Cairo University
Faculty of Computers and Artificial intelligence Department of Computer Sciences

Malignant Lymphoma Classification

Supervised by

Dr. Amin Allam

Dr. Heba Zaki

TA. Nora Abd El Hamed

Implemented by

20188006	*Akram Gamal Mostafa*
20188067	*Mohamed Ahmed Sayed*
20188068	*Mohamed Raafat*
20188047	*Marwan Ahmed Shbeeb*
20188060	*Yousef Mohamed Said*

Graduation Project

Academic Year 2021-2022

Project's Documentation

Table of Content

List of figures

List of tables

List of abbreviations

- MCL: Mantle Cell Lymphoma
- FL: Follicular Lymphoma
- CLL: Chronic Lymphocytic Lymphoma
- CNN: Convolutional Neural Networks
- KNN: K-Nearest Neighbors
- SVM: Support Vector Machine

Abstract

Being able to determine the specific subtype of cancer is the first step towards treatment, as cancer treatment relies completely on early diagnosis of the cancer type to help control it in its early stages. Blood cancer types are the most spreading ones. Previous studies have provided evidence for automatic cancer tissue analysis by using deep learning strategies that retrieve and organize discriminating insights from the images automatically. Therefore, in this study an innovative and empowered deep learning.

framework is proposed to classify three types of lymphoma as Follicular Lymphoma (FL), Chronic Lymphocytic Lymphoma (CLL) and Mantle Cell Lymphoma (MCL). Which are the most common three types of non-Hodgkin Lymphoma cancer. In this research, we developed and advanced CNN model that efficiently predicts the type of non-Hodgkin Lymphoma with an accuracy of 98%.

A publicly available dataset from National Institute of Ageing (NIA) is used in this study. This study performs the histogram normalization on all the images to enhance the performance of model. The data augmentation has been carried out on the dataset so that overfitting can be avoided, we also segmented the images to get higher accuracy.

Chapter 1: Introduction

Introduction

Lymphoma is a blood disease (cancer) that develops in the immune cells, which are the cells that mainly defend our bodies from infections and viruses. Thus, a vulnerable body with no defense system requires attention from researchers and machine learning specialists to help eliminate it.

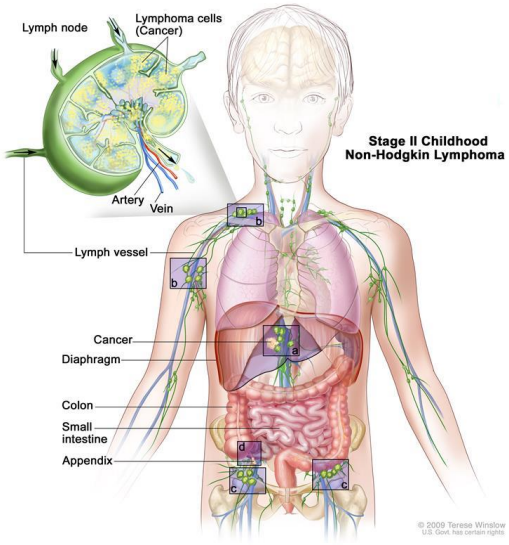


Figure 1 (Lymphocytes)

Because lymphocytes have physiologic immune functions that vary both by lineage and by stage of differentiation, the classification of lymphomas arising from these normal lymphoid populations is complex.

Lymphoma is mainly regarded as: Hodgkin Lymphoma and non-Hodgkin Lymphoma. Having only two developing sorts or Lymphoma by high rate gives us the opportunity to focus and find a suitable solution for both of them. Lymphomas types are classified based on the normal counterpart, or cell of origin, from which they arise.

We decided to focus on the most common three types of non-Hodgkin lymphoma types among cancer patients, which are: CLL (Chronic lymphocytic leukemia): is a type of cancer of the blood and the bone marrow, MCL (Mantle cell lymphoma): which is a cancer type that make B cells cancerous, FL (Follicular lymphoma): which is a cancer type that involves certain types of white blood cells known as lymphocytes.

1.1 Problem Definition

Having the fact that curing cancer is difficult does not mean that we will not try exerting some effort to collect some information about it. So, we focused on classifying the lymphoma's three types based on screenshots of samples from patients, thus it would be easier to diagnose the lymphoma type correctly.

Because as explained in the introduction the right diagnose of cancer saves a lot of time determining the cure for the specific type of cancer. To achieve this, there is a need to design a workflow. After analyzing and gathering the requirement of study, the research shall use image dataset of tissues which is structured into different folders based on classes. Such data will be stored and pre-processed which is then divided into testing and training dataset.

1.2 Motivation

Cancer diseases diverse in many body organs in a way that encourages researches and pathologists to exert more effort to help finding early effective diagnosis for most of its kinds. Having consistently developed technologies and software methods have shown progress in many medical aspects, so it is preferable to get the most out of our technological resources to help cure the most stubborn disease that has ever faced humanity. When cancer comes to threaten the immune system, a person's body's defenses break down making the body unable to defend itself. Hence, we choose Lymphoma cancer to focus our efforts on because it is critical type of cancer that attacks a vital part of the body.

Hence, having big data of high-quality Lymphoma images and biopsies-as long as other cancer types- helps us get high quality images that we could insert into existing advanced machine learning models to help it get trained using these preprocessed images. If we were able to get high accuracy classifications of the Lymphoma types- especially Non-Hodgkin Lymphoma.

1.3 Objectives

In this study an innovative and empowered machine learning framework is proposed to classify three types of lymphoma as Follicular Lymphoma (FL), Chronic Lymphocytic Lymphoma (CLL) and Mantle Cell Lymphoma (MCL).

We built a CNN model to classify different three Non-Hodgkin Lymphoma types as elaborated previously. We fed this CNN model with synthetic data and augmented data. The model produced low accuracies in the beginning, but after training the model, eventually, the model produces acceptable results and accuracies that are reliable enough to get this CNN model integrated into our website to use it for classification.

1.4 Time plan

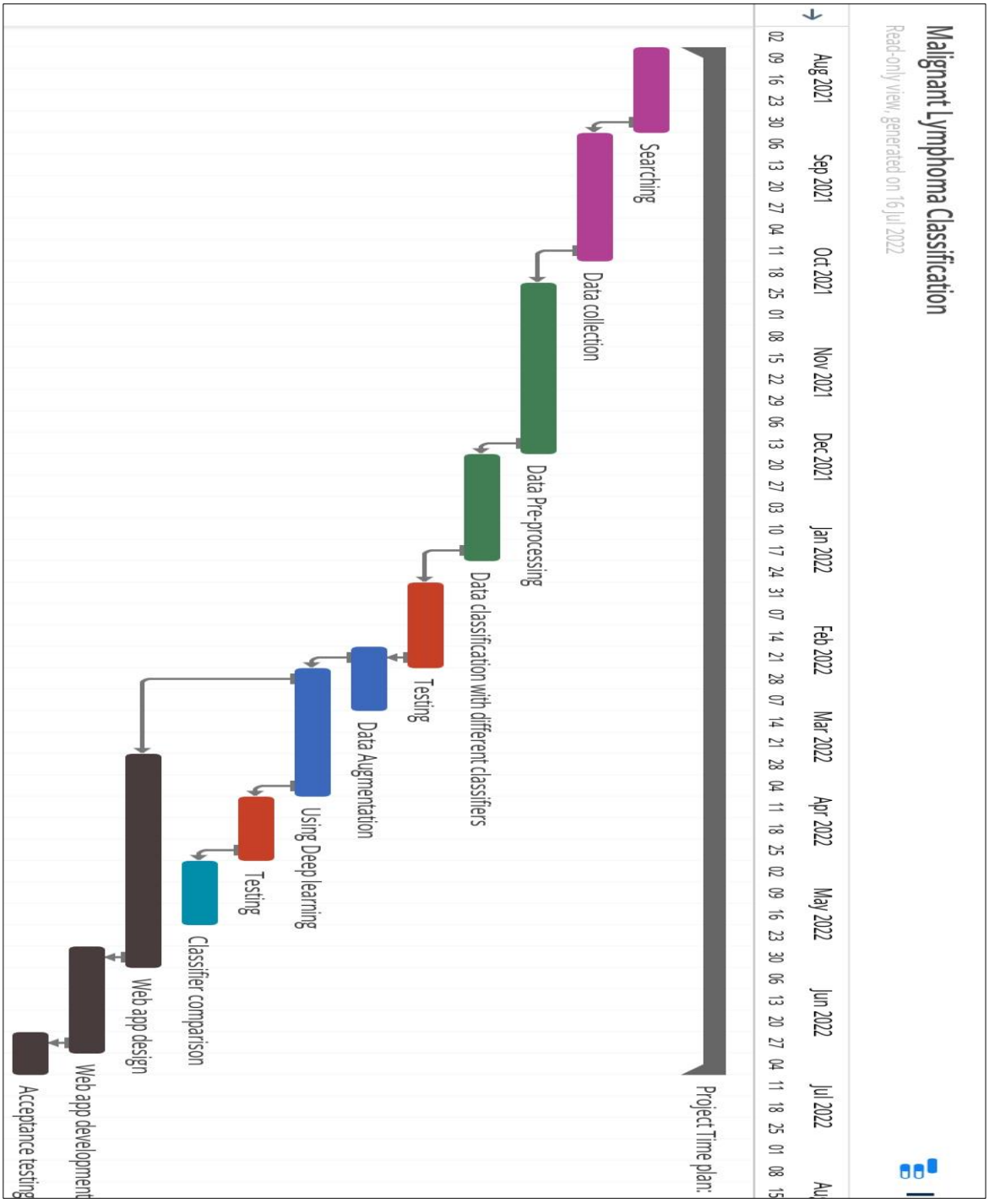


Figure 2 (Gantt chart)

1.5 Project Development Methodology

In our project we are using Waterfall method where the steps are sequential where there is we cannot return to a previous step that has ended.

The first step was preprocessing images, then we built SVM, Logistic Regression, Decision tree, and KNN models. Then we used a deep learning model to increase the accuracy.

After that we used Agile method where the steps are incremental and by which we can return to previous steps after having them done with.

In CNN model weights and architecture are updated regularly.

Advantages:

Flexibility: When compared to the Waterfall technique, the Hybrid method provides substantially more flexibility after the planning stage. You can make changes as they're asked as long as the requirements don't alter significantly.

More structured: The Hybrid technique tackles one of the most common concerns about Agile – a lack of structure and preparations – by adopting the first planning phase from Waterfall

1.6 Tools

- Machine Learning Technique
  - SVM
  - Logistic Regression
  - Decision Tree
  - KNN
- Deep Learning Algorithm
  - Convolutional Neural Network (CNN)
- Google Collaboratory
- Python Programming Language
- Python Libraries
  - Tensorflow
  - Numpy
  - Decision Tree Classifier
  - SVC
  - Logistic Regression
  - SV2
  - K neighbors classifier
  - Keras
  - PILE
  - OS
  - Shutil
  - Glob
- JavaScript Programming Language
- HTML
- CSS
- PHP
- Ajax
- Bootstrap
- jQuery

1.7 Report Organization

Chapter 2:

We will be presenting the previous problems' solutions and other related work, describe its methodology and criticize its disadvantages.

According to the previous analysis, we figured out that we need to find better solutions and machine learning technique to get higher accuracies.

After months of searching and studying, we managed to develop the previously used techniques that helped us implement our ideas.

Our data consists of 3 types:
- Chronic Lymphocytic Leukemia (CLL)
- Follicular Lymphoma (FL)
- Mantle Cell Lymphoma (MCL)

Which was a little hard to find a big data set of, but luckily, we found enough.

Chapter 3:

We will determine the most suitable functional requirements and non-functional requirements for our project.
Then we will illustrate the use case diagram to our project according to functional requirements for our project.

Chapter 4:

We will present the project by designing System Component Diagram, System Class Diagrams, Sequence Diagrams and System GUI Design.

Chapter 5:

After we collected the data, we applied different processes on our data, such as: Augmenting, slicing, resizing, scaling. We applied these processes to train our data to get better accuracy.

Chapter 6:

After we trained the data, we classified the trained data by machine learning models.
Then, we compared different models results. After that we used deep learning (CNN).
After several trials using different models of CNN to achieve the best accuracy.

Chapter 7:

- After classifying the data we found related to Lymphoma cancer, we found out that it is not easy to extract all the features in sample photos as accurate as pathologists. So, we developed the different machine learning models we used to help classifying the data better.

- We are working on expanding the classes of our model to more than only 3 types of Non-Hodgkin Lymphoma Types.

- We are elaborating the obstacles that faced us during our project, and even solutions for them

Chapter 2: Related Work

Article 1:

- 4 convolutional neural network layers
- Dropout is used by 30% of in each layer
- filters all of size 5
- max pooling all of size (3*3) and 3 strides
- used tanh activation function in each layer
- used activation function softmax for output
- 2 added fully connected layers, layer with dense 500 and layer with dense 250
- used optimizer Adam

This model shows an accuracy of 35%

Article 2:

- used VGG16 model with none weighs and max pooling
- Dropout is used by 50%
- 2 fully connected layers are added with dense 4000
- used activation function relu for layers
- used activation function softmax for output
- used optimizer rms

This model shows an accuracy of 35%

Article 3:

- 5 convolutional neural network layers
- filters of size (11*11) for first 2 layers and the rest of size(3*3)
- max pooling of size 2 and strides 2
- used relu activation function in each layer
- used activation function softmax for output
- used adam optimizer
- used dropout by 40% in each layer
- 3 fully onnected layers are added with dense 4000

This model shows an accuracy of 45%

Article 4:

- 4 convolutional neural network layers
- Dropout is used by 30% of in each layer
- filters all of size 3
- max pooling all of size (3*3)
- used relu activation function in each layer
- used activation function softmax for output
- 1 added fully connected layers, layer with dense 128
- used optimizer adam

This model shows an accuracy of 90%

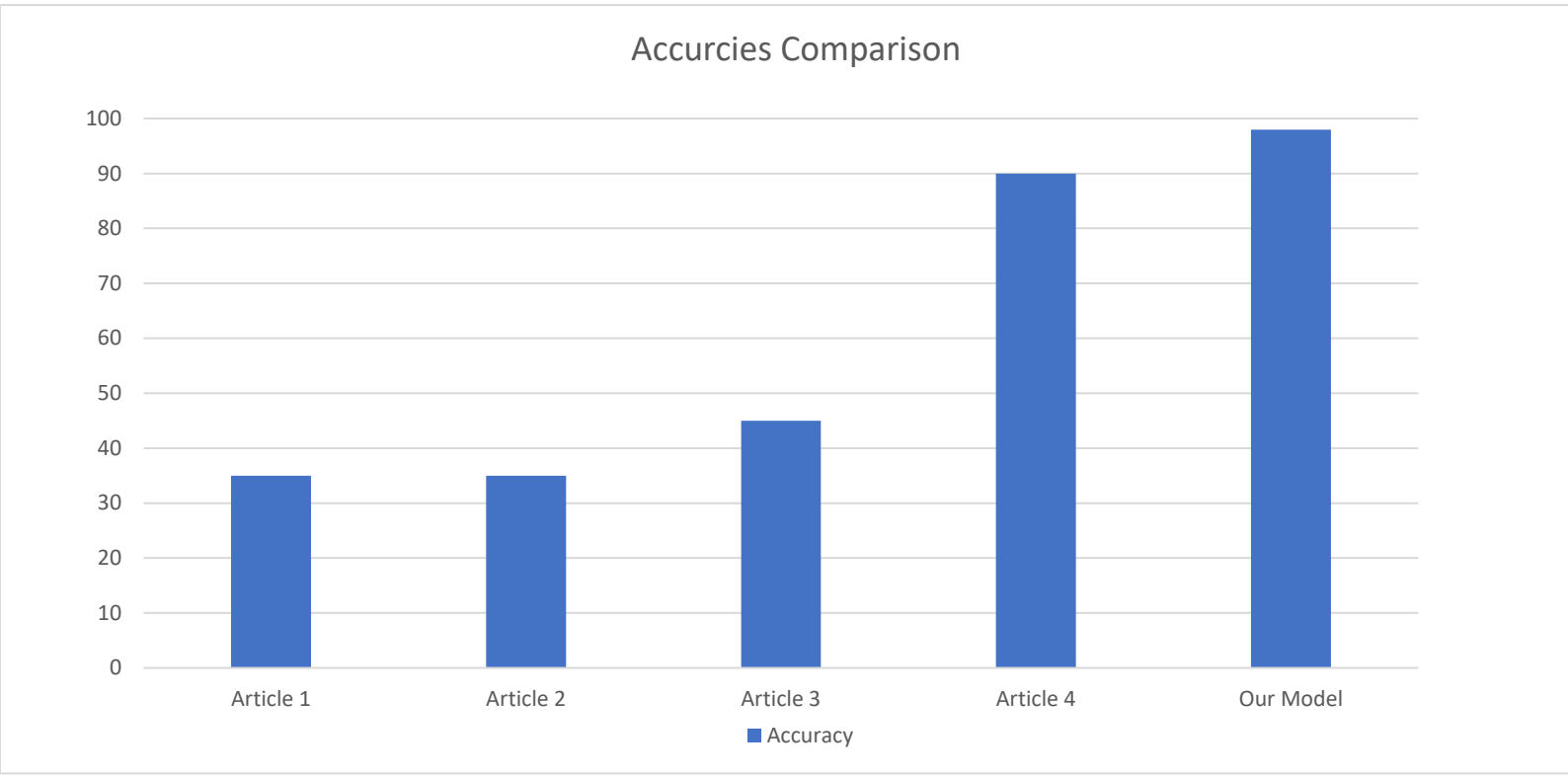


Figure 3 (Accuracies comparison)

As it shows in the graph:

- Model 1 produced 35% accuracy
- Model 2 produced 35% accuracy
- Model 3 produced 40% accuracy
- Model 4 produced 90% accuracy

But our model shows an accuracy of 98% which shows the efficiency of the chosen techniques in our model.

