

Project Report
on
**Effective Convolutional Neural Network Model for Malaria
Classification**

Submitted in partial fulfilment of the requirements for the award of the degree

of

Bachelor of Technology

in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

Mr. Rajat Deoli

(170102035)

Mr. Arju Aman

(170102009)

Under the esteemed Supervision

of

Dr. Dilip Kumar Choubey



भारतीय सूचना प्रौद्योगिकी संस्थान भागलपुर
Indian Institute of Information Technology
Bhagalpur

Department of Electronics and Communication Engineering
Indian Institute of Information Technology Bhagalpur

June 2021

CERTIFICATE

This is to certify that the Final Project entitled “*Effective Malaria Detection Model using CNN*” presented by

Mr. Rajat Deoli

(170102035)

Mr. Arju Aman

(170102009)

B. Tech. students of IIIT Bhagalpur under my supervision and guidance. This project has been submitted in partial fulfillment for the award of “*Bachelor of Technology in Electronics and Communication Engineering*” degree at *Indian Institute of Information Technology, Bhagalpur*.

No part of this project has been submitted for the award of any previous degree to the best of my knowledge.

Dr. Dilip Kumar Choubey,
(Guide)

Dept. of Computer Science and Engineering

Examined and Approved by

Examiner I

Examiner II

Dr. Dheeraj Kumar Sinha
(Head)

Dept. of Electronics and Communication Engineering



Department of Electronics and Communication Engineering
Indian Institute of Information Technology Bhagalpur

Declaration

We hereby declare that the work reported in this project on the topic “*Effective Convolution Neural Network model for Malaria Classification*” is original and has been carried out by us independently in the **Department of Computer science Engineering, IIIT Bhagalpur** under the supervision of **Dr. Dilip Kumar Choubey**. We also declare that this work has not formed the basis for the award of any other Degree, Diploma, or similar title of any university or institution.

Mr. Rajat Deoli
170102035

Mr. Arju Aman
170102009

Acknowledgement

During the course of this Final Project Report preparation, we have received lot of support, encouragement, advice and assistance from many people and to this end we are deeply grateful to them all.

With great pleasure we express our cordial thanks and indebtedness to our admirable Guide, *Dr. Suraj*, Assistant Professor, Department of Electronics and Communication Engineering. His vast knowledge, expert supervision and enthusiasm continuously challenged and motivated us to achieve my goal. We will be eternally grateful to him for allowing us the opportunity to work on this project.

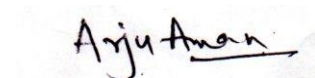
We have great pleasure in expressing my sincere gratitude and thanks to the *Prof. Arvind Choubey*, Director, Indian Institute of Information technology Bhagalpur and all the faculty members of department of Electronics and Communication Engineering, IIIT Bhagalpur. We express our sincere gratitude to *Dr. Dheeraj Kumar Sinha*, Assistant Professor and Head of Department for his valuable help providing me all the relevant facilities that have made the work completed in time.

The present work certainly would not have been possible without the help of our friends, and also the blessings of our parents.



(Rajat Deoli)

170102035



(Arju Aman)

170102009

June 2021, IIIT Bhagalpur

Abstract

Malaria is a serious and a life-threatening disease caused by plasmodium parasites that are transmitted to people through the bites of infected Anopheles mosquitoes.

The most common and standard method of diagnosing malaria is to have a qualified technician visually inspect the blood smear through a microscope for red blood cells infected with parasites.

Diagnosis by this relies on the experience and knowledge of the person conducting the inspection. Its time consuming too, and maybe prone to human error in rare cases.

Through modern deep learning tools, we can automate this process effectively. In this research, we propose a fully automated convolutional neural network (CNN) based model for diagnosing malaria in micrographs of blood smear specimens.

Our deep learning-based model can detect malarial parasites from microscopic images with an accuracy of about 96%, which can be further boosted when trained and tested on larger datasets.

