CLASSIFICATION AND DETECTION OF PLASMODIUM FALCIPARUM MALARIA LIFE CYCLE STAGES FROM BLOOD SMEAR MICROSCOPIC IMAGES

BY

63011370 TUNCHANOK NILLAPAT

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 2023

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Classification and Detection of Plasmodium Falciparum Malaria Life Cycle Stages from Blood Smear Microscopic Images.
Student Name	Miss Tunchanok Nillapat Student ID. 63011370
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Signed: _____ (Assist. Prof. Dr. Veerayuth Kittichai)
Project Co-Advisor	Signed: _____ (Assoc. Prof. Dr. Chuchart Pintavirooj)
Project Co-Advisor	Signed: Ukkrit Jansri (Aj. Ukkrit Jansri)
Committee	Signed: _____ (Assoc. Prof. Dr. Chuchart Pintavirooj)

Committee Signed:_____

(Asst. Prof. Dr. Treesukon Treebupachatsakul)

Committee Signed:_____

(Asst. Prof. Dr. Kasama Srirussamee)

Head of Department Signed:_____

(Assoc. Prof. Dr. Sarinporn Vitsitsattapongse)

Project Title	Classification and Detection of Plasmodium Falciparum Malaria Life Cycle Stages from Blood Smear Microscopic Images.
Student Name	Miss Tunchanok Nillapat Student ID. 63011370
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Assist. Prof. Dr. Veerayuth Kittichai
Project Co-Advisor	Assist. Prof. Dr. Chuchart Pintavirooj Aj. Ukkrit Jansri
Academic Years	2023

ABSTRACT

Malaria is one of the zoonotic diseases that cause high morbidity and mortality in many developing countries located in tropical areas worldwide. Typically, in rural areas near the mountains or the forests where plenty of female Anopheles mosquitoes that carry malaria parasites are found. Additionally, among 5 different species of malaria, Plasmodium falciparum is the most often found in Thailand. To reduce the spread of disease, early detection of infected patients leads to proper medication and prevention for transmitting the disease to surrounding people. A blood examination is a gold standard to identify whether a patient is infected through microscopic observation on a blood smear slide. Moreover, this method can be used to identify 4 different forms of malaria parasites including ring, trophozoite, schizont, and gametocyte forms as well as their concentration of each form which can refer to the symptom's severity. Therefore,

this project aims to apply the deep learning network, 2 types of the ResNet network including the ResNet-50 and the ResNet-101 to create models for malaria screening and stages classification as well as the use of YOLO version 3 and 4 to detect different stages in the blood smear images. When YOLO version 4 requires some modification to improve detection capability.

The results show that the use of ResNet-50 on the diagnostic model gives a high accuracy of 95.5% While using the same network to classify 4 stages of malaria parasite only gives 79.61% validation accuracy, however, the result of using ResNet-101 to do the same task shows slightly lower validation accuracy, 76.38%. The outcome of 3 detection models displays undesirable results, all of them can not precisely detect all infected cells from an image. However, they tend to be more accurate in detecting ring form above other forms since a high proportion of ring form in the training dataset leads to a higher ability to detect ring form.

ACKNOWLEDGEMENTS

I would like to sincerely thank Assist. Prof. Dr. Chuchart Pintaviroo from the School of Engineering, and Assist. Prof. Dr. Veerayuth Kittichai and Aj. Ukkrit Jansri from the faculty of medicine, KMITL for all of their guidance and encouragement throughout this project. Without their contributions, this research would not have been done completely.

LIST OF TABLES

Table		Page
1	Diagnostic Classification Summary Results	67
2	Malaria stages classification using ResNet-50 network summary	70
3	Malaria stages classification using ResNet-101 network summary	73
4	Detected Results of YOLO v3 model	77
5	Detected Results of Modify YOLO v4 model	80

LIST OF FIGURES

Figures		Page
1	Mortality rate from malaria statistic (1)	1
2	The changes in cytosine composition of red blood cells after being infected by malaria parasites.	5
3	Overview of Malaria's Life Cycle	9
4	Overview of Artificial Intelligence, Machine, and Deep Learning	10
5	Machine learning workflow	12
6	Deep learning workflow	13
7	Forward Propagation or Feedforward	14
8	Backpropagation	15
9	Architecture of Convolutional Neural Network (CNN)	16
10	3x3 Convolution, the left side is the input of the convolution layer, and the right side is the convolution filter/kernel	17
11	Performing convolution to get a Feature Map	18

12	Completed Convolution in 2D using 3x3 Filter	18
13	Stride during convolution	19
14	Stride is 1	19
15	Stride is 2	20
16	Padding with the same input image results in a feature map with the same size as the input	21
17	Max Pooling and Average Pooling	22
18	ResNet-50 Architecture	24
19	Object Detection and Classification	26
20	Classification Diagram 1	28
21	Classification Diagram 2	29
22	Object Detection Diagram	30
23	Plasmodium falciparum Dataset1 (single-cell images)	33
24	Plasmodium falciparum Dataset2 (Blood smear field)	34

25	P. falciparum in the ring, early trophozoite, mature trophozoite, schizont, and gametocytes forms.	35
26	Separate dataset images into 2 folders; Parasitized and Uninfected	36
27	Separate parasitized dataset into 4 folders according to their morphologies	37
28	The ground truth code	37
29	The “Define New ROI Label” tab	39
30	Image Labeller in MATLAB, labels all parasite cells according to their stages as well as a few examples of normal red blood cells.	40
31	Label Summary tab	40
32	Command to save gTruth parameter as malariaDatasetGroundTruth.mat	41
33	Layer Graph imported from the workspace	42
34	Auto Classification Layer as the last layer	43
35	Two Outputs in the Fully Connected Layer	43
36	Analyze ResNet-50 Network	44

37	Training and Validation Options for Import Image Data	45
38	Import Image Data information	46
39	Set Training options	47
40	Training Progress	47
41	Exported parameters in the workspace and Training network MATLAB files	48
42	Test Network by classifying the test dataset	49
43	Edit Output size for fullyConnectedLayer to 4 classes for both models	50
44	Analyze Modified ResNet-50 Network	51
45	Analyze Modified ResNet-101 Network	51
46	Import Image Dataset for both models	52
47	Training Progress for ResNet-101 Model	53
48	Save Traning Networks	54
49	Modify Lines 3 and 4	54

50	Template for every function	55
51	Display dataset information in 4 rows	57
52	Display the sample image undergoing augmentation	58
53	Display preprocess training data with their bounding boxes	59
54	Learning Rate and Total Loss during Training	60
55	Detector Network Properties	63
56	YOLO v4 Object Detector Properties	63
57	Analyze Detector Network	64
58	Detection command	65
59	Test Diagnostic Classification Model	68
60	Training Results of ResNet-50 Model	69
61	Training Results of ResNet-101 Model	70
62	Malaria Stages Classification with ResNet-50 Model	72

63 Malaria Stages Classification with ResNet-101 Model 75

64 A Detected Image by YOLO v3 Model 76

65 A Detected Image by YOLO v4 Model 79

66 A Detected Image by Modified YOLO v4 Model 80

LIST OF SYMBOLS/ABBREVIATIONS

Symbol/Abbreviation	Terms
ANN	Artificial Neural Network
AI	Artificial Intelligence
CNN	Convolutional Neural Network
R	Ring form of malaria parasite
T	Trophozoite form of malaria parasite
S	Schizont form of malaria parasite
RBC	Healthy Red Blood Cell

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGEMENTS	v
LIST OF TABLES	vi
LIST OF FIGURES	vii
LIST OF SYMBOLS/ABBREVIATIONS	xiii
CHAPTER 1	1
1.1 Clinical Problems	1
1.2 Objectives of the Study	2
1.3 Scope of the Study	3
1.4 Report Outline	3
CHAPTER 2	4
2.1 Introduction of blood	4
2.2 Pathology of Malaria	7
2.3 Deep Learning	10
2.4 Convolutional Neural Network (CNN)	15
	xiv

2.4.1 Convolution: Filters/Kernels 17

2.4.2 Stride 19

2.4.3 Padding 21

2.4.4 Pooling 22

2.4.5 ResNet-50 23

2.5 YOLO (You only look once) 25

2.6 Literature review 26

CHAPTER 3 27

3.1 Introduction 28

3.2 Install required materials in MATLAB 30

3.3 Input Dataset Preparation and Labeling Process 32

3.3.1 Malaria Dataset 32

3.3.2 Dataset Organizing and Labeling 34

3.3.2.1 Criteria to Identify the Stages of Malaria Parasite 34

3.3.2.2 Manually Saparate Dataset Folders according to their Classes 36

3.3.2.3 Image Labeller in MATLAB 37

3.4 Methodologies for Classification models 41

3.4.1 Diagnostic Classification using ResNet50 41

3.4.1.1 Prepair ResNet50 Model 41

3.4.1.2 Import Dataset 44

3.4.1.3 Training 46

3.4.1.4 Testing 48

3.4.2 Malaria Stages Classification using ResNet-50 and ResNet-101 49

3.4.2.1 Preparing CNNs Models 49

3.4.2.2 Import Datasets 52

3.4.2.3 Training 53

3.4.2.4 Testing 54

3.5 Methodologies for Object Detection models 54

3.5.1 YOLOV3 55

3.5.1.1 Required Supporting Functions 55

3.5.1.1.1 Model Gradient Function 56

3.5.1.1.2 augmentData Function 56

3.5.1.1.3 preprocessData Function 56

3.5.1.2 Develop YOLOv3 model 56

3.5.2 YOLOV4 60

3.5.2.1 Required Supporting Functions 60

3.5.2.2 Develop YOLOv4 model 61

3.5.3 Modify YOLOV4 Detector 62

3.5.4 Detect object from a microscopic image 64

CHAPTER 4 66

4.1 Introduction 66

4.2 Results of Classification models 66

4.2.1 Results of Diagnostic Classification using ResNet50 66

4.2.2 Results of Malaria Stages Classification using ResNet-50 and ResNet-101 69

4.3 Results of Object Detection models 76

4.3.1 Results of YOLO v3 model 76

4.3.2 Results of YOLO v4 models 78

4.3.3 Results of Modified YOLO v4 models 79

CHAPTER 5 83

5.1 Discussion 83

5.2 Conclusion 84

REFERENCES 86

APPENDICES 90

APPENDIX A 91

APPENDIX B 92

APPENDIX C 99

APPENDIX D 103

APPENDIX E 110

CHAPTER 1

INTRODUCTION

1.1 Clinical Problems

Malaria is one of the global zoonotic diseases that remains a significant cause of morbidity and mortality in many developing countries, affecting approximately 40% of the world's population as shown in Figure 1. According to the statistic obtained from The World Health Organization (WHO), the world malaria report during 2019-2021 states that more than 90% of malaria-related mortality worldwide is attributed to plasmodium falciparum infection from the bite of an infected Anopheles mosquito, which continues to pose a severe threat to public health on a global scale. The blood examination is the gold standard for diagnosing malaria-infected patients. This is usually done by manual microscopic examinations of patients' blood smear slides along with the concentration of parasitized stages to affirm severity. However, this method has some difficulty in determining the exact stage of the malaria parasite correctly and there has some limitations for inspecting a large number of slides due to their accuracy, consistency, and diagnosis speed depending on microscopists' diagnostic and technical skills, especially in rural areas in most developing countries that hardly found one (1).

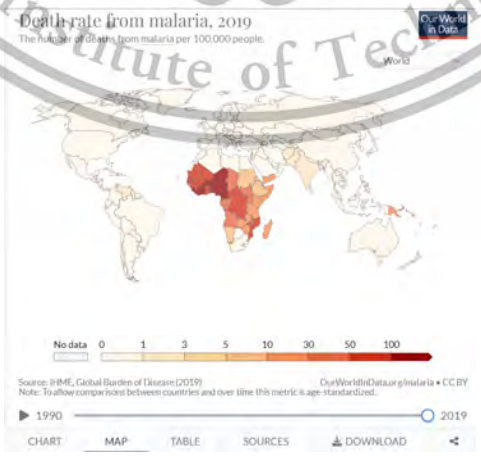

Figure 1: Mortality rate from malaria statistic (1)

Therefore, rapid diagnosis is required to alleviate this issue by applying classification and object detection models for screening asymptomatic patients and identifying the malaria parasite stages which usually benefit confirming the disease's severity from microscopic videos of thin blood smear slides. To do so, the use of deep learning networks such as ResNet-50 will be applied to classify what kind of blood smear slide images refer to the patient who is already infected by Plasmodium falciparum malaria parasites as well as the application of all of the state-of-the-art object detection models, YOLOV3, and YOLOV4 that are available in MATLAB will be applied to compare accuracy and speed in detecting objects when versions 4 of YOLO need some modification to improve their ability to detect microscopic objects without significantly slowing down detection speed. Moreover, they will be used to classify and detect 4 different types of red blood cells which are healthy cells, and 3 stages of malaria-infected cells; ring form, trophozoite form, and schizont form.

1.2 Objectives of the Study

1.2.1 To look for applying deep learning to facilitate the diagnosis process, informing whether a patient is infected.

1.2.2 To instantly detect infected cells with extremely high accuracy.

1.2.3 To compare networks and models' accuracies to find the most suitable model for malaria-parasitized cell detection.

1.3 Scope of the Study

1.3.1 The datasets were obtained from only 2 sources that will be used to train models in this project; the first dataset was received from the NIH website and the second dataset was received from the professor.

1.3.2 Develop models for malaria screening and infected cell detection using 2 types of pretrained convolutional neural networks including the ResNet-50 and ResNet-101 networks.

1.3.3 Utilize the MATLAB application to create all of the models.

1.4 Report Outline

Chapter 1 Introduction consists of clinical problems, the objective of the study, and the scope of the study.

Chapter 2 Review of theories and principles consists of theories involved in this research including components of blood, pathology of malaria, information on deep learning, convolutional neural network, YOLO, and literature review.

Chapter 3 Research methodology states our method for each type of model.

Chapter 4 Experiment results demonstration.

Chapter 5 Discussion and Conclusion.

CHAPTER 2

REVIEW OF THEORIES AND PRINCIPLES

2.1 Introduction of blood

Blood is a specialized body fluid that is responsible for carrying out oxygen, carbon dioxide, nutrients, and waste products throughout the circulatory system in the human body. Blood plays a crucial role in maintaining the health and well-being of the body, and any abnormalities in its composition or function can have serious consequences for health. The amount of blood that circulates in each person's body is determined by their weight and height. Normally, the blood volume made up 7 to 8 percent of the entire body weight (2). The composition of blood can be divided into three major components including red blood cells or erythrocytes, white blood cells or lymphocytes, and platelets or thrombocytes. Apart from these cells, the blood also composes a liquid component called plasma to circulate other cells and tissues in the blood, plasma accounted for 55 percent of total blood volume, while erythrocytes account for almost 45 percent (3). Each type of blood cell has its own specific function. Plasma is a yellowish liquid that is made up of water, sugar, fat, protein, and salts. The prominent role of plasma is to circulate the blood cells along vessels as well as other components that transport through blood circulation such as antibodies, nutrients, protein, and some hormones (2).The levels of these substances can be affected by various factors, including diet, exercise, and illness.

Red blood cells are responsible for transporting oxygen from the lungs to other organs and tissue throughout the entire body. Red blood cells have a biconcave disk shape with a flattened center. The differentiation of red blood cells is under the regulation of a specific hormone produced by the kidneys called erythropoietin. The primary location of red blood cell

formation is in the bone marrow and after approximately 7 days, these immature red blood cells become mature red blood cells and release into the circulatory system and ready to perform their function. The deformation in shape and changes in the volume of red blood cells refers to the impairment of both physiological and cellular functions (4). This usually occurs when the red blood cells are infected by various pathogens or during normal red blood cells undergo a cell aging process, these lead to abnormal modification of cell membrane or shifting in cytosol composition (5). This would further affect the ability in transporting oxygen to the cells. For instance, sickle cells, which resemble crescent moons or sickles, develop when red blood cells harden and become sticky (6). These strangely shaped cells have the potential to become trapped in tiny blood vessels, slowing or preventing the passage of blood and oxygen to other areas of the body and causing anemia in the patient. Additionally, a patient who gets infected by the malaria parasite also shows some changes in the cell membrane and the cytosol composition of red blood cells as shown in Figure 2. After that the parasite multiplies inside the cells then bursting and further invading the nearby healthy red blood cells.

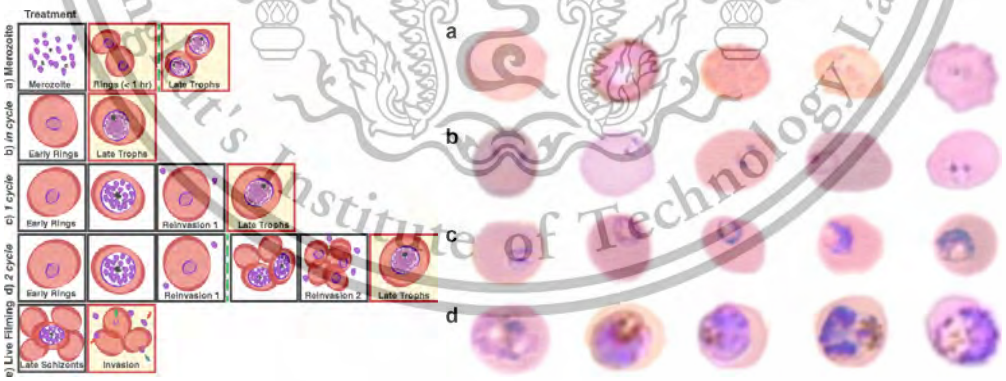

Figure 2: The changes in cytosine composition of red blood cells after being infected by malaria parasites.

In the immune system, white blood cells are essential for fighting off infections and other diseases. They exist in very small amounts when compared with other components in the blood, accounting for approximately 1 percent of total blood volume (2). They are relatively larger when compared to red blood cells. The white blood is a granulocyte which refers to the immune cell that is composed of granules or small particles inside the cell (7). It can be classified into 2 major types based on their granular components; granulocytes or polymorphonuclear cells which contain multiple nuclei in a single cell, and agranulocytes or mononuclear cells which contain only one nucleus for each cell (8). The granulocytes are divided into 3 types according to their granular appearance, staining of cytochemical dyes on their nucleus, and their functions, this includes neutrophils, eosinophils, and basophils. Neutrophil is the most prevalent type of white blood cell, with approximately 55% to 70%. It has multilobed nuclei and is responsible for getting rid of the pathogen and foreign particles by phagocytosis and is involved in the inflammation process that our immune system will produce roughly 10 times more than usual during the inflammation. Eosinophil has bilobed nuclei and functions in destroying larger particles by working with antibodies. This eosinophil is stained red or pink. Basophil has bi-tri lobed nuclei that stain in blue to purple. This cell has similar functions as the mast cells, basophil produces substances related to the immune system such as histamine that trigger allergic symptoms (8).

While agranulocytes can be divided into 2 types based on similar criteria to the classification of granulocytes, this includes lymphocytes and monocytes (8). The lymphocyte has single large and spherical nuclei and is the smallest size when compared to other types of white blood cells. Lymphocytes are also divided into 2 subgroups according to their cluster differentiation markers; T lymphocytes and B lymphocytes. T lymphocytes coordinate with other

immune cells to control how they carry out their duties and directly eliminate tumors and infected cells whereas the B lymphocytes produce antibodies which is the specific protein targeted specific pathogen (2). Monocytes are the largest white blood cells, they have kidney-shaped nuclei. They eliminate foreign materials by phagocytosis and function as a precursor for many mononuclear phagocytic cells such as macrophages, osteoclasts which are the bone resorption cells, and microbial cells in tissues and organs (8).

2.2 Pathology of Malaria

Malaria is a zoonotic disease transmitted through humans by the bite of female Anopheles mosquitoes, these mosquitoes carry a malaria parasite called Plasmodium into the human circulatory system before invading the liver where the parasite multiplies (1). This disease is usually found in rural areas close to the forest in many tropical regions including Africa, Asia, and Central and South America (10). This Plasmodium parasite causes fever and flu-like illnesses such as headaches, shaking chills, tiredness, and muscle aches. In some cases, they also show additional symptoms including nausea, vomiting, and diarrhea but these symptoms are not common (9). However, this disease is preventable and curable. Since malaria parasites target erythrocytes or red blood cells that function to transport oxygen throughout the body, infecting multiple erythrocytes leads to a decrease in the number of healthy erythrocytes in transport oxygen, this may cause anemia and jaundice (9). If the patient never receives prompt and appropriate treatment, the condition becomes more severe and results in more complications such as kidney failure, seizures, fatigue, confusion, difficulty breathing, and coma until death (9).

According to the World Health Organization (WHO), the latest malaria report in 2021 reported that approximately 247 million people got infected by malaria and approximately 619000 deaths by this disease (11). People with weak immune systems including infants, children aged under 5 years old, pregnant women, and people with HIV or AIDS have a higher risk of getting severe symptoms. This can be seen from the statistics reported by WHO during the COVID-19 pandemic, about 80 percent of the entire mortality rate in Africa come from children under 5 years old (11).

Among 5 malaria parasite species including *P. falciparum, P. vivax, P. malaria, P. ovale, and P. knowlesi,* there are only 2 species that are prevalent in Thailand; P. falciparum and P. vivax (13). As the population of Thai people is approximately 70 million, people at risk of this disease are estimated to be 13 million since they live in rural areas close to or in the middle of forests and mountains with plenty of mosquitoes (13). Normally, suppose a patient got bitten by a primarily infected mosquito. In that case, its saliva together with sporozoites will invade the human's bloodstream and travel to liver cells where the parasite begins a new life cycle. After they already differentiate into schizont forms, these cells will rupture and infect new erythrocytes where they replicate themselves. Then it progresses into 3 different stages starting from the ring, trophozoite, and schizont forms repeatedly until the secondary mosquito bite this patient and carries infected cells or gametocytes to the next patient (14), these life cycle stages of malaria are shown in Figure 3. These different distinct forms of the parasite in the circulatory system are used as the target during the diagnostic process to confirm whether a patient is infected through either microscopy or a rapid diagnostic test using a blood sample (11). As the schizont form refers to the late stage of the malaria parasite when the cell rupture and release cytoplasm content including parasite to invade nearby cells, this means a high proportion of schizont cells indicates

that the patient will have more severe symptoms (15). Therefore, the diagnosis should be done based on observation of the changes in blood morphology and the concentration of schizont forms.

