Assessing blood samples for malaria

Tomáš Ťažár

2004694

Supervisor: Dr. Adrian F. Clark

Second Supervisor: John Q. Gan

A thesis submitted for the degree of

Bachelor of Computer Science

at the

Department of Computer Science and Electronic Engineering

University of Essex

Acknowledgements

Firstly, I would like to extend my gratitude to Dr. Vishwanathan Mohan and Professor Anthony Vickers for curating a dataset of projects, through which I was able to choose this exciting and challenging project.

I would also like to express my deep appreciation to my supervisor Dr. Adrian F. Clark for his guidance and unwavering support throughout the project. His insightful feedback and encouragement inspired me to go above and beyond my comfort zone and achieve my objectives.

Last but not least, I would like to acknowledge the resources listed below, these datasets and powerful tools were the backbone of the project and are of utmost importance, they allowed me to produce a complete and robust solution and make the project a reality:

- TensorFlow, an end-to-end machine learning platform. [1]
- Keras, Deep learning for humans. [2]
- OpenCV, a real-time optimized Computer Vision library. [3]
- National Institutes of Health Malaria dataset. [4]
- University College London Malaria dataset. [5]
- MNIST & The Cell Image Library External Test datasets. [6, 7]

Abstract

This project aims to develop a software solution that classifies red blood cells among at-risk patients to support the diagnosis and prevention of malaria. Using computer vision and machine learning techniques, the software accurately segments individual blood cells from blood smear images and classifies them according to their respective infection status. The software utilizes uniquely tailored state-of-the-art models that can analyze single images or folders of images with low computational complexity, making it suitable for use in less technologically advanced regions. The software is user-friendly with a simple design, allowing non-specialized users to operate it. It displays an evaluated result to the user almost instantaneously and has the ability to collect and save images and information for later analysis by experts. With its high level of accuracy and versatility, this project solution is ideal for affected at-risk environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this section, we provide an introduction to malaria, its societal impact, and the ongoing efforts taken to combat it. We also explore the challenges encountered in the field of malaria diagnosis. To address these challenges, a solution is proposed by the author of this thesis that can be applied effectively to the issues at hand.

1.1 Introduction to Malaria: Origins and Importance

Malaria, a life-threatening disease that is mostly characterized by periodic attacks of chills, fever, anemia, and often fatal complications, has plagued the human race for thousands of years. Throughout history, malaria has been prominent in ancient Egyptian and Greek civilizations and is even thought to have been infecting Old World monkeys [8]. In the 21st century, the incidence of malaria remains high with an estimated 247 million cases of malaria in 2021 accompanied by a death toll of 619 000 [9]. Even though this number is shown to be decreasing, malaria remains most prominent in tropical and subtropical regions of the world and countries experiencing disproportionate growth. It is speculated that entire populations, especially in sub-Saharan Africa, are infected more or less constantly. Malaria is also common in Central America, Northern South America, South, and Southeast Asia, and countries bordering the Mediterranean. This ascertains malaria as an internationally pernicious disease that poses a major threat to society as it is yet to be eradicated [9].

Malaria is caused by single-celled parasites of the genus *Plasmodium*, of which *P. falciparum* and *P. vivax* pose the greatest threat. *P. falciparum* is the deadliest malaria parasite

to date and is most prevalent on the African continent, on the other hand, *P. vivax* is the dominant malaria parasite in most countries outside of sub-Saharan Africa. *Plasmodium* parasites are spread by the bite of infected female *Anopheles* mosquitoes which act as the disease vector.

Malaria is preventable and curable, prevention methods including vector control, preventive chemotherapies, and vaccine administration are vital in controlling the spread of malaria and play an important role in eliminating high infection rates and reducing disease transmission. Vector control consists of two core intervention methods, Insecticide-treated Nets (ITNs) and indoor residual spraying, these are thought to be highly effective in preventing indoor infection. In addition, preventive chemotherapies administration is the action of taking medicines or a combination of medicines to prevent malaria and its consequences. It requires giving a full treatment course of an antimalarial medicine to vulnerable populations at designated time points during the period of greatest malarial risk, regardless of whether the recipients are infected with malaria. Finally, one of the most groundbreaking and significant methods of prevention is vaccines, World Health Organization (WHO) recommends the use of the RTS,S/AS01 (Malaria Vaccine) among children that are at threat of the specific strain of *P. falciparum*. The vaccine has been shown to reduce deaths and concentration of malaria significantly among young children [10].

1.2 Problem Statement and Solution Overview

The most time-consuming part of malaria is its diagnosis, this poses a massive issue as manual segmentation and classification is required to be done by specialised personnel which is rarely available, especially in endemic areas. Rapid classification of malaria is essential in affected environments as if not discovered in its early stages, there is potential for life-threatening consequences. It is crucial to be able to alleviate pressure from specialized workers and affected populations by introducing a software solution that is capable of automatic, speedy, and reliable diagnosis to prevent and remedy the effects of malaria.

In addition, prevention methods for controlling and eliminating malaria are readily available, however, there exist problems that prevent these methods from being effective. Factors such as increased resistance of *Anopheles* mosquitoes to insecticides and the unavailability of mosquito nets cause prevention methods such as vector control to stagnate in effectiveness and fail, this ultimately causes vector prevention to be essentially meaningless. Devel-

opmental factors such as limited resources and poor infrastructure make it challenging to administer preventive vaccination to an affected population and make it almost impossible to consistently receive and administer preventive chemotherapies on a regular basis. Regular administration of preventive chemotherapies is essential for herd immunity protection. These aforementioned factors are common in areas of low development and occur mostly in areas that are at the biggest risk of malaria, this causes an exacerbated impact of malaria on the population, and leads to malaria becoming more dangerous locally and internationally.

With malaria being too time-consuming to diagnose manually, and with the impact of poor development in select areas on prevention strategies, a solution that is easily accessible, easy to operate, and non-resource intensive is essential for patient diagnosis and subsequent treatment. This solution needs to have the following attributes:

- --→ Contains an automatic segmentation methodology
- --→ Contains an automatic classification methodology
- --→ Must be easy-to-operate
- --→ Needs to be non-resource-intensive
- --→ Must be accessible to a wide population
- --→ Has to be low-cost
- --→ Possesses high accuracy & reliability

A solution with the following attributes would be positively influential in malaria diagnosis, however, it would also be capable of providing the necessary information that would lead to prevention methods being applied more carefully and effectively. This would subsequently alleviate resource pressure from endemic areas and provide the ability to better allocate much-needed resources.

The attributes mention above are the core requirements of the solution that we propose in this paper. They act as the project aims and objectives and are essential in the development of a solution that is capable of alleviating the pressure malaria has on affected populations.

1.3 Innovative Approach: Our Solution to Combat Malaria

An end-to-end software solution has been developed to diagnose malaria patients utilizing the power of machine learning-based classification and computer vision-based segmentation in order to efficiently, accurately, and reliably classify malaria-ridden blood cells in blood smear images. We bridge the gap that exists in-between prevention and diagnosis methods by introducing an efficient and reliable end-to-end solution that can be utilized for malaria diagnosis. We provide a solution for the need for tedious diagnosis by medical specialists in endemic areas thereby decreasing resource demand, and resource sparsity.

We bridge the gap between diagnosis and prevention by making it possible for our software solution tool to be used in conjunction with treatment and prevention methods alike. When malaria is identified in a region or within an at-risk group through the use of our software, preventative chemotherapies can be applied to the population more effectively which decreases the overall resource load. Specialised personnel is no longer needed to manually identify malaria-infected cells which decreases the risk of disease spread in endemic areas. This is essential as the alleviation of risk is one of the most important prevention methods in disease-ridden areas.

With the use of our innovative software solution approach, we are able to gather invaluable information and data from affected regions and forward it to external operating experts for final validation if necessary. This reduces the need for travel to at-risk regions further decreasing the risk factor experienced in endemic areas. The data collected is extremely influential in prevention methods and surveillance methods that continuously and systematically collect, analyse, and interpret malaria-related data which is then used to aid the fight against malaria.

Chapter 2

Literature Review

The literature review chapter provides an overview in the current and related work done in the field of malaria diagnosis and automatic detection. It aims to provide a comprehensive understanding of the existing research and highlight the strengths and limitations of previous studies.

In a study, Kassim et al. [11] proposed a clustering-based dual deep learning architecture to reduce the computationally expensive cost of object detection networks on large thin smear microscopy. The `RBCNet` architecture was employed to detect red blood cells in malaria diagnostic smear with a cell detection accuracy of up to 97% and higher. Unlike most malaria screening algorithms, the proposed method uses cell-clustering methodology instead of thresholding methods, which can encounter problems or fail under challenging light conditions. The proposed method also outperforms methods aimed at separating touching cells, with up to six times lower standard deviation than standard. The proposed method was trained on a data set of thin blood smear images from human patients, which were photographed using a smartphone camera. This process has proven to be efficient and proves to be a robust solution for detecting red blood cells in thin blood smears.

In a study related to automatic object detection, N. E. Ross et al. [12], investigates the use of an automated image processing method for diagnosis and classification of malaria on thin blood smears. As this is a problem of classification, a standard pattern recognition method is adopted. The problem consists of four stages: image acquisition, pre-processing, feature generation and classification. A morphological method is adopted for identification of malaria. Thresholding segmentation is also employed which determines two threshold levels,

one for the red blood cells and one for the parasites, this provides higher accuracy on the overall result. Finally, a tree classifier with two nodes using BFF neural networks determines whether or not a cell is infected, and if so, the species of the malaria. Potential parasites are segmented from the image with a sensitivity of 92% and classification is able to positively identify malaria parasites with a sensitivity of 85%.

Accurate classification of parasite species suffers from inherent technical limitation of human inconsistency as vigorously trained specialists are required to carry out tedious malaria diagnosis work for appropriate drug administration. An automatic methodology by S. Kaewkamnerd et tal. [13] proposes an automatic prototype device for detection and classification of malaria parasites in thick blood film. This prototype is equipped with mountable motorized units for controlling the movements of objective lens and microscope stage, and the system is based on digital image classification analysis. Its design allows it to be mounted on conventional light microscopes used in endemic areas. The classification performance in thick blood film, the P. vivae parasite was correctly classified with the success rate of 75% while the accuracy of P. falciparum classification was 90%, this was achieved by the use of the motorized system to observe the sample as a whole.

Different methodologies by researchers are adopted in order to automate detection and classification methods in the field of malaria diagnosis, with the aim of improving diagnosis accuracy, reducing human error and enabling timely intervention. The studies by Kassim et al. [11], N. E. Ross et al. [12], and S. Kaewkamnerd et al. [13] highlight the significance of using automated image processing and pattern recognition techniques for identifying malaria parasites in thin and thick blood smears. These studies all employ a different approaches and provide promising results, it is important to note that a no one-size-fits-all approach should be adopted meaning that further research and experimentation are needed to optimize and improve current methodologies. These advancements in the field of automatic detection and classification methods have the potential to revolutionize malaria diagnosis, particularly in remote and low-resource settings, by providing a more efficient and accurate diagnosis process.

