ASHESI UNIVERSITY COLLEGE

CLASSIFICATION AND QUANTIFICATION OF MALARIA PARASITES USING CONVOLUTIONAL NEURAL NETWORKS

UNDERGRADUATE THESIS

B.Sc. Computer Science

Maxwell Mbabilla Aladago

2018

ASHESI UNIVERSITY COLLEGE

Classification and Quantification of Malaria Parasites Using Convolutional Neural Networks

UNDERGRADUATE THESIS

Thesis submitted to the Department of Computer Science, Ashesi University College in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science

Maxwell Mbabilla Aladago

April 2018

DECLARATION

I hereby declare that this Undergraduate Thesis is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

.....

Candidate's Name:

.....

Date:

.....

I hereby declare that preparation and presentation of this Undergraduate Thesis were supervised in accordance with the guidelines on supervision of Undergraduate Thesis laid down by Ashesi University College.

Supervisor's Signature:

.....

Supervisor's Name

.....

Date:

.....

Acknowledgment

This thesis was completed through the selfless contributions of many people. As a result, I cannot acknowledge everyone individually here. Nonetheless, I would like to recognize with profound gratitude, some individuals for their specific contributions to the success of this thesis.

Foremost, I would like to express my sincere gratitude to my advisor Dr. Elena V. Rosca and to Dr. Lorenzo Torresani for supporting, motivating and providing constructive criticism throughout this research. Dr. Rosca's vast network in the research community made it possible for me to obtain the much needed data for conducting this research, while Dr. Torresani's immense experience and knowledge in machine learning helped improved the thesis phenomenally.

Further, I would like to thank Dr. Linda Amoah and the Noguchi Memorial Institute for Medical Research for generously providing the data used in this research. I would also like to sincerely thank Ms. Urmila Sampathkumar at the University of Missouri for freely making available to me the tool used in annotating the images. I am also highly indebted to Dr. Etoka-Beka Kosso Mandingha in Congo Brazzaville for selflessly dedicating her time and expertise to annotate the images for me at no cost. Her knowledge in molecular biology and applied immunology helped me make progress on this thesis. Finally, I want to thank Dr. Stefan Jaeger at the US National Library of Medicine for introducing me to Ms. Urmila Sampathkumar

Abstract

Malaria is currently one of the most deadly diseases in the world. While there are different treatment methods for the disease, the search for new drugs against malaria is still a very important area of research. One of the main challenges in manufacturing drugs against malaria is efficiently evaluating the performance of the drugs on the parasites since it requires, amongst others, precise measurements of the parasite growth-stages as well as their counts in blood smear images. The current gold-standard for making such detail diagnosis is manual microscopy which is tedious. This research showed that convolutional neural networks can be used to identify the different growth-cycle stages of *Plasmodium* parasites, even in situations where there is little data. Employing a variety of data augmentation techniques and transfer learning, a semantic segmentation model was built to discriminate between trophozoites, gametocytes and normal red blood cells with an accuracy of 85.86% in 353 Giemsa-stained thin blood smears. The results showed that it is possible to perform dense predictions on Giemsa-stained thin blood smears using convolutional neural networks.

Keywords:

Convolutional neural networks; machine learning; malaria; Plasmodium parasites

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction And Background

Nearly 50% of the world's population is at risk of malaria, a dangerous infectious disease caused by the *Plasmodium* parasite. Literally, a child under five years dies of malaria every two minutes. About 429,000 people died from the disease in 2015; sub-Saharan Africa alone accounted for 90% of all deaths (WHO, 2017). These are very shocking statistics, considering that different treatment methods of the disease exist. Besides the loss of lives, malaria creates other numerous economic and social problems to many countries, especially to developing countries where the disease is endemic. As an example, Ghanaian businesses suffered losses of up to US$6.58 million in 2014 to malaria (Nonvignon et al., 2016). Yet, these statistics might just be a microcosm of the problems malaria poses to societies as a lot more cases go unreported.

Malaria in human beings is caused by four main species of the *Plasmodium* parasite: *Plasmodium falciparum, Plasmodium vivax, Plasmodium malariae* and *Plasmodium ovale*. Of these four major species, *Plasmodium falciparum* is the most prevalent parasite in Africa. It is also the species responsible for the most severe form of malaria (WHO, 2017). Like many other organisms, *Plasmodium* parasites have different growth stages. In humans beings, the parasites have four major growth-cycle stages *sporozoites, merozoites, trophozoites* and *gametocytes* (Srinivas, 2015). There are other stages of the parasites which occur outside the human body.

As part of the measures to mitigate the impact of malaria on the progress of societies, the World Health Organization(WHO) directed an estimated US$2.7 billion in 2016 in funding for malaria control and elimination research projects (WHO, 2017). WHO's efforts have mainly been in vector control which has yielded the greatest success in controlling malaria so far (Karunamoorthi, 2011). However, while vector control measures are effective in reducing the transmission of the disease from infected people to uninfected people, they are not effective at helping to eradicate the *Plasmodium* parasites. This is largely due to vector resistance. The female *Anopheles* mosquito, the main vector of the disease, is known for its ability to build resistance against control measures such as its resistance to in-

secticides in sub-Saharan Africa (WHO, 2017). Consequently, increasingly more complex control measures are required to prevent the spread of the disease.

The limitations of vector control measures call for preventive measures as well as control measures to fight the disease. In this regard, good anti-malarial drugs have been identified as quintessential in the ultimate eradication of malaria (White, 2008). Research and manufacture of effective anti-malarial drugs is therefore important, especially if WHO is to achieve its target of reducing malaria deaths by at least 90% by 2030 (WHO, 2017). However, the multiple incidences of drug resistance such as the *Plasmodium* parasites resistance to chloroquine[1] in the 1950s (Wellems & Plowe, 2001), make the research and development of anti-malarial drugs a challenging task (Ojha & Roy, 2015). Nonetheless, research for anti-malarial drugs can be conducted effectively with the right tools, such as with automated systems, as discussed later in this paper.

The process of developing anti-malarial drugs can be described broadly as an iterative assessment of the effectiveness of some chemicals at killing *Plasmodium* parasites. That is, proposed anti-malarial drugs are applied to *Plasmodium* parasites and the number of parasites that are killed by the drugs is assessed. In this process, counting the number of parasites accurately is crucial in the assessment of the proposed drugs samples. Besides assessing how the parasites react in general to anti-malarial drugs, it is equally important to measure the impact of the drugs on the different growth-cycle stages. For instance, instead of killing the parasites, partially effective drugs can increase the production of gametocytes, the sexual stage of the parasite (White, 2008). Thus, accurately identifying the growth-cycle stages of the parasites is also necessary for discriminating the performance of different proposed anti-malaria drugs.

Yet, despite the salient importance of obtaining accurate results of the two assessments above, the only currently available method for conducting them is manual microscopy (Pollak, Houri-Yafin, & Salpeter, 2017). Sadly, manual microscopy, the benchmark method for testing the presence of *Plasmodium* parasites in a stained image (Pollak et al., 2017), encounters many problems from human errors in practice. Manual microscopy involves

[1]Chloroquine is a chemotherapeutic drug used for treating malaria. Developed in 1947, some species of *Plasmodium falciparum* developed resistance against the drug after three years.

observing a slide of an appropriately stained image under a powerful microscope and then determining whether there are *Plasmodium* parasites on the image or not. The image could be prepared from a sample of blood or from cultured parasites in a laboratory. The reason why microscopy works is that, *Plasmodium* parasites assume a different color from red blood cells(RBCs) when a suitable stain is applied to a blood sample containing the parasites. The color differences and the shapes can then be used to distinguish the parasites from the rest of the blood components. The shape of the parasite is especially important when information of the growth-stage of the parasite is desired. The staining methods as well and the parasite growth-cycle stages are covered in detail in chapter 2.

Estimating parasitaemia[2] in microscopy is achieved by manually counting the number of parasites on different fields of the slide. The values from the various receptive fields are then used to make a global estimate of parasite density in the blood sample. This process is tedious and time consuming. Manual microscopy is also prone to errors because the accuracy of the exercise is dependent on the expertise of the pathologist[3] performing the examination. Unsurprisingly, inaccurate diagnosis are quite frequent in manual microscopy due to the inability of some pathologists to properly diagnose malaria. Kahama-Maro *et al.* (2011) showed that the sensitivity of routine microscopy is about 71.4% whilst its specificity is only 47.3%. Compared with expert microscopists, the accuracy of typical microscopists is between 45% to 60% (Pollak et al., 2017). These statistics show that manual microscopy is not an effective method for research purposes, especially in places where there are only few expert pathologists.

In addition, Rapid Diagnostic Tests (RDTs) are other techniques for testing the presence of *Plasmodium* parasites in a blood sample. RDTs are relatively easy to perform and produce results faster compared to microscopy (WHO, 2015). Despite these advantages over microscopy, RDTs are not used for research purposes because they do not provide parasite species and growth-cycle identification nor offer any estimates of parasitaemia level. As such, RDTs are not discussed further in this paper.

[2]Parasitaemia is the ratio of the parasite infected RBCs to the total number of RBCs

[3]A pathologist is a specialist who perform disease diagnosis by often conducting tests in a medical laboratory

The downsides of manual microscopy as explained above make a case for a more reliable, independent, fast and accurate method for identifying the growth-cycle stages of *Plasmodium* parasites, as well as estimating the parasitaemia in blood smear. The importance of accurate results in terms of parasite count in the development of anti-malarial drugs cannot be over emphasized as it gives an objective measure of the effectiveness of the new drugs. Automated microscopy can offer not only accurate results but also consistent results due to its independence from the expertise of the person conducting the test. Computer vision methods are the means to intelligent malaria diagnosis systems.

Computer vision methods aimed at automating malaria diagnosis is currently an active area of research (Roy, Sharmin, Mufiz Mukta, & Sen, 2018; Bibin, Nair, & Punitha, 2017; Pinkaew, Limpiti, & Trirat, 2015; Ghosh, Ghosh, & Kundu, 2014). However, so far, the focus of automated malaria diagnosis has been on developing intelligent systems for use in hospitals. There has been minimal efforts at developing machine learning methods for use in *Plasmodium* parasites research laboratories. As a result, most of the methods are mostly qualitative diagnosis. Although a few attempts have been made to identify parasite species, as well as parasite growth-cycle stages (Nanoti, Jain, Gupta, & Vyas, 2016; Moon et al., 2013), there has not been any attempt (to the best of my knowledge) to count the life-cycle stages of parasites. This project aims to fill that gap by providing techniques to identify and count the growth-cycle stages of *Plasmodium* parasites on digitized images.

The main objective of the research is to improve the efficiency of developing anti-malarial drugs. To achieve this objective, the project employs convolutional neural networks (CNNs), a variant of deep learning to classify and count growth-cycle stages of *Plasmodium falciparum* parasites on Giemsa-stained images.

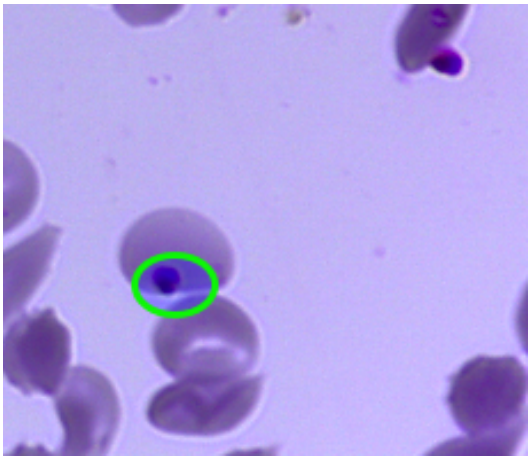
Chapter 2: Related Literature

2.1 Malaria and *Plasmodium* Parasites

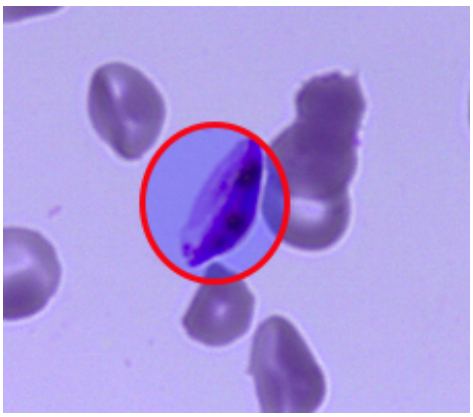
This study involved only *Plasmodium falciparum* because it is the most prevalent species of *Plasmodium* in Africa. Consequently, the brief review of *Plasmodium* parasites as presented in this chapter is in the context of the species *P.falciparum*. In addition, in this chapter and in the rest of paper, *Plasmodium* parasites can be used loosely to refer to *P. falciparum*. Finally, for brevity, parasites is used in placed of *Plasmodium* parasites whenever the meaning is unambiguous.

As discussed in chapter 1, there are four major stages of growth of *Plasmodium* parasites in human beings. Sporozoites are the first stage of the parasites in humans. When matured, they rapture releasing merozoites into the blood stream. Merozoites then invade RBCs and produce trophozoites through asexual reproduction. The trophozoites develop into schizonts which also burst and release merozoites and gametocytes. The merozoites re-invade other RBCs while the gametocytes wait to be transfered into a mosquito during a blood meal by the later (Srinivas, 2015). Although gametocytes do not cause malaria infection directly, they aid in the transmission of the disease. Therefore, it is important for new therapeutics to be able to target all the life stages of the parasite (Delves et al., 2012). The ability to get an accurate count of each of the parasite life stages is essential in testing these therapeutics.

The various stages of the parasite have different shapes and survival mechanisms (Srinivas, 2015). Gametocytes have crescent shapes like a half-moon. Trophozoites have circular shapes and are often located completely inside RBCs. The unique characteristics of the different growth-cycle stages of the parasites can be revealed through staining. Images of trophozoites and gametocytes are shown in Figure 2.1.



(a) The blue marking depicts a trophozoite, one of the early life-cycle stages. As in the picture, it is almost circular in shape.



(b) The red marking depicts a gametocyte in its advanced stages. Gametocytes generally have elongated shapes as shown in the picture.

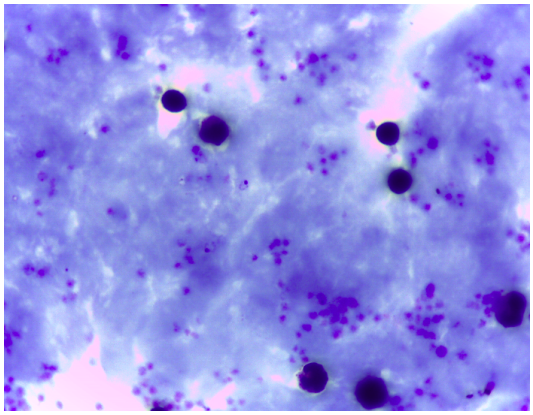
Figure 2.1: *Plasmodium* parasites growth-cycle stages

2.2 Staining

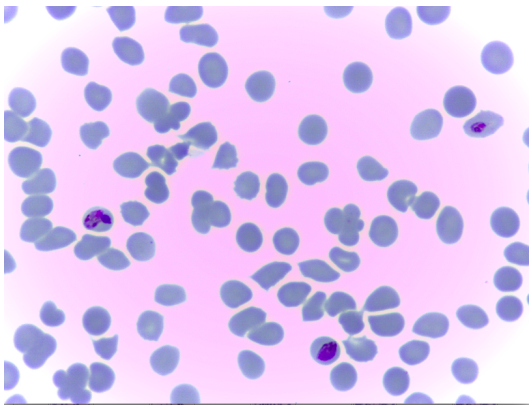
Staining is a technique which is often used to enhance visualization and identification of *Plasmodium* parasites in manual microscopy. A correctly stained blood smear is crucial in malaria diagnosis, especially when precise identification of growth-cycle stages is desired (Gonzales, 2016). Although there are other staining substances and techniques such as Leishman (Sathpathi et al., 2014) and DAPI (Moon et al., 2013), Giemsa is the standard and most reliable stain for all forms of blood smears. Examples of Giemsa-stained images (thin and thick blood smears) are shown in Figure 2.2 below. Thick blood smears can detect a few parasites in a blood film. However, their preparation process destroys the RBCs. Hence, thick smears are mostly used for *Yes* or *No* diagnoses. On the other hand, thin blood smears as shown in Figure 2.2b preserve the structure of the RBCs and the parasites. This property makes thin smears the suitable method for diagnosing malaria when the species or the life-cycle of the parasites is needed. This project used thin blood smears for the parasites growth-cycle stages detection.

