National College of Ireland

Malaria Diagnosis Using Convolutional Neural Networks on Thin Blood Smear Images

MSc Research Project
Data Analytics

Shikhar Srivastava
Student ID: x18106960

School of Computing
National College of Ireland

Supervisor: Mr. Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shikhar Srivastava
Student ID:	x18106960
Programme:	Data Analytics
Year:	2018
Module:	MSc Research Project
Supervisor:	Mr. Christian Horn
Submission Due Date:	20/12/2018
Project Title:	Malaria Diagnosis Using Convolutional Neural Networks on Thin Blood Smear Images
Word Count:	6240
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	12th August 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Malaria Diagnosis Using Convolutional Neural Networks on Thin Blood Smear Images

Shikhar Srivastava
x18106960

Abstract

Malaria is a life-threatening disease which has claimed a lot of lives in the past and is still considered as one of the most life-threatening diseases. It is commonly diagnosed by taking blood samples from patients and preparing slides which are then observed by experts under a powerful microscope to check for infected cells. The blood samples are taken several times and are sent to laboratories for expert examination. This is a time-consuming process which involves manual intervention. Computer vision has proven to be very helpful in many image classification and object detection problems. Machine learning and deep learning are nowadays widely used to solve many real-life problems which is achieved by training these algorithms up to an extent after which they can identify and classify images. This research project uses thin blood smear images and aims to build deep learning models to identify whether any of the blood cells are infected by malaria parasite or not. First a normal CNN model has been built to classify images in two categories i.e. Infected and Healthy followed by Resnet50 for same purpose. After this, Faster RCNN has been implemented for detecting infected blood cells.

Keywords: Malaria Diagnosis, Image Classification, Object Detection, Keras, Tensorflow, Python, CNN, ResNet-50, Faster RCNN, Google Colaboratory

1 Introduction

1.1 Background

Malaria is one of the major health problems the world had been dealing with in the past, and scientists are still finding efficient ways to lower the death toll claimed by it. Malaria parasite is spread by the bite of female Anopheles mosquito. These malaria causing parasites are called Plasmodium parasites. In total there are 5 parasite species that cause malaria in humans among which P. falciparum and P. vivax pose greatest threat. In 2017, 435,000 people died across the globe from malaria most of them belonging to African region. The African continent was home to around 92% of malaria cases and 93% of malaria deaths on a global scale *WHO* (2019).

Although over the last decade or two, malaria cases are reduced to over half but, still getting rid of it completely may take some time. To achieve complete elimination of this

fatal disease, there is a need to understand more about the disease and the parasite which causes it and come up with new tools, technologies and drugs. Due to this, approved anti-malarial vaccine and drugs are being developed of which some of them are being used for human clinical trials Cowman et al. (2016). Its not just the drugs and vaccines which need to be developed, it is also necessary to come up with new methods for malaria diagnosis which are faster than the existing processes and require least manual intervention so that blood sample testing could be done quickly followed by the required treatment. This leads us to our problem statement.

1.2 Problem Statement

Keeping the long-term complete malaria elimination goal in mind, it has become necessary to not just only develop new vaccines and drugs but also develop new efficient methods to diagnose malaria. Malaria diagnosis is time taking and involves taking multiple blood samples from patients in a course of multiple days. Computer vision nowadays is widely used in many practical situations for the purpose of classifying images or detecting an object. This research project intends to use these technologies for finding out malaria infected thin blood smears and is an attempt towards diagnosing malaria using deep learning in near future.

1.3 Research Question

Based on the problem stated above, following research questions has been formulated:

Research Question 1: *"Can deep learning models help diagnose malaria by applying computer vision technology on thin blood smear images?"*

Research Question 2: *"Can this research achieve presentable results for a small sample of images?"*

1.4 Scope of Study

To test and answer our research question, research project uses 1364 P. vivax infected human blood smear images from *Broad Bioimage Benchmark Collection* (2019) containing roughly 80k cells. This data consists of two classes of uninfected cells (RBCs, leukocytes) and four classes of infected blood cells (gametocytes, rings, trophozoites, and schizonts).

2 Related Work

Malaria is a deadly disease which is a cause of death for many, especially in the African continent as discussed above. Due to this reason, a lot of research work is going on around it to develop new drugs and to understand the biology of the parasite. New methods to

diagnose malaria in a fast and efficient manner are also needed to be developed so that the treatment could start as soon as possible.

This chapter discusses about researches done on malaria followed by discussion on Tensorflow and usage of Keras on top of Tensorflow. This section also discusses about Convolutional Neural Networks in general and further discusses various research work done on ResNet50. In the last part, discussion has been done on object detection and usage of Faster RCNN for detecting objects.

2.1 Malaria

As discussed in the introduction section above, malaria is caused by parasites transmitted by female anopheles mosquitoes. This life-threatening disease is responsible for deaths of many. Malaria still is a cause of many deaths in the African continent but number of fatalities due to malaria has decreased considerably in the last two decades. A research carried out on the estimation of these statistics reveal that Plasmodium falciparum infections in Africa has nearly halved from 2000 to 2015 Bhatt et al. (2015). The paper also states that there few preventive measures which were taken in the course of these 15 years that led to these low figures. Preventive measures like insecticide-treated bed nets (ITNs), indoor residual spraying (IRS) were used to kill mosquitos to control the spread of this disease. Out of the two, ITNs proved to be the largest contributor in averting the disease (68%). Researchers also found that prompt clinical treatments using artemisinin based combination therapy (ACT) was also one of the reasons for depleting number of fatalities related to malaria. Although infections in Africa has nearly halved from 2000 to 2015, there has recently been a severe outbreak in Burundi Beaumont (2019) killing around 1800 people since starting of this year. WHO said that the outbreak is severe and number of deaths are almost equal to number of deaths during Ebola crisis in Democratic Republic of Congo. Worst thing which has happened is the government there has refused to declare an emergency.

Another research was carried out to analyse about the number of malaria infections only in the US Mace et al. (2018). The researchers acquired a rich dataset from National Malaria Surveillance System (NMSS), the National Notifiable Diseases Surveillance System (NNDSS) and CDC reference laboratory reports in which they found 1517 confirmed malaria cases in the year 2015. There are other researches which have been done to control the spread malaria like Leffler et al. (2017). In this research, researchers found that large scale deletions and duplications of genes which is commonly called as structural variants are linked to this disease and common in human genome. Their research was based on Plasmodium falciparum which is the major cause of child mortality in Africa. The parasites enter RBCs through interactions with surface proteins including glycophorin. They identified a specific SV which encodes hybrid glycophorin proteins and decrease the risk of severe malaria by around 40%. Another similar reseach Koning-Ward (2018) investigates how malaria parasite survives and reaches the cytoplasm of RBCs using the proteins present.

