SEMANTIC SEGMENTATION BASED METHOD FOR PLATELET QUANTIFICATION IN DENSE AGGREGATES

by

Yogeshwar Shendye

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2024

Dedicated to my Aai and Baba, whose hard work has been the encouragement and motivation behind my own efforts.

Table of Contents

List of Tables

List of Figures

Abstract

Identifying different blood components such as red blood cells, different types of white blood cells, and platelets is an important tool for health practitioners. A local company has developed a lens-less near-field microscope that can take large images of blood samples. These images are analysed with a segmentation and classification neural network to count the different components. One of the most challenging components is thereby the class of platelets, which are small components in our blood that form clots and prevents bleeding. We re-implemented the current production system to be able to experiment with different solutions and tested recent architectures such as SegFormer and MAResUNet.

Furthermore, in this thesis we report on the development of methods to improve platelet counts. In the images of blood samples, platelets are either appearing as small single entities or as aggregates where several platelets are bound together. The approach we developed uses a two-stage process where at first we train a network to classify single platelets and platelet aggregates as separate classes. We then tested several methods to separately estimate the number of platelets in the aggregate state. This included taking the number of pixels and the average size of platelets in pixel units into account. Since platelets have a characteristic signature with higher intensity in the center than the background, we also count the number of intensity peaks. We find that treating singular platelets and platelet aggregates separately better captures overall platelet counts, improving base system.

List of Abbreviations and Symbols Used

CNN Convolutional Neural Network

ViT Vision Transformer

LED Light Emitting Diode

RBC Red Blood Cell

WBC White Blood Cell

AFF AutoFocusFormer

RGB Red Green Blue

BN Batch Normalization

ReLU Rectified Linear Unit

NLP Natural Language Processing

MLP Multi-layer Perceptron

PAM Position Attention Module

CAM Channel Attention Module

Acknowledgements

I wish to acknowledge my supervisor, Dr. Thomas Trappenberg, as well as Dr. Alan Fine, Mr. Paul Hollensen and Dr. Martin Gillis for their consistent support and valuable insights, which helped guide me in my studies and the writing of this thesis. I also want to thank all members of the HAL Lab group for the conversations that led to breakthroughs throughout my thesis.

Chapter 1

Introduction

In computer vision, identifying even the smallest objects is essential when it comes to safety critical systems like medical image analysis and remote sensing. Accurate identification enables applications that require precise object counts. For instance, precise estimation of platelet counts in a complete blood count is vital for medical professionals to diagnose patient health accurately. In this study, we explore various segmentation techniques for labeling different cell types and methods of estimating count of these cells [1].

Alentic Microscience is a company, based in Halifax, Nova Scotia, which has developed a portable device that carries out the complete blood count and other in-vitro blood tests, based on lens-less microscopic imaging. Their goal is to identify blood cells such as red blood cells (RBC), white blood cells (WBC) and platelets from images. Alentic aims to identify each cell type in an image and generate labels so that every pixel corresponds to an object of interest. U-Net [2] is well known architecture for semantic segmentation. As part of this research, we experimented with various architectures, based on U-Net. These experiments were carried out by varying the model size and studying their performance for identification of blood cells. Upon investigation, it was observed that red and white blood cells are being identified precisely, whereas platelet identification needs further improvement. U-Net architectures misclassified the platelets by classifying those as background instead. This was often noticed those platelets appeared in aggregate.

In this study, we propose a method for segmentation of platelets that distinguishes between singular and aggregate platelets, unlike the existing method that labels them the same and introduces a background label between the platelets to specify the existence of multiple platelets. The proposed approach allows researchers to label platelet aggregates more efficiently compared to the existing method. This improvement is evidenced by an increase in the F-1 score for platelet detection. We also demonstrate

the efficiency of this approach by labeling larger platelet aggregates and retraining the network. This process shows that the network can effectively identify larger platelet aggregates which were previously missed. Additionally, we investigate various methods to count the number of platelets in the aggregates and discuss the results of each method in detail.

Proposed device, currently in the development stage, utilizes patented technology to analyze small blood samples, offering the potential for immediate, high-quality test results directly at the point of care. Once fully developed, this innovation aims to enable rapid testing across a range of metrics, leading to improved patient care, shorter wait times, cost reductions, and increased efficiency in healthcare—benefits that have yet to be fully realized. The prototype is designed to generate precise labels for individual blood cells, allowing for accurate counts of RBCs, WBCs and platelets. Ongoing research is focused on ensuring the accuracy of these counts, with repeated testing processes to refine the technology. Our research specifically targets the development of advanced techniques for better platelet segmentation, which is crucial for improving platelet count estimation as the device continues to evolve.

In case of platelet count, the ideal counting method, when employed for these repeated tests on a single sample, should yield results with a coefficient of variation of less than 6%, ensuring a high level of precision and consistency in the measurements. Additionally, the method should exhibit a linear relationship between the counts obtained from different samples and their true counts. This linearity would indicate that, as the actual number of items in a sample increases, the measured count should proportionally increase, thus confirming the accuracy and reliability of the method across varying sample sizes . In terms of pixelwise F-1 scores obtained throughout this research indicate that the proposed method can better identify the individual platelets. But for a clinical practitioner, estimation of platelet count is of vital importance. In this thesis, we discuss about two methods for quantification of platelets and compared those with the existing method. With these experiments, we were able slightly improve the R-2 score from 0.838 to 0.847, indicating that this method provides us with better linear relationship between the predicted and true counts. This improvement aligns precisely with Alentic's requirements, confirming that the platelet counts provided by the new method vary linearly with the actual

counts, thereby ensuring both precision and accuracy in clinical applications.

In addition, we studied models based on Vision Transformer which proposed by Google that performs comparably to Convolutional Neural Networks (CNN). Transformer architectures proposed in 2017 by Vaswani et al. [3] revolutionized natural language processing domain. Authors proposed attention mechanism that captures the context in text by dynamically assigning different weights to different parts of the input sequence. In our dataset, we observed some patterns such as platelets tend to appear individually or in aggregates, they often appear adjacent to RBCs but are never found inside a WBC. We looked at this phenomenon as a context. As attention mechanism has proved to capture the context, we experiment with vision transformer-based models such as SegFormer, AutoFocusFormer and MAResUNet. In this study we also discuss a two-stage approach for identification of platelet, where we train a separate classifier to identify only the platelets. We hypothesize that doing so will help identify the platelets that were originally misclassified.

This thesis proceeds in the following manner. In chapter 2, we discuss background, where we develop the vocabulary and discuss background of methods used throughout this thesis. In this chapter we also discuss about work carried out towards applying image segmentation for detection of blood cells. In chapter 3, we discuss experiments performed on variations of U-Net model. Here we discuss about experiments with different number of output filters for the first convolutional layer of the U-Net. In chapter 4, we talk experiments with various vision transformer-based models. In chapter 5, we talk about two-stage approach we followed. Here, we first discuss a two-stage U-Net model, where first U-Net model classifies all blood cells and the second U-Net model specializes in finding platelets. Following that, we discuss in details the approach we proposed, of treating singular and aggregate platelets different from one another. We discuss various counting methods we experimented with. In the end we discuss the conclusion.

Chapter 2

Background

2.1 Dataset

In this section we talk about the dataset used during our experiments. This dataset consists of images and their segmentation masks. We start by discussing the creation of this dataset where we briefly discuss how images in the dataset were captured and how their segmentation masks are generated. To be able to discuss unique details about this process, the device used to capture images in the dataset is also discussed in depth. After this, we describe elements in this dataset. For this, we include different types of cells that are present, different artifacts that appear in the images, such as debris, bubbles and defects which are often the result of techniques used to prepare the blood sample before capturing the image.

2.1.1 Hardware

The device used to capture images in this dataset contains a sensor chip that can capture images and a lid that can be closed. The chip is monochromatic and consists of 62 LEDs in grid of 8×8. This includes 60 RGB LEDs and 2 ultraviolet LEDs. The ultraviolet LEDs are bigger than the RGB LEDs. These are arranged to illuminate a blood sample from multiple angles. During the imaging process, a drop of blood is placed on the sensor along with a single consumable chambertop and an insert. One end of the chambertop is pressed against the drop of blood which gets rid of excessive blood sample from the perimeter of sensor. This is done to create a thin monolayer of cells ranging from 1μm to 100μm in thickness, ensuring no overlap. Spacer beads are used to prevent cell adhesion to the chambertop. To avoid direct contact between the sensors and blood, inserts are placed over the sample before closing the lid. The distance between the blood sample and insert is determined by the size of spacer bead.

Out of the 60 RGB LEDs discussed above, 12 LEDs are activated to capture images

Figure 2.1: Figure showing the Prospector device used by Alentic.

from different angles, resulting in 36 frames. The data captured in these frames can be written as $12 \times 3 \times height \times width$. These 36 frames are then integrated to produce a single super-resolution image, effectively quadrupling the image resolution. Which can further be written as $3 \times height \times 2 \times width \times 2$. Resulting data is then rearranged as $3 \times 2 \times 2 \times height \times width$. Consequently, the image has now 12 channels. Finally red, green and blue channels from the first LED from the grid of 12 LEDs are added to the super-resolution along with the ultraviolet channel, totalling to 16 channels in the end. Figure 2.1 shows the Prospector device used by Alentic. Permission of using these images was granted by Alentic.

2.1.2 Images and labels

Images in the dataset consists of multiple instances of images of blood cells such as RBCs, WBCs and platelets. RBC are disc-shaped cells essential for oxygen transport throughout the body, facilitated by hemoglobin, an iron-rich protein responsible for the blood's red color. RBCs are synthesized in the bone marrow. Deviations in RBC

count can indicate underlying health issues. For example, elevated RBC levels may be associated with issues such as dehydration and chronic obstructive pulmonary disease, whereas reduced RBC levels often signify anemia. While labelling RBCs, edge and center are assigned different classes. WBCs are essential immune system components found in the bloodstream. There are different types of WBC, each of which is specialized in fighting infection and inflammation to defend human body. Elevated WBC counts may indicate infection, inflammation, or certain cancers, while low WBC counts suggest a compromised immune system, often due to medications or immune disorders. Platelets aren't exactly cells but rather cell fragments produced in the bone marrow. They play a crucial role in the blood clotting process, which is essential for preventing excessive bleeding after an injury. A high platelet count could be a sign of the body overreacting or preparing for a potential major blood loss event.Low platelet count suggests insufficient clotting capability, raising the risk of excessive bleeding from minor injuries. This makes platelets an important indicator of patient's overall health. In the dataset, there are examples of singular platelets and platelets appearing in aggregates. For singular platelets, that are smaller in size, pixel at the center is labelled as platelet. In case of singular platelet bigger than average platelet, certain pixels inside it are labelled as platelet. In case of platelets appearing in aggregate, pixel on the edge of platelets is labelled as background. This is done to guide the model that there are multiple platelets inside the aggregate and that it's not one big platelet.

Scratches on the chambertop are reflected in the captured image like a line like structure which are dark and light in nature. On the other hand, scratches on the sensor often get filled with stains, giving the sensor scratch a blue appearance in the captured images. The spacer beads also vary in size. The most commonly observed width for the beads is around 3.3 μm. Additionally, some beads are larger than the regular ones, with several beads having a diameter of around 10 μm. The layers captured which do not contain single layer of cell are called nomonolayer. They are seen as a large structure which are dark in color.

For the labels, initially all pixels in the label are unlabelled. Using a graphical user interface tool, users can click on specific pixels to assign labels, thereby progressively building the segmentation mask. Each user interaction updates the mask,

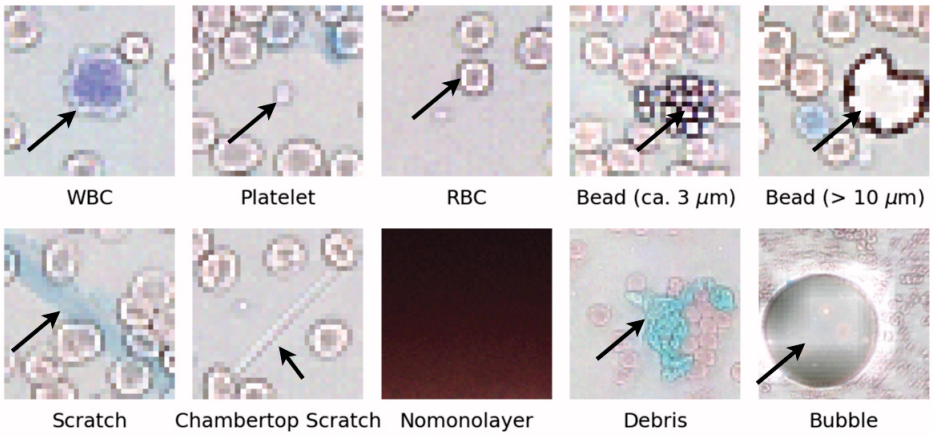


Figure 2.2: Figure showing various classes we have in our dataset. Cell or particular object is centered in the image.

transforming the clicked pixel from its initial unlabeled state to the assigned label. This method allows for precise and controlled labeling. In the end, the image and labels of the image are divided into multiple patches of height and width of 256 pixel each. As one image is of resolution 3,072 × 4,096 and one patch is of resolution 256 × 256, 192 patches are obtained from one image.

Class	Count
Background	80166
WBC	7442
Platlet	1472
RBC Edge	14506
RBC Center	2422
Bead (ca. 3μm)	749
Bead (10μm)	737

Class	Count
Scratch (Sensor)	4021
Scratch (Chambertop)	1069
Nomonolayer	125054
Debris	62161
Bubble	49966
Light Artifact	26829

Table 2.1: Table showing classwise distribution of instances of pixels.

2.2 Image Segmentation

Image segmentation is one of the most critical tasks in automatic image analysis. It consists of subdividing an image into its constituent parts and extracting these parts of interest (objects). A great variety of segmentation algorithms have been developed in the last few decades and this number continually increases each year [4]. The goal of image segmentation is to simplify the representation of an image, making it easier to analyze and extract meaningful information. One of the significant applications

of image segmentation is in medical imaging, particularly in the field of hematology, where it plays a crucial role in counting various cells present in blood images.

In the context of medical imaging, particularly for analyzing blood samples, image segmentation is exceptionally useful. Blood images contain numerous cells that require to be accurately identified and counted to diagnose and monitor various health conditions. By applying model-based image segmentation techniques, we can automate the process of counting different types of cells, such as red blood cells, white blood cells, and platelets. This automation significantly enhances the efficiency and accuracy of blood analysis, which is critical for timely diagnosis and treatment of diseases such as prostate cancer [5]. Traditional manual counting is not only time-consuming but also prone to human error, making automated methods a reliable alternative

Semantic segmentation, a specific type of image segmentation, is particularly useful in this scenario. Semantic segmentation involves classifying each pixel in an image into predefined categories or classes, without differentiating between different instances of the same class [6]. In the context of counting cells from blood images, semantic segmentation allows for the classification of each pixel as either belonging to a cell or to the background, without distinguishing between different types of cells.

By employing semantic segmentation in cell counting tasks, it becomes possible to automate and streamline the process, reducing the need for manual intervention and minimizing human error. This is especially beneficial in high-throughput environments where large volumes of blood images need to be analyzed rapidly and accurately. Additionally, semantic segmentation provides a foundation for further analysis and characterization of the segmented cells, such as determining their size, shape, and distribution within the image.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent a groundbreaking advancement in the field of deep learning, particularly in the domain of computer vision with supervised learning [7]. CNNs are engineered to automatically and adaptively learn hierarchical representations of data. This makes them stand out for tasks such as image classification, object detection, and image segmentation.

Convolutional layers are core component of a CNN. These layers apply a set of

learnable filters often known as kernels, to the input data. When convolved with the input image, these layers produce feature maps that capture local patterns and spatial relationships [8]. Through the process of training, the network learns to adjust the parameters of these filters to extract relevant features from the input data.

CNN architectures typically consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, sometimes called dense layers. Convolutional layers extract local features from the input data, while pooling layers downsample the feature maps, reducing their spatial dimensions and the computational complexity of the network. Fully connected layers aggregate the extracted features and perform high-level reasoning, ultimately producing the final output [9].

One of the key advantages of CNNs is their ability to automatically learn hierarchical representations of features. In the context of image processing, lower layers of the network learn to detect simple patterns such as edges and textures, while higher layers learn to detect more complex features and object representations. This hierarchical feature learning enables CNNs to achieve remarkable performance on various visual recognition tasks [10].

CNNs have demonstrated state-of-the-art performance in a wide range of applications, including image classification, object detection, facial recognition, medical image analysis, and natural language processing. Their success can be attributed to their ability to effectively capture spatial dependencies and hierarchical structures in data, as well as their capacity for end-to-end learning from raw input [11].

Moreover, CNNs have been instrumental in advancing the field of computer vision by enabling the development of more accurate and efficient algorithms for tasks such as image segmentation. Semantic segmentation, in particular, has benefited greatly from CNN-based approaches, with architectures like U-Net and DeepLab consistently achieving top performance on benchmark datasets [12].