2.3 Machine Learning Algorithms for Automated Malaria Diagnosis

Image processing and pattern recognition has been at the center of research in computer vision with regards to improving the accuracy of identifying *Plasmodium* parasites in



(a) Thick blood smear - in the thick blood smear, all the cells including the structure of the parasites are destroyed. It is easy to prepare thick smears compared to thin smears but they cannot be used for very informative diagnosis.



(b) Thin blood smear - in the thin blood smear, all the red blood cells, as well as the true form of the parasites are maintained. Also, thin blood smears expose artifacts such platelets and chemical compounds.

Figure 2.2: Example images of Giemsa-stained blood smears

digitized images (Bibin et al., 2017; Rosado, Correia, Elias, & Cardoso, 2016; Nanoti et al., 2016; Park, Rinehart, Walzer, Ashley Chi, & Wax, 2016; Mehrjou, Abbasian, & Izadi, 2013). These algorithms work by learning patterns in mostly labeled data. They then use the knowledge gathered to make informed predictions about the presence or absence of parasites in datasets the algorithms have not seen before. These approaches yield consistent diagnosis and have been shown to even outperform human beings in some classification tasks (He, Zhang, Ren, & Sun, 2015). Therefore, if applied properly, computer aided malaria diagnosis can address the problems associated with manual microscopy.

One of the factors contributing to the success of computer vision applications is the availability of large labeled data for training the algorithms (Brynjolfsson & Andrew, 2017). However, achieving optimum performance of the algorithms on the data is not trivial. It typically involves making many design choices about the algorithms as well as the formatting and presentation of the data. Data preprocessing is the first step in many approaches with regards to data preparation tasks. Preprocessing tasks usually involve tasks such as data cleaning and other transformations such as conversion to grayscale.

Preprocessing is important in computer vision applications for the identification and counting of *Plasmodium* parasites on digitized stained images. This is because the images

usually contain other foreign objects such as platelets which are also stained during slide preparation. Removing these artifacts before classification improves the accuracy of the diagnosis. It is also crucial to perform other preprocessing functions such as luminance normalizations (Díaz, González, & Romero, 2009) to reduce the impact of of illuminations of the images on the performance of the algorithms. In research laboratories, however, the problem of foreign objects on images is minimal since the data is often prepared from cultured samples.

The color and morphological properties(shape) of the objects in the images are the primary parameters of interest to the machine learning algorithms. Therefore, it is important to use good feature generation techniques to extract the relevant properties of the data. However, this is only an important task when using traditional machine learning algorithms[4]. Research in computer vision malaria diagnosis using traditional machine learning algorithms also generally convert red green blue (RGB) images to grayscale (Rosado et al., 2016; Mehrjou et al., 2013) to reduce the complexity of the images. Some of these approaches in automated malaria diagnosis are reviewed in the succeeding paragraphs.

Due to the large number of studies in automated malaria diagnosis which have appeared in the literature in the last ten years, it is impractical to conduct a detail review of all the proposals. As such, this section presents only an overview of a few of the proposed systems. However, a full review of intelligent malaria diagnosis systems can be found at Poostchi *et al.* (2018).

Ghosh *et al.* (2014) developed a method for estimating parasitaemia in thin blood smears using mainly statistical and mathematical techniques. This involved computing the features of the images such as centroids, mean and variance of the different regions under the assumption that the images were positive RGB images. They then fed these features to the algorithm for classification. Although Gosh *et al.* did not include any performance metrics in their paper, they demonstrated that parasitaemia in thin blood smears can be measured automatically using computers. However, the performance of the algorithm relied on the good engineering of the features. Thus, their method cannot be applied to images where

[4]Traditional machine learning algorithms are standard algorithms such as KNN, SVMs which are not part of deep learning

it is difficult to construct the features manually. Finally, their approach cannot be used for estimating parasitaemia in images which may not have parasites at all.

Unlike Gosh *et al.*, Bibin *et al.* used deep belief networks pre-trained with Restricted Boltzmann Machines (RBMs) to detect malaria parasites in digital blood smear images. Their approach resulted in an F-score of 89.66% and a specificity of 97.60% on a binary classification of images into parasites and non-parasites[5]. These scores are indications that deep networks can classify parasites in digitized images with high accuracies. Support vector machines (SVMs) are by far the most used machine learning algorithms for malaria diagnosis studies due to their simplicity (Rosado et al., 2016; Pinkaew et al., 2015; Linder et al., 2014; Savkare, 2011). Like the method employed by Gosh *et al.*, SVMs require explicit feature generation from the images. Pinkaew *et al.* for instance, extracted the kurtosis, mean, standard deviation, skewness and entropy from four color channels of the images which were then projected to the SVMs to classify between *Plasmodium falciparum* and *Plasmodium vivax* species. Like DBNs, SVMs perform well on non-linear data due to their ability to transform lower dimensional data into higher dimensions using kernels. However, they are limited by the fact that they can typically only perform binary classification tasks. Adapting SVMs for multi-class classification problems such as the classification of the growth-cycle stages of *Plasmodium* parasites is time inefficient and prone to errors.

Moon *et al.* (2013) developed an approach to identify the various growth-cycle stages of malaria parasites as well as discriminate between dead and active parasites in drug-treated samples. Their algorithm used the intensity of DAPI-stained parasite nuclei spots in fluorescence images to identify infected RBCs. The algorithm also detected and quantized three different growth-cycle stages of the *Plasmodium* parasite(rings, trophozoites, schizonts). The gametocyte stage was not considered in this study. In computing the total number of RBCs per image field, the average area of isolated RBCs is first calculated. Afterwards, the number of RBCs in each cluster is estimated from the ratio of the area of the cluster to the average area of each RBC. The combined total of the isolated RBCs and sum

[5]F-score is an average of the precision and recall values. Precision is the number of true positives divided by the number of positives predicted by the classifier. Recall is a measure of the sensitivity of the classifier. Specificity is a measure of the true negative rate. It denotes the percentage of true negatives were correctly classified

of RBCs in all clustered regions is taken as the definitive RBCs count of the image field under consideration. This approach saves a lot of time in estimating the RBCs count since no work is done to untangle clustered RBCs. The diagnosis process involved identifying infected RBCs and then estimating the parasite density. The proposed method achieved a 100% accuracy when tested on a dataset of nine images. However, the number of images was too small to consider the performance of the algorithm conclusive.

A genetic programming algorithm has also been used for the classification of *Plasmodium* parasites on thick-blood smear images (Purnama, Rahmanti, & Purnomo, 2013). Purnama *et al.* provided a more complete model compared to the studies above. Their system included functionality for identifying the species of the *Plasmodium* parasite (*P. falciparum, P.vivax, P.ovale and P.malarae*) responsible for a particular infection. The genetic algorithm approach also identified the predominant life-cycle stage(schizonts and trophozoites, gametocytes) present in the image at the time of diagnosis. Whilst their model had different phases, it also required a lot of pre-processed features. Hence, even with a classification accuracy of 94.82%, it is still exposed to the same problems traditional machine learning algorithms encounter at feature extraction.

Other machine learning algorithms which have been used in malaria diagnosis are K-Nearest Neighbors classifiers(KNN) and linear discriminant classification (LDC) (Nanoti et al., 2016). Nanoti *et al.* for instance, used a two stage KNN method to identify the growth-cycle stages of the parasites with a 90.17% accuracy. Other researchers used more morphological techniques in the automated systems compared to those explored above (Linder et al., 2014; Kareem, Kale, & Morling, 2012). In the case of Kareem *et al.* (2012), they used morphological filtering to remove foreign objects from the images. Their main classification of the images was also composed of both morphological approaches and properties of the images such as color histogram and intensity. The system achieved an accuracy of 87% on a qualitative diagnosis of Giemsa-stained *Plasmodium falciparum* parasites.

Chakrabortya (2015) used unsupervised machine learning algorithms to classify *Plasmodium vivax* parasites in Jaswant Singh Battacharya (JSB)-stained thick blood smear images. Employing color based pixel discrimination methods for the classification process, the

system showed 94.5% accuracy and 0.10% false positive rate on 75 images. Unsupervised machine learning techniques are particularly useful for dealing with small labeled training datasets. Other techniques such as data augmentation and transfer learning have also been proposed to deal with data sparsity.

2.3.1 Convolutional Neural Networks for Malaria Diagnosis

The challenge with using traditional machine learning algorithms such as those in the literature discussed above is that they need hand-engineered features. Unfortunately, constructing these features is not always easy. Hand-engineering might also result in the lost of useful information to the classification task. Deep learning methods address this problem by performing end-to-end learning. That is, they learn the features from the raw data in addition to performing classification task. Convolutional neural networks (CNNs) is a type of deep learning specialized for image analysis. In particular, CNNs preserve the spatial structure of the images. This subsection discusses the literature in automated malaria diagnosis using convolutional neural networks.

Since Alex Krizhevsky won the ImageNet challenge in 2012 (Krizhevsky, Sutskever, & Geoffrey E., 2012) with the famous AlexNet, deep learning has been applied to a wide range of problems including self-driving cars, speech recognition, natural language processing and disease diagnosis. Quinn *et al.* (2016) proposed one of the first computer vision malaria diagnosis systems using CNNs. Their approach involved annotating thick blood smears captured using a custom-made microscope smart phone adapter. The annotations were used as positive samples. They generated negative samples by randomly selecting regions of the images not intersecting any annotation. Their method obtained a 1.00 true positive rate and 0.97 precision recall on receiver operator characteristics (ROC) curves. Like most studies in malaria diagnosis, Quinn *et al.* investigated only the binary classification problem using thick blood smears.

A CNN-based application capable of diagnosing *P.falciparum* in thick blood smears at competency level 1 in the WHO external competency assessment was proposed in (Mehanian et al., 2017). Through the use of novel computer vision tricks such as data augmentation,

global white balancing, adaptive non-linear grayscaling and local thresholding, the prototype application achieved acceptable accuracy in parasite quantization required for drug resistance studies. Although the application used a variety of deep learning techniques and was the first of its kind to use CNNs on large datasets(5,707,947 field-of-views), the CNNs were used for feature extraction only. Using CNNs for only feature extraction increases the complexity of the algorithm as separate models must be developed for the classification and the quantification tasks. By using thick blood films, the application do not apply to the multi-class task of parasite growth-cycle stages classification.

Gopakumar *et al.* (2017) proposed a completely automated system capable of performing quantitative malaria diagnosis. The system uses CNNs on focus stack images acquired from a US$1500 custom-built slide scanner. By processing patches of the images, the CNNs eliminated the need for hand-engineered features. This yielded 97 .06% sensitivity and 98.50% specificity accuracies for object detection in the binary classification setting. One of the main contributions of Gopakumar *et al.* is that they showed CNNs can still attain good performance by processing only patches of the image. However, conventional methods such as SVMs were used in generating the suspected patches for the CNNs to operate on. Thus the performance of the CNN was limited by the accuracy of the SVMs in generating good patches.

With a dataset of 363 images, Penas *et al.* (2017) obtained a 92.4% for *Plasmodium* parasite detection using CNNs. Their model also managed to discriminate between *P.falciparum* and *P.vivax* species at 87.9% accuracy. Penas *et al.* showed that even with small datasets CNNs can obtain good scores on malaria diagnosis tasks using data augmentation and transfer learning,. Although they used thin blood smears they focus on only binary classification. Nonetheless, they showed that CNNs can be effective on thin blood smears.

2.4 Summary of Related Literature

Thus far, the literature shows that CNNs outperform conventional machine learning algorithms such as KNN in automated malaria diagnosis. This is because CNNs generally

learn the features from the raw images which often result in effective classification. The problem with using CNNs, however, is that they often require thousands of images for good performance. Such large datasets are not commonly available especially in medical imaging. Novel techniques such as transfer learning, data augmentation and context encoding are often used for tasks where the datasets are small.

The literature also shows that using CNNs for malaria diagnosis has gained much prominence in recent years[2015-2018]. However, like the conventional machine learning algorithms, CNNs have mostly been applied to the binary classification task as far as malaria diagnosis is concerned. Although a few models experimented discriminating between *Plasmodium* species, no CNN-based method has been used for parasite growth-cycle identification. This research (to the best of my knowledge) is the first to investigate using CNNs for the classification and quantification of the growth-cycle stages of *P.falciparum* on thin blood smears. The novelty of this approach is that it combines the advantages of data augmentation, unsupervised pre-training and transfer learning to perform a semantic segmentation on thin blood smears given very little data. As far as I know, semantic segmentation is yet to be applied to *Plasmodium* parasites detection.

Chapter 3: Methodology

The main hypothesis of this thesis was that CNNs can be used to identify and count the growth-cycle stages of *Plasmodium* parasites with high accuracies. This section discusses the steps which were taken to evaluate the above proposition. Before discussing the machine learning techniques used in the application, a broad overview of the datasets used in the application will be described first.

3.1 Datasets

Three different datasets were used for this research. One of the datasets was used to experiment the binary classification task (Yes or No diagnosis), another dataset for context encoding and the third dataset was used for parasite growth-cycle classification. The datasets for the binary classification and growth-cycle stages classification were obtained from the Noguchi Memorial Institute for Medical Research. The images were taken by malaria research specialists under laboratory conditions using a $\times 100$ magnification light microscope. The dataset used for the context encoding was retrieved from (Quinn et al., 2016). Table 3.1 below shows a summary of the three different datasets.

Table 3.1: Datasets

Task	Type	Number of Images	Positive Images	Negative
Binary classification	mixed	458	351	107
Context Encoding	thick smears	2,703	2, 703	0
Growth-cycle classification	thin smears	403	353	50

Of the 458 images for the binary classification, 351 images were positive samples while 107 images were negative samples. The difference in sample sizes was compensated for through data augmentation. The positive samples were Giemsa-stained thin blood smears but the negative samples were from varied sources. The differences in contexts between the two classes had severe implications on the performance of the model some of which discussed extensively in Chapter 5.

All the four major growth-cycle stages were represented in the dataset for the parasite growth-cycle stage classification. However, only two of the stages(trophozoites and gametocytes) were considered for classification. This was because the schizonts could not be differentiated from trophozoites in some images due to poor image quality. This issue was addressed by considering all schizonts as trophozoites. The details of the annotations of the images for the classification is discussed later in this chapter. Some of the images for the growth-stage classification are shown in Figure 3.1 below.