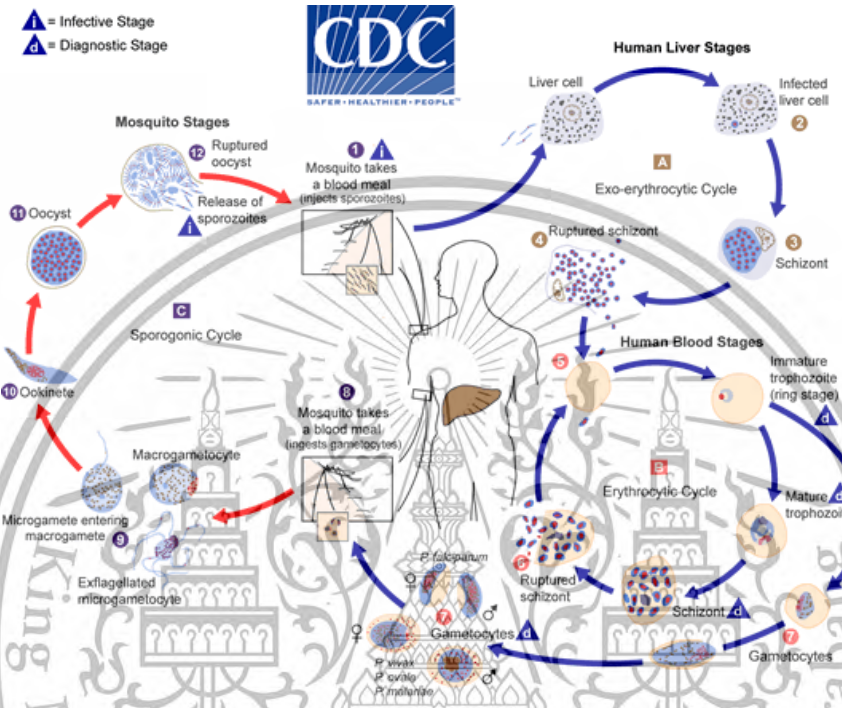

Figure 3: Overview of Malaria's Life Cycle

Moreover, the quantity of parasitemia has been considered the major risk factor that causes severe symptoms (16). There are 2 methods for calculating parasitemia depending on the types of blood slides. If examining the thin blood smear slide, we can estimate the percentage of parasitemia by expressing the number of infected cells as a percentage of total red blood cells (17). On the other hand, if examing the thick blood smear slide, parasitemia can be calculated by counting the asexual parasites present in a specific number of white blood cells and express as a percentage (18).

The information about parasite species, the proportion of each stage, and parasitemia will contribute to determining the most suitable medications for each patient. Normally, the first line

therapy for uncomplicated P. falciparum is the combination of artemisinin derivatives (19) since it has resistance to chloroquine usually used as the therapy for non-falciparum malaria in many countries (20). However, there are a few non-falciparum that are resistant to chloroquine such as chloroquine-resistant P. vivax which is very common, this can be treated the same as for P. falciparum malaria therapy (20). Severe falciparum malaria can be treated by parental quinine or quinidine until the patient can take oral medications (20). However, intravenous artesunate is considered the first-line therapy for all types of severe malaria (19).

2.3 Deep Learning

It is a subset of artificial intelligence and machine learning that involves the use of artificial neural networks (ANNs) which attempt to simulate the behavior of the human brain in order to learn and make predictions or decisions from a large amount of data (22). A deep learning network is a type of ANN that consists of multiple layers of interconnected nodes, which are designed to process and extract increasingly abstract and complex features from the input data (22). The relationships between artificial intelligence, machine learning, and deep learning are summarized in Figure 4.

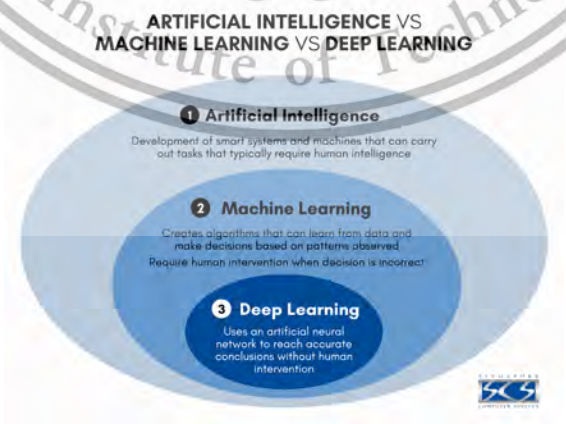

Figure 4: Overview of Artificial Intelligence, Machine, and Deep Learning (22)

Artificial Intelligence (AI) refers to the field of computer science and technology that is concerned with building intelligent computers that are capable of carrying out tasks that would ordinarily need human intelligence. It involves the development of computer systems or models that can analyze, evaluate and interpret data, learn from experience, make decisions, and solve challenging problems. They are made to mimic several cognitive processes as perception, logic, learning, problem-solving, and judgment. These systems can be trained to identify patterns from large datasets, comprehend natural language, communicate with people orally or in writing, and even display a certain level of creativity.

When Machine Learning is a subfield of artificial intelligence (AI) that deals with developing statistical models and systems that allow computers to automatically learn from their mistakes and improve over time without explicit programming. In other words, machine learning allows computers to analyze data, identify patterns, make predictions, and make decisions based on examples and past observations. Rather than following explicit instructions through coding, machine learning models learn from data and iteratively refine their performance over time. They use statistical approaches to determine patterns, extract insightful information, and develop mathematical models that can generalize from the existing data to make predictions or conduct actions on new, unseen data (23). However, if the decision is made incorrectly or undesirably, human intervention is required to solve the issue. To begin machine learning, relevant features are manually extracted from images. After that, a model is developed using these features to classify the objects in the image and present an output as the object's class (25). This workflow of machine learning is shown in Figure 5.

Figure 5: Machine learning workflow

Deep Learning is a further subset of machine learning. It makes use of an artificial neural network to process data using many layers of models and arrive at a precise judgement without the need for human intervention. Typically, deep learning models require 3 or more layers in an artificial neural network. Generally, artificial neural networks are composed of 5 main components including data inputs, weights, biases, activation functions, and outputs. Data inputs are the dataset aimed to process. Weights are used to determine the importance of each output on the outcome. Biases represent the number of assumptions that are made, while higher bias refers to more assumptions being made whereas lower bias means fewer assumptions are made. Activation functions are a combination of the weighted sum of the inputs and the bias applied. They determine whether the data will be transmitted to the next layer of the network. Lastly, outputs refer to outcome decisions made by the deep learning program. These 5 components are grouped together into 3 different layers which are the input layer where the deep learning model ingests the data for processing, hidden layers (weights, biases, activation functions), and the output layer where the final prediction or classification is made (24). Relevant features are automatically retrieved from images using a deep learning approach. Additionally, deep learning accomplishes "end-to-end learning" in which a network is given unprocessed data and a task to complete, such as classification, and it automatically learns how to do it (25) without the need for

manual feature engineering. This advantage makes them particularly useful in domains where the input data is complex and high-dimensional, such as images or speech. However, training deep learning networks can be computationally intensive and requires large amounts of labeled data, making them more challenging to apply in some settings.

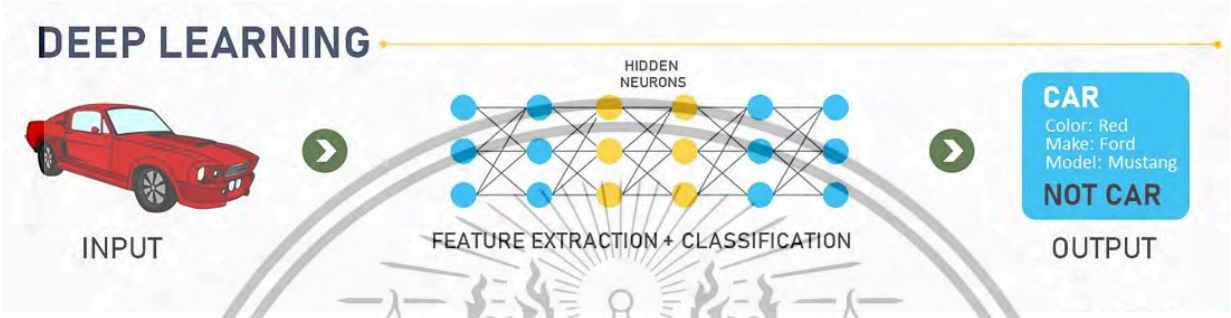

Figure 6: Deep learning workflow

Deep learning network works through the interaction between artificial neurons as shown in Figure 6. These neurons also knowns as units located in each layer of the neural network which is the group of neurons. These neurons are interconnected and always contain a bias. When the input layer receives input data which can be single or multiple data as input neurons, they are transmitted to the first hidden layer via weighted channels and pass through the following hidden layers sequentially. These hidden layers perform mathematical computations on our inputs. Choosing the amount of hidden layers and neurons for each layer is one of the difficulties in building neural networks. This could be the reason why "Deep" in Deep Learning refers to having "more than one" hidden layer (24). When the data arrives at the neurons, their weighted value is added to the bias and used in the activation function. Based on the result of the function, the neuron may or may not be activated. The information will be passed on to the next layer if the neuron is activated. However, if it's not activated, the information will not proceed to the following layer. This process will be repeated through the layers until the output is produced through the output layer as shown in Figure 7.

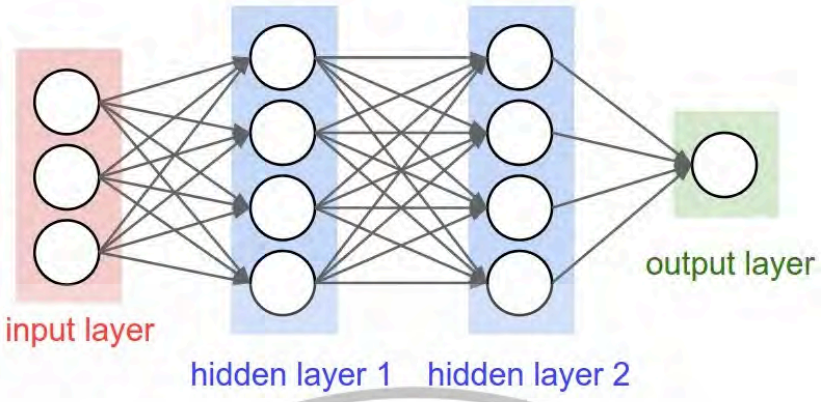

Figure 7: Forward Propagation or Feedforward

This progression pattern is called forward propagation since the processing information is flow only in one direction (26). Starting from the input layer, through multiple hidden layers, to the last output layer, without any feedback loops. They are often used for classification, regression, and prediction tasks, and have been successfully utilized in a variety of applications, including speech recognition, image classification, and natural language processing. Forward propagations use different activation functions, such as sigmoid, tanh, or ReLU, to introduce nonlinearity into the network (26).

The opposite progression is called backpropagation which is shortened form of backward propagation of errors. This model is the most popular and widely used for training artificial neural networks (26). Typically, backpropagation consists of 2 phases; forward propagation and backward propagation. After the neural network receives input data and calculates the output through forward propagation, the error between the predicted output and the actual output is calculated during the backpropagation, followed by adjusting the function's weights and biases in each neuron to reduce error (22) as shown in Figure 8. This approach uses the gradient descent optimization method to update the weights and biases of the network. When using the gradient descent approach, the error function's gradient is calculated with respect to the weights and

biases, and the weights and biases are then updated in the direction of the negative gradient, which decreased the error (26). Backpropagation applications include time series prediction, speech recognition, image classification, and natural language processing. It is a potent and effective technique for feedforward neural network training, and it has been applied effectively in numerous real-life situations.

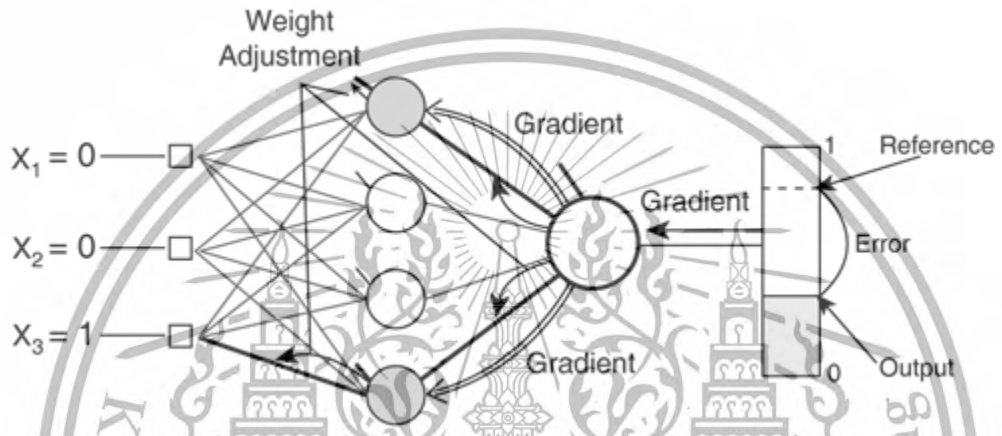

Figure 8: Backpropagation

Deep learning networks are frequently employed for a variety of applications, including autonomous driving, speech recognition, natural language processing, and image and image recognition. The network is trained using big datasets to optimize its parameters and enhance performance. The architecture of the network, including the number of layers and nodes, can be customized for the particular task and data at hand.

2.4 Convolutional Neural Network (CNN)

It is a type of deep learning network that is specifically created for tasks involving image and video recognition. It is based on the concept of convolution, which involves applying a filter (or kernel) to an input image to extract features from it. Convolution is the element-wise product of two matrices added together mathematically (27).

Overall, convolutional neural networks consist of several layers, including the input layer, convolutional layers, pooling layers, fully connected layers, and the output layer as shown in Figure 9. In the first convolutional layer, the network applies several filters to the input image, producing a set of feature maps that attract prominence to various patterns in the image. The size of the feature maps is then reduced by the pooling layer by using the maximum or average value of each group of pixels. Finally, the fully connected layer takes the flattened output of the pooling layer and uses it to make a prediction about the input image (27). The output is softmax when conducting multiclass classification (28).

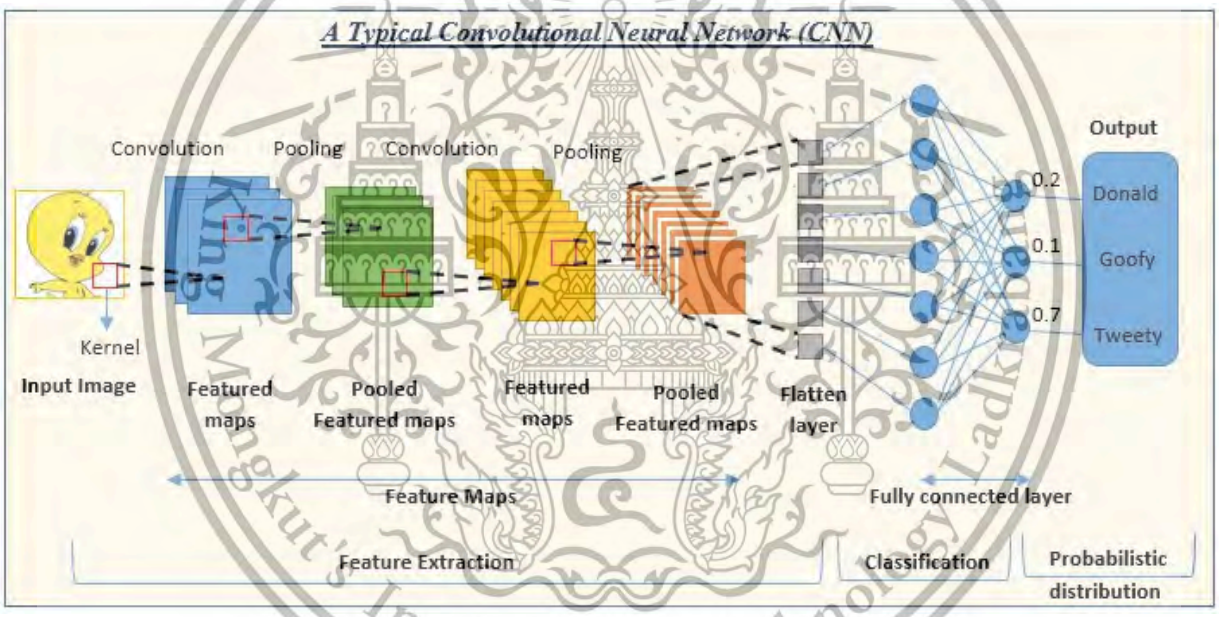

Figure 9: Architecture of Convolutional Neural Network (CNN)

Convolutional neural networks have several advantages over traditional machine learning models for image recognition tasks. They are able to automatically learn and extract features from raw image data, without the need for manual feature engineering. They are also able to capture spatial dependencies between pixels in the image, making them more effective at recognizing complex patterns. They have been used for a wide range of applications, including object detection, facial recognition, medical image analysis, and self-driving cars.

2.4.1 Convolution: Filters/Kernels

The convolutional layer is the foundation of CNN. Typically, **a convolution filter** also called the **kernel**, is used to construct a **feature map** by applying convolution to the input data. The example of convoluted filter/kernel is shown in Figure 10. This filter provides a metric for measuring how closely an input patch or area resembles a feature. Any noticeable aspect, such as a vertical or horizontal edge, an arch, a diagonal, etc., can be considered a feature (27). Noted that name the convolution according to their filter size.

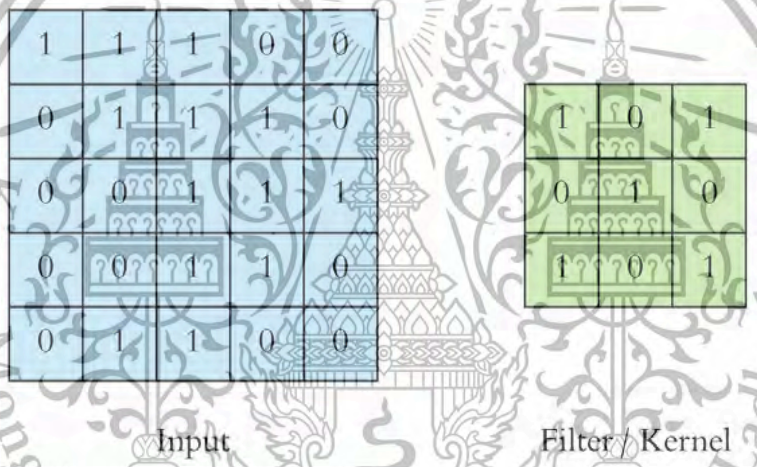

Figure 10: 3x3 Convolution, the left side is the input of the convolution layer, and the right side is the convolution filter/kernel

Performing the convolution operation by sliding this filter across over the input, from left to right, top to bottom, pass every location. During convolution, the filter will act as a template or pattern which will compare various areas/regions of the input image with the stored template to see where there are similarities (27). This will be done by performing element-wise matrix multiplication and summing the result, this result will go to the **feature map** as can be seen in Figure 11, the receptive field is the name given to the green area where the convolution operation proceeds. The receptive field is also 3x3 due to the filter's size (28).

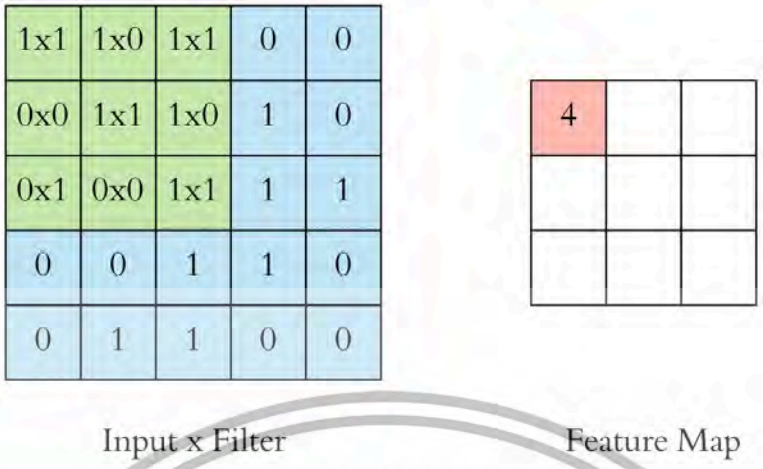

Figure 11: Performing convolution to get a Feature Map

Then continue doing the same process from the top left of the input until reach its bottom right. After the convolution operation is completed, the completed feature map is illustrated in Figure 12.

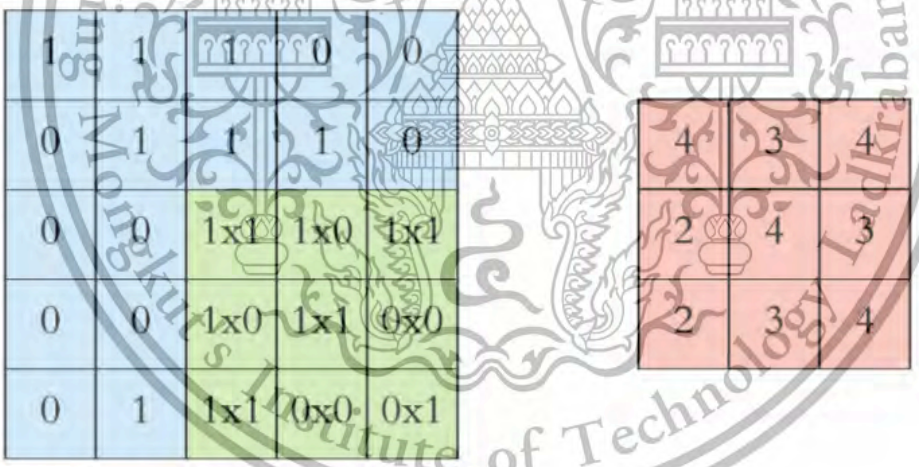

Figure 12: Completed Convolution in 2D using 3x3 Filter

The dimension of the convolved output can be calculated if the input image size and the filter size are known. The output size can be calculated using the following formula; **(Size of input image – filter size + 1) x (Size of input image – filter size + 1)** (26).

2.4.2 Stride

The term "stride" describes the number of pixels or the step size by which the convolutional filter travels over the input data or feature maps horizontally and vertically as shown in Figure 13. It controls the amount of spatial compression or downsampling that takes place throughout the convolutional operation. The filter shifts one pixel at a time if the stride value is 1. If the number is 2, two pixels are moved at a time, and so on.

Figure 13: Stride during convolution

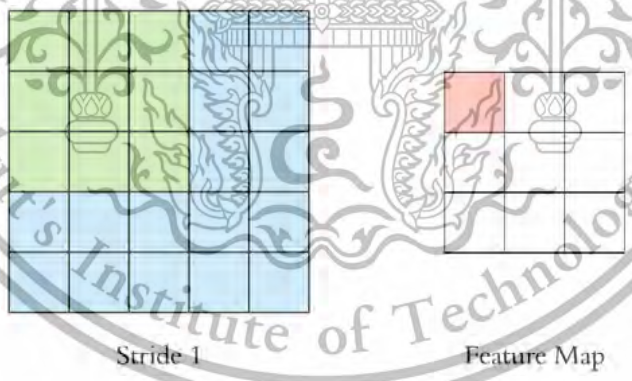

Figure 14: Stride is 1

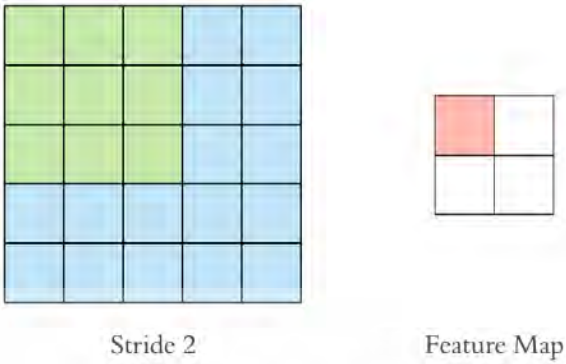

Figure 15: Stride is 2

By adjusting the stride value, we can control the spatial dimensions of the output feature maps. Specifically, a larger stride value leads to a reduction in the spatial dimensions (width and height) of the output feature maps, resulting in spatial downsampling as can be seen in Figure 15 which uses a stride equal to 2, the result feature map is smaller in size when compared with the result obtained from using stride 1 as in Figure 14. On the other hand, a stride of 1 preserves the spatial dimensions, providing a more detailed output feature map. Using a larger stride can be useful in certain cases, such as when we want to reduce the size of the feature maps or when we desire a coarser representation of the input data. It can help reduce computational complexity and memory requirements, as well as increase the receptive field of the network (28).