Chapter 3: System Analysis

3.1 Project Specifications

3.1.1 Functional Requirements:

-GetInput()

-Sixteen slices of 1 image

-Preform image processing as: resizing

-PredictOutput()

-Classify the user into 3 classes of Non-Hodgkin Lymphoma

3.1.2 Non-Functional Requirements:

-Performance: as quick as possible response

-Availability: available for anyone to use at anytime

-Usability: User friendly GUI

-Scalability: Handles growing number of patients

3.2 Use Case Diagram:

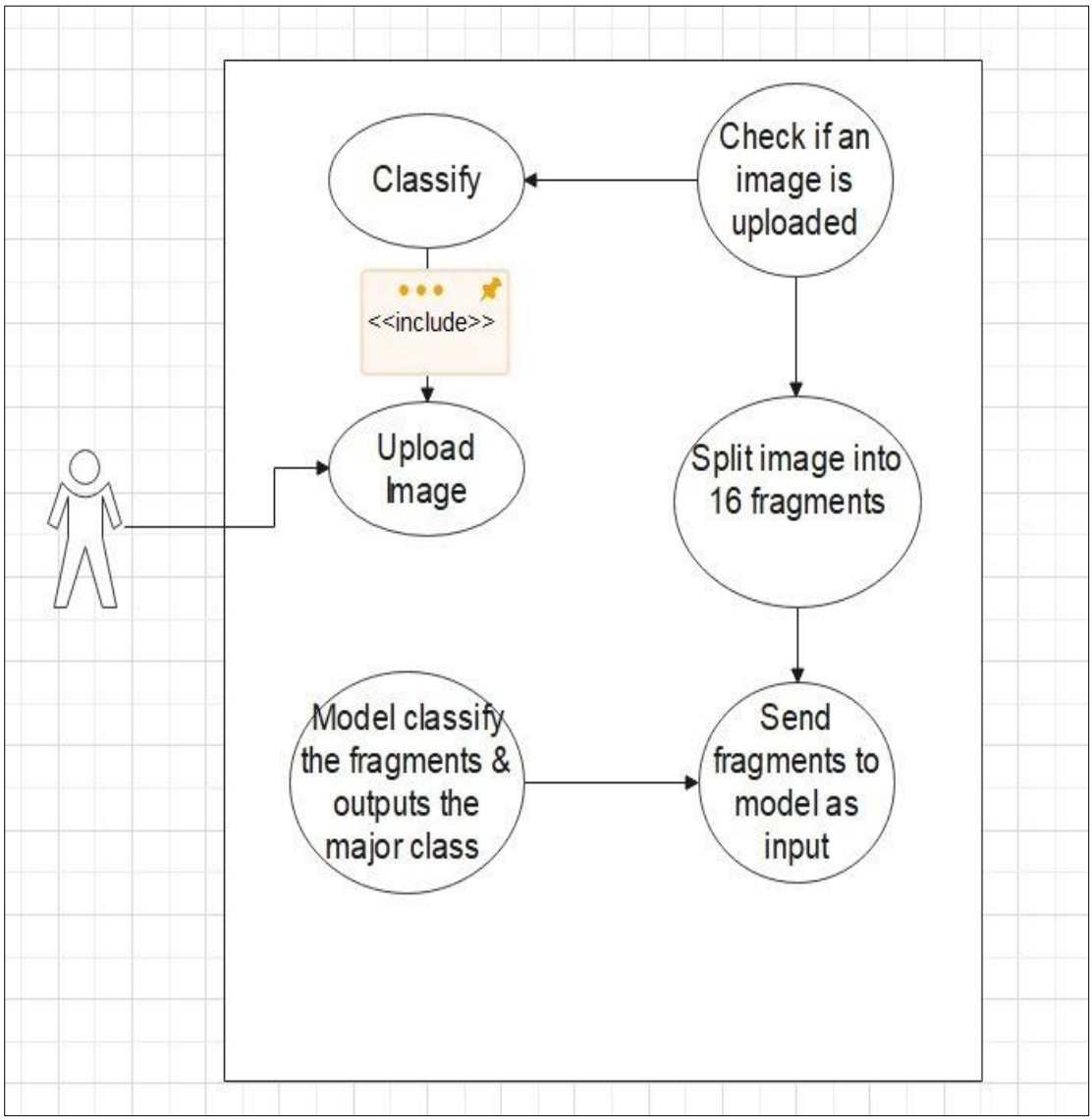


Figure 4 (Use case diagram)

Chapter 4: System Design

4.1 System Component Diagrams

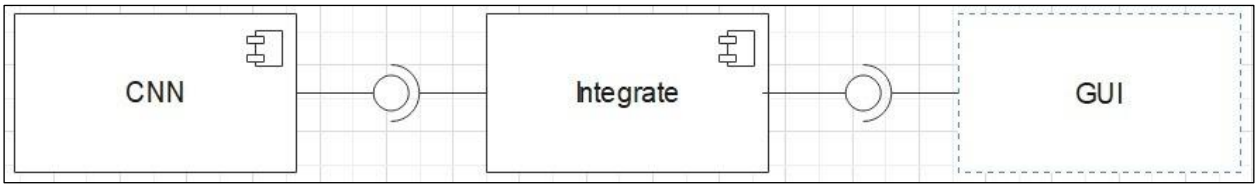


Figure 5 (System component diagram)

4.2 System Class Diagrams

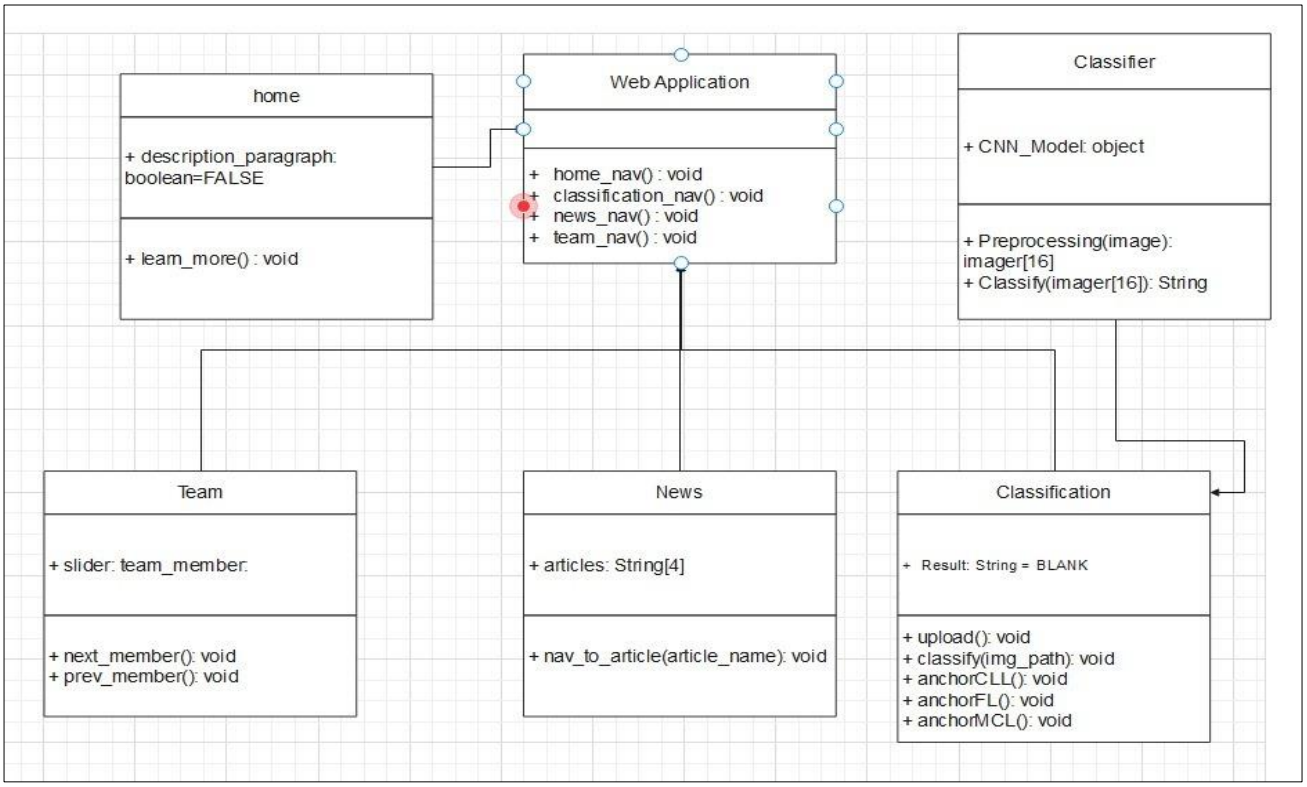


Figure 6 (System class diagram)

4.3 Sequence Diagram

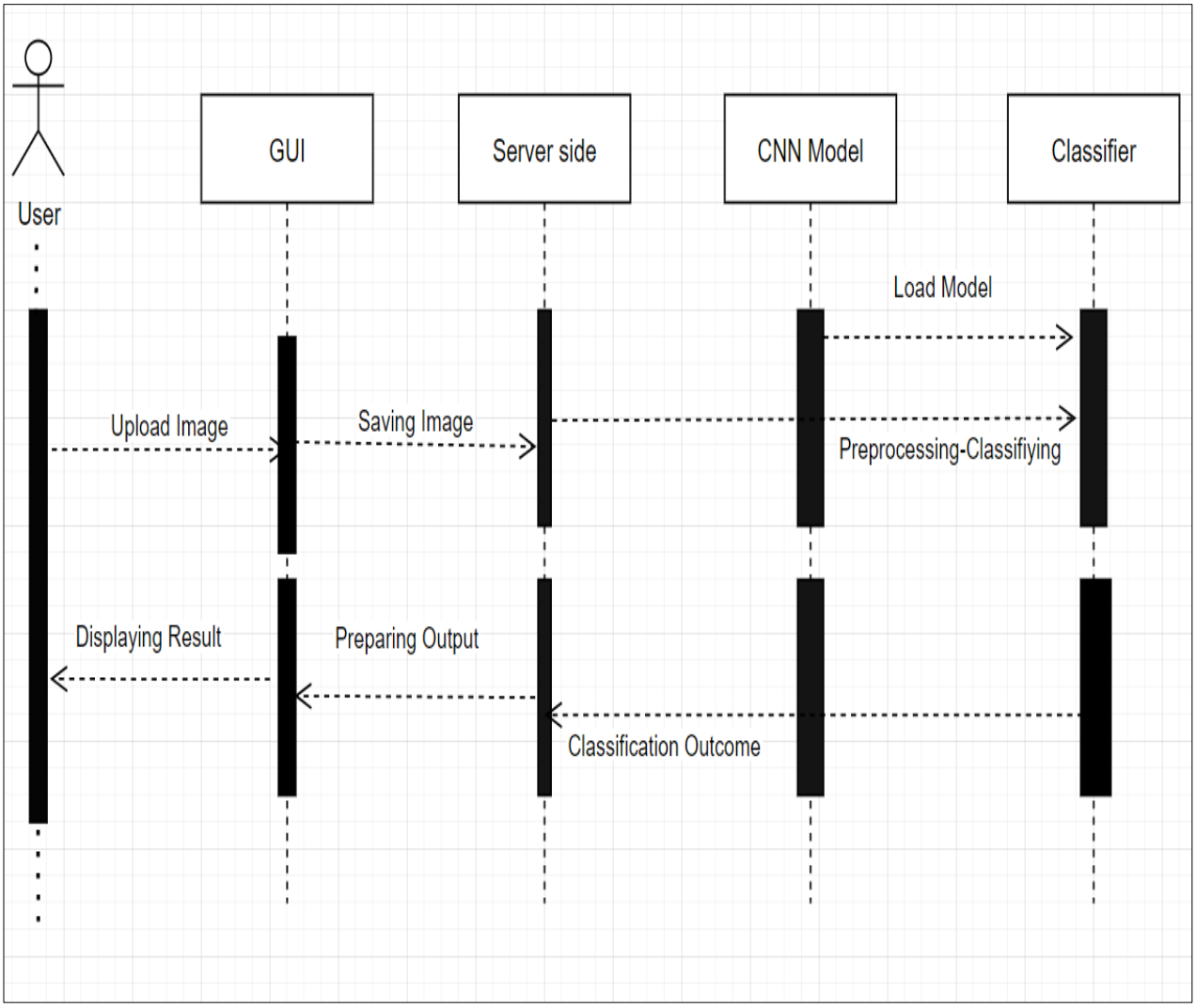


Figure 7 (Sequence diagram)

4.4 System GUI Design

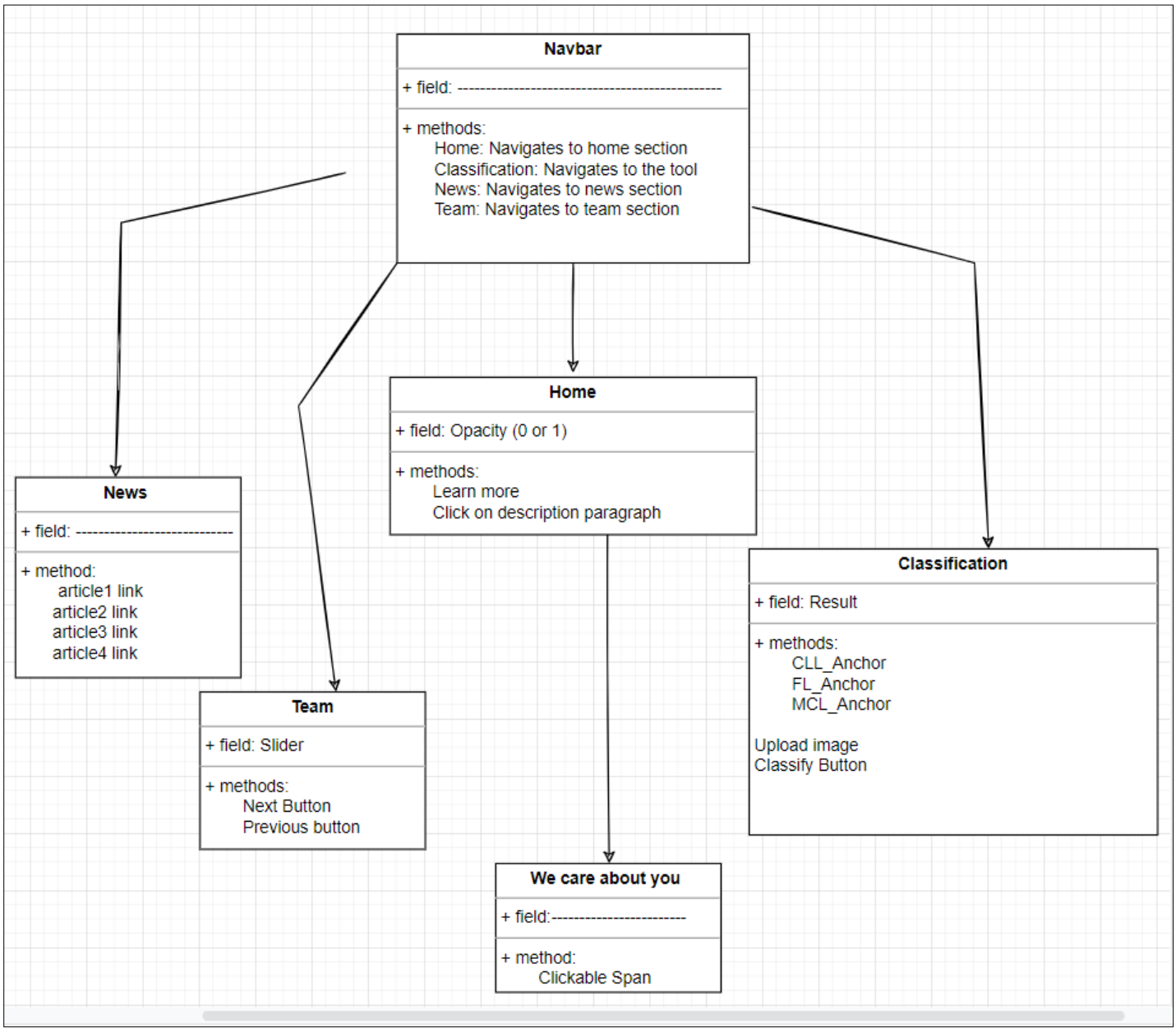


Figure 8 (System GUI design)

Chapter 5: Data Preprocessing

5.1 Augmentation

The accuracy of deep learning models largely depends on the quality, quantity, and contextual meaning of training data. However, data scarcity is one of the most common challenges in building deep learning models.

In production use cases, collecting such data can be costly and time-consuming.

Today, there are a lot of privacy concerns revolving around data collection and usage. Hence, many researchers and companies are using synthetic data generation techniques to build datasets. However, due to limitations such as its lack of resemblance to the original data, augmented data is generally preferred over synthetic data.

A question may arise about the difference between augmented data and synthetic data.

- **Synthetic data**: When data is generated artificially without using real-world images. Synthetic data are often produced by Generative Adversarial Networks
- **Augmented data**: Derived from original images with some sort of minor geometric transformations (such as flipping, translation, rotation, or the addition of noise) in order to increase the diversity of the training set.

In machine learning, the situation when the model does not generalize well from the training data to unseen data is called **overfitting**. As you might know, it is one of the trickiest obstacles in applied machine learning.

After identifying the problem, you can prevent it from happening by applying regularization or training with more data. Still, sometimes you might not have additional data to add to your initial dataset. Acquiring and labeling additional data points may also be the wrong path. Of course, in many cases, it will deliver better results, but in terms of work, it is often time-consuming and expensive.

That is where Data Augmentation comes in.

Data augmentation is a process of artificially increasing the amount of data by generating new data points from existing data. This includes adding minor alterations to data or using machine learning model to generate new data points in the latent space of original data to amplify the dataset.

Using set of techniques to artificially increase the amount of data by generatin g new data points from existing data. This includes making small changes to d ata or using deep learning models to generate new data points.

Why is it important now?

Machine learning applications especially in the deep learning domain continue to diversify and increase rapidly. Data augmentation techniques may be a good tool against challenges which the artificial intelligence world faces.

Data augmentation is useful to improve performance and outcomes of machine learning models by forming new and different examples to train datasets. If the dataset in a machine learning model is rich and sufficient, the model performs better and more accurately.

For machine learning models, collecting and labeling of data can be exhausting and costly processes. Transformations in datasets by using data augmentation techniques allow companies to reduce these operational costs.