Chapter 3

Sustainability, Legality, Ethics, and Their Interrelated Issues

Sustainability, legality, and ethics have become critical topics of concern within software development. These issues are highly interrelated and therefore we must strive to create and implement sustainable, legal, and ethical practices in the development of our software solution. In this chapter, we delve into our project's interrelated issues of sustainability, legality, and ethics and explore how they can affect each other and the world as a whole. Furthermore, we explore methodologies that ascertain our project's software solution as sustainable, legal, and ethical.

3.1 Context of the Project

An end-to-end malaria diagnosis software solution plays a key role in affected regions, as it not only can speed up the diagnosis process and alleviate pressure from affected areas, but also promotes the implementation and research of deep learning-based software solutions into disease diagnosis as a whole. This can subsequently lead to an introduction of similar software solutions which would benefit an endemic region immensely. It is however important to consider the impact a software solution like this can have in different scenarios, as some regions could potentially be burdened by something they are not prepared to use. The impact of the project and other factors are explored in thorough detail in the sections below.

3.2 Health and Safety

Health and Safety play a key role in developing a project of this size, therefore it is vital to consider and take precautionary measures in order to prevent safety hazards and any form of injury.

The project's recommended development length has been set to 450 hours by the academic staff supervising project allocations. This development period has been set to the length of two academic terms, meaning that the development of the project was to be progressively and iteratively carried out. Work was done in regular daily intervals over the two academic terms and the author of this project was careful not to exceed the number of work hours that could be considered harmful. Risks associated with working long periods of time have been considered and can be found in a risk register form in Appendix A under *Activities* in the *Health and Safety* section.

3.3 Legal Considerations

The project aims to comply with all relevant legal requirements and is built primarily on open-source software. All data used for training, validation, and testing is referenced and the individual intellectual property laws are respected. The target end user of the software solution is urged to follow and respect all personal information laws under the respective jurisdiction of the country or area where the software operates. Data privacy and integrity are at the forefront of this project's legal considerations, and therefore measures should be taken in order to keep it this way.

The software of this project is under an open-source license and therefore issues of algorithmic transparency and the methodology of how results are achieved are available to be investigated by the public. This is done to keep the transparency of the project in check as the author of the project believes it is important to provide the end users with this information, especially when it comes to tackling an important issue like this.

3.4 Ethical Considerations

With an automated system being used for the diagnosis of patients, a lot of ethical questions come to mind. The software needs to be available in resource-limited areas, and areas of low

development, this raises an ethical question about the accessibility of healthcare and potentially further marginalizing vulnerable populations. To combat this, an important decision was made in the development of the software solution, this was to provide the software solution at no cost to affected environments mitigating any potential financial marginalisation.

Another concern is the possibility that the tool may be seen as a burden rather than a benefit. This could occur if the software solution is too complex to use, or if it requires significant resources to operate, such as electricity or a stable internet connection. In this case, the software solution would be rendered useless. To address this concern, the software solution was developed to be easy to use and to be non-resource intensive. Electricity is necessary for the use of the software solution however internet connection is not necessary, this comes as a slight drawback, however, without this, the tool would not be possible to use.

The software solution has been designed to be the most accurate and reliable as possible, however, it must be mentioned that it does not act as a replacement for human expertise and judgement in the diagnosis and treatment of malaria, it is designed to be used as a tool providing help and insight into a patients condition, and it is not a replacement for the diagnosis process as a whole.

3.5 Sustainability

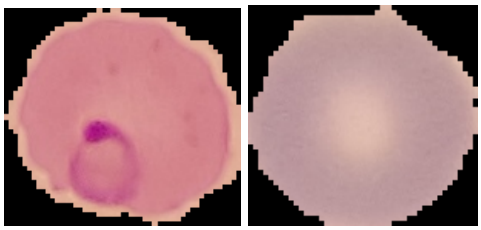
To address the issue of sustainability of our project we must consider the following; the use of energy-efficient hardware for training and evaluating our software solution, in turn also considering the environmental impact this creates, using a renewable energy source to power our software solution and optimise the software to have a low computational and time complexity. These methods would in turn reduce the amount of energy the software solution consumes making it well-adapted to endemic areas which usually suffer from a lack of resources. The project implementation does not require energy-intensive specialist hardware and can be used on a computer readily available in the technology market. The software solution was also designed to only use relevant resources during the evaluation process, meaning that when it is not being used, no additional resources are being used. We ensure that the production of the software solution and disposal of the software solution does not have a negative environmental impact by spending most of our development time in an environment that is capable of using sustainable energy sources.

Chapter 4

Classification Process

4.1 Initial Approach

A key stepping-stone in creating the end-to-end software solution was to be able to create uniquely tailored classifiers for our specific need to classify infected and uninfected cells efficiently and accurately. The initial approach considered the use of only one classifier as this was originally the primary goal of the project, however, throughout the project, it was decided to implement multiple classifiers for comparison and performance purposes. The NIH malaria dataset has been instrumental in the development of uniquely tailored and state-of-the-art machine-learning models for the detection and classification of malaria, and has contributed significantly to the progress in the field of computer-aided diagnosis of infectious diseases. This is why the NIH dataset was chosen for training, validation, and testing. The dataset contains two sub-folders of parasitized and uninfected cell images. Both of these categories contain 13,780 images making our sample size reasonably large for the state-of-the-art classification standard.



(a) Parasitized Cell (b) Uninfected Cell

Figure 4.1: Sample Cells from the NIH Malaria Dataset.

4.2 Data Pre-processing

Data from the NIH dataset has to be appropriately pre-processed
in order to be compatible for training, validation, and testing with our different classification models. The following pre-processing methods are implemented to create compatible image pipelines.

The NIH dataset has been pre-processed with the use of the `tf.keras.utils.image_dataset_from_directory` function [14] which generates a `tf.data.Dataset` [15] from image files in a directory for use with our Keras sequential model API [16] implementation. The `tf.data.Dataset` API supports writing descriptive and efficient input pipelines and allows dataset transformation and iteration, this is essential as the NIH image data pipeline can be converted to a compatible dataset suitable for classification. The sequential model API works best when the images from our training pipeline are resized to a format of `100x100` pixels, this is required for standardisation purposes as all the images in the dataset have different heights and widths. The size of `100x100` is specified within the `tf.keras.utils.image_dataset_from_directory` function as an argument and a training pipeline of pre-processed images is created. We then perform normalization operations on this pipeline by normalizing the image data into floating point values between 0 and 1. The normalization step is an essential pre-processing step as it helps the sequential models converge faster and produce better results. In addition, normalization makes the data more consistent and easier to work with which helps prevent the models from becoming too sensitive and reduces the impact of outliers and extreme values. After the normalization process, batching is carried out for the training, validation, and testing data where batches of 32 are created, of which we take 500 for training, 262 for validation, and 100 for testing. This translates into 16,000 training, 8,384 validation, and 3,200 testing images ready to be used with our classification models.

As previously mentioned, different models require different pre-processing methods and therefore the NIH dataset is pre-processed with the use of an `ImageDataGenerator` from the `keras.preprocessing.image` module [17] to be used with the state-of-the-art VGG16 [18] model for classification. The data pipeline takes the form of the `ImageDataGenerator` which uses the `flow_from_directory` function which takes the path to a directory and generates batches of augmented images from the NIH dataset. The `ImageDataGenerator` normalizes the image data to floating point values between 0 and 1 for faster and better

results, just like with the sequential model. The default input size of the VGG16 model is `224x224` pixels, and therefore the image format is resized to `224x224` pixels as this is required for image standardisation. The data is then transformed into batches of size 32 and doesn't get shuffled, the `class_mode` is set to categorical which determines what array the labels of the image data will be returned in. In this case, the array will be a 2D numpy array of one-hot encoded labels. This is required as most machine learning models cannot operate on label data directly, and therefore it must be converted to a binary representation for each unique label. This is a key pre-processing step as otherwise, the model would be able to assume the natural ordering of labels which would result in poor performance or unexpected results. This results in compatible training and validation sets that can be used for training with the state-of-the-art VGG16 model.

4.3 Model Selections and Choices of Architecture

In this section, the author discusses the choice and motivation behind choosing the classification models used in the end-to-end software solution.

The first set of models originates from the Keras sequential model API. These models are heavily customizable as a custom architecture of layers can be implemented, this means that we are able to construct our own specific model architecture aimed at image detection and classification. This freedom of custom layer implementation also gives us the ability to fine-tune respective layers for more efficient performance and higher accuracy. The first sequential model was optimized for general image classification, however, the second model introduced adjustments to its architecture where a spatial attention mechanism was implemented. This mechanism was implemented to focus on specific parts of the training and validation images with the goal of improving efficiency and classification accuracy. The customizability of the models is one the largest factors of why they have been chosen, these architectures allow us to tailor the model until a desired, high-quality result is achieved.

The state-of-the-art VGG16 model has been chosen as it is one of the most widely accepted and commonly used image classification models. The VGG16 is a Deep Convolutional Neural Network (DCNN) [19] with a depth of 16 layers. K. Simonyan et al. [20] investigate the effect model depth has on accuracy in a large-scale image recognition setting in their paper, and are able to conclude that with the use of very small convolution layers, the accuracy

is able to achieve state-of-the-art. The DCNN layers have been made publicly available and therefore we are using the pre-trained layers of the VGG16. Due to the fact that the VGG16 model has been pre-trained on a different large dataset, we are applying transfer-learning techniques while retraining some of the weights on our NIH dataset. While the VGG16 DCNN is large depth-wise, it still remains computationally efficient, allowing for faster training and evaluation times with the use of a Graphics Processing Unit (GPU). This model is a reliable and efficient choice for our end-to-end software solution and provides a valuable comparison to our sequential models.