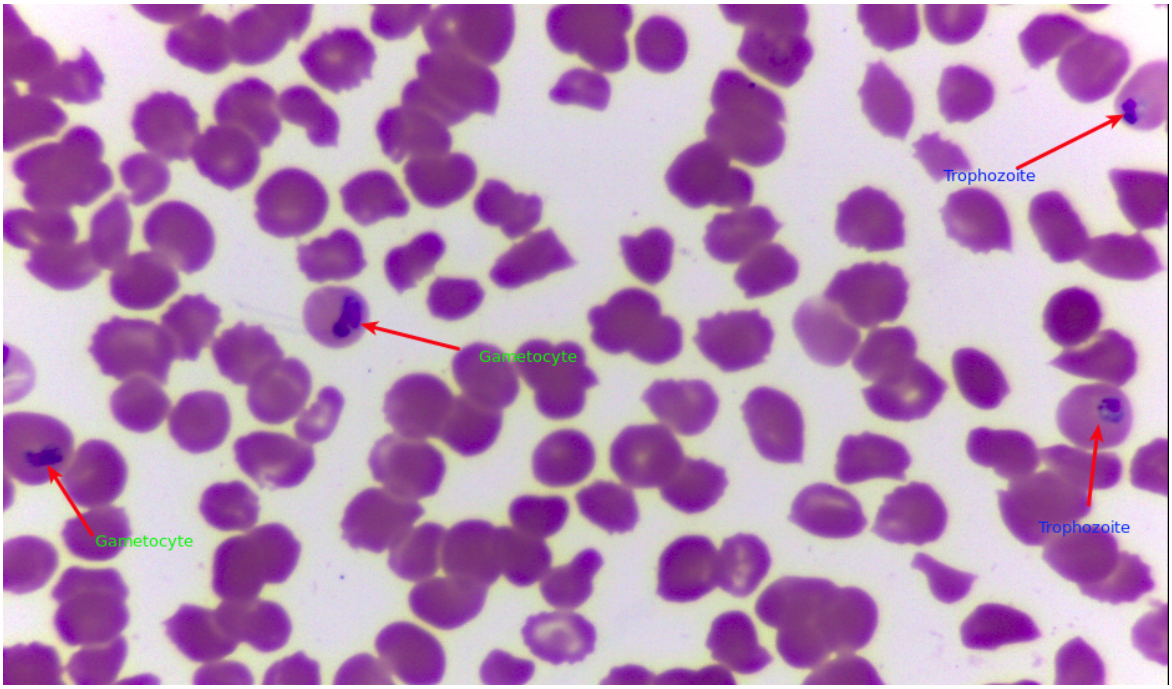


Figure 3.1: *P.falciparum* parasites in thin blood smear image

3.2 Data Processing Tasks

Automated malaria diagnosis systems usually start with preprocessing functions such as foreign objects removal from the images. These cleaning tasks often result in improved performance of the algorithms. I did not conduct such preprocessing tasks because most of the images were prepared from cultured parasites. As such, the images do not contain white blood cells, platelets or chemical substance as is often the case for images taken in clinical settings. Also, the methods employed in this thesis work with only local regions of the images. Thus, the presence of stains or debris at other regions of the images do not

affect the performance of the algorithm. Beyond basic data cleaning, however, advance data processing functions were applied to the images. The techniques employed included parasite annotations for the growth-cycle classification task and gray-scaling for the binary classification task.

3.2.1 Image Annotations

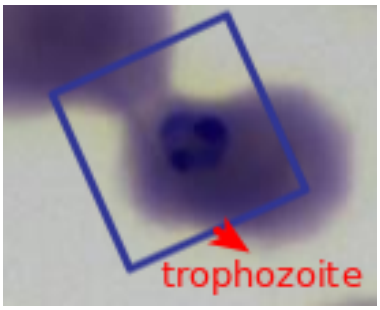
A typical image in the dataset could contain more than one growth-cycle stage of the parasite. For those images, assigning a label to them is ambiguous; it could take the label of any of the growth-cycle stages present in it. Instead of assigning labels to the image, labels were assigned to sections of the image. In particular, square bounding boxes were drawn around parasites. Each bounding box was then assigned the ground truth label of the growth-cycle stage in that region. Negatives patches were taken from random regions of the images not intersecting any bounding box.

The annotations were performed by a malaria research expert using a tool called DragonFly. The tool was generously provided by a researcher at the University of Missouri. Trophozoites were annotated with blue bounding boxes and gametocytes were annotated with brown bounding boxes. Figure 3.2 shows examples of the annotations.

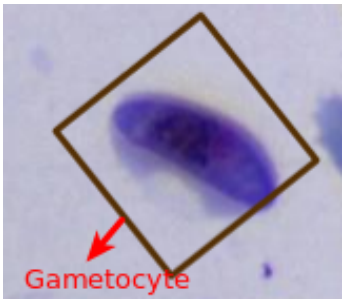
A total of 1068 annotations were generated from 353 images. The other 50 images for the growth-cycle stage classification were completely negative (did not contain any parasite). Of the 1068 annotations, 614 were for trophozoites. There were 454 annotations for gametocytes. Table 3.2 shows the distribution of the annotations for the various classes.

Table 3.2: Number of annotations for growth-cycle stage classification

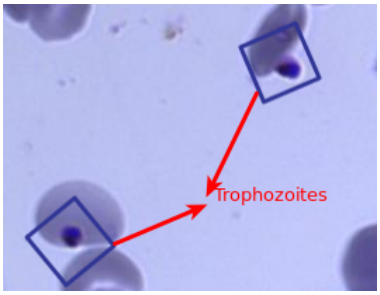
Annotations	No._ Annotations	Percentage of Total Annotations (%)
Trophozoites	614	57.491
Gametocytes	454	42.509



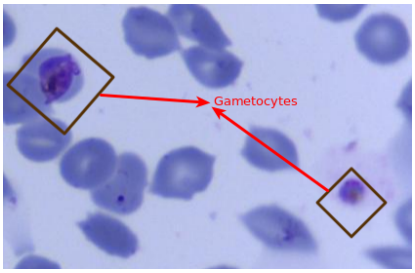
(a) A crop with only one trophozoite



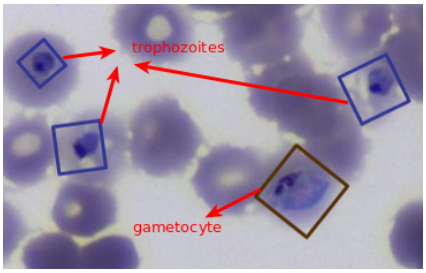
(b) A crop with only one gametocyte



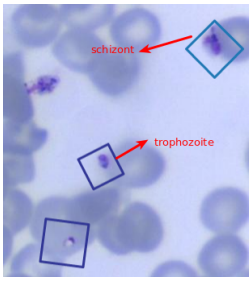
(c) Multiple trophozoites in one image



(d) Multiple gametocytes in one image.



(e) Multiple gametocytes and trophozoites in one image.



(f) Trophozoites and schizonts

Note: All annotations are treated independently.

Figure 3.2: Crops of parasites growth-cycle stage annotations

3.2.2 Grayscaling

Two separate experiments were conducted for the binary classification task, one using the red green blue (RGB) images and the other using grayscale images. This was necessary because the background color of the negative samples was utterly different from the background color of the positive samples. It therefore became trivial for the algorithm to discriminate between the two classes using the color and other global statistics. This meant that the algorithm was not learning any representation peculiar to *Plasmodium* parasites. Gray-scaling was therefore introduced to reduce the influence of the color difference on the classification. However, as discussed in Chapter 5, both methods attained similar performance. This section describes the method which was used in the conversion from RGB to grayscale.

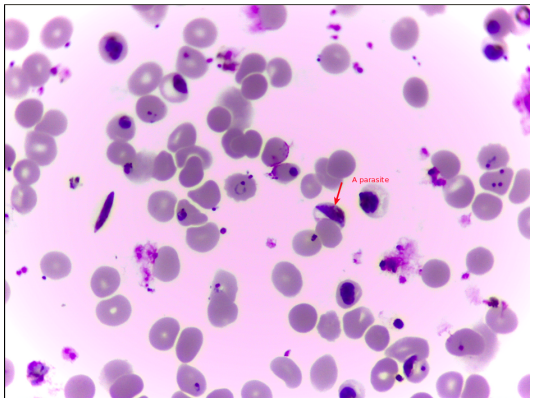
Choosing an RGB-to-grayscale conversion method is not trivial because the conversion method can impact the performance of the classification algorithm (Kanan & Cottrell, 2012). Some researchers adopt the Hue Saturation Value (HSV) color-to-grayscale conversion method (Bibin et al., 2017; Penas, Rivera, & Naval.Jr, 2017) while others employ a variant of the HSV method called the HSI method (Pinkaew et al., 2015). A third category of researchers combine the two methods above as different features in the conversion process (Purnama et al., 2013).

In the HSV method, the maximum value of the RGB channels is used as the grayscale value. That is, $Y_{HSV} = max(R, G, B)$. This preserves the maximum brightness of the image in question. Unfortunately, preserving the maximum brightness makes HSV vulnerable to changes in the image brightness (Kanan & Cottrell, 2012). This renders HSV an unsuitable RBG-to-grayscale conversion method for problems where the images vary. The brightness problem induced by HSV is resolved by averaging the gamma corrected R, G, B values. That is, $Y_{Gleam} = \frac{1}{3}(R', G', B')$ where *R', G', B'* are the gamma corrected values. This method is the best RGB-to-grayscale conversion method for most classification tasks (Kanan & Cottrell, 2012). They used gleam for RGB-to-grayscale conversion because of the advantages it has over HSV and the other conversion methods. An implementation of

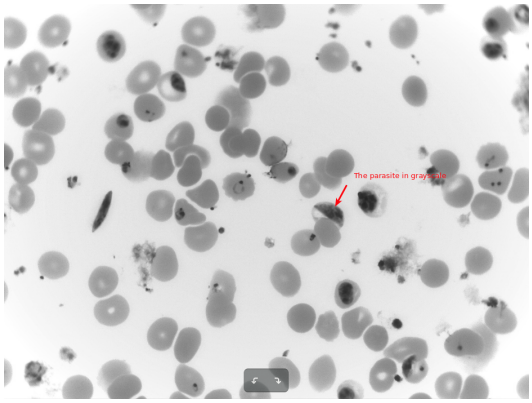
Gleam was retrieved from OpenCV which used Eqn 1 in the implementation.

$$Y \leftarrow 0.229.R + 0.587.G + 0.114.B \tag{1}$$

Figure 3.3 shows an example of an image and its grayscale version



(a) RGB Image. As shown in the picture, the background color of the RGB is so conspicuous that it can affect the classifier's performance.



(b) Grayscale version of (a). Although the grayscale form has less background color, it preserves the structure of the parasites which is the important feature for classification

Figure 3.3: RGB and grayscale images

Unlike most studies in malaria diagnosis, no hand-engineered features were supplied to the convolutional neural networks. However, a lot of techniques including data augmentation, transfer learning and context encoding have been employed to prevent the model from over-fitting the data. These techniques are discussed in succeeding sections.

3.2.3 Data Augmentation

CNNs often require big datasets in oder to obtain good performance on new unseen examples. Data augmentation is often used to increase the training datasets in cases where the number of training examples is small (Penas et al., 2017). In data augmentation, new samples are generated artificially using mostly transformation functions. Besides increasing the size of the training set, data augmentation is used to balance the number of examples for each class in the training set (Quinn et al., 2016). This is achieved by exerting a weighted

augmentation to each of the classes. Classes with large representations in the dataset are augmented less compared to classes with small number of examples in the training set. Balancing the training set usually improves the performance of the model.

Three artificial examples were generated for each of 1068 annotations as reported in the subsection 3.2.1 above. Augmentation was achieved through the three methods below.

1. Horizontal flip: the new image is an instance of the original image reflected along its horizontal axis.

2. Vertical flip: the new image is an instance of the original image reflected along its vertical axis.

3. Rotation through $90°$: the new image is an instance of the original image rotated $90°$ clockwisely.

3.3 Parasites Classification Algorithm: Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are deep, feed forward Artificial Neural Networks (ANNs) specialized for processing image data (Krizhevsky et al., 2012). ANNs typically consist of several layers stacked together to collectively decode the properties of the input data conditioned on some ground-truth labels associated with the input. Thus, ANNs are largely used for supervised classification problems although they can be adapted for unsupervised tasks as in (Pathak, Krahenbuhl, Donahue, Darrell, & Efros, 2016). This covers only a general overview of CNNs, detail and comprehensive review of deep learning and ANNs can be found at (Rawat & Wang, 2017).

In its most basic form, a CNN can consist of only three layers; an input layer, a hidden layer (convolutional layer) and an output layer. However, CNNs are not typically used in this form because the more hidden layers a network has, the greater its capacity to learn interesting patterns in very complex data. Since CNNs are tailored for analyzing spatial image data, they usually have many more layers than the vanilla ANNs. Networks with many layers are said to be deep networks. CNNs and other deep networks have become ubiquitous following the remarkable performance of the AlexNet in ImageNet challenge in 2012

(Krizhevsky et al., 2012). The excitement around CNNs is mostly due to the recent triumphs they have made in very difficult tasks such as autonomous driving. On the downside, CNNs require large amounts of data and are computationally expensive to train.

3.3.1 Layering in Convolutional Neural Networks

As stated earlier, the structure of CNNs (true for all ANNs) are organized into layers. Each layer contains many neurons in analogy with biological neurons. In reality however, a neuron is simply a mathematical function of the form

$$\begin{aligned} y &= f\left(\sum_{i=1}^{n} \theta_i x_i + b\right) \\ &= f\left(\theta^T x + b\right) \end{aligned} \tag{2}$$

where

$y =$ output, $\quad x \in R^n =$ input vector, $\quad \theta \in R^n =$ weights or parameters and $b =$ bias

f is the so called activation function. A specific instance of f is discussed below.

Generally, a layer in the network is simply a bunch of the function above. Specifically, Eqn 2 describes a neuron in a fully connected layer. The fully connected layers are responsible for the classification task. Before the connected layers, CNNs normally implement other hidden layers. These hidden layers are usually organized into blocks called convolutional blocks. Each convolutional block contains convolutional layers and often pooling layers. A convolutional block may also contain other layers such as dropout layers and normalization layers. The two primary layers in a convolutional block are described next.

Convolutional layers are typically used as feature extractors. A convolutional layer exploits the spatial properties of images by sliding a filter known as a kernel over the entire image. The sliding of the kernel along the image is known as convolving. A complete convolution over an image produces a feature (activation) map whose values are of the form

shown in Eqn 3. The computation of the output outputs is the same as the computation for the fully connected layer performed over a smalled region of the image whilst preserving the spatial structure of the image.

$$y_{i,j} = f\left(\sum_{q=0}^{m-1}\sum_{r=0}^{m-1}\theta_{q,r}^{T}x_{i+q,j+r}\right) \tag{3}$$

where

$y_{i,j}$ is the value at position i, j in the activation map

θ *is the kernel of size* $m \times m$*, and* x is an input neuron

f is an activation function

Convolving a kernel over an image produces an activation map. Although there are no mathematical theory behind choosing the size of a kernel, square kernels of smaller sizes($m < 9$) are common (Sathpathi et al., 2014; Krizhevsky et al., 2012). The number of pixels to skip after each convolution is called a stride. Again, no mathematical theory exist for choosing a stride but 2×2 strides are often used. Following the logic of convolutions, it can be deduced that given an image of size $N \times N \times d$, the size of the activation map is $N^* \times N^* \times k$, $N^* < N$. However, the activation maps are often padded so that, $N^* = N, N^*$ is given as

$$N^* = ((N - m)/s) + 1 \tag{4}$$

where

$d =$ The depth of the input

$k =$ the number of kernels, which becomes output depth

$m =$ the kernel size

$s =$ the stride

Usually after the convolutional layers are pooling layers. The purpose of pooling layers is simply to reduce the dimensionality of the activation maps. This is important

because it prevents the network from over-fitting the data which helps in generalization (Rawat & Wang, 2017). The structure of pooling layers are very similar to the convolutional layers except that pooling layers do not learn new features of the data. Thus, pooling layers have kernels and strides but no activation functions.

