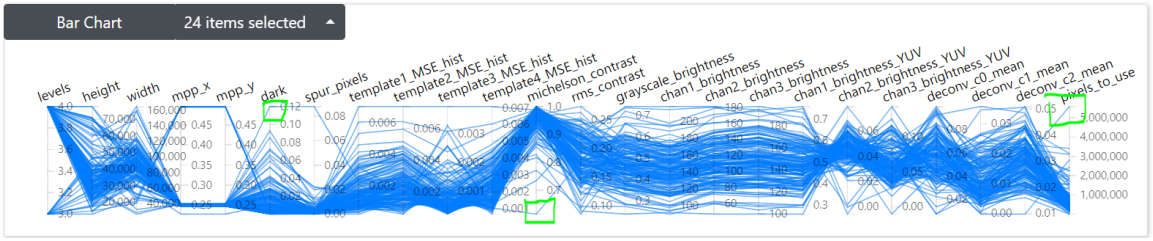


Figure 4: This shows that there are images which are either too dark, have low contrast or have less useful pixels. They have been addressed below.

- Get rid of very dark image as shown below

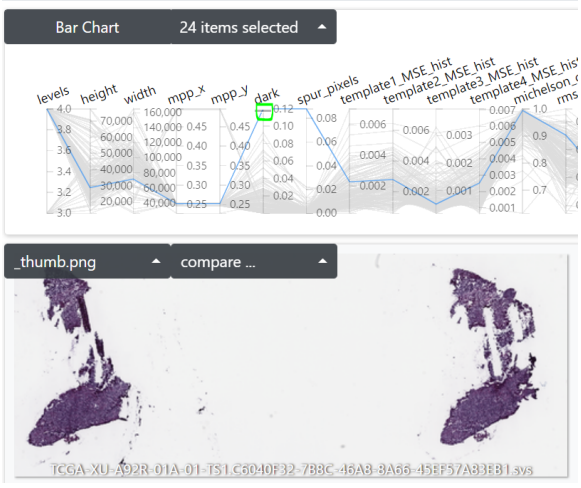


Figure 5: One odd data point has a very high dark index.

- Get rid of images with very low contrast. This can also mean that the stained tissue was folded during scanning or there were lighting problem with scanning conditions. Given below is an image with low contrast.

Figure 6: This image has low contrast and may not be a good image for the train set.

- Remove images with very less number of useful pixels.This is obtained by sorting the table on pixel_to_use column and taking the ones with the least pixels. I removed images having total pixels less than 5% as compared to total pixels for all images in the dataset.

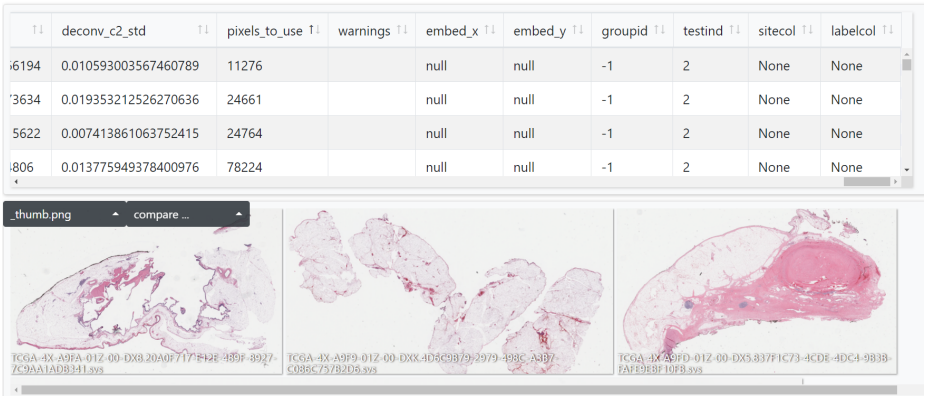
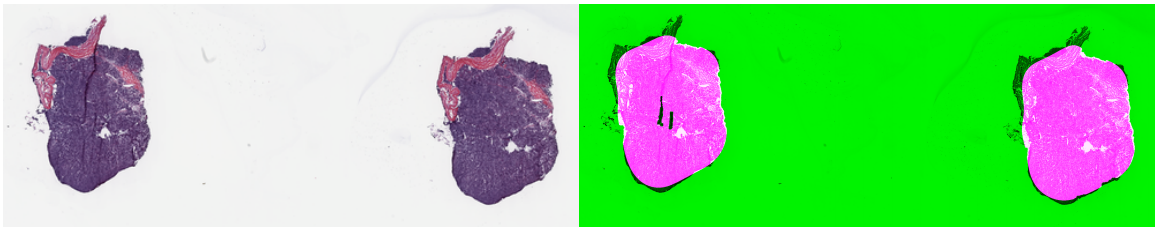