4.4 In-depth Analysis of Keras Sequential Models

The Keras sequential models are simple convolutional neural networks built using the Keras Sequential API. The first sequential model consists of a stack of three sets of layers: convolutional layers, pooling layers, and dense layers. The input to the model is an appropriately pre-processed image of size `100x100` pixels with three colour channels, this is specified within our first Conv2D [21] layer as it is our input layer to the model. This layer consists of 16 filters of size `3x3` pixels and a stride of 1. The layer performs spatial convolution operations on images with kernels of size `3x3` and subsequently creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. This generates 16 feature maps by applying 16 different filters through the convolution kernels. It uses a ReLu [22] activation function which applies the rectified linear unit activation function, this is necessary to ascertain non-linearity to the outputs of the convolution operations. The following layer is a MaxPooling2D [23] layer with a pool size of `2x2` pixels. This layer performs a pooling operation on the output of the previous convolution layer, its purpose is to reduce the spatial dimension of the feature maps by taking the maximum value within each `2x2` subregion of the feature maps. MaxPooling2D is found to work well in combination with convolution outputs as it is assumed that it helps to extract the sharp and smooth features of the input image, helping to find low-level features like edges and points. After this, another Conv2D is applied with a filter size of 32 and a kernel size of `3x3` pixels, and a stride of 1. The stride is key in the convolutional layers as it defines the number of pixels we are to skip when moving to the next convolutional operation. This layer performs another convolution operation on the output of the previous max pooling layer with a larger number of filters to generate more feature maps, it also uses the ReLu activation to ascertain non-linearity. Following this layer, we use

a copy of the previous MaxPooling2D layer and use it to pool the outputs of the convolution yet again. This process is repeated once more where a Conv2D layer with a filter size of 16 and kernel size of `2x2` is applied after the current MaxPooling2D layer followed by another MaxPooling2D layer with a pool size of `2x2` pixels to further reduce spatial dimensions of the feature maps.

After these six layers, a Flatten [24] layer is introduced which flattens the output of the previous pooling layer into a 1-dimensional array that can be fed into a fully connected neural network. This flattened output is fed into our last two Dense [25] layers, with one containing 256 neurons and a ReLu activation function, and the other containing 1 neuron with the Sigmoid activation function. The first Dense layer takes the flattened 1-dimensional array output and generates a 256-dimensional output vector, and subsequently, the ReLu activation function is used to introduce non-linearity to the output. Our last Dense layer is another fully connected neural network layer that takes the output of the previous layer as input and generates a single scalar output between 0 and 1, which is used to classify the input image as belonging to one of the two infection classes. It uses a sigmoid activation function which is essential as it aids in the scalar output being decisive in outputting an infection class. These Dense layers are essential as they are arguably the most important part of the hidden layer in a neural network, they perform matrix-vector multiplication operations and all of the neurons of these layers are connected to all the previous neurons making them fully connected layers. This completes the architecture of the first model, however, there is one more step that is necessary to make the model ready for training. The last Compile [16] layer compiles the model for training and introduces the optimizer, loss function, and evaluation metric for the model. In our case, the Adam [26] optimizer is used that implements the Adam algorithm, this optimization is a stochastic gradient descent algorithm that is based on adaptive estimation of first-order and second-order moments. According to Kingma et al. [27], the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters". The BinaryCrossentropy [28] loss function computes the cross-entropy loss between true labels and predicted labels, this is especially relevant in our case as we are looking for a binary classification in our output. Finally, the evaluation metric of accuracy looks for how often the model predictions equal the true labels. This completes the architecture of the first sequential model and makes it ready for training on our NIH dataset.

The second sequential model possesses similar attributes in its architecture, however, one

major difference is the implementation of a spatial attention mechanism aimed at focusing on specific parts of the training image. This makes its architecture slightly more complex. The beginning of the enhanced sequential model architecture consists of an Input layer which takes in the shape of our appropriately pre-processed NIH images with the dimension of `(100,100,3)`. Following the input layer, a Conv2D layer is established with a filter size of 32 and a kernel size of `3x3` pixels, and a ReLu activation function, followed by a MaxPooling2D pooling layer with a pool size of `2x2` pixels. The second convolutional layer has 64 filters with a size of `3x3` and a ReLU activation function, followed by another max-pooling layer. The final convolutional layer has 128 filters with a size of `3x3` and a ReLU activation function. This final convolution layer is then passed into our spatial attention function which takes the layer's output feature map as an input argument. This spatial attention function brings the ability of the model to be able to focus on specific parts of the image by finding the weighted average of the feature maps from the last convolutional layer. The feature maps from the last convolutional layer are then passed into the GlobalAveragePooling2D [29] layer which reduces the shape of the feature maps to `1x1` pixel. This is then passed into a Conv2D layer which applies one convolutional filter of size `1x1` pixel, accompanied by a sigmoid activation function. This generates a weight for each channel in the feature map creating the basis of our attention mechanism, the resulting attention map is element-wise multiplied with the original feature map to create a new feature map that emphasizes the most important parts of the input. After our spatial attention mechanism is applied, the output tensor is flattened and passed into our final set of layers. The flattened tensor is then passed through a fully connected Dense layer with 256 units and ReLU activation, which applies a linear transformation to the input vector and introduces non-linearity to the output. After the Dense layer, a Dropout [30] layer is introduced to prevent overfitting. The Dropout layer randomly sets input units to 0 with a rate of 0.5 at each step during training time to also prevent over-reliance on specific units in the network. Since the Dense layer is fully connected the Dropout layer affects all the inputs in the neural network. Finally, a fully connected Dense layer with a single output and a sigmoid activation function is added to generate the binary classification of our infection labels. The model is then compiled for training with an Adam optimizer, a loss function of BinaryCrossentropy, and an evaluation metric of accuracy. This completes the architecture of the second sequential model and makes it ready for training on the NIH dataset.

4.5 In-depth Analysis of the VGG16 Model

The VGG16 model is a convolutional neural network that has been pre-trained on the ImageNet [31] database and consists of 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers. This makes the total number of layers 21, however, only 16 are trainable. One of the most unique things about the VGG16 model is that instead of having a large number of hyper-parameters it is focused on having convolution layers of a `3x3` filter size with a stride of 1 and always uses the same padding and max-pooling layers of `2x2` filter with a stride of 2. There are 5 different groups of convolutional layers in the VGG16 model. The first convolution group has a filter size of 64, the second has a size of 128, the third has a size of 256, and the convolutional layer group 4 and 5 have a size of 512 filters. The convolution layers are consistently arranged throughout the whole architecture in groups of two or three followed by a pooling layer, similar to our sequential model implementation. These convolutional groups are followed by a layer of three fully connected Dense layers where the first two have 4096 channels each, and the third performs classification for the datasets it has been trained on. We are able to take advantage of this pre-trained architecture structure by invoking pre-trained weights while initializing the model. We begin this process by specifying the input shape for the VGG16 model taking in the input tensor size of `224x224` pixels with a third dimension for the colour of the image. We set the weights parameter to 'imagenet', which means that we are initializing the model with pre-trained weights on the ImageNet [31] dataset. We do not include the fully connected layers at the top of the model stack due to the fact that we will be training our own. We set all the layers in the model instance to be non-trainable, as we are using the pre-trained ImageNet weights. We add a Flatten layer which will flatten the output of the VGG16 pre-trained convolutional layers forming the basis of our fully connected layers. We then add a Dense layer with 2 neurons representing the number of classes in our dataset, with a softmax [32] activation function. This final layer acts as the fully connected layer to the output of our previous flattening layer and is responsible for outputting the probability distribution for the infection classes. We then instantiate a Keras Model [33] instance which takes the input of our VGG16 model and produces an output from the neural network's class predictions. This is then compiled with an Adam optimizer, a loss function of CategoricalCrossentropy [34] which also computes the cross-entropy loss between true labels and predictions, and finally the evaluation metric of accuracy is ascertained. With this final step, the trainable weights of the model are ready to be trained on the NIH dataset.

4.6 Model Training

We standardized the training process for our classification models by using a batch size of 32 and training for 20 epochs. the VGG16 model was trained on a training length of 10 epochs. Transfer learning was applied in the VGG16 model, utilizing pre-trained weights from the ImageNet [31] dataset. Our evaluation metric of choice was accuracy, which forms the core of our results discussed in section 4.7. Table 4.1 shows the number of trainable and non-trainable parameters for each model.

Model	Trainable Parameters	Non-Trainable Parameters
Keras Sequential	419,825	0
Keras Sequential (Spatial Attention)	14,544,578	0
VGG16	50,178	14,714,688

Table 4.1: Number of trainable & non-trainable parameters for each model.

The models were trained on `NVIDIA GTX 1080` and `NVIDIA RTX 2060` GPUs, with an average training time of 4 minutes for sequential models and 10 minutes for the VGG16 model. There was an option of using a Central Processing Unit (CPU) for training, however, the training time was significantly longer.

4.7 Results, Metrics and Performance Analysis

This section focuses on the performance analysis of our classification models. We examine key performance metrics such as Accuracy, Recall, Precision, F-score, Average Precision Score [35][36], to evaluate the effectiveness of our models. Furthermore, we explore the training process performance graphs and employ statistical analysis, specifically McNemar's test, to compare and contrast the performance of our classifiers. This enables us to give a strong overview of the effectiveness of our models.

One of the most important metrics that our classification models produce is accuracy. Accuracy is represented as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

and it is vital as we are able to gauge the effectiveness of our classification models early on

with the use of accuracy. However, there are better metrics for evaluating class-imbalanced problems such as ours. These metrics are Precision & Recall. Precision is represented as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

and Recall is represented as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

where *TP* represents the True Positives, *FP* the False Positives and *FN* the False Negative model predictions. These predictions are vital as we can see whether our classification model finds the correct answer or makes a mistake in its prediction. This is why Precision is the proportion of how many positive class identifications were actually correct, and Recall is the proportion of how many actual class positives were identified correctly by the models. These metrics provide a better insight into the performance of our models, however, Precision and Recall are usually in tension as improving one usually results in reducing the other, this is why it is important to investigate the balance of these two metrics. This can be done with the use of the F-score. The F-score is a statistical analysis of binary classification and is calculated as a harmonic mean of Precision and Recall. The F-score is represented as:

$$\text{F} - \text{Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

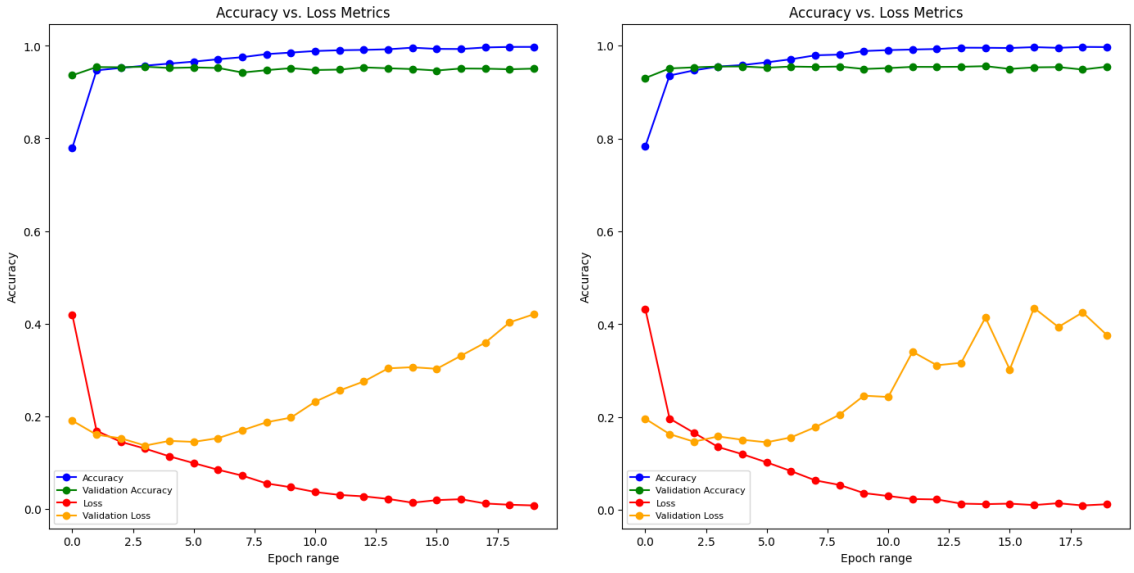
This symmetrically represents both precision and recall in one metric. The F-score is vital as we are able to find the best of both metrics that would otherwise be in tension, this provides the most accurate evaluation metric for our classification models. Additionally, we include the Average Precision Score (APS) in our evaluation metrics. APS is the weighted mean of Precision scores achieved at each PR curve threshold, with the increase in Recall from the previous threshold used as the weight. A mathematical representation of APS is available in Appendix B.

