Deep Learning Convolutional Neural Networks (CNNs) on Recognition and Classification of White Blood Cells (WBCs)

BY

SALILA ONGTRAKUL

ANYARIN THITIRATTANAPONG

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN BIOMEDICAL ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2021

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Deep Learning Convolutional Neural Networks (CNNs) on Recognition and Classification of White Blood Cells (WBCs)	
Student Name	Miss Anyarin Thitirattanapong	Student ID. 62011092
	Miss Salila Ongtrakul	Student ID. 62011236
Degree	Bachelor of Engineering in Biomedical Engineering	

Project Advisor Signed: _____
(Assoc. Prof. Dr. Chuchart Pintavirooj)

Committee Signed: _____
(Assoc. Prof. Dr. Wibool Piyawattanamatha)

Committee Signed: _____
(Asst. Prof. Dr. Treesukon Treebupachatsakul)

Committee Signed: _____
(Asst. Prof. Dr. Kasama Srirussamee)

Head of Department Signed: _____
(Assoc. Prof. Dr. Sarinporn Visitsattapongse)

Project Title	Deep Learning Convolutional Neural Networks (CNNs) on Recognition and Classification of White Blood Cells (WBCs)
Student Name	Miss Anyarin Thitirattanapong
	Miss Salila Ongtrakul
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Prof. Dr. Chuchart Pintavirooj
Academic Years	2021

ABSTRACT

Artificial Intelligence advancements have been a rapidly evolving field. Machine Learning and Deep Learning are being utilized to overcome historical limits in a variety of sectors, particularly in healthcare. This research project seeks to apply Deep Learning to image processing in comparisons of several neural networks for white blood cell (WBCs) recognition and classification. In addition, an algorithm called YOLO to further segment the individual white blood cells.

The dataset is acquired from the Core Laboratory at the Hospital Clinic of Barcelona through a public platform. The images acquired will then be used in four different types of neural networks: GoogLeNet, Resnet, VGG, and Squeezenet. GoogLeNet provided 98.81% accuracy in training while Resnet, VGG, and Squeezenet provided 99.19%, 99.68%, and 99.19%, respectively.

The results indicated were the efficiency of the training procedure. Which were adapted in terms of input size, output size, weight learn factor, epochs, and MiniBatchSize. With all neural networks performing exceptionally well and differing

only slightly, Resnet50 the second runner-up was considered to be used as a model along with YOLO to further advance the classification and segmentation.

The testing phase of the Resnet50 was performed with the 10% of the initial dataset that was separated beforehand. The data consists of 1,030 pictures in total of the five different classes. The training and validation stages consists of 9,268 pictures of the five classes of white blood cells. The result of the training was promising with Mini-batch RMSE of 0.45, Validation RMSE of 2.15, Mini-batch Loss of 0.2, and Validation loss of 4.64. The test results provide over 90% in each class. While the accuracy of most of the neural networks are relatively high, future studies on the advancement of dataset, efficacy of the network, and parameters required in distinguishing the data is still required.

The net efficiency of the process allowed the process of White Blood Cells classification relatively faster with a lower cost in capital. Which can be used in medical aiding or research strategies. With higher development and precision, the Deep Learning Convolutional Neural Network will be able to overcome conventional strategies and reduce the limitations towards white blood cell classification.

ACKNOWLEDGEMENTS

This project was successfully done with the aid of Prof. Dr. Chuchart Pintavirooj, our project advisor. Through his guidance and advice, we were able to formulate and work through the obstacles and achieve our desired outcome. Also, with the support from our Faculty of Engineering, Biomedical Engineering, we were able to obtain research funding and equipment required to do and finish our project.

Salila Ongtrakul

Anyarin Thitirattanapong

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
AI	Artificial Intelligence
CNNs	Convolutional Neural Networks
DIP	Digital image processing
RGB	Red Green Blue
WBCs	White Blood Cells

CHAPTER I
INTRODUCTION

This chapter introduces the report's major topics and contextualizes the work's objective. Following that, the reasons and aims for the project's investigation are described, followed by an overview of the entire project. Finally, a chapter-by-chapter description of the research is provided.

1.1 Limitations in Conventional Methods

The process of cell recognition and classification has been a demanding process in many fields. A viable cell count is essential for the study of eukaryotic cells for a variety of reasons, including cell culture management, cell population titration, as well as in medical diagnostics [1]. For its low cost and availability, manual counting using a hemocytometer has been the most widely used viable cell count method established thus far. This method relies on the analyst's ability to evaluate various cell properties and it also utilizes the benefit of various staining procedures depending on the analysis' goals. However, because the technique is time consuming, it cannot be used to analyze a large number of samples at once. The image under the microscope shows faint lines indicating sections and manual counting and dyeing techniques are used to distinguish between living and dead cells.

Although it has several inherent drawbacks, such as the need for skilled personnel, high inaccuracy, and the possibility of generating post-analytical errors due to manual transcription of data, microscopic analysis has been the standard method for quantification and classification of WBC. Automated flow cytometry, employing either urine or hematological equipment, is thus another interesting option for screening or analysis of biological fluids, as it overcomes most of the drawbacks of manual analysis previously discussed [2]. Flow cytometry is a technique for analyzing single cells in solution promptly and with multiple parameters. Flow cytometers use lasers as light sources to generate scattered and fluorescent light signals, which are detected by photodiodes or photomultiplier tubes. These signals are translated into electronic signals, which a computer interprets [3]. However, there are still certain drawbacks to this procedure. The fact that the laser can only examine one cell at a time as shown in Figure 2, and cells must be in suspension to be evaluated. Therefore, tissues cannot be analyzed. Highly skilled operators are also necessary, and cells must be viable to be analyzed.

1.2 Objectives of the study

1.2.1 Compare the accuracy of four different Neural Networks on the recognition and classification of WBCs.

1.2.2 To provide a more efficient method in WBC classification.

1.2.3 To lower the cost per capita of WBC classification

1.2.4 To implement the use of YOLO to enhance WBC segmentation and localization.

1.3 Scope of the study

1.3.1 Research the unique parameters of different Convolutional Neural Networks (CNNs).

1.3.2 Experiment on the accuracy of each model through different datasets to train the CNNs.

1.3.3 Implementation of YOLO with other neural networks

1.3.4 Collect data from the samples for training and testing of the Neural Networks

CHAPTER II

REVIEW OF DEEP LEARNING METHODS AND THE DETAILS OF WBC CLASSIFICATION

This chapter elaborates on the process of image processing, conventional white blood cell (WBC) classification, the definition of Deep Learning, definition and components of Convolutional Neural Networks (CNNs), and the structures of four types of CNNs: GoogLeNet, Resnet50, VGG16, and Squeezenet. The advantages of Deep Learning will be observed when compared to conventional methods and the differences in each CNNs will be illustrated to compare the efficacy of each individual network.

2.1 Image Processing

The advancement of computer vision and image processing systems has paved the path for any engineers to extract quantitative data sets from visual picture outputs. Digital image processing (DIP) eliminates the subjectivity of hand interpretation by properly measuring and quantifying the image [4].

With the emergence of deep learning, computer vision has shown to be effective in a variety of applications. The application of deep learning in computer vision can be divided into several categories, including image and video classification, segmentation, object detection, and detection inside images and videos [5]. Object detection is a Computer Vision task that automatically classifies various objects from images into categories of interest. Object detection, in addition to classifying images, seeks to

precisely identify the placement of objects inside the image and label the concepts to gain a better knowledge of the images.

Images can be classified into gray-scale images and RGB images. The composition of digital images consists of pixels and pixel values. The size of the images refers to the numbers of pixels in width and height. As in Figure 1, there are 22 pixels in height and 16 pixels in width, thus the size of the picture will be 22 x 16 pixels. The fundamentals of an image are pixels, and these pixels hold a value. In a gray-scale picture the values range from 0 - 255 to indicate the intensity of the color. Zero represents black while 255 represents white, the values between depends on the intensity of the pixel [6].

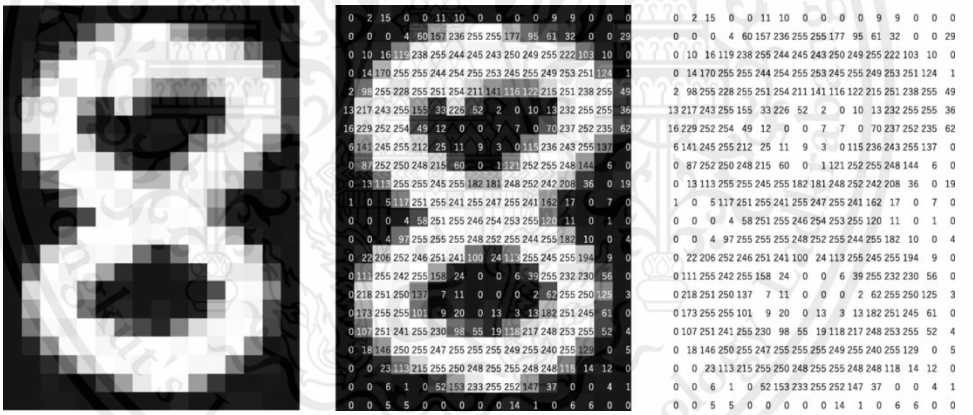

Figure 1. Gray-Scale Image

Colored Images on the other hand, possesses pixels as well, but the configuration of the channel is different. Color can be classified using the three primary colors, red, green, and blue [6]. The matrix which would usually be 1 in gray scale images, would be 3, representing the matrices of each color as shown in Figure 2. The colored images are the compilation of the three matrices, thus inserting another dimension, height x weight x 3 (representing the 3 layers).

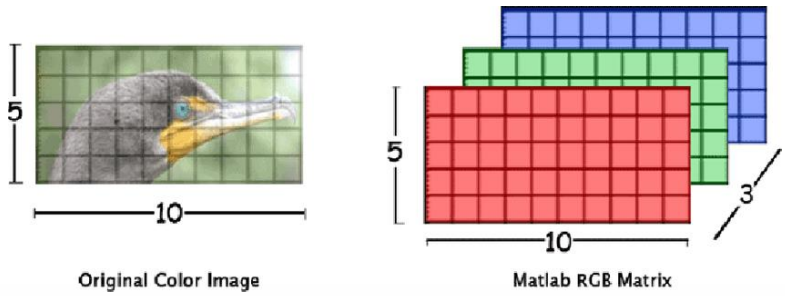

Figure 2. RGB layers in a colored image

2.2 White Blood Cell (WBC) Recognition and Classification

White blood cells, also known as leukocytes or leucocytes, are immune system cells that help to protect the body against infectious disease and foreign threats. Granulocytes and Agranulocytes are the two basic forms of white blood cells. Neutrophil, Eosinophil, and Basophil are examples of Granulocytes, whereas Lymphocytes and Monocytes are examples of Agranulocytes [7]. Each type of white blood cell differs in their physical appearance, due to its different function in the body. Classification can be clearly seen under the microscope as shown in Figure 3.

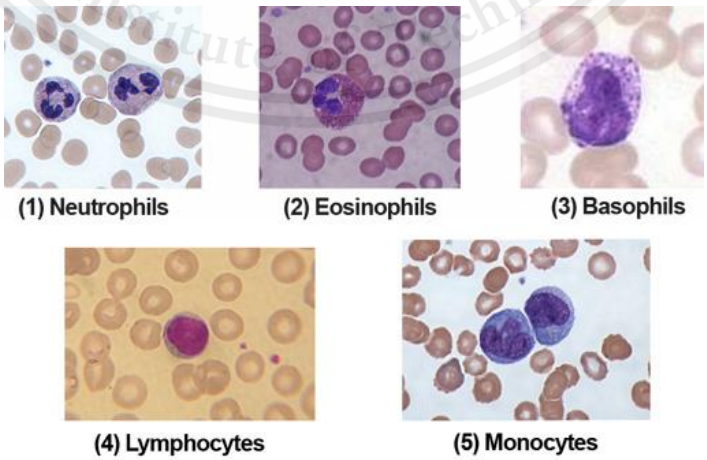

Figure 3. Classification of White Blood Cells (WBCs)

Leukocytes or white blood cells (WBCs) total count and differential count in blood samples are critical pathological indicators for illness diagnosis [8]. Manually counting WBCs is through hemocytometers, which is a thick glass chamber used to determine the number of WBCs through well-dispersed cell or nuclei suspensions. Thick glass chambers are separated into areas and depths that are calibrated. From the counts of cells in the hemocytometer chamber, the total number of cells in a suspension can be simply estimated. Figure 4 shows the structure of a hemocytometer.

Figure 4. Hemocytometer

The number of cells per grid square is counted using a microscope after a calibrated grid is placed over the culture chamber. The hemocytometer is divided into nine 1mm × 1mm main squares. The four corner squares are subdivided further into four 4 × 4 grids as shown in Figure 5. Since the cover glass creates a chamber with a height of 0.1 mm, a 1 mm x 1 mm x 0.1 mm chamber has a capacity of 0.1 mm^3 or 10^{-4} ml. Add 15-20 µl of cell suspension between the hemocytometer and the cover glass. The target is to have 100-200 cells per square. Divide the total number of cells in each of the four outer squares by four (the average number of cells per square). The number of cells per square multiplied by 10^4 equals the number of cells per milliliter of suspension [9].

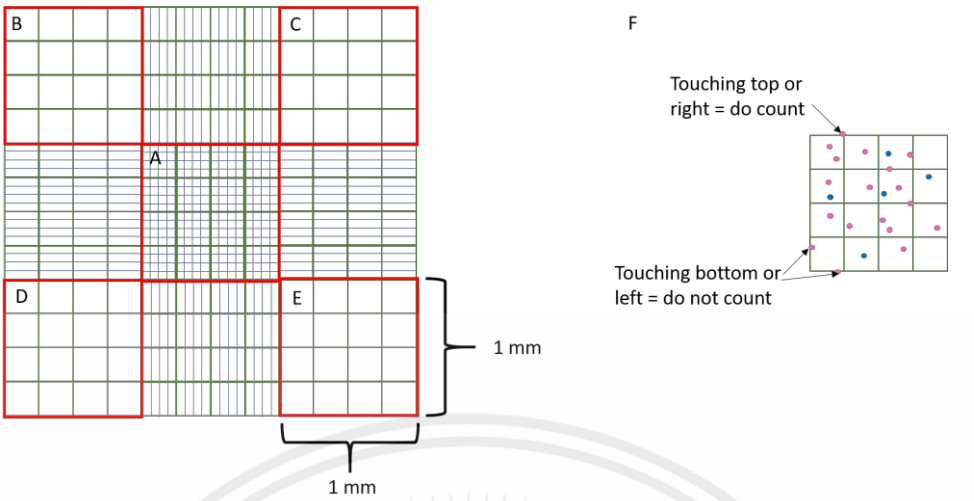

Figure 5. Counting Methods of Hemocytometer.