However, a larger stride may also lead to information loss or underrepresentation of small-scale patterns or details in the input data. Thus, it's a trade-off between spatial resolution and computational efficiency. Therefore, the choice of the stride value depends on the specific task, the complexity of the input data, and the desired balance between efficiency and accuracy in the CNN architecture.

2.4.3 Padding

Padding is a method of applying additional boundary pixels to feature maps or input data before performing the convolution operation. During the convolutional operation, it assists in maintaining the spatial dimensions and information at the input's boundaries. This is used because when a convolutional filter is applied to the input data, the output feature map always has smaller spatial dimensions compared to the input. This reduction in size can result in the loss of information at the edges of the input. Therefore, padding is used to address this issue by adding extra pixels around the input data, creating a padded input.

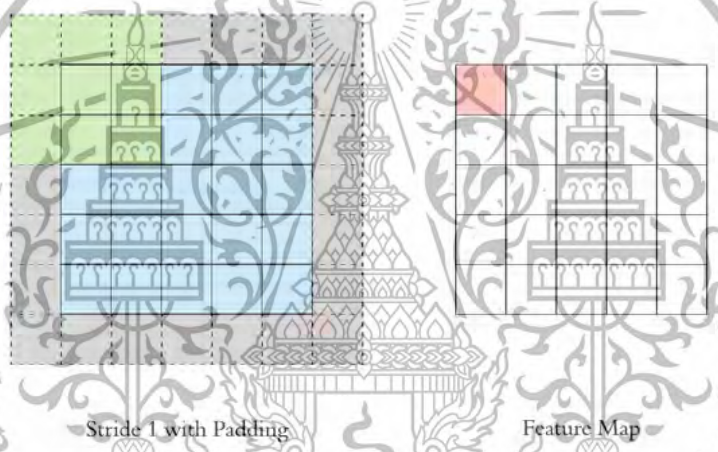

Figure 16: Padding with the same input image results in a feature map with the same size as the input

In the above figure shows the padding in a grey area, adding one pixel surrounding the input image. Therefore, if using stride 1 which will move along the feature horizontally and vertically by 1 pixel will result in the same size as the input image, the dimensionality of the feature map will finally match the input. Padding is an important technique in CNNs to control the spatial dimensions and maintain the integrity of the input data. It provides flexibility in designing network architectures and helps in achieving more accurate and robust feature extraction.

2.4.4 Pooling

Pooling, also known as downsampling or subsampling, is used to reduce the spatial dimensionality of feature maps. It helps in gathering the most crucial and pertinent information from the input while also reducing the network's computational complexity (28). Pooling minimizes the height and width while maintaining the depth on each feature map individually, and replaces a patch of pixels with a single value. The pooling operation is applied with a sliding window over the feature map, and the pooling function aggregates the values within the window to produce a single output value.

There are two common types of pooling operations in convolutional neural networks including max pooling and average pooling while the former one is more popular and widely used. For max pooling, it will select the maximum value from the pooling window as the output value. It effectively retains the strongest or most dominant feature within the window while also helping in preserving important spatial features and can provide some degree of translation invariance (28). While average pooling requires calculation in computing the average of the value within the pooling window and using it as the output value. It provides a smooth downsampling effect and can help reduce the impact of noise or outliers in the feature map. The example max pooling and average pooling are shown in Figure 17.

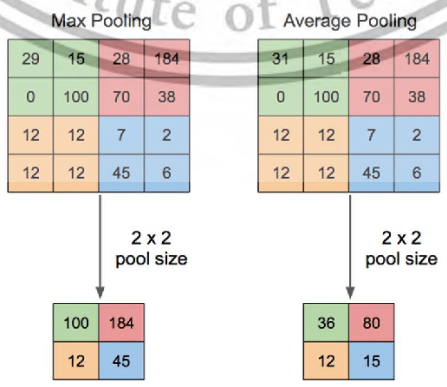

Figure 17: Max Pooling and Average Pooling

Pooling is typically applied after convolutional layers in CNN architectures and is performed multiple times, progressively reducing the spatial dimensions of the feature maps. The downsampling achieved through pooling helps in reducing the number of parameters and computations in subsequent layers, making the network more efficient and preventing overfitting. It also has the advantage of providing a degree of translation invariance. By taking the maximum or average value within the pooling window, pooling operations can capture the presence of features regardless of their precise spatial location, making the network more robust to small translations or shifts in the input data. However, pooling also leads to a loss of spatial information and can result in the loss of fine-grained details. The downsampling effect can reduce spatial resolution, potentially affecting the localization of objects or precise spatial relationships.

Overall, pooling plays a crucial role in CNNs by reducing spatial dimensions, improving computational efficiency, providing translation invariance, and helping to control overfitting. Its application and choice of pooling function depend on the specific task, network architecture, and the desired balance between spatial resolution and computational efficiency.

2.4.5 ResNet-50

ResNet-50 is a convolutional neural network architecture that belongs to the ResNet (Residual Network) family. ResNet-50 is widely used for image classification tasks and is known for its deep structure and residual connections, which help alleviate the vanishing gradient problem and improve training performance (29). Figure 18 displays the overview of the ResNet-50 architecture.

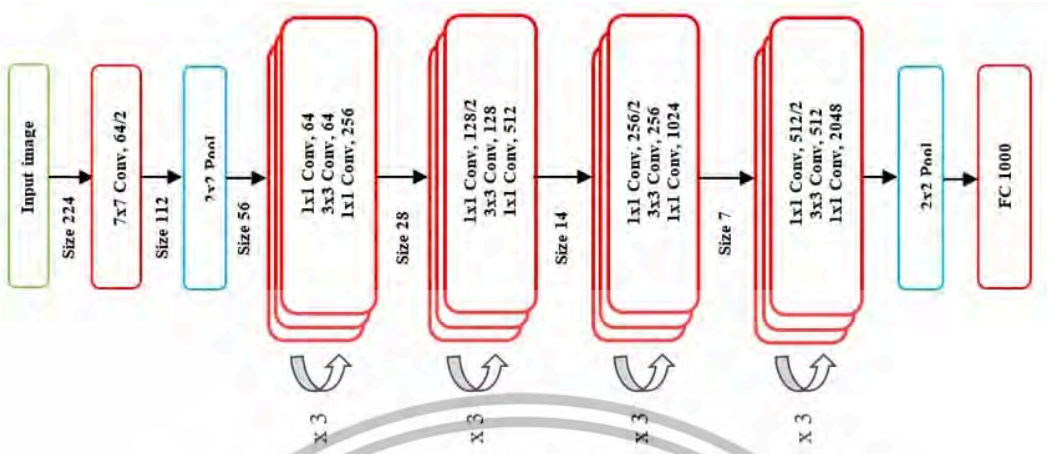

Figure 18: ResNet-50 Architecture (29)

The architecture of ResNet-50 is composed of the following:

1. Input Layer: The input to ResNet-50 is an RGB image typically of size 224x224 pixels.

2. Convolutional Layers: This layers are used to reduces the spatial dimensions of the input image. It is composed of 7x7 filters, a stride 2 and a max pooling layer with a 3x3 window and stride 2.

3. Residual Blocks: Multiple residual blocks stacked on top of each other. There are numerous convolutional layers with various filter sizes present in each residual block. The key feature of the residual block is the residual or skip connections that allow the gradient to flow directly through the block without vanishing.

- First Residual Block: The first residual block consists of two convolutional layers with 64 filters each, followed by a batch normalization layer and a ReLU activation function.

- Middle Residual Blocks: The middle section of ResNet-50 contains several residual blocks with different numbers of filters. Each block typically has three convolutional layers with 64, 64, and 256 filters, respectively.

- Final Residual Block: The final residual block is similar to the middle blocks but with a larger number of filters. It includes three convolutional layers with 128, 128, and 512 filters, respectively.

4. Global Average Pooling: This is to reduce the spatial dimensions of the feature maps to a vector representation. This operation pools the input feature maps into a single value per channel, which captures the global information of each feature map.

5. Fully Connected Layer: It is added to the network for the final classification. The output of previous layer is connected to this layer, which maps the features to the desired number of classes.

6. Softmax Activation: This is used to produce a probability distribution over the classes.

2.5 YOLO (You only look once)

YOLO (You Only Look Once) is an object detection model that uses a convolutional neural network as a feature extraction network to classify objects in images or videos and adds the detection heads of YOLO at the end of these network to used as the detection layers to detect specific object in an image. YOLO divides the image into a grid and predicts the bounding boxes and class probabilities for each grid cell in a single forward pass of the neural network, in contrast to standard object detection models that demand numerous passes over an image.

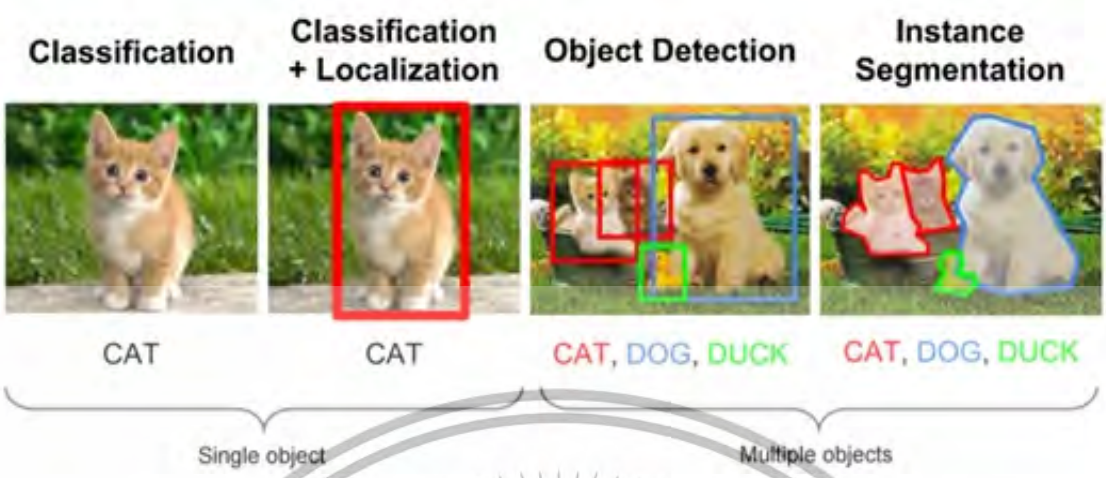

Figure 19: Object Detection and Classification

The YOLO model works by taking an input image and dividing it into a grid of cells. For each cell, the model predicts the bounding boxes of the objects that are present in the cell, along with the class probabilities for each object as shown in Figure 19. The bounding boxes are represented as four numbers that indicate the x and y coordinates of the top left corner of the box, as well as its width and height. This model is able to achieve high accuracy and fast processing times due to its use of a single neural network to predict object bounding boxes and class probabilities. This makes it well-suited for real-time object detection applications, such as self-driving cars or surveillance systems.

YOLO has been widely used in computer vision applications and has been applied to a wide range of tasks, such as face detection, traffic sign recognition, and animal tracking.

2.6 Literature review

There is some similar objective research related to using AI as object detection to detect malaria-infected red blood cells, this research applies similar types of models, YOLO which is the most popular way to detect objects that give high accuracy without decreasing detecting

speed. Based on Malaria parasite detection in thick blood smear microscopic images using modified YOLOV3 and YOLOV4 models published by the National Library of Medicine, they can provide 96.32% accuracy when using YOLOV4 and 95.46% and 96.14% when using 2 modified YOLOV3. They can use AI to locate the parasite cells and classify the malaria species only to detect plasmodium falciparum in thick blood smear microscopic images captured using a smartphone camera. Since they were observed on thick blood smear images, this type of image will contain many layers of blood cells, and difficult to observe the morphology of cells, as well as to count the number of cells to calculate the percentage of parasitemia. Therefore, the microscopic images required in this project need to be in the form of a thin blood smear because it will be only a single layer of cells there, which is easier to observe the morphology and to identify the growth stages of malaria-infected red blood cells. From the mentioned research, they have modified the YOLOV3 because they wanted to enhance the capability to detect small objects without notably decreasing detection speed by increasing the feature scale and adding more detection layers. This is because the former version of YOLO provides low accuracy when detecting objects when compared with the later one. Therefore, in my project, I need to modify the layers and features as well with the YOLOV2 and YOLOV3 before comparing them with the YOLOV4.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter describes the methodology of developing models to classify and detect red blood cells infected by Plasmodium falciparum from microscopic images. Primarily, the convolutional neural networks will be applied to classify the infected patient just from their blood smear test. This will be followed by developing YOLO models to detect the different stages of each parasite cell from patients' thin blood smear images. This will be done by using 2 different versions of YOLO or You Only Look Once including YOLO versions 3 and 4 that are available in MATLAB.

After the training is done, the testing data which is excluded from training data is classified and detected by the network and the performance of the network is then determined and compared by the accuracy.

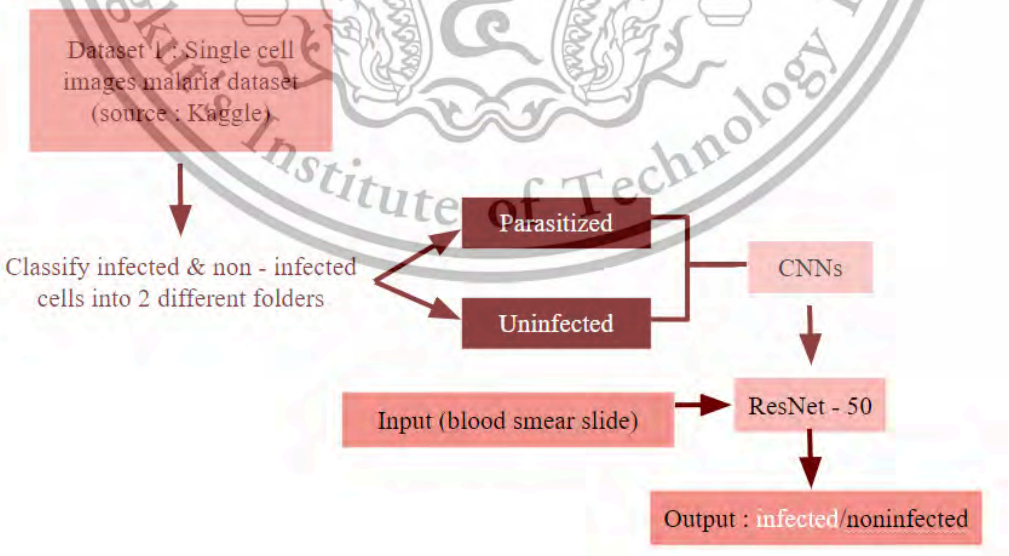

Figure 20: Classification Diagram 1

The above block diagram illustrates the overall procedures of this project. They are starting with receiving 2 types of datasets; one from an open-source published on the Kaggle website and another one from a malaria dataset of plasmodium falciparum species in both microscopic images and a video from a supervisor which have been approved by the ethics committee of KMITL. Then manually separate each image of the first dataset into 2 folders; parasitized and uninfected, these will be used to train with a convolutional neural network, ResNet-50. The output of this network is classified into 2 classes to diagnose whether a patient is infected. Secondly, if the output shows that the patient is infected then develop another network to classify malaria stages.

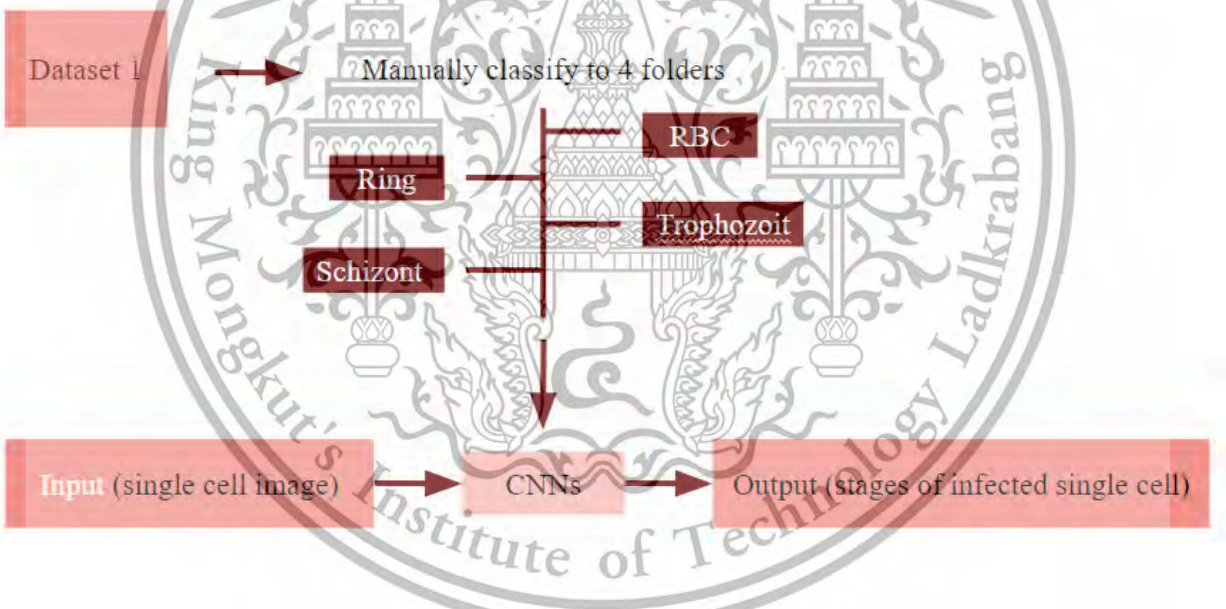

Figure 21: Classification Diagram 2

The second network as shown in diagram in Figure 21 aims to classify the stages of the malaria parasite after knowing that this patient is already infected. Similar to the first network, this second network will be trained using the first dataset. However, when the dataset is manually classified, it will be divided into four folders rather than two, with each folder containing a

different stage of the malaria parasite, including the ring, trophozoite, schizont form, and healthy red blood cell form. Next, train 3 distinct networks—ResNet-50, ResNet-101, and Darknet-53—on this dataset. Lastly, compare the classification accuracy of different networks after testing them using particular samples.

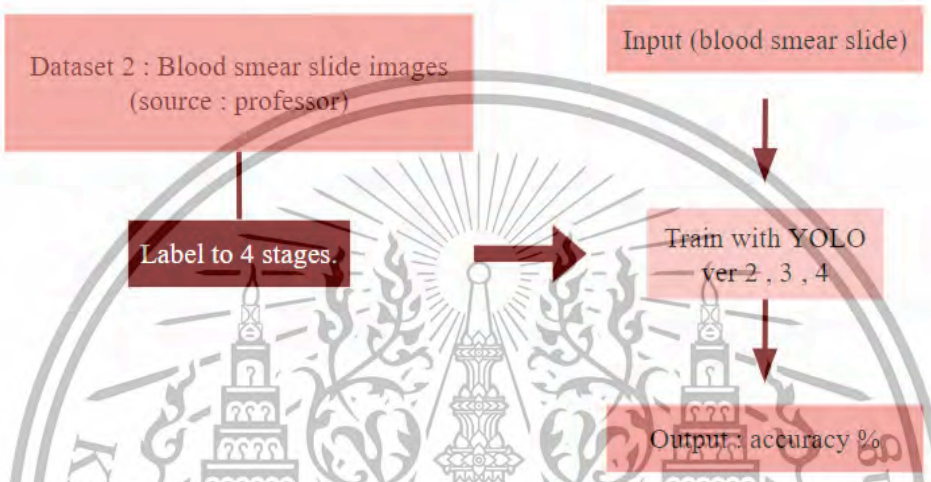

Figure 22: Object Detection Diagram

The above diagram displays the overall process of developing object detection models used to detect infected red blood cells in the blood smear field. Starting with obtaining the second dataset from the professor and then start labeling each cell according to its classes or stages. Followed by training these data with the object detection models which are the 2 versions of YOLO and testing with some samples of microscopic images to compare which models show the most accurate results.

3.2 Install required materials in MATLAB

There are two tools that you should ensure are already installed before moving on to the next steps., the Deep Learning Toolbox and the Computer Vision Toolbox which offers many

required functions and materials for computer vision tasks such as image processing, classification, and object detection. However, if you use the recent version of MATLAB, these built-in toolboxes are usually available in MATLAB. If you are using an older version of MATLAB that does not include these toolboxes, you may need to upgrade to a newer version that includes them.

Additionally, the general procedures to make sure that your MATLAB program has the Deep Learning Toolbox and the Computer Vision Toolbox installed and accessible are described in the instructions below:

1. Open MATLAB on your computer.
2. Verify product name, version number, and release name for MATLAB and all other installed products by typing the following command in MATLAB Command Window: “ver”

This command displays the information about the installed toolboxes and their versions. Look for the “Deep Learning Toolbox” and the “Computer Vision Toolbox” in the displayed list.

3. If some or both of these toolboxes are not listed or not installed, you may need to update your MATLAB installation. You can check for updates by going to the "Home" tab on the MATLAB desktop, dragging down on “Help” and selecting "Check for Updates."

4. If updates are available, follow the instructions to install them. Make sure to select the Deep Learning Toolbox and the Computer Vision Toolbox during the installation process.

5. Once the Deep Learning Toolbox and the Computer Vision Toolbox are installed, you can start using their functions and features in MATLAB. You can find examples, documentation, and tutorials in the MATLAB Help Center or by typing "help" followed by the function name in the MATLAB Command Window.

3.3 Input Dataset Preparation and Labeling Process

3.3.1 Malaria Dataset

There are two types of datasets that will be used in this project, the first dataset is composed of 27,558 single-cell images of both plasmodium falciparum-infected red blood cells and healthy red blood cells as shown in Figure 23. This dataset is received from the Kaggle website which also took this dataset from the official NIH Website since they are quite slow to download this dataset from the original source (30,31). This dataset is publicly accessible anybody can access any time. The dataset was used in research done by researchers at the Lister Hill National Center for Biomedical Communications (LHNCBC), which is part of the National Library of Medicine (NLM). This research was aim at developing a mobile application for screening malaria parasites which will reduce the burden for microscopists, especially in rural areas, in the diagnostic process and also enhance diagnostic accuracy. These single-cell images are gained by dying thin blood smear slide with Giemsa-stained of 298 infected patients and 95 uninfected people. The image size of all images are not the same since they are already

segmented from the blood smear slide that different cells may have different size resulting different image size. This dataset is more appropriate to do the classification since it contains a large number of images.