Many other researches have been carried out on the prevention from malaria parasite with the help of vaccination. Draper et al. (2018) discusses in detail about the recent

advances and future prospects of malaria vaccines. The other thing which needs to be taken care of is the time taken to diagnose malaria. The process takes some time because multiple blood samples are taken for examination in laboratories. Here is where we can make use of deep learning tech to help diagnose malaria fast and quick.

2.2 Tensorflow

This research project is revolving around finding a suitable way to make malaria diagnosis process fast and efficient. To achieve this, the research attempts to build deep learning models for images classification and object detection. As this project will be dealing with deep learning models, it is using Tensorflow for building these models because Tensorflow is one of the best libraries for deep learning available nowadays.

Tensorflow works on dataflow graphs which is one of the main features of Tensorflow. It uses dataflow graphs in the backend to represent computation, flow and operations and it can also efficiently work on heterogeneous environments Abadi et al. (2016). The first step which Tensorflow follows is to build a dataflow graph which can clearly be seen in the Tensorflow architecture in Figure 1.

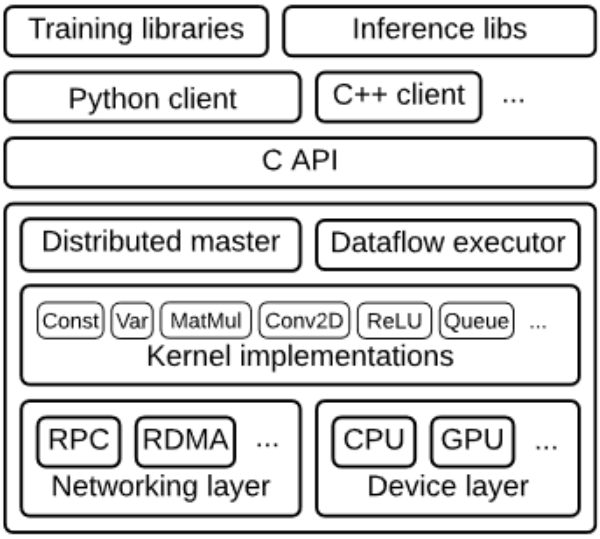

Figure 1: High Level Architecture of Tensorflow Abadi et al. (2016)

Tensorflow is a complex library and to understand how Tensorflow works, one needs to understand more about how it operates on dataflow graphs. TensorFlow Graph Visualizer is a key component of tensorflow machine intelligence platform Wongsuphasawat et al. (2018). The researchers in this paper have given a detailed explanation of dataflow graphs. With the help of TensorFlow Graph Visualizer, users can easily analyse the models they have built because TensorFlow Graph Visualizer builds a high-level diagram of their model with an added option to explore the nested structure of their model, if needed. Tensorflow nowadays is widely used to build deep learning networks for image classification. One such research carried out recently uses Tensorflow for classification of liver disorders Raghesh Krishnan et al. (2018). The researchers are using ultrasound

images to classify nine types of liver disorders both focal and diffused liver disorders.

There are also a lot of researches which happen to improve Tensorflow library. One such research was recently published in the month of May by Google Cohen et al. (2019). In this research, researchers have described a new framework Neural Query Language (NQL) for accessing soft symbolic databases using only differential operators from Tensorflow. This is a new type of dataflow language built in python and Tensorflow which can be used to build complex neural network models in a convenient manner. NQL is closely related to TensorLog but better than it. Researchers also mentioned that NQL is not a logic but a dataflow language like Pig.

2.3 Keras on Tensorflow

Although Tensorflow is one of the best deep learning libraries, but it requires some complex coding to build a deep learning model in Tensorflow. To make the task to build a model using Tensorflow, Keras came into existence. Keras is a library which works on top of Tensorflow making the coding part comparatively easy and understandable.

There have been a lot of research using Keras on Tensorflow. Keras can not only be integrated with Tensorflow but also with other deep learning platforms like Theano and CNTK. One such interesting research done using Keras on Tensorflow is Yin et al. (2018). The researchers in this research were collecting eye tracking data using a device called Tobii X60 which was further used to decide whether the user id looking at google news or news map. The device recorded GazePointX and GazePointY along with few other fields. These GazePoint values were stored in a 2D array which was equal to size of screen resolution and researchers used MongoDB to store data. Figure 2 shows the two type of news made available to users.

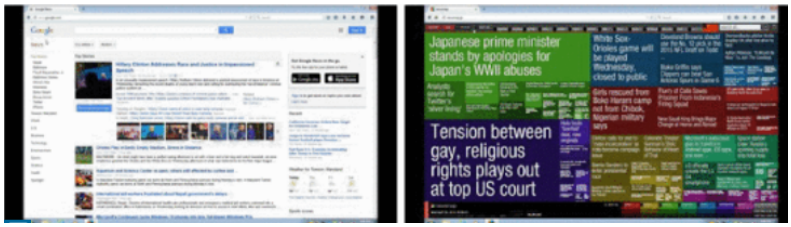

Figure 2: Left: google news UI Right: news map UI

There are many other research papers on deep learning which use Keras on Tensorflow for building a model like Wagner et al. (2019) uses Keras on Tensorflow to classify natural forests and eucalyptus plantations utilizing remotely sensed images. Another group of researchers build a Twitter health Surveillance (THS) system in Rodrguez-Martnez and Garzn-Alfonso (2018) where they collected 56013 tweets which were further classified in three categories: 0 - tweet is not about diseases, 1 - tweet is related with diseases and 2 - tweet is ambiguous.

2.4 Convolutional Neural Network

As CNNs are very popular for image recognition and object detection, this research project uses CNNs for classifying malaria infected & healthy blood smears and for detecting infected cells in blood smears.

The breakthrough research done on convolutional neural networks by Krizhevsky et al. (2012) marked the time when CNN started to become very popular in the field of computer vision. They achieved ground breaking results from their CNN with five convolutional layers and three fully connected layers and each fully connected layer containing 4096 neurons.