Table of Contents

S.no	Content	Page no.
1	Introduction	7
1.1	Background	7
1.2	Objective	12
1.3	Problem Statement	13
1.4	Deep Learning	14
2	Literature Review	16
2.1	Background	16
2.2	Review of Works in Malaria Detection	16
2.3	Summary of the chapter	22
3	Analysis and Design	23
3.1	Introduction	23
3.2	Requirement Analysis	25
3.3	Design Summary	27
4	Proposed Methodology	28
4.1	Base	28
4.2	Algorithm Description	30
4.3	Implementation Details	38
5	Experimental Result and Analysis	57
5.1	Introduction and Terminologies	57
5.2	Analysis of Malaria Cell Image dataset	62
5.3	Experimental Results	64
6	Conclusion	68
7	Future Works	69
	References	70
	APPENDIX	76

TABLE OF FIGURES

No.	Figure Name	Page No.
1	Project on mobile	13
2	Waterfall Model	24
3	Proposed Model	27
4	Human Neural Network	31
5	Neural network in deep learning	33
6	Structure of Neuron	35
7	Backpropagation	37
8	Parasitized Cells	39
9	UnInfected Cells	40
10	Preprocessing code snippet	42
11	Model in code	45
12	Model representation	51
13	ReLU activation function	53
14	Sigmoid activation function	55
15	Confusion Matrix	64
16	Accuracy Chart	65
17	Model Accuracy	66
18	Loss Graph	67

CHAPTER 1

1. Introduction

1.1) Background

Malaria still poses a huge burden on the health of the world, with about 200 million people suffering worldwide and more than 400,000 deaths per year. It is one of the most common disease in the world, children, pregnant women, patients with other health condition like HIV/AIDS and travelers who have no prior exposure to malaria. In addition to biomedical research and political endeavors, modern information technology is also playing an important role in many attempts to fight this disease.

The World Health Organization (WHO) carries out a program to control malaria on a global scale, focusing on health care, early diagnosis of the disease, early treatment, and prevention of disease. The sub-tropical areas and tropical areas are the one that hit the hardest.

1.1.1) Types of Malaria:

There are 100+ plasmodium diseases but only 5 causes malaria in human:

- Plasmodium falciparum (or P. falciparum)
- Plasmodium malariae (or P. malariae)
- Plasmodium vivax (or P. vivax)
- Plasmodium ovale (or P. ovale)
- Plasmodium knowlesi (or P. knowlesi)

Out of these 5 only 2 are dangerous for human beings (P. falciparum. and P. vivax).

1.1.2) How transmission of parasite happens?

Only female Anopheles mosquitoes spread the malaria parasites. When a Plasmodium infected female mosquito bites a person who has malaria, it drinks the person's blood, which contains the Plasmodium parasites. When the mosquito bites another person, it injects the parasites into that person and make their way into the bloodstream and head to the liver. From the liver the infection spread to Red blood cells and that's when a human body start showing symptoms. That's how the disease spreads.

Once a plasmodium gets enter in human body it mainly effect and tries to destroy Red blood cells(RBC) and liver.

Each species of these mosquitoes has its own preferred habitat. Some species prefer small and shallow collections of fresh water.

Also, transmission is more in places where the mosquito lifespan is longer than expected as the parasite has time to complete its development inside the mosquito and there they prefer to bite humans rather than other animals.

It also depends on climatic conditions that may affect the number, life span and the survival of mosquitoes, such as rainfall patterns, temperature and humidity.

In many places, transmission depend on seasons. The peak is during and after the rainy season.

What are the symptoms?

In a human being these symptoms usually appear 10–15 days after the infective mosquito bite.

Symptoms can include,

- Heavy breathing
- Rapid heart rate
- High fever
- Shaking chills
- Sweating
- Headache
- Diarrhoea
- fatigue
- Body aches
- Yellow skin (jaundice)
- Kidney failure
- Seizure
- Bloody stools
- Convulsions

These symptoms are so similar to common cold or flu symptoms, so it might be hard to tell what an individual have at first. Also, they don't always show up within 2 weeks.