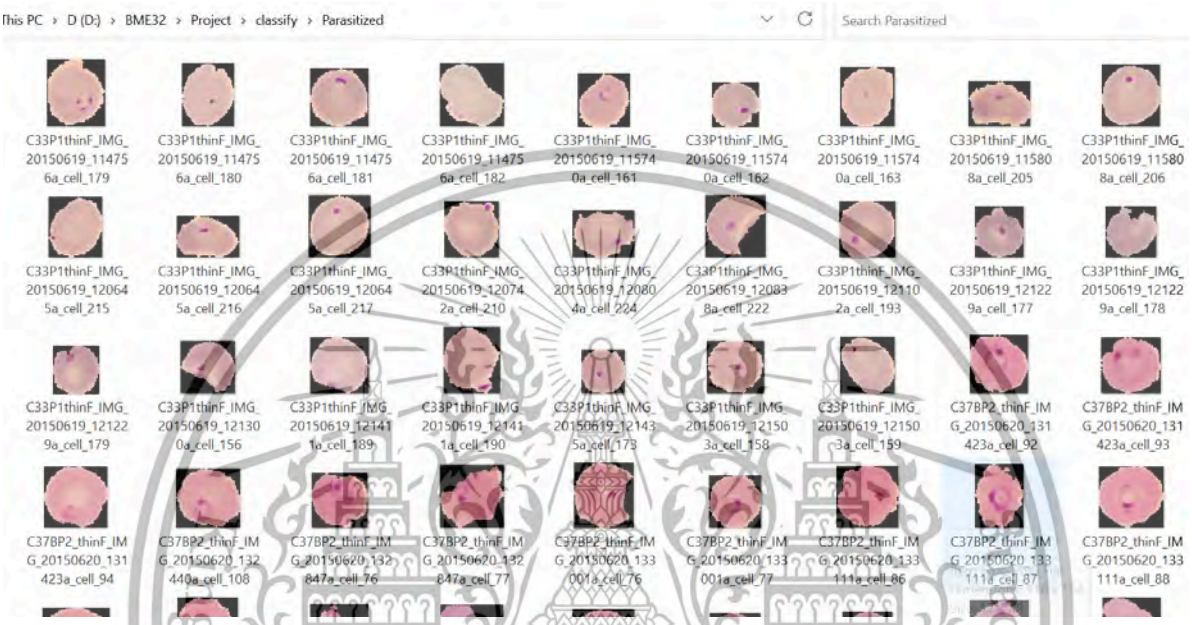

Figure 23: Plasmodium falciparum Dataset1 (single-cell images)

The second dataset as shown in Figure 24 contains 99 microscopic images of plasmodium falciparum-infected patients in the form of thin films that are optimal for assessing the morphology of any parasitic forms and can classify the stages of malaria effectively. All images are in 1600x1200 pixels. These slides were dyed with Giemsa dye made of eosin and methylene blue so that it will be red color on the nucleus of the cell and show the cytoplasm of cells in blue, this dye is costly but more stain is consumed which is easier to observe both healthy red blood cells and infected red blood cells. It also contains a video microscopic film used to test the deep learning models in the final process in order to count the number of cells in a single slide, this video is a different image from the rest dataset. The received dataset has been

approved by the KMITL Human Research Ethics Committee, the code for this dataset is EC-KMITL_65_103.

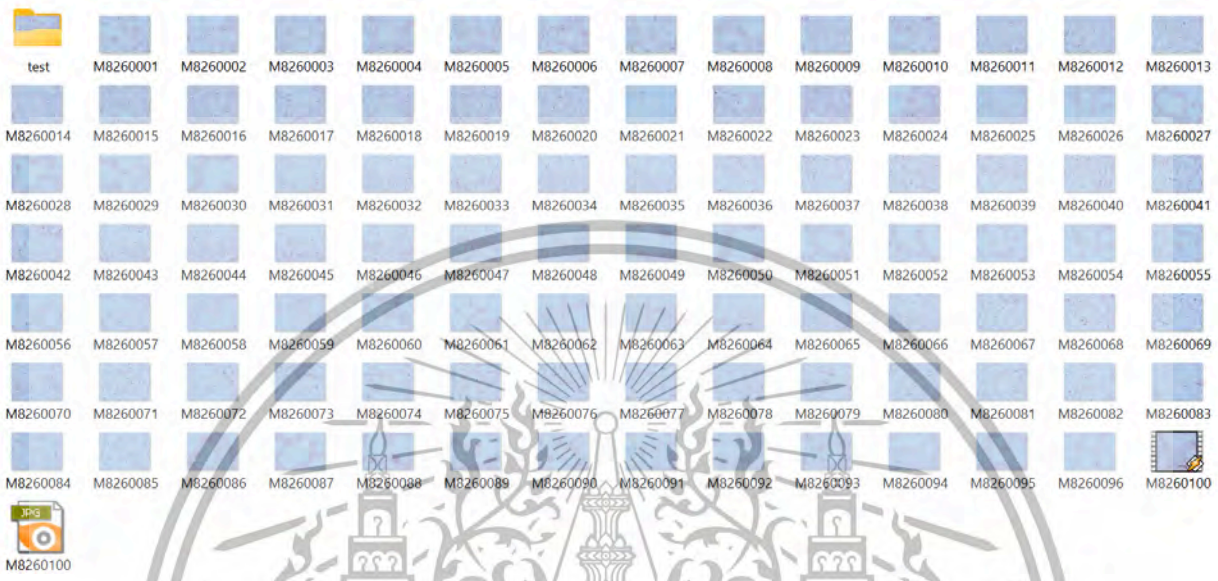

Figure 24: Plasmodium falciparum Dataset2 (Blood smear field)

3.3.2 Dataset Organizing and Labeling

3.3.2.1 Criteria to Identify the Stages of Malaria Parasite

There are 4 major stages of malaria that can occur in human blood circulation; the first stage is called ring form, the second stage is trophozoite form and another stage that we want to detect is schizont form which is the stage when patients start to show symptoms such as fever and flu-like illness, including shaking chills, headache, muscle aches, and tiredness some nausea, vomiting, and diarrhea may also occur. Moreover, the reproductive form of this parasite will be detected, and gametocytes form.

The following paragraph shows the criteria for identifying 4 different forms of malaria parasites and Figure 25 shows different shapes of each form.

1. Ring forms – appear as a pale blue ring with a pink/purple chromatin dot on the ring, and more than one may be present in a single red blood cell, usually found 1-3 rings in a single cell.

2. Early trophozoite – the ring is still observable but the addition of some blur chromatin dot in the cytoplasm occurs.

3. Late and mature trophozoites – parasites with denser cytoplasm but still blur masses showed in pink/purple color.

4. Schizont – parasite with multiple masses of obvious nuclear chromatin and clumps of dark brown pigment, some cells will burst during this phase and release parasite contents to infected nearby cells.

5. Gametocytes – parasites have a crescent (banana) shape with round/pointed ends, approximately one and a half larger in size than normal red blood cells.

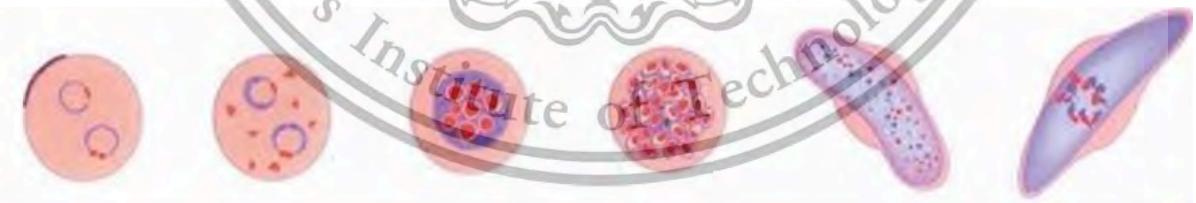

Figure 25: P. falciparum in the ring, early trophozoite, mature trophozoite, schizont, and gametocytes forms.

3.3.2.2 Manually Saparate Dataset Folders according to their Classes

The first dataset must be classified into 2 major folders depending on the using purpose in the following steps. One folder should separate malaria parasites into two subfolders; parasitized and uninfected as shown in Figure 26. Since the mature red blood cells do not contain a nucleus, thus they should not show any dot or contamination in healthy red blood cells in the uninfected folder and if there are some dots or contents inside the red blood cells, those images will be classified as infected cells which will be located in the parasitized folder. This folder will be used to develop a deep learning network to classify whether a patient's blood smear slide is infected or not, used as a tool to facilitate the diagnostic process. Lastly, each folder contains the same number of images which is 13,780 images for each.

Figure 26: Separate dataset images into 2 folders; Parasitized and Uninfected

The other folder should use the same dataset, single-cell images to be manually separated into 4 different classes according to their morphologies including ring, trophozoite, schizont forms, and healthy red blood cell form (RGB) as hown in Figure 27. The criteria for organizing images into distinct folders are described as mentioned in 3.3.2.1 Criteria to Identify the Stages of Malaria Parasite in the above section. Finally, this dataset is composed of 267 RBC images, 10,303 ring images, 796 schizont images, and 2,417 trophozoite images.

Figure 27: Separate parasitized dataset into 4 folders according to their morphologies

3.3.2.3 Image Labeller in MATLAB

The second dataset in the form of blood smear slide images will be used to develop the ground truth data in order to prepare the dataset before entering the training process of object detection part and evaluating computer vision models and models, starting from downloading the dataset folder name 'Slide' into the Image Labeler window. To download a folder containing the targeted dataset, you can type "fullfile" followed by the name of your dataset folder. After that, launch the Image Labeler app in MATLAB by typing "imageLabeler" in your code. This ground truth code is shown in Figure 28.

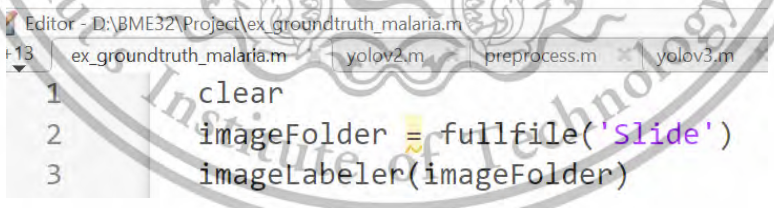

Figure 28: The ground truth code

Users can interactively label items or regions of interest in an image, a series of photos, using the Image Labeler app in MATLAB. It offers an easy-to-use interface for labeling and annotating data for tasks including object detection, semantic segmentation, and image classification. It is a part of the Computer Vision Toolbox. The user-friendly interface of

the Image Labeler software offers interactive tools and settings for quick and accurate annotation of image data.

The users can perform multiple tasks by this Image Labeler app as mentioned in the following section:

- Image Annotation: Load an image or a sequence of images and create a label class to annotate the object, then draw bounding boxes surrounding regions of interest, which can be drawn in many shapes such as rectangle or polygons.
- Image Region Labeling: Assign labels or categories or subcategories to the annotated regions to indicate the class or type of the objects.
- Label Editing: Resize, move, or remove existing annotations to edit existing labels. The labels that have been given to the regions can also be changed.
- Labeling Automation: Use interpolation tools to automatically propagate annotations from one frame to another in a sequence of images.
- Export Labeled Data: Save and export the annotated images and corresponding labels in various formats, including MATLAB workspace variables, image label files, or ground truth data for training machine learning models.

After opening the Image Labeler containing your dataset folder, create 5 classes according to the developmental stages of plasmodium falciparum; ring, trophozoite, schizont, gametocyte, and normal red blood cell (RBC). Each class can be created by going to the ROI Labels tap (region of interest) in the Image Labeler window and selecting “Lebel”. The “Define New ROI Lebel” window will appear as shown in Figure 29, this window is used to

name a new class, select the shape of the region of interest as “Rectangle” and the color can also be changed in this window. After finishing setting all parameters, click “OK” to create a new class, then this new class will display in the ROI Label tab. Then continue the same process to create all 5 classes as mentioned above.

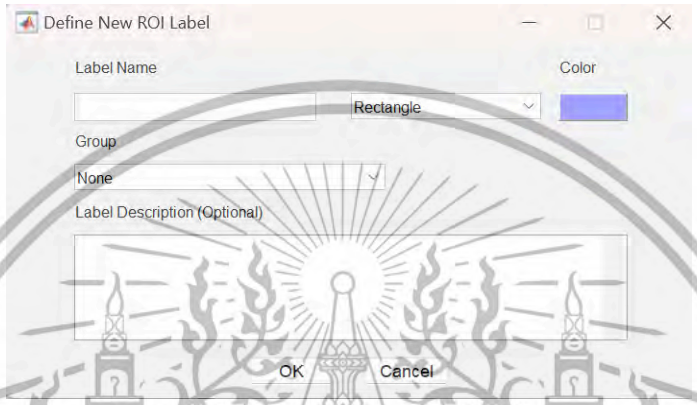

Figure 29: The “Define New ROI Label” tab

Then start labeling each class in each image sequentially from the first image to the last image without skipping any image to prevent any error during labeling. To label each cell, select the class accordingly and crop the area of each cell, there will show different colors on different classes as shown in Figure 30. Or if you want to delete any selected cells, you can click that rectangle and press “Delete” on your keyboard.

Since the number of healthy red blood cells is relatively much higher than those infected cells in this dataset, therefore, label only a few cells on a single image as the dataset is enough. However, to enhance the accuracy of object detection, adding more datasets and labeling as many as possible cells are required, leaving some healthy red blood cells the same as in this case is just to reduce the labor work during the labeling process because it is quite difficult to label all healthy red blood cells in every image of this dataset.

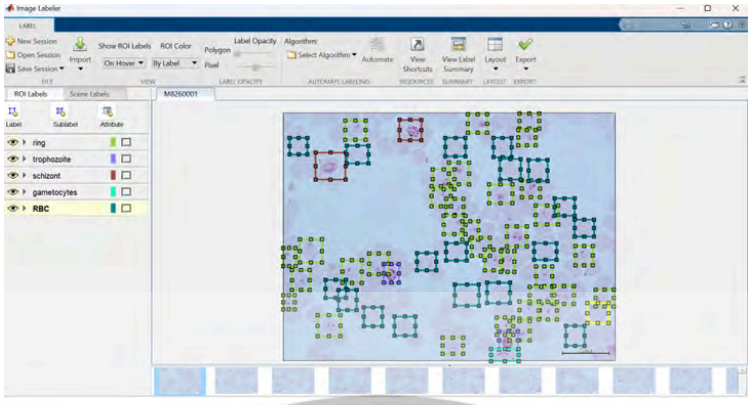

Figure 30 : Image Labeller in MATLAB, labels all parasite cells according to their stages as well as a few examples of normal red blood cells.

After finishing labeling the dataset, summarize the number of each class by clicking on "Lebel Summary" on the Image Lebeller window. It will display the statistical information of each class in each image in your dataset as shown in Figure 31. However, it does not count the total number of each class as a whole.

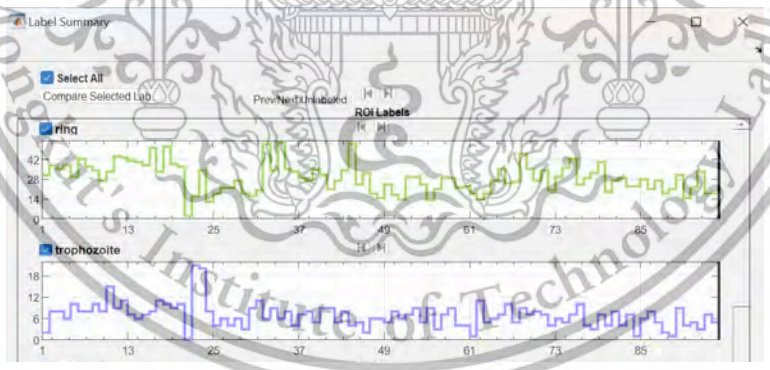

Figure 31: Label Summary tab

Lastly, export the labeled data to your workspace by going to "Export" under the Label tab and selecting "to Workspace", the "Export to workspace" window will appear. Then change the Export format to "table" and click "OK". Thus, this allows you to observe the gTruth parameter in your workspace of oMATLAB window. Therefore, you can save

this labeled dataset in your folder for future reference during the training process by “save” command as shown in Figure 32. This command can be located in the same file as your groundtruth codes but it must be commented on while running groundtruth commands to prevent errors.

```
7    save malariaDatasetGroundTruth.mat gTruth
```

Figure 32: Command to save gTruth parameter as malariaDatasetGroundTruth.mat

3.4 Methodologies for Classification models

3.4.1 Diagnostic Classification using ResNet50

3.4.1.1 Prepair ResNet50 Model

Download ResNet-50 Network

Open the Deep Network Designer to edit the pretrained ResNet50 model by typing the “deepNetworkDesigner” in the command window. Then select Open the “Resnet-50” network (make sure to install this model first by going to the Add-On Explorer window and downloading this network). Second, click export this network to the workspace, the 1x1 layerGraph will appear in the workspace. Then save this network as “resnet50” by right click on this parameter and selecting “save as” to save them in the same location as the dataset and is always ready to be loaded anytime. Then this pretrained network will be saved as a resnet50.mat file.

Edit some layers

First, type “close all” and “clear all” in the command window to delete any unnecessary parameters in the workspace and to prevent any interruptions. Second, type “load resnet50.mat” in the command window then 1x1 layerGraph will appear in the workspace. Third, this pretained network is not appropriate for the new datasets which means it requires some modification by editing them on the Deep Network Designer, typing “deepNetworkDesigner” in the command window. Click import network from the workspace, then select ok to import a LayerGraph with 177 layers to the Deep Network Designer as shown in Figure 33.

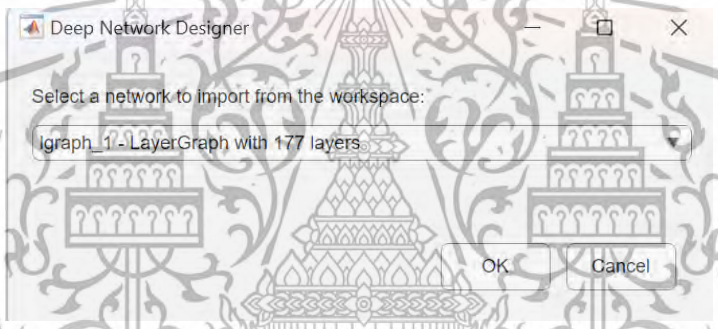

Figure 33: Layer Graph imported from the workspace

Lastly, explore and modify the network to make them appropriate for the dataset by editing the following layers:

- Remove the last output layer which is the classificationLayer that was used to classify 1000 classes of objects and replace it with new classificationLayer output that will automatically set the number of classes as shown in Figure 34.

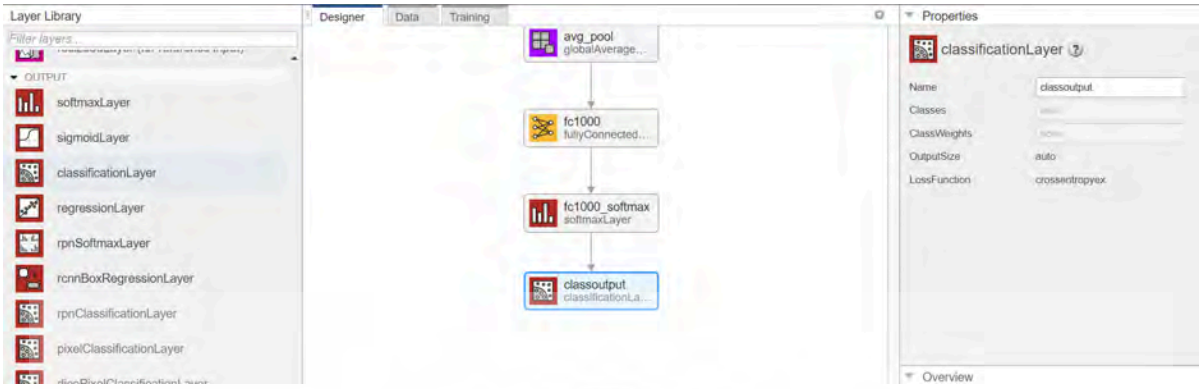

Figure 34: Auto Classification Layer as the last layer

- Remove the Fully Connected layer located at the third position counting from the last layer and replace it with the new one that can adjust the output to 2 classes since we want to classify whether a patient is infected and the input will be automatically set as shown in Figure 35.

Figure 35: Two Outputs in the Fully Connected Layer

Therefore, after finishing the layer modifications, the total layers is 177 layers. This modified network can be saved for use next time by clicking on the "Export" to the workspace and right-clicking at its parameter to save the modified network in the same location as the dataset.

Finally, the modified network should be analyzed to check whether there are any errors before training, zero warning and error should appear on the top right corner as shown in Figure 36.

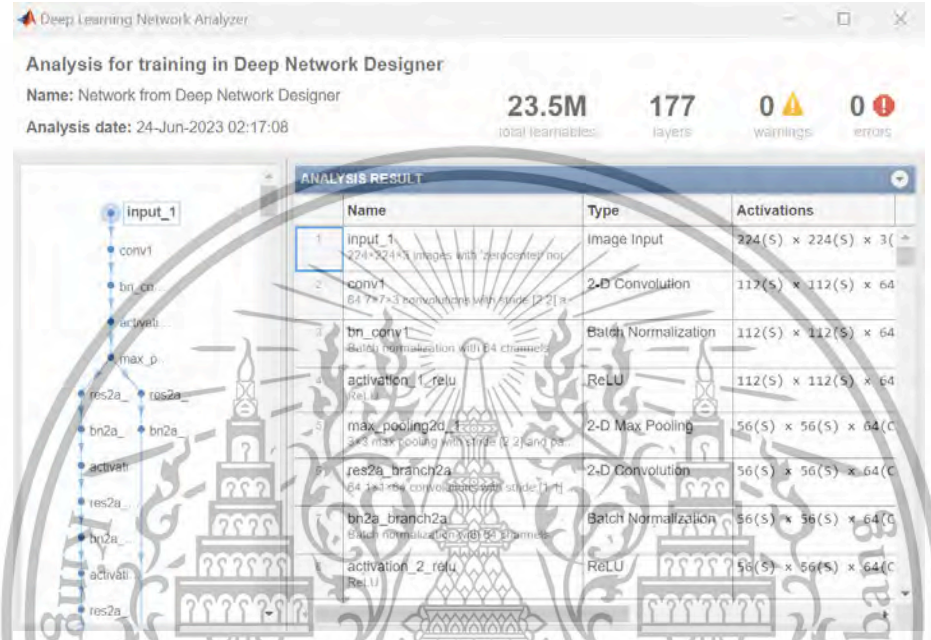

Figure 36: Analyze ResNet-50 Network

3.4.1.2 Import Dataset

After finishing editing and analyzing the network in the Designer tab on the Deep Network Designer window, then move on to the Data tab on the same window to import our dataset for visualization before continuing the training process. The dataset was previously divided into two folders, Paratisized and Uninfected folders; therefore, choose "Folder" as our data source and navigate to the folder's location to import them. Then select options for augmentation as shown in Figure 37 because the image proportion of each folder is not balanced, some folder contains much more images than others, such as the ring folder

containing almost 40 times larger than the RBC folder. The augmentation will be done to add more variety to the training data without actually having to increase the number of labeled training samples. It will randomly transform the original data during training such as randomly flipping the image multiple times. Lastly, split validation data for 20% for validation. This is used to certify learning outcomes from the training process that the existing network has reached the expected standard and also validate that the entire network is designed correctly.