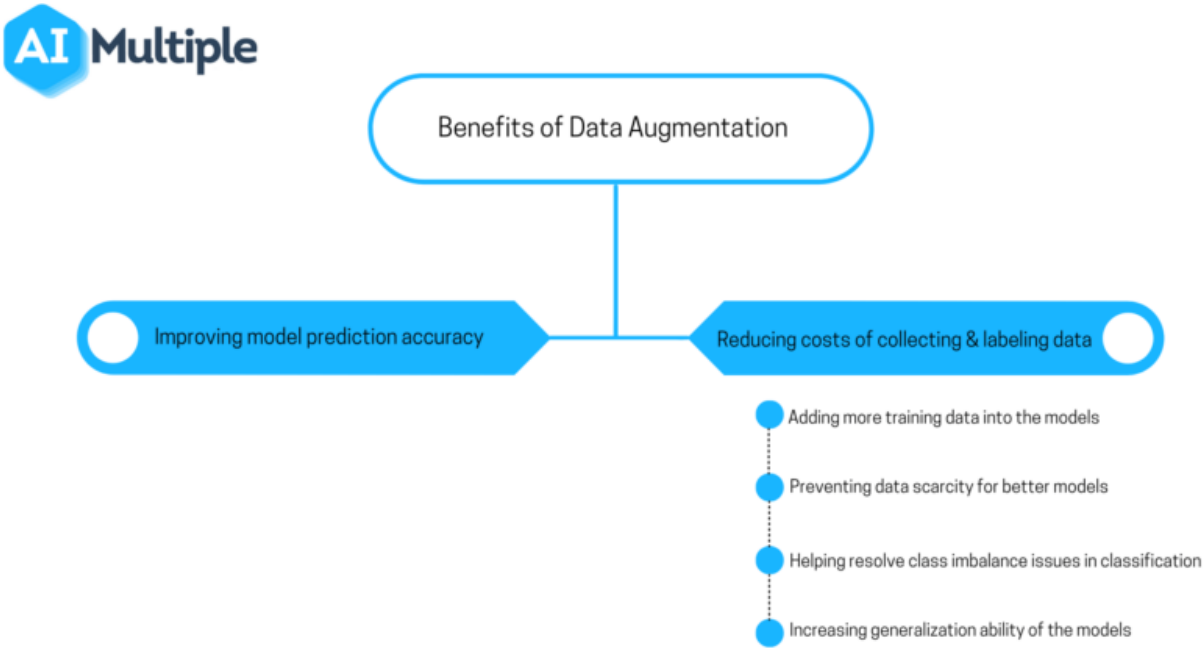


Figure 9 (Augmentation benefits)

The importance of Data Augmentation

- Improving model prediction accuracy
    - adding more training data into the models
    - preventing data scarcity for better models
    - reducing data overfitting (i.e. an error in statistics, it means a function corresponds too closely to a limited set of data points) and creating variability in data
    - increasing generalization ability of the models
    - helping resolve class imbalance issues in classification
- Reducing costs of collecting and labeling data
- Enables rare event prediction
- Prevents data privacy problems

Data Augmentation techniques

Finally, let's take a look at some of the most popular data augmentation methods.

1. Position Augmentation

1. **Vertical & Horizontal flip**: Horizontally flip the given image randomly with a given probability.
2. **Rotation**: Rotate the image by some angle.

2. Color Augmentation

1. **Brightness**: One way to augment is to change the brightness of the image. The resultant image becomes darker or lighter compared to the original one.
2. **Contrast:** The contrast is defined as the degree of separation between the darkest and brightest areas of an image. The contrast of the image can also be changed.

In our project, Due to the scarcity of our data, our model didn’t have enough data inputs to train (374 image), hence, the accuracy of our models was low.

Most of the analyzed images are extracted from Biopsy, all of which are expensive to collect and labor intensive.

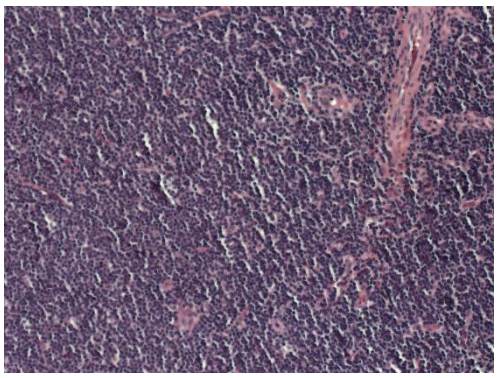
Because of the rarity of diseases, the confidentiality of patients, the need for medical experts to mark, and the cost and manual effort needed to carry out medical imaging processes, it is especially difficult to create large medical image datasets.

Convolutional Neural Networks have achieved great attention in disease classification. It is very important for a model to train on a huge data. When a neural network is trained on a small data, there is a chance of overfitting. A model is considered over-fit when it perform well on training data but poor on any new dataset

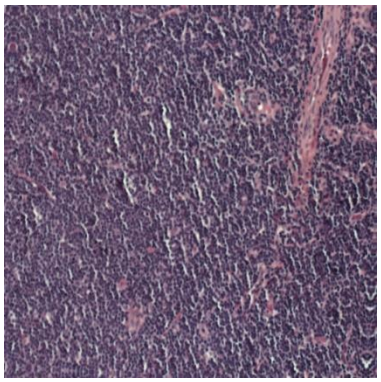
Different approaches have been recommended over the period of time to reduce overfitting. The most common ways to avoid the problem of overfitting are adding regularization term or dropout technique, There is also another technique called batch normalization that is helpful in this scenario

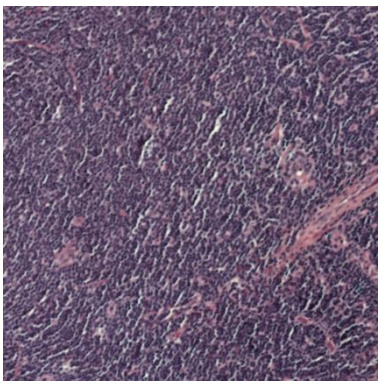
The training dataset was increased with the help of techniques like flipping the images, rotating by 45◦, Changing Brightness or Contrast Range, sixteen new images were created for each original image using data augmentation.

CLL



ROTATION RANGE OF 0->45:





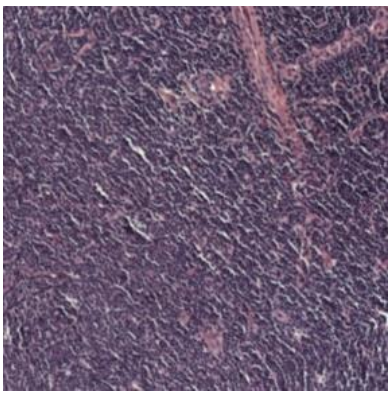


Figure 10 (Rotation CLL)

VERTICAL AND HORIZONTAL FLIPPING:

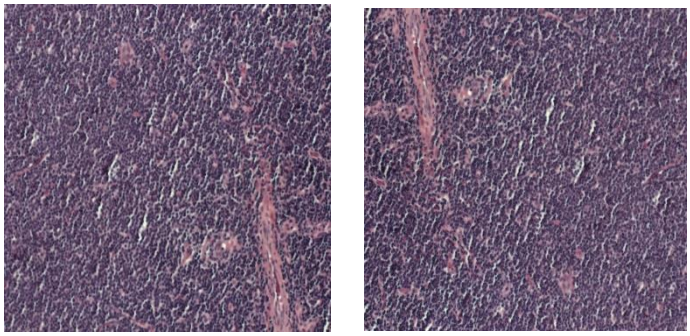


Figure 11 (Flipping CLL)

BRIGHTNESS RANGE OF 0.5->1.5:

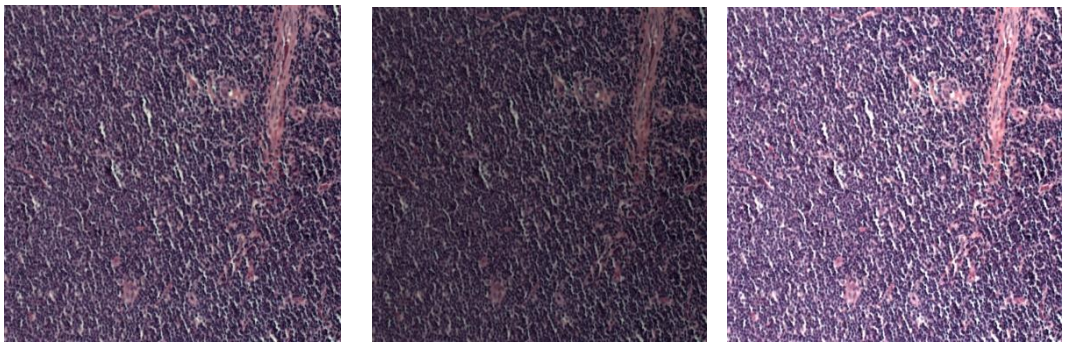


Figure 11 (Brightness CLL)

CONTRAST:

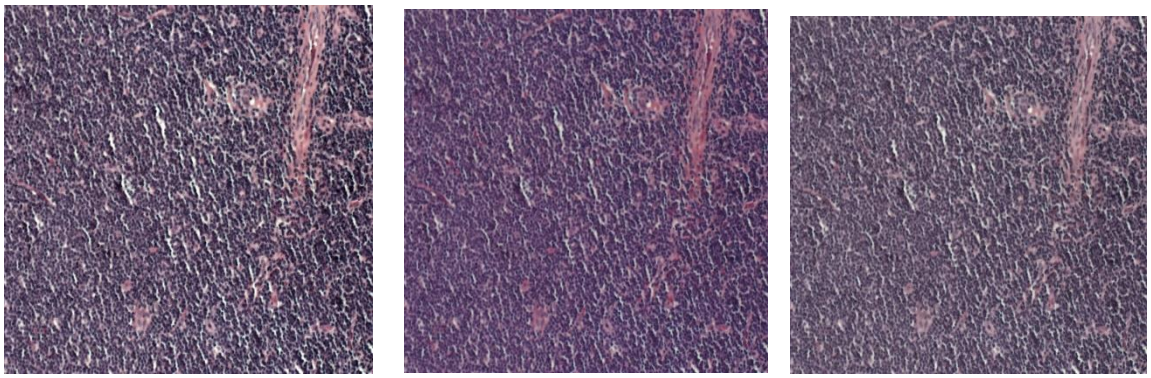
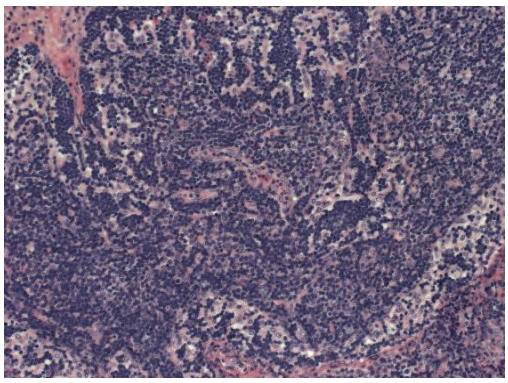
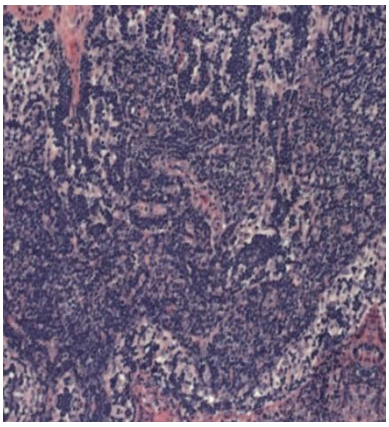


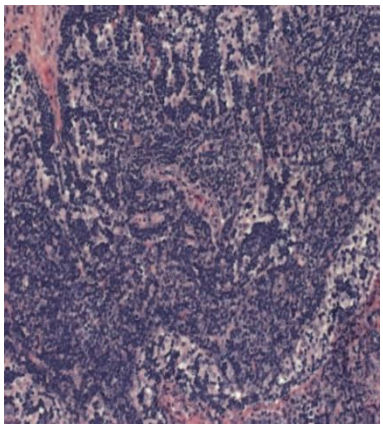
Figure 13 (Contrast CLL)

FL



Rotation Range of 0->45:





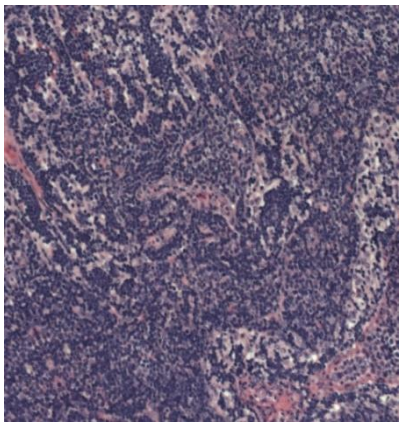
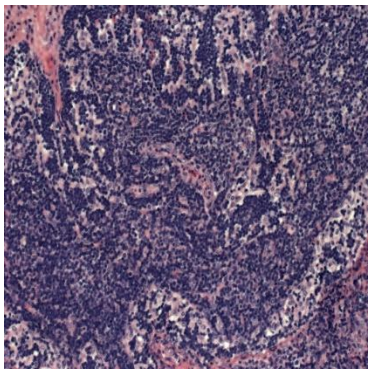
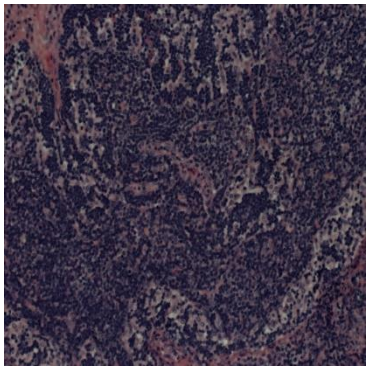


Figure 14 (Rotation FL)

BRIGHTNESS RANGE OF 0.5->1.5:





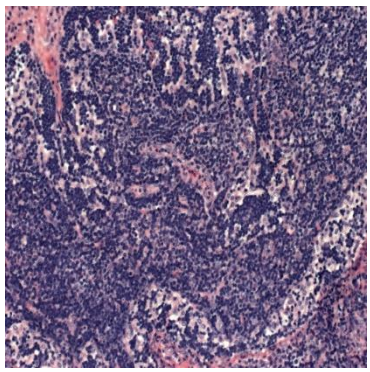


Figure 15 (Brightness FL)

Vertical and Horizontal Flipping:

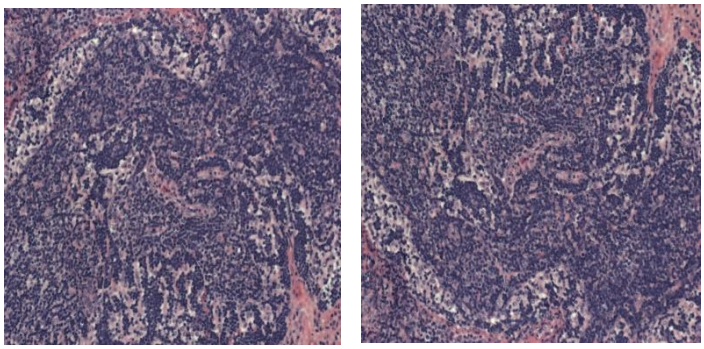


Figure 16 (Flipping FL)

Contrast:

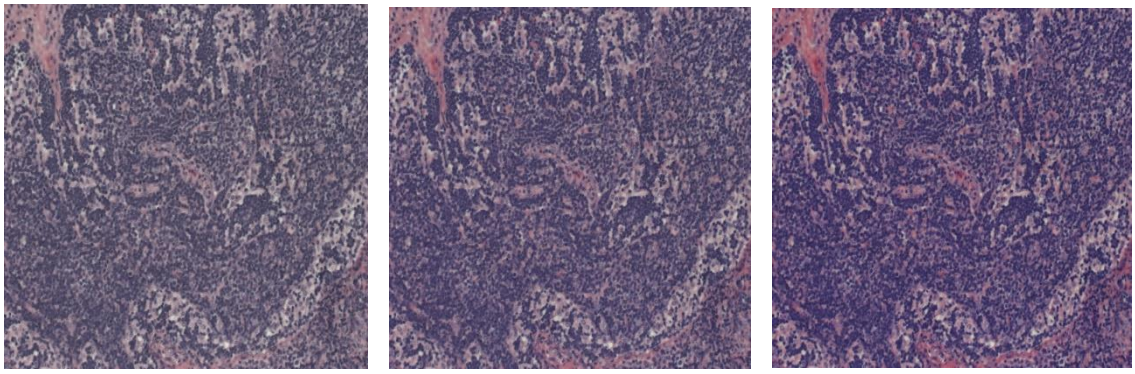
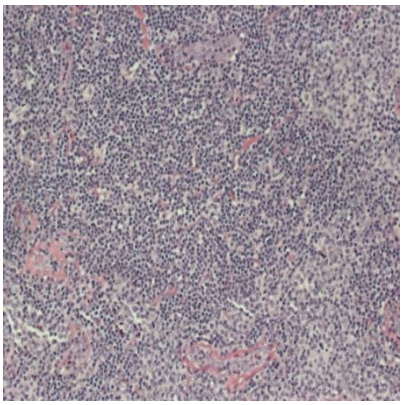
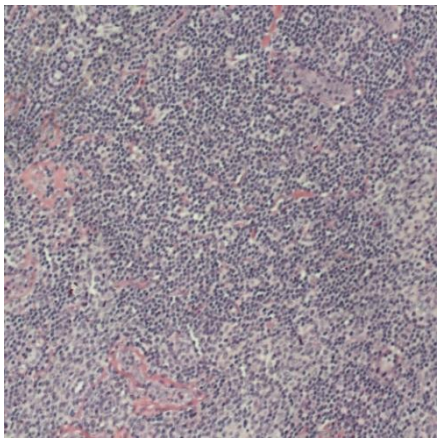


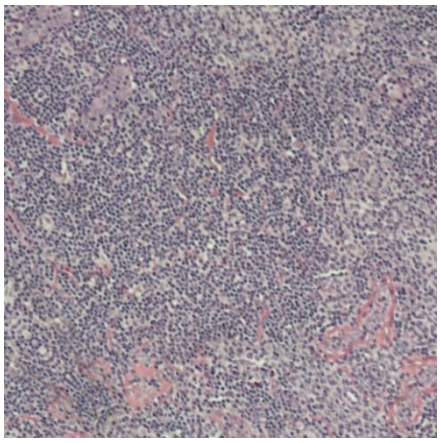
Figure 17 (Contrast FL)

MCL



Rotation Range 0->45:





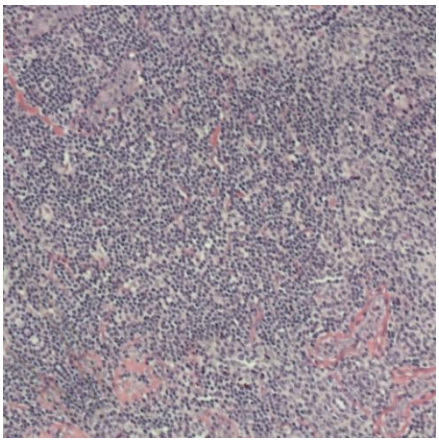
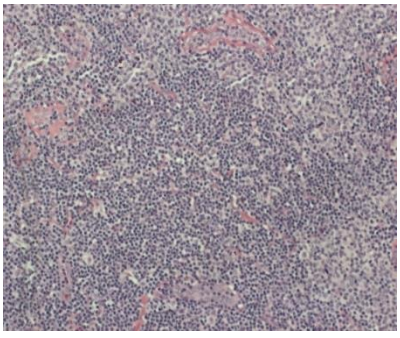


Figure 18 (Rotation MCL)

Vertical and Horizontal Flipping:



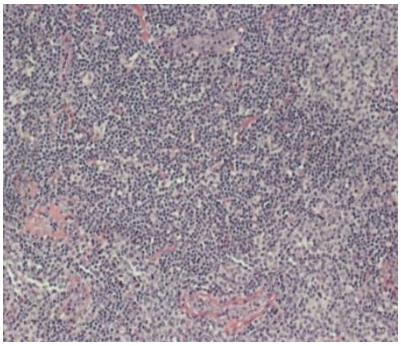
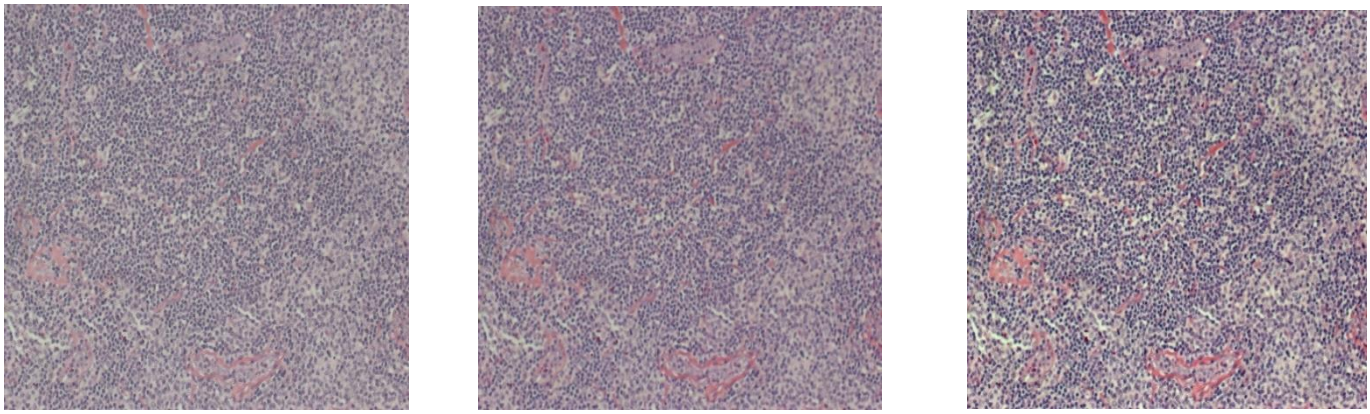


Figure 19 (Flipping MCL)

Contrast:



BRIGHTNESS >1.5: *RANGE OF 0.5-*

Figure 20 (Contrast MCL)

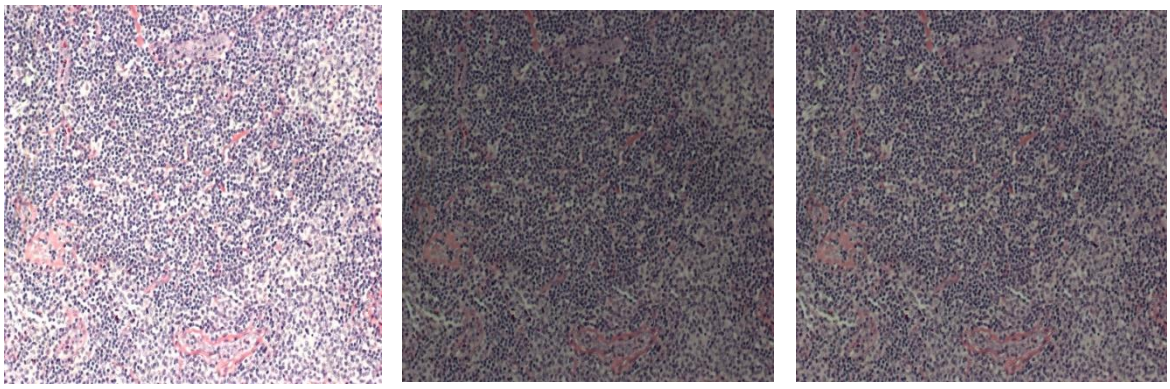


Figure 21 (Brightness)

Slicing

Because we should resize images before putting them into models from 1388 X 1040 to 224 X 224

The features of the images are hard to classify by models ex. From FL type

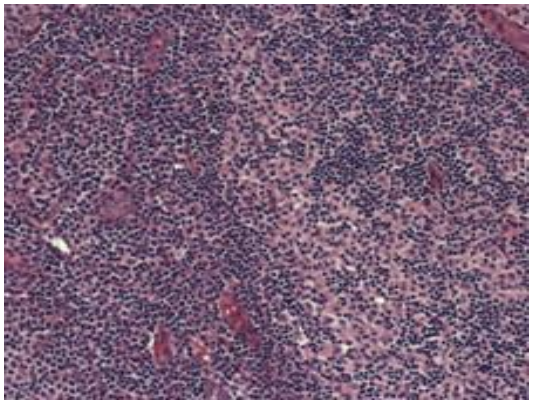


Figure 22 (Original image)

So we should slice image to multiple images of the same type ex. Slice it to 16 image and this is two examples of them

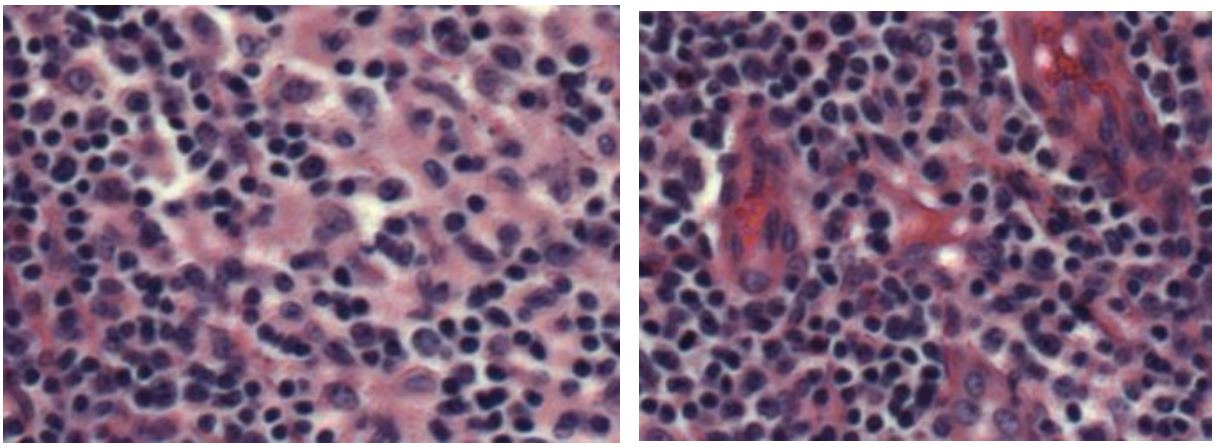


Figure 23 (Sliced images)

Scaling

As we know the pixel values range from 0 to 255. And in Deep Neural Network, the computation of high numeric values may become more complex.

To reduce this we can normalize the values to range from 0 to 1.

Resizing

Resizing images is a critical pre-processing

Deep learning models train faster on small images. A larger input image requires the neural network to learn from four times as many pixels, and this increase the training time for the architecture.

Chapter 6 Implementation and Testing

Dataset Description

Malignant lymphoma is a cancer affecting lymph nodes.

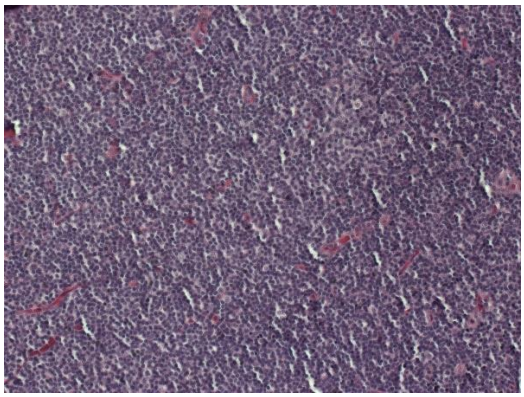
Three types of malignant lymphoma are represented in the set:

- CLL (chronic lymphocytic leukemia);
- FL (follicular lymphoma);
- MCL (mantle cell lymphoma).

The ability to distinguish classes of lymphoma from biopsies sectioned and stained with Hematoxylin/Eosin (H+E) would allow for more consistent and less demanding diagnosis of this disease. Only the most expert pathologists specializing in these types of lymphomas are able to consistently and accurately classify these three lymphoma types from H+E-stained biopsies. The standard practice is to use class-specific probes in order to distinguish these classes reliably.

This dataset is a collection of samples prepared by different pathologists at different sites. There is a large degree of staining variation that one would normally expect from such samples.

Chronic Lymphocytic Leukemia (CLL)



Follicular Lymphoma (FL)

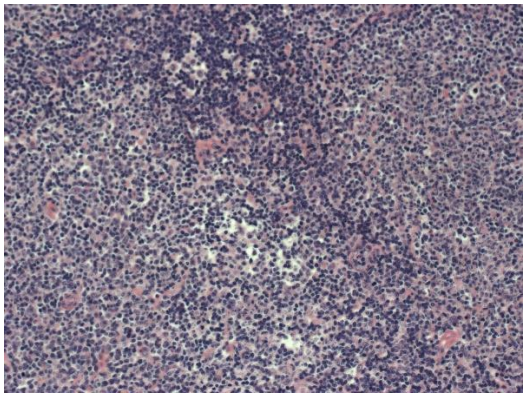
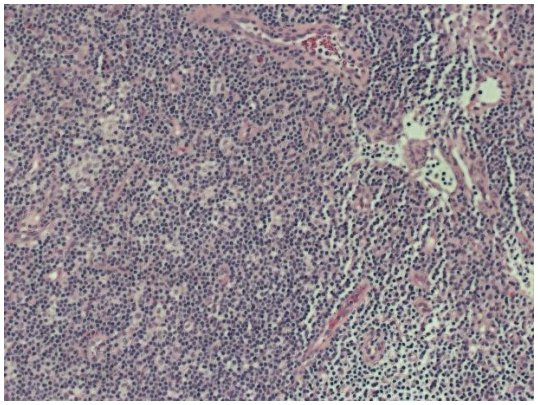


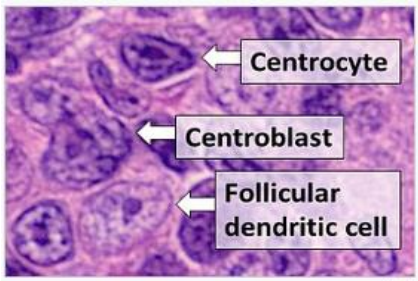
Figure 24 (Types of lymphoma)

Mantle Cell Lymphoma (MCL)

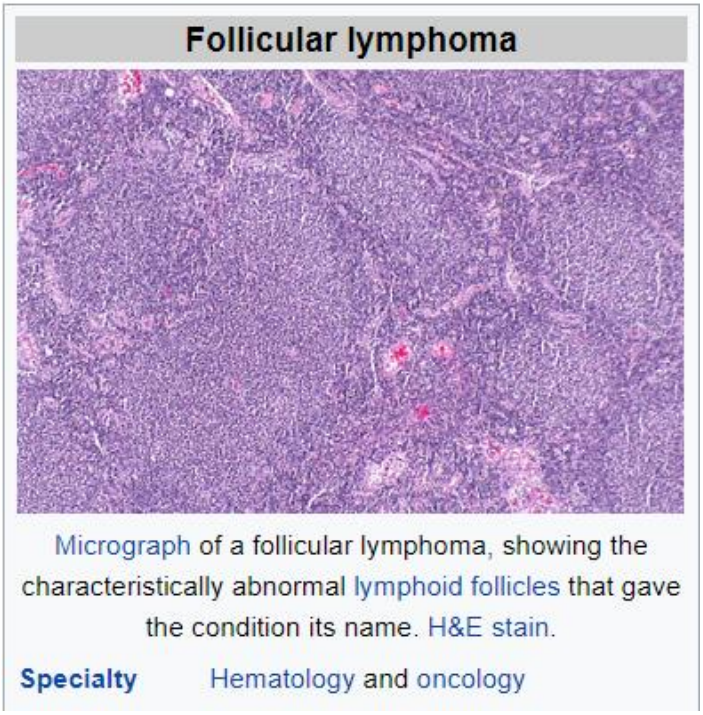


Feature Description

FL

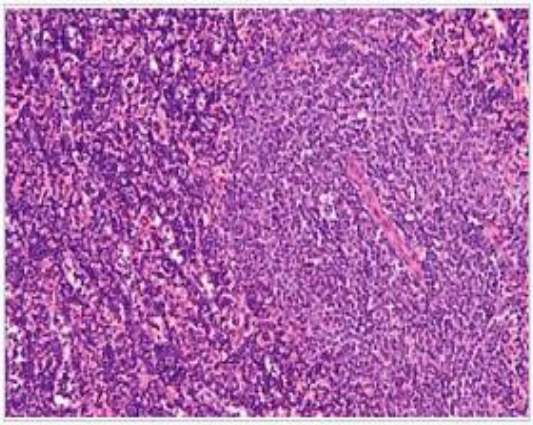


Histologic comparison of cell types in a germinal center, H&E stain:
- **Centrocytes** are small to medium size with angulated, elongated, cleaved, or twisted nuclei.
- **Centroblasts** are larger cells containing vesicular nuclei with one to three basophilic nucleoli apposing the nuclear membrane.
- **Follicular dendritic cells** have round nuclei, centrally located nucleoli, bland and dispersed chromatin, and flattening of adjacent nuclear membrane.

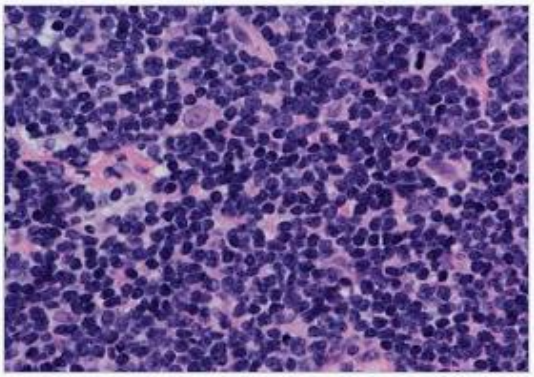


Micrograph of a follicular lymphoma, showing the characteristically abnormal lymphoid follicles that gave the condition its name. H&E stain.

Specialty Hematology and oncology



Micrograph of a lymph node affected by B-CLL showing a characteristic proliferation center (right of image), composed of larger, lighter-staining, cells, H&E stain



Mantle cell lymphoma. Notice the irregular nuclear contours of the medium-sized lymphoma cells and the presence of a pink histiocyte. By immunohistochemistry the lymphoma cells expressed CD20, CD5 and Cyclin D1 (high power view, H&E)

CLL **MCL**

Figure 25 (Features description)

5.1.2. Project stages

Phase 1

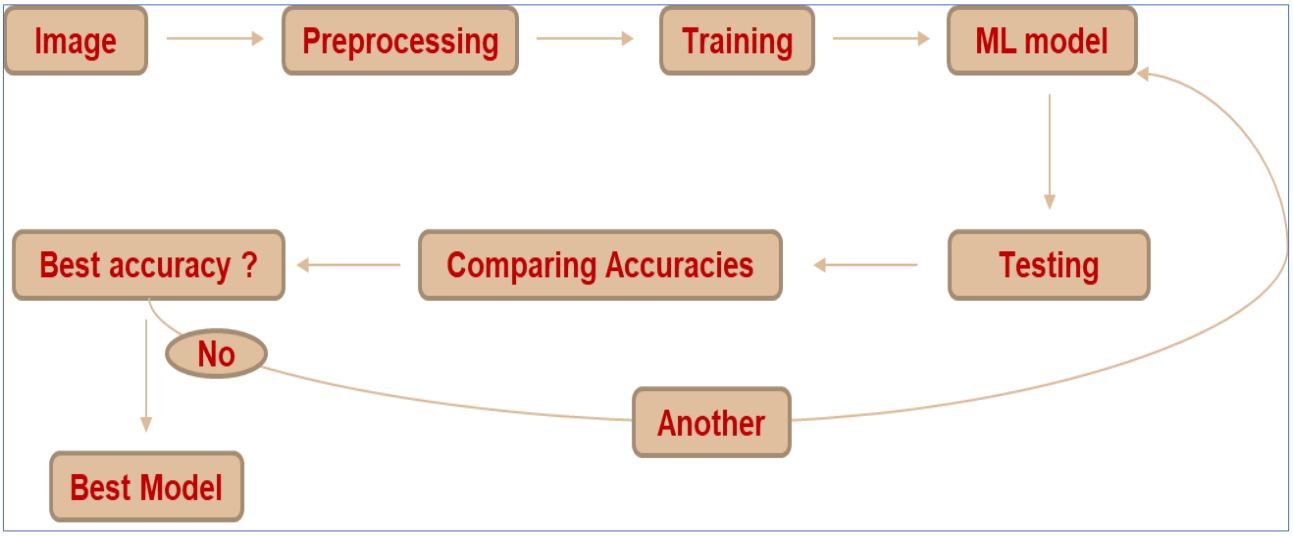


Figure 26 (Phase 1)