Table 4.2 below shows the performance metrics for each of our classification models.

Model	Accuracy	Precision	Recall	APS	F1-Score
Keras Sequential (Spatial Attention)	0.9809	0.9795	0.9584	0.9945	1.0000
Keras Sequential	0.9506	0.9396	0.9650	0.9933	0.9412
VGG16	0.9258	0.9846	0.9028	0.9537	0.927

Table 4.2: Model performance metrics.

Furthermore, investigating training performance is vital in deducing whether our classification models are learning at an appropriate rate and whether they suffer from any training inaccuracies. We are able to gain insight by investigating training performance graphs by looking at metrics such as training accuracy, validation accuracy, loss, and validation loss. This is examined over the length of our training period described in section 4.6. Figure 4.2a shows the training performance graph for the first Keras Sequential model, and Figure 4.2b shows the training performance graph for the Keras Sequential model equipped with the Spatial Attention mechanism.



(a) Keras Sequential

(b) Keras Sequential (Spatial Attention)

Figure 4.2: Training performance graphs for Keras Sequential models.

The training performance of the VGG16 model was also investigated and is shown in Figure 4.3. The VGG16 model has been trained over a training period of 10 epochs as the training process for the state-of-the-art model is very computationally expensive and time-consuming.

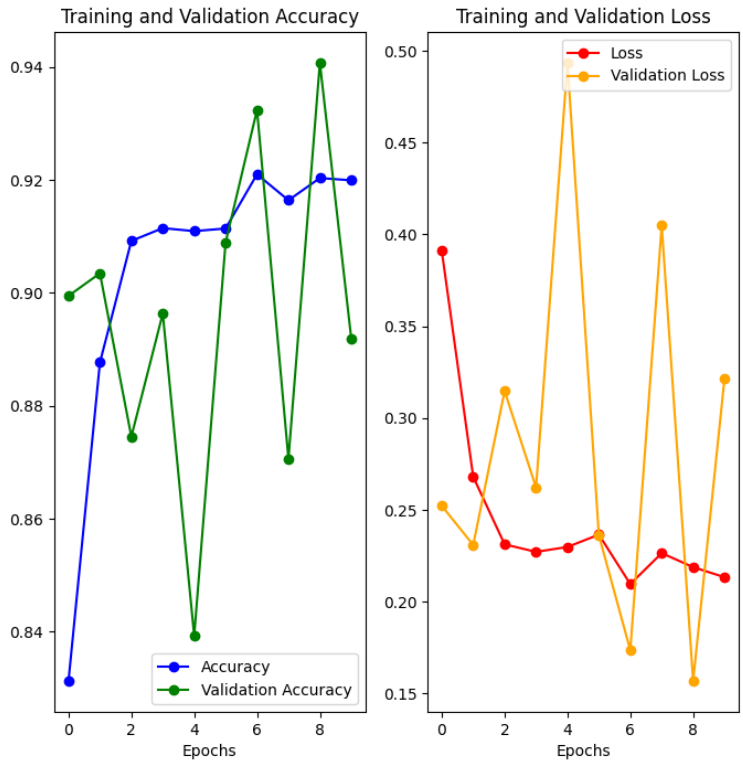


Figure 4.3: Training performance graph for the VGG16 model.

Training Performance Analysis

These performance graphs give us a strong indication of the model's training process. The Keras sequential models have shown a stable increase in accuracy and a very high constant standard of validation accuracy, this means that the model was able to learn from the NIH training data very well, this same phenomenon was observed in the Keras sequential model with the spatial attention mechanism included. Both of these almost achieved a training accuracy of 1, which is remarkable. Both of these models were able to showcase a stable decrease of loss, the lower the loss of the model the better it is able to perform, therefore, this

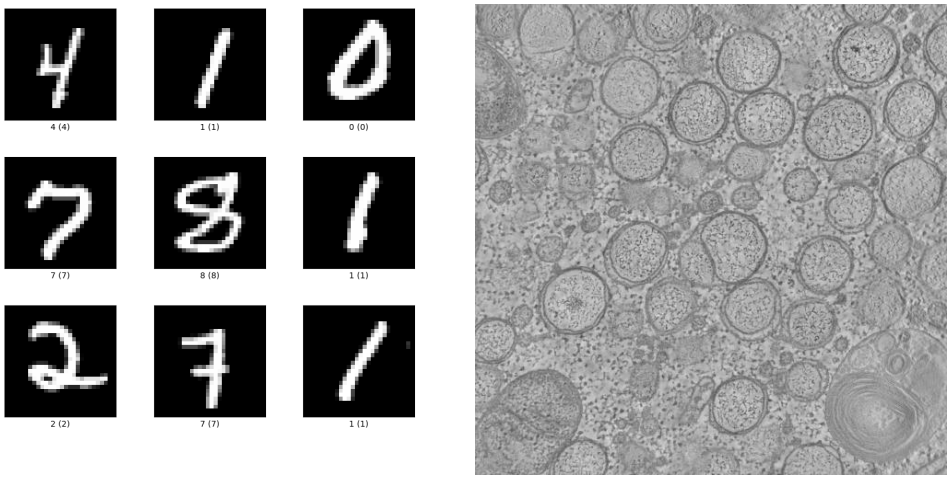
is observation is what we were looking for.

Validation Loss, on the other hand, was an unexpected outlier in both of our Keras sequential models, this could be a potential indication of the model overfitting. Overfitting occurs when the model becomes too complex and isn't able to generalise well on new data, this causes it leads it to perform poorly on data it has never seen before. Techniques such as cross-validation and early-stopping aim to decrease the chances of this happening, this is also the reason why the model was not trained longer as an early-stopping strategy was implemented. Regardless of this, the Keras sequential models have proven to be performing extremely well over the training process.

During the training process of the VGG16 model, accuracy showed a stable increase, indicating good performance and learning from the NIH data. Validation accuracy showed a more unstable behaviour where a stable increase was still seen, however it was accompanied by large fluctuations. Training loss followed a similar trend to accuracy, however, in the opposite direction, where it was slowly decreasing other than an anomaly experienced between epochs 4 and 6, this indicates good performance and shows that the VGG16 model was able to learn from the training data quite well. Validation loss showed to be fluctuating heavily from epoch to epoch which could spell the same fate as its impact on the Keras sequential models. Validation loss is still shown to be decreasing however if the problem persisted and affected our model performance, a new training process with validation loss mitigation strategies would have to be considered. Based on the training performance graph, we are able to conclude that the VGG16 model was able to train the trainable weights to a very high standard as it performed very well during the process reaching a training accuracy of 0.91.

4.8 Reaction to Unknown Data

The MNIST dataset and project *P2005* from CIL have been used to investigate the effect unknown datasets have on our trained classification models [6, 37]. MNIST is a dataset of handwritten digits used commonly to benchmark the performance of classifiers. Project *P2005* consists of monkey `VeroE6` kidney cells and investigates the membrane modifications induced by coronavirus. The project uses '*Electron Tomography of Membrane Modification*' for its findings. Reconstructed images *6020*, *6021*, *6022* have been used to investigate the response of the classifiers. MNIST and CIL example images are shown below in Figure 4.4.



(a) MNIST example images. (b) Image *6021* from CIL.

Figure 4.4: Example images from the MNIST and CIL datasets.

The classifiers came to the following predictions:

Image	**Keras Prediction**	**Keras (Spatial Attention) Prediction**	**VGG16 Prediction**
6020	Negative	Negative	Negative
6021	Negative	Inconclusive	Negative
6022	Negative	Inconclusive	Negative

Table 4.3: Predictions of the trained models on the CIL dataset.

Model	**Accuracy**
Keras Sequential	0.098
Keras Sequential (Spatial Attention)	0.098
VGG16 Model	0.1137

Table 4.4: Accuracy of predictions on the MNIST dataset.

The classifiers tried to classify the CIL images correctly even though they were trained on a completely different dataset, likewise, the classifiers were able to reach an accuracy score shown in Table 4.4 even though the MNIST data differentiates heavily from the NIH training dataset. This shows that classification models cannot distinguish the 'correct' type of data they need to be predicting on and still try and predict the class regardless of the input image.

4.9 Implications of Results on the Classification System

The results obtained from Table 4.2 show that the Keras sequential model equipped with the spatial attention mechanism was able to achieve the highest results from all three models, with the f-score reaching a remarkable result of 1. This suggests that the model should theoretically be performing the best in the end-to-end system out of all the other classification models. The Keras sequential model produced strong results being closely behind the spatial attention sequential model, this also suggests that the model should theoretically be performing very well in our end-to-end software solution. The Keras sequential models are also relatively lightweight performance-wise when compared to the VGG16 model, therefore they should be quite efficient. The results from the VGG16 model were the lowest, however, close behind all the other results. The model was able to achieve the highest precision and a relatively high f-score of 0.927. A prediction therefore can be made that the model is expected to perform quite well in the end-to-end solution. The VGG16 model is therefore a good candidate for the end-to-end software solution.

McNemar's test was performed on the results obtained from the Keras sequential models as these were the models that had compatible test data pipelines. The results from the McNemar's test showed that there was no significant performance difference between the two models as it was unable to reject the null hypothesis.

All classifiers showed a stable increase in training accuracy in the performance graphs showing they were able to learn well from the NIH dataset, it is however imperative to consider that these results do not actually indicate the prediction performance of the classifiers.

It is also important to consider the results obtained by looking at unknown data. The results primarily show that the classification models are unable to distinguish data that they definitely should not be predicting. The results in Table 4.3 and 4.4 show that the classifiers weren't equipped to understand the data provided to them, therefore, producing poor results. The conclusion we are able to draw from this data is that the VGG16 model had the highest accuracy on the MNIST dataset showing that it was able to predict on the dataset with a higher accuracy due to having pre-trained weights.

