A Novel AI based prognostic approach using histopathology images for Hodgkin Lymphoma

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

Submitted by
Rakshit Rohan

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

Under supervision of
Supervisor: Dr. Kshitij Jadhav (IITB)
TAC Member: Dr. Anindiya Goswami (IISER Pune)

Certificate

This is to certify that this dissertation entitled **A Novel AI based prognostic approach using histopathology images for Hodgkin Lymphoma** towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Rakshit Rohan at Koita Centre for Digital Health, IIT Bombay under the supervision of Dr. Kshitij Jadhav, Professor IIT Bombay, Koita Centre for Digital Health , during the academic year 2023-2024.

Dr. Kshitij Jadhav

Committee:

Supervisor: Dr. Kshitij Jadhav

TAC Member: Dr. Anindiya Goswami

This thesis is dedicated to my parents

I pray for their happiness and well-being

Declaration

I hereby declare that the matter embodied in the report entitled A Novel AI based prognostic approach using histopathology images for Hodgkin Lymphoma are the results of the work carried out by me at the Koita Centre for Digital Health, Indian Institute of Technology (IIT), Bombay, under the supervision of Dr. Kshitij Jadhav and the same has not been submitted elsewhere for any other degree. Wherever others contribute, every effort is made to indicate this clearly, with due reference to the literature and acknowledgement of collaborative research and discussions.

Rakshit Rohan
Roll 20191171

Acknowledgments

I am extremely thankful to Dr. Kshitij Jadhav for giving me an opportunity to work with him on this project. His guidance and his constant source of encouragement and support have helped me throughout the period of my project. His input always helped me and showed me a way out whenever I got stuck on any problem that came up in the thesis. Along with him, I would also like to take this opportunity to thank all the members of the Koita Digital Center for their helpful advice in times of need.

I would also like to thank my friends. They were the ones who showed me the way forward whenever I felt lost and confused about what to do next.

Finally, I thank my parents and brother for the constant moral support that I needed during this period of the project.

Abstract

Hodgkin Lymphoma is a type of tumor originating from a particular white blood cell type called lymphocyte. It is marked by the presence of multinucleated Reed-Sternberg cells (RS cells) which are found in the lymph nodes of the patient. It accounted for 0.5% of all the newly reported cancer cases and 0.1% of all cancer-related deaths in 2023, which makes it a rare cancer type, but still is very relevant for study.

This study applied various approaches on histopathological images to identify RS cells along with the microenvironment cells in order to predict whether the patient is suffering from Hodgkin's Lymphoma and also predict the type of microenvironment for the patient. The data was obtained from Tata Memorial Hospital, Mumbai.

The study initially predicted the results on the Hodgkins dataset after getting trained on the online histopathological open-source dataset. Due to issues related to training on an open-source dataset, it was finally trained on a manually labeled dataset annotated by an expert annotator who individually annotated the patches from the whole slide images to obtain the required cells.

Therefore, the study describes an attempt to create a pipeline in order to automate the process of segmentation and classification of cells in a given Whole Slide Image to predict if the patient is at an advanced stage of Hodgkin's Lymphoma and also predict the tissue microenvironment.

Contents

Abstract vi

1 Introduction 1

1.1 Contribution and Thesis Structure . . . 2

2 Preliminaries 4

2.1 Deep Learning . . . 4

2.2 Deep Neural Networks . . . 5

2.2.1 Feed Forward Neural Networks . . . 5

2.2.2 Convolutional Neural Networks . . . 6

2.3 Activation Functions . . . 8

2.4 Loss Functions . . . 10

2.5 Transformers . . . 13

2.6 Vision Transformers . . . 14

2.7 Foundation Models . . . 16

3 Literature Review 17

3.1 Competition Datasets . . . 18

3.1.1 MoNuSac . . . 18

3.1.2 MoNuSeg . . . 18

3.1.3 PanNuke . . . 18

3.1.4 Conic . . . 19

3.2 U-Net . . . 20

3.3 Stardist . . . 21

3.4 Segment Anything Model . . . 22

4 Data and Methodology **24**
4.1 Data Preparation 24
4.2 Methodology 25
4.2.1 Segmentation 25
4.2.2 Classification 26

5 Results and Discussion **28**
5.1 Stardist Results 28
5.2 Segment Anything Model 32
5.2.1 Background Noise Classifier 33
5.2.2 Cell Classifier 35

6 Conclusion **39**
6.1 Future Work 40

List of Figures

2.1 Feed Forward Neural Network 6
2.2 Convolutional Neural Network Source [1] 7
2.3 Activation functions 10
2.4 Loss functions 12
2.5 Transformer Architecture, Source [2] 14
2.6 Vision Transformer Architecture, Source [3] 15

3.1 MoNuSac Dataset, Source [4] 18
3.2 MoNuSeg Dataset, Source [5] 19
3.3 PanNuke Dataset, Source [6] 19
3.4 Conic Dataset, Source [7] 20
3.5 U-Net Architecture, Source [8] 21
3.6 Stardist Architecture, Source [9] 22
3.7 SAM Architecture, Source [10] 23

4.1 Different zoom levels in Whole Slide Image 25

5.1 Mask created using Stardist 29
5.2 Stardist trained on Conic dataset 30
5.3 Stardist Conic Implementation (HL) 31
5.4 Stardist trained on Conic dataset (Plasma only) 31
5.5 Stardist Conic Implementation (Plasma Only) 32
5.6 SAM implemented on an Image Patch 33
5.7 Cells of Interest 35

List of Tables

5.1 Stardist Conic Implementation 29
5.2 Cell Count Stardist (Figure 5.3) 30
5.3 Stardist Conic Implementation (Plasma only) 31
5.4 Background Classification Results 34
5.5 Background Classification Results (Undersampling) 34
5.6 Background Classification Results (Oversampling) 35
5.7 CNN Cell Classifier 36
5.8 CNN Cell Classifier (Balanced) 36
5.9 CNN Cell Classifier (Oversampled) 37
5.10 CNN Cell Classifier (Balanced and Oversampled) 37

Chapter 1

Introduction

Cancer cell study has been one of the most important studies in the medical field. There are many types of cancers and each one requires their own study in a detailed setting. Hodgkin Lymphoma is a type of cancer of lymphoma (group of blood and lymph nodes), where Reed Sternberg cells are formed from the lymphocytes which are a part of the patient's lymph nodes. It is a rare type of cancer, as out of all new cancer cases it accounted for 0.5% of all cancer-related cases and 0.1% of all cancer-related deaths in 2023.

The first and foremost step for the cancer cell study is obtaining data for the study. It is often challenging for medical cases, especially life-threatening diseases such as cancer. The patient data can be obtained in many ways. In our case, for the diagnosis of Hodgkin Lymphoma digitally scanned slides of HnE (Hemotaxylin and Eosin) staining is used. These stains mark the tissue microenvironment (TME) in a way that the pathologist is able to identify the condition of the patient using the cell count of various cells present in the TME.

In our case, we are digitally using these slides to make the predictions. To do this, the glass slides on which staining is done are digitally scanned using special machines that can provide the required glass slide data in high-resolution digital files. Whole Slide Imaging is the creation of digital scans from the glass slides and images thus produced are called Whole Slide Imaging and the digital images formed are called Whole Slide Images. These WSIs are scanned at different zoom levels therefore they have the ability to provide the data digitally at different zoom levels as well. They can be found in the magnification levels of the order 10x, 20x, and 40x (i.e. 10 times zoom, 20 times zoom, and 40 times zoom respectively).

There are various approaches that can be used to study the information present in the WSI.

Two of the most conventional ways to study are to directly study the information at the level of WSI (of the resolution 50,000x50,000 pixels) or divide the complete WSI into small patches that can be studied individually. It depends on the computational power that one has as well as the aim of the study to decide which procedure to follow.

The tumor's microenvironment (TME) is known by identifying the cells that are present around the tumor in the tissue. Identification of the tumor microenvironment is a very important step as it gives us information about the tumor initiation, development, invasion, and possible cure. Therefore, the microenvironment identification task was also considered by us along with the segmentation and the identification of the tumor cells.

Main aim of our study requires us to automate the process of counting number of Reed Sternberg (RS) cells as well as the microenvironment cells (major cells except RS cells present in the WSI). Counting of the RS cells gives information about the presence of an advanced stage of HL in the patient. Also, if the tissue microenvironment cells are counted then it can help further subclassify the kind or HL the person is suffering from.

This activity of counting the RS as well as the TME cells can be performed by a pathologist as well, but it takes a large amount of time for a repetitive task, hence this pipeline saves a lot of precious time as well as resources which can be employed on further important tasks. This procedure also increases the efficiency with which the WSI's are marked because once the algorithm is ready it can be applied to multiple WSI's at once to count and provide prediction of the results. Therefore, this is a novel problem that implements a prognostic approach to identify if the patient is suffering from Hodgkin's Lymphoma.

1.1 Contribution and Thesis Structure

The recent advances in the deep learning and image processing domain allow us to implement significant and various new approaches in order to count the number of RS cells along with the TME cells in the WSI's. The approach that we are trying to solve this problem allows us to divide the task into 2 parts: Segmentation of the cells of the patches produced from the WSIs and the classification of the cells in the patches thus produced to count each cell type.

The thesis is written and organized into the following parts: Chapter 2 provides important background theory to understand the problem as well as approaches taken in the thesis. Chapter 3 provides a literature review of the important papers, models, and datasets used in

this thesis. Chapter 4 discusses the dataset used, manipulation of the data, and methodology followed . Chapter 5 consists of results and discussions. Chapter 6 covers the conclusion as well as future work of the study.

Chapter 2

Preliminaries

This chapter discusses the basic theories of the concepts that were implemented in this thesis. These topics cover the most essential basics to be able to understand the main problem proposed in the thesis as well as the various approaches that were followed to address the problem. A lot of concepts were borrowed from the book "Dive into Deep Learning" [11].

2.1 Deep Learning

It is a subset of machine learning that is based on neural network architecture to provide predictions. DL has the capability of learning data patterns and relationships that can be very complex. The growth in the branch DL has been very recent and monumental, some reasons for its success can be attributed to its ability to correctly classify and predict images, text, speech, and time series. DL has also shown significant results in the fields of healthcare problems, satellite imaging, and gaming industries.

DL is a subfield of machine learning and we see find the term learning is common in them both. Learning here refers to the reassignments of mapping between the input and output features in order to provide the desired results. DL is different from machine learning in its approach of learning the features of the data. In DL layers of data is learned through a model called as neural networks which are based on the representation of human brain in order to compute and predict results.

Every DL algorithm follows some main steps which in order to yield results. First the data is received from an external source, and it is passed to the input layer of the neural network.

This input layer passes it to the hidden layer, that is the second layer. This information transfer happens through the neurons present in the previous layer. These neurons are the nodes through which data flows. The connection between two neurons are weighted and hence information from the previous layer neurons are enhanced when they reach the neurons of the next layer. This process continues till the final layer and produces the output.

A loss function calculates the difference between the target and the output produced by the model. The loss calculated serves as a way to modify the weights in the network which using an optimizer which updates the weights in the network. The algorithm used for backtracking the weights used in any iteration is called backpropagation algorithm.

One iteration of training refers to all the above steps happening once and the weights of the model getting updated once. This process is repeated multiple times until there is convergence in the values of loss function. Once the convergence is achieved, it is said that the DL model has been trained.