Phase 2

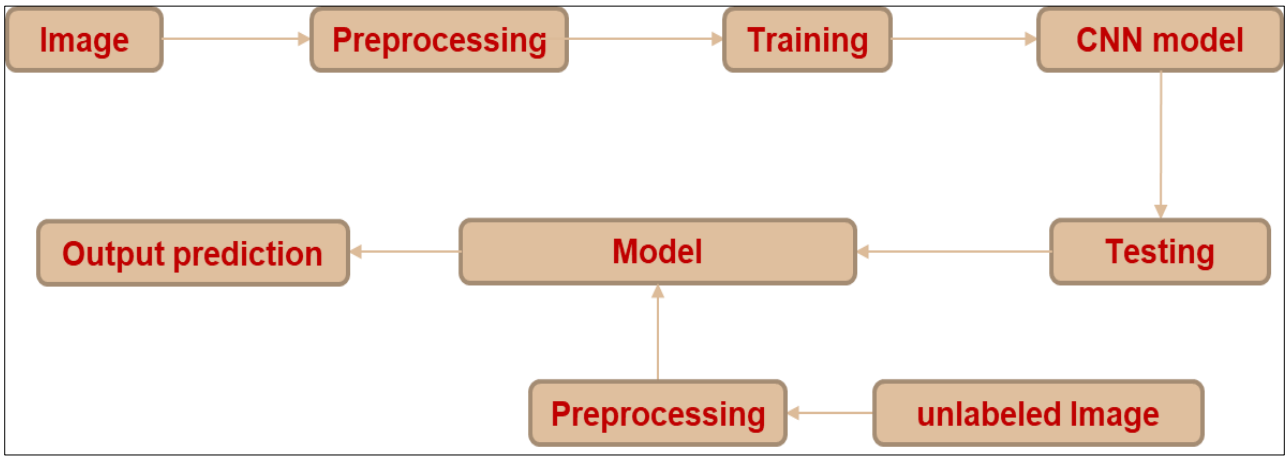


Figure 27 (Phase 2)

Traditional Machine Learning Models:

Logistic regression:

Trial 1 without augmentation

	Precision	Recall	F1-Score	Support
CLL	0.6	0.44	0.51	27
FL	0.56	0.72	0.63	25
MCL	0.52	0.52	0.52	23
Accuracy			0.56	75
Macro Avg	0.56	0.56	0.55	75
Weighted Avg	0.56	0.56	0.55	75
train accuracy : 1.0				
test accuracy : 0.56				

Table 1 (LR)

Trial 2 with augmentation

	Precision	Recall	F1-Score	Support
CLL	0.42	0.49	0.45	100
FL	0.64	0.6	0.62	134
MCL	0.58	0.52	0.55	113
Accuracy			0.54	347
Macro Avg	0.54	0.54	0.54	347
Weighted	0.55	0.54	0.55	347
Train Accuracy : 1.0				
Test Accuracy : 0.54				

Note: testing accuracy in logistic regression without augmentation is better

Decision tree

Trial 1 without augmentation

	Precision	Recall	F1-Score	Support
CLL	0.43	0.33	0.38	27
FL	0.39	0.56	0.46	25
MCL	0.39	0.3	0.34	23
Accuracy			0.4	75
Macro Avg	0.4	0.4	0.39	75
Weighted Avg	0.4	0.4	0.39	75
Train Accuracy : 1				
Test Accuract : 0.4				

Table 2 (DT)

Trial 2 with augmentation

	Precision	Recall	F1-Score	Support
CLL	0.41	0.36	0.38	100
FL	0.62	0.62	0.62	134
MCL	0.37	0.41	0.38	113
Accuracy			0.48	347
Macro Avg	0.47	0.46	0.46	347
Weighted	0.48	0.48	0.48	347
Train Accuracy : 1.0				
Test Accuracy : 0.475				

Note: testing accuracy in decision tree with augmentation is better

Support vector machine (SVM):

Trial 1 without augmentation

SVM with Kernel rbf

	Precision	Recall	F1-Score	Support
CLL	0.63	0.52	0.57	23
FL	0.66	0.74	0.7	31
MCL	0.52	0.52	0.52	21
Accuracy			0.61	75
Macro Avg	0.6	0.6	0.6	75
Weighted	0.61	0.61	0.61	75
Train Accuracy : 1.0				
Test Accuracy : 0.613				

Table 3 (SVM)

SVM with kernel

FL	0.53	0.88	0.66	24
MCL	0.75	0.11	0.19	27
Accuracy			0.61	75
Macro Avg	0.6	0.6	0.6	75
Weighted	0.6	0.53	0.47	75
Train Accuracy : 0.95				
Test Accuracy : 0.533				

SVM with kernel polynomial

	Precision	Recall	F1-Score	Support
CLL	0.58	0.61	0.6	23
FL	0.52	0.79	0.63	28
MCL	0.56	0.21	0.3	24
Accuracy			0.55	75
Macro Avg	0.55	0.53	0.51	75
Weighted	0.55	0.55	0.51	75
Train Accuracy : 1				
Test Accuracy : 0.54				

Trial 2 with augmentation:

	Precision	Recall	F1-Score	Support
CLL	0.65	0.55	0.59	100
FL	0.69	0.93	0.8	134
MCL	0.68	0.5	0.57	113
Accuracy			0.68	347
Macro Avg	0.67	0.66	0.66	347
Weighted	0.68	0.68	0.67	347
Train Accuracy : 0.94				
Test Accuracy : 0.68				

Note: we used best kernel (linear) with augmentation and it gives best accuracy

K-nearest neighbors (KNN):

Trial 1 without augmentation

	Precision	Recall	F1-Score	Support
CLL	0.5	0.67	0.57	27
FL	0.5	0.6	0.55	25
MCL	0.44	0.17	0.25	23
Accuracy			0.49	75
Macro Avg	0.48	0.48	0.46	75
Weighted Avg	0.48	0.49	0.46	75
Train Accuracy : 0.608				
Test Accuract : 0.493				

Trial 2 with augmentation

Table 4 (KNN)

	Precision	Recall	F1-Score	Support
CLL	0.42	0.43	0.43	100
FL	0.7	0.77	0.73	134
MCL	0.45	0.39	0.42	113
Accuracy			0.55	347
Macro Avg	0.52	0.53	0.53	347
Weighted	0.54	0.55	0.54	347
Train Accuracy : 0.683				
Test Accuracy : 0.547				

Note: testing accuracy in KNN with augmentation is better

Conclusion:

- SVM with augmentation gave the best accuracy among all the trials
- Overall augmentation increases accuracy
- We are going to use deep learning instead

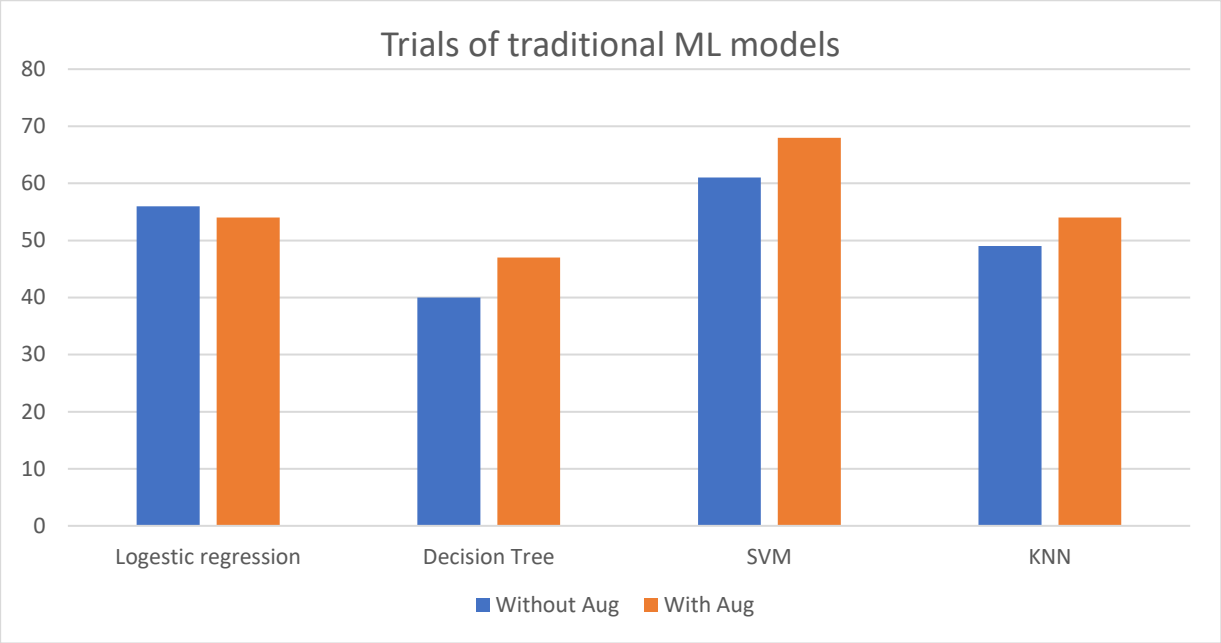


Figure 26 (Trials of ML models)

Model overview

Convolutional Neural network (CNN)

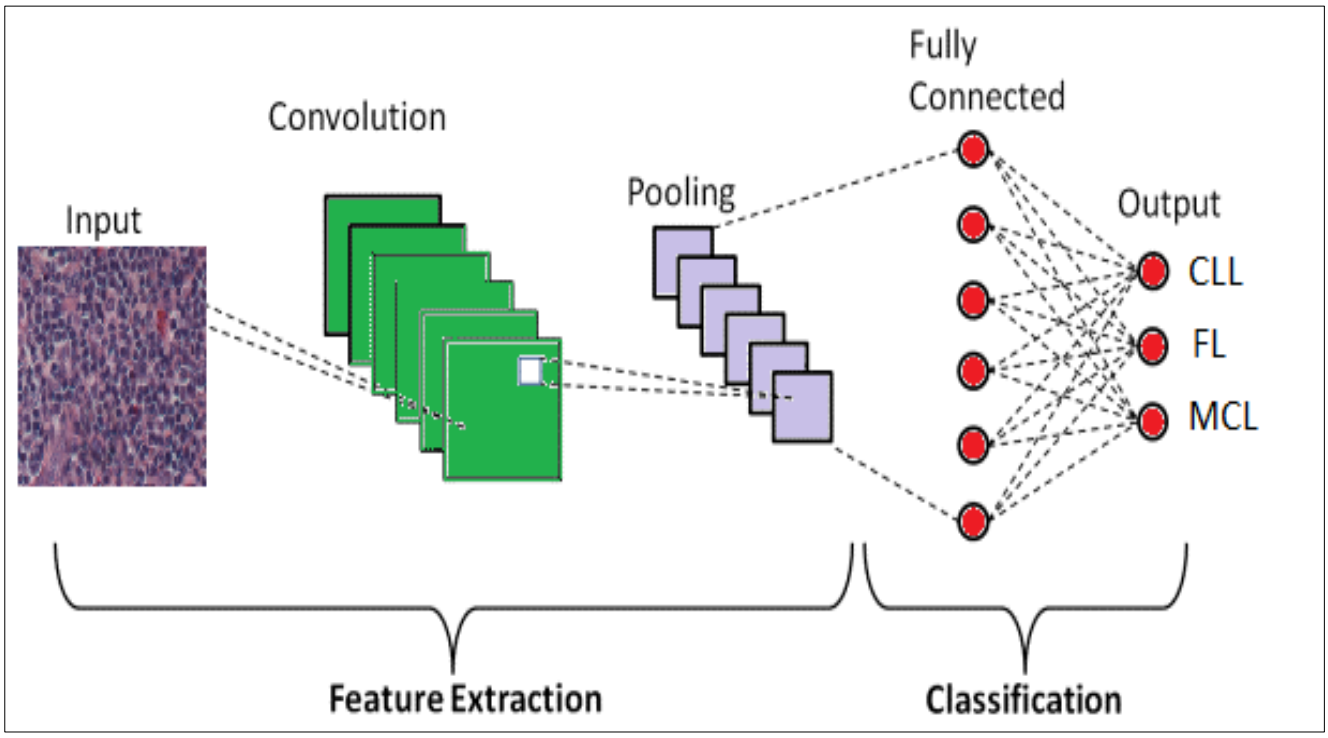


Figure 27 (CNN overview)

Steps:

1-first layer take input shape (224*224*3)

2-filters to extract features (3*3)

3-max pooling (2*2)

4-flatten fully connected layer

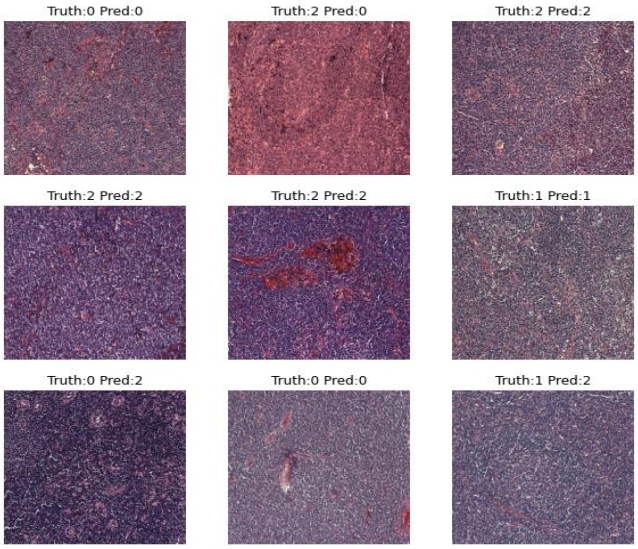
5-output layer either CLL, FL or MCL

Deep Learning Implementation and Testing

Trial 1 without augmentation and without slicing:

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 224, 224, 16)	448
batch_normalization_12 (Batc	(None, 224, 224, 16)	64
activation_12 (Activation)	(None, 224, 224, 16)	0
max_pooling2d_8 (MaxPooling2	(None, 112, 112, 16)	0
conv2d_9 (Conv2D)	(None, 112, 112, 32)	4640
batch_normalization_13 (Batc	(None, 112, 112, 32)	128
activation_13 (Activation)	(None, 112, 112, 32)	0
max_pooling2d_9 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_10 (Conv2D)	(None, 56, 56, 64)	18496
batch_normalization_14 (Batc	(None, 56, 56, 64)	256
activation_14 (Activation)	(None, 56, 56, 64)	0
max_pooling2d_10 (MaxPooling	(None, 28, 28, 64)	0
conv2d_11 (Conv2D)	(None, 28, 28, 128)	73856
batch_normalization_15 (Batc	(None, 28, 28, 128)	512
activation_15 (Activation)	(None, 28, 28, 128)	0
max_pooling2d_11 (MaxPooling	(None, 14, 14, 128)	0
flatten_2 (Flatten)	(None, 25088)	0
dense_4 (Dense)	(None, 128)	3211392
batch_normalization_16 (Batc	(None, 128)	512
activation_16 (Activation)	(None, 128)	0
dense_5 (Dense)	(None, 3)	387
batch_normalization_17 (Batc	(None, 3)	12
activation_17 (Activation)	(None, 3)	0

Total params: 3,310,703



We classified images into three types CLL, FL, MCL .we used convolutional neural network (CNN) of 4 layers containing input layer with shape (224*224*3) and pooling layers (2x2), activation function relu, softmax and one output layer

Number of epochs: 20

Train data size: 70%

Test data size: 5%

Validation data size: 25%

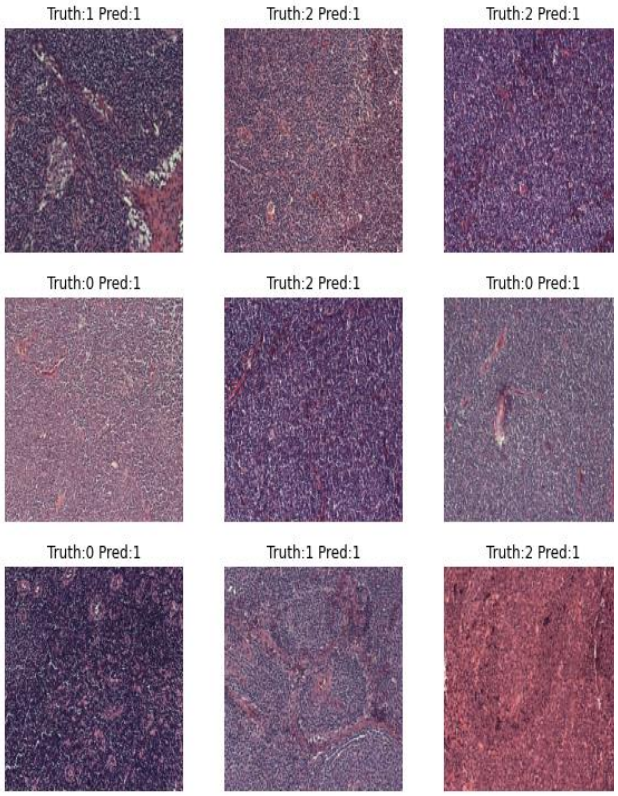
Figure 28 (Model summary and testing sample)

```
298/298 [==============================] - 27s 91ms/step - loss: 1.0948 - accuracy: 0.3725 - val_loss:
906.1590 - val_accuracy: 0.3750
Found 20 images belonging to 3 classes.
1/1 [==============================] - 0s 998us/step - loss: 797.2403 - accuracy: 0.5500
Test loss: 797.2402954101562
Test accuracy: 55.000001192092896 %
```

Trial 2 with augmentation and without slicing:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 16)	448
batch_normalization (BatchNo	(None, 224, 224, 16)	64
max_pooling2d (MaxPooling2D)	(None, 112, 112, 16)	0
conv2d_1 (Conv2D)	(None, 112, 112, 32)	4640
batch_normalization_1 (Batch	(None, 112, 112, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_2 (Conv2D)	(None, 56, 56, 64)	18496
batch_normalization_2 (Batch	(None, 56, 56, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 28, 28, 64)	0
conv2d_3 (Conv2D)	(None, 28, 28, 128)	73856
batch_normalization_3 (Batch	(None, 28, 28, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3211392
batch_normalization_4 (Batch	(None, 128)	512
dense_1 (Dense)	(None, 3)	387

Total params: 3,310,691



We classified images into three types CLL, FL, MCL .we used convolutional neural network (CNN) of 4 layers containing input layer with shape (224*224*3) and pooling layers (2x2), activation function relu, softmax and one output layer