1.1.3) Prevention and treatment

If anyone lives in or is traveling to an area where malaria is very common among people, then one should take steps to avoid mosquito bites because Mosquitoes in that area are so prevalent that it's a guarantee that the person will end up getting malaria. Mosquitoes are most active between dusk and night.

To protect from mosquito bites, one should:

- Always cover your body parts. Wear pants and long-sleeved t-shirts and shirts. Try to cover most of your body parts with clothes.
- Apply insect repellent to body parts that are not cover. Use only those insect repellents registered with Environmental Protection Agency. Never use a spray directly on your face.
- Apply repellent to clothing. Sprays containing permethrin are safe and can be applied to clothing.
- Sleep under a mosquito net. In particular, those treated with insecticides help prevent mosquito bites while you are sleeping.
- Try to be in an air-conditioned room. If your room isn't air-conditioned, keep the fan always switched on. Mosquitoes are less active in circulating air.
- Antimalarial medicines can also be used to prevent malaria to some extent. No anti-malaria drug is 100% protective.

1.1.4) Diagnosis

It's important to get medical care as quickly as possible. Young children, infants, and pregnant women have an especially high chance for severe cases of malaria.

Early diagnosis and treatment of malaria reduces disease and prevents deaths. It also contributes to reducing malaria transmission. The best available treatment, particularly for *P. falciparum* malaria, is artemisinin-based combination therapy (ACT).

Seek care if you get a high fever while living in or traveling to an area that has a high chance for malaria. You should still get medical help even if you see the symptoms many weeks, months, or a year after your travel.

A blood test can show if you have malaria, but your doctor will also ask you about your medical history and any recent travel and do a physical exam.

The blood test can tell your doctor:

- If the parasite is in your blood
- If certain medications will work against the parasite
- If your body has ever made antibodies to fight off malaria

Definitive diagnosis of malaria generally requires direct observation of malaria parasites in Giemsa-stained thick and thin blood smears. Thick blood smears are more difficult to interpret than thin blood smears but they are much more sensitive, as more blood is examined. Thin blood smears, in which parasites are seen within erythrocytes, are used to determine the species of the infecting parasite. The presence of diagnostic forms can vary markedly with the stage of the life cycle, especially early in disease.

1.2) Objective

In the last 20 years, the role of technology in the healthcare sector has grown exponentially along with the technological advancement of our society. Our ability to store, share and analyze medical information is directly related to improved technology. The use of technology has improved the capabilities of healthcare providers and the opportunities for patients to see a doctor, while improving the quality of life for some patients and saving the lives of others. We are entering an era where doctors can see patients remotely via telemedicine and accurately diagnose their problems, even in most rural areas. We have evolved from using technology to improve patient care and the healthcare industries to impacting our society as a whole.

At the present time, everything is done by computer because it gives more accuracy and has very good computing power to perform any task in the blink of an eye. Providing best treatment best treatment to the patient at a reasonable cost is one of the biggest challenges faced by hospitals, for which computer-based information can be used by using modern advancements in cognitive computing, deep learning, machine learning and soft computing.

The basic idea behind this study is to motivate the detection of malaria parasite in an early stage by using CNN on Malaria Dataset provided on Kaggle to train the model. After training, the model can be taken as a package which can be used as a mobile phone app where on uploading the image, the user will get the result.

To implement the model on a smartphone, we need to convert to a TensorFlow Lite model, because this is the best solution to accurately run the trained model with limited memory and computing power available on phones.

1.3) Problem Statement

The most common method of diagnosing Malaria, which is one of the most serious and life-threatening disease in the world, is to have a qualified technician visually inspect the blood smear under a microscope.

Diagnosis by this method relies on experience and knowledge of person conducting the experiment. Through modern deep learning tools, we aim to automate this process as manual processes are time consuming and prone to human error.