There are several methods for pooling. However, max pooling which is the method used in this thesis is the most common (Rawat & Wang, 2017). Max pooling preserves the maximum activation in a given local region as the activation for that region as expressed in Eqn 5.

$$
\begin{aligned}
y_{i,j} &= \max\left(x_{i:i+m,j:j+m}\right) \\
&\text{where} \\
y_{i,j} &= \text{output matrix} \\
m &= \text{kernel size} \\
x &= \text{input matrix}
\end{aligned}
\tag{5}
$$

Like convolutional layers, if given an input volume of $N \times N \times D$, a pooling layer produces an output volume of $N^* \times N^* \times D$ where N^* is given by Eqn 4 above.

3.3.2 Activation functions

At the heart of CNNs are activation functions. Activation functions transform the linear computations into non-linear outputs. This allows the network to learn interesting properties of the data. The activation function used in this thesis is the Rectified Linear Unit (ReLU) primarily because of its speed. The ReLU is a simple function which fires at only positive inputs. The ReLU can be expressed mathematically as

$$f(x) = max(0, x) \tag{6}$$

The simplicity of ReLU makes its implementation easy and it often results in reduced training time. However, it is undesirable to use ReLU as the activation function for the

output layer. The reason is that the output layer should produce the probabilities for the classes under consideration given the input. However, it is evident from Eqn 6 that the ReLU can produce values outside the valid probability range [0 - 1]. Hence, a squashing function called the Softmax is used in the output layer. Softmax squashes an input into the range [0, 1]. The Softmax is a generalization of the Sigmoid function which is given by

$$\begin{aligned} g(z) &= \frac{1}{1+\exp\{-z\}} \\ \implies g(\theta^T x) &= \frac{1}{1+\exp\{-\theta^T x\}} \end{aligned} \tag{7}$$

$g(\theta^T x) > 0.5$ whenever $\theta^T x > 0$. This property of the Sigmoid function is effective for binary classification problems. For multi-class classification problems, the normalized values of the Sigmoid over all the classes is used. This function known as the Softmax is expressed in Eqn 8.

$$S(\theta^T x)_i = \frac{\exp\{\theta_i^T x\}}{\sum_{k=1}^{K} \exp\{\theta_k^T x\}} \qquad \forall i = 1, ..., K, \quad K > 2 \tag{8}$$

Where K is the number of classes. The Softmax produces probabilities of the various classes given the input data. The class with the largest probability is taken as the predicted growth-cycle stage.

3.3.3 Loss Function

A loss function is a mathematical construct which measures the divergence (or compatibility) between the label predicted by the model, $\hat{y}$ and the ground-truth label, y. In its simplest form, a loss function measures the average error between the model's predictions and the true labels. The model uses the loss to judge its performance on the given task and update its parameters accordingly. This process, known as optimization is discussed in the next subsection.

There are several loss functions but cross-entropy is the best loss function for classification problems (Li, Johnson, & Yeung, 2017). Cross-entropy is used for classification problems because it measures performance where the outputs are probability values. The higher the error between the $\hat{y}$ and y, the higher the cross-entropy loss and vice-versa. This

project used categorical cross-entropy for the growth-cycle stages classification. Binary cross-entropy was used as the loss function for predicting whether a given image contains a *Plasmodium* parasite or not. These variants will be explained shortly. First, is a description of the representation of the ground-truth labels and the form of the predicted labels.

For the growth-cycle stages classification, the following assignments were made

$$negative = 0, \quad trophozoites = 1, \quad and \quad gametocytes = 2$$

The labels for the binary classification task were $negative = 0 \quad and \quad positive = 1$

Instead of feeding the model with the labels above, the model was fed with one-hot encoded labels. The problem associated with using the labels above is that they introduce an idea of ordering into the labels. That is, they make the model believe that if the true label is *gametocytes* and it predicts *trophozoites*, it is doing a better job than if it predicts *negative*. However, this is not true, the two cases are both wrong predictions and should be penalized equally. One-hot encoding removes the concept of ordering by representing each label as vector which has 0 everywhere except at the index of the true label which has 1. Given the labeling above, the one-hot encoded label for each of the classes is given as:

$$\begin{aligned} negative &= [1, 0, 0] \\ trophozoites &= [0, 1, 0] \\ gametocytes &= [0, 0, 1] \end{aligned}$$

That is, y is a matrix of the form

$$y \in \{0, 1\}^{m \times k} \tag{9}$$

where m is the number of examples and k is the number of classes.

The Softmax function introduced in the previous section produces a similar matrix as output

except that the entries are probabilities. That is, $\hat{y}$ has the form

$$\hat{y} \in \mathbb{R}^{m \times k}, \quad y_{i,j} \in [0,1] \quad \forall i = 0, ..., m \quad \forall_j = 0, ..., k \tag{10}$$

Having established the nature of the ground-truth labels and the predicted labels, the formulation of the cross-entropy loss is considered next. Cross-entropy minimizes the negative log likelihood of the parameters. Since cross-entropy is the function of interest, the derivation of the log likelihood function is not covered here. It is worthy of note that cross-entropy works under the assumption that samples are independent and identically distributed. This assumption is not always true but it enables the application of cross-entropy to probabilistic models.

The log likelihood $\mathcal{L}$ of all K classes for all m examples is given by

$$\mathcal{L} = \sum_{i=1}^{m} \sum_{j=1}^{K} y_j^{(i)} \log \hat{y}_j^{(i)} \tag{11}$$

The bigger the value of $\mathcal{L}$, the little error the model is making. That is, maximizing $\mathcal{L}$ results in the best performance of the models. Instead of maximizing the log likelihood, cross-entropy performs an equivalent function which is minimizing the negative log likelihood. Therefore, the objective becomes the unconstrained optimization problem

$$\begin{aligned} & min\ H \\ \text{where} \qquad & H = -\sum_{i=1}^{m} \sum_{j=1}^{K} y_j^{(i)} \log \hat{y}_j^{(i)} \end{aligned} \tag{12}$$

When the number of classes is greater than two ($K > 2$), Eqn 12 is called categorical cross-entropy. The binary form is a simplification of Eqn 12 to the case where $K = 2$. The binary cross-entropy has the simplified objective

$$min\ H_b$$
$$H_b = -\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log\big(1 - \hat{y}^{(i)}\big) \tag{13}$$

3.3.4 Optimization Method

The last aspect of the algorithm is the optimization method. An optimization method is an algorithms which searches for the parameters that minimize the loss function. Adam is the optimization method which was used in this research. Adam was chosen primarily because of its computational efficiency and little memory requirements (Kingma & Ba, 2014). Furthermore, Adam does not usually require a lot of hyper-parameter tuning. Adam optimizes stochastic objective functions by "computing adaptive learning rates for different parameters from estimates of first and second moments of the gradients" (Kingma & Ba, 2014). The gradients are computed once for each update step using back-propagation. Adam uses the update rule

$$\theta_t \leftarrow \theta_{t-1} - \alpha.\hat{m}_t/(\hat{v}_t^{0.5} + \epsilon)$$

where

$$\begin{aligned}
\theta_t &= \text{All parameters at time t} \\
\alpha &= \text{The learning rate, a hyper-parameter set by the user} \\
\hat{m_t} &= \text{The corrected first moments at time t} \\
\hat{v_t} &= \text{The corrected second moments at time t} \\
\epsilon &= \text{A small constant to prevent numerical issues}
\end{aligned} \tag{14}$$

The hyper-parameter α is fixed throughout the training. However, the first moments estimates and the second moments estimates take different values at different times of the optimization which makes the overall update step adaptive. The values are computed from the gradients and controlled by two other hyper-parameters $\beta 1$ and $\beta 2$. These hyper-parameters and the optimization method in general are discussed extensively in (Kingma & Ba, 2014).

This paper used the values of $\beta 1$ and $\beta 2$ in the original paper but experimented with different learning rates.

3.4 Dealing with Little Labeled Data

The main challenge with using CNNs is that they require large datasets. This project did not have such a large dataset required by the model. The issues posed by the small dataset were addressed through transfer learning and unsupervised pre-training. These methods were use to generate good initialization weights for the classification tasks. This section discusses these two techniques.

3.4.1 Transfer Learning

In many applications related to medical imaging, it is often impractical to obtain large labeled datasets sufficient to train a CNN from scratch (Afridi, Ross, & Shapiro, 2018). Transfer learning is often used to address the problem of limited labeled data. Transfer learning is passing on the knowledge contained in a model pre-trained on a larger dataset to the problem with limited data. Transfer learning has been used in many computer vision malaria research studies resulting in improved performance (Sivaramakrishnan, Antani, & Jaeger, 2017; Penas et al., 2017). In the context of CNNs, transfer learning is often used for fine-tuning. This is the transplantation of the learned feature layers of a CNN trained on a large dataset to initialize the CNN for the smaller dataset.

In order to obtain optimum performance using transfer learning, the source dataset and the target dataset must have similar characteristics (Sivaramakrishnan et al., 2017; Yosinski, Clune, Bengio, & Lipson, 2014). However, other studies suggest that transfer learning can still yield improvement in performance even when the source task is superficially different from the target tasks (Afridi et al., 2018). This means that if there are no pre-trained models in the target problem's domain, a more general pre-trained model can still be fine-tuned with the unrelated data.

Although more images were created through data augmentation, the total number of samples was not still sufficient to train a CNN from scratch to obtain good performance.

Transfer learning was used to supplement the data augmentation to prevent over-fitting. Ideally, the source CNN should have been pre-trained on Giemsa-stained images if not on images of *Plasmodium* parasites. Unfortunately, such pre-trained models are currently not available. The reason is that generally, Giemsa-stained images datasets are always small to begin with. Since no ideal pre-trained model could be found, the VGG16 network (Simonyan & Zisserman, 2014) pre-trained on ImageNet (Russakovsky et al., 2015) was used as the source model.

Only the convolutional blocks of the VGG16 network were downloaded. However, only the last three layers were trainable. The activations from the last convolutional layer was fed to a custom-made fully connected layer before the output layer. The diagram below shows the architecture of the pre-trained model. The VGG16 network was chosen for its simplicity and speed. The focus of the experiment, however, was not on the details of the specific network used. The architecture of the fine-tuned model is shown in Figure 3.4.

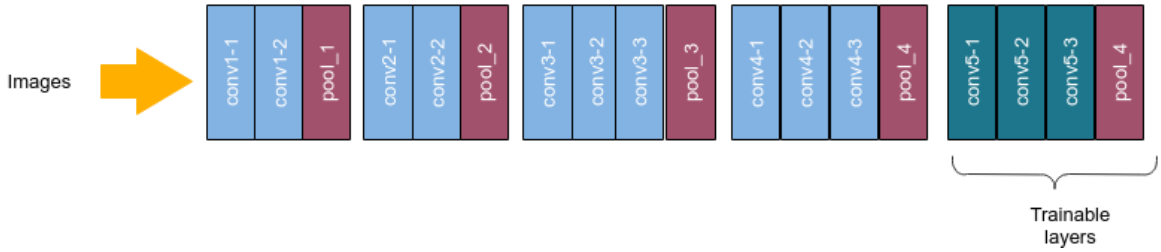


Figure 3.4: Fine-tuned VGG16 architecture

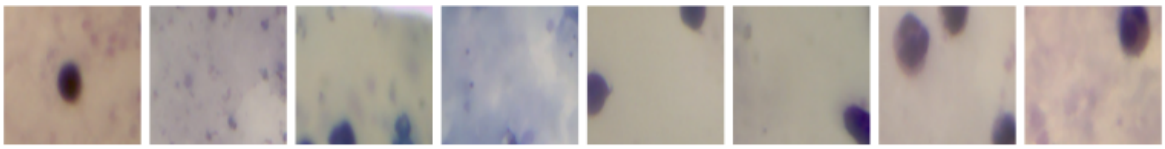
3.4.2 Context Encoding

Besides data-augmentation and transfer learning, context encoding has typically been used to address the challenges introduced by the lack of large labeled data. Context encoding is the process of training a CNN to generate the contents of an image region based on its surroundings (Pathak et al., 2016). The context encoder can then be used as an initialization to other tasks where there is no enough data.

A context encoder typically has two components: an encoder network and a decoder model. The encoder learns the properties of the inpainted image based on the context. The decoder reconstructs the original image from the output of the encoder. This project

used the context encoder used in (Pathak et al., 2016). Pathank *et al.* used AlexNet up to layer 5 for the encoder. The decoder was implemented through rescaling and up-sampling. Unlike Pathank *et al.*, this project used only the L2 reconstruction loss in training the context encoder.The implementation of the adversarial discriminator as in Pathank *et al.*was not considered because of the large training time required for the adversarial loss.

Two-thousand seven hundred and three (2,703) Giemsa-stained thick blood smears were used to train the context encoder. Arbitrarily selected region of each image was inpainted (destroyed). That region of the image in addition to 35 pixels wide context around it was feed to the encoder. The decoder was task to reproduce the contents of the images which were destroyed. Examples of the inpainted images and the corresponding ground-truth images are shown in Figure 3.5.



(a) 227×227 Random crops of the images. The context encoder is tasked to reproduce these crops give the inpaints in (b)

(b) 153×153 Inpaints of the crops in (a) above with 35×35 context at all sides. These impaints are the inputs to the context encoder. The crops in (a) are the targets

Figure 3.5: Random crops and their inpaintings for unsupervised pre-training

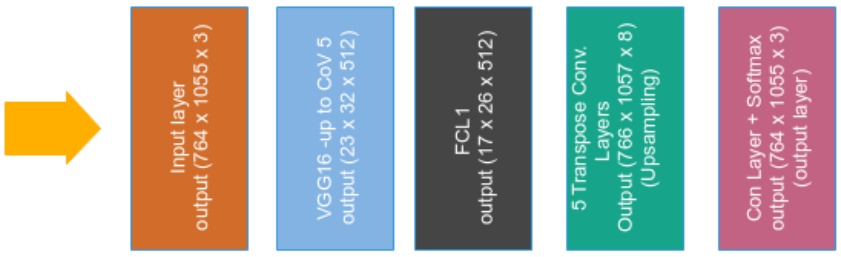
3.5 Semantic Segmentation

This subsection discusses the processes which were used to conduct the growth-cycle stages classification. Each Giemsa-stained thin blood smear typically contains *Plasmodium* parasites at different growth-cycles and normal RBCs. Therefore, different regions of a given image should contain different labels in order to detect different growth-cycle stages. Regions of the images containing parasites were identified through annotations as discussed in subsection 3.2.1.

Given the annotations, one way to perform the growth-cycle stages classification is to treat each annotation as an independent image (Quinn et al., 2016). In that approach, negative samples are created by taking random crops of the images not covered by any annotation. This approach, known as the sliding-window approach poses several challenges. The first challenge is that the annotations must have the same spatial dimensions. Secondly, passing windows of the image through the network individually is computationally expensive. Finally, the sliding-window approach is slow at test time since all patches(windows) of the test image must be evaluated. These deficiencies render the sliding-window technique ineffective.

Instead of creating labeled patches of the images, each pixel in the image can be assigned a label. This approach is called semantic segmentation. Semantic segmentation is a pixel-wise classification. Each pixel in the input image is classified into one of K possible classes. Although semantic segmentation requires many predictions, it is computationally efficient because all pixels in an image are passed through the network together. Different CNN based semantic segmentation techniques have been proposed in the literature (Gupta, Girshick, & Arbel, 2014; Cires & Giusti, 2012) but that of only Long, Shelhamer and Darrell (2015) is discussed here.