2.3.1 Encoder-Decoder Architecture

Encoder-Decoder models form a category of models that acquire the capability to translate data points from one domain to another through a dual-phase network: The encoder, symbolized by an encoding function $z = f(x)$, condenses the input into a latent-space representation; the decoder, denoted by $y = g(z)$, endeavors to forecast the output, based on the latent space representation. The latent representation in this

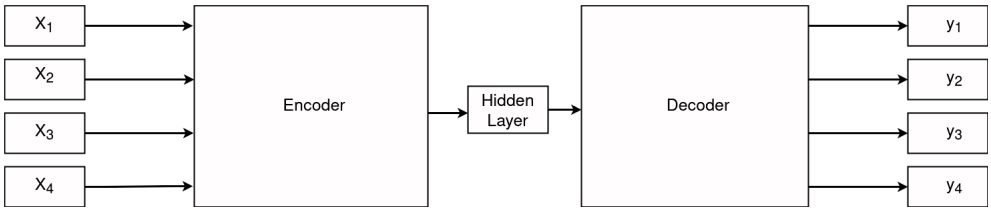


Figure 2.3: Figure depicting working on encoder decoder architecture for 4 input features. X is the input vector and y is the output vector.

context essentially denotes a feature vector representation adept at encapsulating the underlying semantic information of the input, which proves instrumental in output prediction. These models enjoy widespread popularity in tasks like image-to-image translation and sequence-to-sequence tasks in natural language processing (NLP) [13]. Figure 2.3 represents a basic encoder-decoder architecture.

2.3.2 U-Net

U-Net is a convolutional neural network (CNN) architecture, specifically designed for the task of biomedical image segmentation. It was introduced by Ronneberger, et al. in 2015 and has since become a cornerstone in the field due to its distinctive and effective design [2]. The primary motivation behind U-Net's development was to create a model that could accurately segment medical images, such as those from MRI or CT scans, where precise localization and boundary delineation are crucial. The architecture is particularly adept at handling the limited availability of annotated training data, which is a common challenge in medical imaging. Figure 2.4 depicts the architecture that has been followed throughout our studies.

U-Net's architecture is symmetrical, mirroring the structure of an encoder-decoder design. The encoder is a traditional convolutional network that applies successive convolutional and pooling layers to downsample the input image, thereby capturing increasingly abstract and complex features at smaller spatial resolutions. This process results in a latent-space representation that encodes the essential characteristics of the input image. The decoder conversely, uses upsampling layers and convolutional layers to incrementally restore the spatial dimensions of the feature maps, converting the latent representation back into an output image of the same size as the input. One of the key innovations of U-Net is the incorporation of skip connections, which directly link corresponding layers in the encoder and decoder. These connections allow

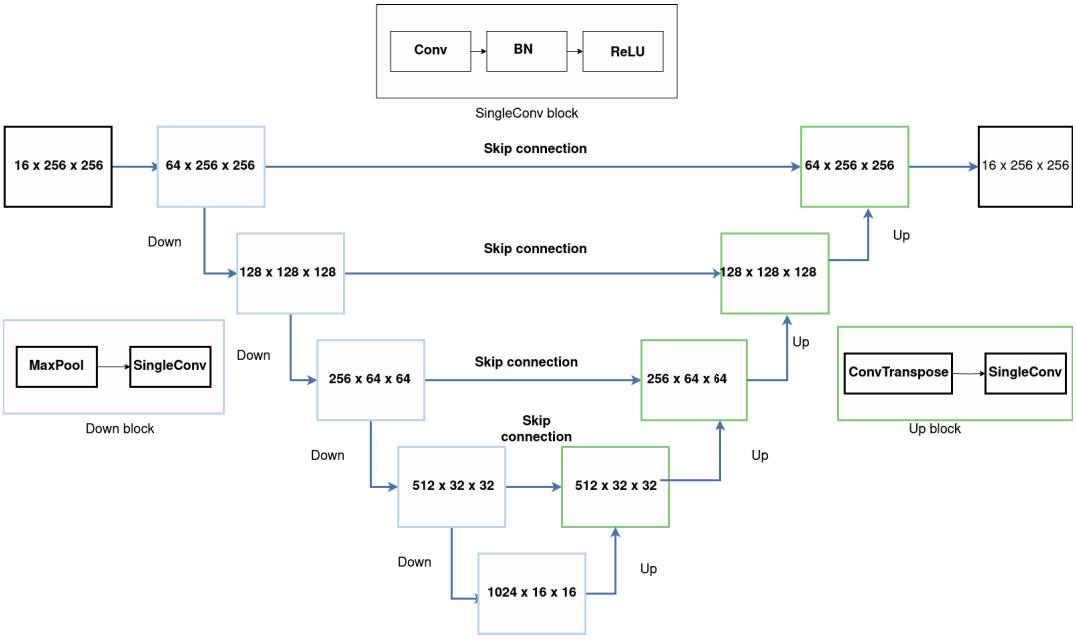


Figure 2.4: Figure depicting the architecture of U-Net model we have used for our experiments. Conv represents convolutional block, BN represents batch normalization and ReLU represents rectified linear unit. MaxPool represents 2 dimensional maxpooling operation. ConvTranspose represents 2 dimensional transposed convolutional block. SingleConv is the modified building block of U-Net model. This block consists of Conv followed by BN followed by ReLU. In traditional U-Net model, this is repeated for 2 times (Adapted from [2]).

the model to utilize fine-grained information from the encoder, which significantly enhances the accuracy and quality of the segmentation by preserving spatial details.

U-Net's versatility and efficacy extend beyond biomedical applications. It has been successfully applied in image segmentation tasks such as satellite image analysis, where it helps in identifying geographical features. Applications can also be found in industrial inspection, where U-Net detects defects in manufacturing processes. The ability of U-Net to handle different types of image data and produce high-quality segmentation results with limited training data has made it a widely adopted tool in both academic research and industry. Its design principles have also influenced the development of numerous subsequent models, underscoring its impact on the broader field of computer vision.

2.4 Attention Mechanism

Attention mechanism is a fundamental concept in modern neural network architectures, particularly within the domain of natural language processing (NLP) and computer vision. It was designed to enhance the model's ability to focus on relevant parts of the input data. Vaswani et al. [3] introduced attention, originally for task of machine translation.Attention mechanism allows a model to dynamically prioritize different parts of the input sequence. Instead of treating all input elements equally, the model learns to assign varying levels of importance to different parts, enabling it to concentrate on the most relevant segments. From a given input, queries, keys and values are constructed. Query can be looked as current point under consideration, key and value are both contents of the input sequence. For queries, keys and values represented as Q, K and V, attention scores are calculated as,

$$Attention(Q, K, V) = softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V \tag{2.1}$$

Where, $\sqrt{d_k}$ is the size of the key, and Q, K and V are assumed to be of the same dimensions. Division by the factor of $\sqrt{d_k}$ is ensures that exploding and vanishing gradient problem is not observed. It is obtained by dividing the size of embedding dimension by the number of attention heads.

Attention scores computed by the equation above indicate the relevance of each key relative to the query. These scores are normalized using a softmax function

to make sure that attention weights produced by them sum to one. The attention weights are then used to compute a weighted sum of the value vectors, effectively producing a context vector that emphasizes the most relevant parts of the input.

One of the most powerful aspects of the self-attention mechanism is the concept of multi-headed self attention. Instead of performing a single self attention function with d_{model}-dimensional keys, values, and queries, the model linearly projects the queries, keys, and values h times with different, learned linear projections to d_k, d_k, and d_v dimensions, respectively. These projections are concatenated and once again projected, resulting in the final values. This allows the model to jointly attend to information from different representation sub-spaces at different positions. With a single attention head, averaging inhibits this.

Mathematically, this can be given as

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h) \times W^0 \tag{2.2}$$

2.4.1 Vision Transformer

Dosovitskiy et al. proposed Vision Transformer (ViT) as a novel architecture that leverages the self-attention mechanism from natural language processing to process image patches. They proposed that this approach offers a promising alternative to traditional convolutional neural networks in the field of computer vision [14]. The standard Transformer receives a 1D sequence of token embedding as the input. To handle 2 dimensional images, image of shape $R^{(H\times W\times C)}$ is first transformed into a sequence of flattened 2 dimensional patches $X_p \in R^{N\times(P^2\cdot C)}$, where (H, W) is the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and number of patches N is calculated as $N = \frac{H\cdot W}{p^2}$ is the resulting number of patches, which also serves as the effective input sequence length for the Transformer. These patches are then flattened and mapped to D dimensions with a linear projection given as,

$$Z_0 = [X_{class}; {X_p}^1 E; {X_p}^2; ...{X_p}^n E] + E_{pos} \tag{2.3}$$

where $E \in R^{(P^2\cdot C)\times D}$ and $E_{pos} \in R^{(n+1)\times D}$.

Original transformer architecture was designed for sequence to sequence networks. As there is no decoder in ViT, length of input sequence becomes length of output

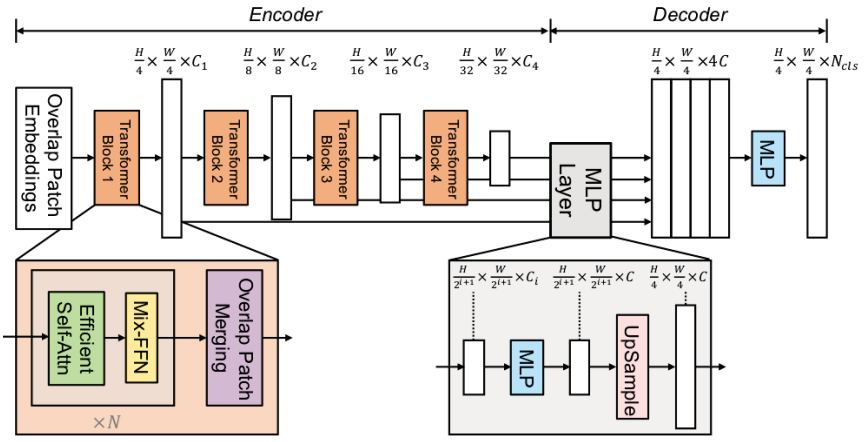


Figure 2.5: Figure demonstrating architecture of SegFormer model (Adopted from [15]).

sequence. For classification, a class token is appended on the sequence and classification head is applied on it to make sure that it is not biased on a particular token. The classification head is a Multi-layer Perceptron (MLP). The encoder part of ViT makes use of of multi headed self attention. Layer normalization is applied before each block and there are residual connections after every block. This model breaks the image into patches and process them independently. While the model learns relationships between patches later, it doesn't have the same built-in assumptions about their locality or 2 dimensional arrangement like CNNs. This gives ViTs more flexibility but may require more data to learn spatial reasoning.

2.4.2 SegFormer

SegFormer is a state-of-the-art model specifically designed for image segmentation tasks, integrating advanced features to enhance performance and flexibility. Traditional segmentation models often struggle with maintaining performance across different resolutions and require complex architectures to handle multi-scale features. SegFormer addresses these challenges by introducing a novel approach that combines a hierarchical Transformer encoder and a simplified MLP decoder. Figure 2.5 shows the architecture of SegFormer model.

Novel Hierarchical Transformer Encoder

One of the standout features of SegFormer is its novel hierarchically structured transformer encoder. SegFormer's encoder does not require positional encoding, which

often causes performance degradation when training and testing resolutions are different. This design choice allows SegFormer to maintain high performance across various test resolutions, ensuring adaptability and robustness. The absence of positional encoding eliminates the need for interpolation of positional codes, which can otherwise negatively impact performance.

The hierarchical structure of the encoder allows the generation of both high-resolution fine features and low-resolution coarse features. This contrasts with models like ViT, which are limited to producing single low-resolution feature maps with fixed resolutions. The ability to produce multi-scale features enables SegFormer to capture a richer set of information from the input images, leading to more accurate and detailed segmentation outputs.

Lightweight MLP Decoder

In addition to the innovative encoder, SegFormer features a lightweight MLP decoder. The key innovation here lies in its ability to leverage the diverse attention patterns induced by the Transformer. The attention mechanisms in the lower layers of the Transformer tend to remain local, focusing on fine-grained details, while those in the higher layers are more global, capturing broader contextual information.

By aggregating information from these different layers, the MLP decoder successfully combines local and global attention, resulting in powerful and comprehensive representations. This approach not only simplifies the decoder architecture but also enhances the overall performance of the model. The simplicity of the MLP decoder, in contrast to more complex decoders, contributes to the efficiency of the model while maintaining high accuracy and robustness in segmentation tasks.

2.4.3 AutoFocusFormer

AutoFocusFormer(AFF) [16] is a local attention transformer backbone for image recognition. It achieves adaptive downsampling by learning to preserve the most crucial pixels for the task. Uniform downsampling is less effective for tasks that require pixel-level details, for example, segmentation. Uniform downsampling tends to make tiny objects even smaller, which causes loss of crucial pixel-level information. One technique that solve this is increasing the input resolution to achieve better segmentation performance. While this approach intuitively helps by producing a higher

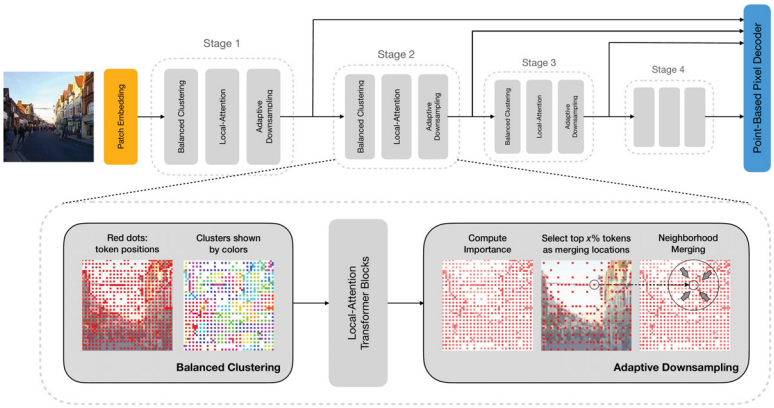


Figure 2.6: Figure depicting the architecture of AutoFocusFormer model (Adopted from [16]).

resolution after downsampling, it is costly in terms of memory and computation, as it doesn't address the underlying issue of uniform downsampling. Figure 2.6 shows the architecture of AutoFocusFormer.

Thus AutoFocusFormer uses adaptive downsampling instead of uniform downsanpling that retains maximum features from important areas and summarizes the features from areas which do not have a specific texture. This is first end to end segmentation network to use successive adaptive downsampling. For feature masks with high resolution, instead of using the global attention which would be computationally expensive, local attention blocks are used. These local attention blocks are defined using a balanced clustering algorithm which groups the irregular points into equal sized neighborhoods using space filling curves. The adaptive downsampling module also learns to calculate importance score, which is used in the process of merging neighborhoods. The segmentation heads are also modified for irregularity in representation space.

2.4.4 MAResUNet

Dot-product attention mechanisms are crucial for many advanced vision and language tasks. They help capture long-distance relationships by covering the entire input in one go. This ability is key to the success of models like transformers in NLP and non-local modules in computer vision, allowing them to perform exceptionally well on

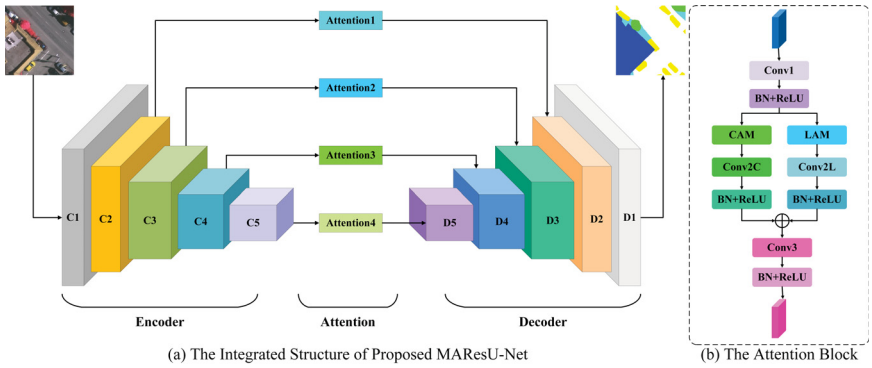


Figure 2.7: Figure depicting the architecture of MAResUNet (Adopted from [17]).

many tasks. However, the computational and memory requirements of dot-product attention increase significantly with the size of the input, making it challenging to handle large-scale data like long video sequences, lengthy texts, or high-resolution images. To address these issues, researchers have developed various strategies. For instance, [18] designed sparse factorization to lower complexity to $O(N\sqrt{N})$ [19], employed locality-sensitive hashing to achieve $O(N \log N)$ complexity, and others have reformulated self-attention mechanisms to achieve linear complexity, $O(N)$. Figure 2.7 shows the architecture of this model.

Expanding on these advancements, MAResUNet aims to reduce the complexity of dot-product attention mechanisms to $O(N)$ while enhancing U-Net architecture performance. This method integrates a novel linear attention mechanism and utilizes ResNet-34 as the encoder. It also replaces the traditional skip connections of the standard U-Net with attention blocks at multiple stages, refining the multi-scale feature maps captured by the network. Utilizing encoder decoder structure of U-Net, where the encoder captures low level, fine grained details and the decoder reconstructs high-level, coarse-grained semantic information, MAResUNet enhances the connectivity between these components.