2.3 Deep Learning

New discoveries have emerged to complement the rapidly rising knowledge as the globe has developed towards a technology-based community. The deployment of Artificial Intelligence (AI) to further expand the limitations is one of the most essential advancement in many fields. In this research we will concentrate on Deep Learning, a subset of Machine Learning. Machine Learning is a more advanced version of AI that allows machines and software programs to adapt or predict outcomes based on the data they are provided. While Machine Learning is a successful sophisticated pathway that is utilized in a variety of applications, it is still necessary to employ AI in a more elaborate context.

Deep Learning makes use of structures that are modeled after the human brain. To handle complex problems, the framework of neural networks interconnected in multiple layers is used. The pathway is built up of neural networks and complicated

algorithms, along with a training system that allows the machine to 'learn' from enormous volumes of data. Deep Learning is widely utilized in picture classification, object identification, image retrieval, biomedical imaging, and signal processing, among other applications [10]. Deep Learning is commonly employed in biological and medical disciplines, but it is also used to advance everyday technology.

Deep Learning resembles the structure of neurons, where the weights (w) serve as dendrites, the artificial neuron serves as the cell body, and the output as signals from the axon as shown in Figure 6. Inputs (x) will be calculated by multiplying with the weights of the pathway and then the sum of each pathway will be added together ($\sum \chi_i \, \omega_i$) [11]. In some cases, a biased parameter will be added ($\sum \chi_i \, \omega_i + \mathscr{b}$). This is used to adjust the output, as well as the weighted sum of the neuron's inputs. As a result, bias is a constant that aids the model in fitting the data the best it can.

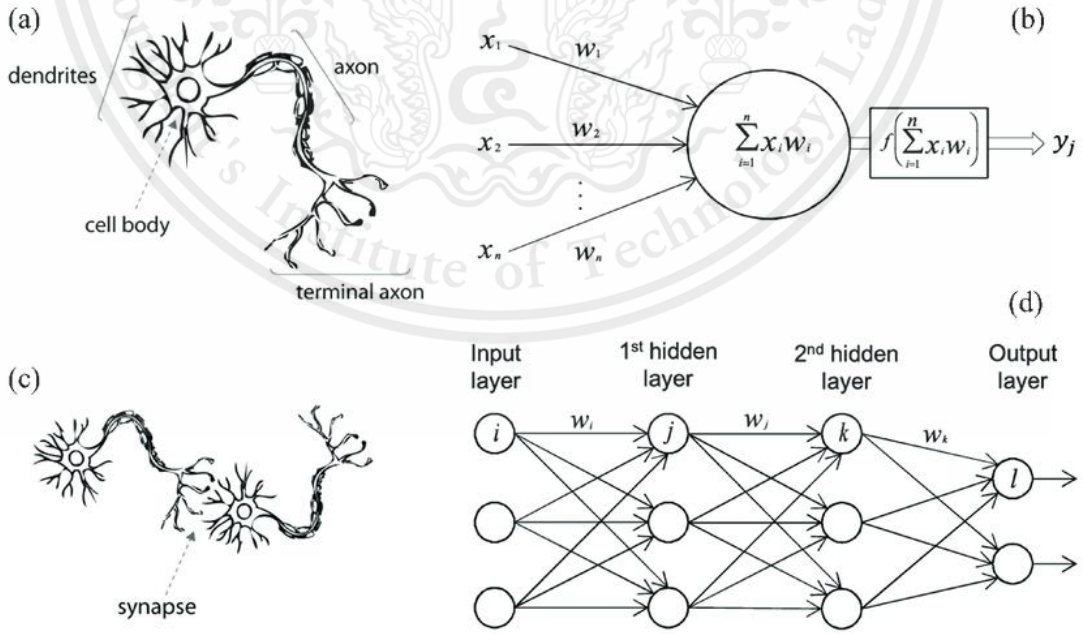

Figure 6. Comparisons of a biological neuron and an artificial neuron

Apart from the layered structures of the repeating units of artificial neurons, the process in which the machine is trained is through processes called feed-forward neural network and backward propagation.

Feed forward neural network describes a one-way direction of the input information through the channels. While the backward propagation enhances the ability for the network to optimize the weights and biases to allow the neural network can learn how to correctly map arbitrary inputs to outputs. In Figure 7 the value of each output parameter (ox) is labeled accordingly, with target values shown at the end. Backward propagation calculates the error by calculating the sum of half the difference of the target and output value, $\sum \frac{1}{2}(target - output)^2$. The weights (w), hidden layers (h), and biases (b) are then recalculated according to the weight change. This reduces the error and allows the machine to learn from the dataset and provides a higher accuracy towards the net accuracy of the neural network.

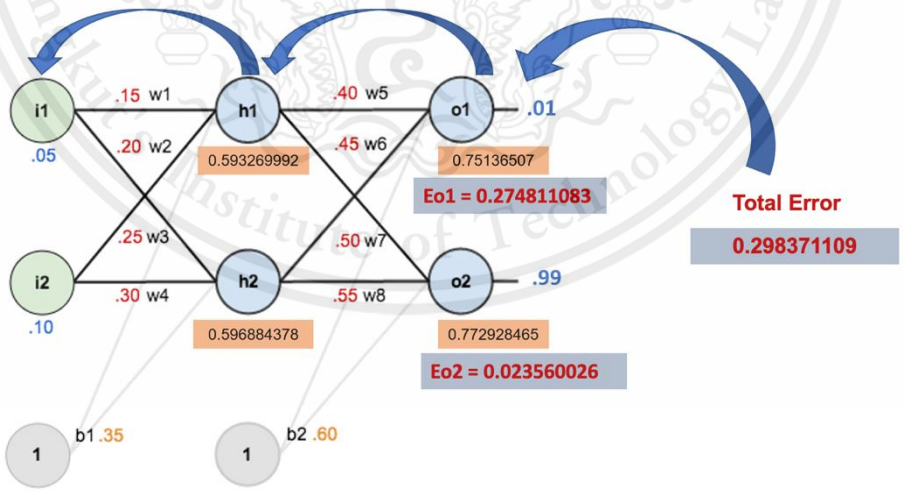

Figure 7. Back propagation of the neural network

2.4 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) is a type of Deep Learning approach where multiple layers are trained. The architecture of CNNs mainly consist of three different layers: convolutional layers, pooling layers, and fully connected layers [12]. These can vary depending on the engineer and purpose of each design.

2.5 Layers of CNNs

The first layer is the convolution layer which extracts features of the image through a combination of linear and non-linear operations. The input tensor is extracted as a kernel and directed onto the feature map. The combination of kernels allows the different characteristics of input tensors to be represented on the feature map, as shown in Figure 8. The hyperparameters (parameters involved in the learning process) that define the operation is the size and number of the kernels, which can generally be 3x3, 5x5, or 7x7 [13].

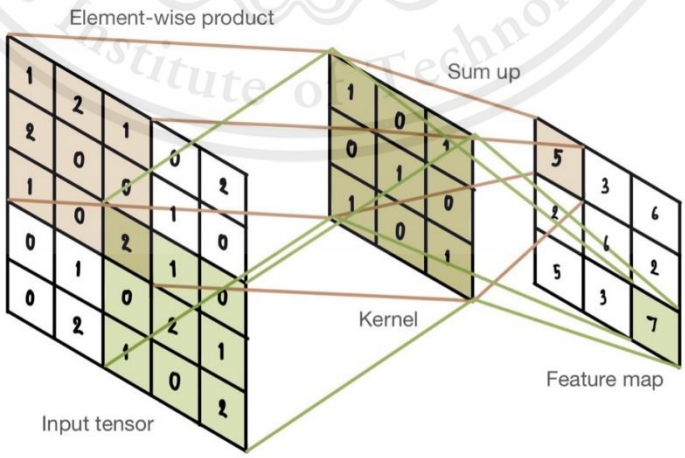

Figure 8.
Extraction of data from input tensor through kernels to display on feature maps

The following layer of the CNNs network is the pooling layer that down-samples the complexity or dimension for further layers [12]. Max pooling is the most common approach of pooling, where the image is dissected into smaller regions and the maximum value of each section is passed on to the next layer. The most common sizes of max pooling used is 2x2 [14]. As shown in Figure 9, the filter size is 2 x 2, with a stride, a parameter of the neural network's filter that defines the amount of movement across an image, of 2.

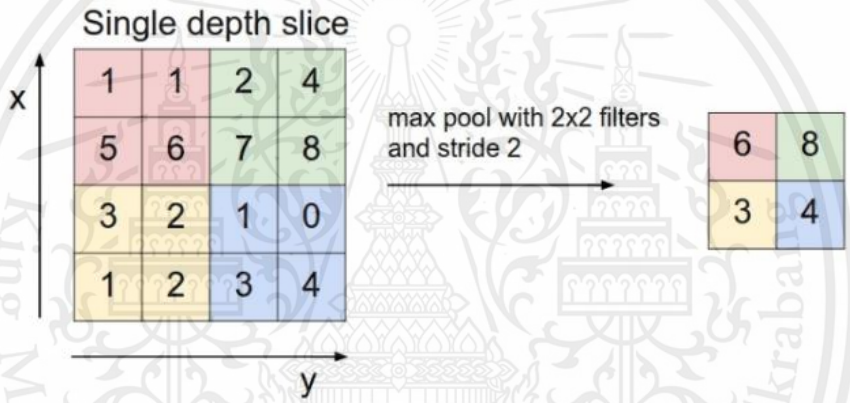

Figure 9. Max pooling of a 4 x 4 input with a 2 x 2 filter.

The final layer of the CNNs is the fully connected layer. Fully connected layers are multilayers of nodes that functions similarly to neurons in a neural network as shown in Figure 10. The nodes interconnect to form complex analysis. These layers are the most complex of all the layers and contain about 90% of the total parameters [12]. It requires high computational efforts in training. Alexnet is an example of a CNNs that utilizes each type of layer with 2 fully connected layers. It holds 650,000 neurons, 60 million parameters and 630 million connections [14].

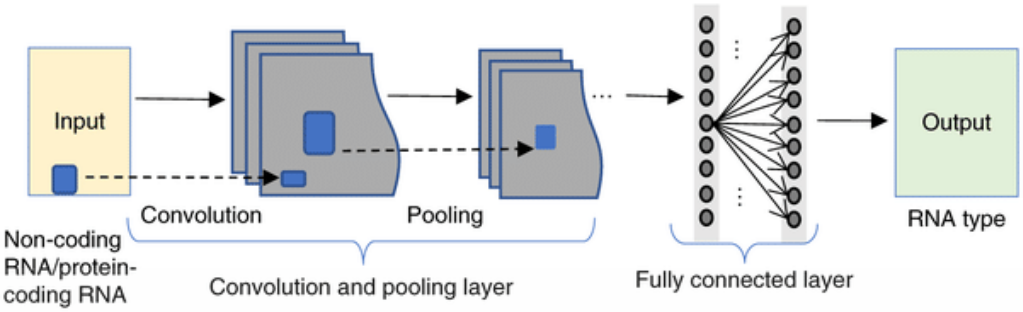

Figure 10. Structure of CNNs.

In Convolutional Neural Networks, the net process reduces the size of the image due to matrix calculation, which could result in loss of information. A technique called padding is introduced to the steps to prevent the loss. Padding, usually zero padding, is a solution for dealing with this problem that involves adding rows and columns of zeros on each side of the input to maintain the same in-plane dimension during the convolution procedure. In order to apply more layers, modern CNN designs commonly use zero padding to keep in-plane dimensions. After the convolution operation, each consecutive feature map would be smaller if there was no padding as shown in Figure 11 [20].

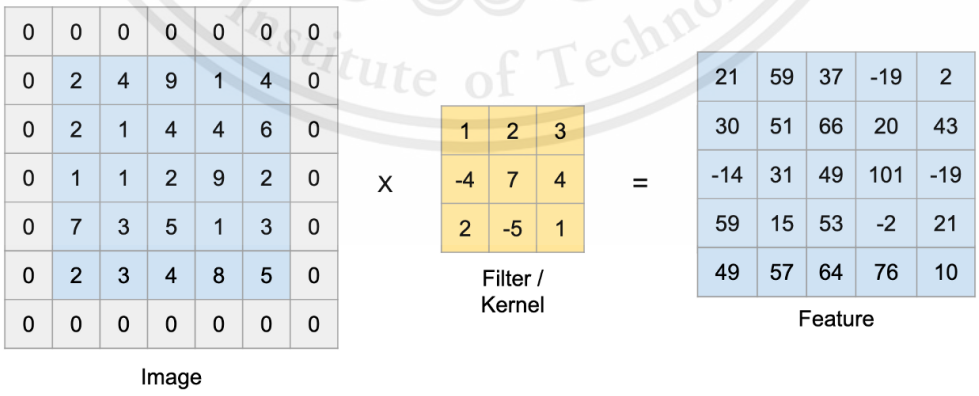

Figure 11. Zero-padding on a 5 x 5 size input to prevent loss in the output tensor.

2.6 Structures of Neural Networks

There are many neural networks available commercially and, in this research, we will apply four different networks: GoogLeNet, Resnet, VGG, and Squeezenet, to classify WBCs. There are a few factors we are considering adjusting to affect the training process in this research. The input, output, fully connected layers, MaxEpochs, MiniBatchSize, Validation Frequency, and Initial learning rate of each neural network, as shown in Figures 12 and 13.

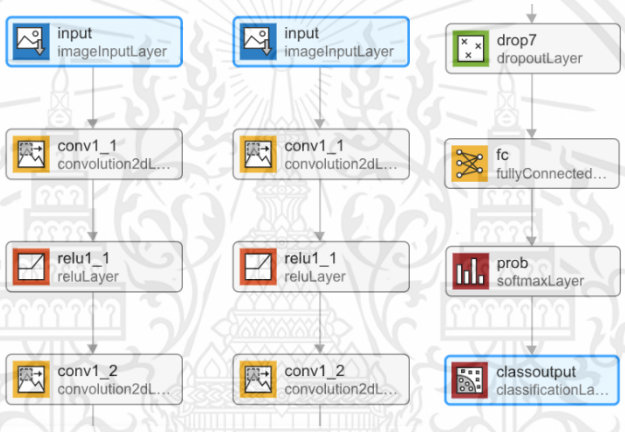

Figure 12. Adjustments made: Input size at 244,244,3, Fully Connected Layers have an output size of 5. Output layer has an auto output size setting.

Figure 13. MaxEpochs and MiniBatchSize was changed according to the performance of the neural network to help amplify and better the training processes. As well as the Validation frequency and Initial Learning Rate.

2.6.1 GoogLeNet

GoogLeNet is a CNN that is pretrained with a dataset called ImageNet. The architecture consists of 22 layers, 12 times less layers than Alexnet [15]. Figure 14 displays the structure of GoogLeNet.