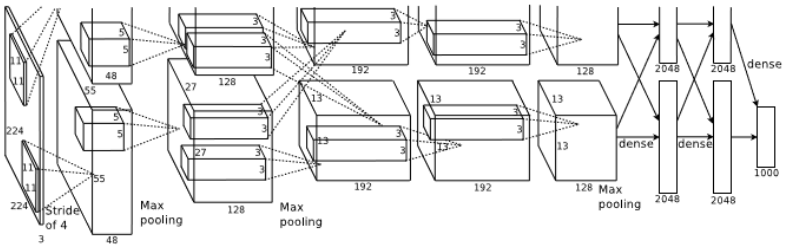

Figure 3: An illustration of the architecture of CNN in Krizhevsky et al. (2012), explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The networks input is 150,528 dimensional, and the number of neurons in the networks remaining layers is given by 253,440:86,624:64,896:64,896:43,264:4096:4096:1000

2.5 Resnet50 (CNN)

After the breakthrough results achieved by a group of researchers, people started doing more research work in CNN. A new concept of residual learning was introduced by a group of researchers at Microsoft He et al. (2015). It was widely noticed that when the depth of network increases, accuracy gets saturated and then degrades rapidly and this is not caused by overfitting but because adding more layers to a considerably deep model increases the training error. This problem is popularly known as degradation. To address the problem of degradation, the researchers came with deep residual learning framework. A simple building block of residual learning is shown in Figure 4.

The approach followed by the researchers to address this problem was that instead of hoping that each few stacked layers directly fit a desired underlying mapping, they explicitly let these layers fit a residual mapping. ImageNet dataset was trained, validated and tested 18-layers, 34-layers, 50-layers, 101-layers and 152-layers. It was noticed that the training error decreased as the number of layers increased. These models built by Microsoft researchers was then named as ResNet-18, ResNet-34, ResNet-50, ResNet-101 and ResNet-152. This research project uses ResNet-50.

After the research work on Residual Networks by Microsoft, many other works were also

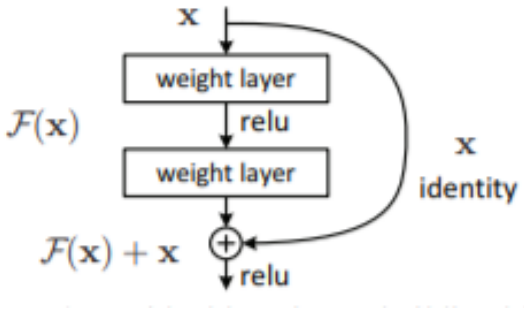

Figure 4: A simple building block of residual learning

done in following years in which researchers tried to optimise the performance of ResNets. In Akiba et al. (2017), a group of researchers trained the ResNet-50 model for 90 epochs in 15 minutes with 1024 Tesla P100 GPUs. They demonstrated that highly-parallel training is possible with a large minibatch size without losing accuracy. To maintain the accuracy, the researchers employed several techniques such as RMSprop warm-up, batch normalization without moving averages, and a slow-start learning rate schedule. Dataset used for this training was same as used by Microsoft researchers i.e. ImageNet. Later in March 2019 another paper Yamazaki et al. (2019) was published which achieved training time of 74.7 seconds, a huge drop from the former research we just discussed. Researchers in this paper addressed the challenge which people usually face to achieve high scalability on large clusters without compromising accuracy. They used some enhanced optimization techniques due to which they achieved a training time of 74.7 seconds on 2048 GPUs and achieved validation accuracy of 75.08%.

ResNet-50 has also been used in many cases to classify things like Rezende et al. (2017) uses residual network architecture to classify malicious software. Here, malware samples were represented as byteplot grayscale images. The experiment gave 98.62% accuracy on a dataset of 9339 samples from 25 different families.

2.6 Object Detection & Faster RCNN

Although image classification is one of the best approaches but, apart from performing image classification, this research project also tries to build an object detection model for detecting infected cells in a blood smear.

Quality research work in the field of object detection started around two decades ago when in 2001 two researchers wrote Michael Jones (2001). In this research paper, they explain their approach towards building an object detection model which was distinguished by three key contributions. First being the concept of Integral Image which made the computation fast, second is building an algorithm based on AdaBoost to select critical features from images and, third to get rid of the background of images to spend more time on promising object-like regions. In real time applications, their detector ran at 15 frames per second.

Faster RCNN is one of the most popular object detection models. A group of researchers at Microsoft came up with a new concept called Regional Proposal Network that shares full-image convolutional features with the actual detection network Ren et al. (2015). They defined RPN as a fully-convolutional network that predicts object bounds and scores at each position. RPN networks are trained end-to-end to generate high quality proposals, which are further used by Fast RCNN for detecting objects. This type of architecture was named as Faster RCNN. Later a lot of researches were done to further improve the performance of Faster RCNN like Sun et al. (2018) where researchers achieved state-of-the-art results after combining a number of strategies like feature concatenation, hard negative mining, multi-scale training, model pre-training, and proper calibration of key parameters while implementing Faster RCNN.

3 Methodology

For this research project, CRISP-DM methodology is used which is one the most popular methodologies used for implementing a deep learning, descriptive analytics or a machine learning project Faure (2018). CRISP-DM is composed of 6 stages which leads to successful completion of a project as shown in Figure 5.

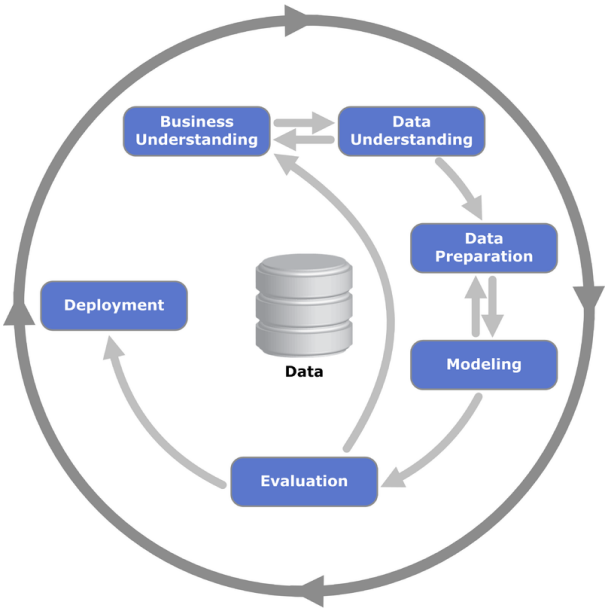

Figure 5: Six Stages of CRISP-DM

3.1 Business Understanding

In order to to build a deep learning model, it is important to first build an understanding about the problem we are trying to address. Keeping this in mind, a thorough understanding of malaria and its impact was developed which included the cause of this disease,

how is it diagnosed and how is it treated. The research has also cited its impact in the past in terms of fatalities and the current situation.