2.2 Deep Neural Networks

Neural networks containing multiple layers of differentiable functions are deep neural networks. These functions of multiple layers are the reason they are called deep. Few of the important neural networks that are relevant to the thesis are explained below:

2.2.1 Feed Forward Neural Networks

They are the type of neural networks that are categorized by their flow that is unidirectional, meaning the information in model flows only in one direction, i.e. forward. The main components that they contain are - input layer, multiple hidden layers, and then the output layers. Here, hidden layers further have weights and biases associated with them. The training is done through the backpropagation method where these weights are learned. Mathematically they can be described as: The activation at this layer is given by: $\sigma(.)$ is called the activation function. It is applied on the values passing from a hidden layer. It is described in detail later.

The hidden layer is given by:

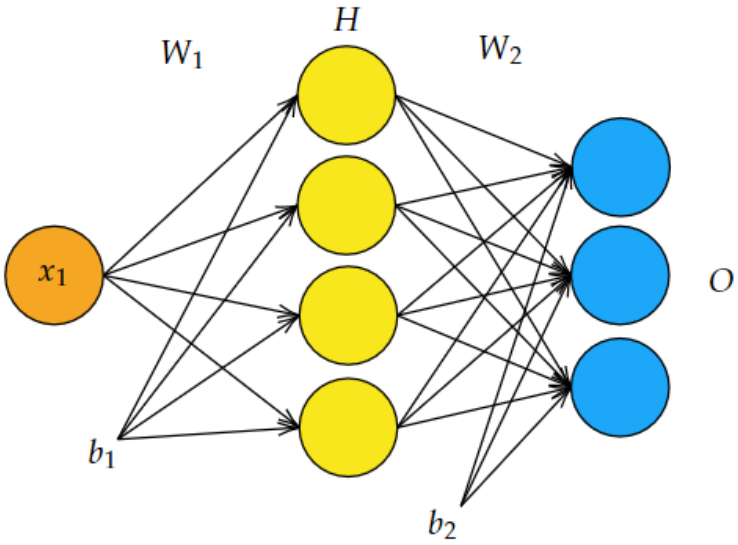


Figure 2.1: Feed Forward Neural Network

$$H = XW^{(1)} + b^{(1)}$$

$$H^{(1)} = \sigma_1(XW^{(1)} + b^{(1)})$$

Here, H has h hidden units and n is the number of examples, so $H \in \mathbb{R}^{n \times h}$. X is the number of examples that were taken so $X \in \mathbb{R}^{n \times d}$, where d is the features of inputs. b is the bias and is represented by $b \in \mathbb{R}^{1 \times q}$.

The output layer is given by:

$$O = HW^{(2)} + b^{(2)}$$

Here, O is the output of 1 hidden layer of feed-forward NN. It is given by, $O \in \mathbb{R}^{n \times q}$ where q are the features after passing through 1 hidden layer. $W^{(2)} \in \mathbb{R}^{h \times q}$ and $b^{(2)} \in \mathbb{R}^{1 \times q}$. In case of multiple stacking of hidden layers, $H^{(1)} = \sigma_1(XW^{(1)} + b^{(1)})$ and $H^{(2)} = \sigma_2(H^{(1)}W^{(2)} + b(2)$, and it will keep on going where σ_1 and σ_2 are 2 different activation functions.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are neural networks that are used to extract features from a matrix dataset. They work really well in detecting patterns and features in images,

audio, and video datasets.

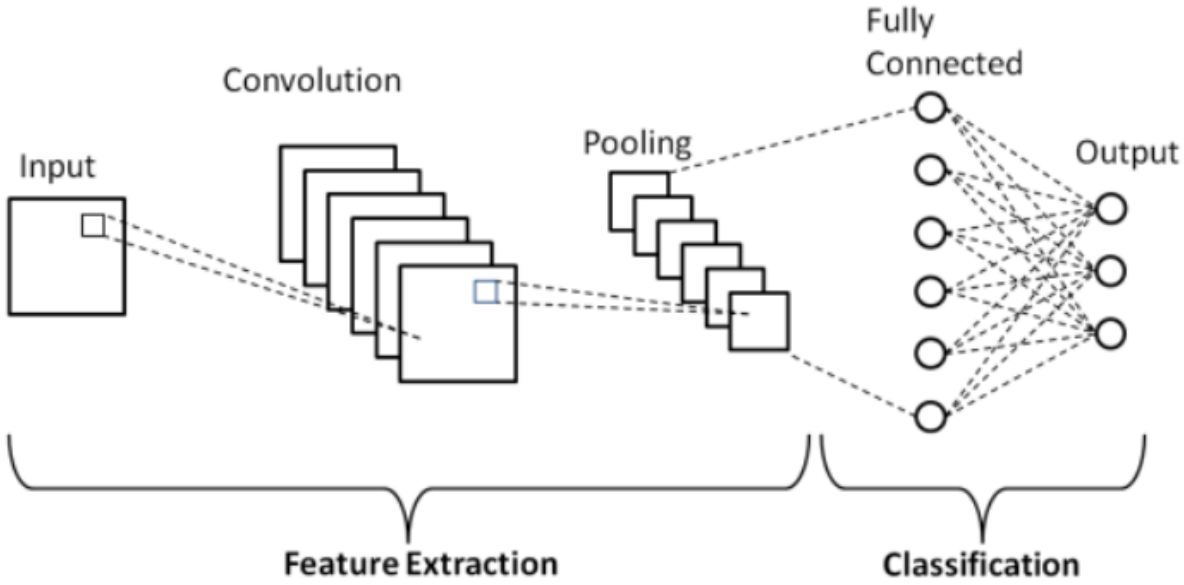


Figure 2.2: Convolutional Neural Network
Source [1]

The general working of CNN can described as follows:

$$O_{ij} = (I * W)_{ij} = \sum_{a=0}^{m-1} \sum_{n=0}^{n-1} I_{i+a,j+b} W_{a,b}$$

Here, I is the input matrix and W is the weight matrix. The shape of the outputs when convolutions are applied are as follows

$$W_f = \frac{W_I - F + 2P}{S} + 1$$

$$H_f = \frac{H_I - F + 2P}{S} + 1$$

$$D_f = K$$

here, F: spatial bound, K: number of filters, S: stride, K: number of filters

CNNs apply a series of pooling and convolution layers that extract the features from input and then the pooling layers downsample the output of convolution layers to reduce the dimensionality of the data, increasing the computational efficiency.

2.3 Activation Functions

They form an important component of neural networks. They are the ones deciding whether the neuron should be activated or not which gets calculated by the weighted sum of inputs and adding bias to this value. Some of the most commonly used activation functions are:

Sigmoid

This activation function that takes values from all real numbers but gives output between 0 and 1. Since the values are confined between 0 and 1, it is used in tasks which are required to calculate the probability of values. The mathematical formulae for sigmoid function is given by:

$$f(y) = \frac{1}{1 + e^{-y}}$$

Hyperbolic Tangent

The Hyperbolic Tangent/Tanh function takes values from all the real numbers and gives the output between [-1,1]. This function is used in neural networks because it allows non linearity in the network which is important for multiple layers. Since it is zero centered it helps in faster convergence in the case of gradient descent. Also, it is easy to calculate the derivative of tanh function. It is given by the formulae:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Rectified Linear Unit (ReLU)

Rectified Linear Unit or ReLU is a piecewise linear function and it is used to introduce non-linearity into the model, i.e. the relationship between input and output is not a simple straight line. It gives zero for negative inputs which gives its main advantage over other activation functions. At the same time, its main disadvantage is also related to these zero values provided, i.e. it stops gradient flow during backpropogation form neurons that were not activated during the forward pass. These neurons are called dead neurons. ReLU is represented by:

$$f(y) = max(0, y)$$

Leaky ReLU

It is a variant of ReLU which instead of returning zero for negative values, returns a small negative value that is proportional to the input provided. This helps Leaky ReLU from never completely dying and learning the negtive inputs at the same time. Therefore it solves the dying neurons problems present in ReLU but still the implementation of this activation function depends on the specific requirements of the model. It is given by:

$$f(y) = max(a * y, y)$$

where a is very small positive value

Parametric ReLU

Parametric ReLU (PReLU) also addresses the dying neurons problem present in ReLU. The difference between Leaky ReLU and PReLU is that PReLU has a learnable parameter which is learnt during the training process. This learnable parameter can help the model to increase its accuracy during the training process. It is more generalised version of ReLU, such that when the value of a_i is small it functions like Leaky ReLU.

$$f(y) = max(ay, y)$$

where a is the slope of the graph for negative values.

Exponential Linear Units

Exponential Linear Unit or ELU also fixes the problem of ReLU by applying log values for the negative inputs. This allows for ELU to have a smooth curve in the negative region. This allows it to provide better predictions during the training process as it increases the model flexibility. But this increases the computing time in the training process.

$$f(y) = max(ay, y)$$

Here, the value of a has to be changed manually.

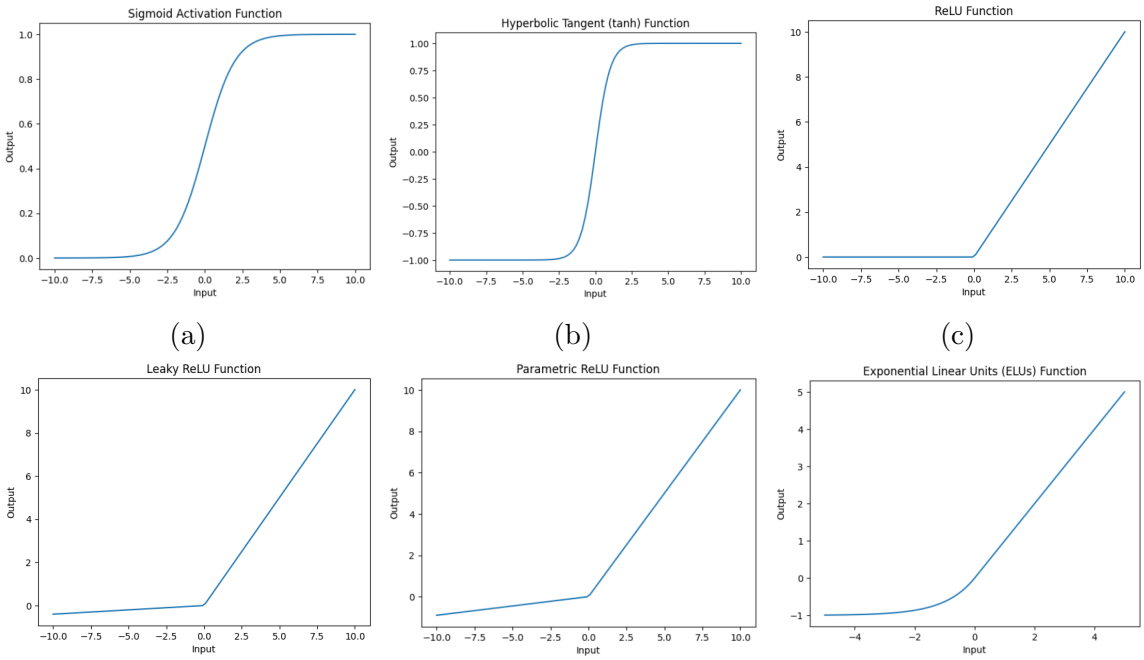


Figure 2.3: Activation functions

2.4 Loss Functions

The loss/objective function is important for optimization in the training process. This function determines whether the model is predicting well or not. If the loss function is lower, it implies the model is closer to the target values and hence it is a good model. Loss functions are specific to the case we are trying to implement them, therefore different loss functions produce different results and we need to find which loss function works the best for our case. Loss functions can be divided into two broad categories on the basis of problems that we solve:

1. Classification Loss: These are used for predicting the class that a particular the result belongs to and used mostly in discrete cases.