2.5 Related Works

In this section we discuss studies about applying image segmentation on cellular images. In the field of cellular biology, image segmentation plays a pivotal role by enabling the detailed analysis of cellular structures. Accurate segmentation of cellular images is essential for various applications, such as disease diagnosis. However,

segmenting cellular images poses unique challenges. Cellular images often exhibit significant variability in cell shapes and sizes, overlapping and touching cells, low contrast, and the presence of noise. These complexities necessitate robust and sophisticated segmentation techniques that can accurately delineate cellular components. This section aims to review the significant contributions and advancements in the field of image segmentation as applied to cellular images.

2.5.1 Deep Learning Segmentation and Classification of Red Blood Cells Using a Large Multi-Scanner Dataset

Elmanna et al. compiled a comprehensive dataset comprising 25 manually prepared and stained peripheral blood and bone marrow smears. These smears were scanned using four different digital pathology scanners. Each slide, divided into approximately 2,000 patches, results in over 100,000 images along with their corresponding masks. At the first stage, images were classified into eight clinically significant RBC categories [20].

For the segmentation stage, a U-Net model was trained on the this dataset to develop precise segmentation masks for the RBC images. Subsequently, an EfficientNetB0 model was trained on the same dataset for classifying the segmented RBC images. This two-stage approach ensures robust segmentation and accurate classification, thereby enhancing the diagnosis and study of hematological conditions.

The research underlines the clinical importance of accurately labeling and classifying RBCs, focusing on types like ovalocytes, teardrop shaped RBCs, and fragmented RBCs, which are crucial for diagnosing various types of anemia and other severe medical conditions. Authors of this paper have demonstrated the integration of advanced deep learning models with expert-annotated large datasets to improve the precision and dependability of RBC image analysis.

The major difference in this work and what we are doing is the blood cell which is emphasized. [20] have extensively focused on detection of RBCs whereas in our study, we have focused largely on identification of platelets. Similarly, instead of capturing images from multiple sensors like Elmanna et al., we have kept the source of image the same. Apart from the differences, this paper has discussed the implementation of a two-stage classifier, which we have discussed in Chapter 5.

2.5.2 Automated Complete Blood Cell Count and Malaria Pathogen Detection Using Convolution Neural Network

The complete blood cell count, that measures the density of different blood cells in the human body, is crucial for assessing overall health and detecting various disorders such as anemia, infection, and leukemia. Automating this task can significantly speed up diagnosis and reduce treatment costs. This paper presents a CNN approach for performing CBC on blood smear images and detecting malarial pathogens if present [21]. The images used for training, testing, and validation were sourced from four databases:

1. Leukocyte Images for Segmentation and Classification (LISC), that contains images of five types of WBCs with their binary masks.

2. Isfahan University of Medical Science (IUMC) database, which has labeled images of individual WBCs, except for Basophils, along with binary masks.

3. MAMIC database, containing large blood smear images with healthy RBCs, thrombocytes (THRs), platelet clumps, and malaria-infected RBCs, though without labeled WBCs.

4. A dataset from KAGGLE that features images of individual healthy and infected RBCs without binary masks.

A key challenge the authors faced was the lack of a single annotated database containing all blood cell types and malaria infected RBCs. To address this, a new dataset was created using images from the databases mentioned above as building blocks. The creation process involved cropping individual cells and creating binary masks for them. The dataset included a wide variety of blood cell combinations to ensure robust CNN training.

The resultant dataset consisted of images of resolution 224×224, with various combinations of different cell types. The training set includes images with different combinations. All WBC types were combined into a Partial WBC class to avoid CNN bias towards specific types. For example, if the image contains an Eosinophil or Basophil they are all labeled simply as Partial WBC without specifying the WBC

type. Some images contain multiple cell types, such as Eosinophils and Platelet clumps.

Additionally, there are 4,000 images with a mix of infected and healthy RBC. The testing and validation sets are each about one-tenth the size of the training set but maintain the same distribution of image types. The total number of images is 65,350 for training, 6,560 for testing, and 6,560 for validation. Basophil images are fewer because the LISC database lacked Basophil images. Despite this, Basophil detection performs well, as demonstrated in the experimental results.

The CNN was trained to detect and classify these cells, achieving a mean average precision exceeding 0.95 when compared to the ground truth. The system exhibited perfect accuracy, detecting all malaria-infected images. The segmentation stage used the U-Net architecture for its proven efficiency in medical image segmentation. U-Net's contracting and expansive paths handle the complexities of cellular image data efficiently. The EfficientNetB0 model was employed for classification, leveraging transfer learning from the ImageNet dataset.

In conclusion, the authors proposed extensive dataset and the sophisticated image processing pipeline that demonstrated high accuracy and efficiency in diagnostics in automated CBC and malaria detection from blood smear images. This study used images of resolution 1000×1000 that only contained examples of RBCs. Authors then discuss about using crops of dimension 224×224 for training, testing and validation. This study has discussed implementation of YOLO9000 model [22] for performing object detection on blood cells. Goal of our study is use image segmentation to label every cell in the given blood cell image. For this the image resolution being used is 3072×4096, which is bigger than 1000×1000 pixels. In our study, we are also interested in identifying the count of platelets which are 1×1 pixels in dimension at the smallest. Similarly, in the images of dimension 3072×4096, there are tens of thousands of RBCs. Therefore using object detection for this purpose becomes a tedious task. Therefore, in our research we have extensively experimented with image segmentation.

Chapter 3

Exploring Experimental Setups with U-Net Variants for Blood Cell Segmentation

In this chapter, we discuss the U-Net architecture developed by Alentic for labeling blood cells. Building on their implementation, we conducted various experiments on the provided dataset by altering architectural parameters to optimize the model's performance. We begin by discussing the building blocks of this model, which are the basis for U-Net variations using which the experiments were performed. Following that we discuss the training for each of these architectures. We discuss hyperparameter tuning such as learning rate, weight decay and optimizer functions. The loss function we used throughout our experiment is also discussed in this section.

3.1 Architecture of U-Net model

A block of convolutional layer followed by batch normalization layer followed by rectified linear unit (ReLU) is a core block of U-Net architecture. This convolutional layer uses the kernel size of 3 with padding of 1. Conventional U-Net architecture repeats this block for 2 times. For all our experiments, the number of input channel used is always 16. Throughout the experiments, number of downsampling and upsampling layers used is 4. Downsampling layers reduce the dimensions of the input image and increase the number of output filters. Downsampling layer starts with MaxPooling layer with kernel size of 2. Upsampling on the other hand increases the dimension of input and reduce the number of output filters using 2D- transposed convolution. To make sure that the features learned by downsampling layers are preserved, upsampling layer concatenates the outputs produced by each downsampling with the output of corresponding output of upsampling layer. Downsampling factor of 2 is used for all the experiments. To change the number of trainable parameters, number of output filters are changed from the very first layer.

3.2 Experiments

Convnetional U-Net architecture has the number of output filters set to 64. The first layer of U-Net takes 16 input channels and outputs 64 output channels using 64 filters. [23] discussed the effects of increasing filters in improving model performance. First downsampling layer increases the number of output channels from 64 to 128, second does the same from 128 to 256, third layer goes from 256 to 512 and the fourth one increases the number of output channels to 1024. To maintain the features learned by each of these downsampling layers, residual connections are used. Upsampling layer increases the dimension and at the same time reducing number of filters.

Based on number of output filters of the first convolutional layers, model architectures are named. U-Net16, U-Net32, U-Net64 and U-Net128 corresponds to the U-Net architectures with 16, 32, 64 and 128 output channels by the first convolutional layers respectively. Another experiment was performed with U-Net architecture in which the core block was repeated only once instead of twice. In this model, the first convolutional layer outputs 64 channels. This model is named U64-SL and it is the architecture that was trained by Alentic to label blood cells. We proceed by first discussing about U64-SL, followed by U-Net16, U-Net32, U-Net64 and finally U-Net128. All the models discussed in this section were trained and evaluated on the same training and validation dataset. As we are dealing with tiny objects and class imbalance, weighted focal loss is used. Number of epochs the model has been trained for is 40. Table 3.1 discusses the hyperparameters used for training of these models.

Hyperparameters	U64-SL	U-Net16	U-Net32	U-Net64	U-Net128
paramters(Count)	15,336,144	1,944,704	7,767,280	31,046,096	124,138,384
Optimizer	SGD	RMSProp	RMSProp	RMSProp	SGD
Learning rate	1e-02	1e-04	1e-04	1e-04	1e-02
Weight decay	1e-03	1e-06	1e-06	1e-05	1e-03
momentum	0.0	0.9	0.9	0.9	0.9

Table 3.1: Table describing hyperparamters used for training different U-Net architectures.

3.3 Results of experiments on base dataset

For purpose of testing models trained on various datasets we evaluate them on an image captured by the device which is of resolution 3,072 $\times$ 4,096. As this resolution is

larger than what GPU can handle, image was split in four sub-parts: upper left, upper right, bottom left and bottom right. For a model to generate precise segmentation we make sure that these sub-parts are overlapping. After the model generates output segmentation maps, we get rid of the extra padding. This makes sure that even after evaluating on sub-parts the model does not lose context over the edges.

In the realm of machine learning and artificial intelligence, the performance of models is paramount. Various metrics are employed to evaluate and compare the effectiveness of these models, among which the F-1 score stands out as a crucial indicator. The F-1 score, which is the harmonic mean of precision and recall, provides a balanced measure of a model's accuracy. This metric is particularly valuable in scenarios where the classes are imbalanced, ensuring that both false positives and false negatives are accounted for effectively. We discuss the performance of various models by presenting their F-1 scores. Doing so offers a comprehensive comparison that highlights the strengths and weaknesses of each model in handling different datasets and tasks. The F-1 score serves not only as a benchmark for evaluating model performance but also as a guide for selecting the most suitable model for specific applications. Through this analysis, the goal is to provide insights to identify which of the blood cells or artifacts require improvement in identification. Table 3.2 shows the comparison of classwise F-1 scores computed for each of the models we have experimented with. We calculated standard deviation using cross validation across 5 folds. Looking at the F-1 scores computed we can say all the models including U-Net16, U-Net32, U-Net64, U-Net128 and $U64 - SL^{1,2}$ are statistically similar in the performance for identification of RBCs and WBCs. This table also shows the F-1 scores $U64 - SL^{3}$ and $U64 - SL^{4}$, which correspond to our proposed method of treating singular and aggregate platelets differently. This is further explained in Chapter 5. Looking at F-1 scores, it can also be inferred that F-1 scores are always lowest for platelets. It can be observed in the table that in models U-Net16, U-Net32, U-Net64 and U-Net128, with increasing model complexity, F-1 scores tend to increase. U64-SL model having lesser computational complexity than U-Net64 and U-Net128 models have better F-1 scores for identification of platelets.

Below the table, figures 3.1 and 3.2 show sample segmentation masks generated by U-Net16, U-Net32, U-Net64, U-Net128 and $U64 - SL^{2}$ are shown. To put emphasis

on platelet detection, segmentation masks generated for platelets are also shown.

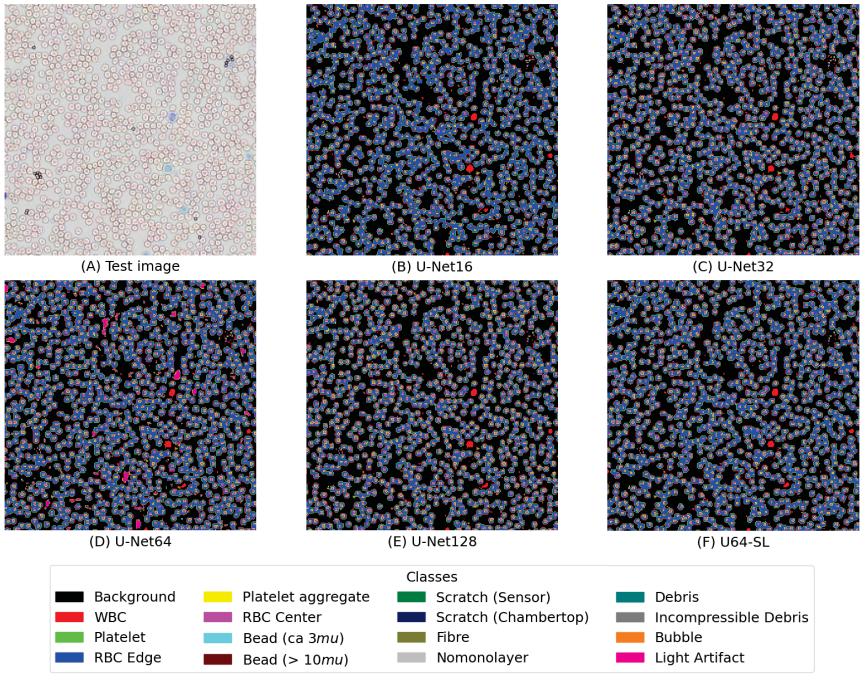
Class	**U-Net16**	**U-Net32**	**U-Net64**	**U-Net128**
Platelet	0.332 ± 0.158	0.304 ± 0.152	0.655 ± 0.187	0.745 ± 0.043
RBC Edge	0.757 ± 0.211	0.721 ± 0.218	0.896 ± 0.152	0.949 ± 0.007
RBC Center	0.631 ± 0.137	0.527 ± 0.222	0.867 ± 0.083	0.900 ± 0.015
WBC	0.805 ± 0.104	0.821 ± 0.079	0.862 ± 0.092	0.939 ± 0.017
Bead (ca. 3μm)	0.624 ± 0.150	0.543 ± 0.177	0.877 ± 0.068	0.880 ± 0.039
Background	0.819 ± 0.178	0.745 ± 0.284	0.744 ± 0.256	0.962 ± 0.010
Scratch (Sensor)	0.506 ± 0.277	0.475 ± 0.246	0.597 ± 0.238	0.906 ± 0.095
Nomonolayer	0.984 ± 0.039	0.986 ± 0.022	0.987 ± 0.021	0.998 ± 0.001
Debris	0.917 ± 0.084	0.919 ± 0.061	0.893 ± 0.114	0.981 ± 0.011
Bubble	0.793 ± 0.263	0.855 ± 0.239	0.802 ± 0.271	0.999 ± 0.001
Light Artifact	0.748 ± 0.192	0.732 ± 0.240	0.704 ± 0.245	0.949 ± 0.023

Class	**U64-SL**	**U64-SL^2**	**U64-SL^3**	**U64-SL^4**
Platelet	0.889	0.874 ± 0.027	0.909 ±0.031	0.969 ± 0.010
RBC Edge	0.975	0.968 ± 0.004	0.971 ±0.007	0.977 ± 0.005
RBC Center	0.946	0.918 ± 0.009	0.936 ±0.019	0.950 ± 0.015
WBC	0.974	0.968 ± 0.006	0.968 ±0.009	0.977 ± 0.008
Bead (ca. 3μm)	0.917	0.944 ± 0.007	0.920 ±0.038	0.957 ± 0.030
Background	0.986	0.974 ± 0.014	0.979 ±0.007	0.981 ± 0.002
Sensor Scratch	0.984	0.916 ± 0.051	0.911 ±0.061	0.958 ± 0.017
Nomonolayer	0.998	0.999 ± 0.001	0.999 ±0.001	0.999 ± 0.000
Debris	0.996	0.993 ± 0.002	0.993 ±0.004	0.996 ± 0.003
Bubble	0.999	0.999 ± 0.001	0.997 ±0.002	0.997 ± 0.001
Light Artifact	0.983	0.946 ± 0.041	0.977 ±0.016	0.979 ± 0.012

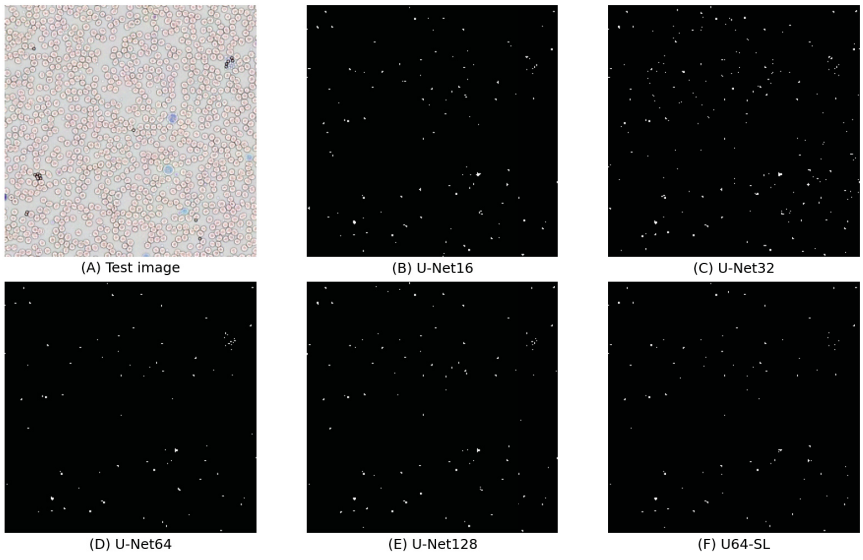
Table 3.2: Table showing classwise F-1 scores of U-Net models with varying filter sizes. Values in platelet aggregate class are all 0 because for these models we did not have any examples for it. U64-SL refferes to the U-Net models with starting number of filters are 64 and the layer containing Conv, BN, ReLU is repeated one time. U64-SL is the model currently being used by Alentic, which was considered as baseline model and thus does not have standard deviations in the table. $U64-SL^2$ is the re-implementation of the U64-SL model. $U64-SL^3$ is the U-Net model trained on the dataset in which we differentiate single platelet and platelet aggregates. $U64-SL^4$ represents the F-1 scores computed after expanding dataset used for training $U64-SL^3$ by labelling more platelets appearing in aggregates.