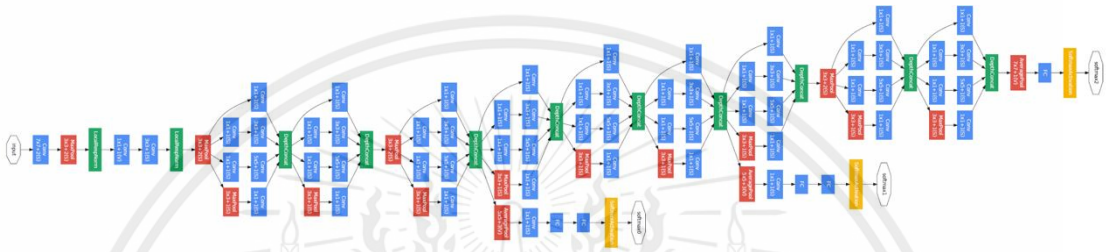

Figure 14. Structure of GoogLeNet

2.6.2 Resnet50

Resnet is another type of CNN model, as shown in Figure 15, which stands for Residual Network. It has three versions including ResNet-34, ResNet-50, and ResNet-101, with the numbers indicating the neural network layers [16]. One of the strengths of Resnet is the concept of skip connections. Skip connections open up an alternative pathway for information to pass through, thus alleviating vanishing gradient. The output of previous layers can be combined to other layers of output, increasing the potential of training of deep networks [16].

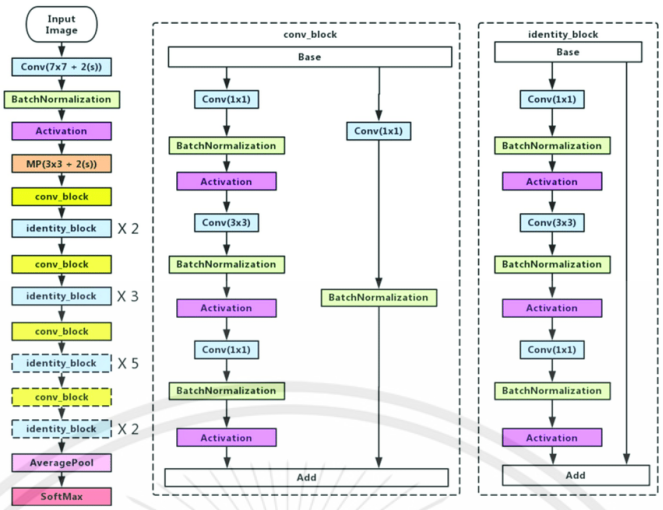

Figure 15. Structure of Resnet-50

2.6.3 VGG16

VGG or VGGNet is a successful CNNs proposed by Simonyan & Zisserman, 2014 [17]. VGG-16 refers to the compilation of 13 convolutional layers and 3 fully connected layers connected together to form 16 layers as shown in Figure 16. With a total of 138 million parameters, VGG-16 has been one of the most used architectures [18].

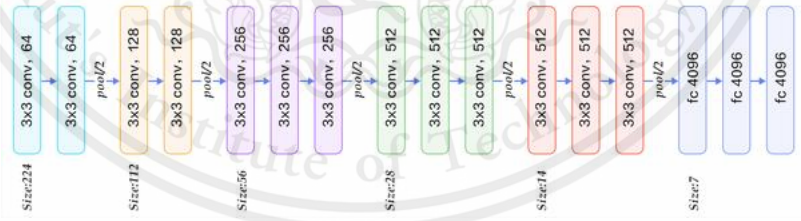

Figure 16. Structure of VGG-16

2.6.4 Squeezenet

The final CNNs is the Squeezenet architecture. Squeezenet 1x1 convolution kernels to reduce the computation on 3x3 kernels, which resulted in 50 times less parameters than AlexNet but an overall similar accuracy [19]. The architecture of Squeezenet is as shown in Figure 17.

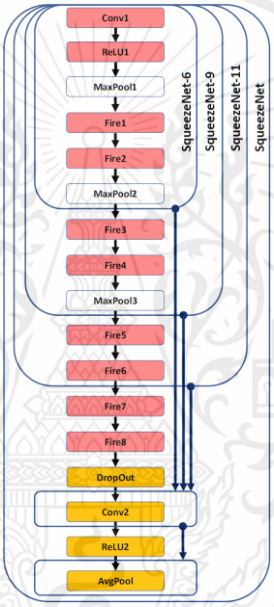

Figure 17. Structure of Squeezenet

CHAPTER III
METHODOLOGY

3.1 Action Plan

Action Plan for the Recognition and Classification of White Blood Cells							
Week / Month	January	February	March	April	May	June	July
Week 1	Research software used in deep learning	Test resnet and alexnet model	Apply to each model	Perform diagnostic test and prepare binary model	Perform diagnostic test and prepare multiclass model	Analyze data	Finalization and presentation of study
Week 2			Debugging				
Week 3	Test google and lenet models	Find and classify data	Adapt chosen model with Yolo	Perfect binary model	Perfect multiclass model	Interpret data and results	
Week 4		Apply to each model					

3.2 Introduction

The development process of the neural network will be illustrated in the following chapter. Including the details of the dataset, the testing procedures, adjustments, progress of each step and the final achieved result of the combined neural network and YOLO recognition.

3.3 Data

The data in this research is from the Core Laboratory at the Hospital Clinic of Barcelona by the analyzer CellaVision DM96 (RGB, 360x363 pixels). The dataset contains 17,092 digital images in total with 8 categories of blood cell images. Five

categories of white blood cells were chosen, neutrophils, eosinophils, basophils, lymphocytes, and monocytes. The total images used is 10,298 pictures with 3,329 neutrophils, 3,117 eosinophils, 1,218 basophils, 1,214 lymphocytes, and 1420 monocytes. The dataset is separated into training data, validation data, and testing data, 70%, 20%, and 10%, respectively. Figure 18. shows a sample of the data.

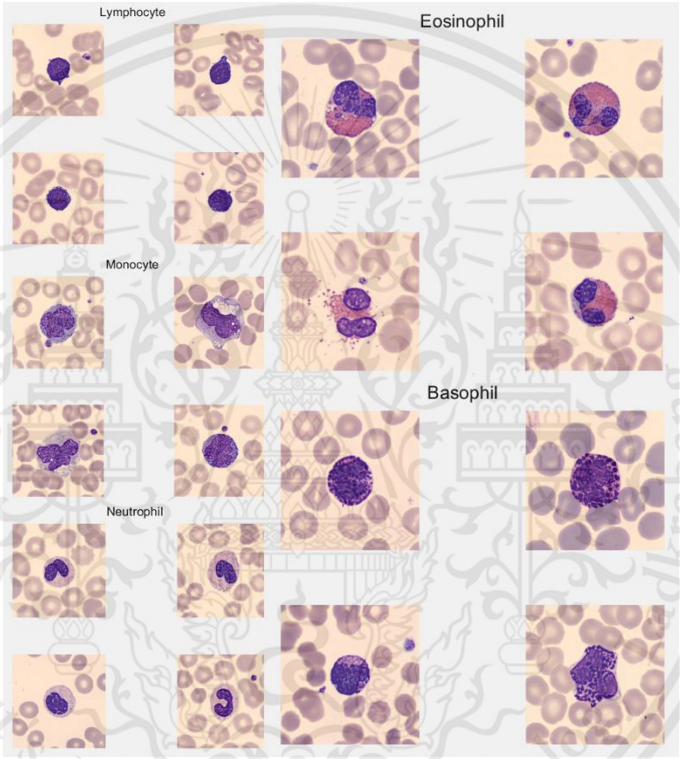

Figure 18. Sample of each white blood cell category of the dataset

3.4 Software preprocesses

The software used in this study is MATLAB. The interface and settings included in the application on MATLAB, Deep Network Designer, will be discussed in the following section. These two interfaces are used in the testing of four different neural networks, GoogLeNet, Resnet50, VGG-16, and Squeezenet. The results of each neural network will be evaluated and the most suitable one will be chosen to be implemented together with YOLO for further segmentation.

3.4.1 MATLAB Interface

The MATLAB interface consists of folder section, workspace, command window, and editor window as shown in Figure 19. The folder section states the location of the current folder of the file and keeps track of the file address and location. The workspace window keeps track of the variables and data from importing files and data from the editor window. The command window provides a short-term coding evaluation as well as process tracing and debugging. The final window, editor window, is where most activity happens, including the main coding part of MATLAB.

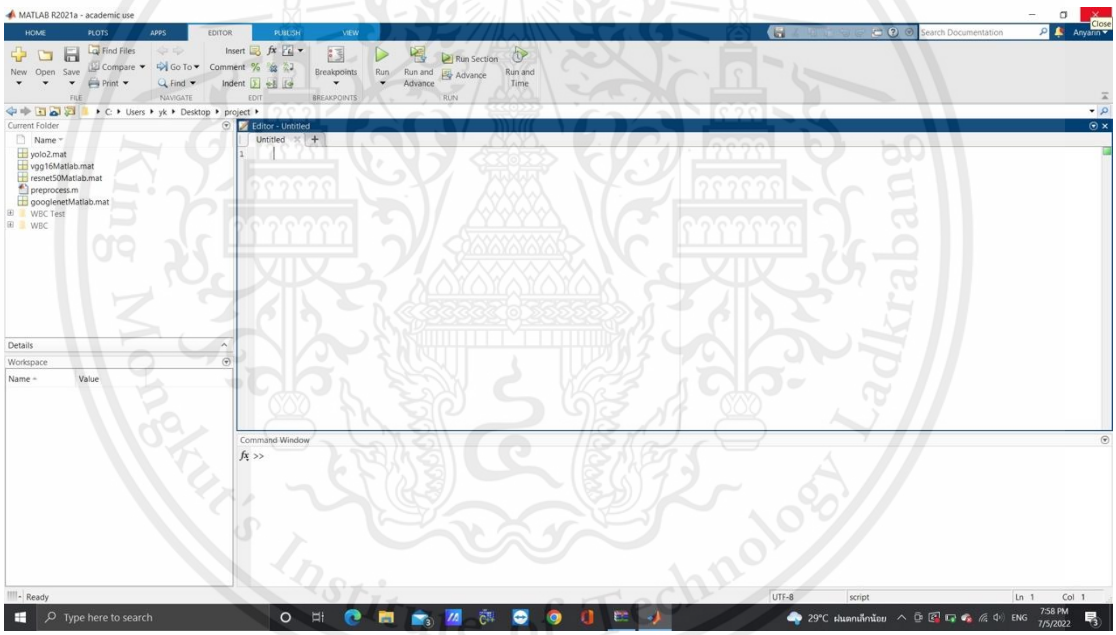

Figure 19. MATLAB Interface

3.4.2 Deep Network Designer

Deep Network Designer is an application provided by MATLAB with three sub interfaces including the designer, data, and training. The Designer is made to import neural structures into the application. Pre-made structures can be imported in and adjusted towards our preference or we can design new structures from the building blocks provided, shown in Figure 20. The data section, shown in Figure 21, allows us to import data from files and the image datastore. The training page provides settings related to the training procedure such as the MiniBatchsize. It tracks whole training process and displays it in two graphs, accuracy, and loss. The detail of the trained program is displayed when finish training and the software can be exported and use to classify other data as shown in Figure 22.