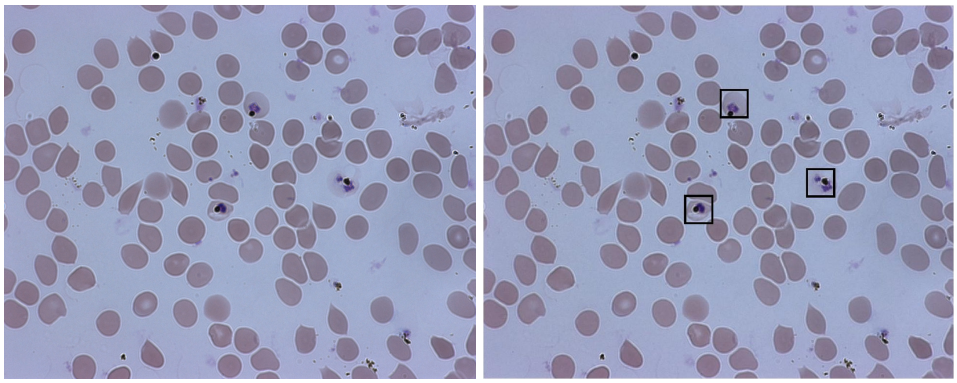
The results indicate that the sequential model with spatial attention performs the best in our end-to-end system. Both the sequential and VGG16 models show competitive performance, with the latter being a good candidate for the end-to-end system.

Chapter 5

Segmentation Process

5.1 Initial Approach

The end-to-end software solution required a robust and accurate methodology in order to be able to detect cells within blood smear images and isolate them from their background. Therefore, the next key step in the development process was to create a segmentation implementation that is capable of these requirements. Multiple segmentation methods were investigated to achieve the most accurate result, these are mentioned in sections 5.3 and 5.4. These methods were applied to the UCL malaria dataset and were used for testing how accurately and reliably our methods could identify potential Regions of Interest (ROIs). The UCL dataset contains 100 ground-truthed labelled images and 100 testing images. These were instrumental in benchmarking the performance of our segmentation methods.



(a) Test Blood Smear Image (b) Expert Labelled Blood Smear Image

Figure 5.1: Example UCL Blood Smear Image.

5.2 Image Pre-processing

It is imperative that the images obtained from the UCL dataset undergo meticulous pre-processing procedures before they can be used in cell detection. Various segmentation techniques demand distinct pre-processing methods, these are described in this section in detail.

For use with Otsu's segmentation method [38], images are required to be read with the use of the OpenCV `cv2.imread` function which is responsible for loading an image from a file [39], in this case, we are reading the image in `IMREAD_COLOR` mode which converts the image to a 3 channel Blue-Green-Red (BGR) image. This makes the image easier to work with. The next step is to apply a Gaussian blurring function from OpenCV's `cv2.GaussianBlur` [40], this method utilizes a Gaussian kernel with specific input dimensions. The input dimensions of our kernel are `(5,5)` and we also specify a standard deviation in the `x` and `y` directions, this is specified to be `0`. Gaussian blurring is extremely efficient at removing Gaussian noise from an image, "sources of Gaussian noise in digital images arise during acquisition e.g. sensor noise caused by poor illumination and/or high temperature, and/or transmission e.g. electronic circuit noise" [41]. It is key to be able to remove this noise as this increases the accuracy of our segmentation methods down the line. The image is then converted to greyscale using the `cv2.cvtColor` function which is responsible for converting an image from one colour space to another [42], with the argument of `COLOR_BGR2GRAY`. The images are now ready for use with Otsu's method of segmentation.

K-means clustering [43] requires a similar however slightly different pre-processing methodology. The image is read with the use of the `cv2.imread` function in BGR mode and blurred using the `cv2.GaussianBlur` function with a filter size of `(5,5)` and a custom standard deviation of `0`. The following step is to convert the image colour space with the use of the `cv2.cvtColor` function again, however, in this instance, we will be converting our image to the `CIELAB` colour space [44]. This colour space is intended "as a perceptually uniform space, where a given numerical change corresponds to a similar perceived change in color" [44]. The colour space is sensitive to slight changes in colour and is device-independent making it a good choice for applications where colour consistency is important such as K-means clustering. In our pre-processing methodology, we are looking to extract the `L*` channel which represents the lightness value in the image. This is important as it will be used in the K-means clustering algorithm, however, before that can happen, a final reshaping function `np.reshape` [45] must be applied, where the image is converted to a 2-dimensional

array for compatibility purposes. After these techniques are finished, the pre-processed image is ready for use with the K-means clustering algorithm.

5.3 Otsu's Method

Otsu's method or sometimes referred to as *Otsu's segmentation* is an algorithm that performs automatic image thresholding. The algorithm returns a single threshold that separates the image into a foreground and background, this is done by minimising intra-class intensity variance, or equivalently, by maximizing inter-class variance [38]. Otsu's method is very similar to a globally optimal k-means, discussed in 5.4, performed on its intensity histogram. The algorithm exhaustively searches for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes: see Appendix C for an in-depth mathematical explanation. This methodology allows us to automatically distinguish the foreground and background from images with ease. Otsu's method is particularly effective on images that have a high variance in terms of light and contrast artifacts. An example of this is shown in Figure 5.2.

(a) (5.2a) Original Image. (b) (5.2b) Otsu-processed image.

Figure 5.2: Example of Otsu's Method.

In the end-to-end implementation we use Otsu's method as follows. We are able to utilise OpenCV's `cv2.treshold` function which performs Otsu's binarization [46] on our pre-processed grey-scale image, this produces a binary image of ones and zeros. These ones and zeros represent the foreground and background of our image identifying our ROIs. An extra flag `cv2.THRESH_OTSU` has to be passed for Otsu's method to automatically find the fore-

ground and background. This also returns an optimal threshold value which is not needed for our use. The output binary image is then used for finding contours, this process is discussed in section 5.5. With the implementation of Otsu's automatic thresholding, we are one step closer to automatically segmenting individual cells from blood smears.

5.4 K-means clustering

K-means clustering is a widely used unsupervised machine learning method that makes use of vector quantization, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of a cluster [47]. K-means assigns data points to clusters based on distance from the cluster centroid and calculates cluster centroids based on cluster membership. The method randomly initializes k centroids, and then iteratively updates cluster assignments and centroid positions until convergence is reached. The k-means method is similar to Otsu's method where it tries to minimize the within-cluster variance creating different positions of cluster centroids on each iteration. The advantage of K-means clustering is that it allows for more than 2 classes, in this case, clusters. The problem is considered to be computationally difficult (*NP-hard*) [47], however, implementations of the K-means method can converge quickly to a local optimum. The process continues until centroids no longer move or change shape or until a maximum number of iterations is reached.

Implementing the K-means method in our end-to-end solutions is by utilising the scikit-learn machine learning library which is a simple and efficient tool for predictive data analysis [48]. We are implementing the `sklearn.cluster.KMeans` [49] function which implements the K-means method, this function requires a critical argument passed to it, which is the number of clusters the K-means method is supposed to converge on. The default number of clusters is set to `4` and is initialised from the `.config` file which is described in more detail in section 6.3. The number of clusters is key as it determines the accuracy of our segmentation method and the computational cost of segmentation which is of significant importance within our project. The `sk.learn.cluster.KMeans` function utilises *Lloyd's* algorithm which is considered a naive k-means algorithm. The `sci.kit` K-means algorithm is considered to be "very fast (one of the fastest clustering algorithms available), but it falls in local minima" [47]. This is why the K-means methodology is required to be run multiple times or to be averaged from multiple iterations. This is also why we include a `random_state`

argument which is set to `0`, this makes the randomness of the algorithm deterministic. This function is then fit onto our pre-processed reshaped image with the use of the `.fit` function, this returns a KMeans object which contains data about clustering results. This data contains several different attributes, including the cluster assignment of each data point, the cluster centroids, and the within-cluster sum of square distances. These are accessed in our code when we `np.reshape` assignments of cluster data points into a three-dimensional array which represents our clustered image.

5.5 Isolating Image Components and ROI Identification

The next step in creating a robust segmentation system in our end-to-end solution is to be able to isolate image components and identify the ROIs after Otsu's method and K-means clustering are applied.

With our binarized and clustered images, we apply the `cv2.findContours` function [50] which returns a list of contours found in the image, there is also a hierarchy of contours returned from this function, however, we will not be needing this in our process. The next step is to be able to filter contours based on the size of the contour area, this is to prevent small image artifacts, outliers, and extremes to be processed in the end product. We set the minimum contour area at `550` and the maximum area at `10,000`, we then append all these contours to a `contours_array` which contains all the valid contours.

We are now ready to work with our segmented contours and apply operations to them to make them ready for final identification and classification. However, before this is done, we iterate through our `contours_array` and use the `cv2.boundRect` function to find the bounding rectangle of each contour in the list of contours. We are then able to create a custom mask through which we can draw contours using the function `cv2.drawContours` on the original image that we read during pre-processing (described in section 5.2). We are able to use this custom mask with the original image to extract ROIs using the `cv2.bitwise_and` operation. These extracted ROIs are then extracted into a final list `regions_of_interest`. The ROIs are now ready for infection status classification in our end-to-end solution.

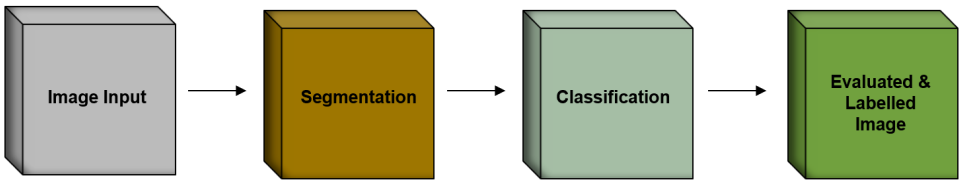
Chapter 6

End-to-end Software Solution

In this chapter, we delve into the seamless integration of our segmentation and classification methodologies, showcasing their dynamic synergy in producing an end-to-end software solution that is reliable and accurate. We showcase the result produced by our end-to-end software solution and we comprehensibly explore features of our integrated approach and describe how end users can leverage its capabilities. To ensure the end-to-end solution performs as expected, the performance is evaluated using appropriate testing methodologies which provide a critical analysis of its strengths and limitations. This chapter represents the culmination of development for our final end-to-end software solution.

6.1 Integrated Approach: Combining Segmentation and Classification Methods

Integrating segmentation and classification into an end-to-end system proves to be one of many ways these bleeding edge methodologies can be used together to automate the detection and diagnosis of malaria cells in blood smear images. We are able to utilize the classification models trained on the NIH dataset and use them to predict the infection status of cells which are automatically segmented and extracted from the UCL testing dataset. This specific architecture structure takes images as input, processes the image using our segmentation methodologies, predicts the infection status of segmentation output using our classification models, and subsequently outputs an evaluation of the image to the user in a robust manner. Figure 6.1a shows a high-level representation of the end-to-end system.