Our model should be also leave scope for future training with more data and could be also used as an application on smartphones.

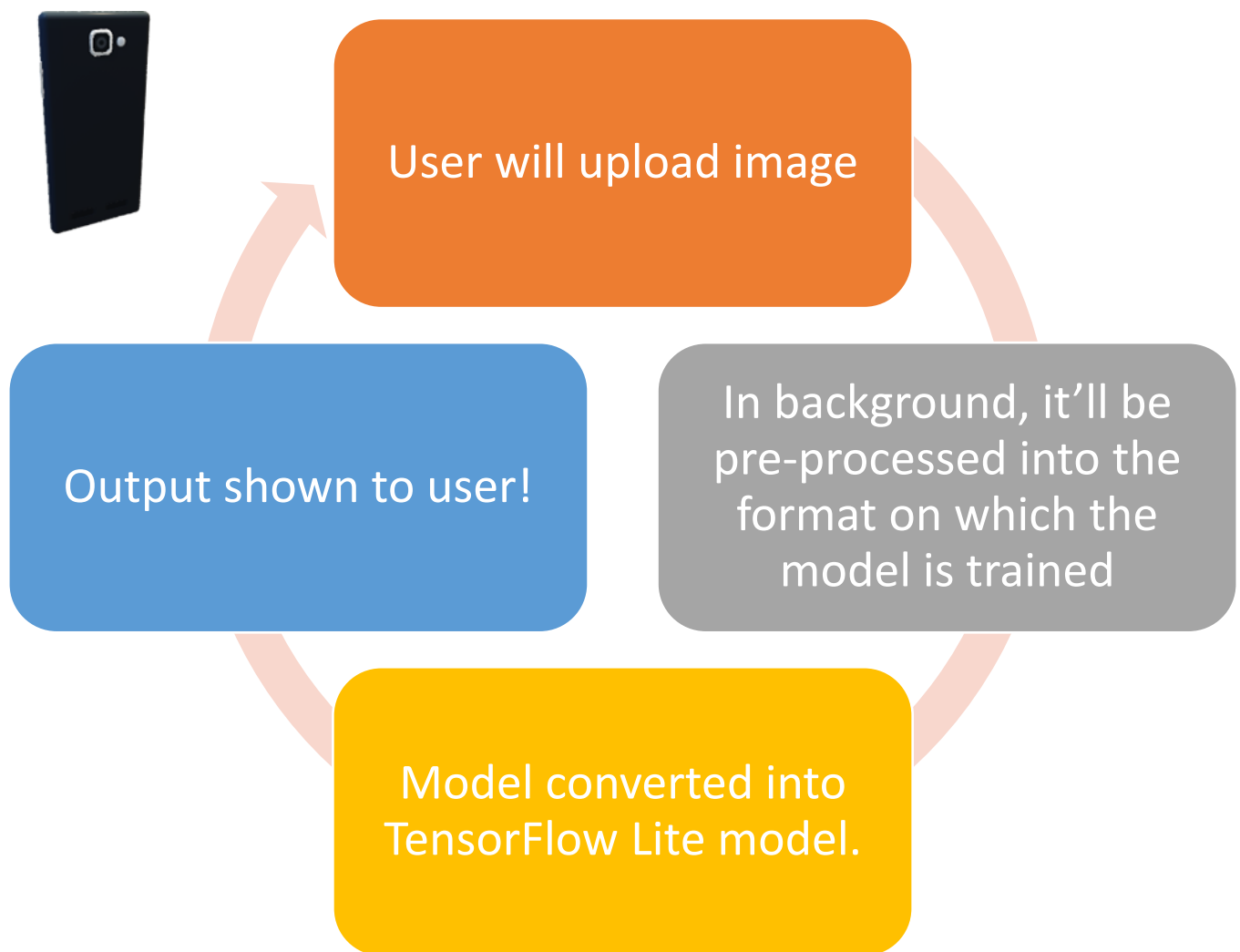


Figure 1: Project on mobile

1.4) Deep Learning

Deep learning (sometimes called deep structured learning) is a subset of machine learning, in which machines use artificial neural networks to process information. It's inspired by human biological nodes. Deep learning can help computers quickly recognize and process images and speech. Then the computer "learns" what these images or sounds represent, and builds a huge stored knowledge database for future tasks. In essence, deep learning allows computers to do what humans do naturally i.e learning through immersive learning and example.