3.2 Data Understanding

Keeping the objective of this research project in mind which is to apply deep learning models to diagnose malaria, this project uses data from Broad Institute website. Data used in this project contains P. vivax infected human blood smears images. The images contain four classes of infected cells (gametocytes, rings, trophozoites, and schizonts) and two classes of healthy cells (RBCs and leukocytes). A total of 1364 images were downloaded from Broad Institute website.

3.3 Data Preparation

Data preparation which in this research project is image pre-processing is one of the most important steps. Image pre-processing for this project works in a loop as also mentioned in the CRISP-DM methodology. There are a series of operations which were performed on the images before feeding them in our CNN models. As this project uses CNN models for classifying the blood smear images in two classes i.e. Infected, Uninfected and also uses Faster RCNN for detecting infected blood cells in the images, so two different sets of image pre-processing were required to do, one for the purpose of classifying images and another for detecting infected blood cells in images. There are few pre-processing steps which are common in both types of pre-processing done like applying gaussian blur. When the dataset was downloaded, all the images were present in a single folder and two csv files were also downloaded. After pre-processing, all images present in same folder setup was used as it is for CNN and ResNet-50 models for classification purpose. For implementing Faster RCNN, these images were first separated in two different folders i.e. Train and Test. It was later noticed that the test images look different from the train images, so just for the task of detecting infected blood cells accurately, few images in test folder having bounding box co-ordinates were moved to train folder and vice-versa so that the model is trained to detect accurately.

For the approach of classifying images, all the images were resized to 224, 224. The images were then cropped into a circular shape. To tackle the problem of class imbalance and smaller number of images for training the models, all the images were rotated by certain angles which summed up to 4075 images in total as shown in Table 1.

Classes	Train	Test
0(Uninfected)	(189 images x 10 angles) 1890	(3 images x 10 angles) 30
1(Infected)	(1019 images x 2 angles) 2038	(117 images x 1 angle) 117

Table 1: Rotated Images Count

For our second approach which is about detecting infected cells in blood smear images,

original number of images were used i.e. 1364. Here it was noticed that train images had different size as compared to test images, so all the test images were reduced to width=1600, height=1200 from width=1944, height=1383. It was also noticed that the test images look a little different from the train images which might cause a problem while detecting infected cells in test images as stated above. Additionally, all the train and test images were then converted to grayscale and then GaussianBlur was applied using cv2 in python to make them look a little similar. After this pre-processing all images were stored in a single folder from where they were pulled and and stored in two different folders i.e. Train and Test folders.

3.4 Data Modelling

After preparing data, first a CNN with 9 layers was built and images were fed in the CNN for the purpose of classifying them into infected and healthy blood smear images. This model was then compiled with loss function categorical_crossentropy and fitted.

While doing literature review, got inspired by He et al. (2015) and planned to use ResNet-50 as a second CNN model for classifying blood smear images into infected and healthy categories. As library keras already contains a pre-trained ResNet-50 model, so it was not required to build a new one from scratch. Six new layers were then added in ResNet-50 in order to train it for classifying the blood smear images. Only the newly added layers were then made trainable and not the entire ResNet-50 model.

Later, after implementing the above two mentioned CNN models, found a potential scope for building an object detection model for detecting infected cells in the blood smear images. Keeping this in mind, the research project later tries to build a Faster RCNN model to detect infected blood cells. This was achieved by making use of the bounding box co-ordinates in the training set. The Faster RCNN model for detecting infected cells in blood smear images was built by following an excellent tutorial from Sharma (2018).

As the neural networks are considered to be the best to solve computer vision problems, this project uses CNNs for each of the above two mentioned approaches.

3.5 Evaluation

Apart from building and using the models for respective classification and object detection jobs, it is very important to evaluate the performance of the models. For the classification models used in this research project, evaluation is done based on the Accuracy, Precision, Recall and F1-score values obtained from the confusion matrix as shown in equations 1, 2, 3 and 4 below.

$$Accuracy = \frac{TP + TN}{Total} \qquad (1) \qquad\qquad Precision = \frac{TP}{TP + FP} \qquad (2)$$

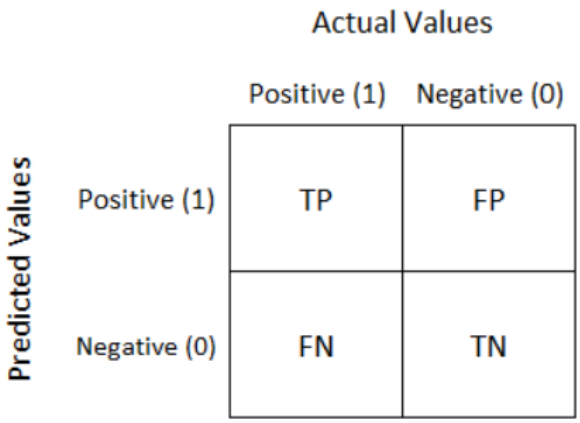

Figure 6: Confusion Matrix

$$Recall = \frac{TP}{TP + FN} \quad (3) \qquad F1 - score = \frac{2 * Recall * Precision}{Recall + Precision} \quad (4)$$

However, in the case of detecting infected blood cells though Faster RCNN for test images we get a probability score for each bounding box it creates for a class. The threshold for this probability score is set to be 0.8 or 80%. What this means is that if the model predicts any bounding box which is having probability score less than 0.8 that prediction will not be considered and will be discarded. All the predictions having probability equal to or greater than 0.8 will be taken into consideration. Faster RCNN can also be evaluated by building line charts for RPN(Region Proposal Network) accuracy and loss and Detector(classification Network) accuracy and loss which are shown in the results section.

3.6 Deployment

The last stage of CRISP-DM is not really applicable for this research project because the model is not going to be deployed anywhere.

4 Design Specification

There are many open source and paid platforms in market today for machine learning and building deep learning models. This research project uses Python programming language version 3.6.9 for building all models as well as pre-processing of all images. Anaconda was installed as a backend platform. All the draft versions of codes were written in Spyder (pre-installed in Anaconda) and tested first. A new environment was created in Anaconda solely for this project. As the project required some GPU computations (to

make the execution faster), all the deep learning models built were finally executed and evaluated on Google CoLab which provides a free GPU service. To use Google CoLab, all respective files and folders were uploaded on Google drive which can be mounted on Google CoLab. Google CoLab is very handy to use not just because of the reason that it provides free GPU but also that you dont have to install any libraries there, almost all libraries are pre-installed on CoLab. Three deep learning models were built in total, two for classifying images into infected and healthy blood smears and one for detecting infected blood cells in blood smears.