(a) 6.1a End-to-end Software Architecture.

The execution of this integrated approach is carried out in the following manner. After the processed image regions are segmented and ROIs are created, they are collected in an array which is to be evaluated. Training models are then loaded into memory with the use of the `tf.keras.models.load_model` function, this enables us to call on the model object to gather predictions on the ROIs. The array of ROIs is iterated through seamlessly and each region is resized to a suitable format for model compatibility, the sizes of the ROIs have to equal to the input size of images the models are trained on, this topic was discussed in Chapter 4 section 4.4, 4.5 in greater detail. After the ROIs have the correct sizes their infection status is ready to be predicted. The `.predict` function is used to create an infection status prediction which is appended to an array of `predictions`. This process creates a result array that contains a prediction of each region of interest's infection status, these regions of interest represent the red blood cells in the blood smear image.

This process tries to evaluate every red blood cell that the segmentation methods find. After this is completed, the result has to be processed further so it can be appropriately labelled and displayed to the user with sufficient information.

6.2 End-to-end System Output

The end-to-end system output is the most crucial part of the end-to-end system as it provides the evaluation information to the user in a comprehensive manner. The default output from the system is an interactive window that is iterable depending on the number of input images provided, these features are described in more detail in section 6.3.

The process of evaluating and outputting the results follows the following methodology implemented in our code. The evaluated ROIs and contours are iterated through at the same time in the `display_labelled_image` function, this function allows us to create a `cv2.boundingRect` around the ROIs which is used as a label for the cell infection status.

We apply a methodology that checks the `predictions` array for the infection status and assigns a colour for the respective infection status. We then draw a rectangle around the ROIs and apply the colour to represent the infection status. This produces the evaluated and labelled image which is shown in Figure 6.2 below.

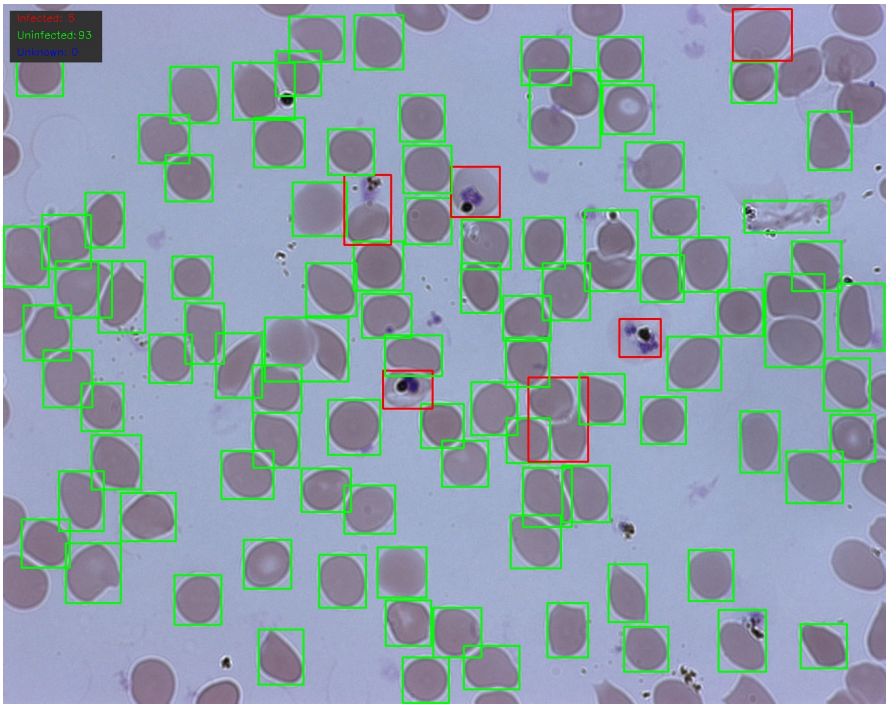


Figure 6.2: Segmented and Evaluated Image.

The segmented and evaluated image provides some key information within the image itself and as plain text in the command line output. This information includes the number of infected, uninfected, and total cells in the image to give the user a quick understanding of the produced output. Some more detailed information can be found the in the command line where the length of the evaluation time is shown in seconds, the classification model and segmentation method used for the evaluation and most importantly the percentage of infected cells in the image. This information can be used as a strong basis to evaluate the performance of the end-to-end system and to evaluate the performance of the classification and segmentation methodologies. Further features are described in section 6.3 below.

6.3 Features of the Integrated Approach

The end-to-end system is a command line application which takes different arguments to control the behaviour of the system. The system is built with the end user in mind therefore the command line arguments are quite intuitive. A `.config` configuration file is included with the system which contains some default parameters that the user is able to freely change. These parameters are discussed in more detail in the sub-section 6.3.2 below.

6.3.1 Example System Runs

Despite the system being quite intuitive, an example system run can be executed in the command line with the following command:

```
python3 malaria_evaluation.py --i input_image.jpg
```

This example run segments, evaluates and displays the output for a single image. Where `input_image` represents the name of the image we want to evaluate. We are also able to segment and evaluate entire folders of images at once. This can be done with the following command:

```
python3 malaria_evaluation.py --f input_folder/
```

This also displays the segmented and evaluated images, however in this case, we are able to interact with the display window and view the entire folder of evaluated images with the use of keyboard inputs. These inputs are described in the command line output.

A default segmentation and evaluation procedure can be run using the following command:

```
python3 malaria_evaluation.py --d
```

this evaluates a default folder included in the file structure which consists of samples from the UCL dataset.

If a user is unsure about how to operate the system, they can display a help menu which describes evaluation and other features in more detail by using the following command:

```
python3 malaria_evaluation.py
```

6.3.2 Configuration File

A configuration file is included by default which contains information about default locations of important folders in the file structure. It also contains the number of images the default evaluation procedure is going to use for evaluation from the UCL folder. The number of clusters the K-means algorithm uses is also included in the `.config`. Crucial information such as the paths of where saved images & image information is going to be saved is also included. This information can be manipulated with either by the user manually or with the use of specialised commands that are describe in more detail in the section 6.3.3 below.

6.3.3 Useful Features

The end-to-end system has a plethora of features aimed at making the user experience rich and informative.

Saving Images & Folders

The user is able to save segmented and evaluated images and folders by adding the `-save` argument to the end of the evaluation command, this by default will be outputted to a location within the file structure specified by the `.config` file. If the user is not happy with this, they are also able to change the default output path with the use of the `-savefolder` argument in the following manner:

```
python3 malaria_evaluation.py --sf output_folder/
```

the user can utilise this to have control of where the saved evaluated images will be stored.

Saving Image & Folder Information

In addition to saving the images and folders, the user has the ability to save the image and folder information into a text file for later use or interpretation. This is done by adding the `-saveinfo` argument after the image or folder that is to be evaluated in the command line. An example command is shown below:

```
python3 malaria_evaluation.py --i input_image --si im_info.txt
```

This allows the user produce a large amount of information about a specific image or folder of images with ease, this information can then be use for statistical analysis of the end-to-end system.

Choosing Segmentation and Classification Methods

The user also has the option to choose which segmentation method and which classification model to use with each run, this is provided as a input to the software from the user when the user requests an evaluation procedure. This information is also represented in the saved information text file if the user chooses to utilise this feature. This gives the user a better understanding of the strengths, limitation and performance of each model and segmentation method with enough experimentation.

Low Time and Computational Complexity

Low time and computational complexity are considered to be features of our end-to-end system, these are achieved by using a straightforward design implementation and an optimal coding practices. The majority complexity load is utilised by the classification model prediction pipeline as it requires the largest computing power. On low-end systems, which this project is designed to work with, an acceptable time complexity is still able to be achieved making the end-to-end system efficient and responsive.

The aforementioned features discussed were designed to make the end-to-end system relatively straight-forward and easy to use, this was done to reduce the need of specialised personnel in the diagnosis of malaria.

6.4 Testing, Unknown Data & Evaluation

This section analyses the performance of the end-to-end system through testing methodologies, including the use of unknown data, and evaluates the system's strengths and limitations.

We used the UCL dataset to determine the most accurate segmentation and classification method combinations. We compared the results to expert-labelled UCL data, which were identified using the `-si` argument output and command line statistics. This efficient methodology allowed us to observe defining system characteristics and allowed us to conclude which segmentation and classification method combinations work the most accurately in tandem.

Performance of Segmentation

Otsu's method proved most accurate in segmenting red blood cells and performed well under different image conditions due to its adaptive nature. In combination with the Keras sequential model, it produced the closest results to the testing dataset as shown in Figure 6.2. Therefore, it was the best choice for segmentation.

K-means segmentation produced strong results but struggled with over-segmenting certain features, such as light artefacts and colour anomalies. While this method was not performing poorly, in the author's opinion, it was not accurate enough. Figure 6.3 shows an example of K-means segmentation being applied with the best performing classification model.

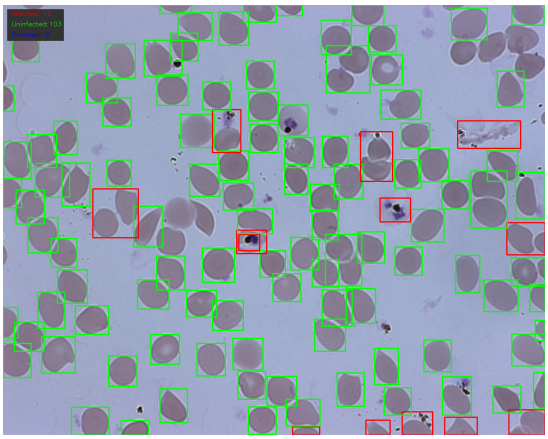


Figure 6.3: K-means & Keras Sequential Evaluated Image.

Performance of Classification models

The Keras sequential model with spatial attention failed to classify any extracted red blood cells, producing anomalous and indistinguishable results, placing them in the 'unknown' category. This result was surprising to the author as the model was theoretically perform the strongest. In contrast, the original Keras sequential model had the highest accuracy in confidently classifying red blood cells, cementing itself as the best-performing model. Evaluation metrics were a strong indicator of this, as the model was close second to the spatial attention model. The VGG16 model produced results of a random nature, this is most likely due to the fact that a very small number trainable parameters represented the parasitized and uninfected red blood cells. This result was however still surprising as better performance was expected by the author. It was therefore not used often, and was not considered a reliable model.

Reaction to Unknown Data

Images *6020*, *6021*, and *6022* from project *P2005* in the CIL were used to represent unknown data for the end-to-end system. This data was evaluated with all segmentation and classification methods, however we were interested in the evaluated result produced with the use of the best-performing combination. The results of this evaluation are shown in Figure 6.4.