As the father of Deep Learning, and now a researcher at Google, Geoffrey Hinton once said that "The point about this approach is that it scales beautifully. Basically, you just need to keep making it bigger and faster, and it will get better. There's no looking back now."

This sums up the essence of Deep Learning.

It's use case can be seen from a wide and diverse range of facilities and services, ranging from but not limited to:

- Machine Vision -> It may include image classification, Facebook tagging by face detection also uses this.
- Object Recognition -> Services providing sites like Clarifai.com uses object recognition
- Image Parsing -> Image Parsing is the breaking down of an image into its constituent visual patterns.
- Video Parsing -> Video parsing is the process of detecting scene changes or the boundaries between camera shots in a video stream.
- Fact Extraction -> Extracting meaningful data from a text excerpt, very useful for detecting fake news.
- Translation -> Can be seen in use by Google Translate
- Sentiment Analysis -> Sentiment analysis is the process of analyzing customer sentiment using natural language processing, text analysis and statistical data. The best companies understand the emotions of their customers: what people say, what they say, what they want to say. You can find customer sentiments in tweets, comments, reviews, or where other people mention your brand. Sentiment analysis is the field of using software to understand these emotions.

This is something that developers and business leaders must understand in the modern workplace. Like many other fields, advances in deep learning have brought sentiment analysis to the forefront of cutting-edge algorithms. Today, we use natural language processing, statistical data, and text analysis to extract and recognize word emotions in positive, negative, or neutral categories.

- Character-level text processing
- Cancer detection, Drug discovery, Radiology
- Finance, Digital advertising -> bidding for ad-space in webpage
- Customer intel -> upselling strategies
- Agriculture -> Identifying problematic environmental conditions

Summarizing, Deep learning is a form of Artificial Intelligence that uses a type of Machine Learning called Artificial Neural Networks with multiple hidden layers in attempt to learn hierarchical representations of underlying data in order to make predictions from given new data.

\

CHAPTER 2

2. Literature Review

2.1) Background

In this research, various research articles, including machine learning and deep learning, are reviewed to get a general idea, such as how to process the dataset, what algorithms should be used, and how to improve the precision from to form an effective system. The following is a review of some studies, including the techniques and methods used, as well as the conclusions.

2.2) Review of works in Malaria Detection

The performances of LeNet-5 [1], AlexNet [2] and GoogLeNet [3] were evaluated by Dong et al., which are the three most famous Convolutional Neural Networks [4]. They also trained an SVM classifier for comparison purposes and they arrived at the fact that CNN is advantageous over SVM in terms of the capacity of learning image features automatically.

Morphological factors were relied upon by Linder et al[5,6] for feature extraction, and then Principal Component Analysis (PCA) [7] and SVM were used for classification purpose. However, compared with the latest research techniques based on deep learning, the accuracy achieved by these types of models is lower.

Rajaraman et al.[8] used a previously trained CNN-based deep learning model to implement a feature extractor for sorting parasitic and uninfected blood cells to facilitate disease recognition. The research uses experimental methods to identify the best model layer using basic data. The CNN model has two fully connected dense layers and three convolutional layers. Measure the performance to extract characteristics from parasitic and uninfected blood cells with VGG-16, AlexNet, Xception, DenseNet-121 and ResNet-50.

Liang et al. [9] reported that with their 16-layer CNN model, its detection accuracy was 97.37%, and

claimed that their model is better than the transfer learning model.

Hung et al.'s model was previously trained in Imagenet [10], but adjusted based on its own data to detect malaria parasites [11].