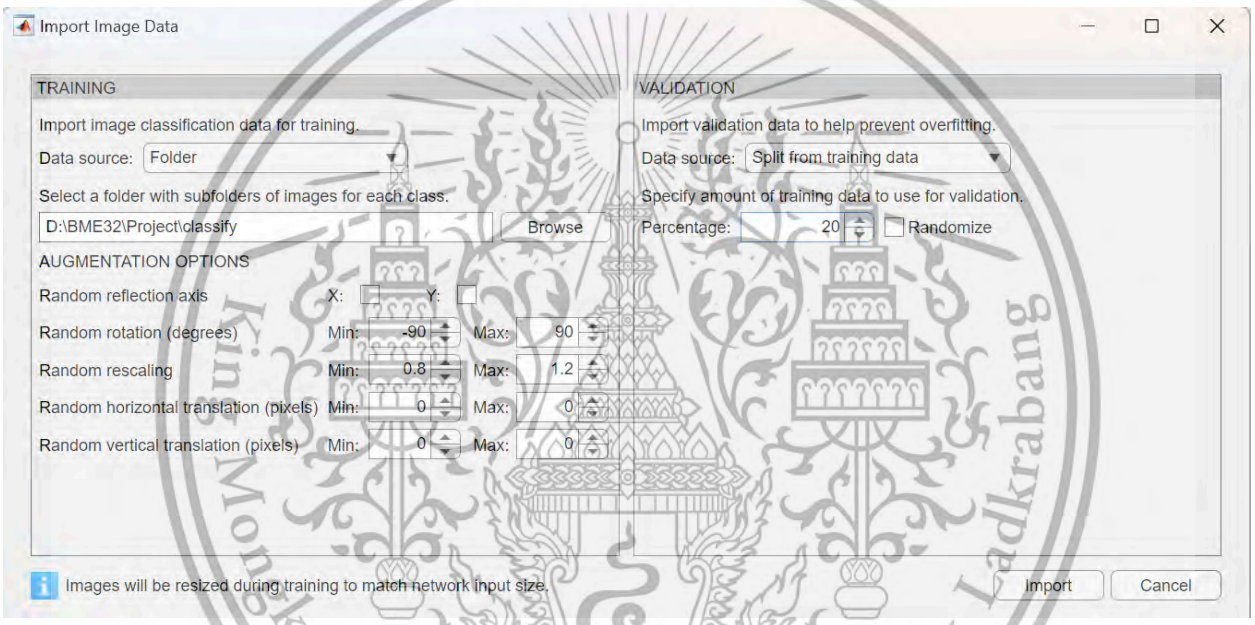

Figure 37: Training and Validation Options for Import Image Data

After finishing setting all parameters for training and validation then click "Import" to import the chosen dataset. Then the data will be automatically uploaded and shown in two categories; Parasitized and Uninfected. The number of observations of each class and some random images are shown in Figure 38 below. It also shows the most and the fewest observations on the same page. This interface clearly shows the quantity and difference in the number of images in each class for the user to adjust the number of datasets if they want the same proportion for each category, then add or remove images to have the same number.

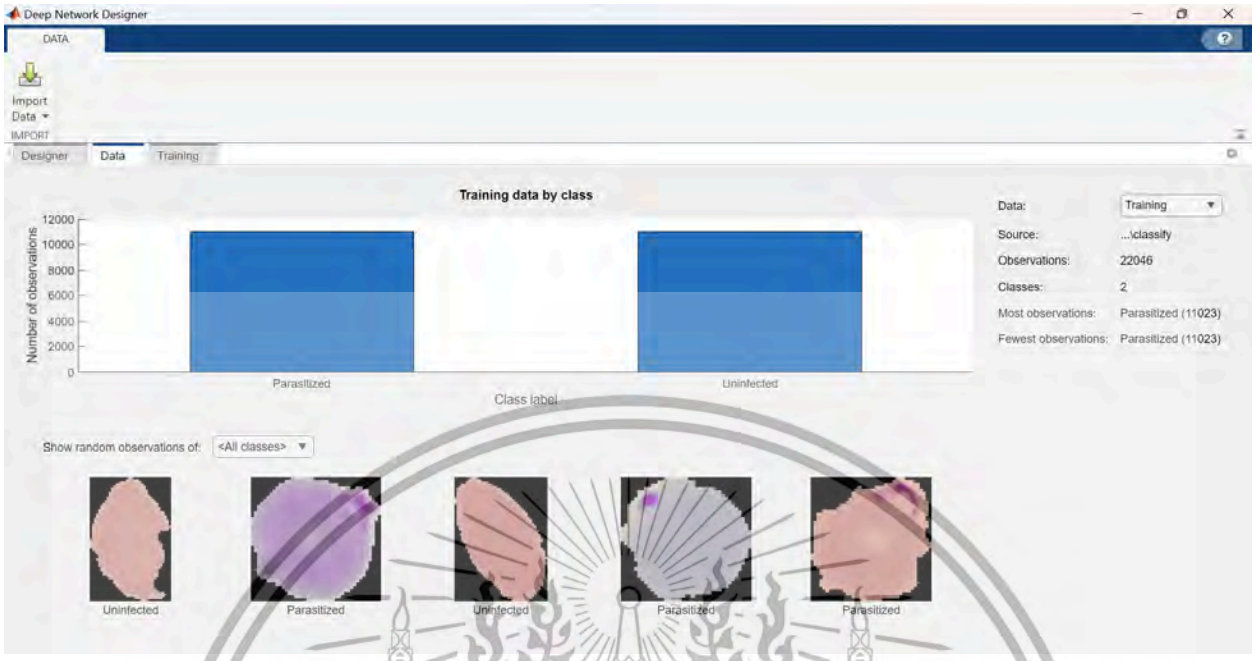

Figure 38: Import Image Data information

3.4.1.3 Training

Moving to the training tab, adjust the training options as shown in Figure 39. Set the Initial learning rate as 0.0001, validation frequency to be 50, maximum epochs as 8, and the default mini-batch size that is 128. Next, click start training, then the training progress will appear as shown in Figure 40, and wait until finish training.

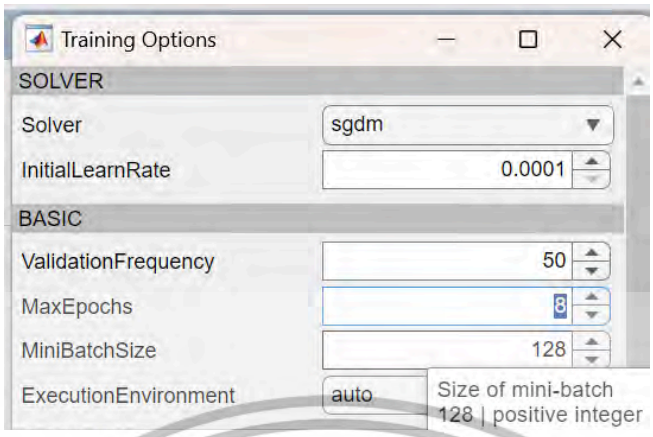

Figure 39: Set Training options

Figure 40: Training Progress

After finishing training, click "Export" to export the training network and trained info struct to the workspace. Then permanently save them as "trainedNetwork_1.mat" and "trainInfoStruck_1.mat", respectively, in the same location as the dataset. Therefore, we can easily recall the training network any time when we want to use them to classify any images.

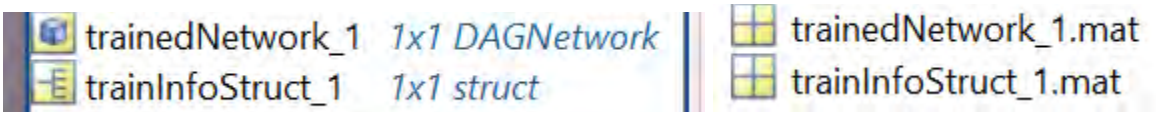

Figure 41: Exported parameters in the workspace and Training network MATLAB files

In Figure 41, the left image shows exported parameters in the workspace which are exported from the training dataset and the right image shows a saved training network that saves in the same location as the dataset.

3.4.1.4 Testing

First, select 20 images from the second dataset which already ensures that those images are received from the patient's blood who is infected by plasmodium falciparum malaria species. Create a new folder called "test" to store these images and use them to test the network. Second, develop MATLAB code to read the test images as shown in Appendix A which is composed of 4 main steps, starting with loading the training network named "trainedNetwork_1.mat" and "trainInfoStruck_1.mat" to the workspace. Then read the test image using "imread" function and resize the image appropriately same as our ResNet-50 network which is 224x224 pixels using the "imresize" function. After that use the "classify" function to classify the tested image and display its predicted label and calculate the class probabilities by using the "imshow" function to display the classified image at the end. A sample of the tested image is shown in Figure 42.

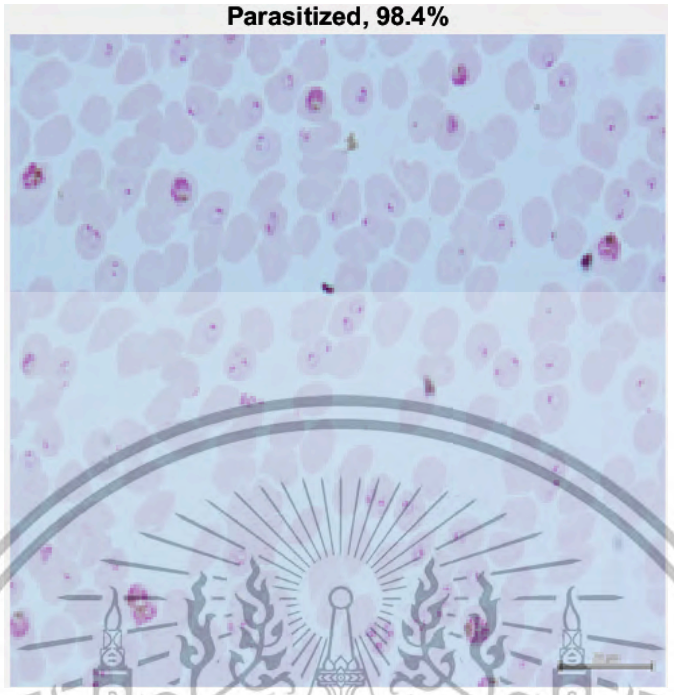

Figure 42: Test Network by classifying the test dataset

3.4.2 Malaria Stages Classification using ResNet-50 and ResNet-101

3.4.2.1 Preparing CNNs Models

Basically, the procedures to prepare ResNet-50 and ResNet-101 models are similar to the one mentioned in 3.4.1.1 Prepair ResNet50 Model which starts from installing these pretrained networks from the Add-On Explorer window. Second, open installed networks in the Deep Network Designer window to edit 2 important layers that are used for classification, which are the fully connected layer located at the third position counting from the last layer, and the classification output layer located at the bottom. The last output layer will be replaced with the new one that will automatically set the number of classes similar to the previous steps. However, the fully connected layer of both models should be replaced with the different output sizes, they should classify 4 classes rather than two since our dataset contains 4 folders including

RBC, ring, trophozoite, and schizont. As a result, this fullyConnectedLayer's properties will match those in Figure 43.

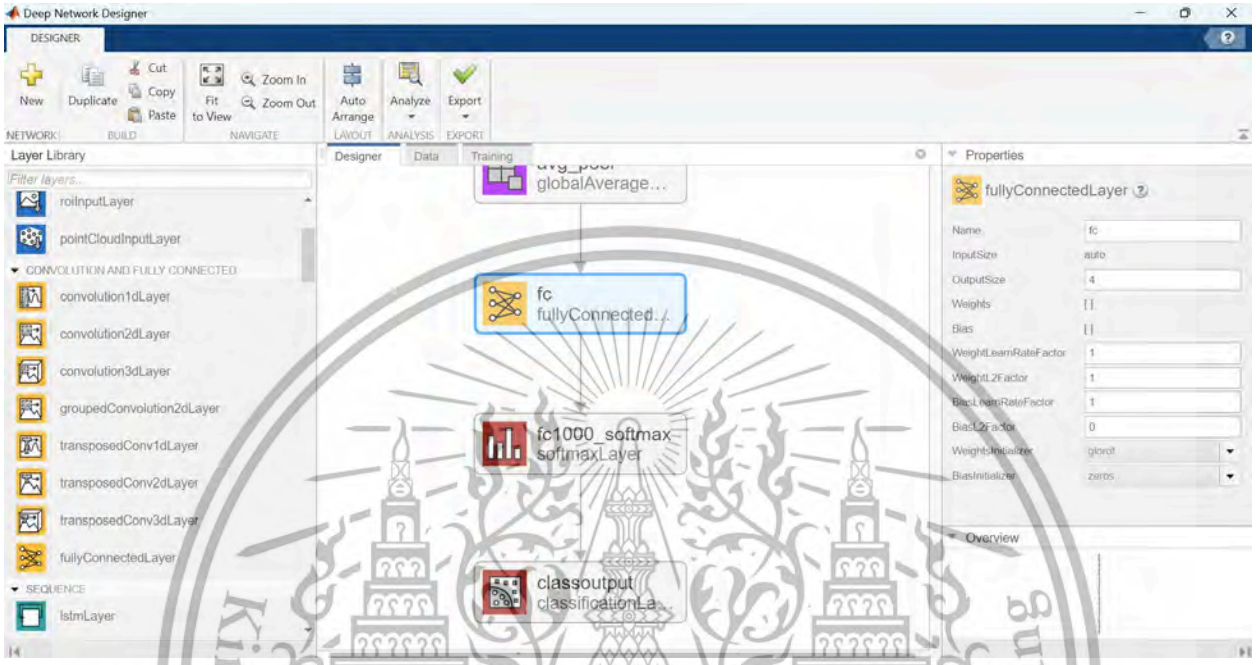

Figure 43: Edit Output size for fullyConnectedLayer to 4 classes for both models

Therefore, after modification, the total number of layers for each model will remain the same, the ResNet-50 model is composed of 177 layers as shown in Figure 44, and the ResNet-101 model is composed of 347 layers as shown in Figure 45. Then export the modified network to the workspace and save them as "resnet50.mat" and "resnet101.mat" in the same location as in the second dataset.

Finally, these modified networks should be analyzed before use to check if are there any errors, zero warnings and errors should appear on the top right corner.

Figure 44: Analyze Modified ResNet-50 Network

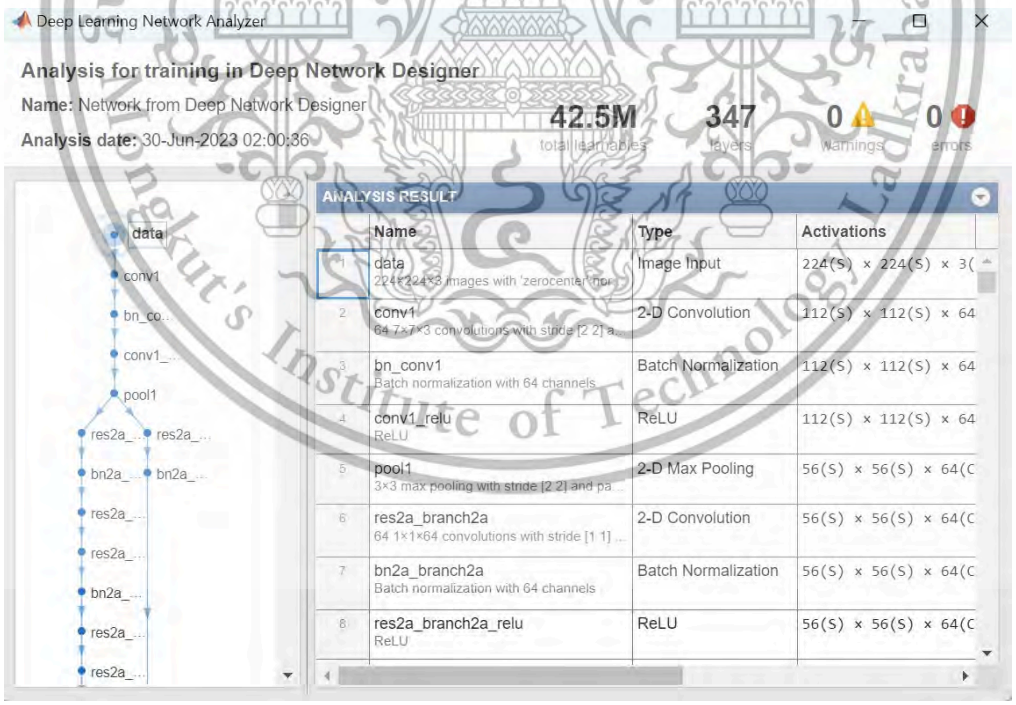

Figure 45: Analyze Modified ResNet-101 Network

3.4.2.2 Import Datasets

This step will be done similarly to the procedure mentioned in the 3.4.1 Diagnostic Classification using the ResNet-50 section. Starting with going to the Data tab and clicking “Import Image Data”, select the data source as a folder and browse the location stores 4 folders of malaria stages. Second, set the augmentation option and validation option the same as in Figure 37 and click “Import”.

Then the dataset will automatically be uploaded and shown in 4 categories; RBC, ring, schizont, and trophozoite. The related information such as the number of total observations, each class observation, and the most and the fewest observations are also shown on the same page as well as some random images are also shown in Figure 46.

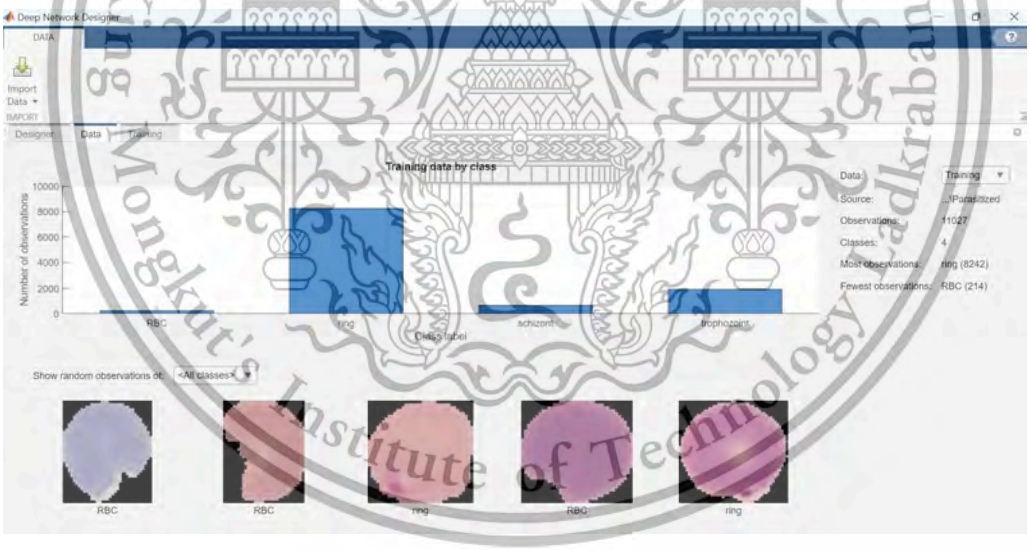

Figure 46: Import Image Dataset for both models

3.4.2.3 Training

Moving on to the training tab, adjust the training options the same as in Figure 39 for both ResNet-50 and ResNet-101 models. Then click start training and wait until they are finished. The training progress will appear as shown in Figure 47.

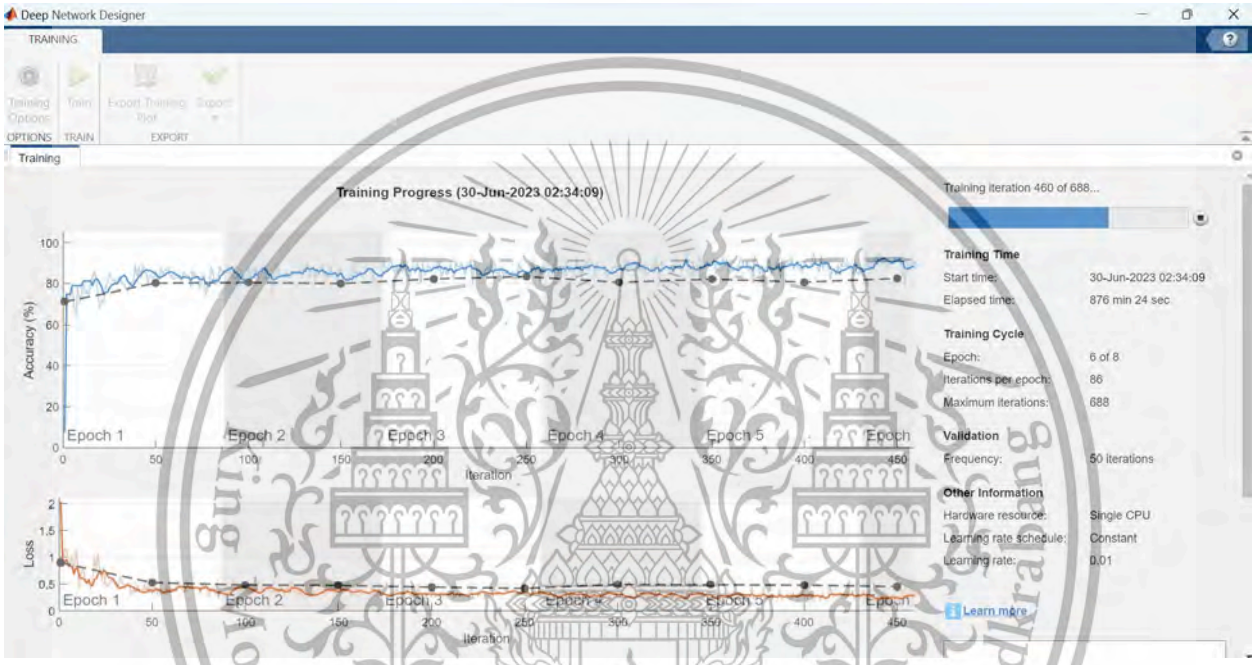

Figure 47: Training Progress for ResNet-101 Model

After finishing training, click "Export" the training networks and trained info struct to the workspace and permanently save them as MATLAB files in the same location as the dataset. Save the training network and trained info struct of the ResNet-50 model as "trainedNetwork_resnet50.mat" and "trainedInfoStruck_resnet50.mat", respectively. And save the training network and trained info struct of the ResNet-101 model as "trainedNetwork_resnet101.mat" and "trainedInfoStruct_resnet101.mat", respectively as shown in Figure 48.

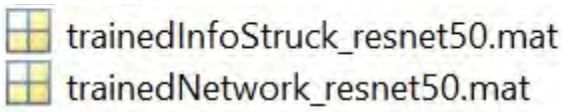

Figure 48: Save Traning Networks

3.4.2.4 Testing

First, select 20 images from the dataset, 5 images per class. Those images must obviously indicate that they belong to specific malaria stages. Create a new folder called "test" and store these 20 test images same location as the dataset. Next, use the same MATLAB code as described in Appendix A to read and classify the test images. Noted that they require some modification if using a different model to test the image. If you want to test an image using the ResNet-50 model, you must ensure that edit lines 3 and 4 according to the ResNet-50 training network files as shown in Figure 49.

```
3    load trainedNetwork_resnet50.mat
4    load trainInfoStruct_resnet50.mat
```

Figure 49: Modify Lines 3 and 4

Run the same code repeatedly until reached all test images and compare the accuracy between the two models.