3.4 Conclusion

In this chapter, we experimented with various U-Net architectures with the primary goal of matching the performance of the existing system used at Alentic. We tested

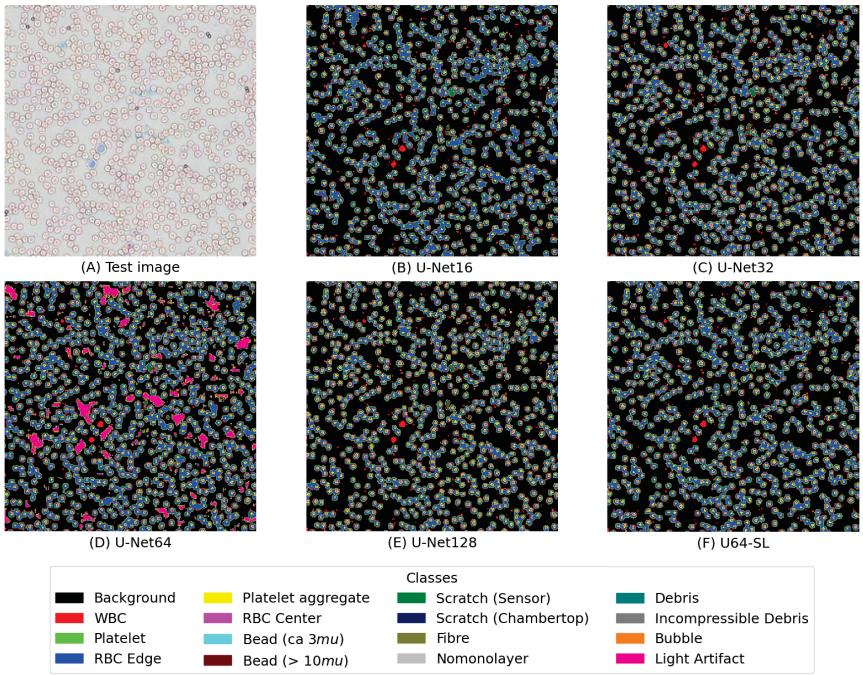


(a) Figure showing segmentation map generated by 5 different models.

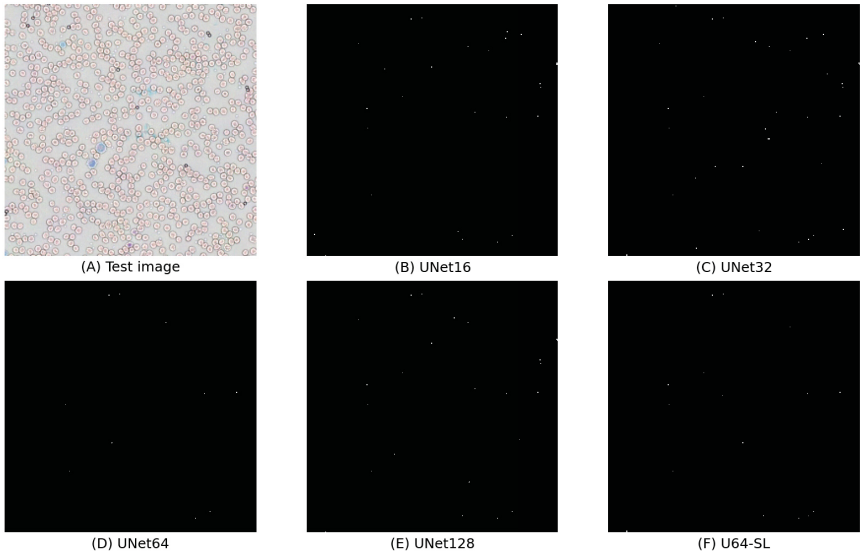


(b) Figure showing platelets in segmentation generated by 5 different models.

Figure 3.1: Figure showing example 1 of segmentation map for all cells showing in Figure a and for all classes in Figure b for platelets.



(a) Figure showing segmentation map generated by U-Net model specialized in identifying platelets.



(b) Figure showing platelets in segmentation generated by 5 different models.

Figure 3.2: Figure showing example 2 of segmentation map for all cells showing in Figure a and for all classes in Figure b for platelets.

different configurations, specifically U-Net64, U-Net128, and U64-SL models. Based on our experiments, we concluded that these models performed similarly in terms of overall effectiveness. As shown in Table 3.2, these models efficiently detect blood cells such as RBCs and WBCs. Additionally, they can differentiate artifacts such as beads, debris, bubbles, and light artifacts from the blood cells.

However, while the U-Net64, U-Net128, and U64-SL models excelled in RBC and WBC detection, there remains a need for further improvement in platelet detection. Platelets are smaller and less distinct than other blood cells, posing a greater challenge for accurate identification. Enhancing the ability of model to detect platelets is essential as it is often used as indicator of human health. In next chapters we discuss step taken towards improving platelet detection and different counting techniques for counting number of platelets.

Chapter 4

Experiments with transformer-based architectures

In the blood images, platelets often occur individually or stuck together in an aggregate. But they are never found to be inside another blood cell. We looked at this phenomenon as a context. Transformer models are good in capturing context. Thus, we hypothesized that transformer-based models would be a better fit for labelling various types of blood cells. In this section, we discuss experiments on various transformer-based architectures. For this we discuss models like SegFormer, AutoFocusFormer and MAResUNet. For each of these models, we first discuss model-based hyperparameters that were used. We also discuss training related hyperparameters such as optimizer and learning rate. Then we discuss the results and our findings in results section. For all experiments in this chapter, focal loss has been used, with the weights calculated as square root of inverse frequency of number of instances per class in order to treat class imbalance.

4.1 SegFormer

Architecture of SegFormer consists of two modules. First module is a hierarchical transformer encoder that generates high resolution coarse features and low resolution fine features, and an MLP decoder that merges the encoded features to generate fine resolution segmentation map. Given an image, SegFormer divides it in multiple patches of 7×7, whereas [3] uses patches of 16×16. Authors hypothesize that smaller patch size is beneficial for dense prediction task. On these patches, hierarchical transformer encoder obtains multilevel features at $[\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}]$. MLP decoder predicts segmentation mask with resolution $\frac{H}{4} \times \frac{W}{4}$ with channels equaling to number of classes.

4.1.1 Hierarchical Transformer Encoder

SegFormer uses hierarchical feature representation to generate multi level features like a CNN. Given An image of resolution H $\times$ W $\times$ C, it generates features of resolution

$\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}} \times C_i$, where $i = \{1, 2, 3, 4\}$. For our experiments with SegFormer, we configured the model with specific parameters to optimize performance. The reduction ratios of the self-attentions were set to 8, 4, 2, and 1 for each successive stage, respectively. Correspondingly, the head numbers of the self-attentions were configured to 1, 2, 5, and 8. The feed-forward network expansion ratios were adjusted to 8, 8, 4, and 4 for each stage. Furthermore, the number of encoder layers was consistently set to 2 for all four stages. This configuration is aimed to balance the computational efficiency and the model's capability to capture hierarchical features across different scales.

4.1.2 Decoder

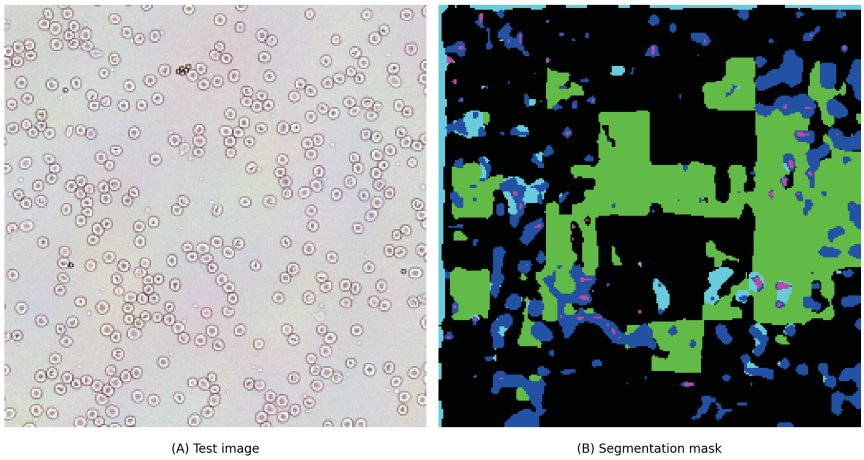
SegFormer proposed a lightweight decoder that only consists of MLP layers. Larger receptive field of SegFormer encoder enables this. Multi-level features F_i from encoder goes through MLP. This unifies channel dimensions. These unified features are then up-sampled to $\frac{1}{4}^{th}$ and then concatenated. These are then fused to F. Finally, another MLP uses these fused features to predict segmentation mask.

4.1.3 Training

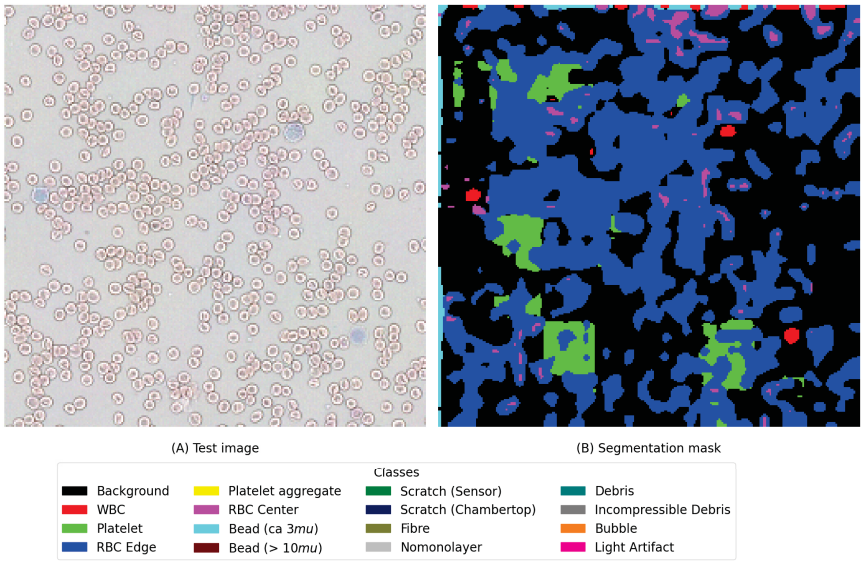
We used hidden dimensions as $[32, 64, 160, 256]$. Number of attention heads were $[1, 2, 5, 8]$. We use 2 layers of encoder. Decoder dimension is 256. Sequence reduction ratio is $[8, 4, 2, 1]$. Patch size we use is $[7, 3, 3, 3]$, with strides $[4, 2, 2, 2]$ and padding of $[3, 1, 1, 1]$. The batch size of 16 is used throughout the experiments. To study if the model performance improves with increasing number of iterations we experiment with number of epochs to be 100, 200 and 500. As the SegFormer outputs masks that are downsampled by the factor of 4, to evaluate predictions, bilinear interpolation was used to restore the dimensions of segmentation mask.

4.1.4 Results

With the patch size of 7×7, SegFormer model outputs the segmentation masks that are downsampled when compared with input resolution. In our experiments, the input images were of dimension 256×256, the output segmentation masks were of dimension 64×64. To produce segmentation masks of dimension 256×256, interpolation with factor of 4 was used.



(a) Figure showing segmentation map generated by SegFormer model.



(b) Figure showing segmentation map generated by SegFormer model.

Figure 4.1: Figure showing segmentation map generated by SegFormer model.

Figure 4.1a shows a sample image and the output segmentation produced by the SegFormer model. The image indicates that the model is largely misclassifying background as platelet. It can be also seen that model is identifying some of the RBC edge and RBC center. As for beads, model identified the beads but also misclassified some of RBCs as beads. In Figure 4.1b it can be seen that model has identified the WBC. In case of RBCs the model did not identify RBC edge and RBC center separately. It can also be seen in this figure that background is being misclassified as platelets.

SegFormer model processes in four stages before the classification head. In the

first stage it uses the patch size of 7×7. For blood cell images, where a tiny cell might be 2×2 in dimension. Thus, we hypothesize that changing the patch size of SegFormer model from 7×7 to a smaller patch size would be beneficial. Throughout our experiments with U-Net, kernel size of 3×3 was used. Thus to keep consistency, patch size of 3×3 was used for SegFormer, which increased the computational complexity of the process. Therefore batch size of 4 was used for training. This model was trained with Adam optimizer with initial learning rate of $1e-04$ with learning rate decay by the factor of 0.98 per epoch. This model was trained for 40 epochs. Table 4.1 shows the F-1 scores obtained from SegFormer model compared with U-Net variants. Below the table, we also discuss sample segmentation maps generated by this model. Upon analyzing the masks generated by the SegFormer model with a 3×3 patch size, the results appeared to be comparable to the performance of U64-SL model, indicating the effective segmentation. However, when examining the masks produced by the SegFormer model with a 7×7 patch size, it was evident that the model's performance was suboptimal, demonstrating a significant decline in segmentation accuracy and quality. Thus we only show F-1 scores calculated for SegFormer model with patch size 3×3.

Class	**U64-SL**	**SegFormer**
Platelet	0.889	0.444 $\pm$0.068
RBC Edge	0.975	0.889 $\pm$0.028
RBC Center	0.946	0.670 $\pm$0.131
WBC	0.974	0.827 $\pm$0.011
Bead (ca. 3μm)	0.917	0.706 $\pm$0.060
Background	0.986	0.915 $\pm$0.011
Sensor Scratch	0.984	0.627 $\pm$0.047
Nomonolayer	0.998	0.985 $\pm$0.004
Debris	0.996	0.996 $\pm$0.003
Bubble	0.999	0.964 $\pm$0.023
Light Artifact	0.983	0.858 $\pm$0.038

Table 4.1: Table showing classwise F-1 scores of U-Net models compared with SegFormer. U64-SL refers to the results of U-Net model being used by Alentic, which was used as a baseline for our experiments. U64-SL refers to the re-implementation of U64-SL model. SegFormer refers to SegFormer model which was trained with patch size of 3×3.

Figure 4.2 discusses the segmentation mask produced by this model. Compared

segmentation masks generated by SegFormer model trained with patch of 7×7 as showed in Figure 4.1, these segmentation masks show better detection of RBCs, WBCs and platelets. It is evident from 4.1 that SegFormer model performs similarly to the U64-SL model in the identification of RBCs and WBCs. However, the U64-SL model demonstrates superior performance. For detection of RBC edge U64-SL has F-1 score of 0.975 whereas SegFormer scores average F-1 score of 0.889 with a standard deviation of 0.028. For RBC center, F-1 score of U64-SL was 0.946, whereas for SegFormer it was 0.670 with a standard deviation of 0.131. As for WBCs, U64-SL had F-1 score of 0.974 and SegFormer has F-1 score of 0.827 with a standard deviation of 0.011. For identification of platelets, U64-SL achieves the F-1 score of 0.889 whereas SegFormer attains a significantly lower mean F-1 score of 0.444 with a standard deviation of 0.068.

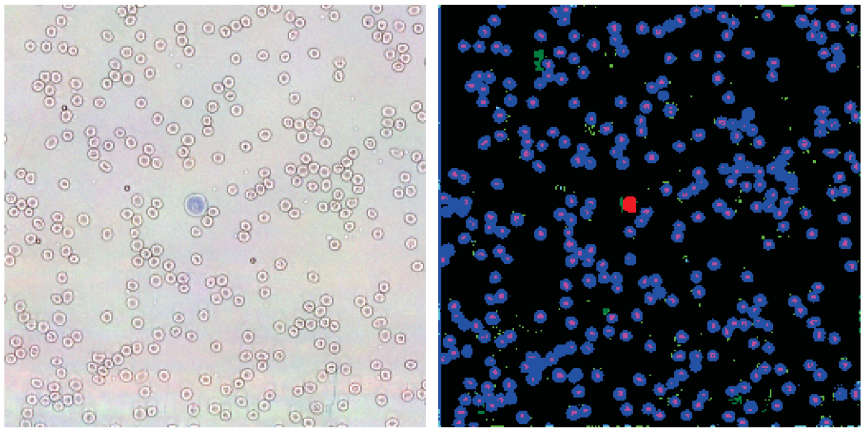
4.2 AutoFocusFormer

Transformer-based architectures start with dividing the input image in patches. For this these architectures often downsampled by factor of 4, 8, 16 and so on. For tiny images this turns out to be harmful. AutoFocusFormer is the very first model that proposes adaptive down-sampling. In their words, they want to retain the more 2 dimensional image locations of informative areas(areas where there are cluttered objects) and find unimportant areas.