Quinn et al. [12] showed off the 3D printable design of the adapter to connect the smartphone to the microscope, although all the images in the experiment were taken with a dedicated microscope camera, which provides a higher pixel resolution than their smartphone camera. They proposed a workflow for automatic analysis of thick blood smears, which involves the calculation of morphological and moment features and a tree classifier trained on these features to distinguish normal plaques and normal plaques containing parasites. The performance they report is 97% of the area under the receiver operating characteristic curve.

Rosado et al.[13] presented an image processing and analysis methodology using supervised classification to assess the presence of *P. falciparum* trophozoites and white blood cells in Giemsa-stained thick blood smears. Using a support vector machine and a mix of geometric, color, and texture features, their automatic detection of trophozoites achieved a sensitivity of 80.5% and a specificity of 93.8%, whereas their white blood cell detection achieved 98.2% sensitivity and 72.1% specificity.

Herrera et al.[14] used smartphone technology and image analysis software to test the diagnostic performance of the device for automatic RDT interpretation. The diagnostic performance of the device is comparable to the visual interpretation of PDR, and there is no significant difference between *Plasmodium falciparum* and *Plasmodium vivax*. In addition to nearly real-time case reports and quality control, the provision of standardized PDR automatic interpretation functions in remote areas will greatly facilitate the large-scale implementation of PDR-based malaria diagnosis programs.

Cesario et al. [15] spoke of mobile support for vector-borne diseases in areas lacking professional medical care. They focus on image analysis and classification components of the system, which aim to reduce the chance of misdiagnosing fewer common diseases such as malaria and help healthcare professionals. His article primarily describes the progress of the image classification and analysis components, but feedback from healthcare professionals is generally positive.

Dallet et al. [16] described a mobile application platform suitable for Android phones, which can diagnose malaria through images of thin films of blood stained with Giemsa. The main components of the images include detailed morphological operations that can detect red and white blood cells and identify parasites in infected cells. The application also recognizes the different life stages of parasites and calculates the level of parasitemia. The app takes less than 60 seconds to diagnose and

has been tested and verified on multiple versions and types of Android phones and tablets.

Skandarajah et al. [17] built a custom cell phone microscope compatible with cell phones from multiple manufacturers. They demonstrated that a quantitative microscope with micron-level spatial resolution can be performed on various mobile phones, and that image linearity, distortion and color can be corrected as needed. Specifically, they demonstrated that phones equipped with cameras with more than 5 megapixels can achieve resolutions close to the diffraction limit at a wide range of magnifications, including the magnifications associated with single-cell images. In addition, they found that the auto focus, exposure, and color gain standards on the mobile phone would lower the resolution of the image; if not corrected, they would reduce the accuracy of color capture, and they designed programs to avoid these obstacles that make it difficult to obtain quantitative images. .

Pirnstill and Cote[18] presented a profitable spherical polarization microscope system for obtaining image malaria pigments known as hevoids, which is a product of elimination of the digestion of the blood of the parasite. Even with a specialized microscope engineer, it can be difficult to determine the presence of background pigments and other artifacts. The pigment is much easier to observe using a polarizing microscope. However, although the implementation of polarized light microscopy is widely spread, the existing commercial equipment has complex designs, and sophisticated maintenance is required, so it tends to be expensive, and existing microscopic engineers require it for recession. The portable polarizing microscopy design presented by PiRSTILL and COTE indicates the possibility of resolution and specificity to detect malaria on low-cost and easy-to-use modular platforms.

Breslauer et al.[19] constructed an optical microscope mounted on a mobile phone, and proved its effect by imaging the red blood cells infected with *Plasmodium falciparum* and *Plasmodium falciparum* and sputum infected with *Mycobacterium tuberculosis* in the bright field under LED excitation fluorescence. Potential clinical use. In all cases, the resolution exceeds the resolution required to detect the morphology of blood cells and microorganisms. For tuberculosis samples, they used digital images and image analysis software to demonstrate automatic bacterial counts.