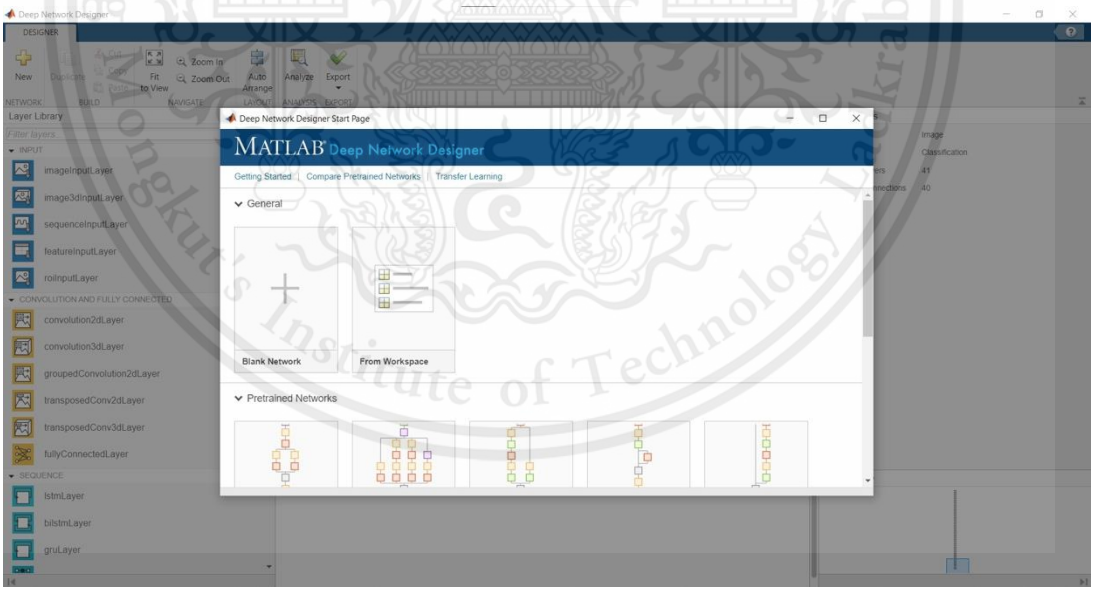

Figure 20. Designer Interface of the Deep Network Designer

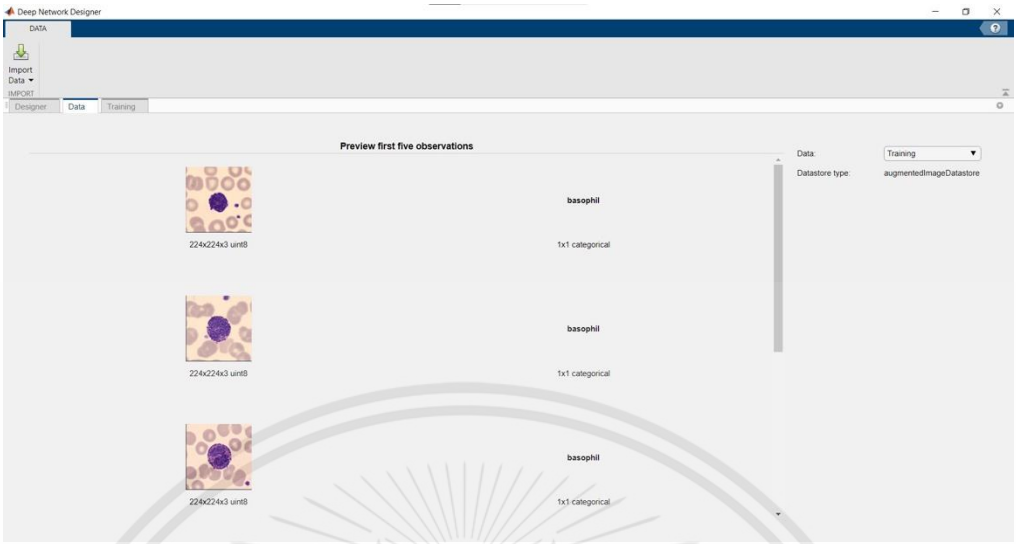

Figure 21. Data Interface of the Deep Network Designer

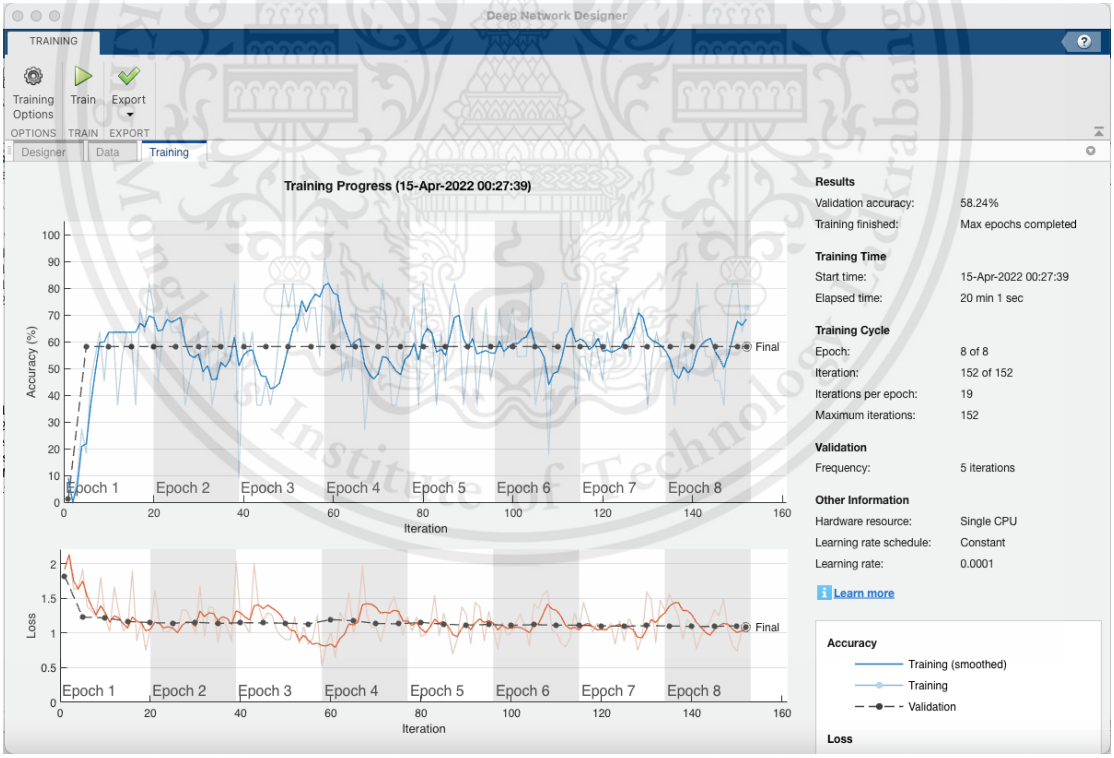

Figure 22. Training interface of Deep Network Designer

3.5 Training each network

Before training, there are a few procedures to follow. In order to prepare the data, we first divide it into training and validation data. The data is then transformed into a similar size and set of preferences. Figure 23 displays the preparation's code with details of the size adjustments and other parameters.

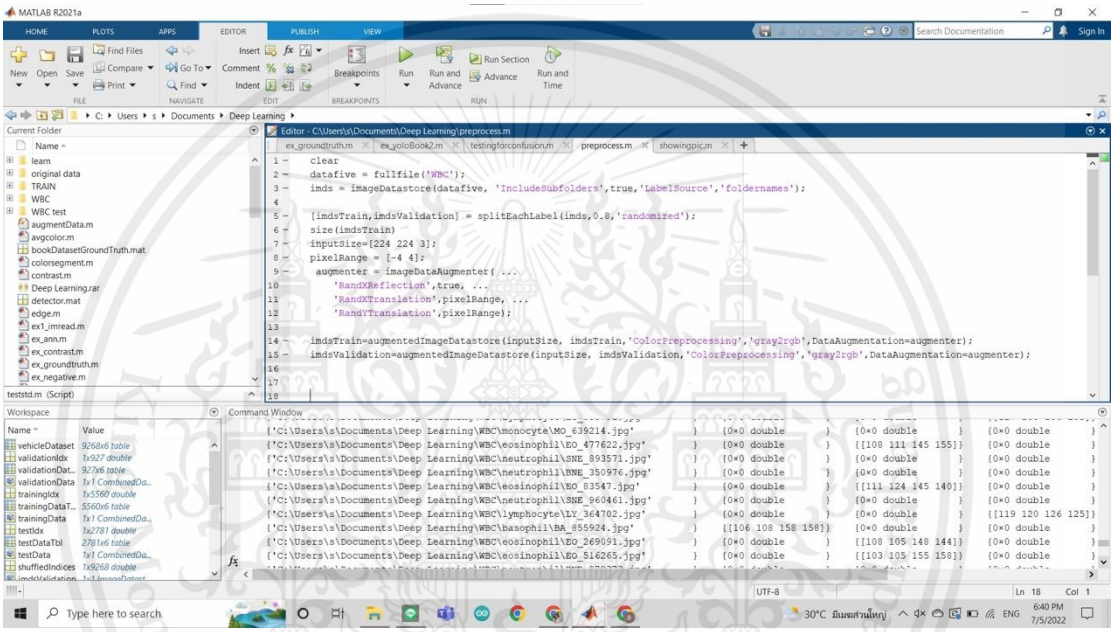

Figure 23. Preprocess code to prepare and store data for the image datastore

The next step is to import the network structure into the Deep Network Designer. GoogleNet, Resnet50, Squeezenet, and VGG-16 structures are imported and adjustments can be seen in Figure 24, 25, and 26, which are settings that needs to be changed in order to suit the classification process.

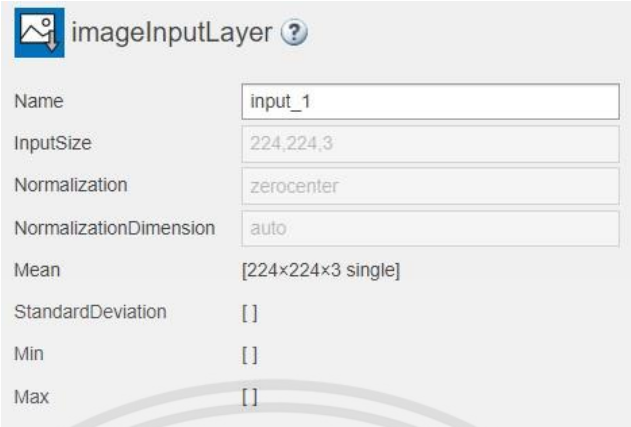

Figure 24. The input size is set according to the preprocess

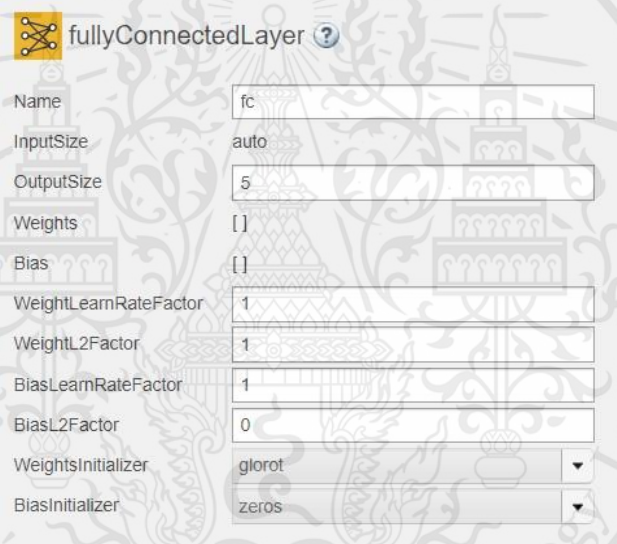

Figure 25. The final fully connected layer adjusted to 5 for the five classes of WBCs.

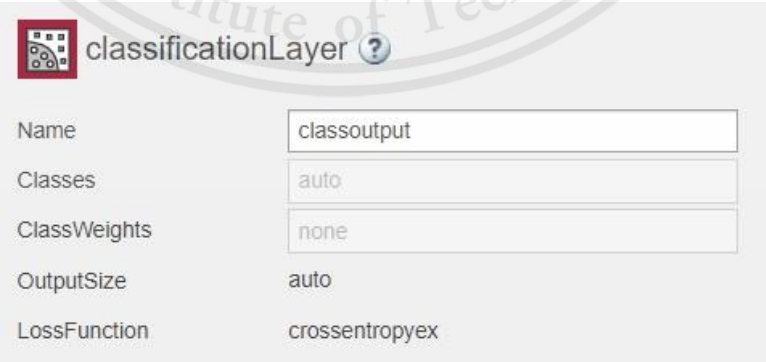

Figure 26. The output layer is changed to auto to compliment the fully connected layer.

From the pre-process in the previous section, the data will be extracted through into the data section and selection of training data and validation data.

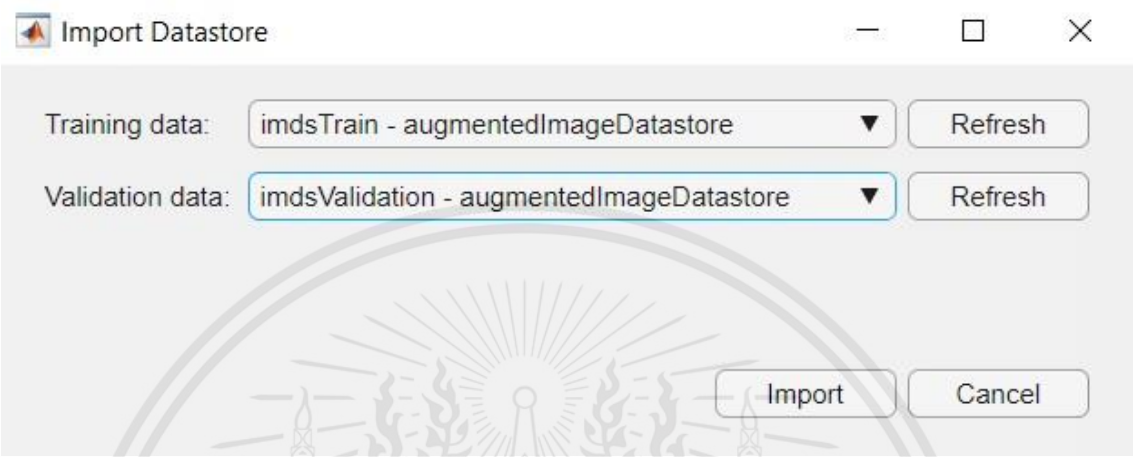

Figure 27. Importing data from the datastore from classification

The training section settings are changed at the initial learn rate, validation frequency, max epochs, and mini batch size, as shown in Figure 28.

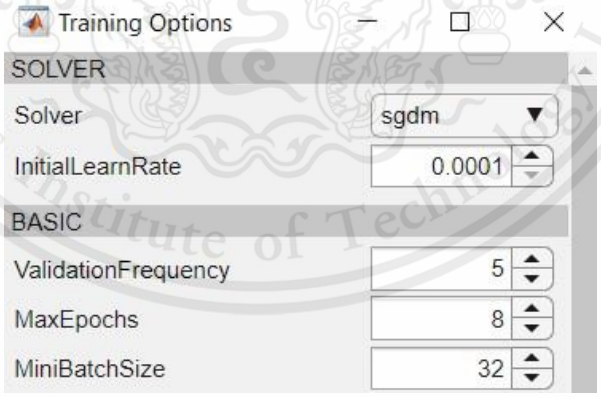

Figure 28. Training options for the neural networks

3.6 Output of CNNs

This section presents the results of each CNNs to further on decide the most suitable network to implement with YOLO. The accuracy of the four different networks, GoogLeNet, Resnet50, VGG-16, and Squeezenet is displayed in Figures 29, 30, 31, and 32, respectively.