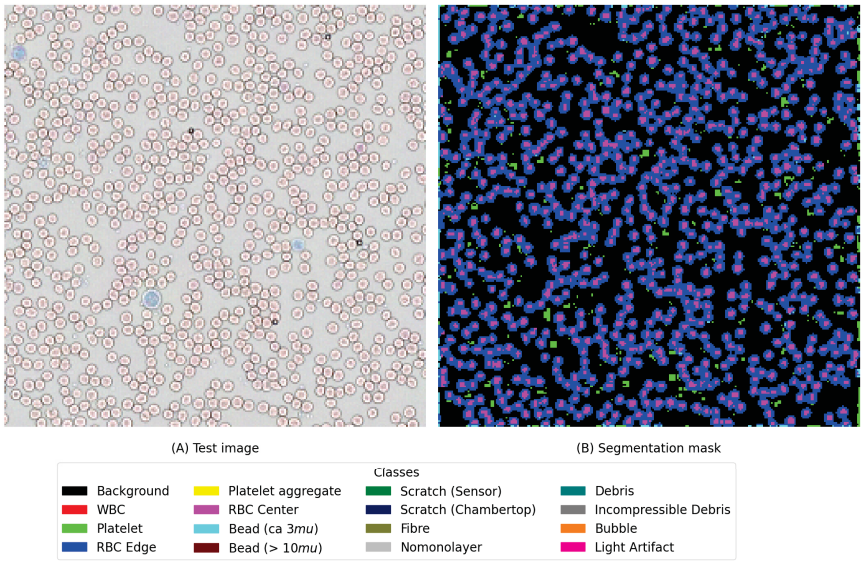
Architecture of AutoFocusFormer begins with a patch embedding layer implemented by 2 layers of 3×3 convolution with stride of 2. Then they have several stages, each stage consists of balanced clustering algorithm followed by attention module followed by adaptive down-sampling layer. Output of each stage is input to the next stage. In the end, they have task specific head, which performs tasks such as classification, instance and semantic segmentation.

The balanced clustering algorithm is used to find local neighborhood. First the image is divided into number of patches. This number of patches is supposed to be similar to number of clusters to be found in the image. Center of each patch is called an anchor point. For all points in the patch, they compute a ratio r, which they use to order the tokens. This ratio r_i is calculated as

Based on this ratio, they implement curve filling algorithm that converts 2-D grid into 1-D sequence. Significance of doing this is that, for a point closer to the anchor



(a) Figure showing segmentation map generated by SegFormer model trained with patch size of 3×3.



(b) Figure showing segmentation map generated by SegFormer model trained with patch size of 3×3

Figure 4.2: Figure showing segmentation map generated by SegFormer model for RBC, WBC and platelets.

this ratio will be smaller than the ratio of distance between anchor and point far from the anchor. This helps them the spatial consistency of image. To obtain groups, this 1-D sequence is divided into equal size groups. To make sure that contextual information stays intact, they allow each token to attend from nearby clusters. This size of neighborhood is multiple of cluster size.

Adaptive down-sampling algorithm has two components, first is learnable score component that identifies most important tokens and second component is neighborhood merging step. Learnable score is then used for selecting merging tokens as merging centers is a weighted sum between score s_i and grid prior g_i with α. Out of these features, they select top x% features which are then input to next stage.

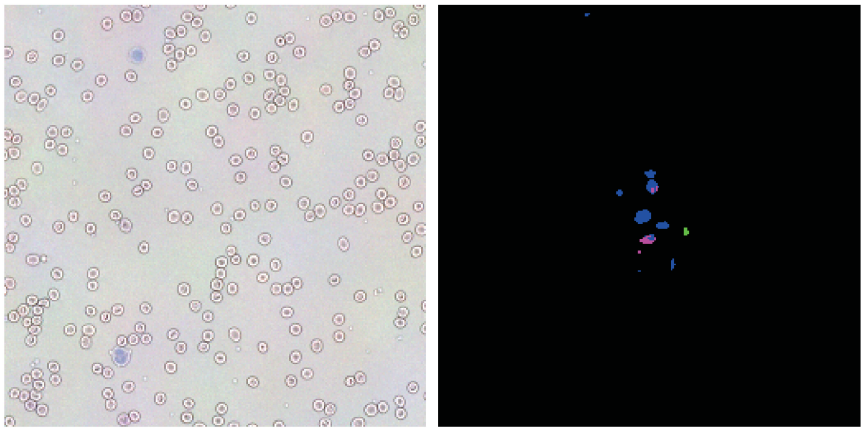
4.2.1 Training

AutoFocusFormer is implemented in detectron2. Backbone of AutoFocusFormer has depths of $[3, 4, 18, 2]$, embedding of size $[96, 192, 384, 768]$ with number of heads $[3, 6, 12, 24]$. We have used scaling factor of 1e-05 to the output of layers. Cluster size of 8 and neighborhood size of 48 are used. This model has $46.8M$ parameters. For this model base learning rate used was 2e-04 and it was trained for 5,000 iterations.

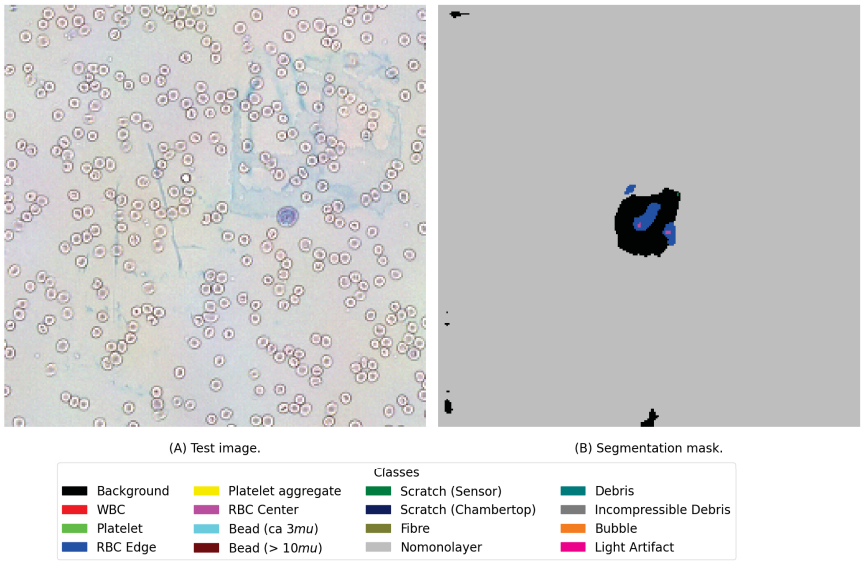
Figure 4.3 shows the sample segmentation map generated by AutoFocusFormer model. Figure 4.3 shows sample image and the output segmentation produced by the AutoFocusFormer model.Figure 4.3a shows a that the model identified some of the RBC in the center, but it was not able to differentiate edge and center Similarly, in Figure 4.3b it can be seen that the model majorly misclassified background as nomonolyaer. In this example as well model identified RBC in the center.

4.2.2 Results

Authors of AutoFocusFormer [16] experimented with datasets such as ADE20K [24] for semantic segmentation. Key difference between these dataset and the blood cell dataset is the size and appearance of object of interest. In datasets like ADE20K, objects of interest cover a huge portion of image, whereas in blood cell dataset the objects are tiny. Similarly, considering the density of objects per patch, blood cells like RBC are much dense, which affects the efficiency of balanced clustering algorithm of AFF.



(a) Figure showing segmentation map generated by AutoFocusFormer model.



(b) Figure showing segmentation map generated by AutoFocusFormer model.

Figure 4.3: Figure showing segmentation map generated by AutoFocusFormer model. This figure shows the segmentation for RBCs.

4.3 Multistage Attention ResUNet

UNet model we utilized used raw-skip connections meaning that they are not processed. MAResUNet model also proposes linear attention mechanism which results in efficient utilization of skip connections. As this method reduces the complexity of attention mechanism, it enables us to use attention blocks on larger feature maps, resulting in a better feature representation for these blocks.

4.3.1 Dot Product Attention

For an image with height H, width W, a input sequence $X \in R^{N \times C}$. dot product attention utilizes weight matrices for query(Q), key(K) and value(V) matrices namely W_q, W_k and W_v. W_q and W_k are of shape $N \times D_k$ and W_v is of shape $N \times D_v$ where $N = H \times W$. A normalization function ρ is used to calculate similarity between i^{th} query from q_t^i and j^{th} query from vector k_j^t as $\rho(q_i^T \times k_j)$ which is not a symmetric function. Dot product attention modulde applied on Q, K, V can be obtained as

$$D(Q, K, V) = \rho(Q^T \times K) \times V \tag{4.1}$$

Where ρ is a weighted summation function. For normalization, softmax is applied on each row of the resultant matrix.

4.3.2 Training

The MAResUNet architecture is a fusion of the U0Net and ResNet34 architectures, tailored for semantic segmentation tasks. It begins with an encoder section inherited from ResNet34, utilizing its deep convolutional layers for feature extraction at multiple scales. As the default ResNet34 is designed to have 3 input channels and we had 16 input channels, the first convolutional layer is updated to incorporate the 16 input channels. This convolutional layer uses kernel of size 7, strides of size 2 and padding of size 3. The integration of attention mechanisms, namely Position Attention Module (PAM) and Channel Attention Module (CAM), enhances the model's capacity to focus on crucial image regions during feature re-calibration. The decoder component follows, employing a series of decoder blocks to up-sample feature maps and concatenate them with corresponding encoder features, facilitating precise segmentation. Finally, transposed convolutions and convolutional layers are utilized in tandem to generate the final segmentation mask with the desired number of output

classes. This design enables MAResUNet to effectively capture spatial information while maintaining contextual understanding, making it a robust choice for various segmentation tasks.

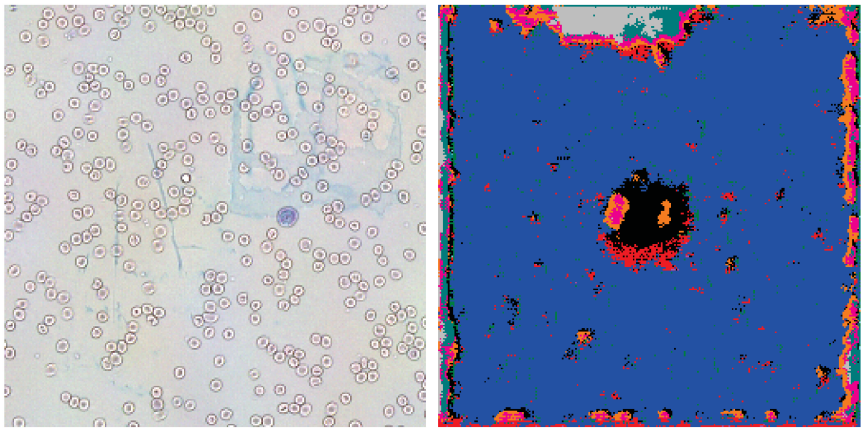
SGD is used as optimizer function with initial learning rate of 1e-02 with weight decay of 1e-03 to prevent over-fitting. StepLR scheduler is used that decays the learning after each step by a factor of 0.98. Initially the model was trained for 40 epochs and to see it the performance improves with increasing number of epochs, it was trained for 100 epochs.

4.3.3 Results

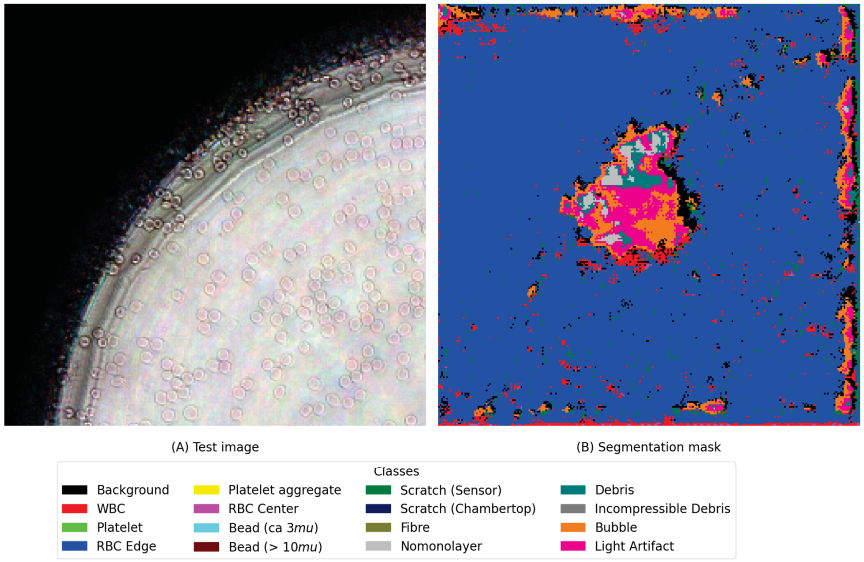
Figure 4.4 shows a sample image and corresponding segmentation map generated by MAResUNet. Figure 4.4a indicate that the model miss-classifies most of the pixels as RBC edge. In the figure on left, there is no presence of nomonolayer but model seems to have misclassified some pixels as nomonolayer. Similarly, Figure 4.4b shows that the model has not been able detect the nomonolayer, although the model seems to identify the structure.

4.4 Conclusion

Looking at the results of SegFormer, AutoFocusFormer, and MAResUNet, we can conclude that these models are not a good fit for the segmentation of blood cells if used with default patch size of 16×16. SegFormer model, when trained with patch size of 3×3 performed much better than the same model when trained with patch size of 7×7 as shown in figures 4.1 and 4.2. AutoFocusFormer and MAResUNet models although robust in nature, did not work for detecting blood cells. This suggests that further customization or entirely different model architectures may be required to achieve the desired accuracy and reliability for identification of blood cells.



(a) Figure showing image and segmentation map generated by MAResUNet model.



(b) Figure showing image and segmentation map generated by MAResUNet model.

Figure 4.4: Figure showing image and segmentation map generated by MAResUNet model.

Chapter 5

Improving platelet detection: Two stage approach

In this chapter, we discuss two different approaches for using a two-stage classifier to identify platelets. First, we explore a method where one network identifies all cells, and a second network specifically identifies platelets missed by the first model. We hypothesize that this will improve the identification of initially missed platelets. Next, we examine an approach that treats singular and aggregate platelets differently. Following that, we discuss another approach where we discuss about treating singular and aggregate platelets different from one another.

5.1 Two stage classifier

We begin with the two-stage classifier approach, where one classifier detects all blood cells, and a second classifier focuses solely on platelets. The second classifier, trained only on platelets, cannot detect any other blood cells. We first discuss the dataset used, noting its differences from the one discussed in the previous section, and the tools used to prepare it. Then, we outline our proposed methodology, detailing the algorithm for the two-stage classification process. Finally, we discuss the training process for this approach.

5.1.1 Modified dataset for two stage approach

The base dataset has examples of blood cells such as RBC, WBC and platelets. Along with that, there are other artifacts including but not limited to bead, scratches and debris. In order to create a secondary dataset that has examples only of platelets and background. For this, we used U64-SL model to generate masks for all images in the dataset. From the generated mask, we masked the labels that correspond to any class other than platelet and background. We went through these masks one by one, and labelled as many platelets as we could. For this we developed a tool as showed in the Figure 5.1. We input the index of an image and click load image. After the image was loaded, we labelled the platelets from the image. For this, we simply clicked on

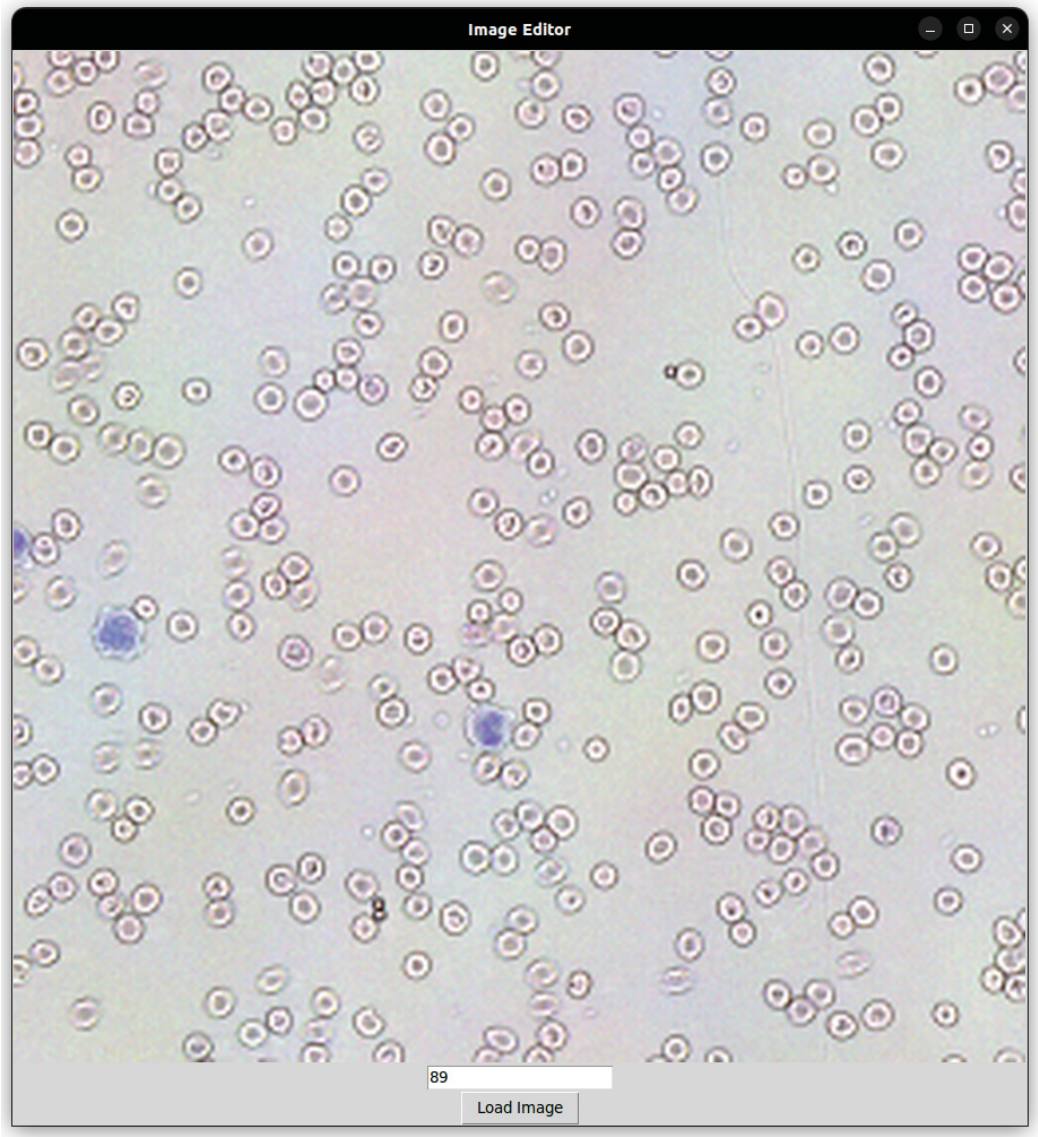


Figure 5.1: Figure representing the labelling tool developed for labelling platelets. Before using this U64-SL model was used to evaluate all images in the dataset and generate their corresponding segmentation masks. Files containing images and their segmentation masks were input to this tool. User needs to input index of image that needs to be annotated for platelets and click on load image button.