Number of epochs: 20

Train data size: 70%

Test data size: 5%

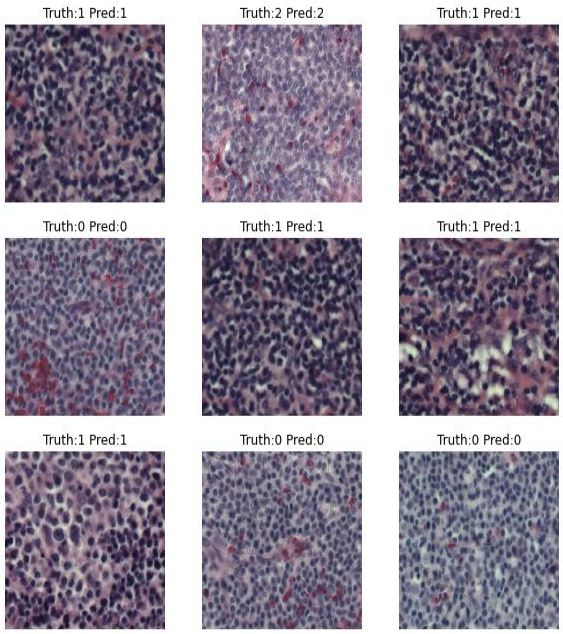
Validation data size: 25%

```
Epoch 20/20
18/18 [==============================] - 19s 1s/step - loss: 0.6067 - accuracy: 0.7411 - val_loss:
0.8880 - val_accuracy: 0.5833
Found 20 images belonging to 3 classes.
1/1 [==============================] - 0s 997us/step - loss: 1.0918 - accuracy: 0.5000
Test loss: 1.0917596817016602
Test accuracy: 50.0 %
```

Trial 3 with slicing and without augmentation:

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 224, 224, 16)	448
batch_normalization_10 (Batc	(None, 224, 224, 16)	64
max_pooling2d_8 (MaxPooling2	(None, 112, 112, 16)	0
conv2d_9 (Conv2D)	(None, 112, 112, 32)	4640
batch_normalization_11 (Batc	(None, 112, 112, 32)	128
max_pooling2d_9 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_10 (Conv2D)	(None, 56, 56, 64)	18496
batch_normalization_12 (Batc	(None, 56, 56, 64)	256
max_pooling2d_10 (MaxPooling	(None, 28, 28, 64)	0
conv2d_11 (Conv2D)	(None, 28, 28, 128)	73856
batch_normalization_13 (Batc	(None, 28, 28, 128)	512
max_pooling2d_11 (MaxPooling	(None, 14, 14, 128)	0
flatten_2 (Flatten)	(None, 25088)	0
dense_4 (Dense)	(None, 128)	3211392
batch_normalization_14 (Batc	(None, 128)	512
dense_5 (Dense)	(None, 3)	387

Total params: 3,310,691



We classified images into three types CLL, FL ,MCL .we used convolutional neural network(CNN) of 4 layers containing input layer with shape(224*224*3) and pooling layers(2x2),activation function relu, softmax and one output layer

Number of epochs: 20

Train data size: 70%

Test data size: 5%

Validation data size: 25%

```
Epoch 20/20
298/298 [==============================] - 301s 1s/step - loss: 0.0144 - accuracy: 0.9964 - val_loss:
0.3016 - val_accuracy: 0.9107
Found 320 images belonging to 3 classes.
10/10 [==============================] - 7s 670ms/step - loss: 0.2616 - accuracy: 0.9187
Test loss: 0.2616333067417145
Test accuracy: 91.8749988079071 %
```

Conclusion:

- Slicing make a huge impact on accuracy of training and testing
- So we used both augmentation to increase number of images and slicing to make features more visible to classify

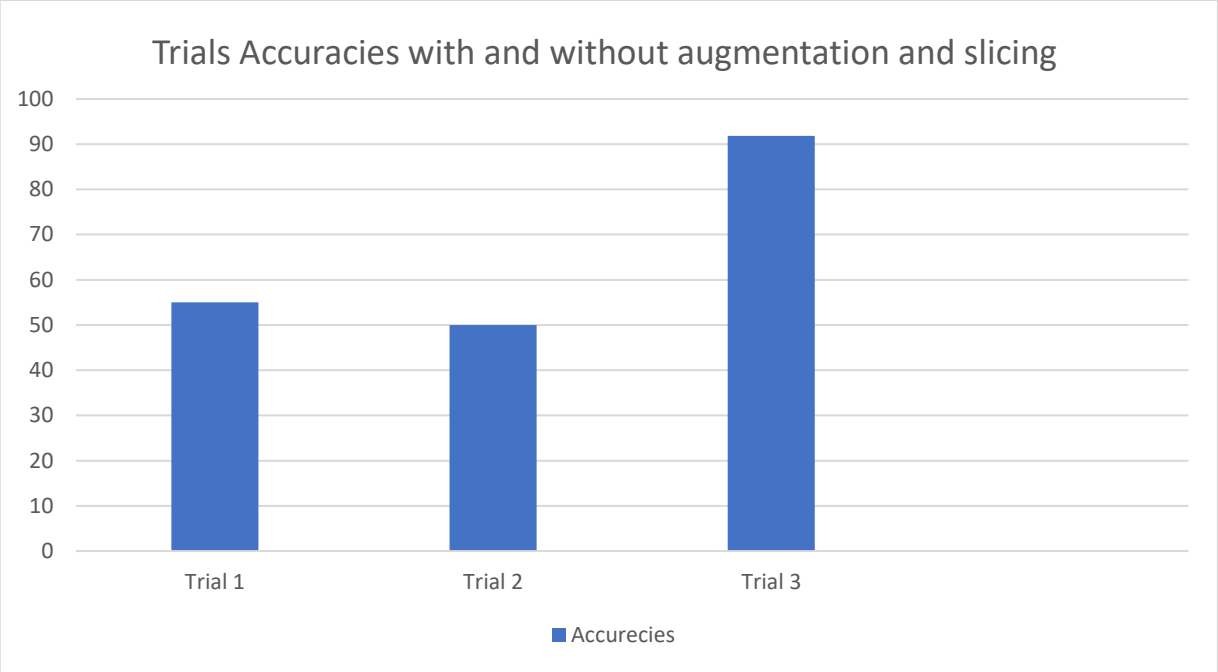
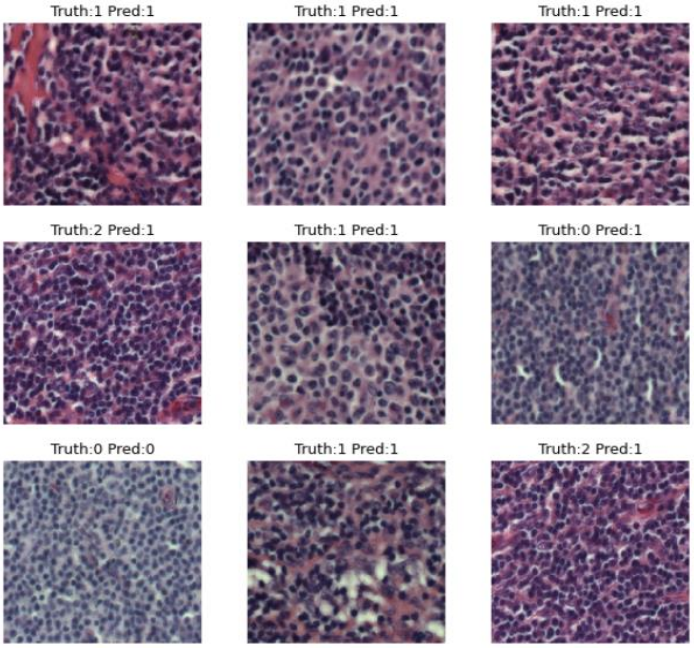


Figure 29 (Trials accuracies with and without augmentation and slicing)

Trial 1 with dropout

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 16)	448
dropout (Dropout)	(None, 224, 224, 16)	0
max_pooling2d (MaxPooling2D)	(None, 74, 74, 16)	0
conv2d_1 (Conv2D)	(None, 74, 74, 32)	4640
dropout_1 (Dropout)	(None, 74, 74, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 32)	0
conv2d_2 (Conv2D)	(None, 24, 24, 64)	18496
dropout_2 (Dropout)	(None, 24, 24, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 64)	0
conv2d_3 (Conv2D)	(None, 8, 8, 128)	73856
dropout_3 (Dropout)	(None, 8, 8, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 2, 2, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 128)	65664
dropout_4 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387

Total params: 163,491
Trainable params: 163,491
Non-trainable params: 0



We classified images into three types CLL, FL ,MCL .we used convolutional neural network(CNN) of 4 layers containing input layer with shape(224*224*3) and pooling layers(2x2),activation function relu, softmax and one output layer

Number of epochs: 20

Train data size: 70%

Test data size: 5%

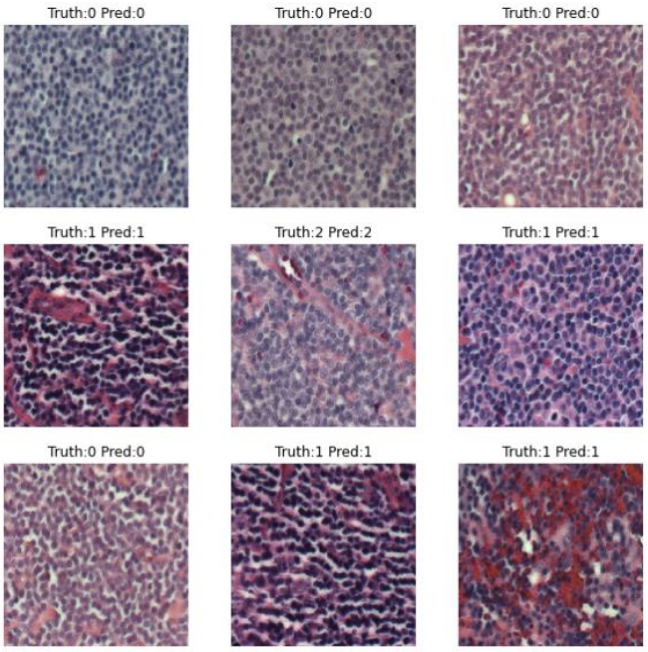
Validation data size: 25%

```
298/298 [==============================] - 194s 653ms/step - loss: 0.3082 - accuracy: 0.8643 -
val_loss: 0.5398 - val_accuracy: 0.7913
Found 320 images belonging to 3 classes.
10/10 [==============================] - 8s 791ms/step - loss: 0.4326 - accuracy: 0.8250
Test loss: 0.43257755041122437
Test accuracy: 82.4999988079071 %
```

Trial 2 without dropout:

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 224, 224, 16)	448
max_pooling2d_4 (MaxPooling2	(None, 112, 112, 16)	0
conv2d_5 (Conv2D)	(None, 112, 112, 32)	4640
max_pooling2d_5 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_6 (Conv2D)	(None, 56, 56, 64)	18496
max_pooling2d_6 (MaxPooling2	(None, 28, 28, 64)	0
conv2d_7 (Conv2D)	(None, 28, 28, 128)	73856
max_pooling2d_7 (MaxPooling2	(None, 14, 14, 128)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_2 (Dense)	(None, 128)	3211392
dense_3 (Dense)	(None, 3)	387

Total params: 3,309,219
Trainable params: 3,309,219
Non-trainable params: 0



We classified images into three types CLL, FL ,MCL .we used convolutional neural network(CNN) of 4 layers containing input layer with shape(224*224*3) and pooling layers(2x2),activation function relu, softmax and one output layer

Number of epochs: 20

Train data size: 70%

Test data size: 5%

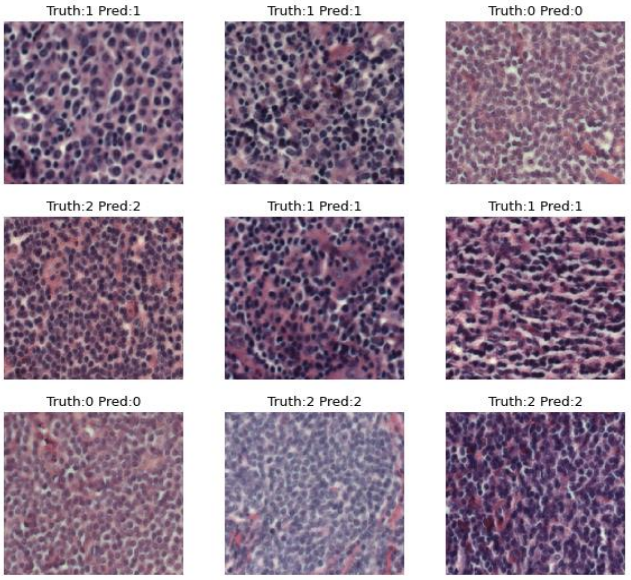
Validation data size: 25%

```
Epoch 20/20
298/298 [==============================] - 204s 683ms/step - loss: 0.2628 - accuracy: 0.8947 -
val_loss: 0.3729 - val_accuracy: 0.8326
Found 320 images belonging to 3 classes.
10/10 [==============================] - 6s 646ms/step - loss: 0.2054 - accuracy: 0.9187
Test loss: 0.20535901188850403
Test accuracy: 91.8749988079071 %
```

Trial 3 without dropout and with batch normalization:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 16)	448
batch_normalization (BatchNo	(None, 224, 224, 16)	64
max_pooling2d (MaxPooling2D)	(None, 112, 112, 16)	0
conv2d_1 (Conv2D)	(None, 112, 112, 32)	4640
batch_normalization_1 (Batch	(None, 112, 112, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_2 (Conv2D)	(None, 56, 56, 64)	18496
batch_normalization_2 (Batch	(None, 56, 56, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 28, 28, 64)	0
conv2d_3 (Conv2D)	(None, 28, 28, 128)	73856
batch_normalization_3 (Batch	(None, 28, 28, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3211392
batch_normalization_4 (Batch	(None, 128)	512
dense_1 (Dense)	(None, 3)	387

Total params: 3,310,691



We classified images into three types CLL, FL ,MCL .we used convolutional neural network(CNN) of 4 layers containing input layer with shape(224*224*3) and pooling layers(2x2),activation function relu, softmax and one output layer

Number of epochs: 20

Train data size: 70%

Test data size: 5%

Validation data size: 25%

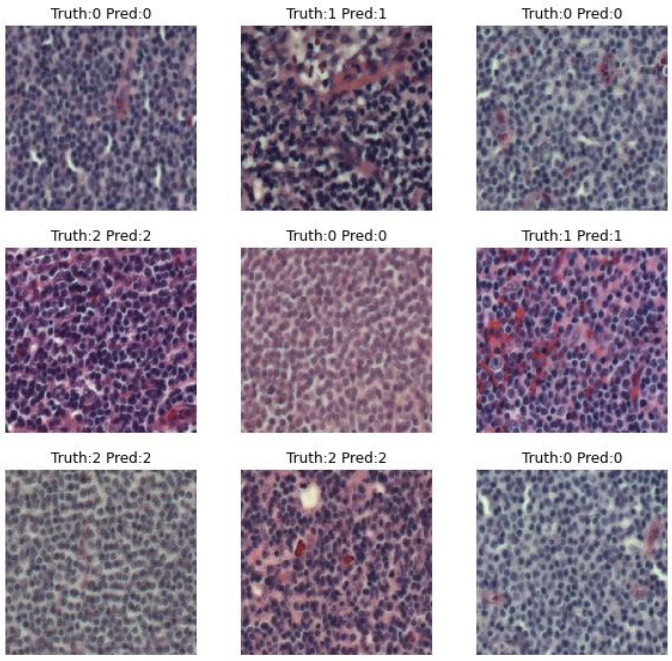
```
Epoch 20/20
298/298 [==============================] - 300s 1s/step - loss: 0.1355 - accuracy: 0.9501 - val_loss:
0.1786 - val_accuracy: 0.9263
Found 320 images belonging to 3 classes.
10/10 [==============================] - 7s 689ms/step - loss: 0.0483 - accuracy: 0.9844
Test loss: 0.0483328178524971
Test accuracy: 98.4375 %
```

Note: The best accuracy of all models

Trial 4 with dropout and batch normalization:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 16)	448
batch_normalization (BatchNo	(None, 224, 224, 16)	64
activation (Activation)	(None, 224, 224, 16)	0
dropout (Dropout)	(None, 224, 224, 16)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 16)	0
conv2d_1 (Conv2D)	(None, 112, 112, 32)	4640
batch_normalization_1 (Batch	(None, 112, 112, 32)	128
activation_1 (Activation)	(None, 112, 112, 32)	0
dropout_1 (Dropout)	(None, 112, 112, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_2 (Conv2D)	(None, 56, 56, 64)	18496
batch_normalization_2 (Batch	(None, 56, 56, 64)	256
activation_2 (Activation)	(None, 56, 56, 64)	0
dropout_2 (Dropout)	(None, 56, 56, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 28, 28, 64)	0
conv2d_3 (Conv2D)	(None, 28, 28, 128)	73856
batch_normalization_3 (Batch	(None, 28, 28, 128)	512
activation_3 (Activation)	(None, 28, 28, 128)	0
dropout_3 (Dropout)	(None, 28, 28, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3211392
batch_normalization_4 (Batch	(None, 128)	512
activation_4 (Activation)	(None, 128)	0
dropout_4 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387
batch_normalization_5 (Batch	(None, 3)	12
activation_5 (Activation)	(None, 3)	0

Total params: 3,310,703



We classified images into three types CLL, FL ,MCL .we used convolutional neural network(CNN) of 4 layers containing input layer with shape(224*224*3) and pooling layers(2x2),activation function relu, softmax and one output layer

Number of epochs: 20

Train data size: 70%

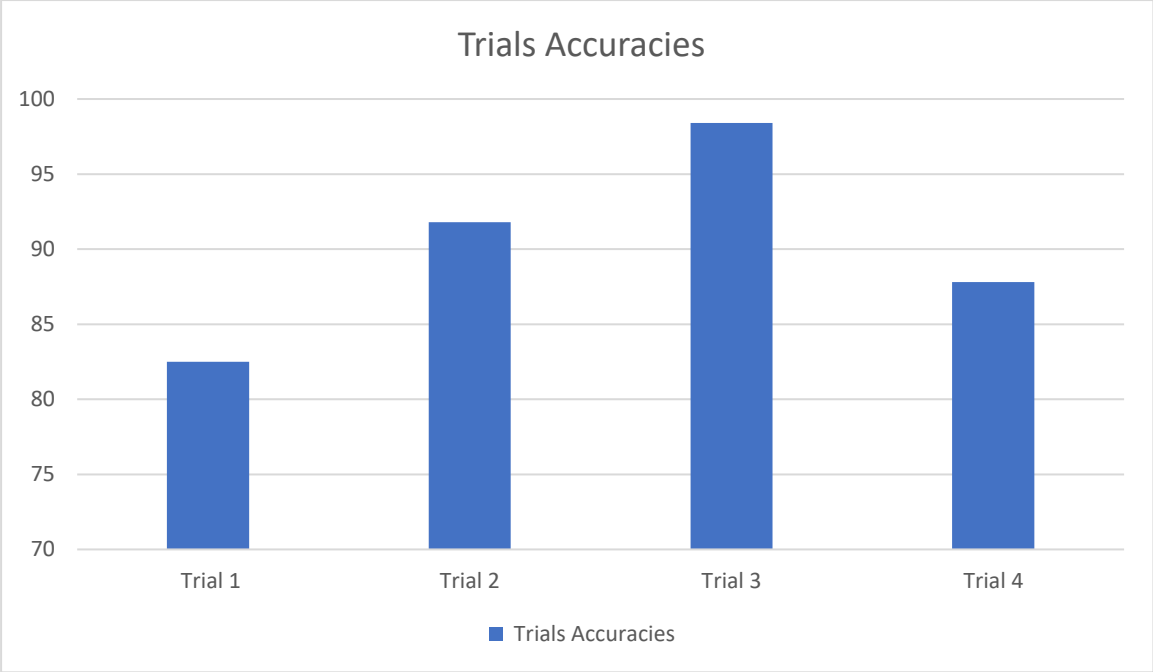
Test data size: 5%

Validation data size: 25%

```
Epoch 20/20
298/298 [==============================] - 361s 1s/step - loss: 0.2874 - accuracy: 0.9035 - val_loss:
0.6172 - val_accuracy: 0.7433
Found 320 images belonging to 3 classes.
10/10 [==============================] - 22s 2s/step - loss: 0.3894 - accuracy: 0.8781
Test loss: 0.3894427418708801
Test accuracy: 87.8125011920929 %
```