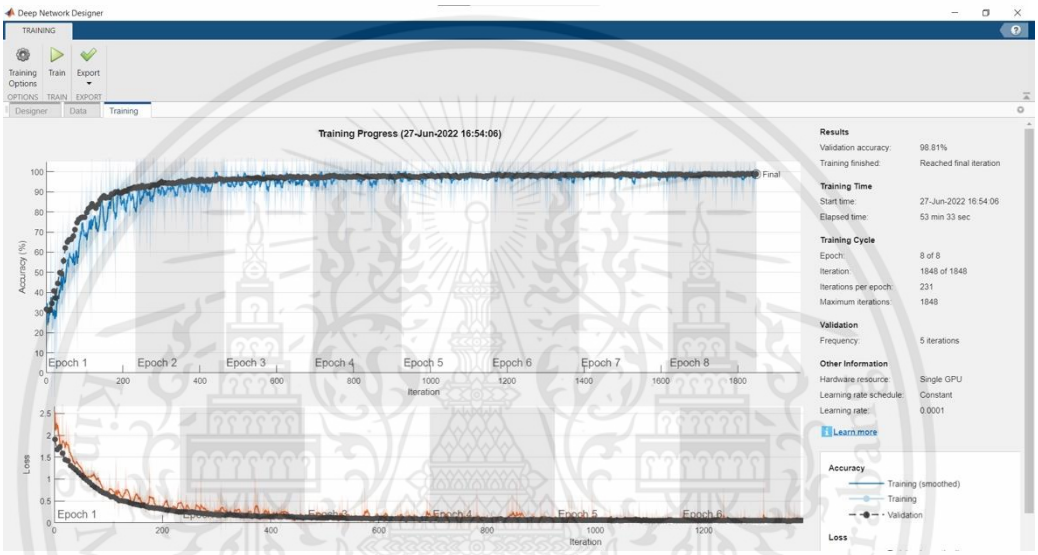

Figure 29. GoogLeNet results in training

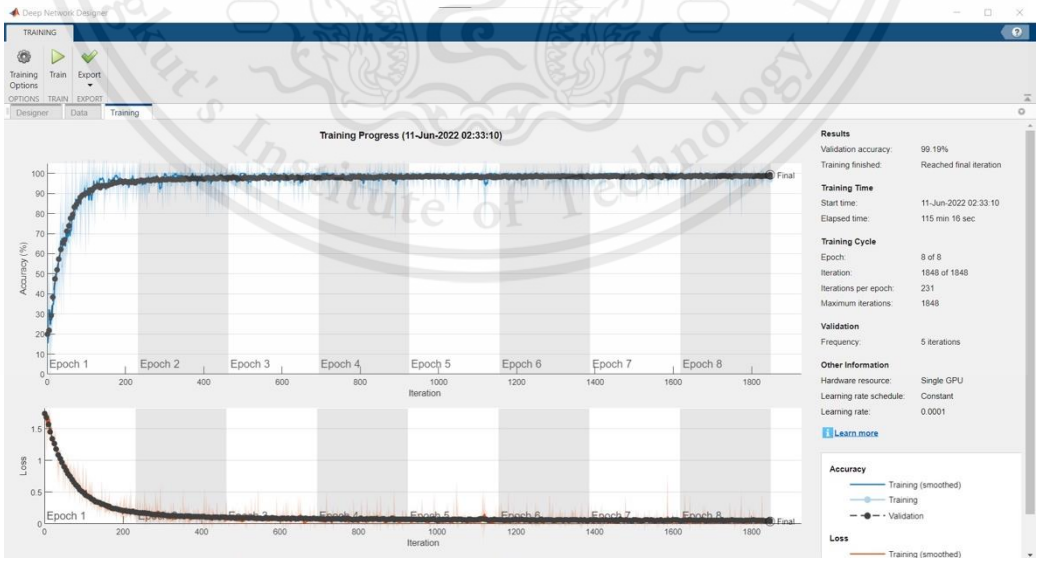

Figure 30. Resnet50 results in training

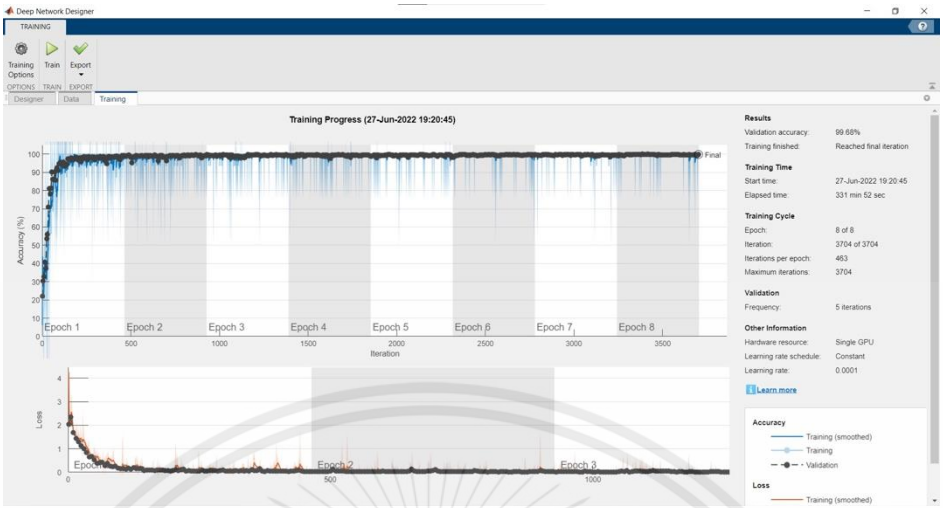

Figure 31. VGG-16 results in training

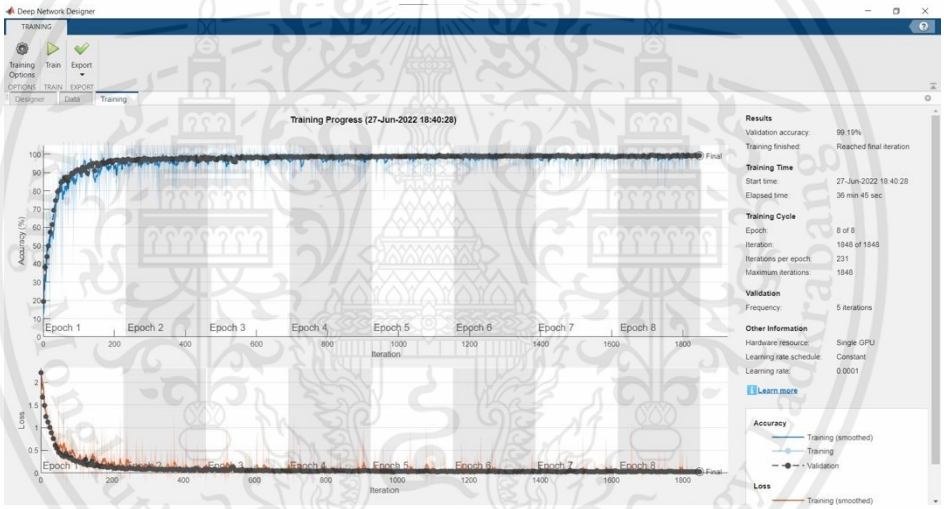

Figure 32. Squeezenet results in training

The results of the training are relatively similar with VGG-16 at 99.68%, Resnet50 and Squeezenet at 99.19% and finally GoogLeNet at 98.81%. However, Resnet50 was chosen to be used together with YOLO due to its advancement in architectural design. The testing accuracy of Resnet50 has a 99.09% accuracy in

Basophils, 99% in Eosinophils, 97.18% in Lymphocytes, 97.62% in Monocytes, and 99.37% in Neutrophils as shown in Figures 33, 34, 35, 36, and 37 respectively.

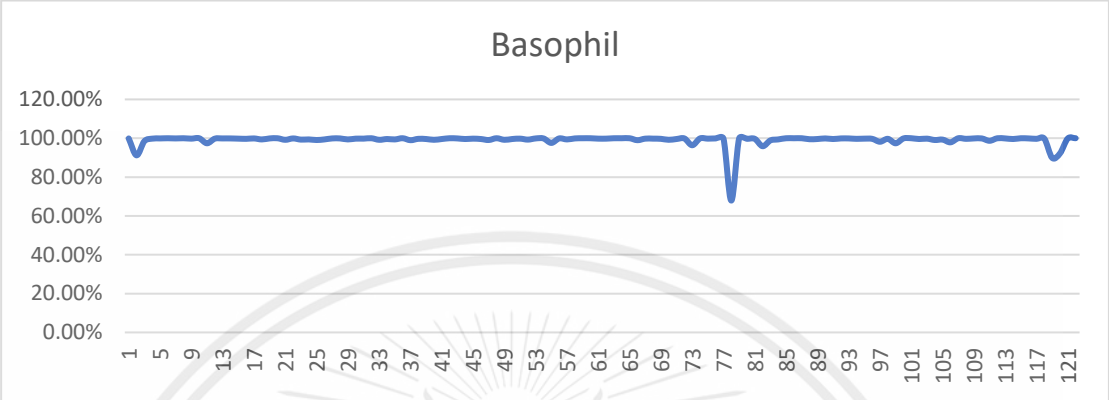

Figure 33. Basophil accuracy in testing

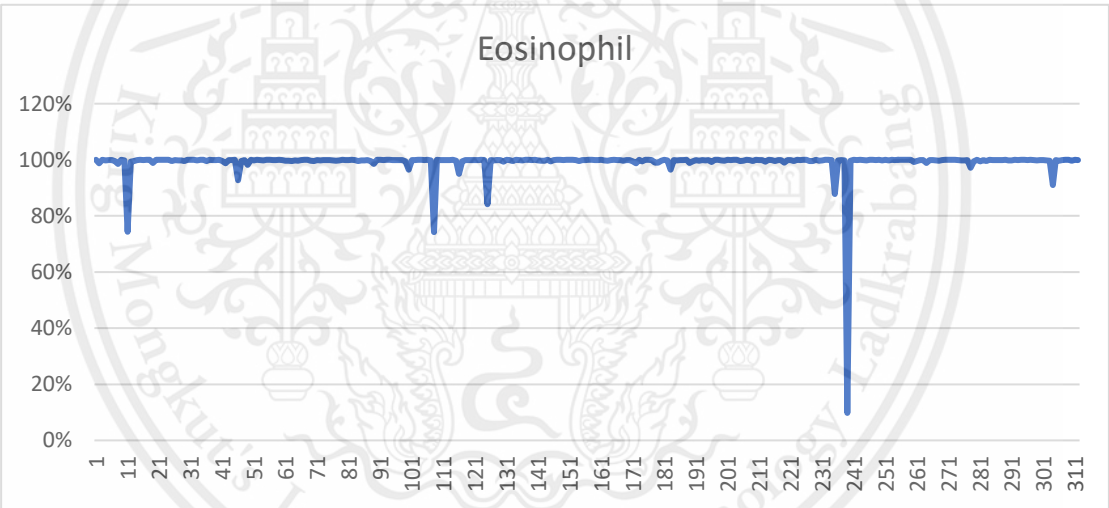

Figure 34. Eosinophil accuracy in testing

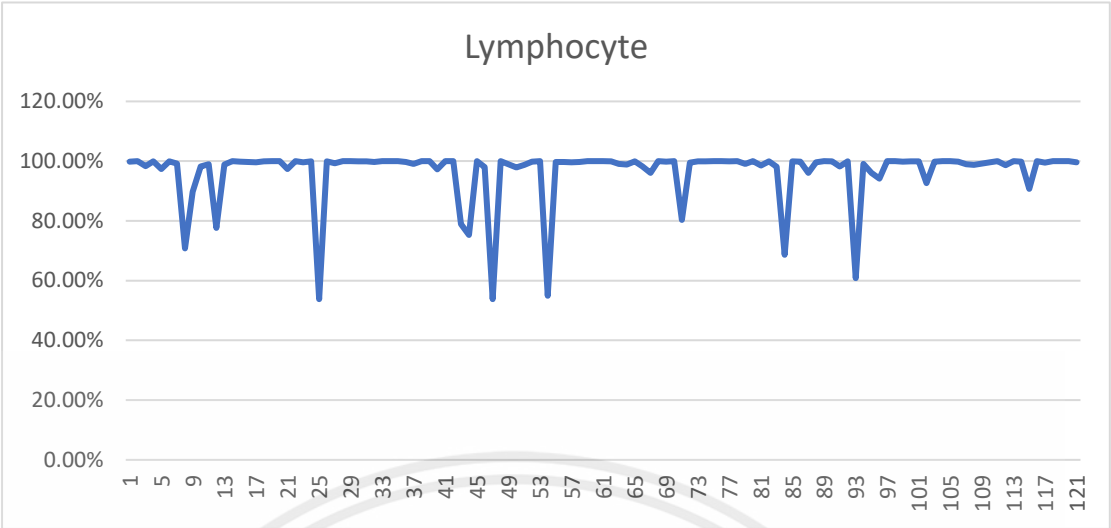

Figure 35. Lymphocyte accuracy in testing

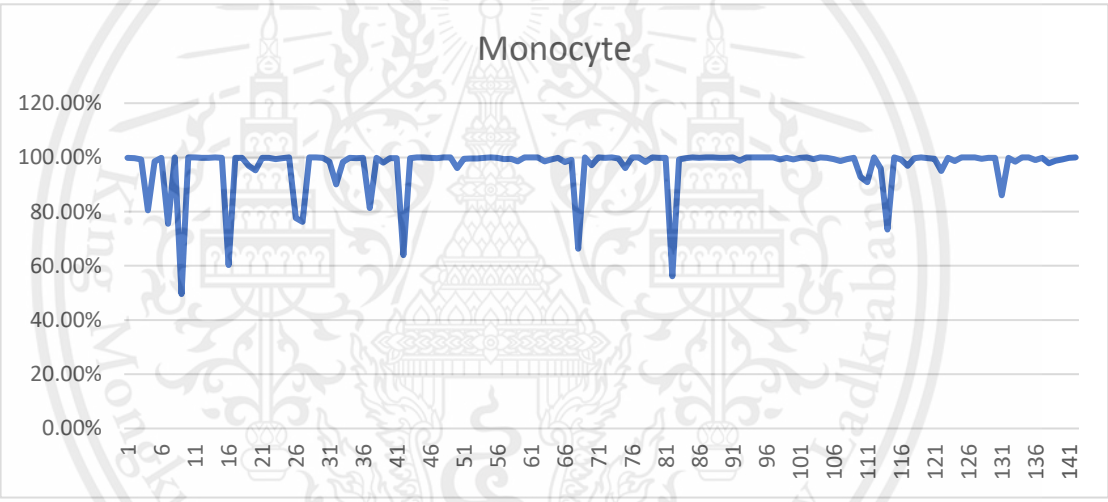

Figure 36. Monocyte accuracy in testing

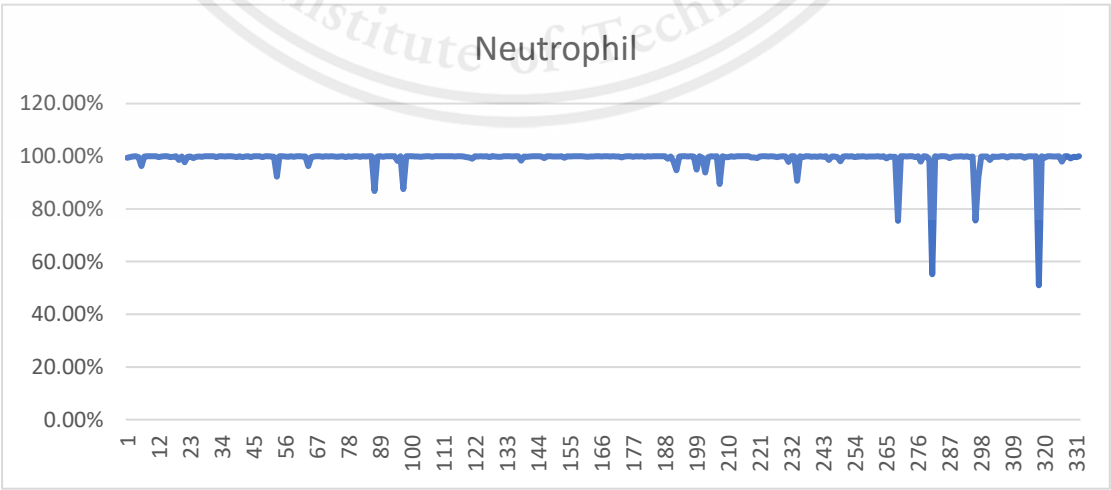

Figure 37. Neutrophil accuracy in testing

3.7 Training with YOLO

The Image Labeler software, which is available on MATLAB, is required for the training process with YOLO. This is used to pre-train YOLO's localization feature. The images will be made accessible for manual selection by importing the data file into the image labeler. As seen in Figure 38, there are five separate classes of white blood cells (WBCs), and each class's label will be colored accordingly. While Figures 39, 40, 41, 42, an 43, displays examples of manual selection of each class of white blood cells (WBCs).