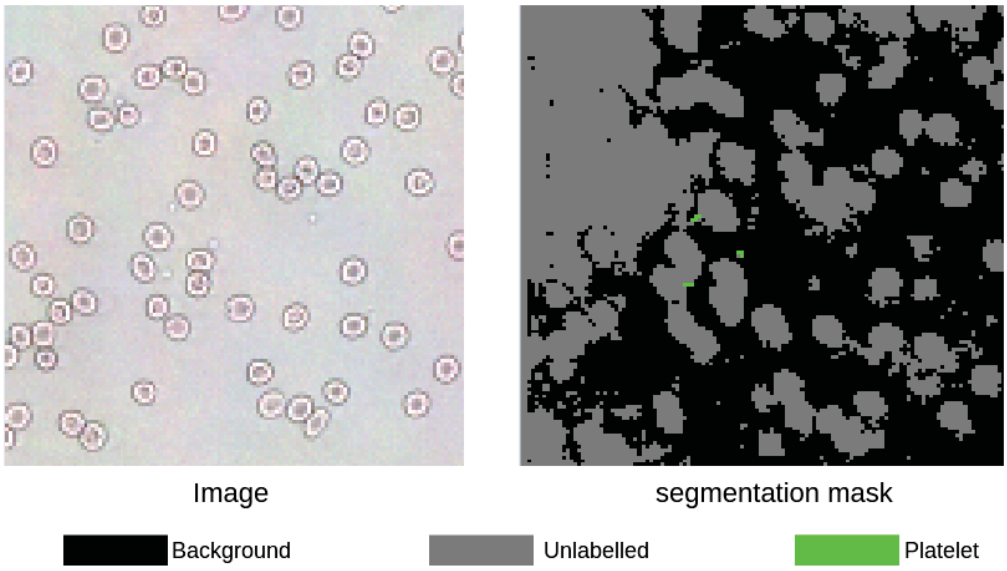


Figure 5.2: Figure representing sample data point from dataset used for stage 2 UNet. Image on left shows the blood cell image, image on the right is the segmentation mask for the image. Background pixels are represented by black color, whereas pixels corresponding to platelets are represented in blue color. Pixels that were kept unlabelled are represented in gray color.

the pixels we found inside the platelet. Due to the detailed examination needed for each image, only 400 images were labeled using this method. This process resulted in 7,210 pixels labelled as platelets and 9,140,488 pixels labelled as background. A sample image in the dataset is represented in Figure 5.2.

5.1.2 Methods

The motivation behind the two stage approach is to be able to identify all the platelets. This is called a two-stage approach because it makes use of two UNet models. First UNet model is trained to identify all the blood cells and the second model is trained only to identify platelets. In this we discuss the proposed algorithm for two-stage UNet approach.

- Use the U64-SLB model that is already trained on the base dataset with all classes to generate segmentation masks.
- In the segmentation masks generated, set all pixels that do not correspond to background class as unlabelled. Then using the tool described in Figure 5.1,

label all the platelets in images. This is necessary to make task of segmentation simple by allowing training to be done only for identification of platelets.

- Train another U64-SL model on this newly created dataset. We call this model U64-SL2.
- For evaluation, first use U64-SLB model to generate the masks. Then use U64-SL2 model to generate the masks.
- Collect indices where the U64-SLB model has predicted background. From masks generated by U64-SL2 model collect indices that correspond to platelet class. Take their intersection, and update the masks at those indices to platelets.

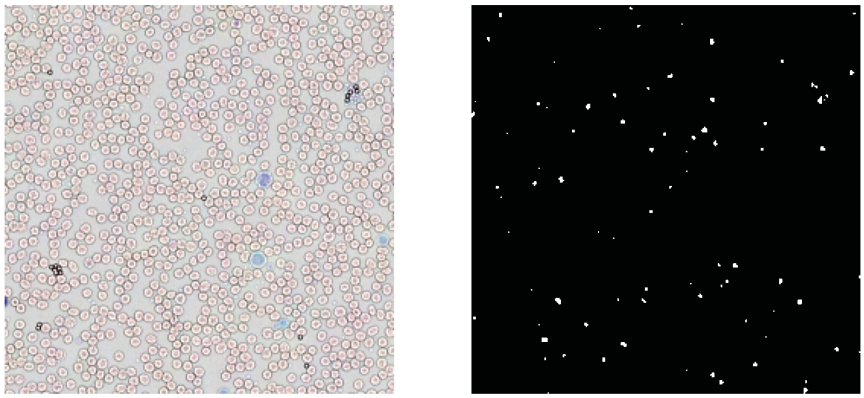
Taking intersection of these indices would mean collecting indices where the U64-SLB model missed the platelet and U64-SL2 model caught the platelet. If U64-SLB is calling a certain pixel as background, and U64-SL2 calls it a platelet then only we will update the mask produced by U64-SLB. This is because, model-1 knows what cells like RBC and WBC look like and U64-SL2 doesn't. If U64-SLB calls a pixel RBC or WBC, and U64-SL2 calls it a platelet, it most likely is actually either a RBC or a WBC. In that case calling that a platelet would be wrong.

5.1.3 Training

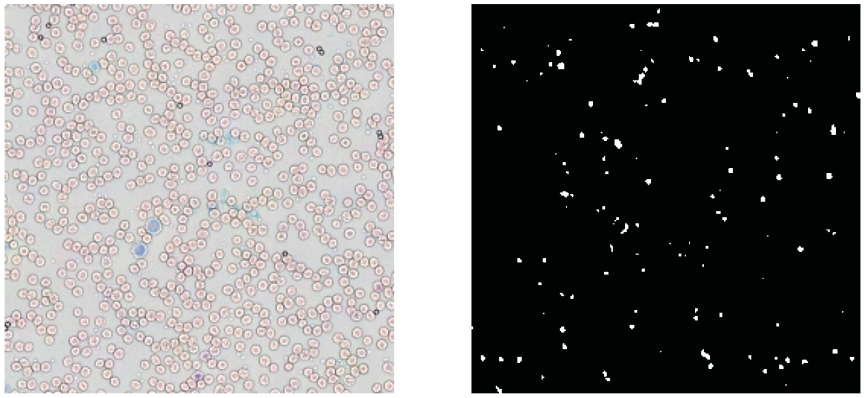
We used first 90% images in the dataset for training and tested the model on rest 10% images. Thus, for training we had 7,885,616 samples of background and 8,349 samples of platelets. Weighted focal loss is used as a loss function. SGD optimizer is used with initial learning rate of 0.15 and weight decay of 0.01 are used. This model was trained for 50 epochs with learning rate decay by the factor of 0.98 every epoch. As there is clear class imbalance, class weights were used, which were computed by taking square root of inverse class frequency.

5.1.4 Results of stage-2 U-Net

Figure 5.3 shows platelets being detected by stage-2 U-Net model. As this model was only trained on platelets, it does not classify any other class.

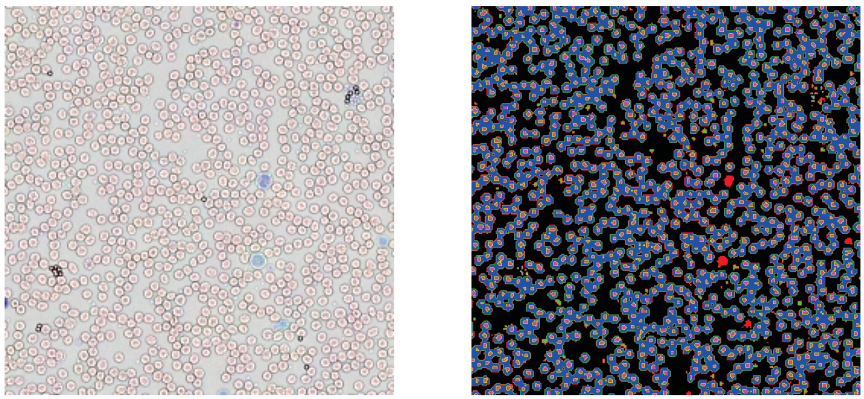


(a) Stage-2 model identifying platelets in test image 1.

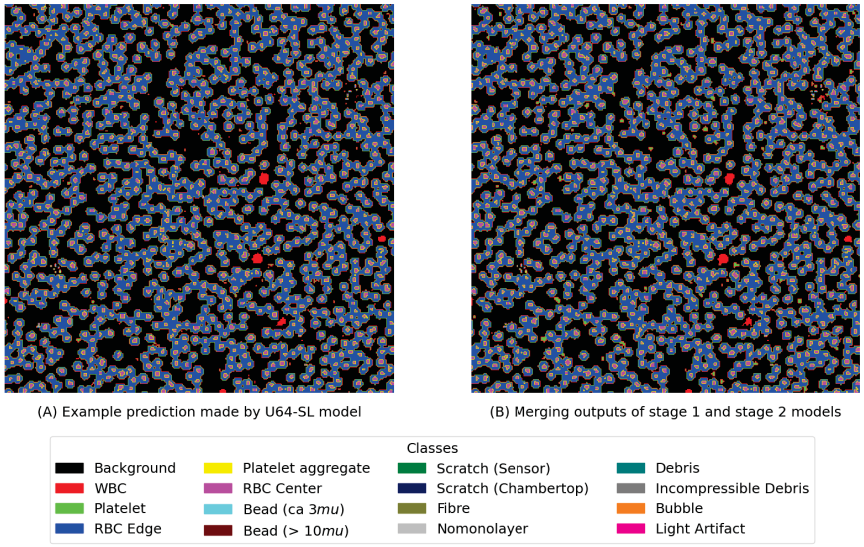


(b) Stage-2 model identifying platelets in test image 2.

Figure 5.3: Figure showing image and corresponding segmentation mask produced by stage 2 U-Net model that specializes in identifying platelets.



(a) Segmentation mask generated by two-stage model.



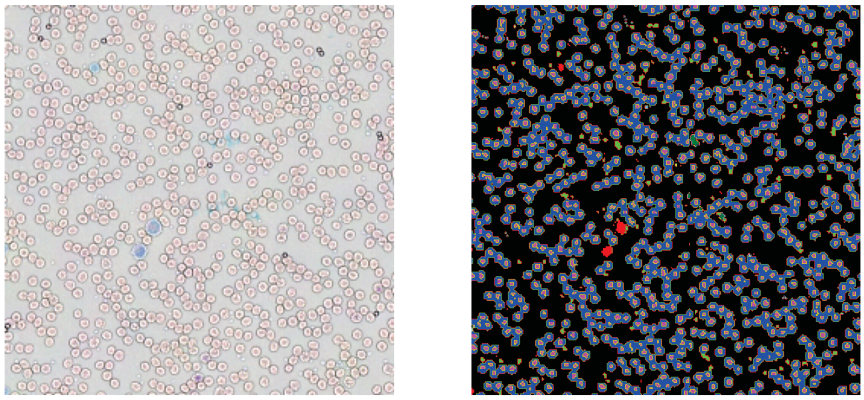
(b) Comparing the masks produced by U64-SL and two-stage model

Figure 5.4: Figures showing stage-2 model identifying platelets in test images.

5.1.5 Results of two-stage approach

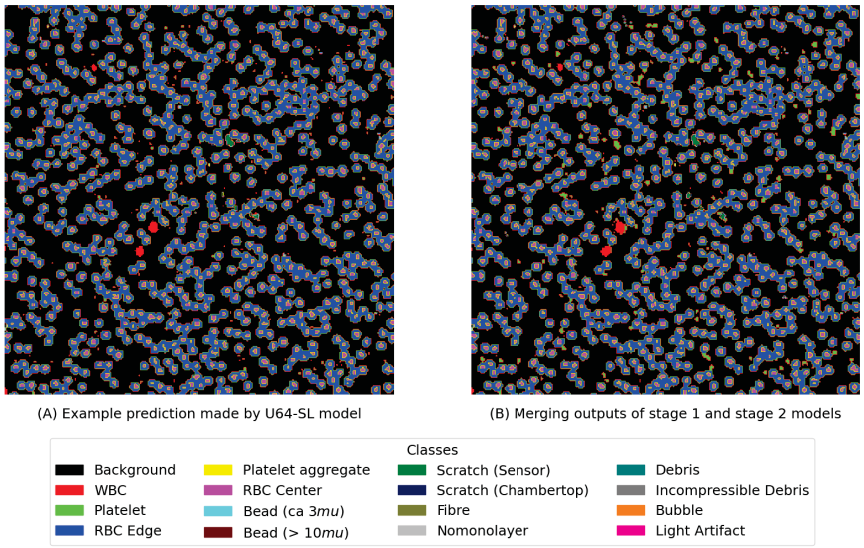
Figures 5.4 and 5.5 show the segmentation mask generated using two-stage approach and how they compare with U64-SL model. In these figures 5.4a and 5.5a show the masks generated by two-stage approach and 5.4b and 5.5b shows the comparison.

Idea behind Two-Stage approach was to be able to identify the platelets that were missed by the U64-SL model. In this section, we first discuss the second model in proposed two stage U-Net model that specialized in identifying platelets from the background. A variation of this approach was followed by Elmanna et al. [20]. In their work they utilized a two-stage approach where they used U-Net model for



(A) Test image (B) Merging outputs of stage 1 and stage 2 models

(a) Segmentation mask generated by merged two-stage model.



(A) Example prediction made by U64-SL model (B) Merging outputs of stage 1 and stage 2 models

(b) Comparing the masks produced by U64-SL and two-stage model

Figure 5.5: Figures showing stage-2 model identifying platelets in test images.

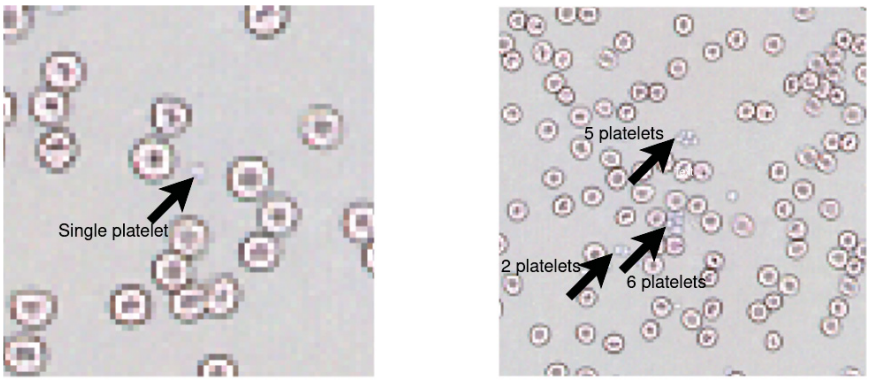
segmentation and EfficientNetB0 model was used for classification of RBCs . Instead of using another architecture for identification of platelets, U-Net was itself trained for identification of platelets. In this section, we first discuss how the second U-Net model performs in identifying platelets and then we discuss the result of combining the outputs of both U-Net models we used in our two-stage approach.

5.2 Differentiating single platelet and platelet aggregates

In this section we discuss the method we propose of treating singular platelets and platelets that are stuck together, further referred to as platelet aggregates, different from one another as opposed to the original work that treats them the same. Appearance of a single platelet is considerably different from the appearance of platelets that are stuck together. Looking at this from convolution point of view, features learned for a single platelet are different than the features learned for the platelet aggregate. Thus using the same features would result in model missing on such platelets. We begin by discussing the changes we propose for the dataset. Then we discuss the details about training U64-SL model on the proposed dataset. With the proposed change, the method used for counting the platelets need to be changed as well. After discussing training, we discuss about different experiments performed to count the platelets.