Conclusion

- Using dropout decreases the accuracy with this data
- Using dropout and batch normalization gives acceptable accuracy but not the best
- Using batch normalization only gives the best accuracy (98.4%)



Note: Trial 3 gives the best accuracy

Figure 30 (Trial accuracies with and without batch normalization and dropout)

Our Model (Trial 3)

```
def ourModel():
    model = Sequential()
    model.add(Conv2D(filters=16, kernel_size=(3,3), padding='same',
                     input_shape=(224, 224, 3)))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(filters=32, kernel_size=(3,3),
              padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(filters=64, kernel_size=(3,3),
              padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(filters=128, kernel_size=(3,3),
              padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size = (2,2)))

    model.add(Flatten())
    model.add(Dense(128))
    model.add(BatchNormalization())
    model.add(Activation('relu'))

    model.add(Dense(3))
    model.add(BatchNormalization())
    model.add(Activation('softmax'))

    model.compile(optimizer=Adam(0.0001),
                  loss='categorical_crossentropy', metrics=['accuracy'])
    model.summary()
    return model
```

Feature Maps


```
(1, 224, 224, 16)
(1, 112, 112, 16)
(1, 112, 112, 32)
(1, 112, 112, 32)
(1, 56, 56, 32)
(1, 56, 56, 64)
(1, 56, 56, 64)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:44: RuntimeWarning: divide by zero encountered in true_divide
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:44: RuntimeWarning: invalid value encountered in true_divide
(1, 28, 28, 64)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 14, 14, 128)
(1, 25088)
(1, 128)
(1, 128)
(1, 3)
```

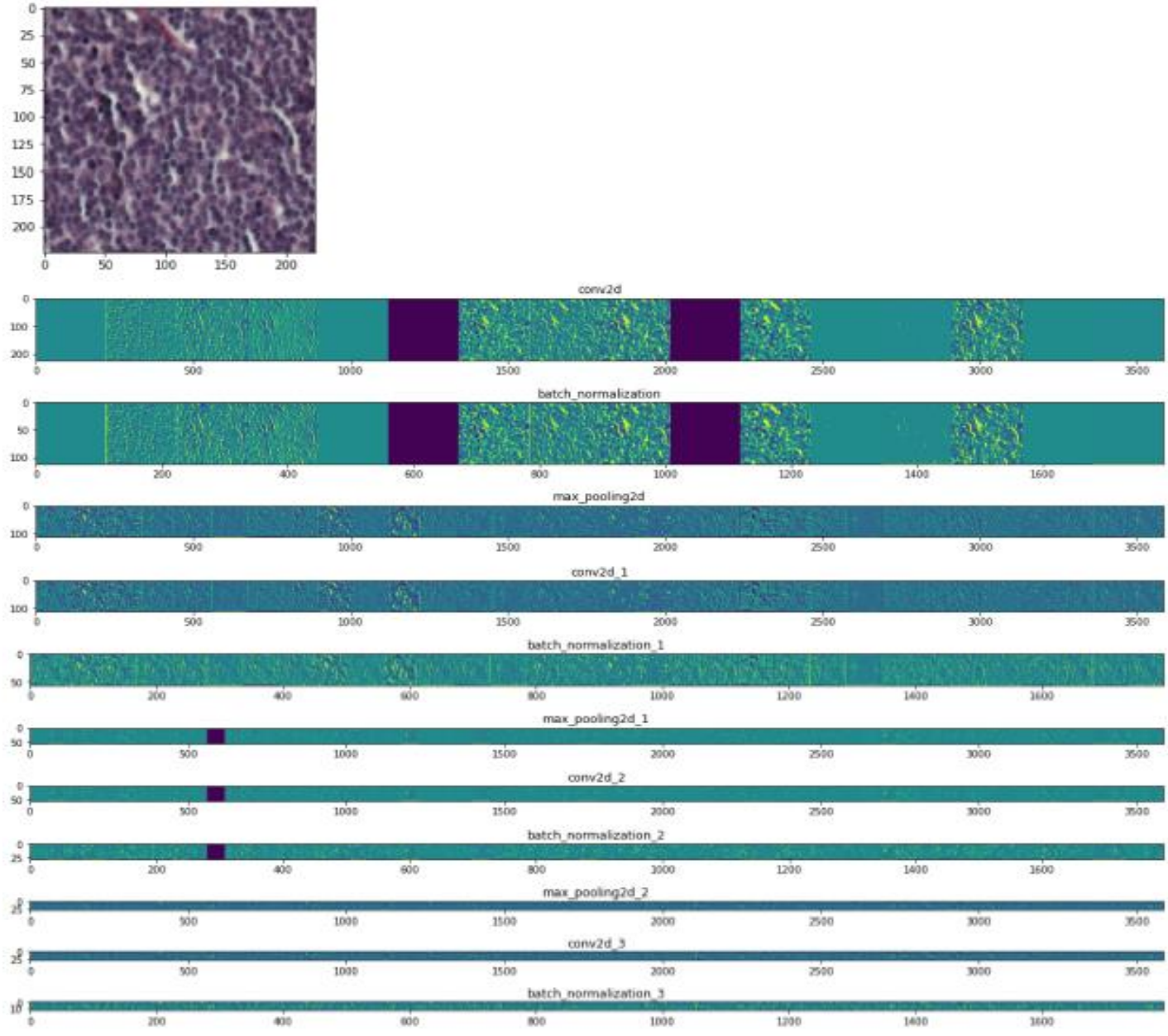


Figure 31 for classified sample CLL

```
(1, 224, 224, 16)
(1, 112, 112, 16)
(1, 112, 112, 32)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:44: RuntimeWarning: divide
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:44: RuntimeWarning: invali
(1, 112, 112, 32)
(1, 56, 56, 32)
(1, 56, 56, 64)
(1, 56, 56, 64)
(1, 28, 28, 64)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 14, 14, 128)
(1, 25088)
(1, 128)
(1, 128)
(1, 3)
```

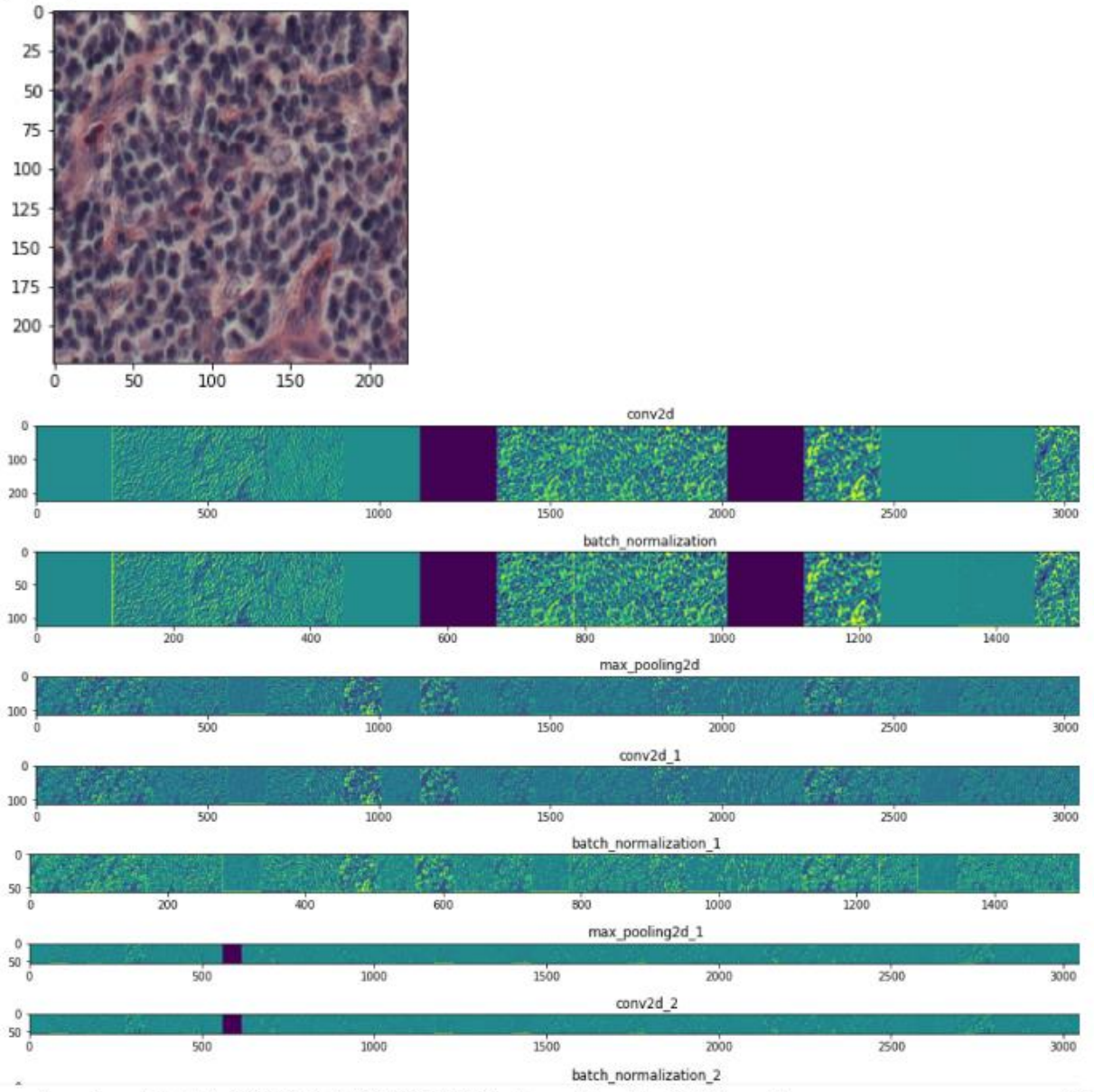


Figure 32 for classified sample (MCL)

```
(1, 224, 224, 16)
(1, 112, 112, 16)
(1, 112, 112, 32)
(1, 112, 112, 32)
(1, 56, 56, 32)
(1, 56, 56, 64)
(1, 56, 56, 64)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:44: RuntimeWarning: divide
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:44: RuntimeWarning: invali
(1, 28, 28, 64)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 14, 14, 128)
(1, 25088)
(1, 128)
(1, 128)
(1, 3)
```

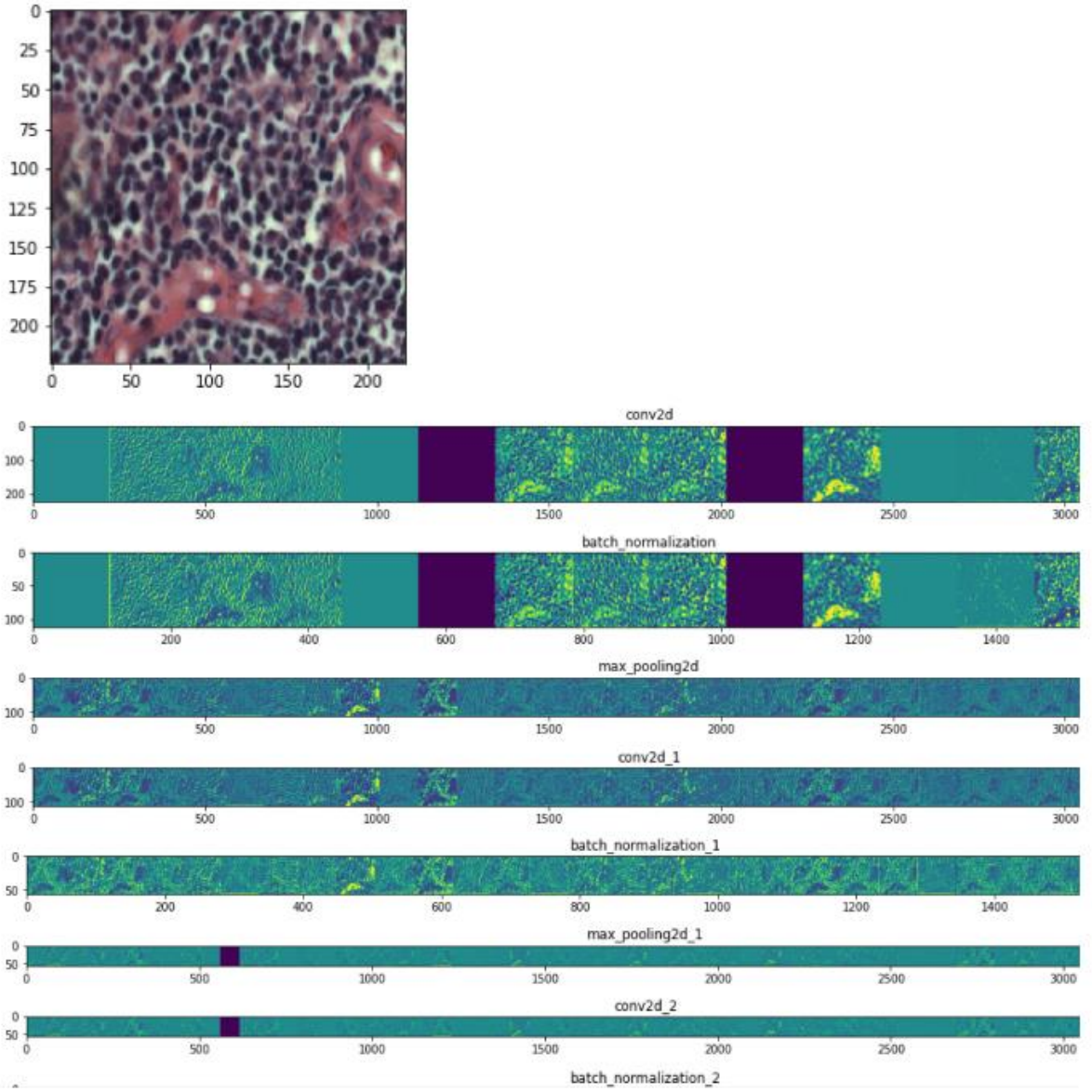


Figure 33 for classified sample (FL)

Chapter 7: Conclusion and Future work

7.1 Conclusion:

Critically, the development of inexpensive point-of-care diagnostic platforms will accelerate the discovery of 3 types of Non-Hodgkin Lymphoma Cancer, especially in countries with health systems devoid of adequate laboratory infrastructure, also enriching the scientific research related to this specific type of cancer by providing illustrative feature maps that helps to precise understanding the causes of this disease.

Since the most common treatments of this disease are Chemotherapy and radiotherapy, those are sharp & may cause severe side effects that needs accurate procedures & diagnosis

Our most important take-away is that having scarce amount of data doesn't necessarily mean a weak model, furthermore, the feature-rich data, which is data with high frequency for its distinct/desirable features offsets for the scarcity of our data, that helped us to build a CNN Model, train on the dataset, and be able to classify the images to one of the three Non-Hodgkin Lymphoma Cancer.

7.2 Future Work

- Expanding the classes of our model to more than only 3 types of Non-Hodgkin Lymphoma Types

- Online host our website, so that it is available for world-wide use

- Use the data we are collecting from either local storage, or after hosting our website, to create a database for the Non-Hodgkin Lymphoma Cancer

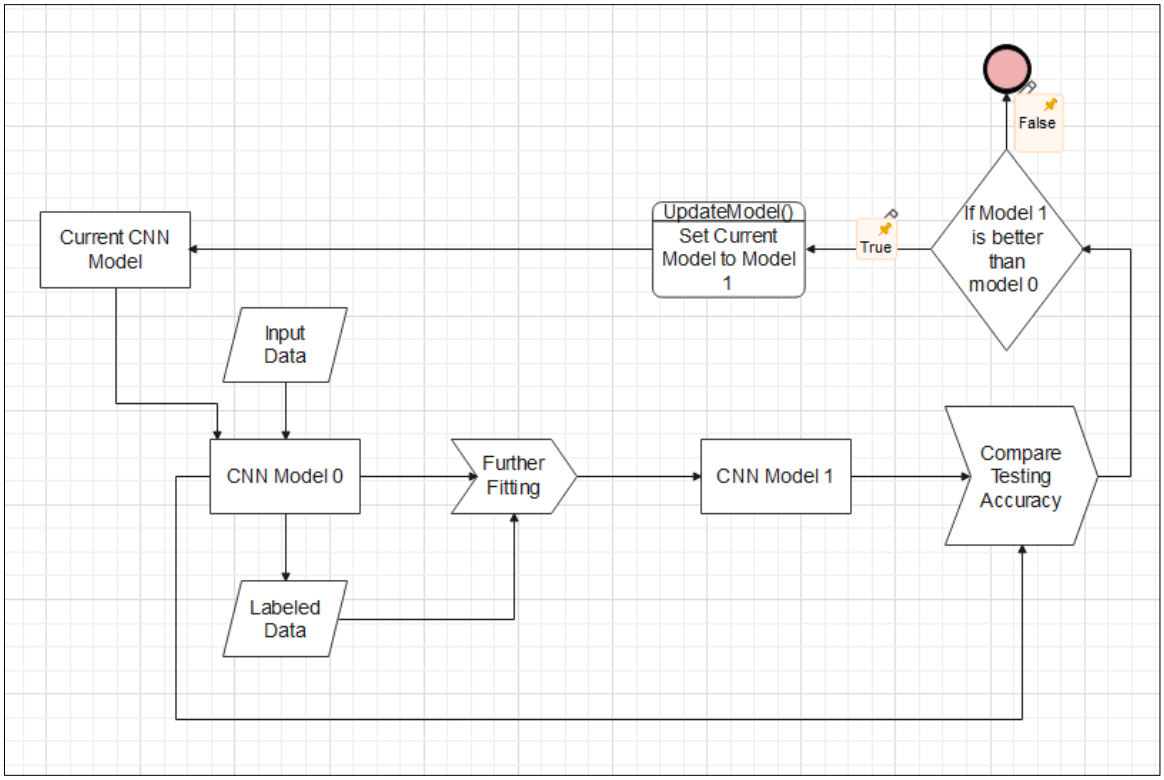


Figure 34 Future work diagram

7.3 Technical Obstacles

It was challenging at first to find a suitable dataset for the specific types of the Non-Hodgkin Lymphoma Cancer, we searched popular datasets websites like Kaggle, and we found a dataset that was published by National Institute of Ageing,

Located in Bethesda, Maryland, the dataset had 374 image that was specific to 3 types of NHL Cancer.