4.1 Image Classification of Blood Smears

Following models were built for classifying blood smear images into healthy and infected:

4.1.1 CNN from Scratch using Keras

At first, a model was built from scratch using Keras in Python. Model diagram is shown in Figure 7 below.

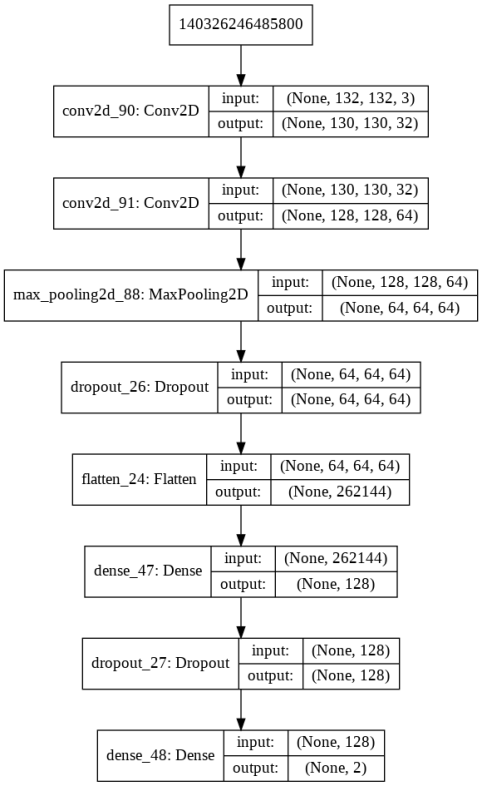

Figure 7: CNN Model Plot

As it is visible from above figure, the model uses two convolutional layers and one Max-Pooling layer. The last dense layer gives class of image as output. Model was made to train for 20 epochs.

4.1.2 Pre-Trained ResNet-50

As cited earlier in literature review, ResNet-50 is considered as one of the best deep learning models for computer vision problems and works on the concept of Residual Learning. As the name suggests, ResNet-50 is composed of 50 layers and 23.5M parameters. It was thoroughly trained on ImageNet data. Although it has 23.5M parameters but for this research project only those parameters are trained which are in the newly added layers. In this project, six new layers are added at the bottom of the original ResNet-50 and only these layers were trained, and rest of the model was left as it is. The idea behind this was to use full power of ResNet-50 while simultaneously being able to classify our images in the end. Input images are of size 224X224. ResNet-50 is having 50 layers so unlike CNN built before this, it was not possible to paste the model plot of ResNet-50 here. The new layers added in original ResNet-50 and trainable parameters after adding new layers are shown in Figure 8 and 9 below.

```
image_input = Input(shape=(224, 224, 3))

model = ResNet50(weights='imagenet',include_top=False)
#model.summary()
last_layer = model.output
x = GlobalAveragePooling2D()(last_layer)
# add fully-connected & dropout layers
x = Dense(512, activation='relu',name='fc-1')(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu',name='fc-2')(x)
x = Dropout(0.5)(x)
# a softmax layer for 2 classes
out = Dense(num_classes, activation='softmax',name='output_layer')(x)

# this is the model we will train
custom_resnet_model2 = Model(inputs=model.input, outputs=out)
```

Figure 8: New Added Layers

```
activation_343 (Activation)     (None, None, None, 2 0          add_112[0][0]
global_average_pooling2d_7 (Glo (None, 2048)         0          activation_343[0][0]
fc-1 (Dense)                    (None, 512)          1049088    global_average_pooling2d_7[0][0]
dropout_13 (Dropout)            (None, 512)          0          fc-1[0][0]
fc-2 (Dense)                    (None, 256)          131328     dropout_13[0][0]
dropout_14 (Dropout)            (None, 256)          0          fc-2[0][0]
output_layer (Dense)            (None, 2)            514        dropout_14[0][0]
==================================================================================================
Total params: 24,768,642
Trainable params: 1,180,930
Non-trainable params: 23,587,712
```

Figure 9: Trainable Parameters

4.2 Detection of Infected Blood Cells

A Faster RCNN model was used to detect infected blood cells in blood smear images. The model was only trained for infected blood cells and only the respective bounding boxes were fed in the model. Idea behind this was to have a model which can detect infected blood cells from an image when tested. Faster RCNN code was downloaded from GitHub (link present in the tutorial cited above). After investigating the code, it was noticed that Faster RCNN uses VGG or ResNet-50 as base models for detecting objects.

Code for training and testing for Faster RCNN is already available as mentioned earlier but, still few changes were made in the python files of respective codes. A small change

was made in simple_parser.py in which image was converted to numpy array on line 36 and instead of cv2, PIL was used to open images. Second change was made in train_frcnn.py where number of epochs was changes from 2000 to 30. A small change was also made in test_frcnn.py where threshold for the bounding boxes was set to 0.3. Threshold plays an important role in predicting bounding boxes for test images. Threshold was kept very low to get an idea about how the model is performing at first but later was increased to 0.8. Threshold 0.3 particularly means that bounding boxes having probabilities less than 30% will not be taken into consideration. The Faster RCNN model achieved following metrics on 30th epoch as shown in Figure 10.

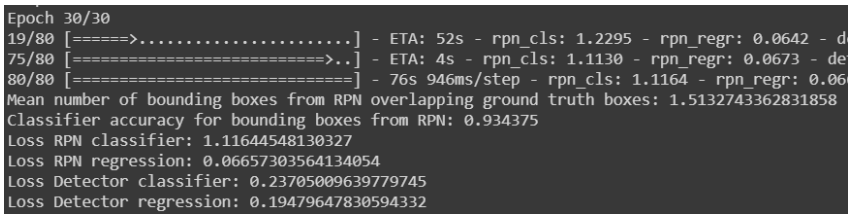

Figure 10: Faster RCNN Epoch 30 Output

The concept behind Faster RCNN is that first the image is passed through a convolutional layer which generates feature maps. After this a sliding window is run throughout this feature maps. The size of sliding window can be set accordingly. Thereafter, anchors are generated which have the same centre. A value is then calculated for each anchor to check how it overlaps with the ground truth boxes. The features extracted from the sliding window are then given as input to smaller network which performs two tasks, classification(cls) and regression(regr). Regression task predicts the bounding box and classification task gives a probability value which determines whether the bounding box contains the object or not.