5.2.1 Dataset

Certain examples in the base dataset consist of singular and aggregate platelets. While labelling platelets, if they are tiny, center pixel was labelled as platelet, in case of platelets that are bigger than the average size, several pixels were labelled. In case of platelet aggregates, an additional pixel on the edge of platelet is labelled as background pixel. This was done to guide the model in identifying that those are multiple platelets and not just one big platelet. In the dataset we propose, two classes correspond to platelet. Singular platelets are labelled as a platelet, platelets that are stuck together, weather two, three or more, are labelled as platelet aggregates. While labelling these, all the pixels corresponding to those platelets are labelled as aggregates. Figure 5.6 depicts the variations in single platelets and platelet aggregates. Table 5.1 shows the distribution of various classes.



(A)Figure containing single platelet.

(B)Figure containing multiple platelet aggregates.

Figure 5.6: Figure showing singular platelet versus platelet aggregates.

Class	Count
Background	80,069
WBC	7,442
Platlet	2,993
RBC Edge	14,506
Platelet Aggregate	3,364
RBC Center	2,422
Bead (ca. 3μm)	749
Bead (10μm)	737

Class	Count
Scratch (Sensor)	4,021
Scratch (Chambertop)	1,069
Fibre	0
Nomonolayer	125,054
Debris	62,161
Incompressable Debris	0
Bubble	49,967
Light Artifact	26,829

Table 5.1: Table showing classwise distribution of instances of pixels.

5.2.2 Training

Before passing this dataset to the U-Net model, to normalize the images, we divide those by 255. As we are dealing with class imbalance and very small objects, we are using focal loss as our loss function. For training this model as well, we are using learning rate scheduler that will decay the learning rate by the factor of 0.98. We trained this model with batch size of 32 for 40 epochs. We used Stochastic Gradient Decent optimizer with initial learning rate of 0.01. To treat the class imbalance we use square root of inverse class frequency.

5.2.3 Methods for counting the number of platelets

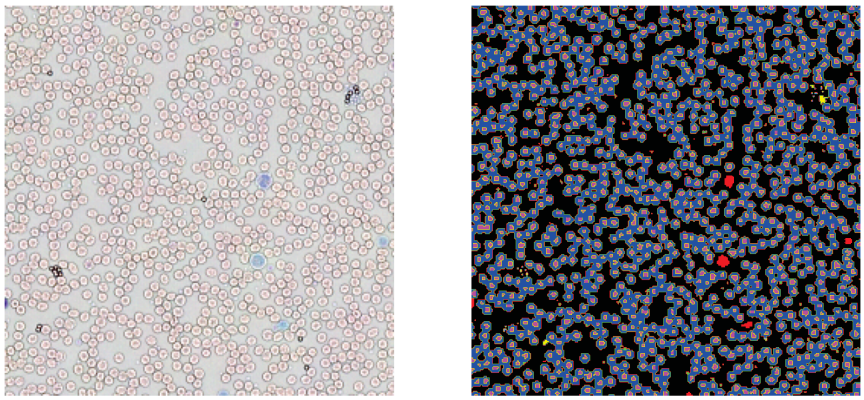
As discussed above, for a aggregate of platelets, we labelled all pixels corresponding to that aggregate. This causes a bunch of pixels adjacent to each other to be labelled as platelet aggregate. The segmentation masks for platelet aggregate generated by the trained model follows the same pattern. This causes segmentation mask corresponding to platelet aggregate resemble a clusters. Thus we use density-based spatial clustering application with noise (DBSCAN) algorithm to access these clusters. This algorithm gives us all the coordinates for a particular cluster. From these coordinates we use minimum and maximum values to identify top left and bottom right corner of a rectangle. Now that we have coordinates in a cluster and coordinates of a bounding box, we evaluate two strategies to estimate number of platelets.

For counting the number of platelets, we experimented with two different approaches. The first approach we used involved counting total number of pixels corresponding to a particular cluster. As the aggregate consists multiple platelets dividing this number of pixels by size of average platelet, which we assumed to be 3 pixels on average. We call this naive counting. One observed characteristic of platelet is that a platelet has highest brightness near its center which makes it stands out from the background. For the second approach, we propose that counting the number of highest intensity pixels from the cluster containing the platelet aggregate will provide us the estimate of number of platelets in that particular aggregate. It is often observed that segmentation mask corresponding a platelet aggregate does not cover all the pixels in that aggregate. To solve this problem, we make sure that the image patch that we compute the number of peaks is larger than or equal to size of 5×5 pixels. Increasing the patch size does not affect the pixel intensities, therefore this approach does not lead to increase in number of peaks.

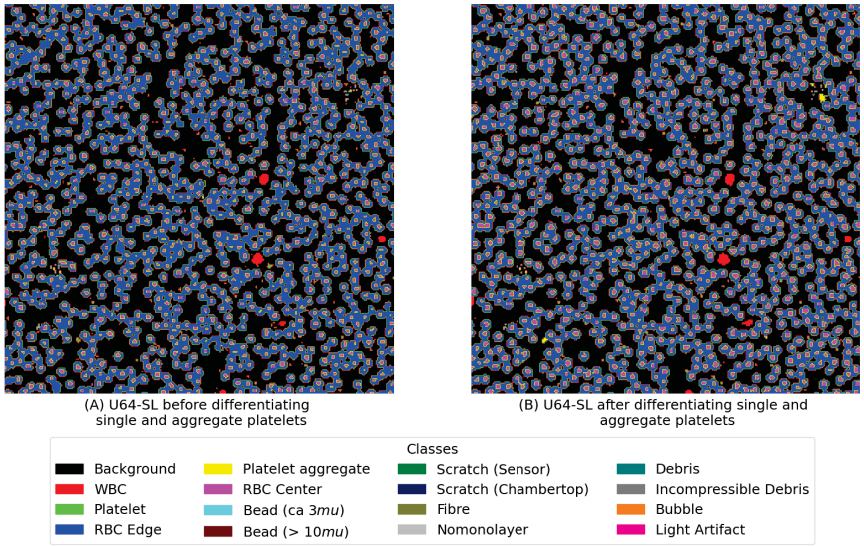
5.3 Results of experiments with treating singular platelets and platelet aggregates differently

$U64-SL^3$ and $U64-SL^4$ from Table 3.2 indicate improvement in platelet detection as compared to U64-SL and $U64-SL^2$. Mean F-1 score for platelets in $U64-SL^2$ is 0.874 with a standard deviation of 0.027, whereas mean F-1 score in $U64-SL^4$ is 0.969 with standard deviation of 0.010. This shows significant improvement for detection

of platelets, which shows that differentiating singular and aggregate platelets help in detection of platelets.

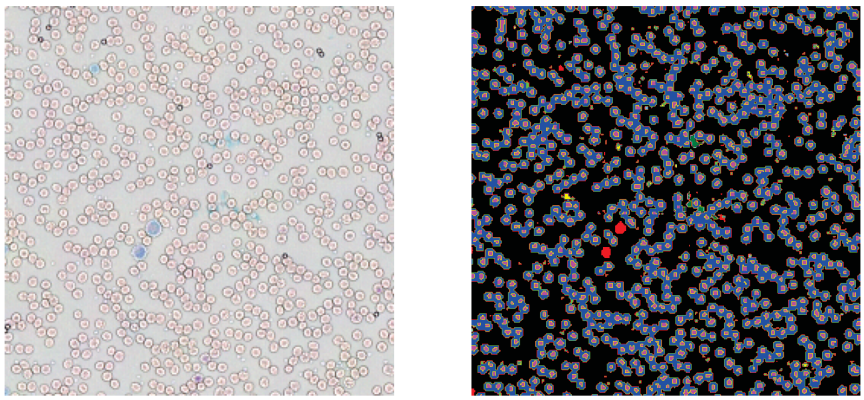


(a) Figure showing a sample image and the output segmentation produced by the U64-SL model model trained on dataset that differentiates singular and aggregate platelets.

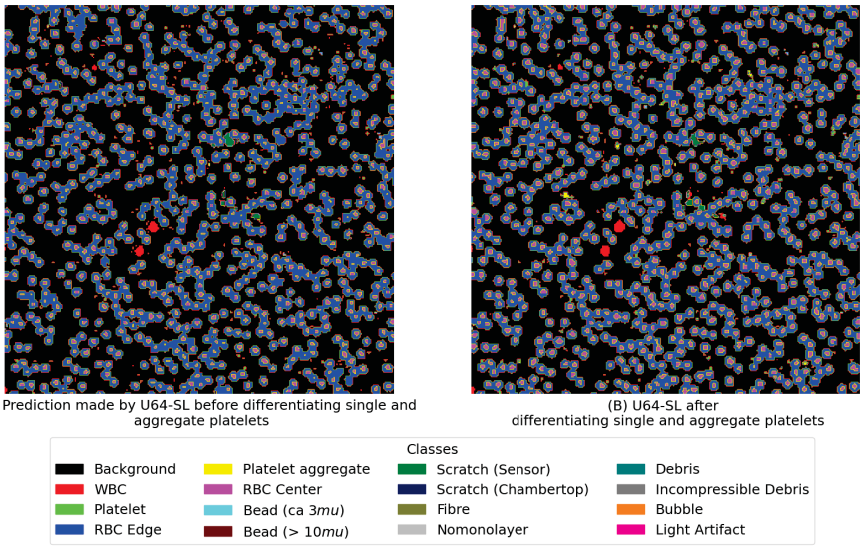


(b) Figure showing comparison between U64-SL models trained before and after differentiating singular and aggregate platelets.

Figure 5.7: Figure showing segmentation map generated by U64-SL model after differentiating singular and aggregate platelets and how the segmentation map compares with the same generated without differentiating.



(a) Figure showing a sample image and the output segmentation produced by the U64-SL model model trained on dataset that differentiates singular and aggregate platelets.

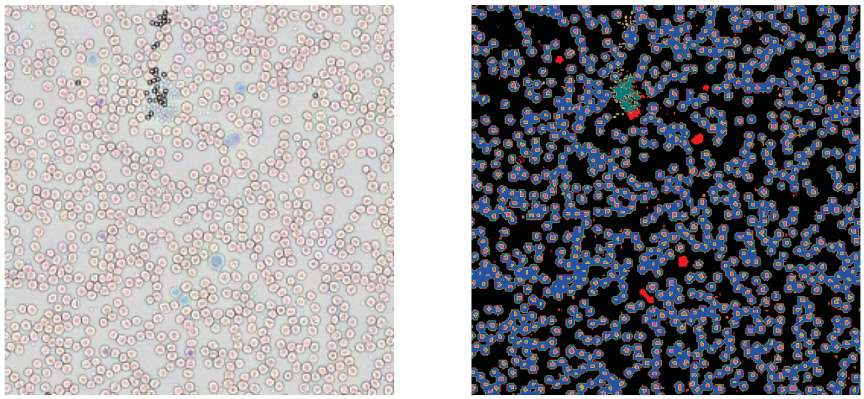


(b) Sample image and the output segmentation produced by the U64-SL model model trained on dataset that differentiates singular and aggregate platelets.

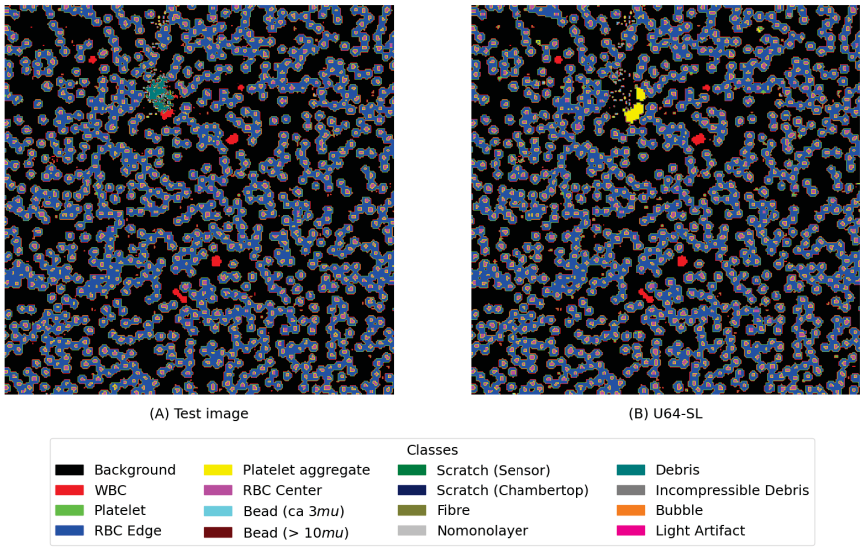
Figure 5.8: Figure showing segmentation map generated by U64-SL model after differentiating singular and aggregate platelets and how the segmentation map compares with the same generated without differentiating.

Figures 5.7 and 5.8 represent sample images and their segmentation masks generated before and after differentiating singular and aggregate platelets in the dataset. In Figure 5.7a, in the segmentation mask, it can be seen that the model has captured the platelet aggregate in yellow color. In Figure 5.7b, it can be clearly seen that without differentiating singular and aggregate platelets the area captured by the model does not include all the area where there are platelets. Chowdhury et al. [21]

talk about treating single and aggregate platelets differently. Inspired from this, we separated singular and aggregate platelets from our dataset.



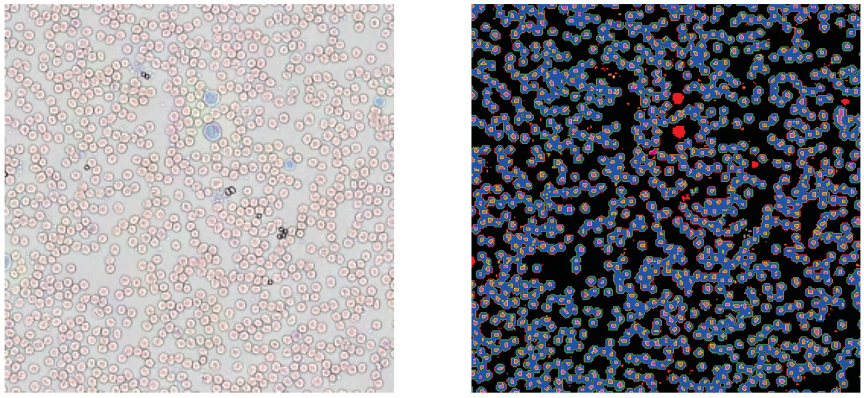
(a) Figure showing a sample image and the output segmentation produced by the U64-SL model model trained on dataset that differentiates singular and aggregate platelets.



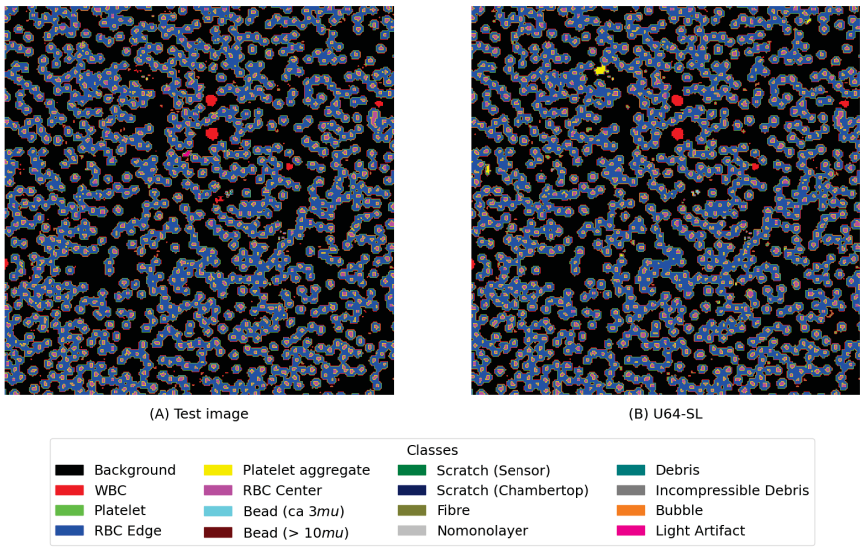
(b) Sample image and the output segmentation produced by the U64-SL model trained after adding bigger platelet aggregates to the dataset.

Figure 5.9: Figure showing segmentation map generated by U64-SL model after differentiating singular and aggregate platelets and how the segmentation map compares with the same generated without differentiating.

Figures 5.9 and 5.10 show the segmentation masks produced by U64-SL model after adding platelet aggregates containing large number of platelets. 5.9a and 5.10a show the U64-SL model either missing the platelets completely or detecting very few platelets from the platelet aggregates. The reason for this was that the model was



(a) Figure showing a sample image and the output segmentation produced by the U64-SL model model trained on dataset that differentiates singular and aggregate platelets.



(b) Sample image and the output segmentation produced by the U64-SL model trained after adding bigger platelet aggregates to the dataset.

Figure 5.10: Figure showing segmentation map generated by U64-SL model after differentiating singular and aggregate platelets and how the segmentation map compares with the same generated without differentiating.

never trained on bigger number of platelets in the aggregate. Whereas 5.9b and 5.10b show the same model identifying those platelets after being trained to identify large platelet aggregates.

5.3.1 Counting platelets

Table below shows counting of platelets across 48 different examples. For this, platelet aggregates of varying sizes. We started with singular platelets and extended our study upto aggregate consisting of 16 platelets. Table 5.2 shows the platelet counts for Figure 5.11.