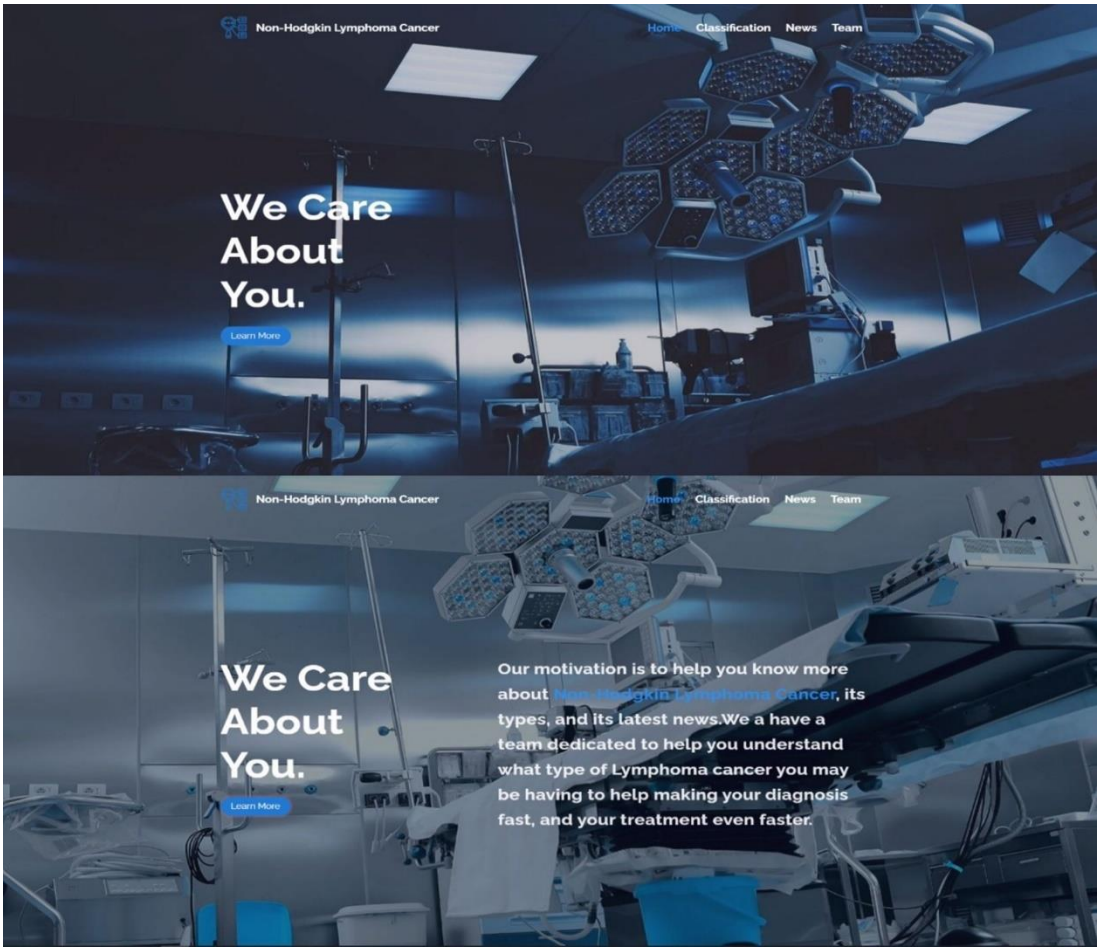
374 Image was not enough for our CNN Model to train effectively, the data was scarce on the internet and our dataset wasn't enough, we had to search for some artificial way to generate images that we can use as data input to our model (Slicing & Data Augmentation)

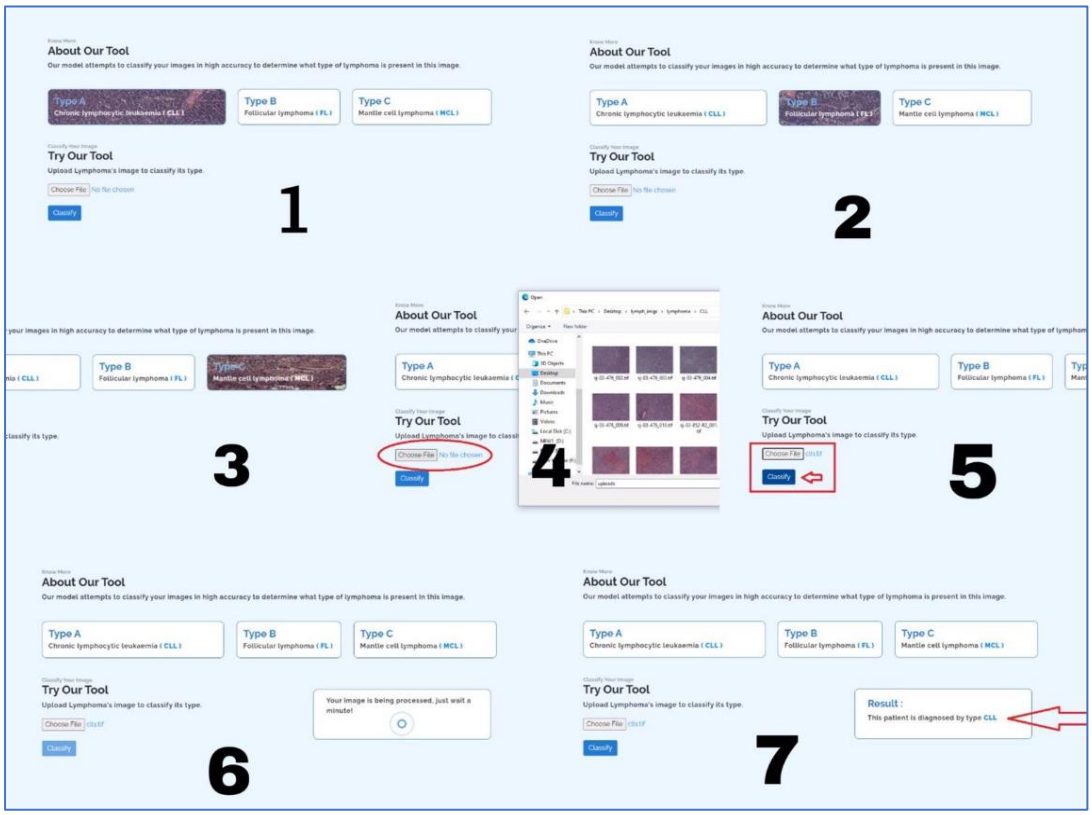
The accuracy of the model itself was still low, so we had to use some techniques to increase the accuracy like Batch normalization & Dropout.

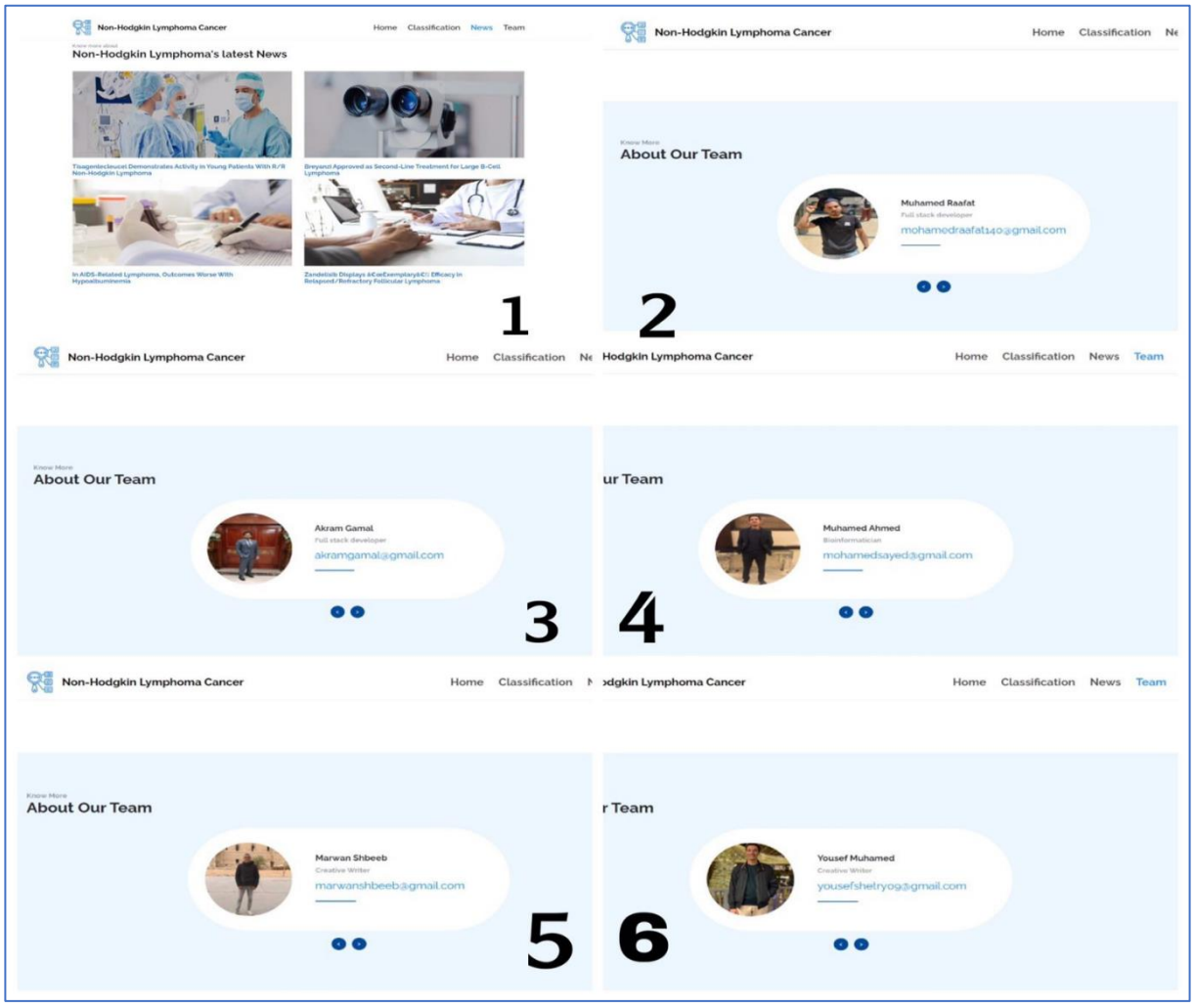
Dropout caused worse results in the accuracy, So we suffice for Batch Normalization.

After slicing/segmentation to the images, the amount of data provided by this technique will be excessive & requiring a powerful computational power, where the data will be approximately 6000 images, Traditional machine learning techniques can be hard to process this amount of data on a normal notebook laptop.

7.4 Web Application Snaps






Non-Hodgkin Lymphoma Cancer
Home Classification News Team
Non-Hodgkin Lymphoma's latest News
1
Non-Hodgkin Lymphoma Cancer
Home Classification Ne
About Our Team
Muhamed Raafat
mohamedraafat140@gmail.com
2
Non-Hodgkin Lymphoma Cancer
Home Classification Ne
Hodgkin Lymphoma Cancer
Home Classification News Team
About Our Team
Akram Gamal
akramgamal@gmail.com
3
ur Team
Muhamed Ahmed
mohamedsayed@gmail.com
4
Non-Hodgkin Lymphoma Cancer
Home Classification
odgkin Lymphoma Cancer
Home Classification News Team
About Our Team
Marwan Shbeeb
marwanshbeeb@gmail.com
5
r Team
Yousef Muhamed
yousefshetryo9@gmail.com
6


Cancer is no more stubborn after today, we will do our best to help keep you safe, because
We Care About You.
Copyright Â© 2022 All Rights Reserved.

Project's Poster

Malignant Lymphoma *Classification*

Marwan[a], Mohamed[r], Akram[g], Yousef[m] , Mohamed[s]

Supervisor: Dr. Amin & Dr. Heba

Abstract

Being able to determine the specific subtype of cancer is the first step towards treatment, as cancer treatment relies completely on early diagnosis of the cancer type to help control it in its early stages. Blood cancer types are the most spreading ones.

Previous studies have provided evidence for automatic cancer tissue analysis by using deep learning strategies that retrieve and organize discriminating insights from the images automatically. Therefore, in this study an innovative and empowered deep learning.

framework is proposed to classify three types of lymphoma as Follicular Lymphoma (FL), Chronic Lymphocytic Lymphoma (CLL) and Mantle Cell Lymphoma (MCL). Which are the most common three types of non-Hodgkin Lymphoma cancer. In this research, we developed and advanced CNN model that efficiently predicts the type of non-Hodgkin Lymphoma with an accuracy of 98%.

A publicly available dataset from National Institute of Ageing (NIA) is used in this study. This study performs the histogram normalization on all the images to enhance the performance of model. The data augmentation has been carried out on the dataset so that overfitting can be avoided, we also segmented the images to get higher accuracy..

Introduction

Lymphoma is a blood disease (cancer) that develops in the immune cells, which are the cells that mainly defend our bodies from infections and viruses. Thus, a vulnerable body with no defense system requires attention from researchers and machine learning specialists to help eliminate it, Because lymphocytes have physiologic immune functions that vary both by lineage and by stage of differentiation, the classification of lymphomas arising from these normal lymphoid populations is complex

Lymphoma is mainly regarded as: Hodgkin Lymphoma and non-Hodgkin Lymphoma. Having only two developing sorts or Lymphoma by high rate gives us the opportunity to focus and find a suitable solution for both of them. Lymphomas types are classified based on the normal counterpart, or cell of origin, from which they arise.

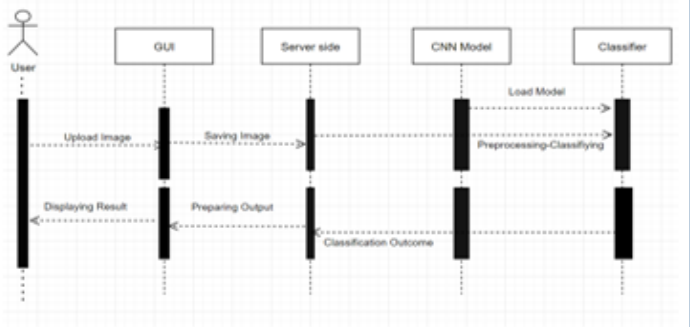
We decided to focus on the most common three types of non-Hodgkin lymphoma types among cancer patients, which are: CLL (Chronic lymphocytic leukemia): is a type of cancer of the blood and the bone marrow, MCL (Mantle cell lymphoma): which is a cancer type that make B cells cancerous, FL (Follicular lymphoma): which is a cancer type that involves certain types of white blood cells known as lymphocytes.

CLL FL MCL

Methods

- Machine Learning Technique (Traditional Approach)
  - -SVM
  - -Logistic Regression
  - -Decision Tree
  - -KNN
- Deep Learning Algorithm
  - -Convolutional Neural Network (CNN)
- Google Collaboratory
- Python Programming Language & Libraries (Tensorflow, Numpy, Keras)
- HTML,CSS & JavaScript & Bootstrap
- JQuery & Ajax
- PHP with execution of Python script using exec() Method

Sequence Diagram



Conclusion

Critically, the development of inexpensive point-of-care diagnostic platforms will accelerate the discovery of 3 types of Non-Hodgkin Lymphoma Cancer, especially in countries with health systems devoid of adequate laboratory infrastructure, also enriching the scientific research related to this specific type of cancer by providing illustrative feature maps that helps to precise understanding the causes of this disease.

Since the most common treatments of this disease are Chemotherapy and radiotherapy, those are sharp & may cause severe side effects that needs accurate procedures & diagnosis

Our most important take-away is that having scarce amount of data doesn't necessarily mean a weak model, furthermore, the feature-rich data, which is data with high frequency for its distinct/desirable features offsets for the scarcity of our data, that helped us to build a CNN Model , train on the dataset , and be able to classify the images to one of the three Non-Hodgkin Lymphoma Cancer.

Authors

- amarawan56@gmail.com
- akramgamall14@gmail.com
- mohamedraafat140@gmail.com
- yousefshetry09@gmail.com
- mohamedsaied8818@gmail.com

References

https://mdpi-res.com/d_attachment/cancers/cancers-13-02419/article_deploy/cancers-13-02419.pdf

Deep Learning for the Classification of Non-Hodgkin Lymphomaon Histopathological Images (nih.gov)

Deep Learning for the Classification of Non-Hodgkin Lymphomaon Histopathological Images - PubMed (nih.gov)

A deep learning diagnostic platform for diffuse large B-cell lymphoma with high accuracy across multiple hospitals | NatureCommunications

(PDF) Deep Learning for the Classification of Non-Hodgkin Lymphoma on Histopathological Images (researchgate.net)

Cancers | Free Full-Text | Machine Learning Based on Morphological Features Enables

Classification of PrimaryIntestinal T-Cell Lymphomas | HTML (mdpi.com)

https://books.google.com/books?hl=en&lr=&id=G86zMVD3yNYC&oi=fnd&pg=PA1&dq=CNN&ots=coG0X3A3Gz&sig=gj5NPvNuHk0FsBFDZoHKLr-ESZM

http://openaccess.thecvf.com/content_CVPR_2020/html/Wang_CNN-Generated_Images_Are_Surprisingly_Easy_to_Spot..._for_Now_CVPR_2020_paper.html