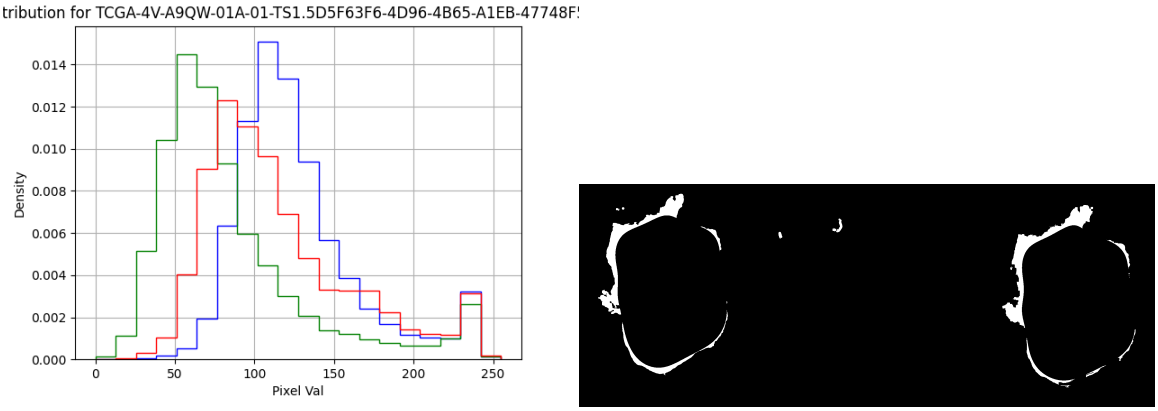
Figure 7: These images have very low number of pixels.

- Remove images which are very large in size. We remove any image which has a width larger than 4000 pixels and height larger than 2210 pixels. The number of images dropped is just 5% of the total dataset. This reduces the storage requirements and ensures faster processing.

  At the end of the data cleaning process, HistoQC provides me with a mask that shows the region of interest over the image. It also provides me the color histogram and the blurriness in the image. Eg. of a mask for one such image is below:



(a) The image on the left is the original WSI image. The image on the right is a fused mask where the brighter pink shows the useful pixels and the green shows noise.



(b) The left image shows the color histogram for the actual slide. The right images shows the images identified as blurry in the original image.

We dropped a total of 169 images as a part of data cleaning (artefact removal etc.). We were left with a total of 151 images.

2.2.2 Data Augmentation

Once data cleaning is complete, we have a mask for each image specifying the useful pixels. The pixels which contributed to artefacts or did not hold and interesting information are discarded in the mask. The next step is to

- Reshape all images to the same size. Since every image is different we need to reshape all the images to have the same dimensions. We do this using Data Augmentation. We augment pixels to images which have a smaller dimension i.e.height or width than the largest image in our dataset. This is how an augmented image looks like:

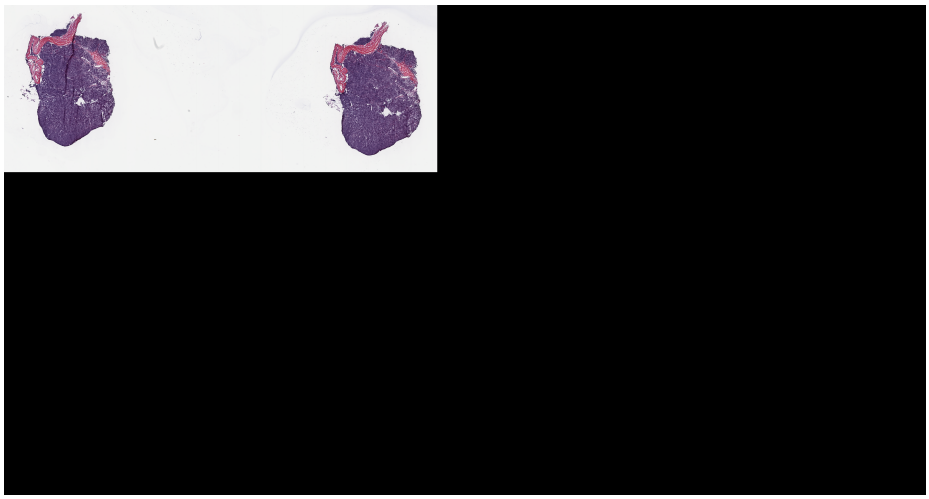


Figure 9: The images is augmented with black pixels to make it an even size of 4000*2200.