3.5 Methodologies for Object Detection models

This section describes the different processes for developing object detection models of 3 versions of YOLO in MATLAB, as well as the evaluating methods to find the most appropriate

and accurate models in detecting small objects such as red blood cells from microscopic images. The overall process will start by mentioning all required supporting functions for different YOLO versions, how to load the dataset, how to create an object detection network, data augmentation, preprocess training data, how to train the dataset by object detector, and how to evaluate the detector.

3.5.1 YOLOV3

3.5.1.1 Required Supporting Functions

There are 11 supporting functions that are needed before running other codes of YOLOv3 otherwise it will error after clicking running. These functions include augmentData, preprocessData, and Model Gradients functions that are available in MATLAB Help Center: Object Detection Using YOLO v3 Deep Learning (12).

Create every new function by going to the Editor tab in the MATLAB window, looking for New, dragging down this tab, and selecting "Function". This is the template to create every new function as shown in Figure 50. Moreover, you must ensure that the function name is the same as its file name otherwise it will error and could not apply inside the function. The other way is just to copy and paste a new function entirely from the first line to the bottom in a new editor file.

```
1  function [outputArg1,outputArg2] = untitled(inputArg1,inputArg2)
2  %UNTITLED Summary of this function goes here
3  %   Detailed explanation goes here
4  outputArg1 = inputArg1;
5  outputArg2 = inputArg2;
6  end
```

Figure 50: Template for every function

3.5.1.1.1 Model Gradient Function

This function is used to calculate the total loss and gradient by performing multiple operations. This function will take the detector object, a mini-batch of input data corresponding ground truth boxes as input arguments and returns the gradients of the loss with respect to the learnable parameters in yolov3ObjectDetector, the corresponding mini-batch loss information, and the state of the current batch.

3.5.1.1.2 augmentData Function

This function is used to artificially increase the number and variety of a training dataset by applying various transformations to the existing data. These transformations can include rotations, translations, scaling, flipping, cropping, and more. It will be used to improve network accuracy without actually needing to increase the number of training samples with labels.

3.5.1.1.3 preprocessData Function

This function is apply a series of preprocessing steps or transformations to input data before it is used for training the YOLOv3 object detector model. This function includes resizing the image and bounding boxes to the target size, sanitizing box data, and resizing boxes to the new image size.

3.5.1.2 Develop YOLOv3 model

In this section will describe the procedure to develop the YOLO version 3 model. Overall, they can be separated into 5 main steps, starting with downloading the dataset to

the workspace. Second, do the augmentation and preprocess. Third, create the object detection network. Fourth, set the training option and start training until finished, and save the trained model. Finally, use the trained model to test with the test dataset.

To begin with, download the labeled dataset that we already stored as "malariaDatasetGroundTruth.mat" in the new parameter called "data" using the "load" command. This is our ground truth dataset and rename it as malariaDataset. Then the malaria data is stored in a two-column table, where the first column contains the image file paths and the second column contains malaria bounding boxes. The following command will display the example of information in the first 4 rows of the malaria dataset as can be seen in Figure 51.

```
malariaDataset(1:4,:)
```

imageFilename	ring	trophozoite	schizont	gametocytes	RBC
{'D:\BME32\Project\Slide\M8260001.JPG'}	{31×4 double}	{2×4 double}	{2×4 double}	{[1000 1146 128 53]}	{5×4 double}
{'D:\BME32\Project\Slide\M8260002.JPG'}	{38×4 double}	{3×4 double}	{0×0 double}	{0×0 double}	{5×4 double}
{'D:\BME32\Project\Slide\M8260003.JPG'}	{35×4 double}	{8×4 double}	{[482 558 133 145]}	{0×0 double}	{5×4 double}
{'D:\BME32\Project\Slide\M8260004.JPG'}	{38×4 double}	{6×4 double}	{0×0 double}	{0×0 double}	{5×4 double}

Figure 51: Display dataset information in 4 rows

Then create training, validation, and test sets from the dataset. 60% of the data should be chosen for training, 10% for validation, and the remaining data should be used to test the trained detector. Use the "imageDatastore" function and the "boxLabelDatastore" function to build datastores to load the label and image information for testing and training. After that combine those image datastores and the box label datastores using the "combine" function.

Second, take the training data to undergo data augmentation, to increase the number of images in the dataset artificially and also improve the variety without increasing the actual labeled dataset. It can be done by using the "transform" command corresponding to the

augment data helper function. This command will alter the color of training images, do the horizontal flipping multiple times as well as randomly scale by 10%. To observe the operation of this command, use the “read” function to read the information of the augmented training data and display some sample images using the “figure” command. The example of an augmented image can be seen in Figure 52.

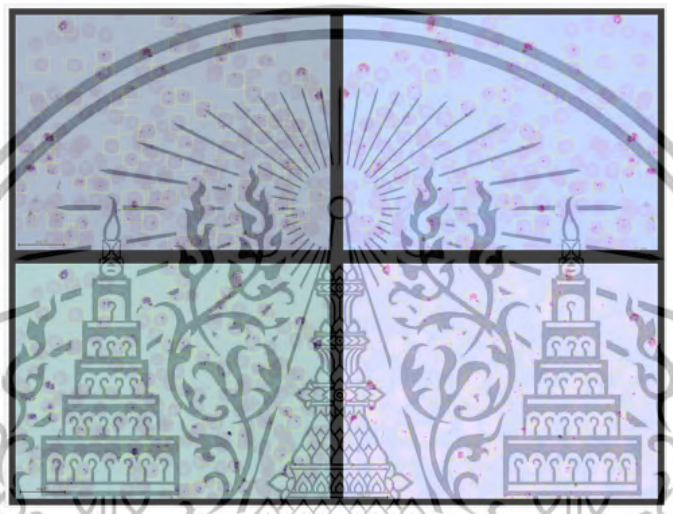

Figure 52: Display the sample image undergoing augmentation

Then, prepare those augmented images for the training process using the preprocess function. It can be done by transforming the augmented image to the preprocess training image with the cooperation of preprocess function. This function will resize the input images (from our dataset) to be the same as the input size of the network but still maintain the image size ratio. This is the reason why we do not have to manually adjust the image size of the dataset. Then we can visualize the preprocess training image using “read” and display it with corresponding bounding boxes as shown in Figure 53.

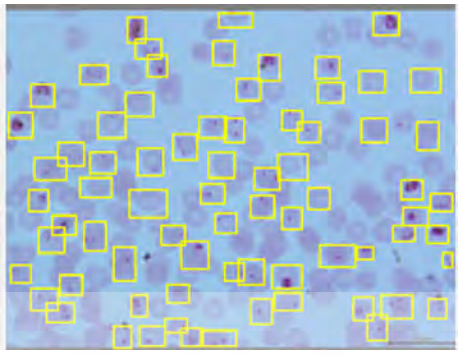

Figure 53: Display preprocess training data with their bounding boxes

Third, create the object detector for the YOLOv3 model. Starting with defining the image input size as 227x227 pixels and using the estimateAnchorBoxes function to estimate the appropriate number of anchor boxes that fit the data by using the intersection-over-union (IoU) distance metric. Since the YOLO version 3 architecture is composed of 2 detection heads, these anchor boxes will be specified for each detection head based on their feature map size. Therefore, the anchors need to be sorted and assign the first set to the first detection head and the other set to the second detection head. In this project, the SqueezeNet network will be used as a feature extraction network to classify each image. So, load this network and specify the class name. Then create the detector using the "yolov3ObjectDetector" function.

Fourth, specify the training option as shown in Appendix B and start training model. After finishing the training, save the trained network as "newtrain.mat" which can recall at any time if you want to detect any object in future work. Finally, test the model to detect infected cells in sample images and calculate the accuracy of detection, learning rate and total loss graphs will be displayed during the training as shown in Figure 54.

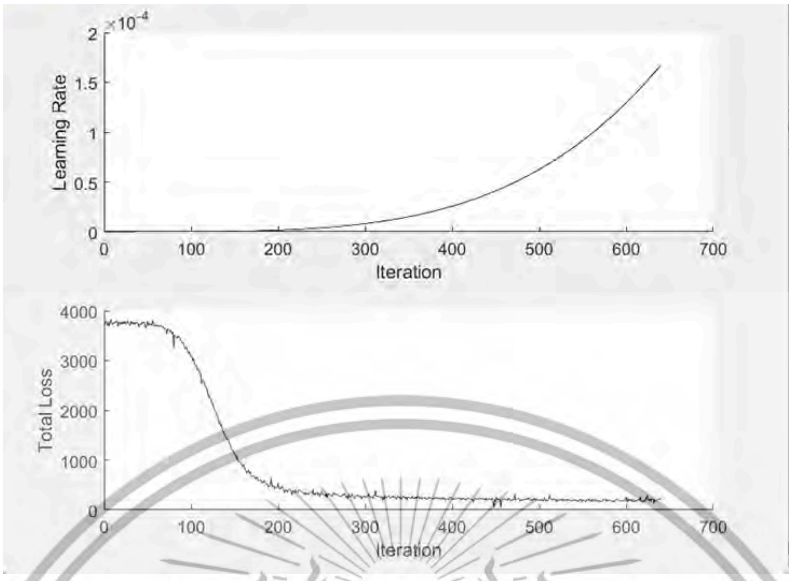

Figure 54: Learning Rate and Total Loss during Training

3.5.2 YOLOV4

3.5.2.1 Required Supporting Functions

There are 2 supporting functions that are required for the YOLOv4 including augmented data and preprocess functions when the code for these functions are shown in Appendix C. These functions are different from the support functions required for YOLOv3. However, their purpose is similar to the augmented data and preprocess functions used in the YOLOv3 model, where the augmented data function aims at increasing the variety of training datasets and the preprocess function aims to prepare the augmented training data to be appropriate before entering the training process.

3.5.2.2 Develop YOLOv4 model

The overall process of developing this model is quite similar to the previous model that is composed of 5 main steps, starting from loading the labeled dataset to the workspace. Second, create an object detector based on pretrained CSPDarkNet-53 network that is training on the COCO dataset. Third, performing data augmentation and preprocess before entering the training process. Fourth, start training and save the trained network. Finally, use the trained model to test with the test dataset.

The first step which includes downloading the dataset, and splitting them into 3 sets; 60% for training, 10% for validation, and 30% for testing and how to create datastores are the same as in the YOLO version 3 model.

Next, create a YOLO v4 detection network from the following steps. Starting with defining the image input size as 416x416 pixels and specifying the class name of all classes that we want to detect. Use the "estimateAnchorBoxes" function to appropriately define the number of anchor boxes similar to what have done in the previous model. The CSPDarkNet-53 network is used as the backbone for YOLO v4 which is composed of 3 detection heads. So, the anchor boxes will be sorted and assigned to each detection head. Then create the YOLO v4 detector using the "yolov4ObjectDetector" function. Third, the code for data augmentation and preprocess data are similar to those in the previous model.

Fourth, set the training options as shown in Appendix D and train the model using the "trainYOLOv4ObjectDetector" function. Then wait until finish training and test the model to detect the test images using the "detect" function and display the results using the "inserObjectAnnotation" and "imshow" functions.

3.5.3 Modify YOLOV4 Detector

This model is modified and developed from the original YOLO v4 detector in the previous section to enhance the ability in detecting small objects to gain higher accuracy. It uses the ResNet-50 as a base network instead of CSPDarkNet-53.

Apart from other section in developing the YOLO v4 model, in the creating detector section need to be modified as the following steps;

- define the base network as "resnet50" and use the "analyzeNetwork" function to visualize the architecture of this network.
- modify the first input layer of the base network by changing the Normalization parameter from "zerocenter" to "none".
- create the YOLO v4 network by extracting the layer graph from the modified base network and store as "lgraph" parameter.
- remove the last output layer, the fully connected network layer.
- replace the image input layer from the original layer to the modified normalization parameter layer.
- create YOLO v4 detector using "dlnetwork" from modified lgraph.
- define the feature extraction layer in the base network to use as the detection heads. There are 2 detection heads, thus the feature extraction layers will be "activation_22_relu" and "activation_40_relu".

Then, specify the class name and the number of anchor boxes similar to the previous models. Followed by creating the YOLO v4 object detector by using the modified base network and detection heads using the "yolov4ObjectDetector" function.

```
dlnetwork with properties:

      Layers: [177×1 nnet.cnn.layer.Layer]
 Connections: [195×2 table]
  Learnables: [212×3 table]
       State: [104×3 table]
  InputNames: {'input_1'}
 OutputNames: {'customOutputConv1'  'customOutputConv2'}
 Initialized: 1
```

Figure 55: Detector Network Properties

Figure 55 illustrates the network properties of the created network which is composed of 177 layers with 195 connections. The name of the input layer (the first layer) is "input_1" and the name of the output layer are "customOutputConv1" and "customOutputConv2".

```
yolov4ObjectDetector with properties:

      Network: [1×1 dlnetwork]
 AnchorBoxes: {2×1 cell}
  ClassNames: {5×1 cell}
   InputSize: [224 224 3]
   ModelName: ''
```

Figure 56: YOLO v4 Object Detector Properties

Figure 56 displays the properties of the detector. It is composed of network called "dlnetwork" which is the network that was created for the YOLO v4 model, two sets of anchor boxes since there are two detection heads, five class names according to different forms of red blood cells that we want to detect, and the input size of this detector is 224x224 pixels.

Finally, analyze the detector to see is there any errors and warnings occur, it should show zero errors and warnings as shown in Figure 57.

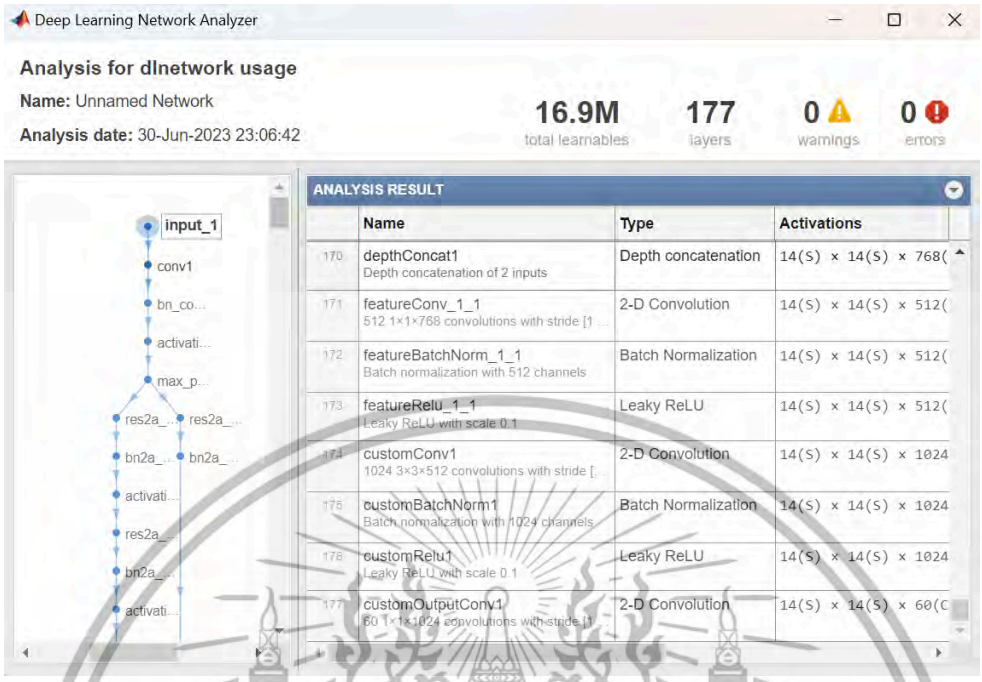

Figure 57: Analyze Detector Network

3.5.4 Detect object from a microscopic image

Use the same test folder as in the classification section to evaluate the performance of 3 models; YOLO v3, YOLO v4, and modified YOLO v4. Each detector will be tested by trying to detect red blood cells both parasitized and uninfected from 20 test images. To detect the object, there are 4 steps starting from loading the trained network and specifying the input size. Second, read the test image using the "imread" function and resize them the same as the setting input size. Third, detect the object in the image using the "detect" function. Lastly, display the results with their annotation and bounding boxes on the figure. These commands are shown in Figure 58.

```
load newtrain_yolo4_edited3.mat
%networkInputSize = [608 608 3];
networkInputSize = [224 224 3];

%%%%%%%% Detect Objects Using YOLO v4 %%%%%%%%
I = imread('M8260019.JPG');
I = imresize(I,networkInputSize(1:2));
[bboxes,scores,labels] = detect(train,I);

ringBoxes = bboxes(labels == "ring", :);
detectRing = insertObjectAnnotation(I,"rectangle",ringBoxes,"ring","Color","cy
figure
imshow(detectRing)
```

Figure 58: Detection command

CHAPTER 4

EXPERIMENTAL RESULT

4.1 Introduction

In this chapter will display the experimental results after training and the testing results of both classification and object detection models. Each model will be tested with the test image as mentioned in chapter 3.

4.2 Results of Classification models

4.2.1 Results of Diagnostic Classification using ResNet50

The trained ResNet-50 network yields 95.50% validation accuracy as can be seen in Figure 40. The total training time is 1040 min 37 sec or approximately 17 hours to train. Then the model will be tested with 20 sample blood smear slide images extracted from the second dataset received from the professor since they all are gained from infected patients, so every image should be parasitized.

Next, test these datasets with the classification MATLAB code as shown in Appendix A and manually observe the result whether the classified image is correct or not. After running the code, a tested result shows the predicted label and calculates the class probabilities. The results of the classified probability of each test image are summarized in Table 1.

Table 1: Diagnostic Classification Summary Results

No.	Tested Image Name	Class, % Probability
1	M8260001.JPG	Parasitized, 93.3%
2	M8260002.JPG	Parasitized, 88.2%
3	M8260003.JPG	Parasitized, 74.3%
4	M8260004.JPG	Parasitized, 79.6%
5	M8260005.JPG	Parasitized, 61.3%
6	M8260006.JPG	Parasitized, 97.3%
7	M8260007.JPG	Parasitized, 89.3%
8	M8260008.JPG	Parasitized, 87.3%
9	M8260009.JPG	Parasitized, 89.6%
10	M8260010.JPG	Parasitized, 96.6%
11	M8260011.JPG	Parasitized, 91.8%
12	M8260012.JPG	Parasitized, 97.4%
13	M8260013.JPG	Parasitized, 90.1%
14	M8260014.JPG	Parasitized, 92.8%

15	M8260015.JPG	Parasitized, 88.8%
16	M8260016.JPG	Parasitized,98.2 %
17	M8260017.JPG	Parasitized, 98.2%
18	M8260018.JPG	Parasitized, 98.1%
19	M8260019.JPG	Parasitized, 98.4%
20	M8260020.JPG	Parasitized, 92.9%

The results from Table 1 can be plotted in line graph as shown in Figure 59. This demonstrated that all of 20 images are classified as parasitized with variety of % probability due to varying number of infected cells being detected in each image, otherwise, each image would be labeled as uninfected if no infected cells are found. Therefore, this sample has no uninfected image (0% for uninfected).

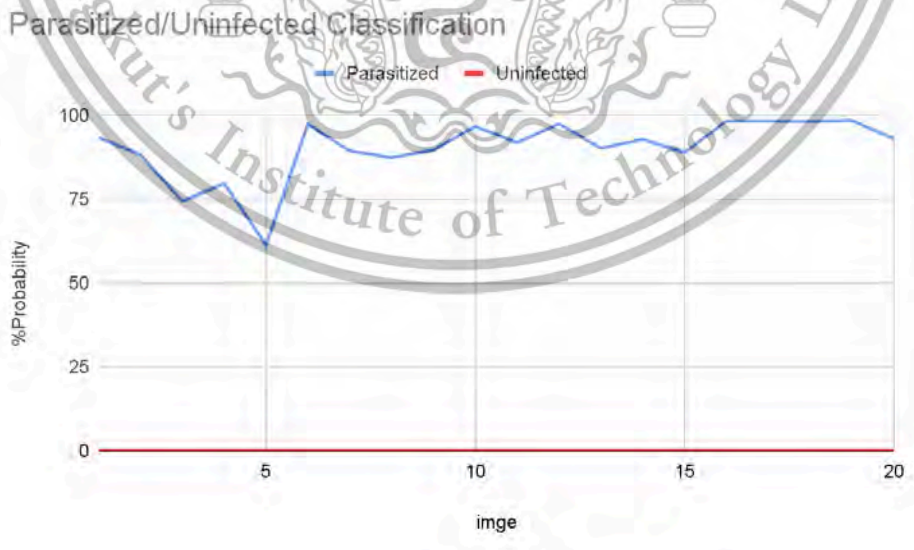

Figure 59: Test Diagnostic Classification Model

4.2.2 Results of Malaria Stages Classification using ResNet-50 and ResNet-101

The trained ResNet-50 Network yields 79.61% validation accuracy as can be seen in Figure 60. The total training time is 726 minutes 15 seconds or approximately 12 hours.

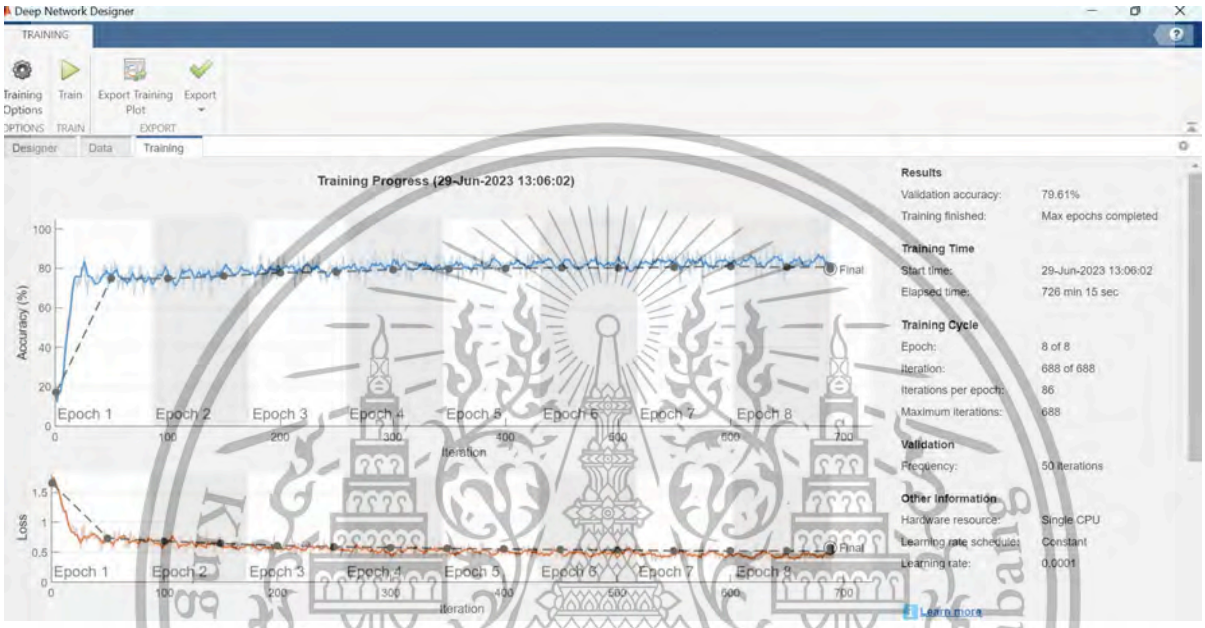

Figure 60: Training Results of ResNet-50 Model

While the trained ResNet-101 Network yields 76.38% validation accuracy as can be seen in Figure 61. The total training time is 1333 minutes 21 seconds or approximately 22 hours.