2. Regression Loss: These are used to predict the numerical values as a result and used in case of predicting continuous values.

Some loss functions that are used most commonly in deep learning are as follows:

Mean Squared Error (MSE)

MSE is also known as quadratic loss and it is given by the average of squared differences between the predicted and actual values. It is mostly used in regression problems where it is required to evaluate the closeness of the prediction to the desired values in the supervised problem. It is calculated using the formulae:

$$\text{Mean Squared Error} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.1}$$

Mean Absolute Error (MAE)

This loss function calculates the average of the absolute differences between the predicted and test values. It is also used mostly in regression tasks. The drawback of this loss function is that it is less sensitive to outliers compared to MSE, but also makes it preferable over other loss functions when there are many outliers present in the dataset. It is calculated using the formulae:

$$\text{Mean Absolute Error} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{2.2}$$

Binary Cross-Entropy

It gives the probability between two values that is between 0 to 1 and based on the dissimilarity between the actual labels and the predicted probabilities it calculates the required loss. This loss function is mostly used in binary classification problems. . It is given by the formulae:

$$\text{Binary Cross-Entropy Loss} = -\frac{1}{n}\sum_{i=1}^{N}\left(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)\right) \tag{2.3}$$

Categorical Cross-Entropy

This loss function calculates the dissimilarity or the error between actual probability and the predicted probability of the target class. This function is mostly used when predicting multi-class classification problems and it requires one-hot encoding due to multiple classes and are represented by vector 1 in the position of correct class and 0 for the wrong class.

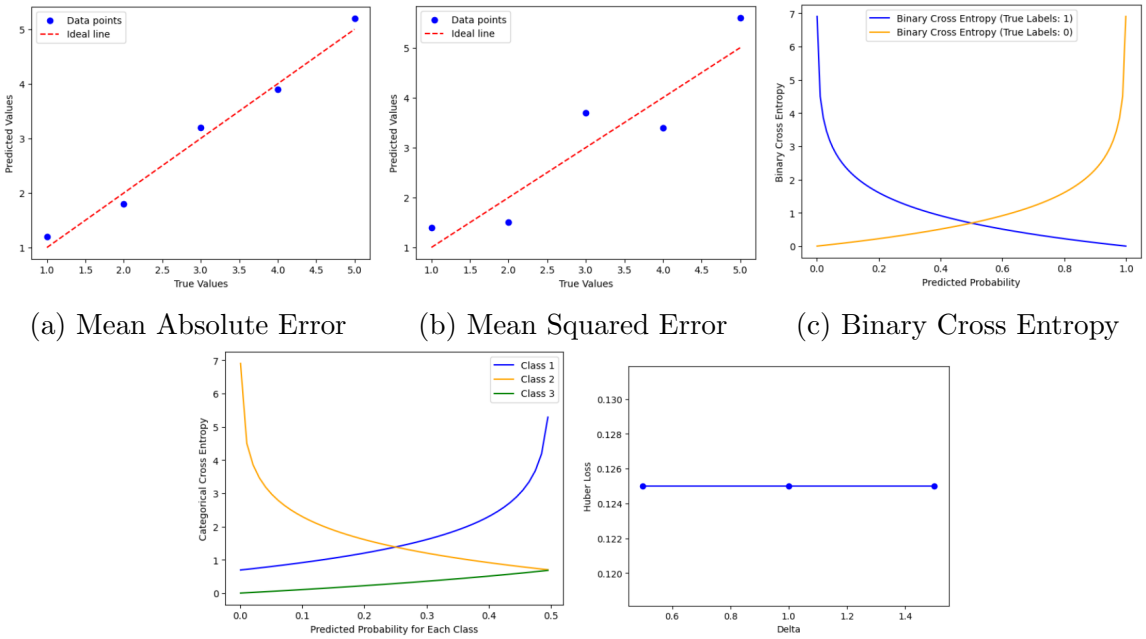
The formulae for CCE is given by:

$$\text{Categorical Cross-Entropy} = -\frac{1}{n}\sum_{i=1} n \sum_{j=1} m \cdot y_{ij} \cdot log(\hat{y}_{ij}) \tag{2.4}$$

Huber Loss

This loss is also called Smooth Mean Absolute Error, it is less sensitive to outliers in data than squared error loss due to piecewise definition of the function. It is also differentiable at 0 and is mostly used in robust regression. The formulae for Huber Loss is given by: Loss =

$$L_\delta(a) = \begin{cases} \frac{1}{2}(Y - \hat{Y}_i)^2 & \text{for } |a| \leq \delta, \\ \delta(|Y - \hat{Y}_i| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases} \tag{2.5}$$



(a) Mean Absolute Error (b) Mean Squared Error (c) Binary Cross Entropy

(d) Category Cross Entropy (e) Huber Loss

Figure 2.4: Loss functions

2.5 Transformers

Transformers are a type of neural network architecture (like CNN's) which were introduced in 2017. They work on the idea of self-attention, that is they use mathematical technique set that helps them to identify the distance of data that have influence over the model are from one another. They were introduced in a paper called "Attention is all you need" in 2017 [2]. The introduction of transformers have been revolutionary in the field of artificial intelligence.

Transformers are designed in the way to handle sequential data. The data include language, speech, time series and genomic sequences. These models have become one of the fundamental models to be applied in Natural Language Processing (NLP).

The need of transformers can be tracked down to the vanishing gradient problem which cause a long-term memory loss. Since the texts are sequentially processed in RNN's therefore the important parts initially could get missed later on. LSTM's addresses this problem but still the words in them are processed one by one. Transformers transcends these issues and can process the entire sentence at once.

The overview of the entire transformer architecture Figure 2.5 can be divided into two parts: the encoder and decoder. Here, encoder processes the input sequence and produces continuous representation of the input. It consists of multiple identical layers. Then the decoder uses the embeddings produced by the encoder to generate the output sequence. The decoder does the process of intake and generating output at the same time allowing parallelization in completing the task

Some of the key aspects of Transformers are:

1. **Input Representation:** Transformers learn the representation of the input data, but the representation is very complex to capture between words and concepts are not very intuitive from the apparent meaning of the text.

2. **Positional Encoding:** They lack the understanding of the sequential order of the elements, so to provide them a positional order, positional embeddings are attached with the input embeddings. The encodings provide the position to which the tokens belong to in a sequence.

3. **Position-wise Feedforward Networks:** The feedforward network in transformers have fully connected layer followed by a non-linear activation function. They transform

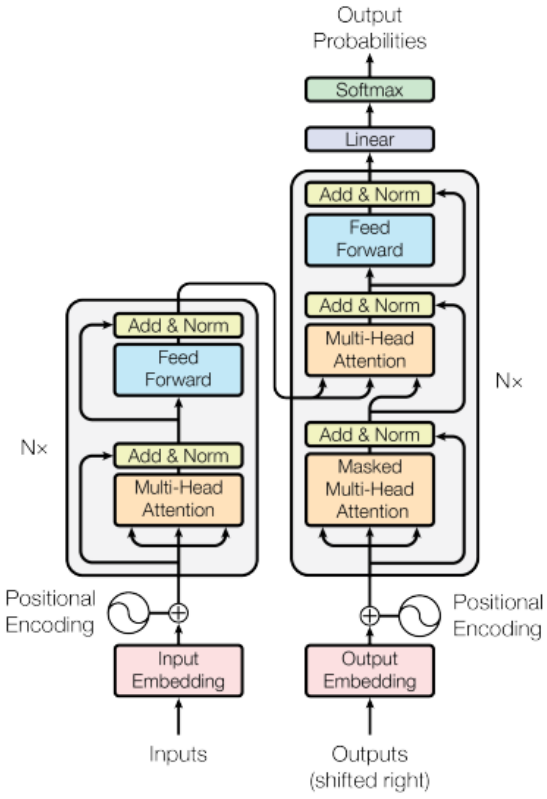


Figure 2.5: Transformer Architecture, Source [2]

the representation at these positions using the same layers.

4. **Attention Mechanism:** Attention is taking the weighted average of the output produced by the encoder which is then fed to the decoder. This step helps the model by feeding only the necessary information which increases the performance.

2.6 Vision Transformers

Transformers were started for sequence-to-sequence learning and with time they have been applied to the fields of vision as well. The approaches of applying transformers to image datasets showed that even without CNN transformers can be applied on image classification tasks to produce good results [3]. It had been theoretically proved that self-attention can learn to behave similarly to convolutions [12].

Vision transformers (ViT) works a little different from how it worked in the case of text

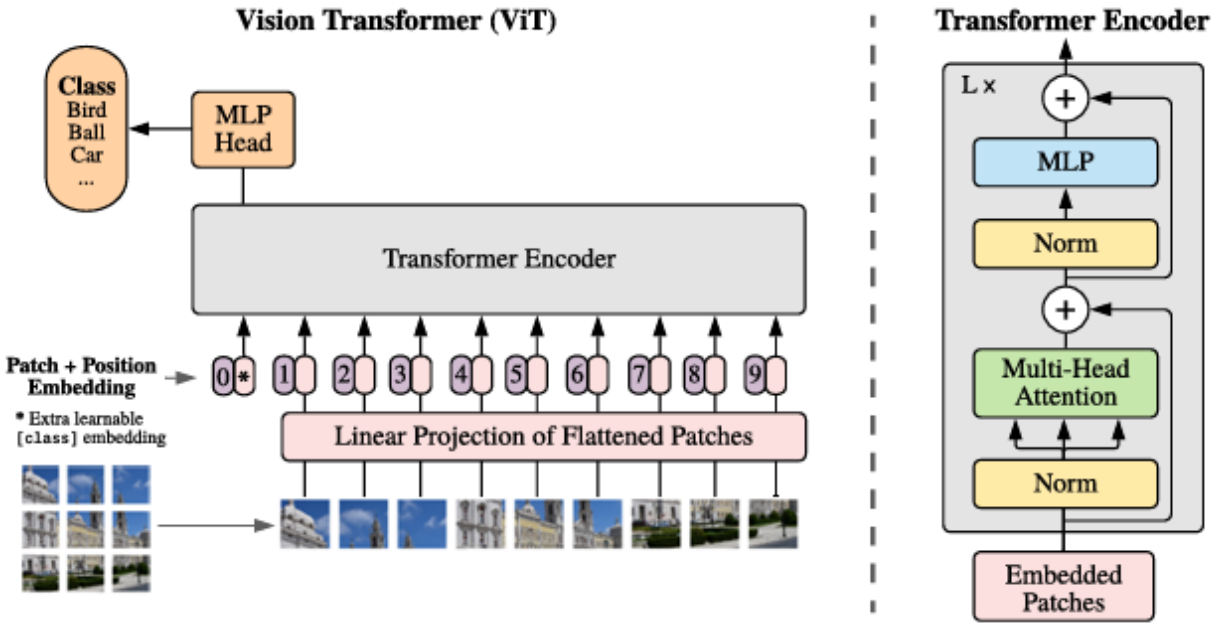


Figure 2.6: Vision Transformer Architecture,
Source [3]

data. The main components of the vision transformer architecture (Figure: 2.6) consist of a transformer encoder and an MLP head which classifies the images. The major steps in the working of Transformers are as follows:

1. **Patch Embedding:** First the image provided to the transformer is split into multiple patches. These patches are then flattened (i.e. converted from a multi-dimensional array into a 1-D array) and are then simplified as a single convolution operation

2. **Vision Transformer Encoder:** The Multi-Layer Perceptron of the vision transformer encoder is almost similar to the positionwise Feed Forward Neural Network of the original transformer encoder. The only difference are that the activation function used here is Gaussian Error Linear Unit which can also be called the smoother version of ReLU and to the output of each fully connected layer of MLP for regularization has dropout applied to them.