5 Implementation

The implementation of this research project is split into two parts, one involving the classification task and one involving object detection as shown in Figure 11.

Data for this research project was collected from Broad Institute website which contains a total of 1364 images of blood cell smears infected by malaria as well as healthy blood smears. The images were then pre-processed for respective classification and object detection tasks in Python. Before this, the original csv files downloaded from source contained four classes of infected cells and two classes of healthy cells. These csv files were transformed in excel to represent only two classes for each image, whether the blood smear image is infected or not by writing down a simple formula in excel. Latest version of Anaconda was downloaded with Python 3.6. A new environment tensorflow was created in Anaconda for this project. As Anaconda comes along with some pre-installed tools, so Spyder was used for all the pre-processing of images at first, but later all pre-processing tasks were migrated to Google CoLab. Keras, Pillow, OpenCV and numpy was installed in the new environment using Anaconda prompt for the pre-processing task. After their

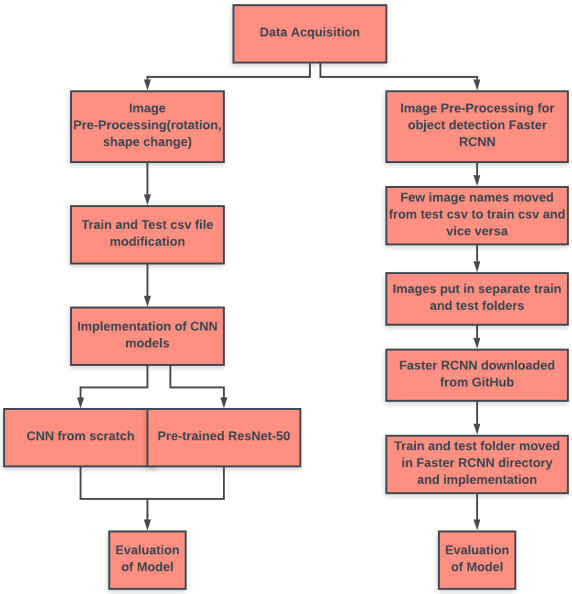

Figure 11: Implementation Process Flow

installation, these libraries were used in Spyder for pre-processing. The processed images were stored in a single folder from where they were later pulled and stored in a numpy array using tqdm library and respective training and testing csv files for classification.

As all the images were rotated up to 10 angles, the csv files further needed changes. Respective rotation angle was concatenated at the front of each image name using csv library in Python. Later, these new names were assigned to respective classes using VLOOKUP in excel. After this a classification CNN model was built from scratch, training and testing images were pulled in using tqdm library and stored in numpy array. After training and testing of the model, the model was evaluated by building confusion matrix and calculating other performance metrics like accuracy, precision, recall and F1-score. A line chart was built to see the training and validation loss per epoch. Similar approach was followed for building ResNet-50 model on same set of images. The difference was that six new layers were added at the bottom of ResNet-50 and only these layers were trained for classification purpose. Both the classification models were finally run on Google CoLab. ResNet-50 though was directly run on CoLab because of large computation power required.

A little different pre-processing was done for images when dealing with the task of detecting infected blood cells. But before pre-processing images the training and testing images in this case were moved to two different folders respectively as part of a necessary requirement to implement Faster RCNN downloaded from GitHub. After separating training and testing images into two different folders it was noticed that the testing images looked a little different from the training images which could potentially be a drawback while detecting infected blood cells in the testing images. To tackle this, few testing images were moved from testing folder to training folder and vice-versa. The csv files were further modified because of this image transfer. Original training and testing files were used for object detecting purpose as they contained bounding boxes co-ordinates. The pre-

requisite before implementing Faster RCNN was to have these csv files in a certain format (filepath, xmin, ymin, xmax, ymax, class). The original files were manually converted to required format. In this case, the images were not rotated but only changed in size and a gaussian blur was applied for both train and test images. Later, one of the pre-requisites was to have text file in the above-mentioned format without headers in it which was built in Python on top of the csv file. The test csv was not used for this task because the model only had to be trained for detecting infected cells details of which were present in training csv. Finally, folders containing training and testing images and the text file were moved to Faster RCNN directory (one of the pre-requisites). Like ResNet-50, Faster RCNN was also directly run on CoLab using the free GPU service.

6 Evaluation and Results

Performance of all models was recorded after implementation for training, validation and testing.

6.1 Classification of Blood Smears

In this section, evaluation of both the models used for classification i.e. CNN and ResNet-50 will be discussed.

6.1.1 CNN

CNN model built from scratch for classification of blood smear images performed well during training and validation (at the time of running this code) as shown in Figure 12 and 13. It was noticed that the model gave good performance for 20 epochs.

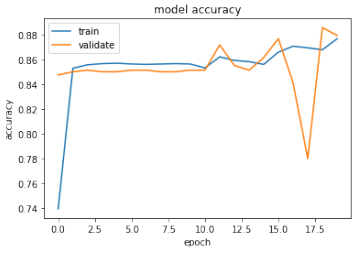

Figure 12: CNN Accuracy

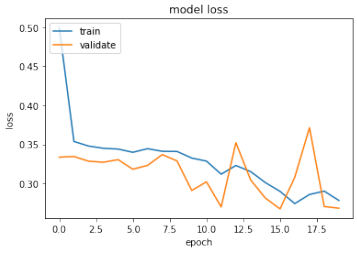

Figure 13: CNN Loss

As it is visible from the charts, the training accuracy increases steadily with increasing number of epochs and loss decreases. However, it was noticed that if the number of epochs is increased any further, accuracy decreases considerably, and the loss increases. So, for the classification task number of epochs is kept 20. It was also noticed that at 17th epoch, there is a sudden drop in accuracy and increase in loss but after it accuracy goes up and loss goes down somehow.

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.80	1.00	0.89
Accuracy = 0.80			

Table 2: Performance Metrics CNN

Although the performance appeared to be very promising in the accuracy and loss charts, but when test images were fed in the model the model didnt perform well. As visible in Table 2, model got an accuracy of 80% but precision, recall and F1-score for class 0 is 0 which means that the model inclined more towards classifying class 1 instead of class 0.

6.1.2 ResNet-50

After implementing a normal CNN model from scratch, ResNet-50 was then chosen with a hope to achieve promising results. Looking at the accuracy, loss charts for training and validation images we can see that for ResNet-50 these metrics are also good. Figure 14 and 15 show the accuracy and loss respectively. ResNet was also trained for 20 epochs.