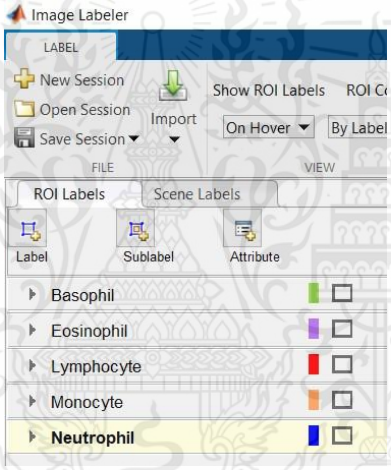

Figure 38. Image Labeler functions

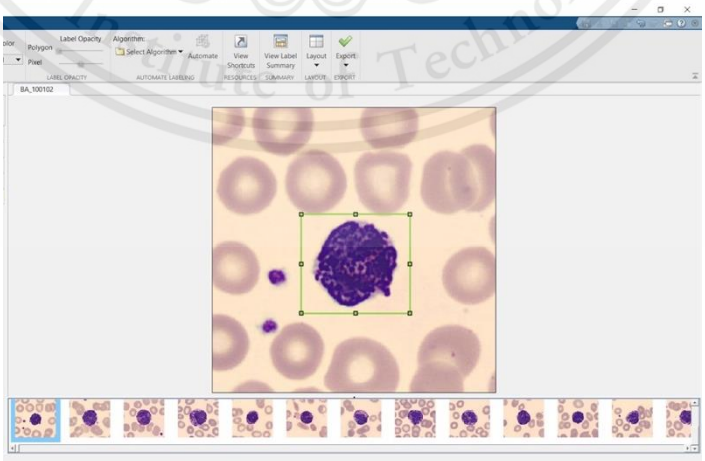

Figure 39. Basophil example of labelling

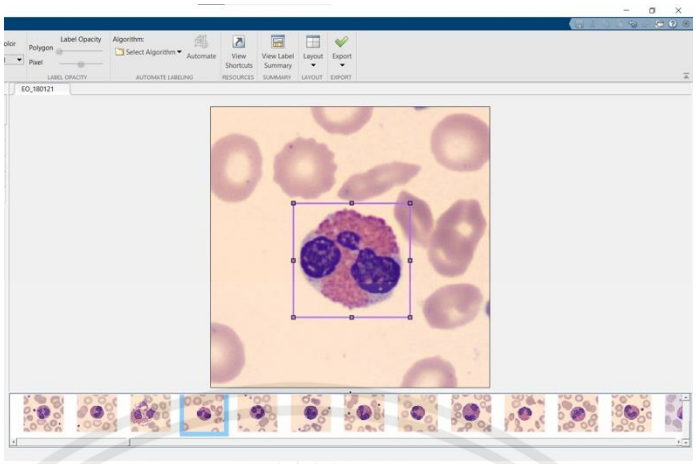

Figure 40. Eosinophil example of labelling

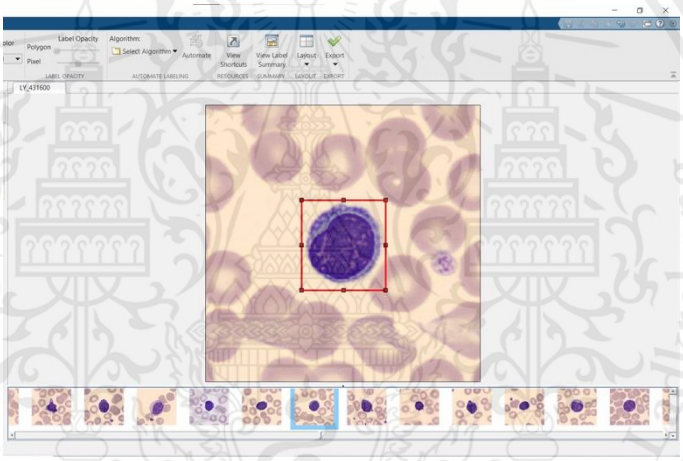

Figure 41. Lymphocyte example of labelling

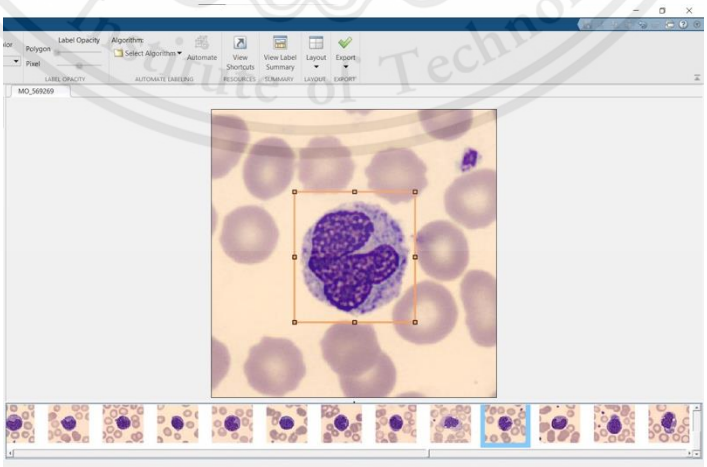

Figure 42. Monocyte example of labelling

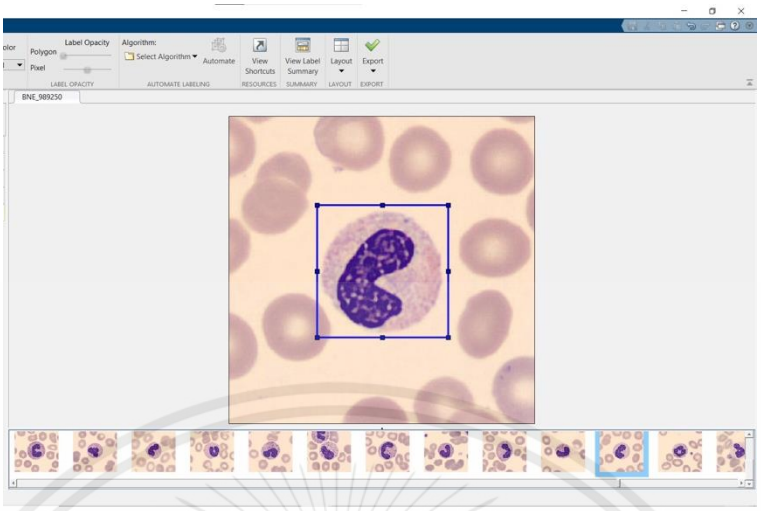

Figure 43. Neutrophil example of labelling

The label summary, as shown in Figure 44, is a spreadsheet of the occurrence of each class throughout the whole sample data.

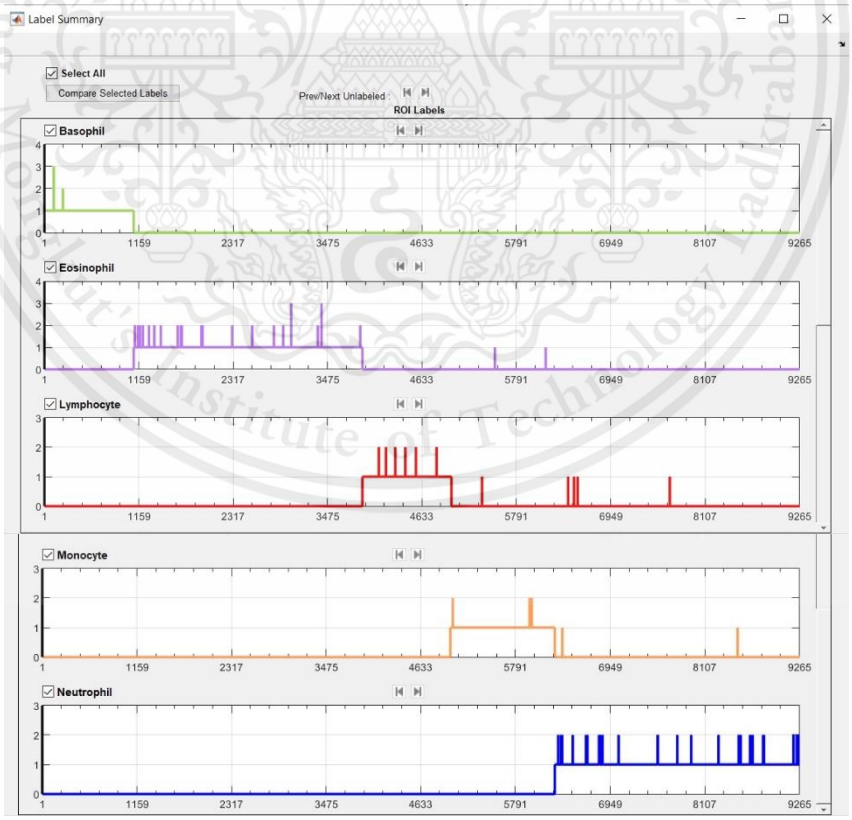

Figure 44. Labeling Summary

When the training data is finished, the entire dataset is exported back to the main interface where the training process will happen. The code depicted in Figures 45 and 46, show the overall code for the training procedure using Resnet50 and YOLO together to classify and segment the white blood cells.

```
clear
doTraining = true;

data = load('yolowbc.mat');
vehicleDataset = data.gTruth;

% Display first few rows of the data set.
vehicleDataset(1,:)
% Add the fullpath to the local vehicle data folder.
%vehicleDataset.imageFilename = fullfile(pwd,vehicleDataset.imageFilename)

rng(0);
shuffledIndices = randperm(height(vehicleDataset));
idx = floor(0.6 * length(shuffledIndices) );

trainingIdx = 1:idx;
trainingDataTbl = vehicleDataset(shuffledIndices(trainingIdx),:)

validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
validationDataTbl = vehicleDataset(shuffledIndices(validationIdx),:);

testIdx = validationIdx(end)+1 : length(shuffledIndices);
testDataTbl = vehicleDataset(shuffledIndices(testIdx),:);

imdsTrain = imageDatastore(trainingDataTbl{:,'imageFilename'});
bldsTrain = boxLabelDatastore(trainingDataTbl(:,2:end));

imdsValidation = imageDatastore(validationDataTbl{:,'imageFilename'});
bldsValidation = boxLabelDatastore(trainingDataTbl(:,2:end));
```

Figure 45. Code for training procedure [1]

```
imdsTest = imageDatastore(testDataTbl{:,'imageFilename'});
bldsTest =boxLabelDatastore(trainingDataTbl(:,2:end));

trainingData = combine(imdsTrain,bldsTrain);
validationData = combine(imdsValidation,bldsValidation);
testData = combine(imdsTest,bldsTest);

data = read(trainingData);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I,'Rectangle',bbox);
annotatedImage = imresize(annotatedImage,2);
figure
imshow(annotatedImage)

inputSize = [224 224 3];
inputImageSize = [224 224 3];
numClasses = width(vehicleDataset)-1;

trainingDataForEstimation = transform(trainingData,@(data)preprocessData(data,inputSize));
numAnchors = 7;

[anchorBoxes, meanIoU] = estimateAnchorBoxes(trainingDataForEstimation, numAnchors)

 load resnet50.mat;
network = lgraph_1;
featureLayer = 'activation_40_relu';
lgraph = yolov2Layers(inputImageSize,numClasses,anchorBoxes,network, featureLayer)
```

Figure 46. Code for training procedure [2]

CHAPTER IV

EXPERIMENTAL RESULTS

4.1 Training Outcome

The training included 8 epochs as shown in Figure 47 and 48. The results show a low root mean square deviation (RMSE) as well as loss. Which proves an excellent accuracy of the network. With Figure 49, displaying the training loss during the training procedure.

```
Training a YOLO v2 Object Detector for the following object classes:

* Basophil
* Eosinophil
* Lymphocyte
* Monocyte
* Neutrophil

Training on single GPU.
Initializing input data normalization.
|=========================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     RMSE     |     RMSE     |     Loss     |     Loss     |      Rate       |
|=========================================================================================================================|
|       1 |           1 |       00:00:38 |         9.38 |         7.74 |      87.9075 |      59.8638 |    1.0000e-04 |
|       1 |          50 |       00:02:32 |         1.94 |         2.08 |       3.7775 |       4.3309 |    1.0000e-04 |
|       1 |         100 |       00:04:25 |         1.32 |         1.60 |       1.7303 |       2.5575 |    1.0000e-04 |
|       1 |         150 |       00:06:03 |         1.26 |         1.63 |       1.5966 |       2.6650 |    1.0000e-04 |
|       1 |         200 |       00:06:52 |         1.18 |         1.66 |       1.3934 |       2.7510 |    1.0000e-04 |
|       1 |         250 |       00:07:43 |         1.23 |         1.70 |       1.5236 |       2.9016 |    1.0000e-04 |
|       1 |         300 |       00:08:34 |         0.85 |         1.75 |       0.7246 |       3.0631 |    1.0000e-04 |
|       2 |         350 |       00:09:32 |         0.77 |         1.74 |       0.5905 |       3.0218 |    1.0000e-04 |
|       2 |         400 |       00:10:24 |         0.91 |         1.78 |       0.8247 |       3.1668 |    1.0000e-04 |
|       2 |         450 |       00:11:16 |         0.76 |         1.82 |       0.5761 |       3.3241 |    1.0000e-04 |
|       2 |         500 |       00:12:08 |         0.82 |         1.85 |       0.6793 |       3.4132 |    1.0000e-04 |
|       2 |         550 |       00:13:00 |         0.69 |         1.83 |       0.4710 |       3.3330 |    1.0000e-04 |
|       2 |         600 |       00:13:52 |         0.63 |         1.84 |       0.4021 |       3.4011 |    1.0000e-04 |
|       2 |         650 |       00:14:45 |         0.58 |         1.84 |       0.3359 |       3.3836 |    1.0000e-04 |
|       3 |         700 |       00:15:43 |         0.50 |         1.87 |       0.2517 |       3.4879 |    1.0000e-04 |
|       3 |         750 |       00:16:35 |         0.53 |         1.86 |       0.2848 |       3.4542 |    1.0000e-04 |
|       3 |         800 |       00:17:28 |         0.68 |         1.90 |       0.4650 |       3.6235 |    1.0000e-04 |
|       3 |         850 |       00:18:20 |         0.70 |         1.91 |       0.4915 |       3.6394 |    1.0000e-04 |
|       3 |         900 |       00:19:13 |         0.74 |         1.91 |       0.5436 |       3.6422 |    1.0000e-04 |
|       3 |         950 |       00:20:05 |         0.54 |         1.91 |       0.2953 |       3.6588 |    1.0000e-04 |
|       3 |        1000 |       00:20:58 |         0.60 |         1.91 |       0.3569 |       3.6569 |    1.0000e-04 |
|       4 |        1050 |       00:21:56 |         0.71 |         1.92 |       0.4975 |       3.6687 |    1.0000e-04 |
|       4 |        1100 |       00:22:49 |         0.42 |         1.95 |       0.1747 |       3.7994 |    1.0000e-04 |
|       4 |        1150 |       00:23:42 |         0.48 |         2.00 |       0.2283 |       4.0016 |    1.0000e-04 |
|       4 |        1200 |       00:24:34 |         0.61 |         1.99 |       0.3667 |       3.9468 |    1.0000e-04 |
|       4 |        1250 |       00:25:27 |         0.45 |         1.98 |       0.1988 |       3.9078 |    1.0000e-04 |
|       4 |        1300 |       00:26:20 |         0.51 |         2.00 |       0.2572 |       3.9827 |    1.0000e-04 |
|       4 |        1350 |       00:27:13 |         0.48 |         2.03 |       0.2315 |       4.1367 |    1.0000e-04 |
|       5 |        1400 |       00:28:11 |         0.36 |         2.00 |       0.1326 |       3.9996 |    1.0000e-04 |
|       5 |        1450 |       00:29:04 |         0.38 |         2.02 |       0.1418 |       4.0873 |    1.0000e-04 |
|       5 |        1500 |       00:29:57 |         0.40 |         2.05 |       0.1577 |       4.2143 |    1.0000e-04 |
|       5 |        1550 |       00:30:50 |         0.36 |         2.05 |       0.1270 |       4.2219 |    1.0000e-04 |
|       5 |        1600 |       00:31:44 |         0.43 |         2.03 |       0.1885 |       4.1248 |    1.0000e-04 |
|       5 |        1650 |       00:32:37 |         0.53 |         2.05 |       0.2777 |       4.2122 |    1.0000e-04 |
|       5 |        1700 |       00:33:30 |         0.33 |         2.08 |       0.1093 |       4.3399 |    1.0000e-04 |
|       6 |        1750 |       00:34:28 |         0.57 |         2.04 |       0.3248 |       4.1690 |    1.0000e-04 |
```