It is seen that the vision transformers do not perform better than ResNet when a small dataset is used (data sets of sizes in 1-10 million), but when the data sizes are considerably large (300 million), then it can be seen that the performance of transformers outperforms CNNs.

2.7 Foundation Models

They are machine learning models that have been trained on a huge amount of data so that these models can be applied on a wide variety of cases. Foundation models have completely changed the landscape in artificial intelligence, one well-known example of a foundation model would be ChatGPT.

Foundation models come with the drawback that it is extremely costly to get enough data to train a foundation model, but comparatively, the cost gets very cheap when it is able to produce results for such a huge variety of data.

Some features of foundation models are:

1. **Broad Training Data:** They are trained on massive datasets and hence are usually used in the cases of self-supervised learning which is the current need in various fields that contain huge amounts of unannotated data and very little annotated data. This broad training helps them to have a high accuracy for specific test cases.

2. **Adaptability:** These models have the feature to work on the basis of the prompts provided to the model. Therefore, on the basis of the prompts provided, they have the ability to huge variety of tasks.

3. **Cost Effectiveness:** They are expensive only till the point they are trained, but once a model is completely trained, it gets much cheaper and faster to work on new applications. Usually, new applications require experts to provide data manually but through the introduction of these models and their wide scalability, this process can be automated.

4. **Multi-Modal Capabilities:** These models have a wide range of modalities that is, they have been applied to different fields such as text (ChatGPT,) images (DALL-E), music (MusicGen), robotic control (RT-2).

Chapter 3

Literature Review

This chapter talks about the various models and literature associated with them that were studied for the task of image segmentation and classification of the Reed-Sternberg cells along with the tissue microenvironment cells. These approaches have helped in reaching important conclusions in the study.

The first major problem at hand was the lack of annotated data. The test data was received from the hospital but there were no annotations attached to them and in order to train a deep learning model that is able to segment and classify the required cells we required an annotated data set. To approach this problem, we had two options.

First, to use open-source data available online in order to train the segmentation and classifier model. Second, to get a professional annotator who can annotate the data for our case. This section mentions the literature associated with the online open-source data that were reviewed to create the model.

The open-source data available online that were used by us for creating a segmentation model were made available in the form of competitions. The competitions that we selected were the following:

3.1 Competition Datasets

3.1.1 MoNuSac

The MoNuSAC dataset [4] consists of about 46,000 hand-annotated data of nuclei that covers 31 hospitals. The main aim of creating this dataset was to identify the epithelial and three types of immune cells that are important for characterizing Tumour Micro-Environment (TME). The dataset is segmented and classified into epithelial cells, lymphocytes, macrophages, and neutrophils.

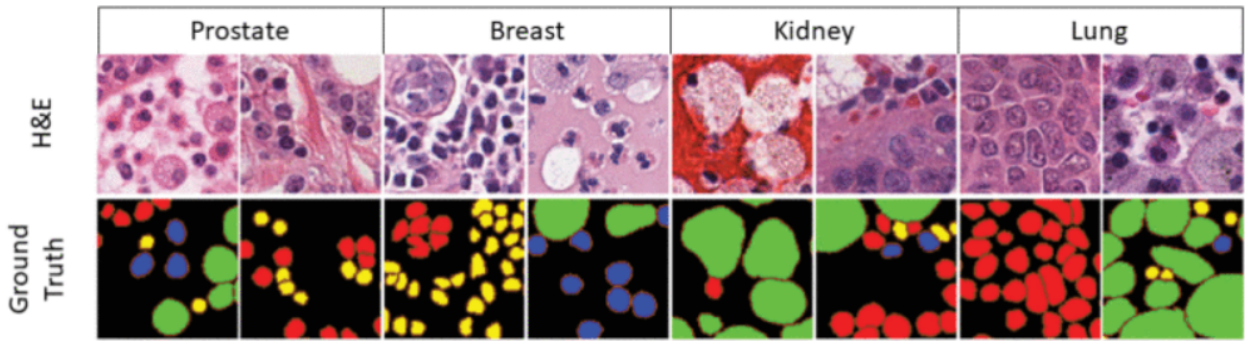


Figure 3.1: MoNuSac Dataset,
Source [4]

3.1.2 MoNuSeg

The MoNuSeg dataset [5] consists of nuclear annotations of the cells of various organs of the body such as the breast, liver, kidney, prostate, bladder, colon, and stomach. But the cells are not classified, i.e. it is not mentioned in the annotations that the nuclei belong to which cell. For example, nuclei of lymphocytes and plasma cells are all merged into one and one cannot predict which certain nucleus belongs to which cell.

3.1.3 PanNuke

The PanNuke dataset [6] consists of about 200,000 nuclei which are categorized into 5 classes that are considered clinically important in order to segment and classify the nuclei in WSIs. It data labeled here has been divided into neoplastic and non-neoplastic categories. The

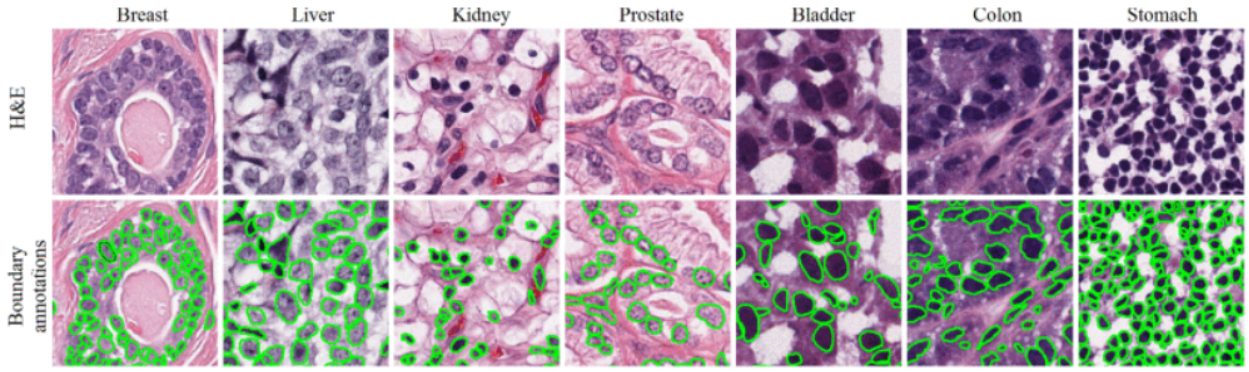


Figure 3.2: MoNuSeg Dataset,
Source [5]

neoplastic (or new growth) contains all kinds of tumors that are either malignant or benign. The non-neoplastic contains everything else. The normal inflammatory, degenerative, metaplatic, etc.

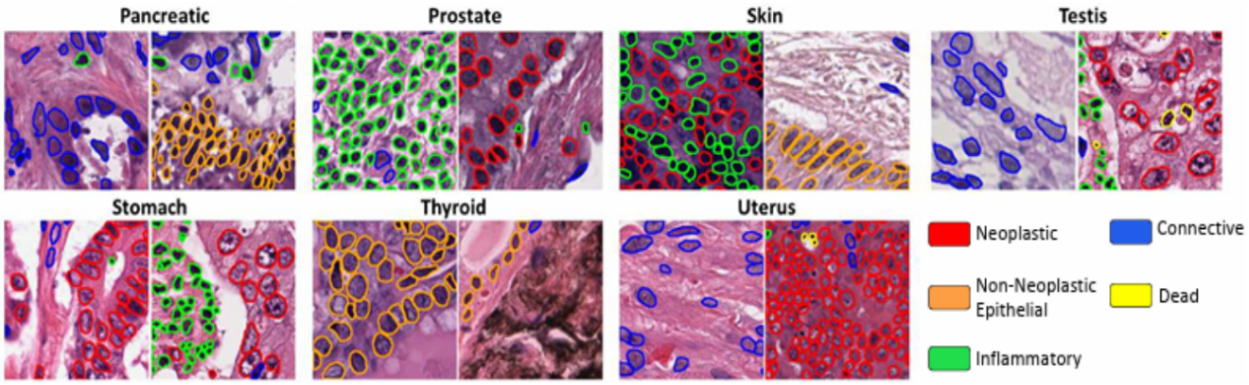


Figure 3.3: PanNuke Dataset,
Source [6]

The division of cells here is into 5 classes, namely: Epithelial, Connective/Soft tissue cells, Lympho-reticular cells, Nervous system cells, and Dead cells. There are many subclassifications of cells that are covered in these five broad classes.

3.1.4 Conic

The CoNIC dataset [7] is a part of the Conic challenge that aims to segment, classify, and count the colon nuclei cells present in the intestines. The dataset is composed of lizard colon cells in a 20x objective magnification. The cell nucleus was classified into the following

categories: epithelial cells, connective tissues, lymphocytes, plasma cells, neutrophils, and eosinophils. Therefore, the model trained on the above class can provide a colon tumor micro-environment.

The dataset provides 4,981 patches of size 256x256 size extracted from the lizard dataset. The segmentation and classification classes are provided in the format 4981 x 256 x 256 x 2. Here the first channel is the segmentation map and the second map is the classification map.

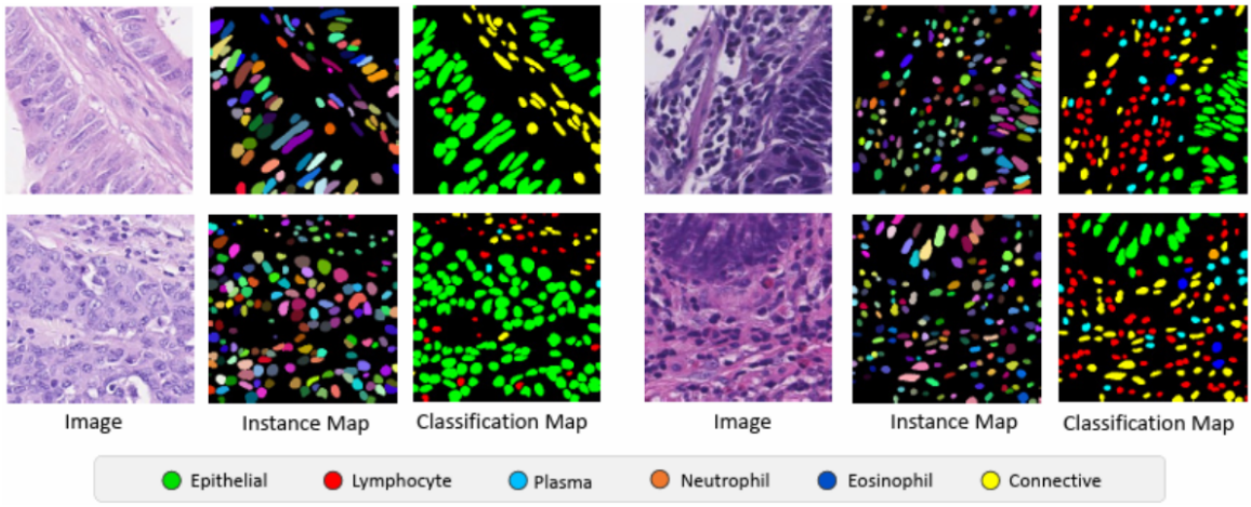


Figure 3.4: Conic Dataset,
Source [7]

3.2 U-Net

U-Net architecture [8] is a model created especially for biomedical image segmentation. A normal Convolutional Network is mostly used for image classification tasks but in biomedical imaging, localisation is required for image classification, i.e. each pixel is required to be classified. Another problem that is associated with biomedical imaging is the lack of many images to train a good model.