Long *et al.* were the first to propose a fully convolutional neural networks(FCNs) for semantic segmentation. Essentially, they replaced the fully connected layers typically at the end of CNNs with convolutional layers. The low resolution outputs of the convolutional layers are upsampled to the input resolutions for pixel wise predictions. FCNs have since emerged as the method of choice for semantic segmentation tasks due to their ability to achieve good performances on image segmentation. It is those same successes which motivated the use of FCNs in this research. The implementation involved adding convolutional layers to a pre-trained VGG16 model as discussed in subsection 3.4.1 to provide initialize parameters. Upsampling was achieved through transpose convolutions (Wojna et al., 2017). Advanced methods such as skip connections were not included in the implementation. The architecture of the semantic segmentation network used for growth-cycle stage classification is shown in Figure 3.6.



The inputs of the Input Layer are the raw images and their labels. FCL=Fully Convolutional Layer, Con Layer = Convolutional Layer. Each succeeding layer takes input from the previous layer.

Figure 3.6: Growth-cycle stages classification model architecture

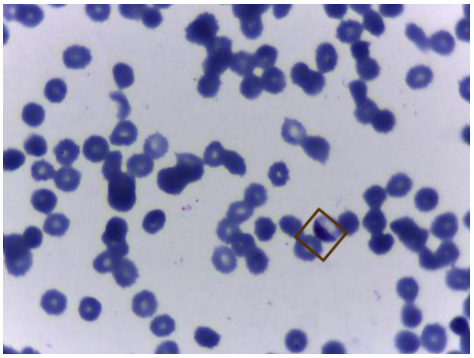
3.5.1 Masks Generation

In order to perform semantic segmentation, each pixel in the images had to be labeled into one of three classes: negative(normal cell), trophozoites and gametocytes. These labels for each image is known as the mask of the image. Each mask contained the same number of pixels as the original images. However, unlike the original images which had three values for each pixel, the masks had only one value for each pixel. Furthermore, each mask had only three distinct values (0, 1, 2) indicating the labels of the pixels. That is, a 1 at index (x, y) in a mask indicates that the pixel at index (x, y) in the corresponding ground-truth image belongs to a trophozoite. Normal cells were labeled with 0 and 2 represented gametocytes.

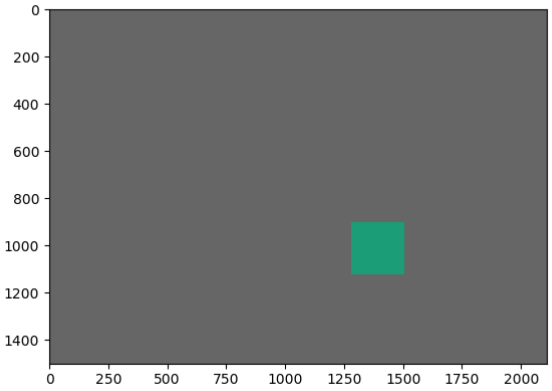
The masks were generated using the ground truth annotations of the images some of which are shown in Figure 3.2 . Each annotation in an image was associated with four distinct coordinates representing the different vertice of that annotation and a label indicating the growth-cycle stage of the parasite. The structure of an annotation was of the form: *filename-annotation index, label, comment, geometric shape, x1, y1, x2, y2, x3, y3, x4, y4, x1, y1*. Thus, each pixel can be labeled using the label of the annotation and the pixel positions given by the coordinates.

In the ground-truth annotations, some of the annotations were not axis-aligned as the example in Figure 3.2. This posed challenges when assigning the labels because it required slicing matrices diagonally which is difficult. As a result, all annotations were rotated to

align with the axes of the images. In order to rotate an annotation, a tight bounding box was drawn around the annotation such that the bounding box covered the area previously occupied by the annotation. That is, the area of an annotation is a lower bound on the area of the bounding box for that annotation. This ensured that no regions of the actual parasites were lost to rotation. In consequence, more normal cells were labeled as either belonging to trophozoites or gametocytes. The detail algorithm of how the bounding boxes were generated is not covered here. The results of the rotations, however, is shown in Figure 3.7. The coordinates of the bounding boxes together with the corresponding labels were then used to label all the pixels.



(a) Ground-truth annotation - as show in the image, the square bounding boxes is tilted at approximately 45° relative to the axis of the image.



(b) Mask of ground-truth image with rotated annotation - Unlike the ground-truth annotation, the bounding box in the mask is axis aligned with the image.

Figure 3.7: A non-aligned annotation and its rotated equivalent

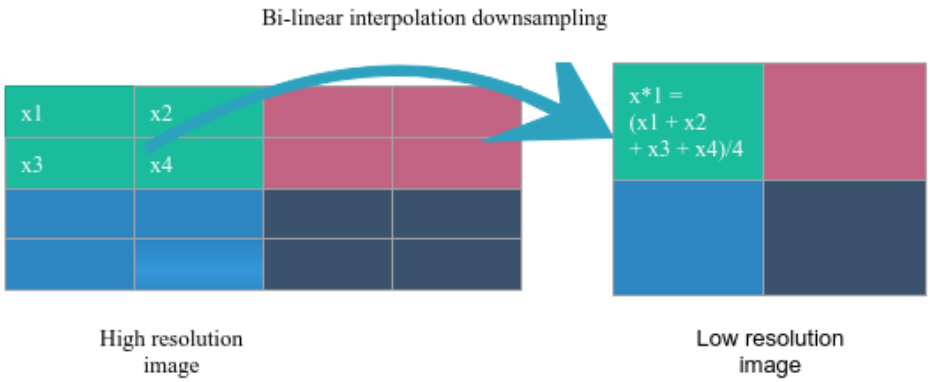
3.5.2 Downsampling

In CNNs, large mini-batch sizes enable the computation of good gradients which improves learning (Radiuk, 2018). The size of mini-batches are limited by the resolution of the images and the amount of available memory capacity. The higher the resolution of the images, the more memory they require to train the models. Semantic segmentation models require a lot of memory to train because the labels(masks) are also large. (The masks have the same dimensionality as the ground-truth images). This makes training high reso-

lution images very expensive. As a result, high resolution images are usually downsampled to lower resolutions to reduce the amount of memory consumption. The amount of downsampling usually depend on the problem domain and the original resolution of the images. That said, the downsampled images should remain meaningful to the problem. Thus, the downsampled images should maintain the characteristics of the original images which are relevant to the classification task.

The images for the growth-cycle stage classification had original resolutions of $1500 \times 2100 \times 3$. At this resolution, the model could be trained on a mini-batch of only two images on a 15GB Graphical Processing Unit(GPU). In order to train at much larger mini-batches, the images were downsampled to $764 \times 1055 \times 3$ resolution. Consequently, the masks generated at the higher resolutions (1500×2110) were also downsampled to 764×1055 pixels. Both the images and the masks were downsampled using inverse mapping. With regards to the interpolation methods, bi-linear interpolation was used for the images. The masks were downsampled using nearest-neighbor interpolation.

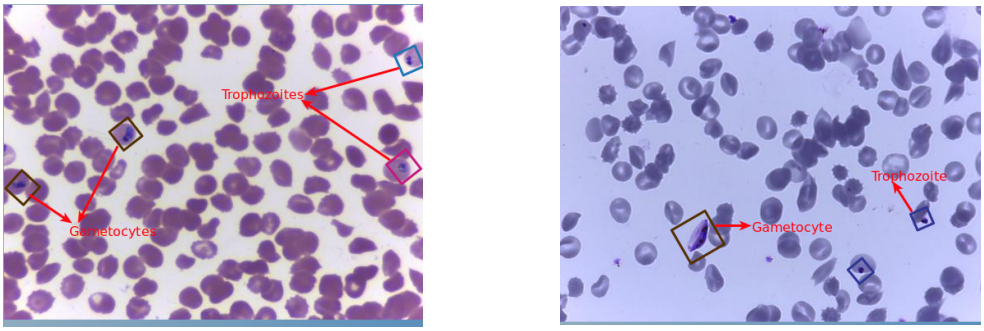
In bi-linear interpolation, the value of a pixel in the downsampled image is a weighted average of the values of all pixels in the original image which map to that pixel. This makes the low resolution images look visually similar to the high resolution images. In nearest-neighbor interpolation, a pixel in the low resolution image is assigned the intensity of the pixel closest to it in the high resolution image determined by some algorithm. Nearest-neighbor is particularly good for downsampling the masks because it preserves the integer labels. Illustrations of these downsampling methods are shown in Figure 3.8. The masks at the original masks and downsampled masks are also shown in Figure 3.9.



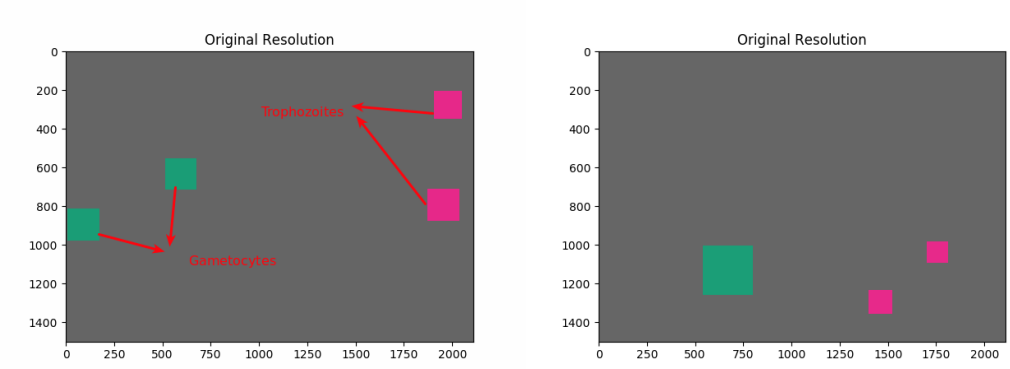
(a) Bi-linear interpolation. The output intensity is an average of the surrounding intensities.

(b) Nearest-neighbor interpolation. The output intensity is the intensity of the closest pixel determine by some algorithm

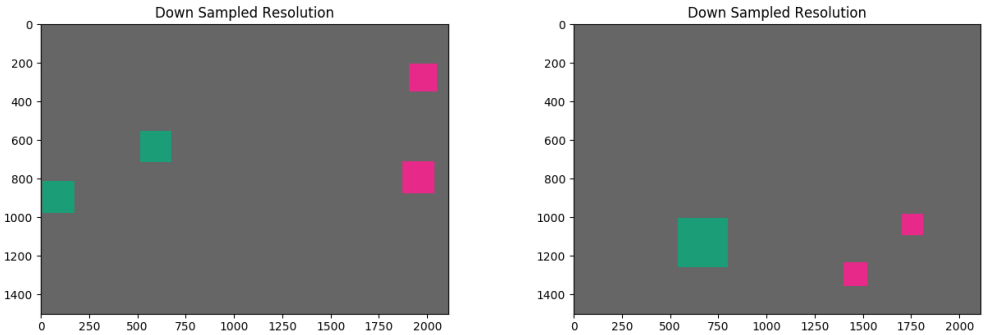
Figure 3.8: Illustrations of downsampling methods



(a) Ground-truth annotations



(b) Masks at original resolution - Unlike the ground-truth annotations, the masks are up-right bounding boxes. Nonetheless, the relative sizes and positions of annotations are preserved. In the masks, all regions of the image not annotated are labeled as negative (gray)



(c) Masks at downsampled resolution - Although the downsampled images are smaller than the original images in terms of size, the relative ordering of the masks are maintained.

Figure 3.9: Ground-truth annotations, original resolution masks and downsampled masks

3.5.3 Weighting of the Loss Function

As noted in the Methodology, all the models use categorical cross entropy as the loss function. In its pure form, categorical cross entropy treats predictions on all the classes equally. As a result, it penalizes all wrong predictions with the same magnitude. This feature of the loss function is undesirable when the training set is unbalanced. This is because when using an unbalanced training dataset with a level-plain loss function, the model will always try to predict the class with the largest proportion. The reason is that by always predicting that class, the model is assured of making correct predictions most of the time. For instance, if the training set comprises of 1000 examples belong to class A and 10 examples belonging to class B, the model is guaranteed $\approx 99\%$ accuracy by always predicting class A.

The training dataset for the semantic segmentation model is extremely unbalance as shown in Table 4.2. This means that using categorical cross-entropy as discussed in the Methodology will induce the model to always classify every pixel as belonging to the negative class. This is undesirable because the model will never learn the characteristics of trophozoites and gametocytes which distinguish them from each other and from normal cells. However, if the model model is made to pay an extra penalty for making wrong predictions on the least represented classes, it will be encouraged to learn the properties of the different classes. Conversely, correct predictions on the least represented classes lead to a low total loss value. The weighted loss function expressed in Eqn 15 was used to resolve the problem of trivial predictions due to the unbalanced training set.

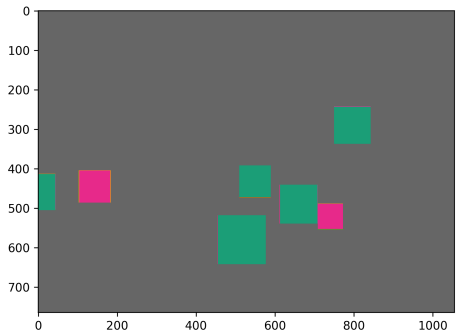
$$H = \sum_{c} \frac{N}{Y_c + \epsilon} H_{y_c}$$

where

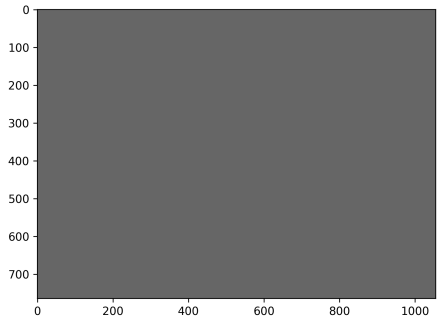
$$\begin{aligned} H &= \text{overall weighted loss} \\ N &= \text{A normalization constant. } N \text{ is set to to the total number of pixels} \\ \epsilon &= 10^{-8}, \text{ A small positive constant to prevent division by zero} \\ Y_c &= \text{The number of pixels in ground-truth labels belonging to class c} \\ H_{y_c} &= \text{The categorical cross entropy loss of the predictions on class c} \end{aligned} \tag{15}$$

Using the above formula, predictions on the negative class are treated with less importance because the denominator is large. Conversely, the predictions on gametocytes is treated with high importance because the loss contributed by wrong predictions on gametocytes is penalized heavily.

To verify that a weighted loss function was actually required, the model was trained using an unweighted categorical cross-entropy. As expected, the model predicted every pixel as negative as shown in Figure 3.10.



(a) Ground-truth label of an image the model had not seen before. Tasked with producing this image, it produced the image in (b) because the number of pixels labeled as negative out-number the other categories



(b) Final activation map - without penalizing errors on wrong predictions of the parasites, the predicts all cells as negative because it will generally be correct. In this test example, the model predicted all pixels as negative. Those predictions resulted in 98.74% accuracy

Figure 3.10: Activation map from unweighted loss

Chapter 4: Methodology 2 - Implementation

4.1 System Architecture

The application had two major components: a training module and testing module. As should be expected, the application behaves differently for the two components. A flow chart diagram of the application is shown in Figure 4.1 below. The details of the training and testing modules are discussed in section 4.3 and section 5.2 respectively.