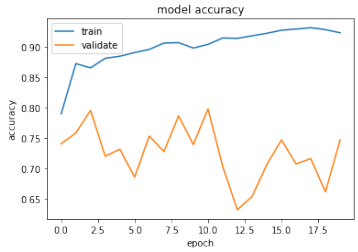

Figure 14: ResNet Accuracy

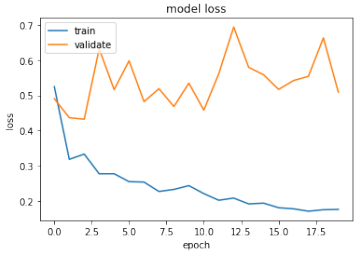

Figure 15: ResNet Loss

The training accuracy increases steadily and loss on the other hand decreases on training images. For validation images, a lot of ups and downs can be seen for accuracy and loss. Here also it was noticed that after epoch number 20 both training and validation accuracy went considerably down. So number of epochs for ResNet-50 was also kept to be 20.

Class	Precision	Recall	F1-Score
0	0.16	0.40	0.23
1	0.75	0.46	0.57
Accuracy = 0.45			

Table 3: Performance Metrics ResNet

Like CNN, ResNet-50 also witnessed the same behaviour. The accuracy for training and validation set of images appears to be good but when the test images were fed in the model it didnt perform so well. In Table 2 we can see that the model got an accuracy of 45% with precision of only 16% for class 0(uninfected).

6.2 Detection of Infected Blood Cells

This section discusses about the results obtained from the model which this research project uses for detecting infected blood cells in blood smear images. As the project uses Faster RCNN to achieve this, key metrics to check the performance of Faster RCNN are RPN Classifier Loss, RPN Regression Loss, Detector Classifier Loss, Detector Regression Loss and Total Loss which is nothing but the sum of the former four losses. Their charts are shown in Figure 16, 17, 18, 19 and 20.

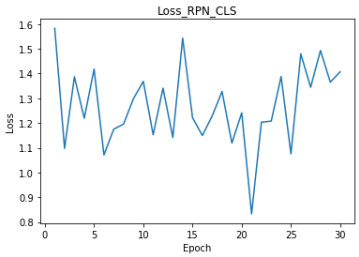

Figure 16: Epoch vs. Loss_RPN_CLS

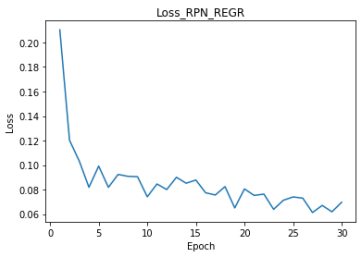

Figure 17: Epoch vs. Loss_RPN_REGR

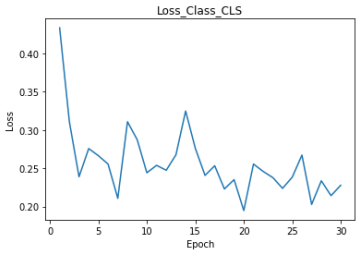

Figure 18: Epoch vs. Loss_Class_CLS

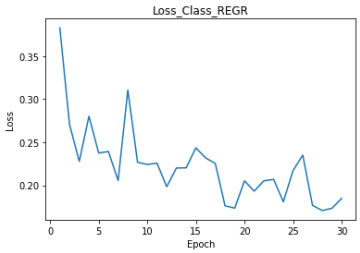

Figure 19: Epoch vs. Loss_Class_REGR

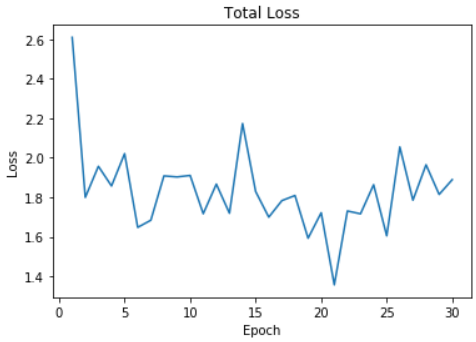

Figure 20: Epoch vs. Total Loss

From the above charts we can clearly see that except RPN regression loss, all other losses dont show a steady decrease, instead they show a fluctuating pattern. One interesting thing to notice in these charts is that all charts have gone down at 21st epoch.

7 Conclusion and Future Work

Malaria is a fatal disease which severely affects a lot of people in the African continent. As mentioned in literature review, it has recently claimed many lives in Burundi. Its quick treatment is very important. Many researches are still going on in this field regarding new drugs and vaccines. Apart from curing malaria quickly it is equally important that it should also be detected or diagnosed quickly. Diagnosing malaria is currently a time taking task because multiple samples are taken during a course of few days and then those samples are sent to laboratories for diagnosis. This research project tries to apply deep learning technologies for making the diagnosis process faster. Although this research project does not produce very promising results which can be seen in the evaluation section, idea behind the project was to use convolutional neural networks for diagnosing malaria through classification of blood smear images into infected and healthy and to detect infected blood cells in the blood smear images. This research project was a small step towards an era where diagnosis of malaria could be faster by using high end deep learning models integrated to an application. Future scope could be to use much higher number of images so that the models could be trained on a variety of images. Currently, there are not a lot of publicly available images present on the internet.

A CNN and ResNet-50 are used to classify blood smear images into infected or healthy and a separate model Faster RCNN is used to detect infected blood cells in the images with the help of bounding box co-ordinates. It is noticed that all the three models dont give promising results for now. Potential reasons behind this could be that the images may need a different kind of pre-processing which has not been used in this project for an example making the background of every image as light as possible like white and making the cells darker. Also, important features can be extracted from these images before feeding them into classification models. Surprisingly, Faster RCNN is also not giving very promising results. A lot of fluctuation can be seen in all the charts, where on the first hand all losses were supposed to go down. More investigation is needed for Faster RCNN to optimise its performance. Another option is to use other sophisticated object detection model like YOLO to detect infected cells in images.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zheng, X. (2016). TensorFlow: A system for large-scale machine learning, pp. 265–283. **URL:** *https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi*

Akiba, T., Suzuki, S. and Fukuda, K. (2017). Extremely large minibatch SGD: Training ResNet-50 on ImageNet in 15 minutes.
URL: *http://arxiv.org/abs/1711.04325*

Beaumont, P. (2019). Burundi malaria outbreak at epidemic levels as half of population infected.