U-Net is a model that is able to solve both the above problems. The network architecture is a U-shaped fully convolutional layer (Figure: 3.5) that allows it to identify the class of each pixel. It is in U shape architecture because it is composed of successive contracting layers consisting of 3x3 Convolution layers with ReLU as the activation function and these

are followed with 2x2 max pooling layers and this forms the first half of the architecture. The second half is composed 1x1 convolution layers followed by 2x2 up-convolution layers.

The second problem of the low dataset is solved by using multiple patches that it extracts from the images in order to train the model for better generalisability. It also employs elastic augmentations which allows it to reach better results.

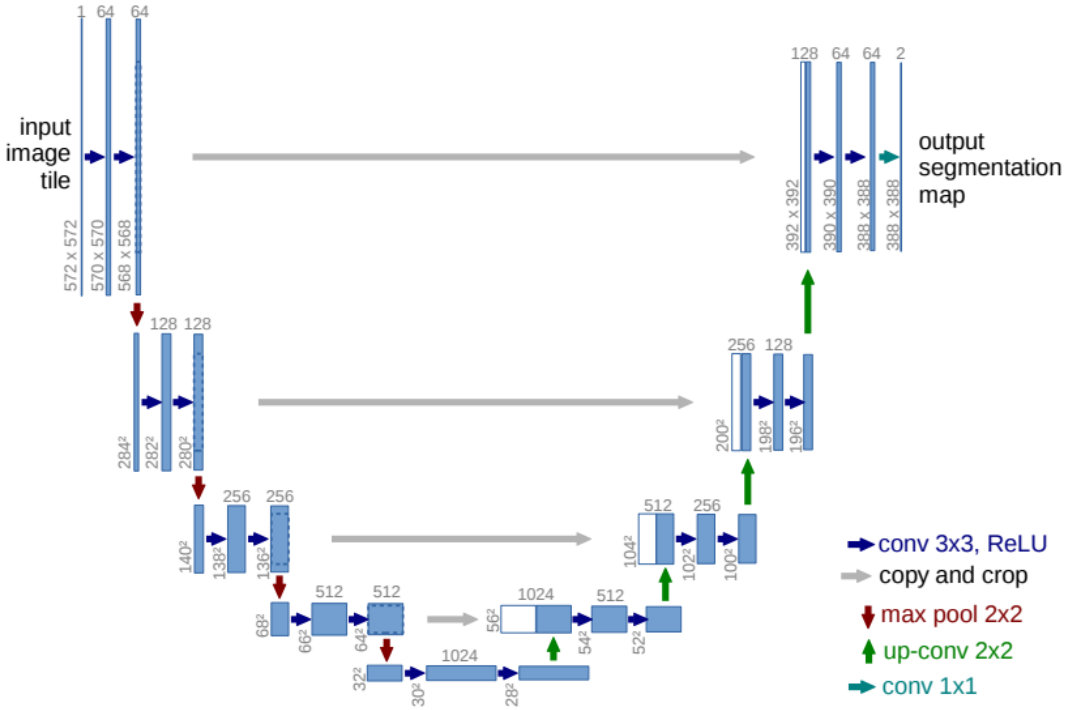


Figure 3.5: U-Net Architecture,
Source [8]

3.3 Stardist

Stardist is a model that was created in order to enable the automatic detection and segmentation of cells in the case of microscopy images [9]. This model overcomes the problem of segmentation in the cases of overcrowded cells following localized cell nuclei using star-convex polygons instead of bounding boxes. This method applies two approaches which help it in getting the prediction results:

1. Object probabilities $d_{i,j}$ and star convex polygons radial distances $r^k_{i,j}$ that are estimated through a simple U-Net architecture.

2. Final images from multiple images are selected using non-maximum suppression (NMS).

In the figure 3.6, the dense candidate prediction part is done using a U-net architecture and the NMS part then finds the best prediction of the cells from all the outputs generated by U-Net. The activation function used in the U-net is relu to avoid 2 layers for fighting over features.

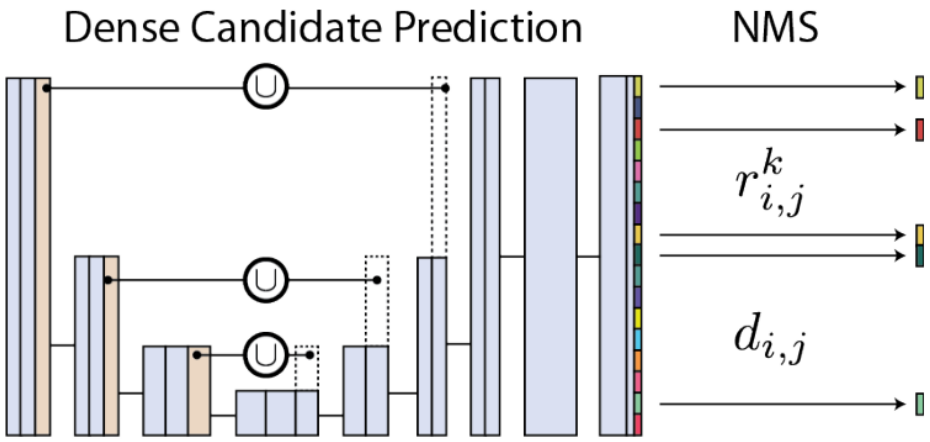


Figure 3.6: Stardist Architecture,
Source [9]

When training, binary cross entropy loss is minimized for predicted object probabilities. The polygon distances are calculated using mean absolute error loss that are weighted with object probability before averaging so that pixel-wise errors are multiplied with object distance before they are averaged. This causes the predictions that are closer to the cente of each object to be weighted more.

For Non-maximum suppression, a greedy approach is applied so only those polygons remain that have the highest object probabilities. So a threshold is set and those polygons are selected with higher value than the threshold, and then their intersection with standard polygon is measured.

3.4 Segment Anything Model

The Segment Anything Model (SAM) [10] is a foundation model for image segmentation. It is the largest segmentation model to date that is trained over 1 billion images. Training over such a large dataset has allowed it to perform zero-shot image segmentations very precisely

in most of cases. It has three components: an image encoder, a flexible prompt encoder, and a fast mask decoder. These are built on transformer vision models.

1. **Image Encoder:** Masked Auto Encoders (MAE) pre-trained vision transformers are used to apply the prior in order to prompt the model.

2. **Prompt Encoder:** This consists of 2 types of prompts: sparse and dense where box and points are represented by positional encodings and the dense prompts (i.e. masks) are encoded using convolutions and summed using element-wise image embedding.

3. **Mask Decoder:** The task of mask decoder is to map the image embeddings, prompt embeddings, and output token to a mask. This is done using a transformer decoder block which is then followed by dynamic mask prediction head. The modified decoder uses a prompt for the updation of all embeddings. Then all the image embeddings are upsampled and a MLP maps the output token to a linear classifier. This helps in predicting the mask probability at each location in the foreground. This allows SAM to predict masks on masks on images very accurately for many specific cases for which it might not have been trained explicitly.

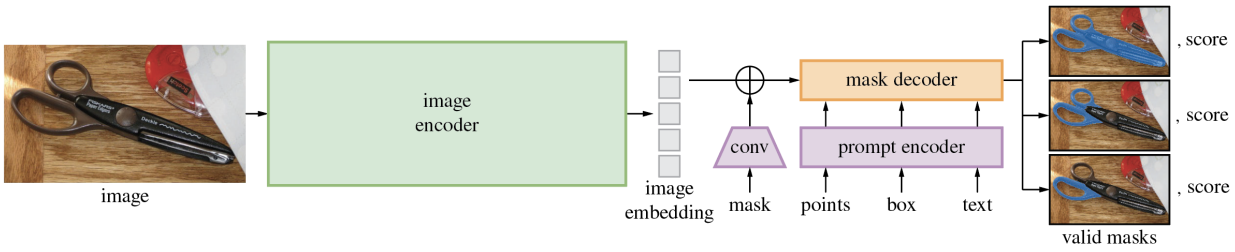


Figure 3.7: SAM Architecture,
Source [10]

Chapter 4

Data and Methodology

The data obtained is divided into two parts. The data used for the training purpose and the data used for the testing purpose. Since our aim is to identify the Reed Sternberg cells, we already had the testing data available with us.

The data on which predictions were to be made were obtained from the Tata Memorial Hospital (TMH), Mumbai. The data contained of Whole Slide Images (WSI) was obtained from the patients who were diagnosed with Hodgkin's Lymphoma (HL). These WSIs were created by scanning the physical slides which were stained using hematoxylin and eosin stain and then an Ultra Fast Digital Pathology slide scanner was used to scan the images at high resolutions. The limit of the scanner was 40x zoom into the whole slide image.

Then the data used for the training purpose was obtained from the online open-source histopathology challenges (see 3.1). These data provide different cells present in the tissues that are segmented and classified into various classes.

4.1 Data Preparation

Data was prepared using a Python library called OpenSlide. This library is used for creating patches from the Whole Slide Images (WSIs) since they are of the resolution 50,000 to 100,000 pixels and hence it is hard to make inferences on them directly. In this library, we specify the zoom level the patches are required and what should be the resolution of each patch that is extracted.

The patch sizes chosen in our case were 256x256 and with a zoom level of 40x magnification.

About 45,000x patches were extracted from each WSI where the dimensions of one WSI were around 50,000 x 50,000. Figure 4.1 shows the complete size of a WSI and different levels of zoom until we see the cells in about 20x to 40x zoom levels.

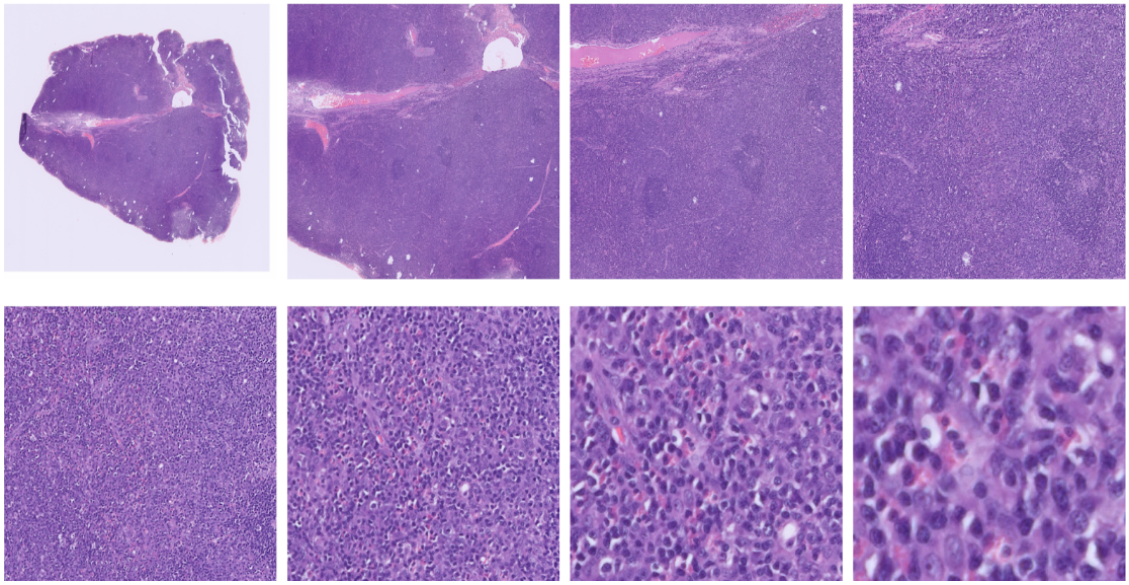


Figure 4.1: Different zoom levels in Whole Slide Image

4.2 Methodology

The main task of the project was to segment the H&E (Hematoxylin and Eosin) stained WSI's and then classify the individual segmented cells in order to provide a total count of each type of cell for Hodgkin Lymphoma and tissue microenvironment prediction. We were required to identify six main types of cells: Plasma, Reed Sternberg, Macrophage, Eosinophil, Neutrophil, and Lymphocytes. We used multiple approaches to solve the problem. The separate methodologies and their respective results are divided on the basis of the approach that we followed. The problem has two broad parts: segmentation and classification.