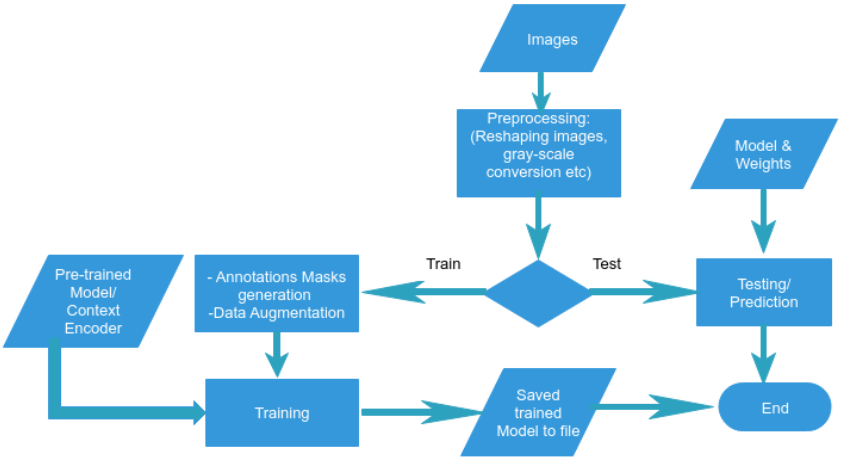


Figure 4.1: System flowchart

4.2 Implementation Resources

The model was developed using three main packages in python: OpenCV, Keras and TensorFlow. These packages provide highly optimized implementations of the machine learning algorithms which used in this project. Adopting these packages reduced the amount of boiler-plate code required for testing the algorithms. Also, since many researchers use those libraries, adopting them makes it easier for other people to verify the results.

4.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) is a BSD-licensed open-source computer vision and machine learning library. OpenCV was mainly used for image preprocessing tasks such as conversion from RGB images to grayscale, images downsampling and

rescaling. Since OpenCV was built to take advantage of multi-core processor systems, the functions are highly optimized for matrix operations. OpenCV thus offered faster implementations of the image processing tasks used in this implementation.

4.2.2 Keras

"Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation" (Keras, 2018).

The extract above, taken from Keras documentation, summarizes the main reasons why Keras has emerged as the default library for computer vision research. Keras works on both GPU and CPU, has many different optimization and loss functions and provides mechanisms for saving the model at various checkpoints. The context-encoder and the main classification models were developed using the Keras API. Keras also provided the pre-trained model as well as the data augmentation functions used in this research.

TensorFlow was the back-end library for Keras. Although no code was written directly in TensorFlow, all the Keras computations were executed in TensorFlow. Thus, the next subsection discusses the motivation why TensorFlow was chosen as the back-end library.

4.2.3 TensorFlow

TensorFlow is a machine learning software library which was developed by Google Brain Team for research in Google. Since becoming an open-sourced library in 2015 under Apache License 2.0., many researchers and software companies have adopted TensorFlow for their intelligence research. TensorFlow uses data flow graphs for all its computations. Mathematical operations are represented as nodes while data is represented as edges(tensors) which designed to handle multidimensional data. This makes TensorFlow well suited for image processing applications since images are generally processed in their spatial form. Finally, TensorFlow provides seamless transition between CPU and GPU, which Keras exploits to speed up training.

4.3 Training

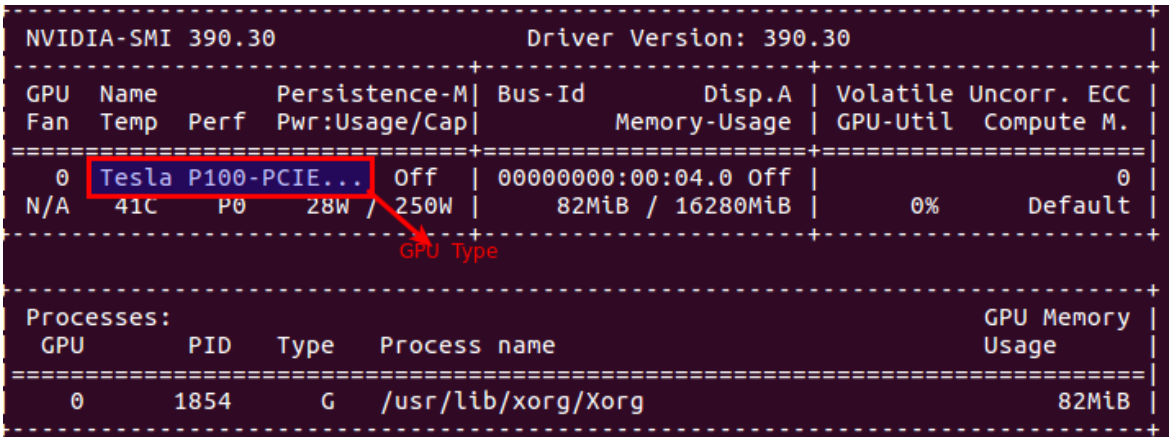
All the models were trained using the Google Compute Engine Application Programming Interface (API). This became necessary because the models could not be trained fast enough on CPUs because they contained many parameters. One of the models experimented for the semantic segmentation task for instance was had 31,095,763 trainable parameters. It takes a long time to train such models on CPUs since CPUs often execute instructions sequentially. An interesting observation, however, is that most of the operations in CNNs are computations on matrices. Since matrix operations are mostly element-wise operations, they can be executed in parallel. Graphical Processing Units(GPUs), designed for processing spatial data have the ability to execute many instructions in parallel. Hence, training on a GPU can reduce training time from days to hours.

For comparison purposes, the binary classification model was experimented on both the GPU and on an i-3 Intel CPU with a 4GB RAM. For twenty epochs, it took the GPU only six(6) minutes to finish training the model while it took the CPU almost three days to train the model. Additionally, the CPU could train on only a mini-batch size of 1 whilst the GPU trained on a mini-batch of 42 images.

Besides the number of parameters, the input features had very high resolution (764 $\times$ 1055 $\times$ 3). The high dimensionality of the data combined with the large number of parameters meant that training the models required large Random Access Memory(RAM). Personal computers, being largely general purpose machines, are not equipped with powerful GPUs and often have relatively smaller RAM compared to that which the models needed. Fortunately, the computational power required for training CNNs is accessible through Google Cloud's API. All the models experimented in this research were trained on a virtual machine[6] provided through the Compute Engine API. The virtual machine was equipped with a NVIDIA Tesla P100 GPU running on Compute Unified Device Architecture(CUDA) Toolkit 9.0 from NVIDIA. Additionally, CuDNN 7.0.5 was installed to enable Keras and TensorFlow execute instructions on the GPU. The machine had a 16 cores CPU with a 30 GB RAM. Finally, the virtual machine ran on the Ubuntu 16.04 LTS operating sys-

[6]Virtual machines on on Google Cloud are referred to as instances.

tem.The CUDA Toolkit required only a registration to use but the virtual machine charged $0.872 per hour of usage. The architecture was setup in us-east1-b[7]. The details of the virtual machine's GPU are shown in Figure 4.2.



The GPU has a total of 16GB internal memory and a compute capability of 6

Figure 4.2: GPU properties

4.3.1 Binary Classification

The binary classification model was trained on 323 images and validated on 168 images. In order to avoid loading load all the images into memory at once, the model was implemented to read the data from disk on demand. Data augmentation techniques were applied to both the training and validation datasets. That is, new images belonging to the two datasets were generated infinitely for training the model and evaluating the model. The binary classification model used a VGG16 pre-trained model for parameter initialization. All the layers of the pre-trained model except the last three were frozen. That is, only the last three layers were fine-tuned. The pre-trained model was extended by stacking two fully connected layers and an output layer on top of it. The model was regularized using one dropout layer immediately before the output layer. The model had a total of 22,087,682 parameters with only 12,092,610 of them being trainable. The input resolution was $500 \times 500 \times 3$ pixels.

[7]Google groups its virtual instances clusters into regions which are subdivided into zones. Some resources are not accessible in some zones. The pricing also vary across different zones. As of the time this research was conducted, there was no region in Africa, much less a zone.

The model was trained for 100 epochs using a learning rate of 10^{-5}. The data was supplied in mini-batches of 32 images for training and 10 images for validation making a combined total of 42 images per mini-batch. The GPU took ≈ 18 seconds for each epoch. Thus, it took 30 minutes to train the model.

4.3.2 Growth-cycle Stages Classification

The growth-cycle stage classification problem was modeled as a semantic segmentation problem as discussed in the Methodology above. The model was initialized using unsupervised pre-training and transfer learning to compensate for the little dataset. This section discusses how the model was trained. First, however, is an overview of the dataset which was used to trained the model.

Three hundred and fifty-three (353) images in total contained at least one parasite. Ninety percent of the 353 images representing 318 images were used to train and validate the model. The other 10% of the dataset was reserved for testing the model at the end of training. The model was evaluated once on the test data. The details of the training and test datasets in terms of annotations and in terms of pixel counts are provided in Table 4.1 and in Table 4.2 respectively. The reported number of pixels are for the low resolution images which were used for training. The number of pixels between the true parasites were approximately equally distributed as the shown through the percentage representations in Table 4.3.

Besides dividing the data into training and test datasets, a subsection of the training dataset was used for validation. Validation is the testing of the model on some data which is not used in computing the gradients. Although, 63 images representing 20% of the training data were used for validation, the composition of the validation set changed across different training jobs. That is, in each training job, the parameters were updated on the gradients of only 255 original images selected randomly from 318 images; with the other 63 images preserved for validation. In addition to these ground-truth images, both the training and validation sets were artificially boosted through data augmentation.

Table 4.1: Composition of annotations in Training and Test datasets

Dataset	Total No._ Images	No._ Trophozoites	No._ Gametocytes	Total Annotations
Training Data	318	571(58.44%)	406(41.56%)	977
Test Data	35	43(47.25%)	48(52.75%)	91

Table 4.2: Number of pixels per class in Training and Test datasets

Dataset	Total No._ of Pixels	Negative Pixels	Trophozoites Pixels	Gametocytes Pixels
Training Data	256, 314, 360	249, 747, 634	2, 874, 959	3, 598, 047
Test Data	28, 210,700	27, 462, 705	245, 912	492, 827

Table 4.3: Number of pixels per class in percentages

Dataset	Total No._ of Pixels	Negative (%)	Trophozoites(%)	Gametocytes(%)
Training Data	256, 314, 360	97.43	1.12	1.45
Test Data	28, 210,700	97.35	0.90	1.75

Training the model

Several models were experimented in this research. However, this section discusses only the model which was eventually used on the test data. That model was trained for 200 epochs using an adaptive learning rate with $\alpha_0 = 10^{-4}$ on a GPU. The GPU could only load up to 19 images due to memory constraints. As a result, 17 images were used for training and two images for validation in each training epoch. The number of training steps and validation steps were computed by diving the size of dataset with the mini-batch sizes. This resulted in 25 training steps and 31 validation steps. It took the GPU ≈ 53 seconds on average to execute each epoch. Thus, it took 2 hours 57 minutes to train the model completely[8].

[8]These values are for the model which was used to evaluate the test data. Other training jobs took about 5hrs to complete. Different runs had slighted different training statistics.

Chapter 5: Experiments and Results

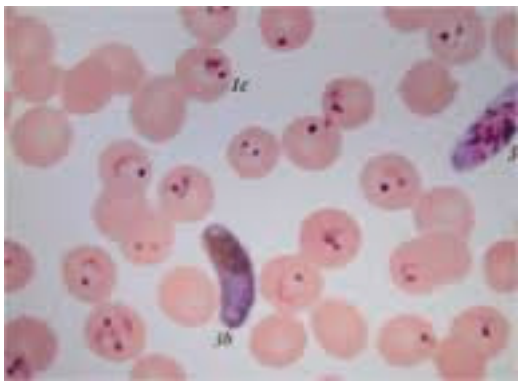
This Chapter discusses the experiments were which conducted to enhance the performance of the models and the results of those experiments. It is worth indicating that this Chapter covers only the experiments which informed the final model and the findings of this research.

5.1 Results of Training

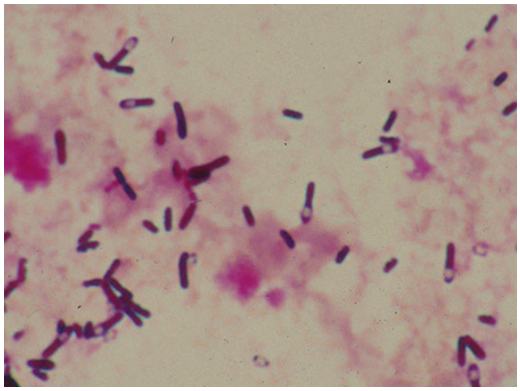
The main concern during training was how the trained model would fair on unseen data. This meant that choices about hyper-parameters values, model size and degree of regularization were based largely on the validation accuracy and validation loss. The training accuracy and training loss were also monitored in addition to the validation metrics.

Binary Classification Results

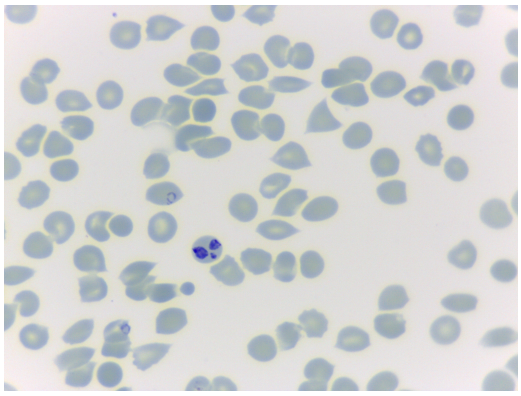
After the 20^{th} training epoch, the model seemed to have reached an optimal point as the training accuracy reached 99.70% and the validation accuracy reached 98.78%. In addition, the validation loss and training losses both decayed towards zero as the plot of the loss in 5.2a show. This meant that the model was not over-fitting the data. This profound ease of discriminating between the two classes after only 20 iterations was baffling. It turned out, however, that in principle the model was not discriminating between the two classes based on the presence or absence of *Plasmodium* parasites. It used trivial properties of the images such as differences in contexts to classify them. From the plot of samples belonging to the two classes shown in Figure 5.1, it is evident that the images could indeed be classified trivially. Methods such as grayscaling were applied to the images to reduce the contextual differences between the two classes. However, all of them proved futile in producing images which could force the model to learn the characteristics of the *Plasmodium* parasites. Advance techniques such as Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) could have produced better performance but there was little time to explore such techniques. Plots of the training and validation metrics of the model are shown in Figure 5.2.



(a) Positive image



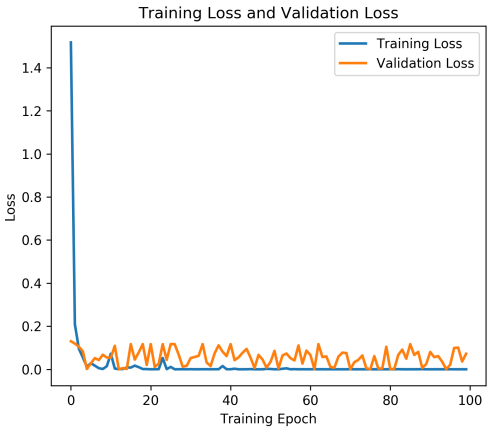
(b) Negative image



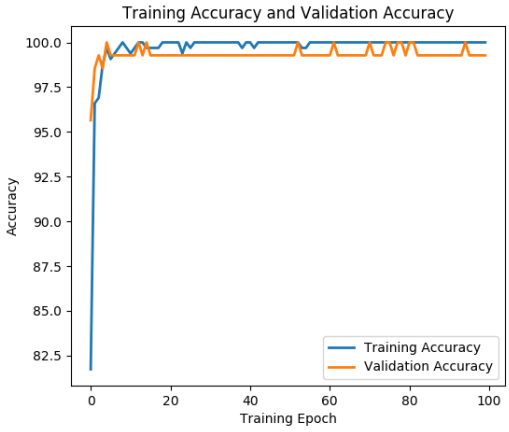
(c) Another positive image - The positive images are a mixture of Giemsa-thin and thick blood smears of *Plasmodium* parasites.However, they are unlike the negative images.