URL: *https://www.theguardian.com/global-development/2019/aug/08/burundi-malaria-outbreak-at-epidemic-levels-as-half-of-population-infected*

Bhatt, S., Weiss, D. J., Cameron, E., Bisanzio, D., Mappin, B., Dalrymple, U., Battle, K. E., Moyes, C. L., Henry, A., Eckhoff, P. A., Wenger, E. A., Brit, O., Penny, M. A., Smith, T. A., Bennett, A., Yukich, J., Eisele, T. P., Griffin, J. T., Fergus, C. A., Lynch, M., Lindgren, F., Cohen, J. M., Murray, C. L. J., Smith, D. L., Hay, S. I., Cibulskis, R. E. and Gething, P. W. (2015). The effect of malaria control on *Plasmodium falciparum* in africa between 2000 and 2015, **526**(7572): 207–211.
URL: *https://www.nature.com/articles/nature15535*

Broad Bioimage Benchmark Collection (2019).
URL: *https://data.broadinstitute.org/bbbc/BBBC041/*

Cohen, W. W., Siegler, M. and Hofer, A. (2019). Neural query language: A knowledge base query language for tensorflow.
URL: *http://arxiv.org/abs/1905.06209*

Cowman, A. F., Healer, J., Marapana, D. and Marsh, K. (2016). Malaria: Biology and disease, **167**(3): 610–624.
URL: *http://www.sciencedirect.com/science/article/pii/S009286741631008X*

Draper, S. J., Sack, B. K., King, C. R., Nielsen, C. M., Rayner, J. C., Higgins, M. K., Long, C. A. and Seder, R. A. (2018). Malaria vaccines: Recent advances and new horizons, **24**(1): 43–56.
URL: *http://www.sciencedirect.com/science/article/pii/S1931312818303202*

Faure, S. (2018). Descriptive analytics, machine learning, and deep learning viewed via the lens of CRISP-DM.
URL: *https://www.kdnuggets.com/2018/05/descriptive-analytics-machine-learning-deep-learning-crisp-dm.html*

He, K., Zhang, X., Ren, S. and Sun, J. (2015). Deep residual learning for image recognition.
URL: *http://arxiv.org/abs/1512.03385*

Koning-Ward, T. F. d. (2018). Spotlight on proteins that aid malaria, **561**(7721): 41–43.
URL: *http://www.nature.com/articles/d41586-018-05977-2*

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks, *in* F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pp. 1097–1105.
URL: *http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf*

Leffler, E. M., Band, G., Busby, G. B. J., Kivinen, K., Le, Q. S., Clarke, G. M., Bojang, K. A., Conway, D. J., Jallow, M., Sisay-Joof, F., Bougouma, E. C., Mangano, V. D., Modiano, D., Sirima, S. B., Achidi, E., Apinjoh, T. O., Marsh, K., Ndila, C. M., Peshu, N., Williams, T. N., Drakeley, C., Manjurano, A., Reyburn, H., Riley, E., Kachala, D., Molyneux, M., Nyirongo, V., Taylor, T., Thornton, N., Tilley, L., Grimsley, S., Drury,

E., Stalker, J., Cornelius, V., Hubbart, C., Jeffreys, A. E., Rowlands, K., Rockett, K. A., Spencer, C. C. A., Kwiatkowski, D. P. and Network, M. G. E. (2017). Resistance to malaria through structural variation of red blood cell invasion receptors, **356**(6343): eaam6393.
URL: *https://science.sciencemag.org/content/356/6343/eaam6393*

Mace, K. E., Arguin, P. M. and Tan, K. R. (2018). Malaria surveillance united states, 2015, **67**(7): 1–28.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5933858/*

Michael Jones, P. V. (2001). Rapid object detection using a boosted cascade of simple features.
URL: *https://www.researchgate.net/profile/Michael*$_{J}$*ones*20*/publication/*3940582$_{R}$*apid*$_{O}$*bject*$_{D}$*etect*

Raghesh Krishnan, K., Midhila, M. and Sudhakar, R. (2018). Tensor flow based analysis and classification of liver disorders from ultrasonography images, *in* D. J. Hemanth and S. Smys (eds), *Computational Vision and Bio Inspired Computing*, Lecture Notes in Computational Vision and Biomechanics, Springer International Publishing, pp. 734–743.

Ren, S., He, K., Girshick, R. and Sun, J. (2015). Faster r-CNN: Towards real-time object detection with region proposal networks, *in* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (eds), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pp. 91–99.
URL: *http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf*

Rezende, E., Ruppert, G., Carvalho, T., Ramos, F. and Geus, P. d. (2017). Malicious software classification using transfer learning of ResNet-50 deep neural network, *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1011–1014.

Rodrguez-Martnez, M. and Garzn-Alfonso, C. C. (2018). Twitter health surveillance (THS) system, **2018**: 1647–1654.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6350799/*

Sharma, P. (2018). Implementing faster r-CNN in python for object detection.
URL: *https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/*

Sun, X., Wu, P. and Hoi, S. C. H. (2018). Face detection using deep learning: An improved faster RCNN approach, **299**: 42–50.
URL: *http://www.sciencedirect.com/science/article/pii/S0925231218303229*

Wagner, F. H., Sanchez, A., Tarabalka, Y., Lotte, R. G., Ferreira, M. P., Aidar, M. P. M., Gloor, E., Phillips, O. L. and Arago, L. E. O. C. (2019). Using the u-net convolutional network to map forest types and disturbance in the atlantic rainforest with very high resolution images, **0**(0).
URL: *https://zslpublications.onlinelibrary.wiley.com/doi/abs/10.1002/rse2.111*

WHO (2019).
URL: *https://www.who.int/news-room/fact-sheets/detail/malaria*

Wongsuphasawat, K., Smilkov, D., Wexler, J., Wilson, J., Man, D., Fritz, D., Krishnan, D., Vigas, F. B. and Wattenberg, M. (2018). Visualizing dataflow graphs of deep learning models in TensorFlow, **24**(1): 1–12.

Yamazaki, M., Kasagi, A., Tabuchi, A., Honda, T., Miwa, M., Fukumoto, N., Tabaru, T., Ike, A. and Nakashima, K. (2019). Yet another accelerated SGD: ResNet-50 training on ImageNet in 74.7 seconds.
URL: *http://arxiv.org/abs/1903.12650*

Yin, Y., Juan, C., Chakraborty, J. and McGuire, M. P. (2018). Classification of eye tracking data using a convolutional neural network, *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 530–535.