Actual	Naive	Peaks	U64-SL
3	2	1	4
2	6	1	2
4	10	5	3
6	13	5	6
4	13	3	4
3	4	4	4
5	4	4	3
2	4	3	4
2	3	1	3
3	9	3	4
6	6	3	5
4	5	2	4
3	5	4	3
3	8	2	3
8	4	5	4
6	13	8	6
3	2	4	2
9	7	5	3
4	4	1	4
16	24	17	10

continued ...

Actual	Naive	Peaks	U64-SL
2	6	4	3
3	4	2	3
3	9	2	3
2	3	3	2
4	2	4	1
2	8	4	2
1	4	1	1
1	4	2	1
4	8	4	3
2	2	1	2
11	25	8	11
3	5	3	3
2	6	3	2
3	7	1	3
1	2	3	1
3	2	2	4
1	3	1	1
1	2	1	1
1	3	2	1
2	4	4	2

Table 5.2: Table showing comparison of platelet count estimation using various methods with the actual platelet count. In this table, actual represent actual count of platelets. Naive represents platelet count as per naive counting approach. Peaks represent platelet count estimated by counting the number of peaks in the region of platelet aggregate. U64-SL represent platelet count estimated by U64-SL model.

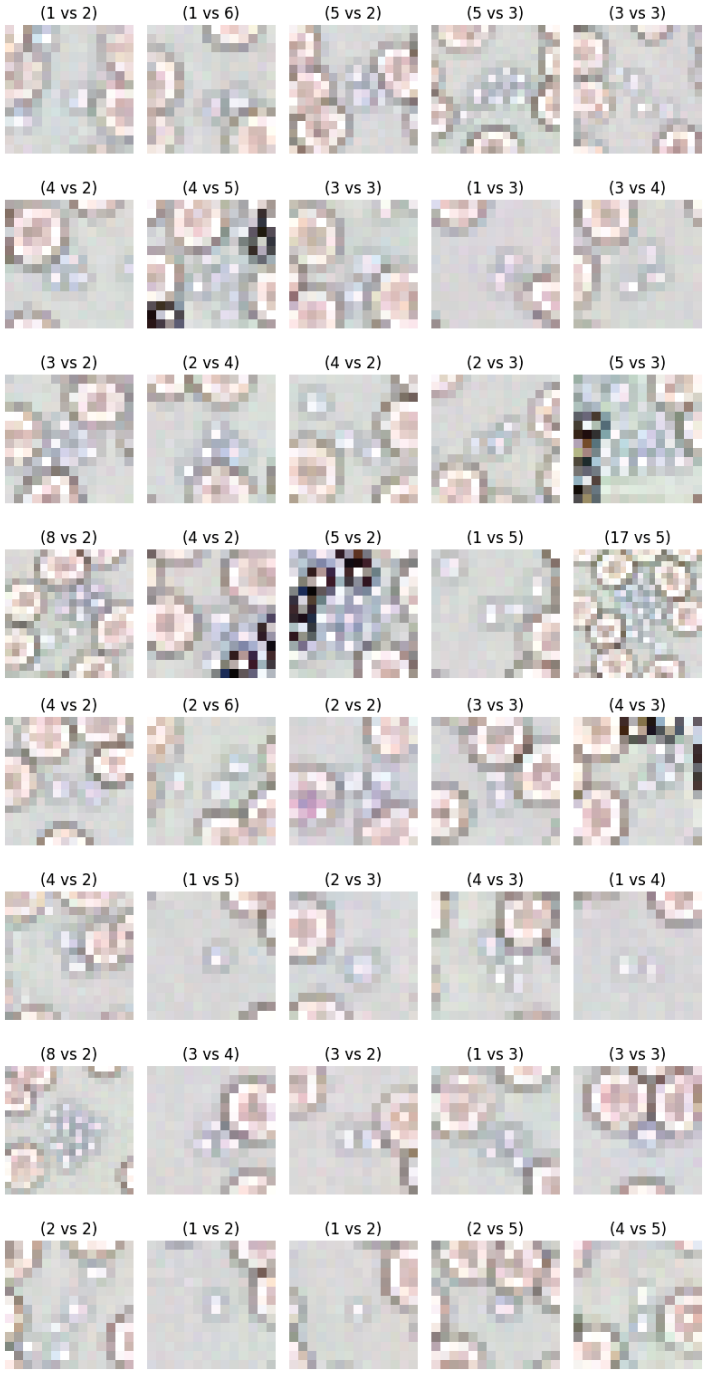


Figure 5.11: Figure showing the clumps identified by U64-SL model trained after differentiating singular and aggregate platelets. Each sub-figure indicates a patch of platelets aggregate captured by the model. On top of each figure, figure on the left indicates count obtained by counting the number of peaks for patch and figure on the right indicates count obtained by naively counting the number of pixels corresponding to one aggregate and dividing the number by average size of a platelet.

Statistics	U64-SL	Counting peaks	Naive counting
$\sigma\Delta$ computed on predicted versus actual count	1.481	0.954	2.898
Standard Error computed on predicted counts	0.063	0.081	0.183
σ computed on predicted counts	1.145	1.473	3.308
Slope of line of regression	0.599	0.800	1.346

Table 5.3: Table showing the comparison of various counting methods. U64-SL represents platelet count estimated by U64-SL model. This table used the values in Table 5.2 and Figure 5.11. Counting peaks represents platelet count estimated by counting number of peaks in region corresponding to platelet aggregate. Naive counting represents platelet count estimated by naively counting number of pixel in platelet aggregate mask.

The platelet count estimated by the naive approach often exceeds the actual count. This is because it is a direct function of number pixels in segmentation map that correspond to platelet aggregates. This often includes some of the background pixels as well, which causes the count to be higher. Counting the number of peaks on the other hand utilizes both the image and the segmentation map. It utilizes the characteristic of platelet to have a high pixel intensity. Thus even if the segmentation mask contains background pixels, their intensity would be lower than the platelet intensity, resulting in a stable platelet count. Figure 5.11 shows the comparison between counts obtained using these methods.

To evaluate and compare performance of different methods of counting platelets we calculate the difference between actual and predicted counts. Let predicted counts be represented as $P = [p_1, p_2, ...p_n]$ and actual count be represented as $A = [a_1, a_2, ...a_n]$. We define $\delta_i = |p_i - a_i|$. This is repeated for 40 examples, which gives us 40 δ values. Thus, for baseline method, we have

$$\Delta_{baseline} = [\delta_{baseline1}, \delta_{baseline2}...\delta_{baseline40}]$$

$$\Delta_{peaks} = [\delta_{peaks1}, \delta_{peaks2}...\delta_{peaks40}]$$

$$\Delta_{naive} = [\delta_{naive1}, \delta_{naive2}...\delta_{naive40}] \tag{5.1}$$

where $\Delta_{baseline}$ corresponds to δ calculated on platelet counts obtained by the baseline method, Δ_{peaks} corresponds to δ calculated on platelet counts obtained by counting number of peaks in the image patch obtained from platelet aggregate and Δ_{naive} corresponds to δ calculated on platelet counts obtained by naively counting number of pixels in platelet aggregate mask and then dividing that number by average size of a platelet. We define σ as standard deviation. In the Table 5.3 we define $\sigma\Delta$ as standard deviation computed on Δs.

Figure 5.12 is showing comparison of lines of linear regression with line of best fit. Line of best fit represents line of regression where predicted count of platelets are equal to the actual counts. From the figure it can be seen that line of regression for counting peaks is closer to the line of best fit than line of regression for U64-SL. Similarly, looking at Table 5.3 it can be seen that slope of naive counting is much closer to line of best fit than other approaches.

5.4 Conclusion

Results of $U64 - SL^3$ from Table 3.2 shows us that differentiating singular and aggregate platelets help improving platelet detection. Looking at Table 5.3 and Figure 5.12, it can be observed that estimating platelet count using the number of peaks yields the platelet counts that are much closer to the line of best fit. Apart from this, the labelling approach that differentiates singular and aggregate platelets provide much easier and efficient way of labelling large platelet aggregates that contain more than about 15 platelets, which otherwise would be a tedious task. This also enables labelling of varying size platelet aggregates that contributes by adding diverse examples of platelet aggregates, reducing the probability of encountering out of distribution problem for platelets.

The architecture of U-Net model, with its contracting and expanding paths, is designed to capture fine details and contextual information in images. This capability is crucial for differentiating between single platelets and platelet aggregates. The model's ability to focus on local features while preserving spatial context allows it to identify individual platelets even when they are part of a larger cluster. The skip connections in U-Net help retain detail from earlier layers, which is essential for distinguishing small, closely packed platelets from one another.

The normal platelet count ranges from approximately 150,000 to 450,000 platelets

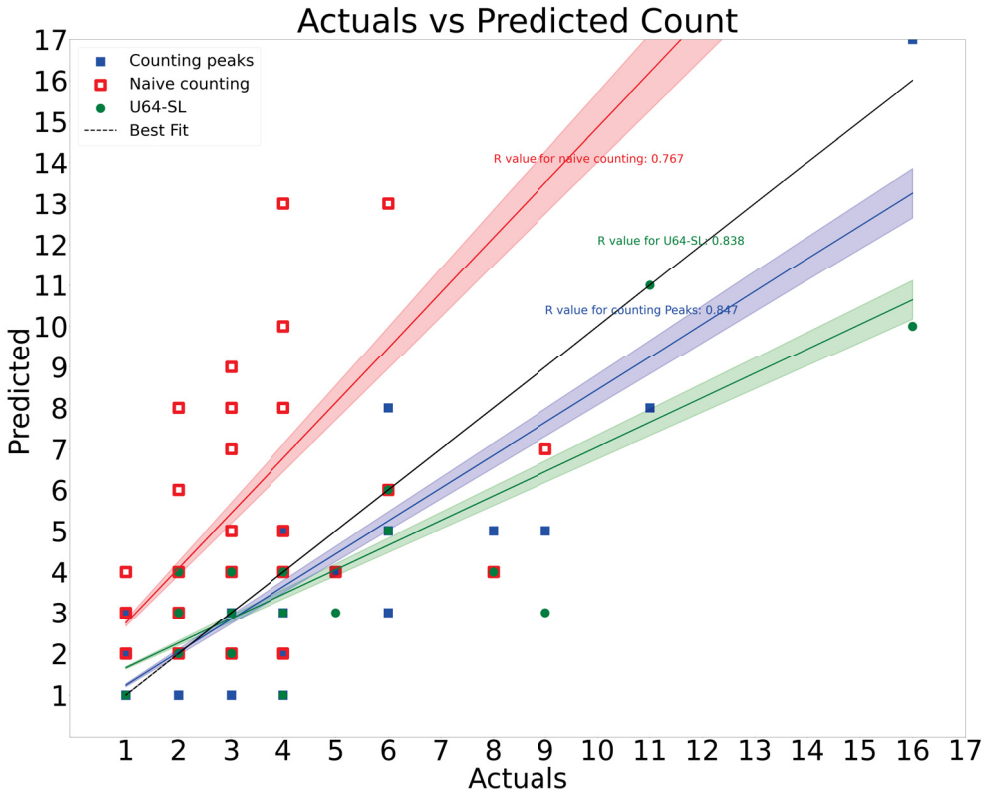


Figure 5.12: Figure showing comparison between regression lines for multiple models with line of best fit. Counting peaks represent platelet count estimated by counting number of peaks in region of platelet aggregate. Naive counting represents platelet count estimated by naively counting number of pixels corresponding to platelet aggregate and dividing that number of average size of platelet. U64-SL represents platelet count estimated by U64-SL model.

per microliter of blood. In cases of Thrombocytosis, this count is abnormally higher than the normal range. In this case, the model might face challenges in accurately identifying and counting abnormally larger platelet aggregates. These dense clusters can complicate the counting process, potentially leading to over-counting or merging adjacent platelets into a single count. The ability of model to capture and distinguish fine details is crucial in such scenarios. If the model has not been sufficiently trained on these dense configurations, it may struggle to identify those aggregates and potentially misclassify them as a different class as seen in Figure 5.9b, where the platelet aggregate was initially misclassified as debris, before being trained with examples of larger platelet aggregates.

Thrombocytopenia is another condition where platelets are sparse and lower in count than the normal platelet range. As the dataset used throughout this research consists of multiple examples of singular platelets and platelet aggregates, the developed method should be able to identify and count the platelets in this case. Future experiments are still required to confirm if this is the case.

In addition to the challenges, future work is required to assess the model's performance on images that are significantly different from those seen during training. All images are captured using lensless microscopy, where a drop of blood is placed under a closed lid and LEDs are used to capture the images. This unique imaging setup can lead to variations in image quality, such as differences in lighting, shadows, and resolution, which might affect the appearance of platelets and other blood components. For instance, the LED lighting might create reflections or artifacts that can obscure details or alter the appearance of platelets. Moreover, the new inserts used by the company have textures that resemble platelets, which could lead to confusion between actual platelets and these textures in the images.

The model's robustness needs to be evaluated against such scenarios, including cases with abnormally large platelet aggregates or textures that mimic platelets. Additionally, the model should be tested on blurry images and out of distribution samples situations where the images deviate significantly from the training data. These variations could lead to misidentification or inaccurate platelet counts, highlighting the need for extensive testing and potential model adjustments to ensure reliable performance across the diverse conditions.

Chapter 6

Conclusion

Based on the experiments with various U-Net architectures, it is clear that different starting numbers of filters (16, 32, 64, 128) significantly impact the model's performance and resource requirements. From the experiments it was clear that U64-SL U-Net model with a single repetition of convolution followed by batch normalization followed by ReLU with starting filters equal to 64 provided us with the best results. Smaller filters, while requiring less computational power, may struggle with capturing complex features, whereas larger filters can capture more details but at an increased computational cost. U64-SL balances this trade-off as it is not computationally expensive, yet is able to capture the important features.

During experiments with the transformer-based models such as SegFormer, it was noticed that patch size plays a crucial role. The default patch size for SegFormer, which was 7×7 did not work for segmentation of blood cell. This was because the blood cells are much smaller than that patch size. Looking at the segmentation masks generated by this model, it was clear that the model was not able to identify RBCs, WBCs and platelets. When we experimented with patch size of 3×3, it was seen that the model was able to identify those blood cells except for platelets. This indicates that further study is required to analyze effects of various patch sizes.

In our experiments, use of two-stage approach proved to be beneficial after differentiating singular and aggregate platelets from one another. Another approach we experimented was to use a separate classifier that specialized in detection of platelets. This approach showed significant improvement in F-1 score for platelets. Based on this approach we also experimented with different techniques for counting of platelets in aggregates, out of which counting the number of peaks yielded the best results. To determine which method estimates platelet count closest to the actual count we calculated the difference between estimated count and actual count and computed standard deviation on the difference. The standard deviation on platelet estimated

by counting the number of peaks was lowest, indicating closeness to the actual count. We conclude that treating singular and aggregate platelets differently not only improves the platelet detection, but can reduce the efforts required for labelling the platelets allowing larger platelet aggregates to be efficiently labelled.

Bibliography

[1] J. Wu, R. FU, H. Fang, Y. Zhang, Y. Yang, H. Xiong, H. Liu, and Y. Xu, "Medsegdiff: Medical image segmentation with diffusion probabilistic model," in *Medical Imaging with Deep Learning* (I. Oguz, J. Noble, X. Li, M. Styner, C. Baumgartner, M. Rusu, T. Heinmann, D. Kontos, B. Landman, and B. Dawant, eds.), vol. 227 of *Proceedings of Machine Learning Research*, pp. 1623–1639, PMLR, 10–12 Jul 2024.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[4] Y. Zhang, "A survey on evaluation methods for image segmentation," *Pattern Recognition*, vol. 29, no. 8, pp. 1335–1346, 1996.

[5] M. N. Gurcan, L. E. Boucheron, A. Can, A. Madabhushi, N. M. Rajpoot, and B. Yener, "Histopathological image analysis: a review," *IEEE Rev Biomed Eng*, vol. 2, pp. 147–171, Oct. 2009.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[12] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[13] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3523–3542, 2022.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *ICLR*, 2021.

[15] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," in *Neural Information Processing Systems (NeurIPS)*, 2021.

[16] C. Ziwen, K. Patnaik, S. Zhai, A. Wan, Z. Ren, A. Schwing, A. Colburn, and L. Fuxin, "Autofocusformer: Image segmentation off the grid," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[17] R. Li, S. Zheng, C. Duan, J. Su, and C. Zhang, "Multistage attention resunet for semantic segmentation of fine-resolution remote sensing images," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[18] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *ArXiv*, vol. abs/1904.10509, 2019.

[19] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020.

[20] M. Elmanna, A. Elsafty, Y. Ahmed, M. Rushdi, and A. Morsy, "Deep learning segmentation and classification of red blood cells using a large multi-scanner dataset," *ArXiv*, vol. abs/2403.18468, 2024.

[21] A. B. Chowdhury, J. Roberson, A. Hukkoo, S. Bodapati, and D. J. Cappelleri, "Automated complete blood cell count and malaria pathogen detection using convolution neural network," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1047–1054, 2020.

[22] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[23] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 818–833, Springer International Publishing, 2014.

[24] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision*, vol. 127, no. 3, pp. 302–321, 2019.