(d) Another negative image - The negative images do not look like anything produced from stained blood smears. Even within the negative images, they are large variations with regards to background and generate semantic properties. This apparent difference between positive and negative images makes the classification extremely easy for the network.

Figure 5.1: Examples of positive and negative images from binary classification dataset



(a) Training and validation losses



(b) Training and validation accuracies

Figure 5.2: Binary classification training metrics

Growth-cycle Stage Classification Results

The growth-cycle stage classification model achieved final training and validation accuracies of 91.25% after several training iterations. The model exhibited several characteristics which made training difficult. The first generation of the model had 4,962,051 trainable parameters. Training at a learning rate of 10^{-5}, the training and validation metrics showed that the model was too simple to properly describe the data as shown in the plots in Figure 5.3. This triggered a revision of the model's architecture to 31,095,763 parameters in an attempt to address the observed under-fitting. The revision process included fine-tuning the last three layers of the pre-trained model and convolving to a resolution of 17×26 pixels before upsampling. The learning rate was also increased to 10^{-4}. Unfortunately, the model became too powerful that it exhibited strong over-fitting as shown in Figure 5.4. It was evident the model was over-fitting the data because both the training and validation unweighted accuracies kept increasing steadily towards 100%. However, unlike the weighted training loss which kept decreasing, the weighted validation loss increased as training progressed. This meant that the model became worse and worse at classifying unseen samples. To verify that the trend was indeed increasing continuously, the number of training epochs was increased to 250. This resulted in the plots shown in Figure 5.5. At about the 250^{th} epoch, both the training and validation unweighted accuracies reached 91% but the validation loss skyrocketed too, a confirmation that the model was over-fitting. Despite the observed over-fitting, this model was kept for evaluation at test time whilst revisions were made for better architectures. This model was labeled as model A.

To prevent the unconventional scenarios where both validation loss and accuracy moved in the same direction, the accuracies were also weighted to be on the same scale as

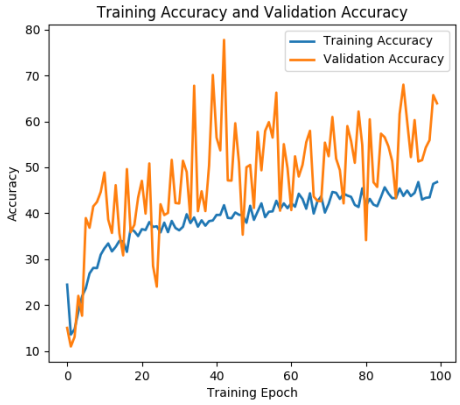
the loss. The accuracies where weighted using the formula

$$
\begin{aligned}
Acc_{weighted} \leftarrow & \frac{1}{N} \sum_{i}^{N} \sum_{c}^{K} 1 \left\{ y^{(i)} = \hat{y}^{(i)} \wedge y^{(i)} = c \right\} \frac{Y_c}{N} \\
& Where \\
& \quad K = \text{The number of classes} \\
& \quad Y_c = \text{The number of pixels belonging to class } c \\
& \quad N = \text{The total number of pixels in the mini-batch}
\end{aligned}
\tag{16}
$$

Using Eqn 16 as the new accuracy metric, adaptive learning rate, advanced regularization and more architecture search, the model was reconstructed to 5,122,307 parameters. The reconstructed model showed evidence of tangible learning after 200 training epochs. This can be seen in the plots shown in Figure 5.6. Additionally, plots of the activation maps of the model on unseen data as displayed in Figure 5.8 indicated strong learning. This became model B. Motivated by the promise of these observations, a similar model, model C, was developed with 6,761,091 learnable parameters through the addition of two fully convolutional layers and changes in the number of filters. Using the same hyper-parameters as the previous model, the new model converged to a local minimum after 130 epochs as shown in Figure 5.7. The test data was then evaluated independently on these three models[9]. It turned out the over-fitting model whose training statistics are shown in Figure 5.5 performed better the other two models on the test images.

[9]This case reminiscent of model ensembles where different models are trained and the best performing one chosen

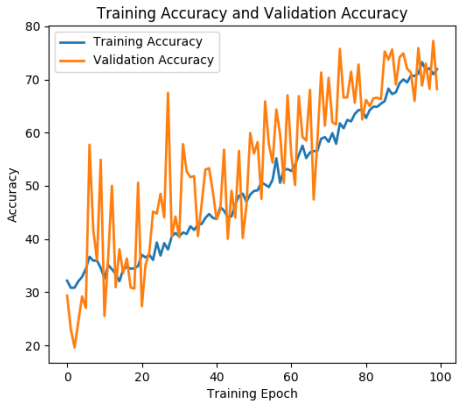
(a) Training and validation loss - Although the training loss and validation are both weighted, the same normalization constant is applied to both. This is the cause of the relatively high validation loss compared to the training loss. The same normalization constant for the two cases scales the validation loss more because the validation data is smaller than the training data



(b) Unweighted training and validation accuracies - After 100 epochs, the training and validation accuracies began to converge. However, unlike the training accuracy, the validation accuracy oscillated widely and was generally worst than the training accuracy. Since both accuracies are quite low, it can be safely assumed that the model is under-fitting the data.

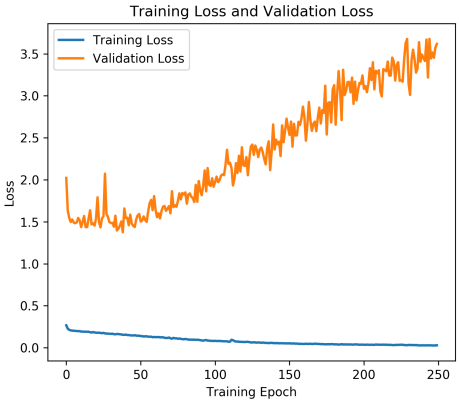
Figure 5.3: Training and validation metrics of under-fitting model

(a) Training and validation losses - Unlike the training loss which decreases, the validation loss increases. This is an indication that the model is memorizing the training data that it finds it difficult to classify unseen examples correctly

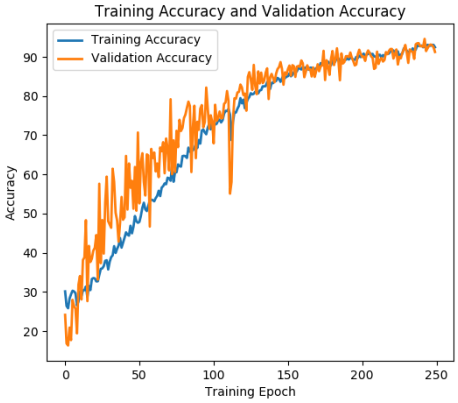


(b) Unweighted training and validation accuracy - Again, the unweighted accuracies are rising sharply towards perfection although the validation loss increases as well

Figure 5.4: Training and validation metrics of over-fitting model

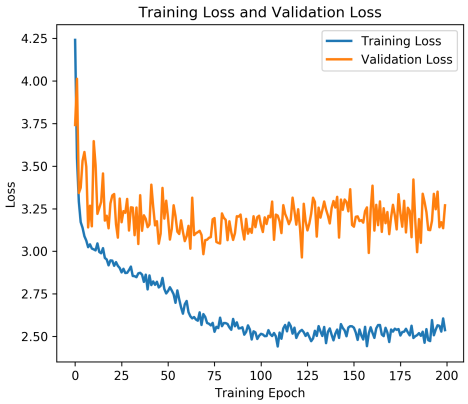


(a) Training and validation losses - Increasing validation loss is a clear evidence of over-fitting

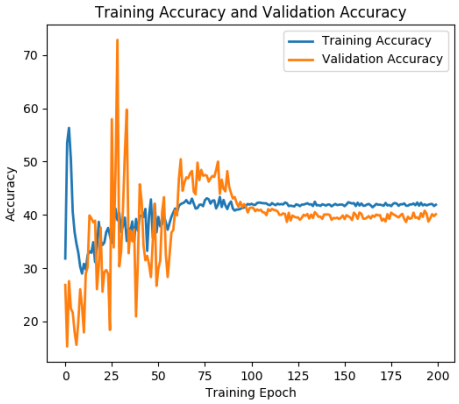


(b) Training and validation accuracies - Due to the extreme over-fitting, both validation and training accuracies continued rising reaching over 90% after 250 epochs

Figure 5.5: Training and validation loss and accuracy: model A

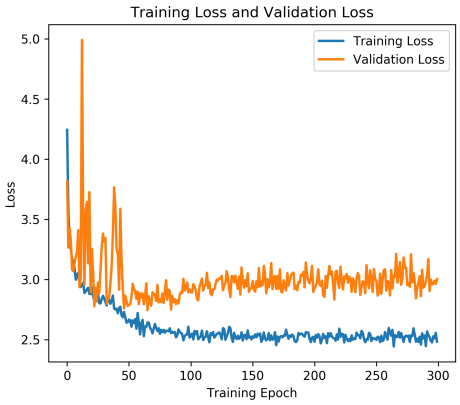


(a) Training and validation losses - Using the weighted accuracy and a revision of the loss function, the model began to behave to expectation. Particularly, the validation accuracy shown in (b) decreased relative to the training accuracy which fits with the observations in this picture.

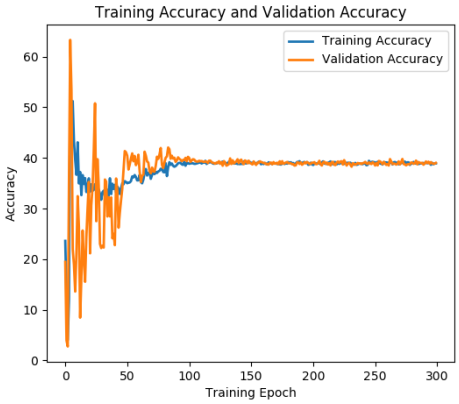


(b) Training and validation weighted accuracies - The plots of the weighted accuracies give a much better indication of nature of the model compared to the unweighted accuracies. At this point, the model is showing signs of convergence after the 100^{th} epoch.

Figure 5.6: Weighted loss and weighted accuracy training results: model B

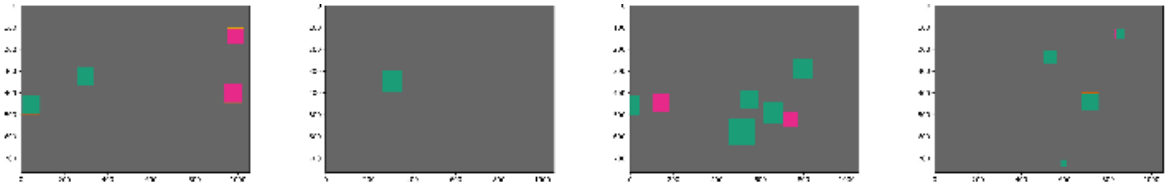


(a) Training and validation losses - Both losses appear to have converged to some minimum.

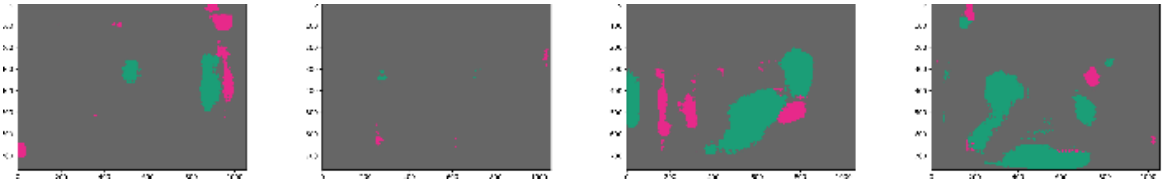


(b) Training and validation accuracies - Like the loss values, the weighted accuracies show clearly that model converged to a minimum after ≈ 130 epochs. The most interesting observation however is that, the model explains both seen and unseen data with near equal accuracies.

Figure 5.7: Weighted loss and weighted accuracy training results: model C



(a) Ground-truth labels - Given this masks as labels and the ground-truth images, the model produced the activations shown in (b)



(b) Final layer activations - the final activation maps are showing that the even at 43% training accuracy, the model is constructing the masks of unseen examples with modest accuracy. More importantly,unlike the square bounding boxes, the model is constructing regions which have similar geometric properties as the parasites

Figure 5.8: Activation maps of trained model

5.2 Results of Testing

Binary Classification Test Results

Hundred images from the growth-cycle stage classification dataset were used to evaluate the binary classification model. Fifty percent of the test data were positive images while the other 50% contained no parasites. Using the images from growth-cycle data to evaluate the binary classification model was motivated by the following main reasons:

1. There was a need to evaluate the model on a test set with similar global statistics between the two classes. This eliminated the influence of background and the context in the evaluation of the model. Since both datasets were Giemsa-thin blood smear images, the model was expected to produce accurate results if it indeed learned the characteristics of *Plasmodium* parasites when trained on dataset for binary classification. On the other hand, if it learned only the global statistics, it was expected to classify every image from the growth-cycle dataset as positive. The dataset for the growth-cycle stage classification task had the desired properties, hence the decision to use some for those images for testing.

2. There was no independent dataset for testing the binary classification model.

As anticipated, the model classified all the 100 images as positive although 50% of the images were negative. To ensure that the model's poor performance was not due to errors in the evaluation process, 100 images from the binary classification training dataset (50 positive and 50 negative images) were also used to evaluate the model. This time round, the model classified all the images perfectly into the two classes. This confirmed that the model's training performance was indeed based on global statistics of the images. The GPU took ≈ 54 seconds to evaluate the 100 images.

Grow-cycle Stages Classification Test Results

The growth-cycle stages classification model was tested on 35 images none of the which the model had seen prior to testing. Like the training dataset, the masks were con-

structed for the test images using their annotations. However, unlike the masks of the training data, the masks of the test data were generated purposely for measuring the accuracy of the predictions and not for gradient computations.

The three most promising models labeled A, B and C as discussed above were evaluated independently on the test data. The best model, model A, was able to predict the class of every pixel in the 35 images with a weighted accuracy of 85.86%[10] . Model B obtained a weighted accuracy of 51.19% while model C achieved an an accuracy of 44.63%. The architecture of best model, model A is shown in Figure 3.6. Given that semantic segmentation is generally a difficult task, it can be concluded that the results though relatively low compared to many categorizations tasks, were modest given the size of the dataset. More importantly, the model achieved similar performance on both training and testing. This indicates that it will perform well when applied to other Giemsa-stained thin blood smears.

[10]This model also achieved an unweighted accuracy of 90.10% on the test set

Chapter 6: Conclusion and Future Work

6.1 Summary

This research explored the application of convolutional neural networks in the detection *Plasmodium* parasites. In particular, a semantic segmentation model was built to classify the different growth-cycle stages of *Plasmodium* parasites. Through the use of techniques such as transfer learning and regularization methods, the model classified pixels belonging to normal cells, trophozoites or gametocytes with weighted accuracy of 85.86%. A binary classification model was also built to classify Giemsa-stained thin blood smear images into positive and negative classes. That model achieved an accuracy of 98.78% during training but performed poorly at test time because of large variations in the training data.

This empirical results from the experiments on the dense predictions are consistent with the literature that convolutional neural networks can be used to automate microscopy especially for the case of malaria diagnosis. In attempt to classify the different growth-cycle stages, this research became the first of its kind (to the best of my knowledge) to perform dense predictions on Giemsa-stained thin blood images. This provides a good foundation for complete segmentations of *Plasmodium* parasites from stained images. Also, the ability to identify different growth-cycle stages of *Plasmodium* parasites at the pixel level can help reveal some properties of the parasites which cannot be accessed otherwise.