- Once we complete augmenting the image, we need to mask the unwanted pixels that we obtained from our data cleaning process. We do that using Python. The result is as below:

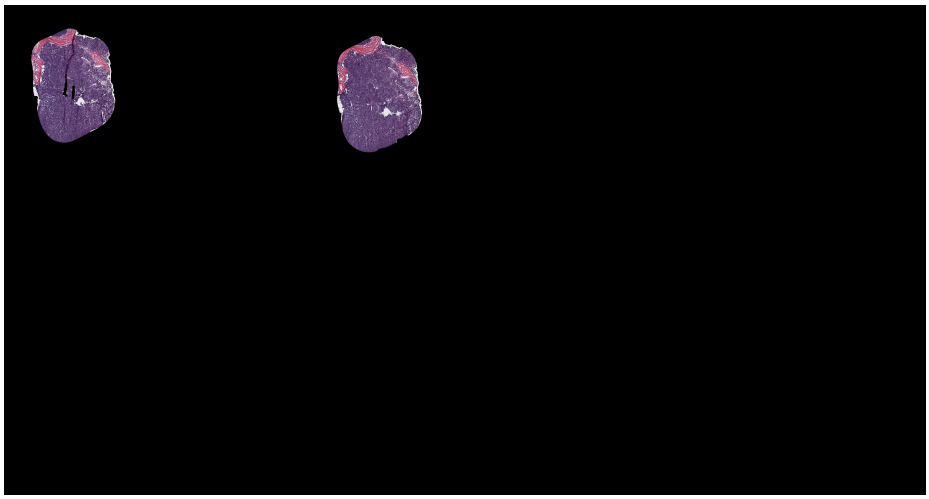


Figure 10: The image is run through a kernel mask, suppressing pixels which are not important.

2.2.3 Data Modelling

Once data augmentation is complete, we have a dataset which we can model to predict cancer type. We want to perform training in mini-batches (since the entire

dataset might be too large to train in a single batch). Steps towards modelling:

- The first step we perform is split the dataset into test and train set. We choose to take 12% of the dataset for cross validation (test set) and the rest for training. We ensure that we take a balanced test set on the outcome variable. Hence, we take 3 images for each output class (cancer type). Below is the code for the same

```
[48]: from sklearn.utils import shuffle
typeab = final_df[final_df['cancer_type'] == 'type AB'].sample(n=3,␣
 ↪random_state =int(random.random()*100) )
thymic = final_df[final_df['cancer_type'] == 'Thymic carcinoma'].sample(n=3,␣
 ↪random_state =int(random.random()*100) )
typea = final_df[final_df['cancer_type'] == 'type A'].sample(n=3, random_state␣
 ↪=int(random.random()*100) )
```



```
typeb3 = final_df[final_df['cancer_type'] == 'type B3'].sample(n=3,␣
 ↪random_state =int(random.random()*100) )
typeb2 = final_df[final_df['cancer_type'] == 'type B2'].sample(n=3,␣
 ↪random_state =int(random.random()*100) )
typeb1 = final_df[final_df['cancer_type'] == 'type B1'].sample(n=3,␣
 ↪random_state =int(random.random()*100) )
final_df_test = pd.concat([typeab, typea, typeb1, typeb2, typeb3, thymic])
final_df = shuffle(final_df[~final_df['#dataset:filename'].
 ↪isin(final_df_test['#dataset:filename'])])
x_test = final_df_test['#dataset:filename']
y_test = final_df_test['cancer_type']
```

Figure 11: Split the dataset using the stratified sampling approach on the cancer type. We take a total of 18 images in dataset, 3 from each class.

- We normalize the data by dividing every pixel by 255 (similar to MaxScaler). This maps the domain of all our features to be between 0 and 1.
- Now, we have a total of 133 images remaining in our training dataset (18 images out of 151 were removed to be a part of the test set). We perform a stratified sampling to select a subset of images for our first epoch. We select 10 images

for each cancer type i.e (a total of 10*6 = 60 images) for as our first batch.