Figure 47. First section of training

```
|       6 |        1800 |       00:35:21 |         0.77 |         2.06 |       0.5956 |       4.2383 |       1.0000e-04 |
|       6 |        1850 |       00:36:16 |         0.49 |         2.05 |       0.2356 |       4.2026 |       1.0000e-04 |
|       6 |        1900 |       00:37:09 |         0.42 |         2.07 |       0.1759 |       4.2896 |       1.0000e-04 |
|       6 |        1950 |       00:38:03 |         0.42 |         2.08 |       0.1737 |       4.3080 |       1.0000e-04 |
|       6 |        2000 |       00:38:57 |         0.34 |         2.07 |       0.1166 |       4.2978 |       1.0000e-04 |
|       6 |        2050 |       00:39:51 |         0.43 |         2.14 |       0.1853 |       4.5723 |       1.0000e-04 |
|       7 |        2100 |       00:40:49 |         0.31 |         2.08 |       0.0982 |       4.3057 |       1.0000e-04 |
|       7 |        2150 |       00:41:42 |         0.38 |         2.10 |       0.1442 |       4.4025 |       1.0000e-04 |
|       7 |        2200 |       00:42:35 |         0.51 |         2.12 |       0.2565 |       4.4788 |       1.0000e-04 |
|       7 |        2250 |       00:43:28 |         0.48 |         2.13 |       0.2335 |       4.5266 |       1.0000e-04 |
|       7 |        2300 |       00:44:21 |         0.38 |         2.10 |       0.1478 |       4.4154 |       1.0000e-04 |
|       7 |        2350 |       00:45:14 |         0.35 |         2.13 |       0.1190 |       4.5464 |       1.0000e-04 |
|       7 |        2400 |       00:46:08 |         0.54 |         2.16 |       0.2899 |       4.6636 |       1.0000e-04 |
|       8 |        2450 |       00:47:06 |         0.45 |         2.12 |       0.2024 |       4.5117 |       1.0000e-04 |
|       8 |        2500 |       00:47:59 |         0.48 |         2.14 |       0.2340 |       4.5629 |       1.0000e-04 |
|       8 |        2550 |       00:48:52 |         0.30 |         2.14 |       0.0896 |       4.5647 |       1.0000e-04 |
|       8 |        2600 |       00:49:45 |         0.39 |         2.16 |       0.1492 |       4.6698 |       1.0000e-04 |
|       8 |        2650 |       00:50:38 |         0.42 |         2.14 |       0.1796 |       4.5915 |       1.0000e-04 |
|       8 |        2700 |       00:51:32 |         0.52 |         2.15 |       0.2655 |       4.6023 |       1.0000e-04 |
|       8 |        2750 |       00:52:26 |         0.57 |         2.15 |       0.3292 |       4.6129 |       1.0000e-04 |
|       8 |        2776 |       00:52:59 |         0.45 |         2.15 |       0.2065 |       4.6354 |       1.0000e-04 |
|=========|=============|================|==============|==============|==============|==============|==================|
Detector training complete.
*******************************************************************************
```

Figure 48. Second section of training

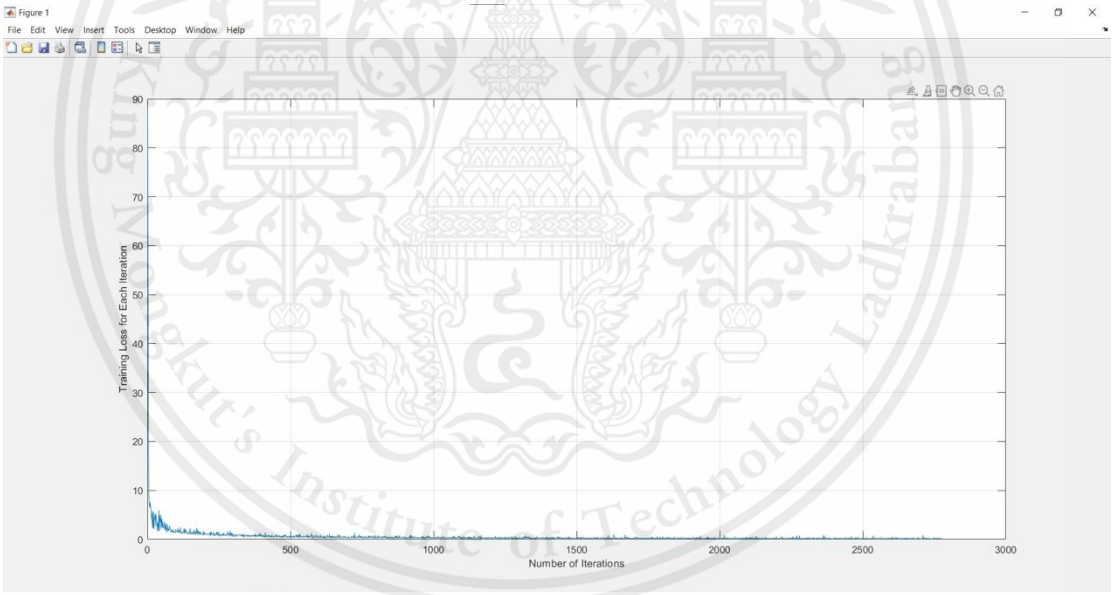

Figure 49. Training loss graph

The data that was separated prior to the training process, comprising of 10% of the original data, and used in testing is assessed and tested using the network that was previously preserved. The confusion matrix in Figure 50. depicts the results accurately in various groups of white blood cells as well as the percentages of error.

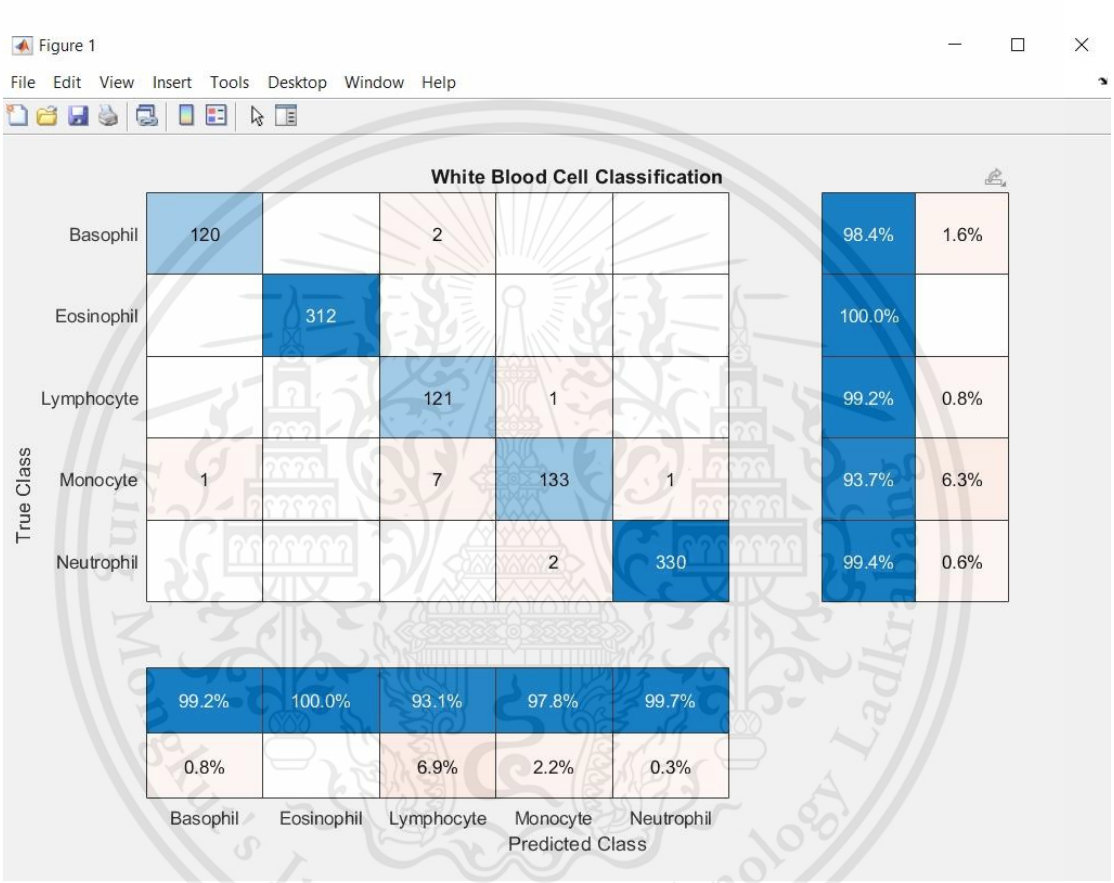

Figure 50. Confusion matrix of results of testing data.

The results of the tests indicate a strong likelihood of success in each class. Following Eosinophils, which had a 100% accuracy rate, were neutrophils, basophils, monocytes, and lymphocytes.