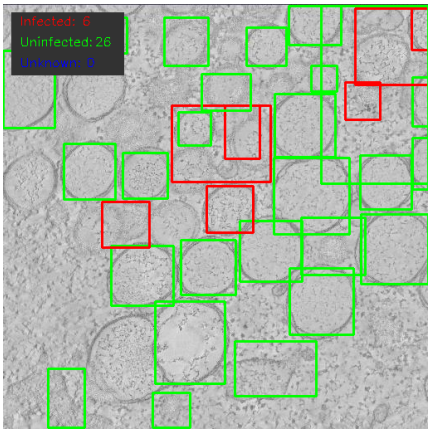


Figure 6.4: Image *6020* evaluated.

Evaluation of the End-to-end System

The end-to-end system performs very well in most configurations however there are a few characteristics of the system that must be investigated. Failure modes of segmentation occur when multiple cells are overlapping or are in close extreme proximity to each other, this causes the cells to be treated as one region of interest which causes the classification model to sometimes produce a False Positive or a False Negative of the infection status. This problem is more common with Otsu's method rather than K-means, and a potential solution for this is to introduce a more detailed edge detection system.

Light conditions and image artefacts play a large role in the accuracy of the end-to-end system, a contour filtering technique is therefore implemented to filter out small light anomalies and large clusters of overlapping cells. This increases the accuracy of the end-to-end system as fewer False Positives and False Negatives are identified.

When the system is given unknown data or data that is foreign to the goal of the evaluation system, it is still able to produce results as shown in Figure 6.4. This shows that the end-to-end system is not able to distinguish between blood smear images and monkey kidney cells such as the ones from the CIL. The main factor in this would be the classification model not being trained for the specific scenario, and therefore, we can confidently deduce that if a correct classification model was to be used with the appropriate input image, it is likely that the end-to-end system would be able to perform to a substantially good standard. This shows the adaptability and scalability of the system, proving that the end-to-end solution can be utilised for a plethora of different evaluation issues.

When considering the overall performance of the end-to-end system, the end-to-end system performs exceptionally well with the use of Otsu's method and the Keras sequential classification model. The system is able to find and distinguish parasitized malaria-ridden red blood cells efficiently with a high degree of accuracy, low time complexity and low computational complexity. Different configurations of the system have varying levels of accuracy however the aforementioned method is the most optimal, efficient, and accurate in assessing blood cells for malaria. Therefore, we can conclude that the end-to-end system is a strong software solution that can be used for assessing blood cells for malaria.

Chapter 7

Project Planning

In this chapter the author discusses his approach to project planning and project development. We discuss the challenges of maintaining momentum, risk identification and management, and adapting to changes in the project development cycle. We also reflect on the achievements, failures and overall performance.

7.1 Project Planning and Execution

7.1.1 Time Management

Effective time management is key when developing a project of this size, and therefore, steps have been taken by the author of this thesis to maintain iterative and efficient progress throughout the development of the project. Steps such as daily iterative work were crucial as these allowed the author to maintain momentum in the development of the project and not lose track of the development objectives. The development of the project has been divided into two major parts, classification was the primary focus in the first academic term and in the second academic term, segmentation, and the end-to-end system were prioritised. This allowed the author to divide the large amount of work that was to be done into digestible sections. A project schedule in the form of Gantt charts was devised to create a guideline for the future of the project and for the author and to keep track of progress, this was also accompanied by project tracking tools such as Jira. These are discussed in section 7.1.2 and 7.3.1 respectively.

7.1.2 Gantt Charts

Gantt charts served as an indispensable tool in visualizing project timelines and monitoring weekly progress throughout the academic terms. Major developmental milestones were clearly mapped out, with estimated lengths of time allocated to each developmental milestone, this was done to incentivise timely completion of important parts in the development cycle. Visualisations of the Gantt charts used for project planning are shown in Appendix D, Table D.1. The inherent benefits of utilizing Gantt charts are not limited to simply tracking progress, but also extend to effectively communicating project objectives in a comprehensive and efficient manner, this is why, the incorporation of Gantt charts proved to be an important and pivotal factor in the successful development of this project.

7.2 Risk Assessment

Risks are an unavoidable and integral part of project planning and development. It is essential to assess risk and implement measures to mitigate them. To achieve this, a risk register was created which can be found in Appendix A. A risk register is essential as it not only outlines potential risks that can occur during the life cycle of a project, but covers a wide range of risks, all the way from technical to personal risks. Therefore, a uniquely tailored risk register was created for this project.

Personal risks such as time allocation were one of the biggest challenges in the planning and development of the project. It was difficult to identify and estimate the time required for the completion of certain features in the project. Academic coursework added to the time allocation problem and therefore some features had to be delayed due to factors such as overworking and lack of time. To mitigate these risks, more comprehensive research was conducted before development began and more effective time management strategies were employed to avoid overworking.

Technical risks were considered however were uncommon as mitigation methods such as backing up important files were carried out to prevent the risk of losing important development progress. Although the author worked on the project on multiple computers, technical risks did not impact the project, but were still kept in mind throughout the development cycle. The risk register proved to be an effective tool in identifying and mitigating potential risks, ensuring the successful completion of the project

7.3 Project Management

7.3.1 Jira

Jira proved to be an invaluable tool in project management as it provided a comprehensive overview of tasks to be completed throughout the academic terms and development cycle. A kanban board was utilised to keep track of project workflow, this is an implementation of Agile practices and methodologies where continual development is key. This approach allowed the author to easily monitor progress and make adjustments where necessary. A cumulative flow diagram, shown in Figure 7.1 shows the development and progress of the project throughout the academic terms [51]. This provides key insight into the pace and progress of the project, and can be used to identify potential problems that the author can learn from in the future. Releases at major milestones were strategically implemented to ensure a clear and organised workflow, as depicted in Appendix D. By utilising Jira, the author was able to successfully keep track of progress and maintain momentum in the development of the project.

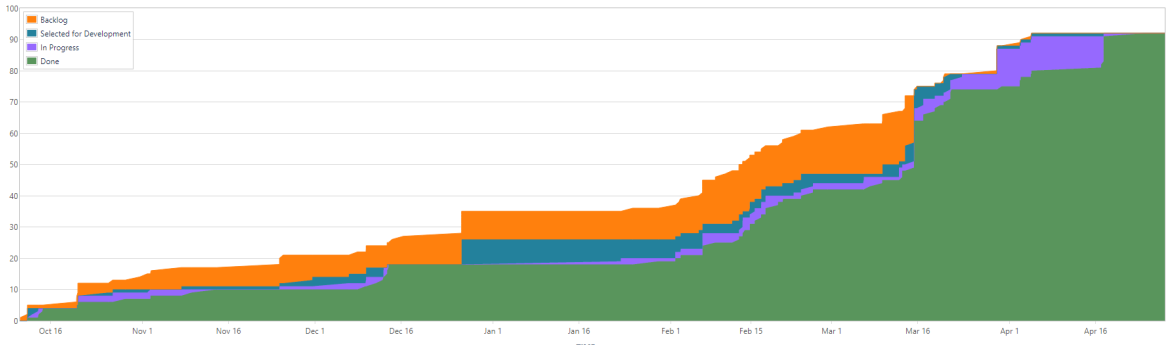


Figure 7.1: Cumulative Flow Diagram.

7.3.2 GitLab

GitLab was used as a repository in which the product was continually being built. Daily and weekly commits were made to the GitLab to track progress throughout academic terms and to back up important files. A total of 133 commits was made throughout the development period of September 29th to April 28th, with an average of 0.6 commits per day [52]. GitLab was an effective tool in tracking progress and backing up important files of the project.

7.4 Reflection on Achievements, Failures and Performance

The journey of developing the project was filled with challenges and victories, however, a large majority of goals that were set by the author were completed. The completion of the classification system at the end of the first term marked a significant milestone and was a great achievement for the project. As the development of the project progressed, the segmentation system and the final end-to-end system were completed by the end of the second academic term, marking another major achievement in the development cycle. These achievements signify the growth and progress of the project and indicate the successful completion of the project, bringing a sense of pride to the author.

The project development journey wasn't without its challenges. Tackling failures was an important part in the development of the project. Most failures have been resolved however not all development methodologies went smoothly, training classification models was tricky to get working in the first place with the author having to resolve numerous issues in the coding process. The segmentation process and compilation of the end-to-end system also enjoyed their fair share of coding problems and the author had to resolve them swiftly in order to stay on track. Fortunately, no major failures occurred during the project, in the development and management aspects.

The author's expectations for the project were high, and the final product delivered met these expectations in terms of quality and standard, this was achieved through hard work and dedication. The author gained valuable experience in the field of machine learning and computer vision, while also learning important lessons in time management, work ethic, and dedication. The author is proud of the final product and is satisfied with the outcome of the project.

Chapter 8

Conclusions

8.1 Summary of Work Undertaken

The work undertaken in this project resulted in the creation of a comprehensive end-to-end software solution that employed cutting-edge machine learning classification and computer vision-based segmentation techniques that work in tandem to classify malaria-infected red blood cells within blood smear images. This was achieved by creating a classification pipeline capable of training different models such as the Keras sequential models and the state-of-the-art VGG16 model which successfully determined the infection status of cells in the end-to-end software solution. The segmentation pipeline used Otsu's method and K-means clustering to process images and isolate ROIs for infection status labelling. By combining the segmentation and classification pipelines, an efficient and informative end-to-end system was developed and an interactive user experience was created to display this information in an informative and efficient manner. The end-to-end software solution was implemented with a plethora of useful features to make the user experience rich and engaging.

8.2 Initial Intention vs. Actual Outcome

The intended product aimed to develop algorithms that can segment out red blood cells in images of microscope slides and then determine whether or not they are infected using machine learning approaches. The ultimate aim of the product is to have the algorithms run on a mobile phone, using its camera to capture the images through a microscope. The majority

of the originally planned tasks were completed by the author; nonetheless, the product was not transferred onto a mobile application platform as intended, rather it was instantiated as a standalone command-line-based program that is capable of evaluating blood smears.

8.3 Future Work

To advance the project further, a smartphone app capable of running the end-to-end system should be developed to enable evaluation on the go, providing an adaptable software solution suitable for endemic and resource-limited areas. Additionally, other future improvements include scaling the software to improve the processing of large-scale data, diversifying the choice of segmentation and classification methods and adapting the classification models to a wider range of training data. Further experimentation with the project's capabilities should also be explored to fully realise its potential.

8.4 Conclusion

In conclusion, it is with great pride that the author presents the end-to-end software solution that has been developed, with the potential to revolutionise the diagnosis of malaria in regions impacted by the disease and those at risk. By providing rapid diagnosis and alleviating resource pressure in endemic areas, this software solution has the ability to make a real difference in the lives of those affected by this disease. Utilizing cutting-edge machine learning and computer vision methodologies, it stands at the forefront of malaria diagnosis and disease research, driving the boundaries of what is possible in this field. The significance of this solution cannot be overstated, as it represents a crucial step in our efforts to eradicate a disease that has plagued humanity for thousands of years. The author's hope is that this technology will continue to evolve, driving us closer and closer to a world free from the burden of malaria.