- We run a total of 15 epochs and choose Stochastic Gradient Descent Classifier as our model for learning. In every epoch, we randomly select 10 training image from each class (stratified sampling) and perform batch training (partial fit) of the data.

```
#param_grid={'C':[0.1],'gamma':[0.01],'kernel':['poly']}
#svc=svm.SVC(kernel='poly',gamma=0.01, C=0.1)
svc = SGDClassifier(random_state=123)

#final_df,x_test,y_train,y_test=train_test_split(final_df_sampled,final_df['cance
 ↪20,random_state=123,stratify=final_df['cancer_type'])
#model=GridSearchCV(svc,param_grid, verbose=True)
for i in range(0, 15):
    print("Running for iteration i=%d"%i)
    random.seed(time.time())
    typeab = final_df[final_df['cancer_type'] == 'type AB'].sample(n=10,␣
 ↪random_state =int(random.random()*100) )
    thymic = final_df[final_df['cancer_type'] == 'Thymic carcinoma'].
 ↪sample(n=10, random_state =int(random.random()*100) )
    typea = final_df[final_df['cancer_type'] == 'type A'].sample(n=10,␣
 ↪random_state =int(random.random()*100) )
    typeb3 = final_df[final_df['cancer_type'] == 'type B3'].sample(n=10,␣
 ↪random_state =int(random.random()*100) )
    typeb2 = final_df[final_df['cancer_type'] == 'type B2'].sample(n=10,␣
 ↪random_state =int(random.random()*100) )
    typeb1 = final_df[final_df['cancer_type'] == 'type B1'].sample(n=10,␣
 ↪random_state =int(random.random()*100) )

    final_df_sampled = shuffle(pd.concat([typeab, typea, typeb1, typeb2,␣
 ↪typeb3, thymic]))
    x_train = final_df_sampled['#dataset:filename']
    y_train = final_df_sampled['cancer_type']
```

Figure 12: Running 15 epochs.

```
#x_train,x_test_n,y_train,y_test_n=train_test_split(final_df_sampled['#dataset;
filename'],final_df_sampled['cancer_type'],test_size=0,random_state=77,stratify=f
  le = preprocessing.LabelEncoder()
  le.fit(y_train)
  y_train_images = le.transform(y_train)
  y_test_images = le.transform(y_test)
  x_train_images = read_images(x_train)
  x_test_images = read_images(x_test)


  svc.partial_fit(x_train_images,y_train_images, classes=[0,1,2,3,4,5])
  print('The Model is trained well with the given images')
  joblib.dump(svc, "model.pkl")

  y_pred=svc.predict(x_test_images)
  print("The predicted Data is :")
  print(y_pred)
  print("The actual data is:")
  print(y_test_images)
  #print(np.array(y_test_images))
  print(f"The model is {accuracy_score(y_pred,y_test_images)*100}% accurate")
```

Figure 13: Testing at the end of every epoch, to check the improvement in accuracy.

- We run a total of 15 epochs and choose Stochastic Gradient Descent Classifier as our model for learning. In every epoch, we randomly select 10 training image from each class (stratified sampling) and perform batch training (partial fit) of the data.

2.3 Results

Below is a screenshot of the test results for every epoch. The output stream is of the form:

Running for iteration i=<epoch number>

The model is trained well with the given images.

The predicted data is:

<The data labels which were predicted by our model>

The actual data is:

<The target label for the test set>

The Model is <accuracy in percentage>% accurate

```
Running for iteration i=0
The Model is trained well with the given images
The predicted Data is :
[2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
The model is 12.5% accurate
Running for iteration i=1
The Model is trained well with the given images
The predicted Data is :
[2 3 3 5 2 1 0 5 0 1 3 2 0 4 4 3]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
The model is 50.0% accurate
Running for iteration i=2
The Model is trained well with the given images
The predicted Data is :
[2 1 3 1 1 0 1 5 1 5 2 2 0 1 4 5]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
```

27

```
The model is 75.0% accurate
Running for iteration i=3
The Model is trained well with the given images
The predicted Data is :
[2 3 3 1 1 0 1 5 3 5 4 2 0 1 5 5]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
The model is 87.5% accurate
Running for iteration i=4
The Model is trained well with the given images
The predicted Data is :
[2 3 3 1 1 0 5 5 3 5 4 2 0 4 4 5]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
The model is 93.75% accurate
Running for iteration i=5
The Model is trained well with the given images
The predicted Data is :
[2 3 3 1 1 0 1 3 3 5 4 2 0 4 4 3]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
The model is 87.5% accurate
Running for iteration i=6
The Model is trained well with the given images
The predicted Data is :
[2 3 3 5 0 0 1 5 3 5 4 2 0 4 4 3]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
The model is 81.25% accurate
Running for iteration i=7
The Model is trained well with the given images
The predicted Data is :
[5 3 3 1 1 0 1 5 3 5 5 2 0 4 4 5]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
The model is 87.5% accurate
Running for iteration i=8
The Model is trained well with the given images
The predicted Data is :
[1 3 3 1 1 0 1 1 1 5 4 1 0 4 4 5]
The actual data is:
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
The model is 75.0% accurate
Running for iteration i=9
The Model is trained well with the given images
The predicted Data is :
[2 3 3 1 1 0 1 5 3 5 4 2 0 4 4 5]
```

Figure 14: Testing at the end of every epoch, to check the improvement in accuracy.