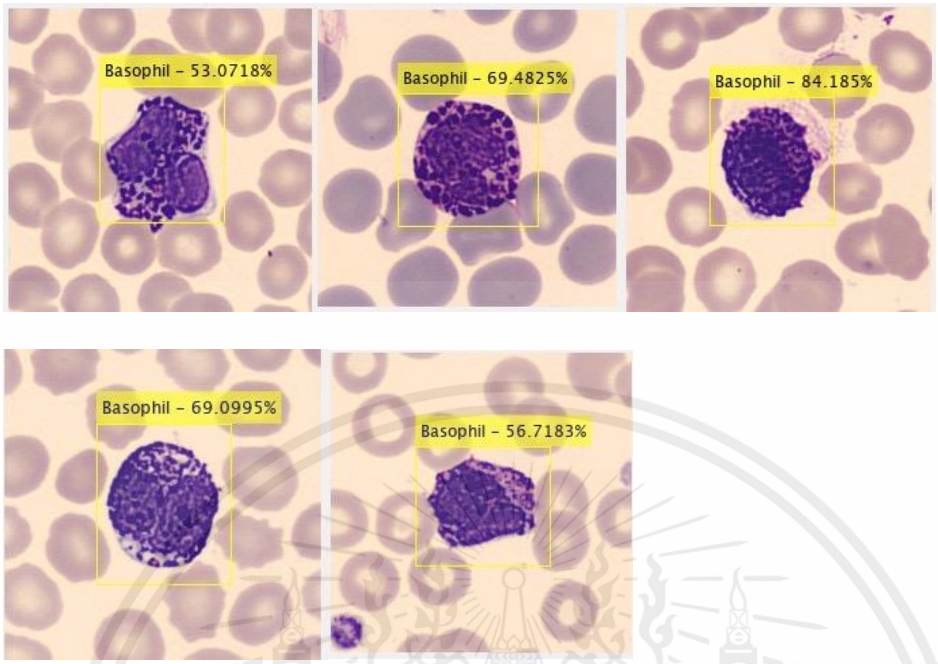

Figure 51. Basophil

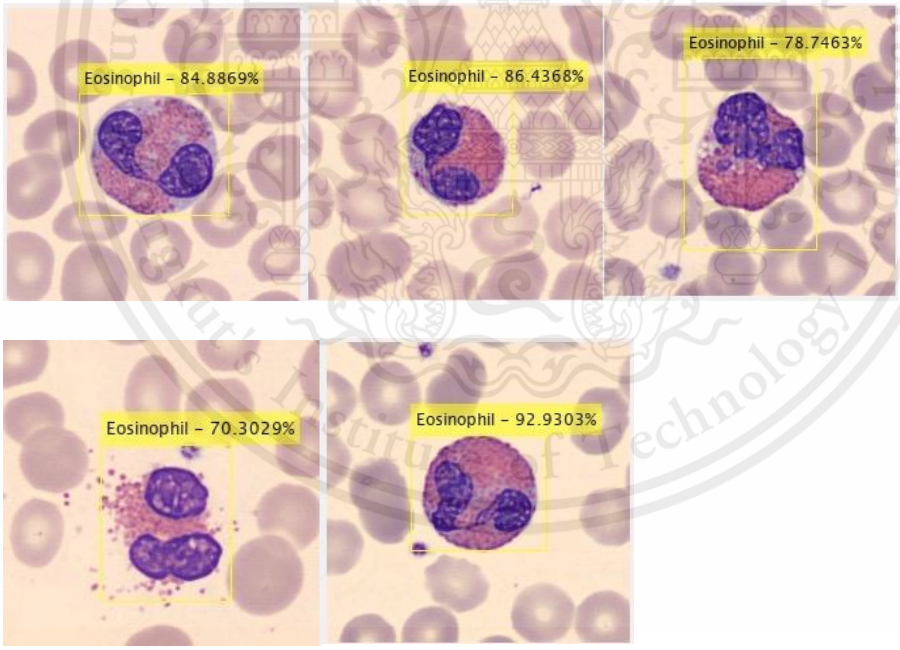

Figure 52. Eosinophil

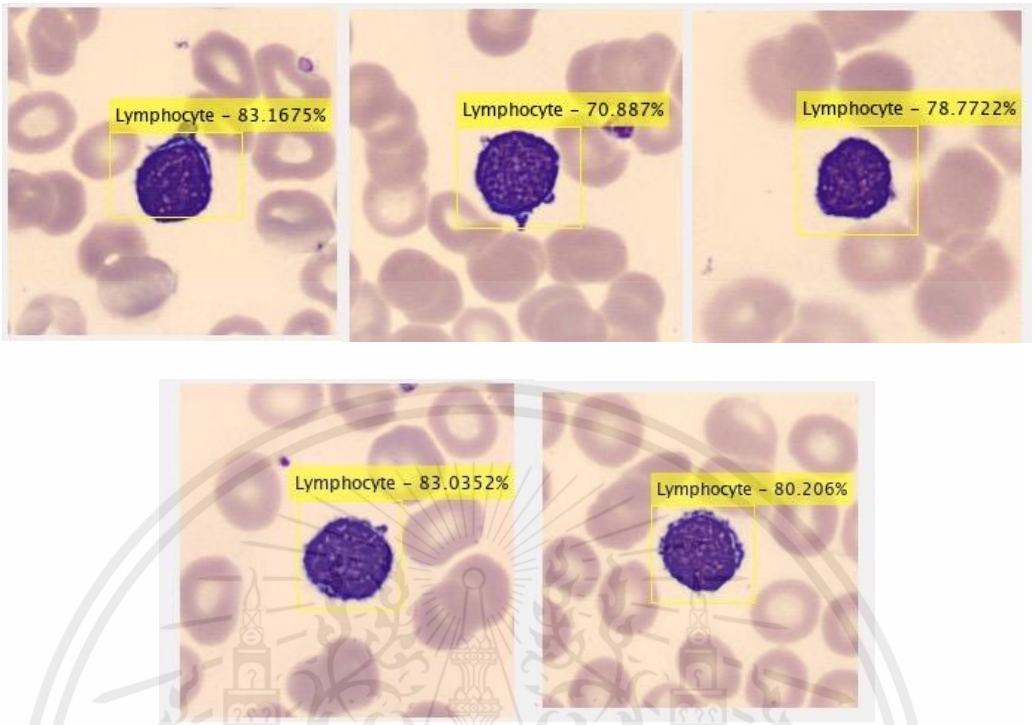

Figure 53. Lymphocyte

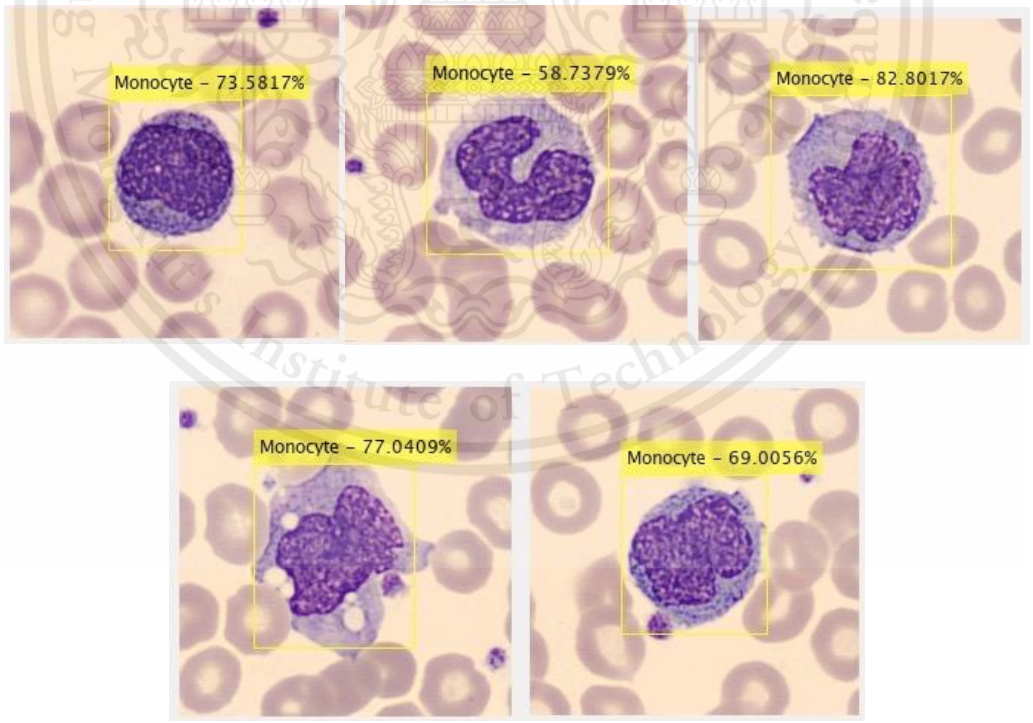

Figure 54. Monocyte

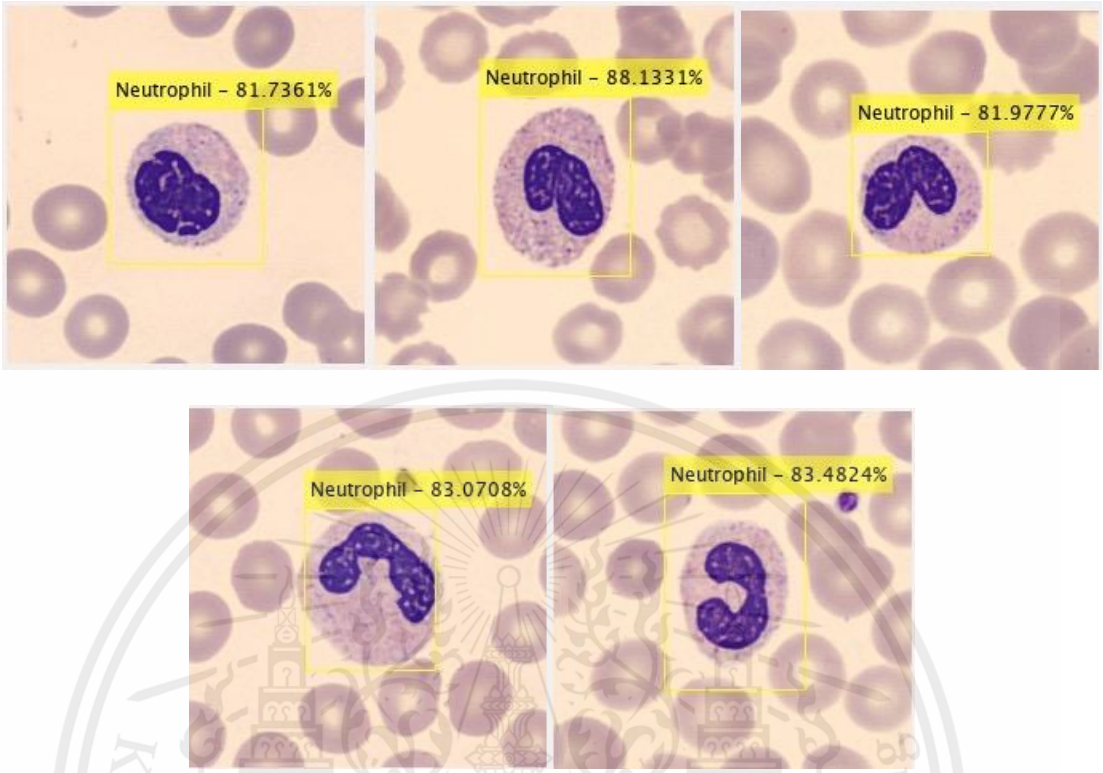

Figure 55. Monocyte

The Figure 51, 52, 53, 54, and 55 illustrate some of example from the YOLO's results in each class.

CHAPTER V

CONCLUSION

5.1 Discussion

The need to adapt medical practices to this new prefound technology is critical as the importance of deep learning using convolutional neural networks becomes more evident. Accuracy reduces the likelihood of human error, minimizes labor expenses, and helps to save time. The results of this trained network's data imply that this technology's potential may be increased by progressively adopting more sophisticated networks with larger quantities of qualifying data. Which will enable the medical industry to advance into an innovative future with fewer constraints and increased service effectiveness. Future studies might collect data and improve the effectiveness of the training process.

5.1 Conclusion

It is advised to continue using Resnet50 with YOLO incorporation since it has demonstrated great accuracy and precision in the classification, segmentation, and localisation of white blood cells.

REFERENCES

[1] Daniela Cadena-Herrera, Joshua E. Esparza-De Lara, Nancy D. Ramírez-Ibañez, Carlos A. López-Morales, Néstor O. Pérez, Luis F. Flores-Ortiz, Emilio Medina-Rivero,Validation of three viable-cell counting methods: Manual, semi-automated, and automated, Biotechnology Reports,Volume 7,2015, Pages 9-16, ISSN 2215-017X, https://doi.org/10.1016/j.btre.2015.04.004.

[2] Giuseppe Lippi, Clarissa Cattabiani, Anna Benegiamo, Daniela Gennari, Fernanda Pavesi, Alberta Caleffi, Silvia Pipitone, Evaluation of white blood cell count in peritoneal fluid with five different hemocytometers, Clinical Biochemistry, Volume 46, Issues 1–2, 2013, Pages 173-176, ISSN 0009-9120, https://doi.org/10.1016/j.clinbiochem.2012.10.034.

[3] McKinnon K. M. (2018). Flow Cytometry: An Overview. Current protocols in immunology, 120, 5.1.1–5.1.11. https://doi.org/10.1002/cpim.40

[4] Michael Rahul Soosai, Y. Camy Joshya, R. Shyam Kumar, I. Ganesh Moorthy, S. Karthikumar, Nguyen Thuy Lan Chi, Arivalagan Pugazhendhi, Versatile image processing technique for fuel science: A review, Science of The Total Environment, Volume 780, 2021, 146469, ISSN 0048-9697, https://doi.org/10.1016/j.scitotenv.2021.146469.

[5] R.J. Pally, S. Samadi, Application of image processing and convolutional neural networks for flood image classification and semantic segmentation, Environmental Modelling & Software, Volume 148, 2022, 105285, ISSN 1364-8152, https://doi.org/10.1016/j.envsoft.2021.105285.

[6] 6. Erchan Aptoula, Sébastien Lefèvre, Chapter 1 - Morphological Texture Description of Grey-Scale and Color Images, Editor(s): Peter W. Hawkes, Advances in Imaging and Electron

Physics, Elsevier, Volume 169, 2011, Pages 1-74, ISSN 1076-5670, ISBN 9780123859815, https://doi.org/10.1016/B978-0-12-385981-5.00001-X.

[7] Dan López-Puigdollers, V. Javier Traver, Filiberto Pla, Recognizing white blood cells with local image descriptors, Expert Systems with Applications, Volume 115, 2019, Pages 695-708, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2018.08.029.

[8] Blood smear analyzer for white blood cell counting: A hybrid microscopic image analyzing technique PramitGhoshaDebotoshBhattacharjeebMitaNasipurib

[9] Shijie Liu, Chapter 11 - How Cells Grow, Editor(s): Shijie Liu, Bioprocess Engineering (Second Edition), Elsevier,2017, Pages 629-697, ISBN 9780444637833, https://doi.org/10.1016/B978-0-444-63783-3.00011-3.

[10] Talavera-Martínez, L., Bibiloni, P., Giacaman, A., Taberner, R., Hernando, L. J. D. P., & González-Hidalgo, M. (2022). A novel approach for skin lesion symmetry classification with a deep learning model. Computers in Biology and Medicine, 145, 105450.

[11] Kaiming He., Xiangyu Zhang., Shaoqing Ren., Jian Sun., (2015) Deep Residual Learning for Image Recognition. Microsoft Research. arXiv:1512.03385v1 [cs.C V]

[12] Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. Neurocomputing, 187, 27–48. https://doi.org/10.1016/j.neucom.2015.09.116

[13] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. In Insights into Imaging (Vol. 9, Issue 4, pp. 611–629). Springer Verlag. https://doi.org/10.1007/s13244-018-0639-9

[14] Bayat, O., Aljawarneh, S., Carlak, H. F., International Association of Researchers, Institute of Electrical and Electronics Engineers, & Akdeniz Üniversitesi. (n.d.). Proceedings of 2017 International Conference on Engineering & Technology (ICET'2017) : Akdeniz University, Antalya, Turkey, 21-23 August, 2017.

[15] AL-Huseiny, M. S., & Sajit, A. S. (2021). Transfer learning with GoogLeNet for detection of lung cancer. Indonesian Journal of Electrical Engineering and Computer Science, 22(2), 1078.

[16] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. http://arxiv.org/abs/1512.03385

[17] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. http://arxiv.org/abs/1409.1556

[18] Hammad, I., & El-Sankary, K. (2018). Impact of approximate multipliers on VGG deep learning network. IEEE Access, 6, 60438–60444. https://doi.org/10.1109/ACCESS.2018.2875376

[19] Kathirgamaraja Pradeep, Kamalakkannan Kamalavasan, Ratnasegar Natheesan, & Ajith Pasqual. (2018). EdgeNet: SqueezeNet like Convolution Neural Network on Embedded FPGA.

[20] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. In Insights into Imaging (Vol. 9, Issue 4, pp. 611–629). Springer Verlag. https://doi.org/10.1007/s13244-018-0639-9

[21] S. Mascarenhas and M. Agarwal, "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification," 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), 2021, pp. 96-99, doi: 10.1109/CENTCON52345.2021.9687944.

[22] Kim, Y. G., Kim, S., Cho, C. E., Song, I. H., Lee, H. J., Ahn, S., Park, S. Y., Gong, G., & Kim, N. (2020). Effectiveness of transfer learning for enhancing tumor classification with a convolutional neural network on frozen sections. Scientific Reports, 10(1). https://doi.org/10.1038/s41598-020-78129-0

[23] Huang, T. S., Schreiber, W. F., & Tretiak, O. J. (1971). Image Processing. In PROCEEDINGS OF THE IEFE (Vol. 59, Issue 11).

[24] Haque, F., Lim, H.-Y., & Kang, D.-S. (n.d.). Object Detection Based on VGG with ResNet Network.

[25] Bayat, O., Aljawarneh, S., Carlak, H. F., International Association of Researchers, Institute of Electrical and Electronics Engineers, & Akdeniz Üniversitesi. (n.d.). Proceedings of 2017 International Conference on Engineering & Technology (ICET'2017) : Akdeniz University, Antalya, Turkey, 21-23 August, 2017.

[26] Ghosh, P., Bhattacharjee, D., & Nasipuri, M. (2016). Blood smear analyzer for white blood cell counting: A hybrid microscopic image analyzing technique. Applied Soft Computing Journal, 46, 629–638. https://doi.org/10.1016/j.asoc.2015.12.038

[27] Cadena-Herrera, D., Esparza-De Lara, J. E., Ramírez-Ibañez, N. D., López-Morales, C. A., Pérez, N. O., Flores-Ortiz, L. F., & Medina-Rivero, E. (2015). Validation of three viable-cell counting methods: Manual, semi-automated, and automated. Biotechnology Reports, 7, 9–16. https://doi.org/10.1016/j.btre.2015.04.004