Figure 61: Training Results of ResNet-101 Model

Both models will be tested with single-cell images extracted from the first dataset. In this dataset, randomly select 5 images form each class and ensure that those images are easily distinguishable so that the classification results should show the same class as we predicted manually. The summary results of ResNet-50 model and ResNet-101 model of test dataset are shown in Table 2 and Table 3, respectively.

Table 2: Malaria stages classification using ResNet-50 network summary

Tested Image Name		Class, % Probability
	RBC1.png	ring,96.6%
	RBC2.png	ring,91.9%
	RBC3.png	ring,90%

1.RBC	RBC4.png	ring,86.6%
	RBC5.png	ring,86.2%
2.ring	ring1.png	ring,99.1%
	ring2.png	ring,65%
	ring3.png	ring,98.6%
	ring4.png	ring,76.6%
	ring5.png	ring,88%
3.trophozoite	trophozoite1.png	schizont,50.8%
	trophozoite2.png	trophozoite,87.2%
	trophozoite3.png	trophozoite,57.1%
	trophozoite4.png	schizont,50.2%
	trophozoite5.png	ring,54.6%
4.schizont	schizont1.png	trophozoite,45.6%
	schizont2.png	schizont,62.2%
	schizont3.png	schizont,69.6%
	schizont4.png	schizont,87.2%

	schizont5.png	schizont,68.7%
Average	11 **correct class** images out of 20 images	55%
	9 **incorrect class** images out of 20 images	45%

These results in the above table can be plotted in a line graph as shown in Figure 62.

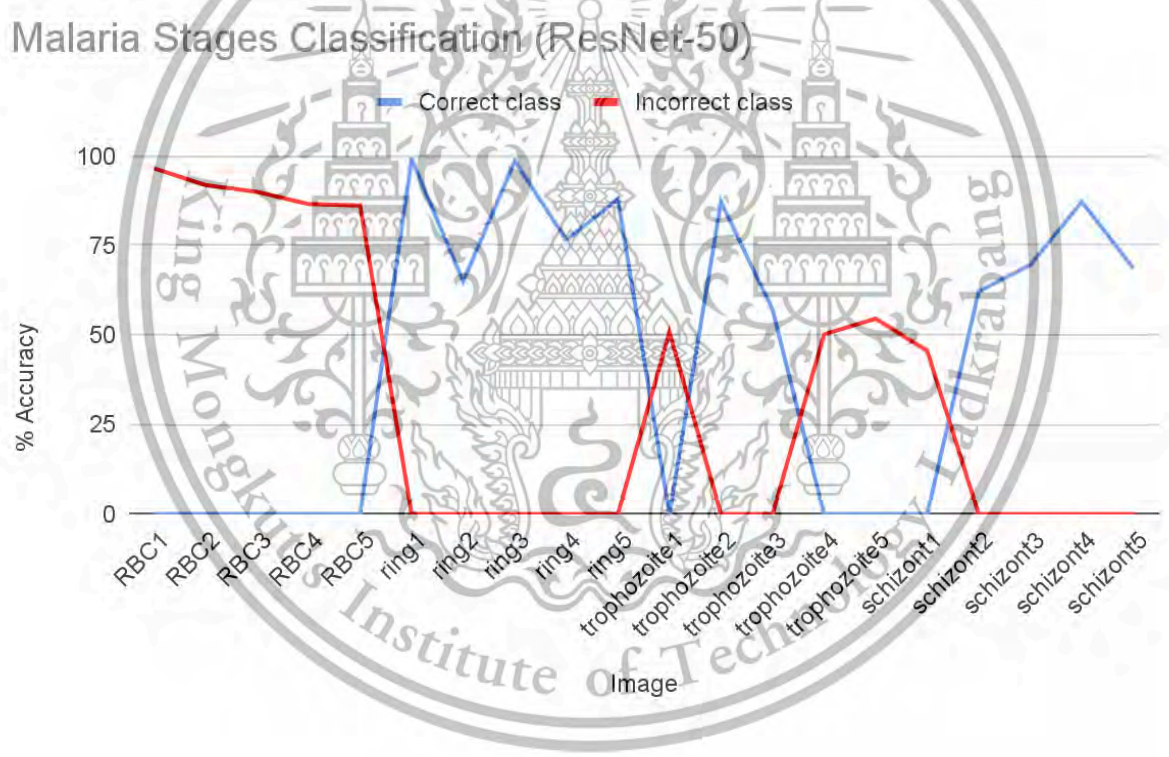

Figure 62: Malaria Stages Classification with ResNet-50 Model

The results from both Table 2 and Figure 62 show that only 55% of the infected cell stages can be accurately categorized by this ResNet-50 model, with the remaining images being classified as non-corresponding stages, which accounted for 45%. This model incorrectly

classifies healthy red blood cell pictures as infected cells having a high probability of being in the ring form rather than being an RBC. This indicates that when normal red blood cells are used, the model incorrectly classifies the stage. While using the same model to detect ring form images, they are highly accurate in classifying correct stages for all 5 images. For trophozoite images, there are only 2 images correctly classified as trophozoite while the other images were labeled as non-corresponding classes, with one image being classed as a ring and the other two as schizont. Lastly, out of five schizont images, this model can accurately classify four of them.

Table 3: Malaria stages classification using ResNet-101 network summary

Tested Image Name		Class, % Accuracy
1.RBC	RBC1.png	RBC,52.6%
	RBC2.png	ring,67.7%
	RBC3.png	RBC,87.3%
	RBC4.png	RBC,70.4%
	RBC5.png	RBC,62.5%
2.ring	ring1.png	ring,93.7%
	ring2.png	ring,72.2%
	ring3.png	ring,99.7%
	ring4.png	ring,97%

	ring5.png	ring,97.4%
3.trophozoite	trophozoite1.png	schizont,81.2%
	trophozoite2.png	trophozoite,90.3%
	trophozoite3.png	schizont,52.3%
	trophozoite4.png	trophozoite,71.7%
	trophozoite5.png	trophozoite,72.5%
4.schizont	schizont1.png	schizont,100%
	schizont2.png	schizont,100%
	schizont3.png	schizont,100%
	schizont4.png	schizont,99.2%
	schizont5.png	schizont,99.9%
Average	17 **correct class** images out of 20 images	85%
	3 **incorrect class** images out of 20 images	15%

These results in the above table can be plotted in a line graph as shown in Figure 63.

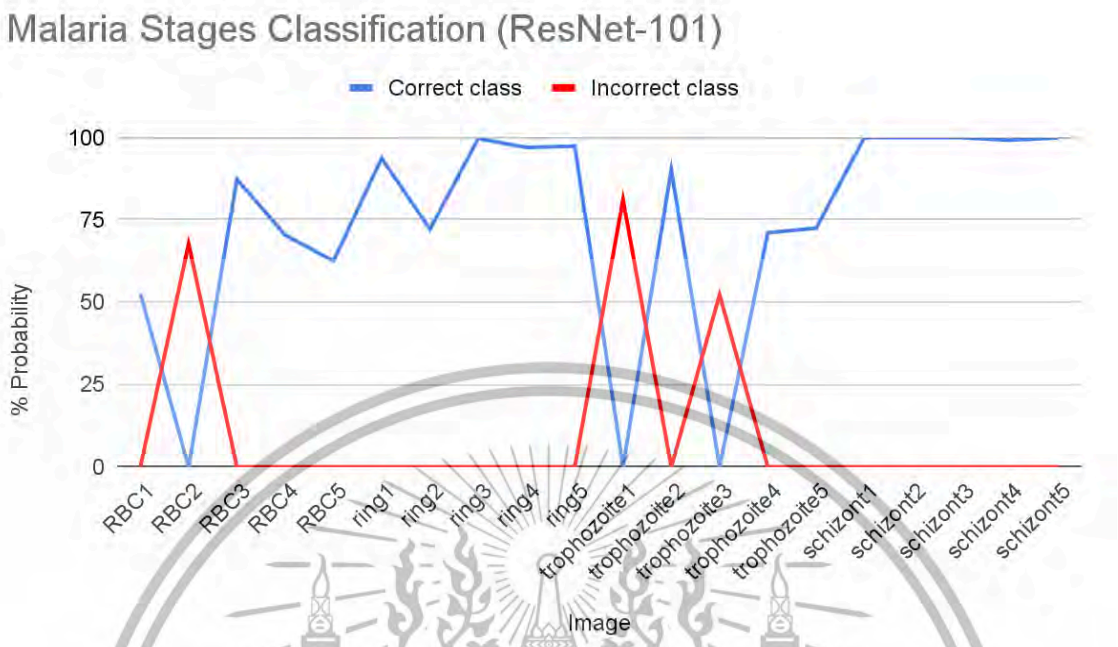

Figure 63: Malaria Stages Classification with ResNet-101 Model

The results from both Table 3 and Figure 63 show that most of images are correctly classify infected stages matching to their exact classes by this ResNet-101 model, which accounted for 85%, while 15% of the remaining images are classified incorrectly. For normal red blood cell images, only one picture is mistakenly labeled as ring form, whereas the majority of RBC images are correctly recognized. For the ring images, this model is highly accurate in classifying correct stages for all 5 images. For trophozoite images, there are only 3 images correctly classified as trophozoite while the other 2 images were labeled as schizont. Finally, this model obtains a remarkable 5 accuracy in classifying the schizont stage in all 5 images.

4.3 Results of Object Detection models

4.3.1 Results of YOLO v3 model

After finishing the training process, use “test” dataset in blood smear slide forms to test the model one by one. Then run the detecting code shown in Appendix B on the dataset. The model will detect all possible cells as well as classify their stages. However, some cells are not able to be detected due to the low ability to detect small objects in this model. A sample of detected result done by this model is shown in Figure 64, some infected cells can be detected and classified as ring form, however, there are more cells that are unable to detected by this model.

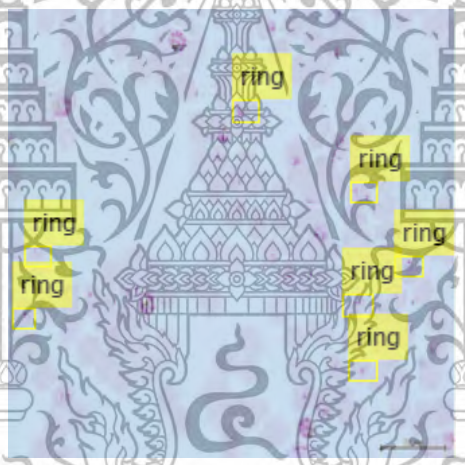

Figure 64: A Detected Image by YOLO v3 Model

Then, assess each image's results by counting the number of infected cells the model recognizes. This includes only those that belong to their actual class with correct bounding boxes, which means it should localize the entire detected cell and exclude those that contain only a portion of the cell. The summary results of using this model on test images are shown in Table 4.

Table 4: Detected Results of YOLO v3 model

No.	Image Name	Quantity of Detection	
		Correct Detection	Incorrect Detection
1	M8260001.JPG	7 R	-
2	M8260002.JPG	11 R	1 S being named as T
3	M8260003.JPG	5 R	1 S being named as T, 1 R incorrect bounding boxes
4	M8260004.JPG	5 R	1 R incorrect bounding boxes
5	M8260005.JPG	2 R	1 T being named as R
6	M8260006.JPG	15 R	1 T being named as R, 1 S being named as T
7	M8260007.JPG	15 R	-
8	M8260008.JPG	12 R	1 T being named as R
9	M8260009.JPG	10 R, 1 T	-
10	M8260010.JPG	9 R, 1 T	1 S incorrect bounding boxes
11	M8260011.JPG	20 R	-
12	M8260012.JPG	23 R	-

13	M8260013.JPG	8 R	2 R incorrect bounding boxes
14	M8260014.JPG	15 R, 1 T	-
15	M82600015.JPG	15 R, 1 T	-
16	M8260016.JPG	20 R	3 R incorrect bounding boxes
17	M8260017.JPG	14 R, 1 S	1 S being named as T 2 R incorrect bounding boxes
18	M8260018.JPG	16 R, 2 T	1 T being named as R
19	M8260019.JPG	17 R	2 T being named as R
20	M8260020.JPG	16 R, 1T	1 T being named as R

The detected results displayed in Table 4 demonstrate that although the YOLO v3 model can accurately localize some infected cells in every image and classify the correct stage, the majority of them have a few errors in their detection, such as incorrect identification or bounding boxes, or there are some infected cells that this model was unable to detect which can be seen in Figure 64.

4.3.2 Results of YOLO v4 models

After fining training, use the same dataset as in the YOLO v3 model to test the model. Run the detection code as shown in Appendix D with all 20 test images. Then manually observe

the correct and incorrect bounding boxes, to check whether the detected cells belong to their actual class. An example of the detected image results of the test dataset is shown in Figure 65.

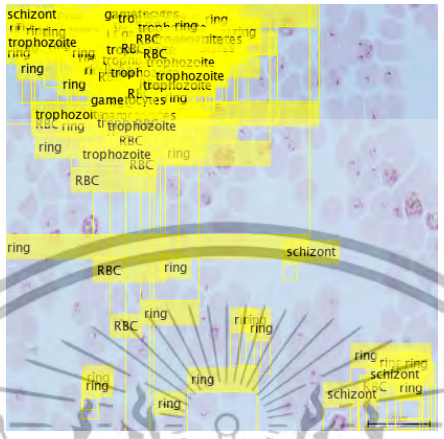

Figure 65: A Detected Image by YOLO v4 Model

In all tested images, the detected results of this model display many inaccurate bounding boxes enclosing the infected cells. Furthermore, the majority of these bounding boxes had clearly distinct sizes, making it impossible to accurately identify just one infected cell. Additionally, even if some cells were one of the malaria parasites, they went undetected. Therefore, the results of using this model can not be summarized as in previous experiments.

4.3.3 Results of Modified YOLO v4 models

Use the training network of the modified YOLO v4 Object Detector to test the same dataset as mentioned in the previous section. Apply the model to detect infected cells in 20 blood smear slide images. A sample of applying this modified model on a tested image is shown in Figure 66, some infected cells can be identified and categorized as being in the ring form; however, this model was unable to identify other infected cells shown in the same image that were in the ring, trophozoite, or schizont forms.

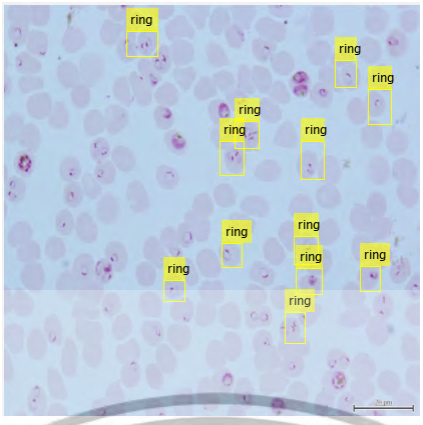

Figure 66: A Detected Image by Modified YOLO v4 Model

Since this result shows a few similar bounding box sizes on infected cells. As a result, we may report the model's findings using the same techniques as those described in 4.3.1: Results of the YOLO v3 model as shown in Table 5.

Table 5: Detected Results of Modify YOLO v4 model

No.	Image Name	Quantity of Detection	
		Correct Detection	Incorrect Detction
1	M8260001.JPG	2 R	-
2	M8260002.JPG	9 R 1T	1 S being named as T
3	M8260003.JPG	3 R	-
4	M8260004.JPG	2 R	-
5	M8260005.JPG	2 R	-

6	M8260006.JPG	5 R	-
7	M8260007.JPG	2 R	2 T being named as R
8	M8260008.JPG	12 R	1 T being named as R
9	M8260009.JPG	3R	-
10	M8260010.JPG	4 R	1 S being named as T
11	M8260011.JPG	11 R	1 T being named as R
12	M8260012.JPG	8 R	-
13	M8260013.JPG	3 R	-
14	M8260014.JPG	8 R, 1 T	-
15	M82600015.JPG	10 R	-
16	M8260016.JPG	9 R	1 S being named as T 1 R incorrect bounding boxes
17	M8260017.JPG	3 R, 2 T	1 S incorrect bounding boxes
18	M8260018.JPG	5 R	2 R incorrect bounding boxes
19	M8260019.JPG	1 R	1 T being named as R
20	M8260020.JPG	5 R, 1 T	1 R incorrect bounding boxes

Given that the modified YOLO v4 model's detected results were derived from the same test image group as the YOLO v3 model's detected results, the two models supposed to produce identical findings. However, we can observe that the quantity of both corrected and incorrect detection in each image as shown in Table 5 are smaller compared with results shown in Table 4. This implies that, in comparison to the YOLO v3 model, the modified YOLO v4 model provides lower detection numbers but fewer incorrect stage classification errors.

CHAPTER 5

DISCUSSION AND CONCLUSION

5.1 Discussion

The outcomes of this research are composed of 2 major topics including developing classification networks to classify plasmodium falciparum malaria blood smear slide images as a screener for diagnosis as well as classify malaria parasite stages and developing object detection models to detect and classify stages of the detected cells.

From the experiment, the diagnostic model that uses the ResNet-50 as a base network for classifying blood smear slide images of infected and non-infected patients shows an extremely high validation accuracy of 95.5%. It can be used to classify all 20 test images with 100% confirming that they all are parasitized which is also correct since the test dataset is extracted from the source affirming that they all are parasitized. This high accuracy must be due to an extremely large number of datasets for both parasitized and uninfected datasets and similar proportions among them. This is the reason why when applying the same ResNet-50 network to classify malaria stages did not give as high validation accuracy as in the diagnostic model. The ResNet-50 network that was used to train with 4 different classes of malaria stages only gives 79.61% validation accuracy compared to one that uses ResNet-101 to do the same task, unexpectedly, showing minimal lower validation accuracy, 76.38%. Although, the ResNet-101 model results in lower validation accuracy, it shows better performance in classifying the same test images as in ResNet-50. Correctly classifying 17 test images out of 20 indicates that ResNet-101 can accurately classify images for 85%. While there are fewer correct classifications in the ResNet-50 model which only correctly classifies 11 test images out of 20, this means only

55% accuracy. This should be due to more layers in the ResNet-101 network that enhance the classification accuracy.

Moving on to the 3 object detection models including YOLO v3, YOLO v4, and Modified YOLO v4. All of them are able to detect the ring form of parasite more than other classes, this may be due to the largest number of training datasets compared to other classes, allowing the model to learn more from large image numbers and resulting in more sensitivity to detect ring form or it may because of there is a much higher proportion of ring parasite in every blood smear slide image. The YOLO v3 model has some ability to detect infected cells and classify their stages precisely. However, not all parasite cells are detected, and the bounding boxes of the result images do not perfectly detect the whole infected red blood cell but rather only detect the parasite contents in the red blood cell. The second model, YOLO v4, shows the worst performance since it cannot detect parasite cells accurately as well as their random size of bounding boxes which is hard to observe what is being detected. The last model was done by modifying the detector part of the old version resulting in the Modified YOLO v4 model. This model shows some performance in detecting objects but is lower than the first model. It can detect infected cells correctly, especially the ring form, with the bounding boxes that can cover the whole red blood cell and not only cover the parasite contents.

5.2 Conclusion

The diagnostic classification model can be used as a screener for malaria diagnosis because it gives relatively high accuracy, 95.5%. Moreover, classification models to classify malaria stages based on the ResNet-101 network is more appropriate when compared to the ResNet-50 network, adding more other classes apart from the ring will increase the validation

accuracy of the model. Additionally, all object detection models are not ready for practical use because there is no model that can detect all infected cell images correctly.

REFERENCES

1. Roser M, Ritchie H. Malaria [Internet]. Our World in Data. 2019. Available from: https://ourworldindata.org/malaria
2. American Society of Hematology. Hematology Glossary [Internet]. www.hematology.org. Available from: https://www.hematology.org/education/patients/blood-basics#:~:text=Blood%20is%20a%20specialized%20body
3. Sharma R, Sharma S. Physiology, Blood Volume [Internet]. PubMed. Treasure Island (FL): StatPearls Publishing; 2021. Available from: https://pubmed.ncbi.nlm.nih.gov/30252333/
4. Barshtein G, Bergelson L, Gratton E, Yedgar S. Human Red Blood Cell Shape and Volume are Changed by Physiological Levels of Hydrostatic Pressure. Journal of Basic and Clinical Physiology and Pharmacology. 1996 Jan;7(4).
5. Barshtein G, Pajic-Lijakovic I, Gural A. Deformability of Stored Red Blood Cells. Frontiers in Physiology. 2021 Sep 22;12.
6. Welcome To State Health Society, Bihar, Thalassemia & Hemophilia web Protal. [Internet]. thalassemiaregistry.bihar.gov.in. Available from: http://thalassemiaregistry.bihar.gov.in/AboutSCA.aspx
7. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/granulocyte# [Internet]. www.cancer.gov. 2011 [cited 2023 Apr 28]. Available from: https://www.cancer.gov/publications/dictionaries/cancer-terms/def/granulocyte#

8. Tigner A, Ibrahim SA, Murray I. Histology, White Blood Cell [Internet]. PubMed. Treasure Island (FL): StatPearls Publishing; 2020. Available from: https://www.ncbi.nlm.nih.gov/books/NBK563148/
9. Prevention CC for DC and. CDC - Malaria - FAQs [Internet]. www.cdc.gov. 2020. Available from: https://www.cdc.gov/malaria/about/faqs.html#:~:text=Symptoms%20of%20malaria%20include%20fever
10. NHS. Overview - Malaria [Internet]. NHS. 2022. Available from: https://www.nhs.uk/conditions/Malaria/
11. World Health Organization. Malaria [Internet]. Who.int. World Health Organization: WHO; 2023. Available from: https://www.who.int/news-room/fact-sheets/detail/malaria
12. Object Detection Using YOLO v3 Deep Learning - MATLAB & Simulink [Internet]. www.mathworks.com. Available from: https://www.mathworks.com/help/vision/ug/object-detection-using-yolo-v3-deep-learning.html
13. Thailand [Internet]. PVIVAX. Available from: https://www.vivaxmalaria.org/thailand
14. Centers for Disease Control and Prevention. CDC - Malaria - About Malaria - Biology [Internet]. CDC. 2020. Available from: https://www.cdc.gov/malaria/about/biology/index.html
15. Parasitemia - an overview | ScienceDirect Topics [Internet]. www.sciencedirect.com. Available from: https://www.sciencedirect.com/topics/medicine-and-dentistry/parasitemia
16. Barber BE, William T, Grigg MJ, Menon J, Auburn S, Marfurt J, et al. A Prospective Comparative Study of Knowlesi, Falciparum, and Vivax Malaria in Sabah, Malaysia:

High Proportion With Severe Disease From Plasmodium Knowlesi and Plasmodium Vivax But No Mortality With Early Referral and Artesunate Therapy. Clinical Infectious Diseases. 2012 Oct 19;56(3):383–97.