Below is a graph of accuracy score over epochs (on the same test set). We notice

that the maximum accuracy is at the 4th epoch and i eventually stabilizes around 87%

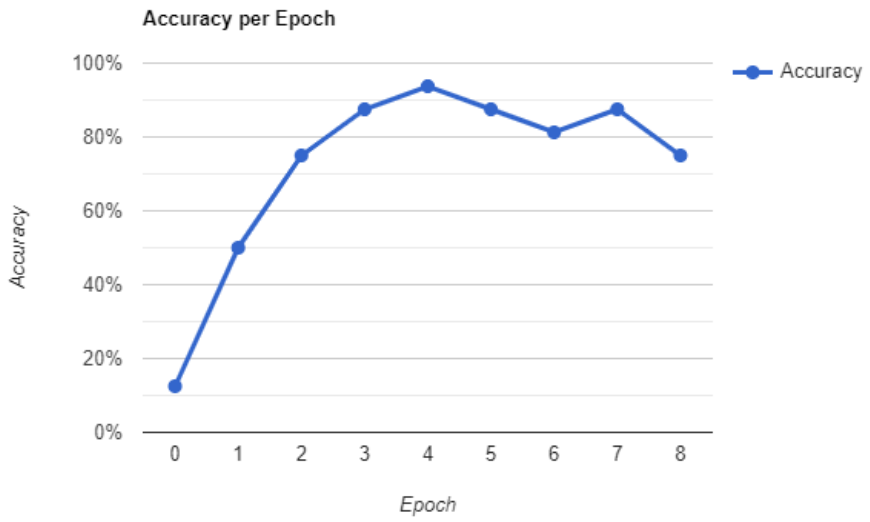


Figure 15: Accuracy vs Epochs

2.4 Future Work

Despite the model having high accuracy, we have not tested for the possibility of over fitting. Our test data is too small and it is possible that the results are biased. The next step will be to use different models and learning techniques with a larger dataset to test the efficacy of the model.[4]

We have discarded over 169 images out of 320 because of lack of per-pixel label information. We only have labels for cancer type for the whole image and not for a pixel (or for a region/patch). This forced us to perform a very aggressive data cleaning procedure. For example, if an image had a few pen markings, with some parts of the image having artefacts, we had to drop the whole image. We took one image as a single sample, instead of taking a subset of patches/pixels in the image. This was due to lack of labelling information for each patch/pixel.[5]

We should also try running different ML algorithms with the current dataset.

However, with such a small sample size it may be difficult to get a clear picture about the model. Hence, the future work should focus on data collection and data annotation so as to enable the use of larger number of ML and statistical procedures.

LIST OF REFERENCES

[1] D. Komura and S. Ishikawa, ''Machine learning methods for histopathological image analysis,'' *Computational and Structural Biotechnology Journal*, feb 2018.

[2] ''Gdc data portal.'' [Online]. Available: https://portal.gdc.cancer.gov/repository

[3] A. Janowczyk, R. Zuo, H. Gilmore, M. Feldman, and A. Madabhushi, ''Histoqc: An open-source quality control tool for digital pathology slides,'' *JCO Clinical Cancer Informatics*, 04 2019.

[4] M. Lu, D. Williamson, T. Chen, R. Chen, M. Barbieri, and F. Mahmood, ''Data-efficient and weakly supervised computational pathology on whole-slide images,'' *Nature Biomedical Engineering*, 06 2021.

[5] N. Dimitriou, O. Arandjelovic, and P. Caie, ''Deep learning for whole slide image analysis: An overview,'' *Frontiers in Medicine*, 11 2019.

APPENDIX

Important Links

A.1 PY WSI Python Package

https://github.com/ysbecca/py-wsi

A.2 HistoQC Github Package

https://github.com/choosehappy/HistoQC

A.3 Cancer Dataset Portal

https://tinyurl.com/2p9hf5ads