4.2.1 Segmentation

We required the instance segmentation of the cells and therefore many generic approaches of segmentation could not be employed. After the literature review, we tried to follow the

approaches that were used in the histopathological image segmentation and classification competitions along with other very recent approaches for image segmentation:

1. **Open source data:** First the open source data (3.1) was used to train a model that can segment our patches into the required cells. In order to segment the patches first the open source histopathological data were used to train the model but, due to lack of significant prediction results, data was manually labeled to check the prediction results.

2. **Foundation Models:** Foundation models (2.7) have the ability to provide zero-shot predictions (predicting results on a dataset without seeing any examples beforehand) on a completely new dataset. So we did not have to provide images from our dataset to train the model for segmentation. We used the Segment Anything Model (3.4) to provide us with segmentation masks and this gave us some significant results to work on. But since the predictions are zero-shot it also produces many unwanted results with the required cell segmentations. Therefore, we built a Convolutional Neural Network to filter out these noise in our output and only keep the required cells.

4.2.2 Classification

The classification task required the output from the segmentation results. However, one of the major problems was the lack of annotations in our dataset (i.e. Hodgkin's Lymphoma dataset). We required some ground truth to train and test the classifier since the online open-source dataset was not sufficient.

Therefore, an expert annotator was asked to annotate the six classes of cells that were required to be predicted. The annotator was provided the patches extracted from WSI's and masks were created for the patches using QuPath software which is used for annotating and viewing H&E images and WSIs.

Training Classifier for Cell Classification

A Convolutional Neural Network (CNN) model was trained on the cells annotated by the expert. The architecture of the CNN had layers which applied 2D convolution operation with 3 input channels and 32 output channels. This layer uses 3x3 kernel, a stride of 1 and a padding of 1. The second convolutional layer takes 32 channels from the previous layer and produces 64 output channels using a 3x3 kernel, stride 1, and padding 1.

The model has max-pooling layer after the convolution layer. The max-pooling with 2x2 kernel and stride of 2 is applied to reduce the spatial dimensions of feature maps.

The model has 2 fuly connected (linear) layers. The first layer takes the flattened output from the previous layer (64*3*3) and produces a 512-dimensional output. The next linear layer produces the output with the number of dimensions that is equal to the number of classes, here which is equal to six.

A flatten operation is applied on the data before passing the data to fully connected layers. A flatten operation converts the 3D tensor from the convolutional layer to a 1D tensor.

The forward pass of the network specifies how the input data is processed through the layers defined in the constructor. It defines that the input goes through the convolution and pooling operations and then the resulting map is flattened. The flattened features are passed through two fully connected layers with ReLU activation and then final output is returned.

Training Classifier for Noise and Cell Classification

As described in the segmentation section, the outputs from the Segment Anything Model provide noise along with the cells of interest, therefore a noise filtering CNN was trained to classify whether the given image was a noise produced by the model or not. A network architecture similar to the cell classifier was used since the images were converted to have the same resolutions. The main difference was that the previous classifier predicted six classes and this classifier predicts two classes.

The model training required manually separating the images by going through each image that the SAM outputs from the patch used. The masks generated from the patches were divided between the ones that gave noise (not essential information) and others that gave the relevant cells that are going to be used for the classification.

Computational Resources

All the models were trained on NVIDIA RTX A6000 48GB GPUs available at the KCDH center at IIT Bombay. RTX A6000. It has many cores which allows for parallel processing. Rest of the codes were run on Google Colab which required a limited amount of GPU for training.

Chapter 5

Results and Discussion

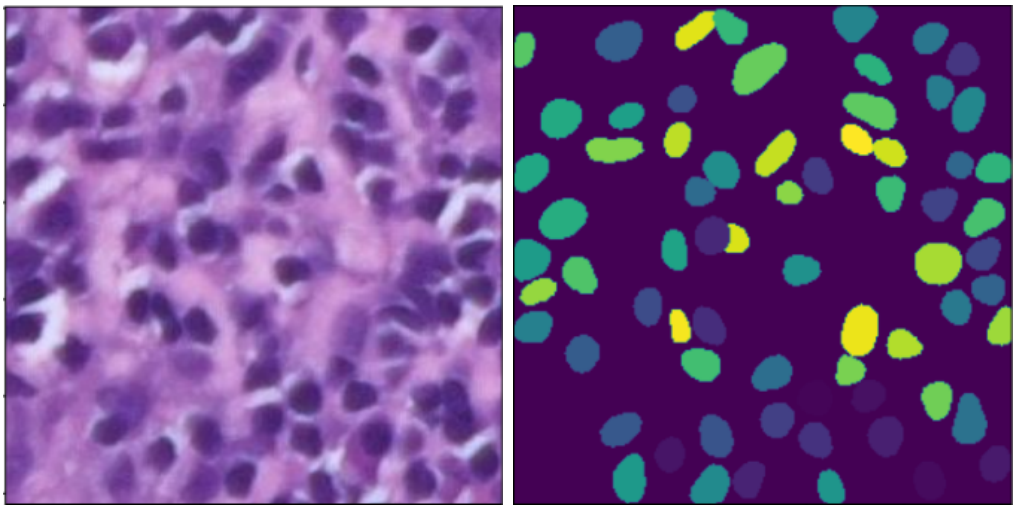
This chapter includes the major results of the thesis along with the reasons why certain procedures were followed and the inferences that were produced from the results. It is divided into sections according to the major model that was used for the task of segmentation and classification.

5.1 Stardist Results

The Stardist model (3.3) was applied to the dataset in order to first segment the cells in the model. Due to the lack of annotated data in our case, we used open-source competition datasets (3.1) for training the model. The stardist has a pre-trained model which is already trained on the MoNuSeg challenge dataset (3.1.2) and the TNBC dataset [13], which is a dataset containing patches and masks of images of different types of tumors.

The results (Figure: 5.1) show that the model is able to find the nucleus of the cells through the pre-trained model on the MoNuSeg dataset. But, for our case, we require the cell classifications as well but the MoNuSeg model does not contain nuclei classifications as all the nuclei classifications are merged into one. To address this issue we try implementing the model trained on the Conic dataset (3.1.4).

Therefore, we used the model trained on the Conic dataset (3.1.4). Figure 5.2 shows that the model is able to predict the results pretty well. The are predictions of multiple classes present in the conic dataset which consist of epithelial cells, lymphocytes, plasma cells, neutrophils, and eosinophils. So, we try to implement this model for our case of Hodgkins Lymphoma.



(a) Image Patch (b) Image Mask

Figure 5.1: Mask created using Stardist

We see from the results in figure 5.3 which shows that the model is able to segment the nucleus of the cells and we also find that there are multiple classes that it is able to find because here, different colors of cells in the figure 5.3 prediction (class) represents different types of cells that are present in the conic dataset. But we also need to verify whether the predicted cells are correct or not.

To check we counted the number of cells of each class that it was predicting. The results (Table: 5.1) show that the model is identifying cells well. However, this is what the model predicted. There is a need for cross-verification of what is the actual count of each of the cells in the given patch image. So, an expert annotator was asked to help us with this task. However, this task of manually checking by the annotator required a higher magnification level for correct identification. It is hard to differentiate between some types of cells predicted here due to their similarity with each other.

Iou	**Precision**	**Accuracy**	**Recall**
0.531	0.735	0.916	0.657

Table 5.1: Stardist Conic Implementation

The problem with the predictions of stardist model trained on conic dataset for HL case prediction can be summarized by the following points:

1. **Misclassifications:** We can see in the table 5.2 that the total number of plasma cells

predicted by the model is 0, but the annotator was able to identify plasma cells (one of the examples has been labeled in the figure 5.3 by red arrow) that were misclassified as lymphocytes due to their similarity between each other.

One more point that was noted is the Reed Stenberg (RS) cells were also classified as epithelial cells along with other cells, so multiple cells were being merged into one type of cell and there were no separate labels for these classes.

2. **Resolution:** The Conic dataset is in 20x magnification due to which we are required to convert our patches down to 20x from 40x. However, this magnification is not enough for the annotator to correctly verify all the labels.

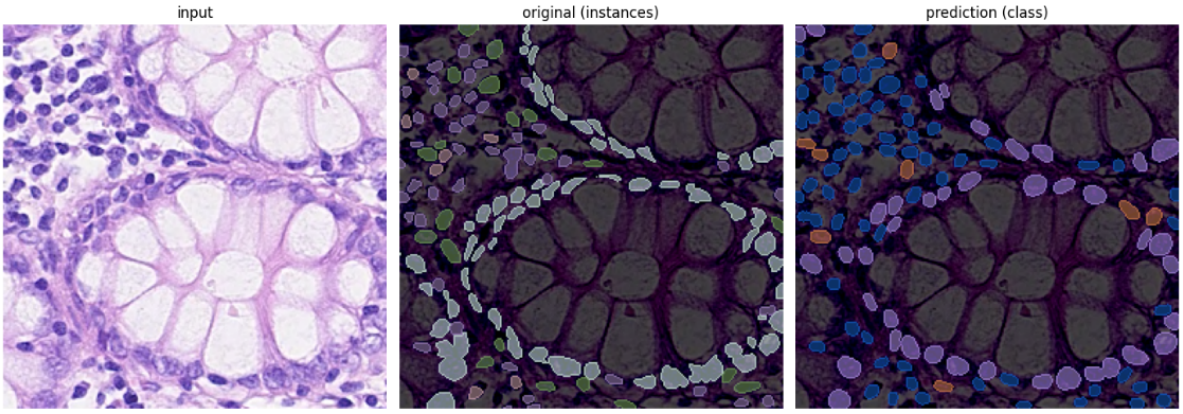


Figure 5.2: Stardist trained on Conic dataset

Neutrophil	Epithelial	Lymphocytes	Plasma	Eosinophil	Connective
0	4	188	0	0	33

Table 5.2: Cell Count Stardist (Figure 5.3)

Due to the above factors, we changed our approach and tried to prepare a model that is able to predict one class at a time. We planned to create a model for each class and finally create an ensemble of these models to correctly identify the cells. We begin by making a model that is able to segment and identify plasma cells using conic dataset and then try to implement it on our own dataset.