Appendix A

Risk Register

Activity	Risk	Severity	Probability	Risk Score	Contingency Methods to Manage the Risk
Personal- Lack of time management and time allocation	Unable to finish the project/Underdelivering due to poor time management or poor time allocation	Critical	Low	4	Developing good time management skills and having efficient project productivity sessions
Technical- File corruption	Files not being backed up would lead the program to not be able to function and results would not be produced	Critical	Low	1	Backing up important files and also backing up training data and results would avoid this risk as it could be easily recovered from the cloud or an external hard drive
Technical- Internet not working	No access to the internet would lead the project to come to a halt as development would be extremely limited	Medium	Low	2	Be able to easily relocate the machine to a point where internet is accessible or use cellular data as a replacement
Health and Safety- Carpal Tunnel Syndrome	Injuring your wrist due to overworking or bad posture at your computer	Medium	Low	3	Good posture and regular breaks with wrist exercises allow mitigation against injury
Health and Safety- Eye soreness/Eye pain	Experience soreness in your eyes or sharp pain caused by looking at your screen for too long	Low	Medium	4	Regular breaks and healthy schedules allow for this risk to be mitigated however not always avoided
Technical- Not enough computational resources for the project	The process might not yield accurate results, or results at all if the machine is not able to handle running the program for extended periods of time	Critical	Low	3	Develop the program on a suitable machine to mitigate this risk or optimize the program for machine which possesses less technological resources than the average machine

Figure A.1: Snippet from the Project's Risk Register.

Appendix B

Average Precision Score

Average Precision Score is a computed average from prediction scores. This is represented as AP.

As a reminder, AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$\text{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{B.1}$$

where P_n and R_n are the precision and recall at the *n*th threshold.

This implementation is restricted to the binary classification task or multilabel classification tasks.

Mathematical explanation sourced from [36].

Appendix C

Otsu's Method

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{C.1}$$

In equation the equation above, weights w_0 and w_1 are the probabilities of the two classes separated by the threshold t, and σ_0^2 and σ_1^2 are the variances of the two classes.

The class probability $w_{0,1}(t)$ is computed from L bins of the histogram:

$$w_0(t) = \sum_{i=0}^{t-1} p(i) \tag{C.2}$$

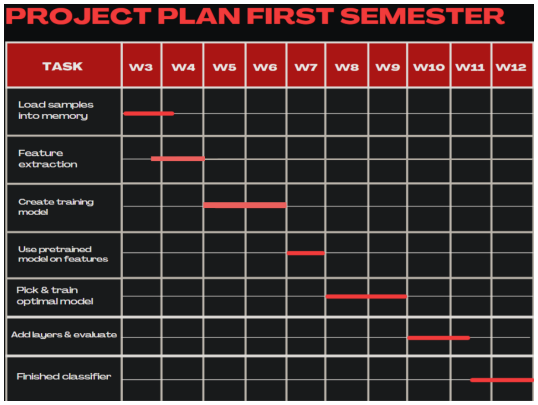
$$w_1(t) = \sum_{i=t}^{L-1} p(i) \tag{C.3}$$

The class probabilities can be computed iteratively. This idea yields an effective algorithm.

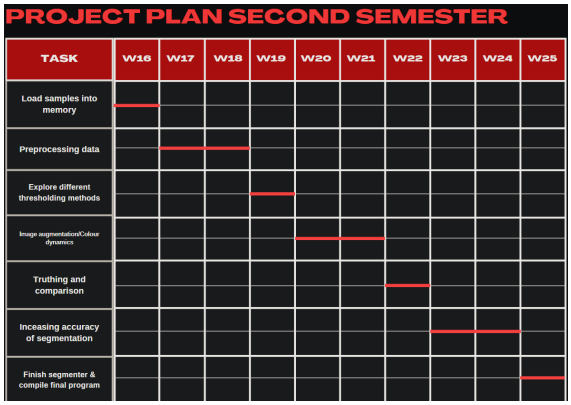
Mathematical explanation sourced from [38].

Appendix D

Project Planning



(a) Semester One Gantt Chart.



(b) Semester Two Gantt Chart.

Figure D.1: Project Planning Gantt Charts.

Version	Status	Progress	Start date	Release date	Description
Classifier complete	RELEASED			14/Dec/22	Release version of functioning classifiers

Figure D.2: Example Jira Release.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[2] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[3] OpenCV, "Open source computer vision library," 2015.

[4] "malaria | tensorflow datasets," TensorFlow. [Online]. Available: https://www.tensorflow.org/datasets/catalog/malaria

[5] "Index of /xfer/." [Online]. Available: http://vase.essex.ac.uk/xfer/

[6] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[7] "The cell image library." [Online]. Available: http://www.cellimagelibrary.org/home

[8] "Malaria - malaria through history | britannica." [Online]. Available: https://www.britannica.com/science/malaria/Malaria-through-history

[9] "Fact sheet about malaria." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/malaria

[10] "Malaria | causes, symptoms, treatment, prevention | britannica." [Online]. Available: https://www.britannica.com/science/malaria

[11] Y. M. Kassim, K. Palaniappan, F. Yang, M. Poostchi, N. Palaniappan, R. J. Maude, S. Antani, and S. Jaeger, "Clustering-based dual deep learning architecture for detecting red blood cells in malaria diagnostic smears," *IEEE journal of biomedical and health informatics*, vol. 25, no. 5, pp. 1735–1746, 05 2021.

[12] N. E. Ross, C. J. Pritchard, D. M. Rubin, and A. G. Dusé, "Automated image processing method for the diagnosis and classification of malaria on thin blood smears," *Medical Biological Engineering Computing*, vol. 44, no. 5, pp. 427–436, 05 2006. [Online]. Available: http://link.springer.com/10.1007/s11517-006-0044-2

[13] S. Kaewkamnerd, C. Uthaipibull, A. Intarapanich, M. Pannarut, S. Chaotheing, and S. Tongsima, "An automatic device for detection and classification of malaria parasite species in thick blood film," *BMC Bioinformatics*, vol. 13, no. S17, p. S18, 12 2012. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-13-S17-S18

[14] "tf.keras.utils.image.dataset.from.directory | tensorflow v2.12.0," TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_directory

[15] "tf.data.dataset | tensorflow v2.12.0," TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/data/Dataset

[16] K. Team, "Keras documentation: Model training apis." [Online]. Available: https://keras.io/api/models/model_training_apis/

[17] "tf.keras.preprocessing.image.imagedatagenerator | tensorflow v2.12.0," TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[18] K. Team, "Keras documentation: Vgg16 and vgg19." [Online]. Available: https://keras.io/api/applications/vgg/

[19] "Convolutional neural network," Wikipedia, 04 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=1149604042

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556v6*, 04 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[21] K. Team, "Keras documentation: Conv2d layer." [Online]. Available: https://keras.io/api/layers/convolution_layers/convolution2d/

[22] "tf.keras.activations.relu | tensorflow v2.12.0," TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/activations/relu

[23] K. Team, "Keras documentation: Maxpooling2d layer." [Online]. Available: https://keras.io/api/layers/pooling_layers/max_pooling2d/

[24] "Keras documentation: Flatten layer." [Online]. Available: https://keras.io/api/layers/reshaping_layers/flatten/

[25] K. Team, "Keras documentation: Dense layer." [Online]. Available: https://keras.io/api/layers/core_layers/dense/

[26] “tf.keras.optimizers.adam | tensorflow v2.12.0,” TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv:1412.6980v9*, 01 2017. [Online]. Available: http://arxiv.org/abs/1412.6980

[28] “tf.keras.losses.binarycrossentropy | tensorflow v2.12.0,” TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy

[29] K. Team, “Keras documentation: Globalaveragepooling2d layer.” [Online]. Available: https://keras.io/api/layers/pooling_layers/global_average_pooling2d/

[30] “Keras documentation: Dropout layer.” [Online]. Available: https://keras.io/api/layers/regularization_layers/dropout/

[31] “Imagenet.” [Online]. Available: https://www.image-net.org/

[32] “Softmax function,” Wikipedia, 04 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=1149309173

[33] K. Team, “Keras documentation: The model class.” [Online]. Available: https://keras.io/api/models/model/

[34] “tf.keras.losses.categoricalcrossentropy | tensorflow v2.12.0,” TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy

[35] M. AMER, “Classification evaluation metrics: Accuracy, precision, recall, and f1 visually explained,” Context by Cohere, 06 2022. [Online]. Available: https://txt.cohere.com/classification-eval-metrics/

[36] “sklearn.metrics.average-precision-score,” scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html#rcdf8f32d7f9d-1

[37] “The cell image library project p2005,” cellimagelibrary.org. [Online]. Available: http://cellimagelibrary.org/project/P2005

[38] “Otsu’s method,” Wikipedia, 03 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Otsu%27s_method&oldid=1147002604

[39] “Opencv: Image file reading and writing.” [Online]. Available: https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56

[40] “Opencv: Smoothing images.” [Online]. Available: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

[41] “Gaussian noise.” [Online]. Available: https://en.wikipedia.org/w/index.php?title=Gaussian_noise&oldid=1143830434

[42] “Opencv: Color space conversions.” [Online]. Available: https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab

[43] “K-means clustering,” Wikipedia, 04 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=1149489138

[44] “Cielab color space,” Wikipedia, 04 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=CIELAB_color_space&oldid=1149341107

[45] “numpy.reshape — numpy v1.24 manual.” [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.reshape.html

[46] “Opencv: Image thresholding,” docs.opencv.org. [Online]. Available: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

[47] W. Contributors, “k-means clustering,” Wikipedia, 02 2019. [Online]. Available: https://en.wikipedia.org/wiki/K-means_clustering

[48] S. learn home, “scikit-learn: machine learning in python — scikit-learn 0.20.3 documentation,” Scikit-learn.org, 2019. [Online]. Available: https://scikit-learn.org/stable/index.html

[49] s.-l. K-mean, “sklearn.cluster.kmeans — scikit-learn 0.21.3 documentation,” Scikit-learn.org, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[50] “Opencv: Structural analysis and shape descriptors,” docs.opencv.org. [Online]. Available: https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f260e0

[51] “Cumulative flow diagram,” Essex.ac.uk, 2023. [Online]. Available: https://cseejira.essex.ac.uk/secure/RapidBoard.jspa?rapidView=6638&projectKey=C301001&view=reporting&chart=cumulativeFlowDiagram&swimlane=10974&swimlane=10975&column=28635&column=28636&column=28637&column=28638

[52] “Repository analytics,” GitLab. [Online]. Available: https://cseegit.essex.ac.uk/22-23-ce301/22-23_CE301_tazar_tomas/-/graphs/master/charts