6.2 Limitations

This section discusses the aspects of the research which had the potential to adversely affect the accuracy of the classification algorithms. The following were constraints on the performance of the models.

1. The VGG16 pre-trained model which was used to initialize the models was not in the problem domain. As stated in the Methodology, the VGG16 model was pre-trained on the ImageNet dataset which do not contain medical image data. The ImageNet pre-trained model was used because no pre-trained model in malaria diagnosis was found. Since transfer learning works well when the source model and the target model

are in the same domain, applying a model trained on cats and dogs to a model on *Plasmodium* parasites could have impacted the performance of the model negatively. To reduce the impact of the differences in the problem domain, the last three layers of the VGG16 model were re-trained. In addition, a fully convolutional layer was put on top of the the pre-trained model before upsampling. Since the first layers of CNNs typically learn general features such as curves, fine-tuning only the last layers improved the model's performance.

2. The dataset employed in the study was very small. It is a well established fact that CNNs perform very well when they are trained on large datasets. In fact, the recent successes of deep learning have been attribute mainly to advancements in archiving massive datasets (Brynjolfsson & Andrew, 2017). CNNs normally tend to over-fit small datasets which subsequently leads to poor performance at test time. As discussed above, several methods were employed to prevent over-fitting but they do not rule out the fact that the project could have benefited from more data.

3. Some of the parasite annotations were ill-constructed. The images were annotated using square bounding boxes because they required less time to construct. In using square bounding boxes, however, the annotations deviated from the true morphologies of the parasites. A gametocyte being crescent in shape cannot be segmented exactly with a square bounding box. Furthermore, there were slight variations in the mount of background the annotations added to infected cells. Finally, some annotations were not axis-aligned and had to be rotated. The rotations had the potential of increasing the area of the infected cells. This meant that some normal pixels could be wrongly labeled as belonging to one of the two growth-cycle stages. To address these problems, the model was built to a very low resolution of 17×26 pixels before upsampling. Nonetheless, some of the bounding boxes still had the potential to misrepresent the true content of the images.

4. The final limitation of the study is that the images had large resolutions. At the original resolution, only two images could be supplied to the model in a mini-batch. Both

the images and the masks were subsequently downsampled by a factor of two in order to feed the model with more images in a mini-batch. (At the downsampled resolutions, 17 images could be supplied to the model in a mini-batch). Although good downsampling techniques were used, the downsampling was still a potential source of error in the performance of the model.

6.3 Suggestions for Future Work

This section offers suggested extensions to this study. Besides employing measures to addressed the limitations above, the findings of this study can become widely applicable if the following extensions are explored.

1. A refinement of the algorithm to count the number of the different growth-cycle stages of the parasites per image. This feature will make it possible to estimate the density of the parasites on the images. This information is useful to health practitioners and researchers since it enables them to estimate the severity of the infection.

2. An improvement of the model to include all the four major growth-cycle stages of the *Plasmodium* parasite. This study considered only trophozoites and gametocytes in addition to normal cells due to constraints on image quality. Future extensions of this project should source data good enough to discriminate between all the different-growth cycle stages. The importance of classifying all life-cycle stages cannot be overstated.

References

Afridi, M. J., Ross, A., & Shapiro, E. M. (2018). On automated source selection for transfer learning in convolutional neural networks. *Pattern Recognition*, *73*, 65–75. Retrieved from http://dx.doi.org/10.1016/j.patcog.2017.07.019 doi: 10.1016/j.patcog.2017.07.019

Bibin, D., Nair, M. S., & Punitha, P. (2017). Malaria parasite detection from peripheral blood smear images using deep belief networks. *IEEE Access*, *5*, 9099–9108. doi: 10.1109/ACCESS.2017.2705642

Brynjolfsson, E., & Andrew, M. (2017). *What's driving the machine learning explosion?* Retrieved from https://hbr.org/2017/07/whats-driving-the-machine-learning-explosion

Chakrabortya, K. (2015). A combined algorithm for malaria detection from thick smear blood slides. *Journal of Health & Medical Informatics*, *06*(01), 1–6. doi: 10.4172/2157-7420.1000179

Chollet, F., et al. (2015). *Keras.* Retrieved from https://keras.io

Cires, D. C., & Giusti, A. (2012). Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. *NIPS*, 2852–2860.

Delves, M., Plouffe, D., Scheurer, C., Meister, S., Wittlin, S., Elizabeth, A., ... Leroy, D. (2012). The activities of current antimalarial drugs on the life cycle stages of plasmodium : a comparative study with human and rodent parasites. *PLoS Med*, *9*(2), e1001169. doi: 10.1371/journal.pmed.1001169

Díaz, G., González, F. A., & Romero, E. (2009). A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images. *Journal of Biomedical Informatics*, *42*(2), 296–307. Retrieved from http://dx.doi.org/10.1016/j.jbi.2008.11.005 doi: 10.1016/j.jbi.2008.11.005

Ghosh, S., Ghosh, A., & Kundu, S. (2014). Estimating malaria parasitaemia in images of thin smear of human blood. *CSI Transactions on ICT*, *2*(1), 43–48. Retrieved from

http://link.springer.com/10.1007/s40012-014-0043-7 doi: 10.1007/s40012-014-0043-7

Gonzales, G. (2016). *Giemsa staining of malaria blood films* (Tech. Rep.). World Health Organization.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative Adversarial Networks. , 1–9. Retrieved from http://arxiv.org/abs/1406.2661 doi: 10.1001/jamainternmed.2016.8245

Gopakumar, G. P., Swetha, M., Sai Siva, G., & Sai Subrahmanyam, G. R. (2017). Convolutional neural network-based malaria diagnosis from focus stack of blood smear images acquired using custom-built slide scanner. *Journal of Biophotonics*, *11*(January 2017), e201700003. doi: 10.1002/jbio.201700003

Gupta, S., Girshick, R., & Arbel, P. (2014). Learning Rich Features from RGB-D Images for Object Detection and Segmentation. *CoRR*, *abs/1407.5*. Retrieved from http://arxiv.org/abs/1407.5736

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers : surpassing human-level performance on ImageNet classification. *CoRR*, *abs/1502.0*. Retrieved from http://arxiv.org/abs/1502.01852

Kahama-Maro, J., D 'acremont, V., Mtasiwa, D., Genton, B., & Lengeler, C. (2011). Low quality of routine microscopy for malaria at different levels of the health system in Dar es Salaam. *Malaria Journal*, *10*. Retrieved from http://www.malariajournal.com/content/10/1/332 doi: 10.1186/1475-2875-10-332

Kanan, C., & Cottrell, G. W. (2012). Color-to-grayscale: does the method matter in image recognition? *PLoS ONE*, *7*(1), e29740. doi: 10.1371/journal.pone.0029740

Kareem, S., Kale, I., & Morling, R. C. S. (2012). Automated malaria parasite detection in thin blood films:-A hybrid illumination and color constancy insensitive, morphological approach. In *Ieee asia-pacific conference on circuits and systems, proceedings, apccas* (pp. 240–243). IEEE. doi: 10.1109/APCCAS.2012.6419016

Karunamoorthi, K. (2011). Vector control: A cornerstone in the malaria elimination cam-

paign. *Clinical Microbiology and Infection*, *17*(11), 1608–1616. Retrieved from http://dx.doi.org/10.1111/j.1469-0691.2011.03664.x doi: 10.1111/j.1469-0691.2011.03664.x

Kingma, D. P., & Ba, J. L. (2014). Adam: a method for stochastic optimization. *CoRR*, *abs/1412.6*, 1–15. Retrieved from http://arxiv.org/abs/1412.6980

Krizhevsky, A., Sutskever, I., & Geoffrey E., H. (2012). ImageNet classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, 1097–1105. doi: 10.1109/5.726791

Li, F.-F., Johnson, J., & Yeung, S. (2017). CS231n: Convolutional Neural Networks for Visual Recognition 2017. Retrieved from http://cs231n.stanford.edu/

Linder, N., Turkki, R., Walliander, M., Martensson, A., Diwan, V., Rahtu, E., ... Lundin, J. (2014). A malaria diagnostic tool based on computer vision screening and visualization of Plasmodium falciparum candidate areas in digitized blood smears. *PLoS ONE*, *9*(8). doi: 10.1371/journal.pone.0104855

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. In *2015 ieee conference on computer vision and pattern recognition (cvpr)* (pp. 3431–3440). doi: 10.1109/CVPR.2015.7298965

Mehanian, C., Jaiswal, M., Delahunt, C., Thompson, C., Horning, M., Hu, L., ... Bell, D. (2017). Computer-automated malaria diagnosis and quantitation Using Convolutional Neural Networks. *IEEE International Conference on Computer Vision Workshops Computer-Automated*, 116–125. doi: 10.1109/ICCVW.2017.22

Mehrjou, A., Abbasian, T., & Izadi, M. (2013). Automatic malaria diagnosis system. In *International conference on robotics and mechatronics, icrom 2013* (pp. 205–211). doi: 10.1109/ICRoM.2013.6510106

Moon, S., Lee, S., Kim, H., Freitas-Junior, L. H., Kang, M., Ayong, L., & Hansen, M. A. (2013). An image analysis algorithm for malaria parasite stage classification and viability quantification. *PLoS ONE*, *8*(4), 1–12. doi: 10.1371/journal.pone.0061812

Nanoti, A., Jain, S., Gupta, C., & Vyas, G. (2016). Detection of malaria parasite species and life cycle stages using microscopic images of thin blood smear. In *2016 international*

conference on inventive computation technologies (icict), coimbatore, 2016 (pp. 1–6). doi: 10.1109/INVENTIVE.2016.7823258

Nonvignon, J., Aryeetey, G. C., Malm, K. L., Agyemang, S. A., Aubyn, V. N. A., Peprah, N. Y., ... Aikins, M. (2016). Economic burden of malaria on businesses in Ghana: a case for private sector investment in malaria control. *Malaria Journal*, *15*(454). doi: 10.1186/s12936-016-1506-0

Ojha, P. K., & Roy, K. (2015). The Current status of antimalarial drug research with special Reference to application of QSAR models. *Combinatorial Chemistry & High Throughput Screening*, *18*(2). doi: 10.2174/1386207318666141229125527

Park, H. S., Rinehart, M. T., Walzer, K. A., Ashley Chi, J. T., & Wax, A. (2016). Automated Detection of P. falciparum using machine learning algorithms with quantitative phase images of unstained cells. *PLoS ONE*, *11*(9). doi: 10.1371/journal.pone.0163045

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A. A. (2016). Context Encoders: Feature Learning by Inpainting. *CoRR*, *abs/1604.0*. Retrieved from `http://arxiv.org/abs/1604.07379` doi: 10.1109/CVPR.2016.278

Penas, E. K., Rivera, T. P., & Naval.Jr, C. P. (2017). Malaria parasite detection and species identification on thin blood smears using a convolutional neural network. In (pp. 1–5). doi: 10.1109/CHASE.2017.51

Pinkaew, A., Limpiti, T., & Trirat, A. (2015). Automated classification of malaria parasite species on thick blood film using support vector machine. In *Biomedical engineering international conference (bmeicon-2015)* (pp. 1–5).

Pollak, J. J., Houri-Yafin, A., & Salpeter, S. J. (2017). Computer vision malaria diagnostic systems—progress and prospects. *Frontiers in Public Health*, *5*(219), 1–5. doi: 10 .3389/fpubh.2017.00219

Poostchi, M., Silamut, K., Maude, R. J., Jaeger, S., & Thoma, G. (2018). Image analysis and machine learning for detecting malaria. *Translational Research*, *194*, 36–55. Retrieved from `https://doi.org/10.1016/j.trsl.2017.12.004` doi: 10.1016/j.trsl.2017.12.004

Purnama, I. K. E., Rahmanti, F. Z., & Purnomo, M. H. (2013). Malaria parasite identifica-

tion on thick blood film using genetic programming. In *3rd international conference on instrumentation, communications, information technology, and biomedical engineering* (pp. 194–198).

Quinn, J. A., Nakasi, R., Mugagga, P. K., B., Byanyima, P., Lubega, W., & Andama, A. (2016). Deep Convolutional Neural Networks for microscopy-based point of care diagnostics. In *Proceedings of international conference on machine learning for health care 2016* (Vol. 56).

Radiuk, P. M. (2018). Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. *Information Technology and Management Science,*, *20*(1), 20–24. Retrieved from `https://www.degruyter.com/view/j/itms` doi: 10.1515/itms-2017-0003

Rawat, W., & Wang, Z. (2017). Deep Convolutional Neural Networks for image classification: A comprehensive review. *Neural Computation*, *29*(9), 2352–2449. Retrieved from `http://arxiv.org/abs/1706.02451` doi: 10.1162/NECO

Rosado, L., Correia, M., Elias, D., & Cardoso, J. S. (2016). Automated detection of malaria parasites on thick blood smears via mobile devices. *Procedia Computer Science*, *90*(2016), 138–144. doi: 10.1016/j.procs.2016.07.024

Roy, K., Sharmin, S., Mufiz Mukta, R. B., & Sen, A. (2018). Detection of malaria parasite in giemsa blood sample using image processing. *International Journal of Computer Science and Information Technology*, *10*(1), 55–65. Retrieved from `http://aircconline.com/ijcsit/V10N1/10118ijcsit05.pdf` doi: 10.5121/ijcsit.2018.10105

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. doi: 10.1007/s11263-015-0816-y

Sathpathi, S., Mohanty, A. K., Satpathi, P., Mishra, S. K., Behera, P. K., Patel, G., & Dondorp, A. M. (2014). Comparing Leishman and Giemsa staining for the assessment of peripheral blood smear preparations in a malaria-endemic region in India. *Malaria Journal*, *13*(1), 512. doi: 10.1186/1475-2875-13-512

Savkare, S. S. (2011). Automatic detection of malaria parasites for estimating parasitemia. *International Journal of Computer Science and Security (IJCSS)*, *5*(3), 310–315.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, *abs/1409.1*, 1–14. Retrieved from http://arxiv.org/abs/1409.1556 doi: 10.1016/j.infsof.2008.09.005

Sivaramakrishnan, R., Antani, S., & Jaeger, S. (2017). Visualizing Deep Learning Activations for Improved Malaria Cell Classification. *First Workshop Medical Informatics and Healthcare (MIH 2017), PMLR*(October), 40–47.

Srinivas. (2015). *Life cycle of Plasmodium parasite.* Retrieved from https://www.malariasite.com/life-cycle/

Wellems, T. E., & Plowe, C. V. (2001). Chloroquine-Resistant Malaria. *The Journal of Infectious Diseases*, *184*(6), 770–776.

White, N. J. (2008). The role of anti-malarial drugs in eliminating malaria. *Malaria Journal*, *7*(1), 1–6. doi: 10.1186/1475-2875-7-S1-S8

WHO. (2015). *How malaria RDTs work.* Retrieved from http://www.who.int/malaria/areas/diagnosis/rapid-diagnostic-tests/about-rdt/en/

WHO. (2017). *Malaria.* Retrieved from http://www.who.int/mediacentre/factsheets/fs094/en/

Wojna, Z., Ferrari, V., Guadarrama, S., Silberman, N., Chen, L., Fathi, A., & Uijlings, J. R. R. (2017). The Devil is in the Decoder. *CoRR*, *abs/1707.0*. Retrieved from http://arxiv.org/abs/1707.05847

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *In Advances in Neural Information Processing Systems*, *27*(NIPS '14), 3320–3328. Retrieved from http://arxiv.org/abs/1411.1792

Yuheng, S., & Hao, Y. (2017). Image segmentation algorithms overview. *CoRR*, *abs/1707.0*. Retrieved from http://arxiv.org/abs/1707.02051