The results (Table: 5.3) for training only on plasma cells show a high accuracy but low Intersection Over Union (IOU), recall and precision. This happens because the iou is calculated by finding area of intersection(aoi) divided by the area of union(aou). Here, all the cells are

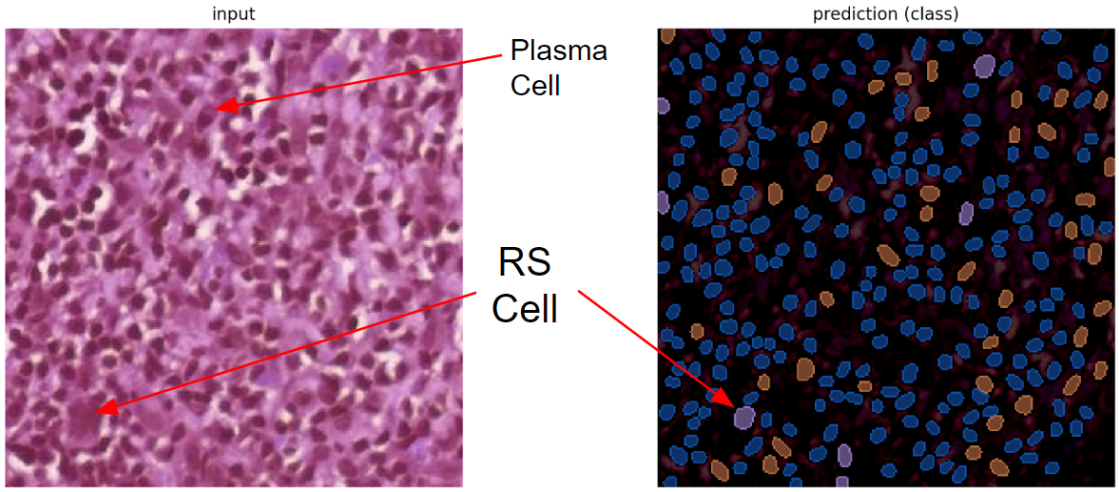


Figure 5.3: Stardist Conic Implementation (HL)

considered for aoi and aou but they are finally predicted only for the plasma cells, so aoi is small but aou is large which makes the iou score low. This can be visualized by seeing the training set image(original (instances) in Figure 5.4) where the colored cells are considered but the predicted cells only consist of plasma cells (prediction (class) in Figure 5.4). Similarly, for recall and precision, the low scores are attributed to the multi-class classification and limiting the prediction classes to the plasma cell.

Iou	**Precision**	**Accuracy**	**Recall**
0.011	0.315	0.802	0.011

Table 5.3: Stardist Conic Implementation (Plasma only)

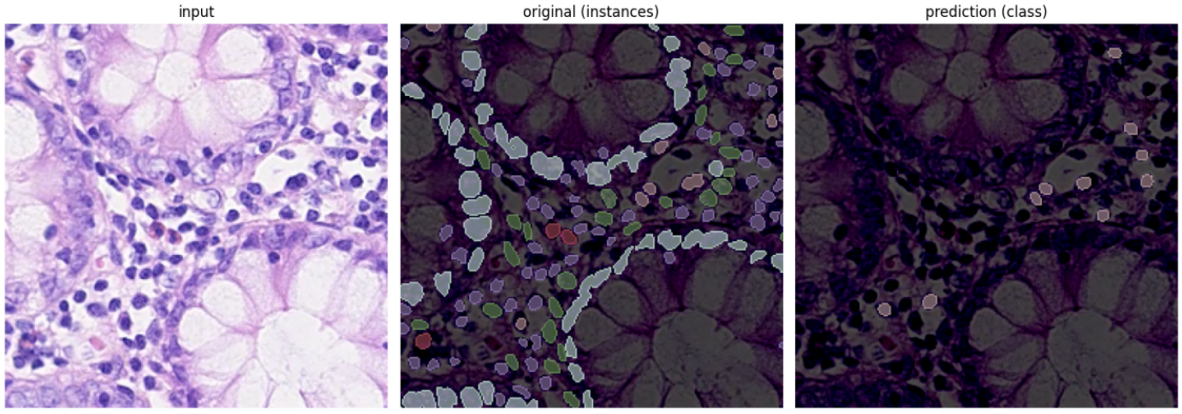


Figure 5.4: Stardist trained on Conic dataset (Plasma only)

But looking at the above accuracy we try to implement this plasma cell-only model for the case of HL. The results (Figure: 5.5) show that the model is not able to identify and predict the results well. The results were also confirmed by the annotator.

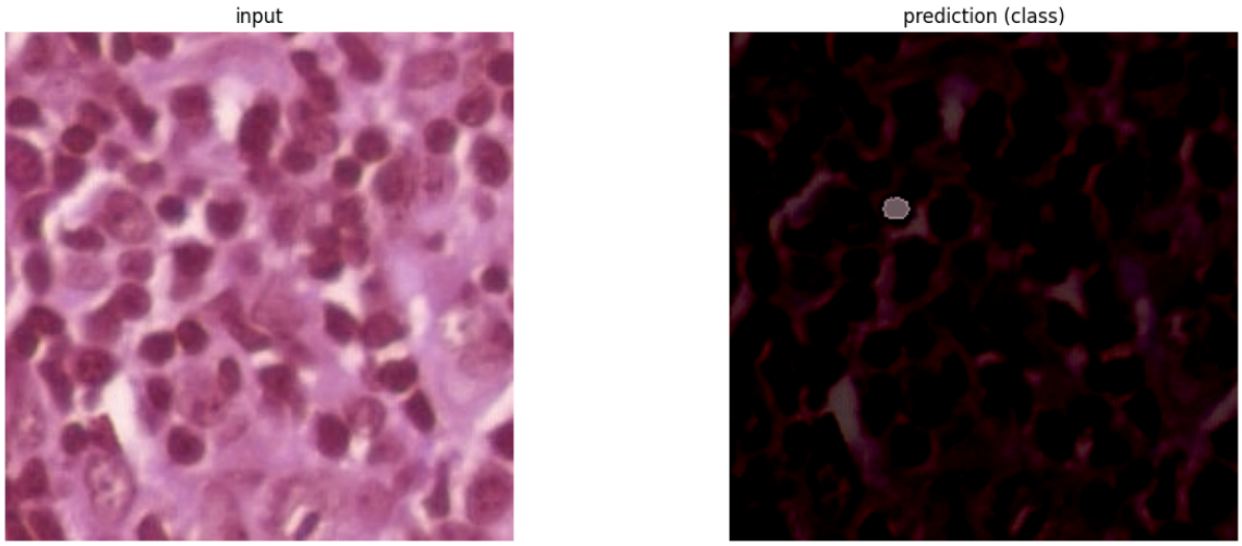


Figure 5.5: Stardist Conic Implementation (Plasma Only)

We were trying this approach and in parallel, we tried many different approaches as well. The previous approach gave us some important conclusions that just using a model trained on an open-source data set is not able to provide us with good results.

We were able to address the segmentation problem to a big extent using the Segment Anything Model (3.4) and the results could be utilized for creating a CNN that could classify the cell images extracted. However, the creation of such classifiers was not possible without annotated patches, where the masks have been created for the cell images within the patch. Therefore, an expert annotator manually labeled the cells in the patches so such a model could be created. The number of cells that were annotated are quantified in the table 5.7. The creation of a dataset also leads to various new approaches that can be implemented to our problem.

5.2 Segment Anything Model

The segment-anything model (3.4) is a zero-shot model and does not require any training to produce segmentations. Using this ability of SAM, we created segmentations from the

patches. Then to find the classifications of individual cells we created two classifiers that can help us predict whether the segmentation mask belongs to the cells of interest or the background, and the second classifier which takes these cells of interest and classifies it into one of the six cells that we are predicting.

5.2.1 Background Noise Classifier

This output given by the SAM model produces many irrelevant segmentations along with the relevant segmentations (Figure: 5.6), so we need to remove these irrelevant segmentations. We train a CNN classifier to remove the background noise from the important cells that are required for the classification of the microenvironment. The images were manually classified into background noise and important cells.

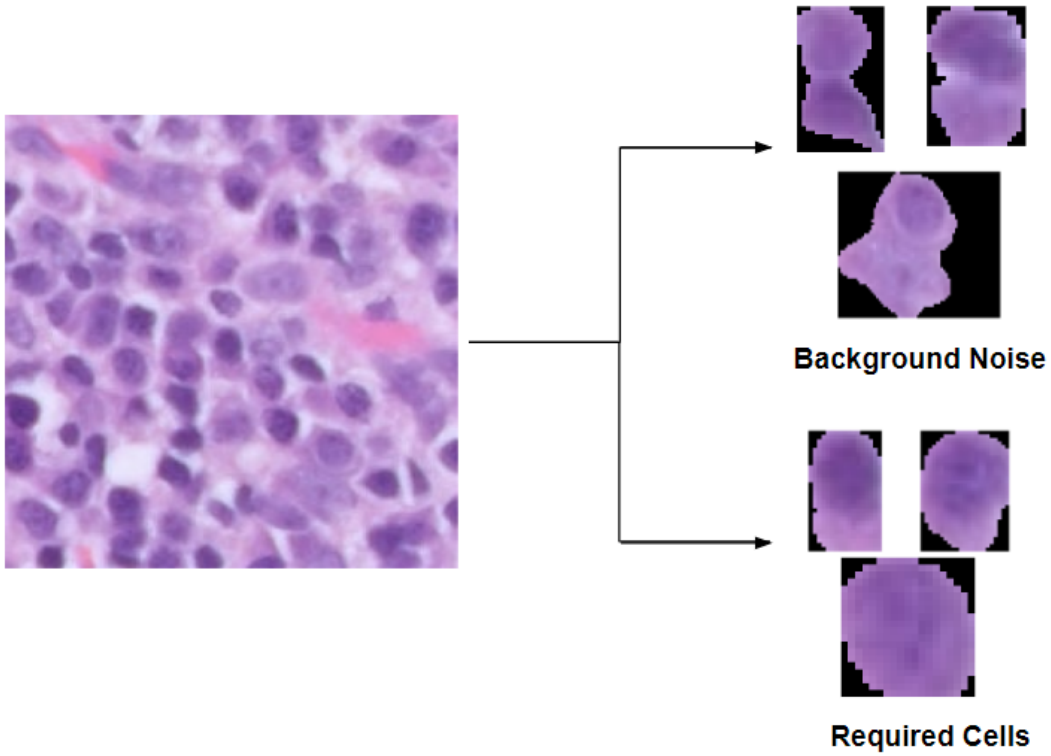


Figure 5.6: SAM implemented on an Image Patch

To train the model we follow the procedure of five-fold cross-validation and keep aside 20% of the images for validating the model. The accuracy is the average accuracy of five iterations of the training and the standard deviation calculated for the five iterations of the training.

The results produced from the training are present in table 5.4. The model was trained on only ten patches to check the performance since annotating the patches takes a long time.

Accuracy	Standard Deviation	Train Set	Test Set
0.746	0.007	753	189

Table 5.4: Background Classification Results

We wanted to do a sanity check on whether the quantity of data has any influence on model accuracy. Therefore, to check if the accuracy increases or not we apply two approaches:

1. **Decreasing Training Data:** By keeping the test data the same as that in the normal case, we decrease the training data to see if it causes lower prediction results. The results (Table: 5.5) show that reducing the training data is actually affecting the predictions by decreasing. This lies in confirmation of our assumption that decreasing the data will decrease the accuracy of our model.

Accuracy	Standard Deviation	Train Set	Test Set
0.683	0.065	203	189

Table 5.5: Background Classification Results (Undersampling)

2. **Applying Oversampling:** The second approach is to create images and augment them into our dataset by performing operations such as flipping and rotation in order to prevent identical images used for training the model. This approach will tell us whether a larger number of images used for training increases the accuracy or not.