17. Manser M, Olufsen C, Andrews N, Chiodini PL. Estimating the parasitaemia of Plasmodium falciparum: experience from a national EQA scheme. Malaria Journal. 2013;12(1):428.
18. Getaneh F, Zeleke AJ, Lemma W, Tegegne Y. Malaria Parasitemia in Febrile Patients Mono- and Coinfected with Soil-Transmitted Helminthiasis Attending Sanja Hospital, Northwest Ethiopia. Journal of Parasitology Research. 2020 Feb 4;2020:1–7.
19. Daily JP, Minuti A, Khan N. Diagnosis, Treatment, and Prevention of Malaria in the US. JAMA. 2022 Aug 2;328(5):460.
20. Bloland PB, Williams HA, Population NRC (US) C on, Program on Forced Migration and Health at the Mailman School of Public Health CU. Curative Services: Malaria Therapy and Case Management [Internet]. www.ncbi.nlm.nih.gov. National Academies Press (US); 2002. Available from: https://www.ncbi.nlm.nih.gov/books/NBK221170/
21. Themes UFO. 51 Apicomplexa and Microsporidia [Internet]. Basicmedical Key. 2017. Available from: https://basicmedicalkey.com/51-apicomplexa-and-microsporidia/
22. IBM. What is Deep Learning? | IBM [Internet]. www.ibm.com. 2023. Available from: https://www.ibm.com/topics/deep-learning
23. IBM. What is Machine Learning? [Internet]. www.ibm.com. IBM; 2023. Available from: https://www.ibm.com/topics/machine-learning
24. Want to know how Deep Learning works? Here's a quick guide for everyone. [Internet]. freeCodeCamp.org. 2017. Available from:

https://www.freecodecamp.org/news/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076/

25. Machine Learning [Internet]. www.mathworks.com. Available from: https://www.mathworks.com/discovery/machine-learning.html#:~:text=Machine%20learning%20uses%20two%20types
26. Feedforward vs Backpropagation ANN [Internet]. www.linkedin.com. Available from: https://www.linkedin.com/pulse/feedforward-vs-backpropagation-ann-saffronedge1
27. Shah S. Convolutional Neural Network: An Overview [Internet]. Analytics Vidhya. 2022. Available from: https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/
28. Dertat A. Applied Deep Learning - Part 4: Convolutional Neural Networks [Internet]. Towards Data Science. Towards Data Science; 2017. Available from: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
29. Rastogi A. ResNet50 [Internet]. Medium. 2022. Available from: https://blog.devgenius.io/resnet50-6b42934db431
30. Malaria Cell Images Dataset [Internet]. www.kaggle.com. Available from: https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria
31. LHNCBC Full Download List [Internet]. lhncbc.nlm.nih.gov. Available from: https://lhncbc.nlm.nih.gov/LHC-downloads/downloads.html#malaria-datasets

APPENDICES

APPENDIX A

CODE FOR DIAGNOSTIC AND MALARIA STAGES CLASSIFICATION

```
1   %test parasitized and uninfected output
2   clear
3   load trainedNetwork_1.mat
4   load trainInfoStruct_1.mat
5   g= imread('test\M8260001.JPG');
6   f= imresize(g, [224,224]);
7   %Classify the test image using the trained network.
8   [YPred,probs] = classify(trainedNetwork_1,f);
9   imshow(f);
10  label = YPred;
11  title(string(label) + ", " + num2str(100*max(probs),3) + "%");
```

APPENDIX B

YOLO VERSION 3 MODEL

```
clear

close all

doTraining = true;

%%%%%%%% Load Data %%%%%%%%

data = load('malariaDatasetGroundTruth.mat');

malariaDataset = data.gTruth;

% Display first few rows of the data set.

malariaDataset(1:4,:)

% Split the data set into a training set for training the network,

% and a test set for evaluating the network. Use 60% of the data for

% training set and the rest for the test set.

rng(0);

shuffledIndices = randperm(height(malariaDataset));

idx = floor(0.6 * length(shuffledIndices));

trainingDataTbl = malariaDataset(shuffledIndices(1:idx), :);

testDataTbl = malariaDataset(shuffledIndices(idx+1:end), :);

% Create an image datastore for loading the images.

imdsTrain = imageDatastore(trainingDataTbl.imageFilename);

imdsTest = imageDatastore(testDataTbl.imageFilename);
```

```
% Create a datastore for the ground truth bounding boxes.

bldsTrain = boxLabelDatastore(trainingDataTbl(:, 2:end));

bldsTest = boxLabelDatastore(testDataTbl(:, 2:end));

% Combine the image and box label datastores.

trainingData = combine(imdsTrain, bldsTrain);

testData = combine(imdsTest, bldsTest);



%%%%%%%% Data Augmentation %%%%%%%%

augmentedTrainingData = transform(trainingData, @augmentData3);

% Read the same image four times and display the augmented training data.

% Visualize the augmented images.

augmentedData = cell(4,1);

for k = 1:4

data = read(augmentedTrainingData);

augmentedData{k} = insertShape(data{1,1}, 'Rectangle', data{1,2});

reset(augmentedTrainingData);

end

figure

montage(augmentedData, 'BorderSize', 10)

%%%%%%%% Define YOLO v3 Object Detector %%%%%%%%

networkInputSize = [227 227 3];
```

```
rng(0)

trainingDataForEstimation = transform(trainingData,
@(data)preprocessData3(data, networkInputSize));

numAnchors = 6;

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

area = anchors(:, 1).*anchors(:, 2);

[~, idx] = sort(area, 'descend');

anchors = anchors(idx, :);

anchorBoxes = {anchors(1:3,:)

anchors(4:6,:)

};

baseNetwork = squeezenet;

classNames = trainingDataTbl.Properties.VariableNames(2:end);

yolov3Detector = yolov3ObjectDetector(baseNetwork, classNames, anchorBoxes,
'DetectionNetworkSource', {'fire9-concat', 'fire5-concat'}, InputSize =
networkInputSize);

%%%%%%% Preprocess Training Data %%%%%%%

preprocessedTrainingData = transform(augmentedTrainingData,
@(data)preprocess(yolov3Detector, data));

% Read the preprocessed training data.

data = read(preprocessedTrainingData);

% Display the image with the bounding boxes.
```

```
I = data{1,1};

bbox = data{1,2};

annotatedImage = insertShape(I, 'Rectangle', bbox);

annotatedImage = imresize(annotatedImage,2);

figure

imshow(annotatedImage)

% Reset the datastore.

reset(preprocessedTrainingData);

% Specify these training options.

numEpochs = 80;

miniBatchSize = 8;

learningRate = 0.001;

warmupPeriod = 1000;

l2Regularization = 0.0005;

penaltyThreshold = 0.5;

velocity = [];

%%%%%%%% Train Model %%%%%%%%

if canUseParallelPool

dispatchInBackground = true;

else

dispatchInBackground = false;
```

```
end

mbqTrain = minibatchqueue(preprocessedTrainingData, 2,...

"MiniBatchSize", miniBatchSize,...

"MiniBatchFcn", @(images, boxes, labels) createBatchData(images, boxes, labels,

classNames), ...

"MiniBatchFormat", ["SSCB", ""],...

"DispatchInBackground", dispatchInBackground,...

"OutputCast", ["", "double"]);

if doTraining

% Create subplots for the learning rate and mini-batch loss.

fig = figure;

[lossPlotter, learningRatePlotter] = configureTrainingProgressPlotter(fig);

iteration = 0;

% Custom training loop.

for epoch = 1:numEpochs

reset(mbqTrain);

shuffle(mbqTrain);

while(hasdata(mbqTrain))

iteration = iteration + 1;

[XTrain, YTrain] = next(mbqTrain);

% Evaluate the model gradients and loss using dlfeval and the

% modelGradients function.
```

```
[gradients, state, lossInfo] = dlfeval(@modelGradients, yolov3Detector, XTrain,
YTrain, penaltyThreshold);

% Apply L2 regularization.

gradients = dlupdate(@(g,w) g + l2Regularization*w, gradients,
yolov3Detector.Learnables);

% Determine the current learning rate value.

currentLR = piecewiseLearningRateWithWarmup(iteration, epoch, learningRate,
warmupPeriod, numEpochs);

% Update the detector learnable parameters using the SGDM optimizer.

[yolov3Detector.Learnables, velocity] = sgdmupdate(yolov3Detector.Learnables,
gradients, velocity, currentLR);

% Update the state parameters of dlnetwork.

yolov3Detector.State = state;

% Display progress.

displayLossInfo(epoch, iteration, currentLR, lossInfo);

% Update training plot with new points.

updatePlots(lossPlotter, learningRatePlotter, iteration, currentLR,
lossInfo.totalLoss);

end

end

else

yolov3Detector = preTrainedDetector;

end
```

```
train = yolov3Detector;

save('newtrain.mat','train')

load newtrain.mat

networkInputSize = [227 227 3];

%%%%%%%% Detect Objects Using YOLO v3 %%%%%%%%

I = imread('M8260099.JPG');

I = imresize(I,networkInputSize(1:2));

[bboxes,scores,labels] = detect(train,I);

% Display the detections on image.

I = insertObjectAnnotation(I,'rectangle',bboxes,labels);

% change 'labels' to 'scores' to display the probalbility

figure

imshow(I)
```

APPENDIX C

SUPPORTING FUNCTIONS FOR YOLO VERSION 4 MODEL

```
function data = augmentData4(A)

% Apply random horizontal flipping, and random X/Y scaling. Boxes that get

% scaled outside the bounds are clipped if the overlap is above 0.25. Also,

% jitter image color.

data = cell(size(A));

for ii = 1:size(A,1)

I = A{ii,1};

bboxes = A{ii,2};

labels = A{ii,3};

sz = size(I);

if numel(sz) == 3 && sz(3) == 3

I = jitterColorHSV(I,...

contrast=0.0,...

Hue=0.1,...

Saturation=0.2,...

Brightness=0.2);

end

% Randomly flip image.

tform = randomAffine2d(XReflection=true,Scale=[1 1.1]);
```

```
rout = affineOutputView(sz,tform,BoundsStyle="centerOutput");

I = imwarp(I,tform,OutputView=rout);

% Apply same transform to boxes.

[bboxes,indices] = bboxwarp(bboxes,tform,rout,OverlapThreshold=0.25);

labels = labels(indices);

% Return original data only when all boxes are removed by warping.

if isempty(indices)

data(ii,:) = A(ii,:);

else

data(ii,:) = {I,bboxes,labels};

end

end

end



function data = augmentData4(A)

% Apply random horizontal flipping, and random X/Y scaling. Boxes that get

% scaled outside the bounds are clipped if the overlap is above 0.25. Also,

% jitter image color.

data = cell(size(A));

for ii = 1:size(A,1)

I = A{ii,1};
```

```
bboxes = A{ii,2};

labels = A{ii,3};

sz = size(I);

if numel(sz) == 3 && sz(3) == 3

I = jitterColorHSV(I,...

contrast=0.0,...

Hue=0.1,...

Saturation=0.2,...

Brightness=0.2);

end

% Randomly flip image.

tform = randomAffine2d(XReflection=true,Scale=[1 1.1]);

rout = affineOutputView(sz,tform,BoundsStyle="centerOutput");

I = imwarp(I,tform,OutputView=rout);

% Apply same transform to boxes.

[bboxes,indices] = bboxwarp(bboxes,tform,rout,OverlapThreshold=0.25);

labels = labels(indices);

% Return original data only when all boxes are removed by warping.

if isempty(indices)

data(ii,:) = A(ii,:);

else
```

```
data(ii,:) = {I,bboxes,labels};

end

end

end
```

APPENDIX D

YOLO VERSION 4 MODEL

```
clear

close all

doTraining = true;

%%%%%%%% Load Data %%%%%%%%

data = load('malariaDatasetGroundTruth2.mat');

malariaDataset = data.gTruth;

% Display first few rows of the data set.

malariaDataset(1:4,:)

% Add the full path to the local vehicle data folder.

%malariaDataset.imageFilename = fullfile(pwd, malariaDataset.imageFilename);

% Split the dataset into training, validation, and test sets.

% Select 60% of the data for training, 10% for validation,

% and the rest for testing the trained detector.

rng("default");

shuffledIndices = randperm(height(malariaDataset));

idx = floor(0.6 * length(shuffledIndices) );

trainingIdx = 1:idx;

trainingDataTbl = malariaDataset(shuffledIndices(trainingIdx),:);

validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
```

```
validationDataTbl = malariaDataset(shuffledIndices(validationIdx),:);

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

testDataTbl = malariaDataset(shuffledIndices(testIdx),:);

% Use imageDatastore to create datastores for loading the image.

% and boxLabelDatastore to label data during training and evaluation.

imdsTrain = imageDatastore(trainingDataTbl{:,"imageFilename"});

bldsTrain = boxLabelDatastore(trainingDataTbl(:, 2:end));

imdsValidation = imageDatastore(validationDataTbl{:,"imageFilename"});

bldsValidation = boxLabelDatastore(validationDataTbl(:, 2:end));

imdsTest = imageDatastore(testDataTbl{:,"imageFilename"});

bldsTest = boxLabelDatastore(testDataTbl(:, 2:end));

% Combine image and box label datastores.

trainingData = combine(imdsTrain,bldsTrain);

validationData = combine(imdsValidation,bldsValidation);

testData = combine(imdsTest,bldsTest);

% validateInputData(trainingData);

% validateInputData(validationData);

% validateInputData(testData);

% Display one of the training images and box labels.

data = read(trainingData);

I = data{1};
```

```
bbox = data{2};

annotatedImage = insertShape(I,"Rectangle",bbox);

annotatedImage = imresize(annotatedImage,2);

figure

imshow(annotatedImage)

reset(trainingData);

%%%%%%%% Create a YOLO v4 Object Detector Network %%%%%%%

% Specify the network input size to be used for training.

%inputSize = [608 608 3];

inputSize = [416 416 3];

% Specify the name of the object class to detect. (many classes)

className = trainingDataTbl.Properties.VariableNames(2:end);

rng("default")

trainingDataForEstimation =

transform(trainingData,@(data)preprocessData4(data,inputSize));

numAnchors = 9;

[anchors,meanIoU] = estimateAnchorBoxes(trainingDataForEstimation,numAnchors);

area = anchors(:, 1).*anchors(:,2);

[~,idx] = sort(area,"descend");

anchors = anchors(idx,:);

anchorBoxes = {anchors(1:3,:)

anchors(4:6,:)
```

```
anchors(7:9,:)

};

detector =
yolov4ObjectDetector("csp-darknet53-coco",className,anchorBoxes,InputSize=input
Size);

%%%%%%%% Data Augmentation %%%%%%%%

augmentedTrainingData = transform(trainingData,@augmentData4);

% Read and display samples of augmented training data.

augmentedData = cell(4,1);

for k = 1:4

data = read(augmentedTrainingData);

augmentedData{k} = insertShape(data{1},"rectangle",data{2});

reset(augmentedTrainingData);

end

figure

montage(augmentedData,BorderSize=10)

%%%%%%% Specify these training options %%%%%%%%

options = trainingOptions("adam",...

GradientDecayFactor=0.9,...

SquaredGradientDecayFactor=0.999,...

InitialLearnRate=0.000001,...

LearnRateSchedule="none",...
```

```
MiniBatchSize=4,...

L2Regularization=0.0005,...

MaxEpochs=70,...

BatchNormalizationStatistics="moving",...

DispatchInBackground=true,...

ResetInputNormalization=false,...

Shuffle="every-epoch",...

VerboseFrequency=20,...

ValidationFrequency=1000,...

CheckpointPath=tempdir,...

ValidationData=validationData);

%%%%%%%% Train Model %%%%%%%%

if doTraining

% Train the YOLO v4 detector.

[detector,info] =

trainYOLOv4ObjectDetector(augmentedTrainingData,detector,options);

else

% Load pretrained detector for the example.

detector = downloadPretrainedYOLOv4Detector();

end

%%%%%%%% Evaluate Model %%%%%%%%

% Run the detector on all the test images.
```

```
detectionResults = detect(detector,testData);

% Evaluate the object detector using average precision metric.

[ap,recall,precision] = evaluateDetectionPrecision(detectionResults,testData);

% Plot precision-recall curve.

figure

plot(recall,precision)

xlabel("Recall")

ylabel("Precision")

grid on

title(sprintf("Average Precision = %.2f",ap))

% save train

train = detector;

save('newtrain_yolo4_2.mat','train')

load newtrain_yolo4_2.mat

%networkInputSize = [608 608 3];

networkInputSize = [416 416 3];

%%%%%%%% Detect Objects Using YOLO v4 %%%%%%%%

I = imread('M8260099.JPG');

I = imresize(I,networkInputSize(1:2));

[bboxes,scores,labels] = detect(train,I);

% Display the results.
```

```
I = insertObjectAnnotation(I,"rectangle",bboxes,labels);

% last term 'scores' used to display the probability

% and 'labels' to display class

figure

imshow(I)
```

APPENDIX E

MODIFIED YOLO VERSION 4 MODEL

```
clear

close all

%%%%%%%% Load Data %%%%%%%%

data = load('malariaDatasetGroundTruth.mat');

malariaDataset = data.gTruth;

% Display first few rows of the data set.

malariaDataset(1:4,:)

% Split the dataset into training, validation, and test sets.

% Select 60% of the data for training, 10% for validation,

% and the rest for testing the trained detector.

rng("default");

shuffledIndices = randperm(height(malariaDataset));

idx = floor(0.6 * length(shuffledIndices) );

trainingIdx = 1:idx;

trainingDataTbl = malariaDataset(shuffledIndices(trainingIdx),:);

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

validationDataTbl = malariaDataset(shuffledIndices(validationIdx),:);

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

testDataTbl = malariaDataset(shuffledIndices(testIdx),:);
```

```
% Use imageDatastore to create datastores for loading the image.
% and boxLabelDatastore to label data during training and evaluation.
imdsTrain = imageDatastore(trainingDataTbl{:,"imageFilename"});
bldsTrain = boxLabelDatastore(trainingDataTbl(:, 2:end));
imdsValidation = imageDatastore(validationDataTbl{:,"imageFilename"});
bldsValidation = boxLabelDatastore(validationDataTbl(:, 2:end));
imdsTest = imageDatastore(testDataTbl{:,"imageFilename"});
bldsTest = boxLabelDatastore(testDataTbl(:, 2:end));
% Combine image and box label datastores.
trainingData = combine(imdsTrain,bldsTrain);
validationData = combine(imdsValidation,bldsValidation);
testData = combine(imdsTest,bldsTest);
% Display one of the training images and box labels.
data = read(trainingData);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I,"Rectangle",bbox);
annotatedImage = imresize(annotatedImage,2);
figure
imshow(annotatedImage)
reset(trainingData);
```

```
%%%%%%%% create new detector %%%%%%%%%

inputSize = [224 224 3];

basenet = resnet50;

analyzeNetwork(basenet);

basenet.Layers(1)

imageSize = basenet.Layers(1).InputSize;

layerName = basenet.Layers(1).Name;

newinputLayer =
imageInputLayer(imageSize,'Normalization','none','Name',layerName);

lgraph = layerGraph(basenet);

lgraph = removeLayers(lgraph,'ClassificationLayer_fc1000');

lgraph = replaceLayer(lgraph,layerName,newinputLayer);

dlnet = dlnetwork(lgraph);

featureExtractionLayers = ["activation_22_relu","activation_40_relu"];

% Specify the name of the object class to detect. (many classes)

className = trainingDataTbl.Properties.VariableNames(2:end);

disp(className)

rng("default")

trainingDataForEstimation =
transform(trainingData,@(data)preprocessData4(data,inputSize));

numAnchors = 12;

[anchors,meanIoU] = estimateAnchorBoxes(trainingDataForEstimation,numAnchors);
```

```
anchors

meanIoU

%anchorBoxes = num2cell(anchors)

area = anchors(:, 1).*anchors(:,2);

[~,idx] = sort(area,"descend");

anchors = anchors(idx,:);

anchorBoxes = {anchors(1:6,:)

anchors(7:12,:)

% anchors(9:12,:)

};

anchorBoxes

% anchorBoxes = {[122,177;223,84;80,94];...

% [111,38;33,47;37,18]};

detector =
yolov4ObjectDetector(dlnet,className,anchorBoxes,DetectionNetworkSource=feature
ExtractionLayers);

disp(detector)

analyzeNetwork(detector.Network)

detector.Network

%%%%%%%% Data Augmentation %%%%%%%%

augmentedTrainingData = transform(trainingData,@augmentData4);

% Read and display samples of augmented training data.
```

```
augmentedData = cell(4,1);

for k = 1:4

data = read(augmentedTrainingData);

augmentedData{k} = insertShape(data{1},"rectangle",data{2});

reset(augmentedTrainingData);

end

figure

montage(augmentedData,BorderSize=10)

%%%%%%% Specify these training options %%%%%%%%

options = trainingOptions("adam",...

GradientDecayFactor=0.9,...

SquaredGradientDecayFactor=0.999,...

InitialLearnRate=0.0001,...

LearnRateSchedule="none",...

MiniBatchSize=4,...

L2Regularization=0.0005,...

MaxEpochs=70,...

BatchNormalizationStatistics="moving",...

DispatchInBackground=true,...

ResetInputNormalization=false,...

Shuffle="every-epoch",...
```

```
VerboseFrequency=20,...

ValidationFrequency=1000,...

CheckpointPath=tempdir,...

ValidationData=validationData);

%%%%%%%% Train Model %%%%%%%%

doTraining = true;

if doTraining

% Train the YOLO v4 detector.

[detector,info] =

trainYOLOv4ObjectDetector(augmentedTrainingData,detector,options);

else

% Load pretrained detector for the example.

detector = downloadPretrainedYOLOv4Detector();

end

% save train

train = detector;

save('newtrain_yolo4_edited3.mat','train')



% detection file

clear

close all

load newtrain_yolo4_edited3.mat
```

```
%networkInputSize = [608 608 3];

networkInputSize = [224 224 3];

%%%%%%%% Detect Objects Using YOLO v4 %%%%%%%%

I = imread('M8260019.JPG');

I = imresize(I,networkInputSize(1:2));

[bboxes,scores,labels] = detect(train,I);

ringBoxes = bboxes(labels == "ring", :);

detectRing =
insertObjectAnnotation(I,"rectangle",ringBoxes,"ring","Color","cyan");

figure

imshow(detectRing)

trophBoxes = bboxes(labels == "trophozoite", :);

detectTroph =
insertObjectAnnotation(I,"rectangle",trophBoxes,"trophozoite","Color","white");

figure

imshow(detectTroph)

schiBoxes = bboxes(labels == "schizont", :);

detectSchi =
insertObjectAnnotation(I,"rectangle",schiBoxes,"schizont","Color","blue");

figure

imshow(detectSchi)

gameBoxes = bboxes(labels == "gametocytes", :);
```

```
detectGame =
insertObjectAnnotation(I,"rectangle",gameBoxes,"gametocytes","Color","green");

figure

imshow(detectGame)

rbcBoxes = bboxes(labels == "RBC", :);

detectRBC = insertObjectAnnotation(I,"rectangle",rbcBoxes,"RBC","Color","red");

figure

imshow(detectRBC)

% Display the results.

detectImg = insertObjectAnnotation(I,"rectangle",bboxes,labels);

% last term 'scores' used to display the probability

% and 'labels' to display class

figure

imshow(detectImg)
```