   The accuracy (Table: 5.6) shows that it does not increase rather it decreases a little. This can be due to the following reasons:

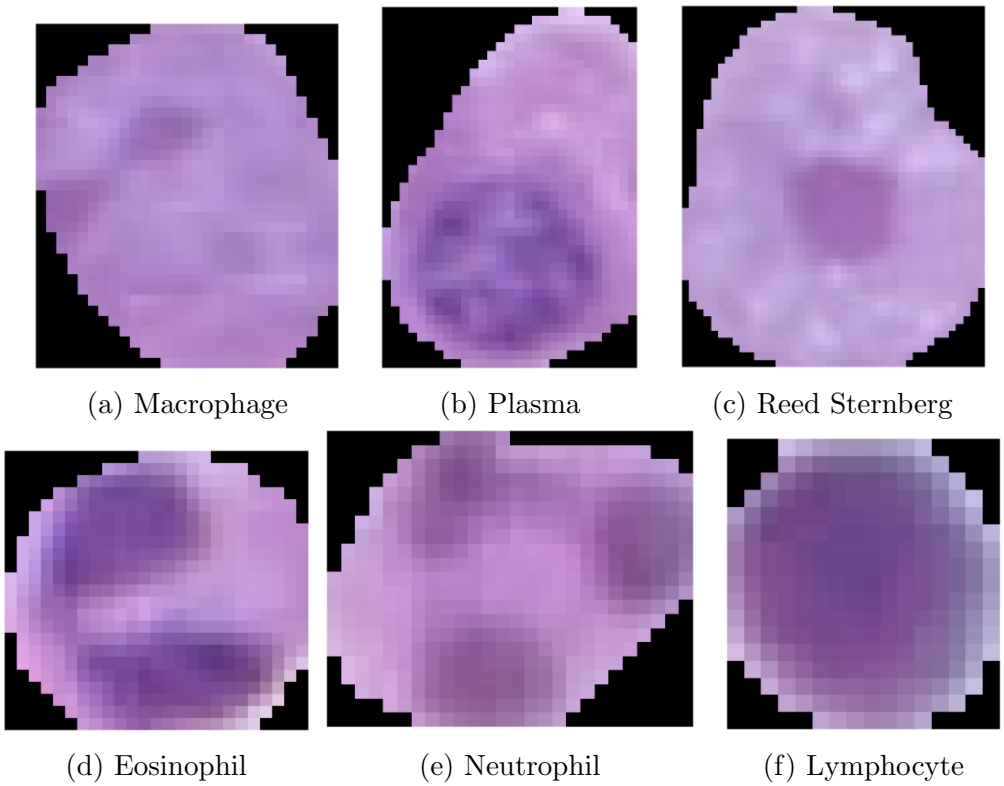
   (a) The model has plateaued and even after feeding more data would not produce any better results. But, since we have only used 10 patches to make these predictions, we will rule out this possibility.

   (b) The second possible reason can be that the model requires more diverse data while training and hence it requires more data for the predictions.

Therefore, from the above two points we come to the conclusion of the requirement for more data to be added to the dataset.

Accuracy	Standard Deviation	Train Set	Test Set
0.735	0.011	1169	189

Table 5.6: Background Classification Results (Oversampling)



(a) Macrophage (b) Plasma (c) Reed Sternberg

(d) Eosinophil (e) Neutrophil (f) Lymphocyte

Figure 5.7: Cells of Interest

5.2.2 Cell Classifier

A Convolutional Neural Network (CNN) was trained on the individual cell images (Figure 5.7) that were extracted from the patches to develop a classifier that can classify the different cells that form the tissue microenvironment. There were six classes of cells that were of interest and were extracted from the patches.

We performed a five-fold cross-validation on the training data and kept aside 50 images for validating the training data. The following results are based on the different cell quantities that were used to train a model with better prediction results. The results in table 5.7 are for the CNN trained on all the cell images except for 50 images kept aside for validation.

The results are not good enough to work as a cell classifier for our case. Therefore, we

Cells	Accuracy	Standard Deviation	Cell Quantity
Plasma	0.468	0.084	501
RS	0.422	0.083	439
Macrophage	0.545	0.032	808
Eosinophil	0.659	0.022	1711
Neutrophil	0.587	0.035	1422
Lymphocyte	0.709	0.021	2083

Table 5.7: CNN Cell Classifier

perform certain changes in the model while training the model. We try to address the two most significant problems that we can observe from the number of datasets with us for each of the classes.

1. **Data Imbalance:** It is a general issue that is faced while training a machine learning or a deep learning problem, especially in classification problems. The imbalance in data can affect the understanding of the model and only predict well for the majority class and perform poorly for the minority class. Therefore, to remove data imbalance we try to equate all the classes and see the results on a balanced training dataset.

   The results (Table: 5.8) show that the accuracy increases for some classes whereas it decreases for other classes. We also note that we have reduced a significant amount of data but still, we find that the model performs with low accuracy and is still not suitable to act as a cell classifier.

Cells	Accuracy	Standard Deviation	Cell Quantity
Plasma	0.640	0.124	439
RS	0.592	0.057	439
Macrophage	0.428	0.047	439
Eosinophil	0.402	0.031	439
Neutrophil	0.503	0.040	439
Lymphocyte	0.335	0.026	439

Table 5.8: CNN Cell Classifier (Balanced)

2. **Data Augmentation:** The other issue that we can notice is the lack of labeled data. This issue is also a very general issue faced in machine learning. To overcome this issue, we employ data augmentation through which we create new images with the help of techniques such as random rotation and flipping (in the case of images) so the

images that are added are not the exact replication of previous images. Adding data through augmentation helps in improving the prediction results of the model [14].

The model was trained with twice the number of cell images from the previous case, i.e. all the images were augmented once to make the total dataset twice its initial size. The results (Table: 5.9) show that the performance of the model has significantly increased from the previous case. This model can be chosen as a cell classifier if we exclude a few classes.

Cells	Accuracy	Standard Deviation	Cell Quantity
Plasma	0.756	0.097	1052
RS	0.526	0.088	928
Macrophage	0.635	0.065	1666
Eosinophil	0.723	0.035	3472
Neutrophil	0.666	0.043	2894
Lymphocyte	0.748	0.029	4216

Table 5.9: CNN Cell Classifier (Oversampled)

Looking at the previous two results, we tried to implement both image augmentation and removing data imbalance in our dataset. We see that the result (Table: 5.10) are the best prediction results among all the previous results. Here, we implemented data augmentation for all the classes of cells whose values were less than twice the value of Reed Sternberg (RS) cells, since it is in the lowest quantity in the given dataset. These results show that this model can be used as a cell classifier for our case.

Cells	Accuracy	Standard Deviation	Cell Quantity
Plasma	0.792	0.085	928
RS	0.690	0.064	928
Macrophage	0.698	0.030	928
Eosinophil	0.751	0.036	928
Neutrophil	0.731	0.034	928
Lymphocyte	0.768	0.039	928

Table 5.10: CNN Cell Classifier (Balanced and Oversampled)

On combining the above two procedures we can get a pipeline which when given a WSI will be able to identify for us how many quantities of each type of cells are present in its microenvironment. Therefore, it will enable us to predict whether the patient is in an

advanced stage of Hodgkin's Lymphoma by counting the Reed Sternberg cell and also being able to predict the microenvironment.

Chapter 6

Conclusion

The main focus of this study was to be able to predict whether the patient is in an advanced stage of Hodgkin's Lymphoma (HL) and classify the type of tissue microenvironment (TME). There were multiple ways to pursue this problem but to get greater precision in the prediction of the result. We employed the method to segment and classify each of the cells present in the Whole Slide Image to get our final predictions of HL and TME.

Therefore, methods involving segmentation and classification of the cells were studied. Due to the lack of an annotated dataset, open-source competition datasets were used to train the models that were used to predict the classifications on our model. The model that was used most by us was Stardist model which is a U-Net based model that is able to identify star-convex shaped objects with very high precision.

Stardist model was able to segment our image patches but was giving wrong classifications for cells. The issue with this approach was also the type of dataset that was being used for training the model as each dataset had different labels associated with them. To solve this problem, an approach of creating an ensemble of models was used where each class of cell had a separate classifier for their prediction. However, this approach also failed due to differences in the model training dataset and the testing dataset. This led us to annotate the HL dataset on our own using the help of an expert annotator.

Then the Segment Anything Model (SAM) was used for segmenting the image patches. SAM provided multiple segmentations both relevant and irrelevant, since we are using zero-shot segmentation here. To classify between the cells of interest and the background noise, a classifier was trained using manually labeled images. Another classifier was trained on the

manually labeled dataset to identify the type of cell any individual cell is.

Using the above two classifiers enables us to create pipeline through which we are able to classify the advanced stage of HL and predict the TME.

6.1 Future Work

From the previous chapter, we can see that there are a lot of tasks that can be approached in order to reach our desired results, i.e. to be able to count the number of Reed Sternberg cells as well as the other important microenvironment cells for microenvironment identification in the case of Hodgkin's Lymphoma. There are two broad tasks in which the further work can be divided into:

1. **Completing the Pipeline:** The first and the most important task following the above work would be to complete a functional pipeline that completes the following steps:

   Take Whole Slide Image → Extracting multiple patches from the WSI → Segmenting the patches → Passing the extracted results from noise classifier → Passing the cells of interest through cell classifier → Count and provide the number of cells of each class in WSI → Predict the microenvironment and affected with Hodgkins Lymphoma

   Once a working pipeline is created then the process of identifying of Hodgkin Lymphoma can be automated, therefore decreasing a lot of repetitive work that is performed by the pathologists on identifying the microenvironment type and whether the patient is affected with HL or not by individually counting each of the cells taken up from multiple regions of the WSIs.

2. **Increasing Model Predictions:** There are many approaches that can be implemented in order to increase the accuracy of the model that is used for predicting the cells of interest and the cell classifier. A few of the methods that can be implemented are:

   (a) Labeling more cells: We know that data is very crucial in making deep learning models and here we face a situation where the models are trained on a lower

amount of data. If more cells are labeled, it will provide better results as it was concluded from the method of data augmentation in the case of the cell classifier and undersampling in the case of noise classifier.

(b) Data Imbalance: The problem with cellular data is that all the cells are distributed unevenly due to which data imbalance is an inherent problem here. To address this, weighted loss functions can be introduced which balances the difference created between the majority and the minority class during the training of the model.

Bibliography

[1] Van Hiep Phung and Eun Joo Rhee. A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences*, 9(21), 2019.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[4] Ruchika Verma, Neeraj Kumar, Abhijeet Patil, Nikhil Cherian Kurian, Swapnil Rane, and et. al. Graham, Amit. Monusac2020: A multi-organ nuclei segmentation and classification challenge. *IEEE Transactions on Medical Imaging*, 40(12):3413–3423, 2021.

[5] Neeraj Kumar, Ruchika Verma, Sanuj Sharma, Surabhi Bhargava, Abhishek Vahadane, and Amit Sethi. A dataset and a technique for generalized nuclear segmentation for computational pathology. *IEEE Transactions on Medical Imaging*, 36(7):1550–1560, 2017.

[6] Jevgenij Gamper, Navid Alemi Koohbanani, Ksenija Benes, Simon Graham, Mostafa Jahanifar, Syed Ali Khurram, Ayesha Azam, Katherine Hewitt, and Nasir Rajpoot. Pannuke dataset extension, insights and baselines, 2020.

[7] Simon Graham, Mostafa Jahanifar, Quoc Dang Vu, Giorgos Hadjigeorghiou, Thomas Leech, David Snead, Shan E Ahmed Raza, Fayyaz Minhas, and Nasir Rajpoot. Conic: Colon nuclei identification and counting challenge 2022, 2021.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[9] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. Cell detection with star-convex polygons. In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, pages 265–273, 2018.

[10] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[11] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023.

[12] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers, 2020.

[13] Naylor Peter Jack, Walter Thomas, Laé Marick, and Reyal Fabien. Segmentation of Nuclei in Histopathology Images by deep regression of the distance map, 2018.

[14] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.