The proposed method by Shen et al.[20] and Mohanty et al.[21] considers unsupervised machine learning algorithms that apply stacked autoencoders to automatically learn the characteristics of infected and uninfected cell images.

Jane and Carpenter [22] proposed an object detection-based model using a convolutional neural network, named as Faster R-CNN. The model is first pretrained on ImageNet [23] and then fine-tuned on their dataset.

Bibin et al. [24] recommended another model using deep relative attributes (DRA) [25]

Razzak and Naz [26] have proposed an automated process that considers the tasks of both segmentation and classification of malaria parasites. Their segmentation network consists of a Deep Aware CNN [27], and the classification network employs an extreme learning machine- (ELM-) based approach [28].

Anggraini et al. [29] developed an application that uses image segmentation technology to separate the background from the blood cells..

MOMALA[30] is an application based on smartphones and microscopes designed to quickly detect malaria at low cost. The MOMALA app can detect the presence of malaria parasites on blood-stained slides. Connect the phone camera to the microscope eyepiece to take a picture of the blood smear and then analyze it. Today, this application relies heavily on microscopes that are heavy, heavy, and difficult to transport.

Researchers have developed a mobile app that can take photos of blood samples to detect malaria immediately.[31] Using a mobile app, we can analyze blood samples without involving microscope technicians. The application requires holding the smartphone on the eyepiece of the microscope, and the application analyzes the image of the blood sample and creates a red circle on the malaria parasite. Subsequently, a laboratory worker reviewed the case.

Most computer diagnostic tools that use machine learning models[32-34] for image analysis rely on manually designed decision-making functions. The process also requires computer vision expertise to analyze the variability of the image size, color, background, angle, and location of interest.

Deep learning techniques can be successfully applied to overcome the common challenges in the hand-designed feature extraction process [35]. The deep learning model combines a series of sequence layers with hidden non-linear processing units, which can reveal the hierarchical feature

relationships in the original image data. The (low-level) features abstracted from the high-level features contribute to non-linear decision-making functions and learning complexity, leading to feature extraction and classification from extreme to extreme [36]. In addition, compared to kernel-based algorithms such as support vector machines (SVM), deep learning models show better performance on large amounts of data and computing resources, making them highly scalable [37].

A protection mechanism is proposed by Zhang et al.[38] to use interactive robots for identity verification and access control, while controlling access to private data stored in the cloud. In later work [39], they introduced a new type of cognitive IoT paradigm using cognitive computing technology.

- Overview of most promising and famous works:

Method	Accuracy	F1 Score	Precision
Ross et al., (2006) [32]	0.730	-	-
Das et al., (2013) [33]	0.840	-	-
SS Devi et al. (2016) [40]	0.9632	0.8531	-
Zhaohui et al., (2016) [41]	0.9737	0.9736	0.9736
Bibin et al., (2017) [24]	0.963	-	-
Gopakumar et al.,(2018) [42]	0.977	-	-
Rajaraman et al., (2019) [8]	0.995	-	-
Proposed Model	0.9636	0.96	0.95

2.3) Summary of the chapter

In this chapter, several studies based on the technology used, tools used, and algorithms used are reviewed, and conclusions are drawn. These studies help the authors prepare for the situation to come.

CHAPTER 3

3. Analysis and Design

Before developing the system, a requirements analysis and design model is carried out to form the virtual framework of the system, and determines what the system needs.

3.1) Introduction

Artificial intelligence (AI) or machine learning (ML) has been developing rapidly. Today, we hardly see companies that do not use machine learning capabilities. Daily organization relies more on machine learning or artificial intelligence.

Recently, many products have been developed, such as self-driving cars, Google support, Netflix recommendation system and so on. AI or ML achieves this accuracy with the help of deep learning algorithms.

Deep learning is part of machine learning based on artificial neural networks. Computer vision is one of the tasks for AI and ML to achieve extremely high accuracy.

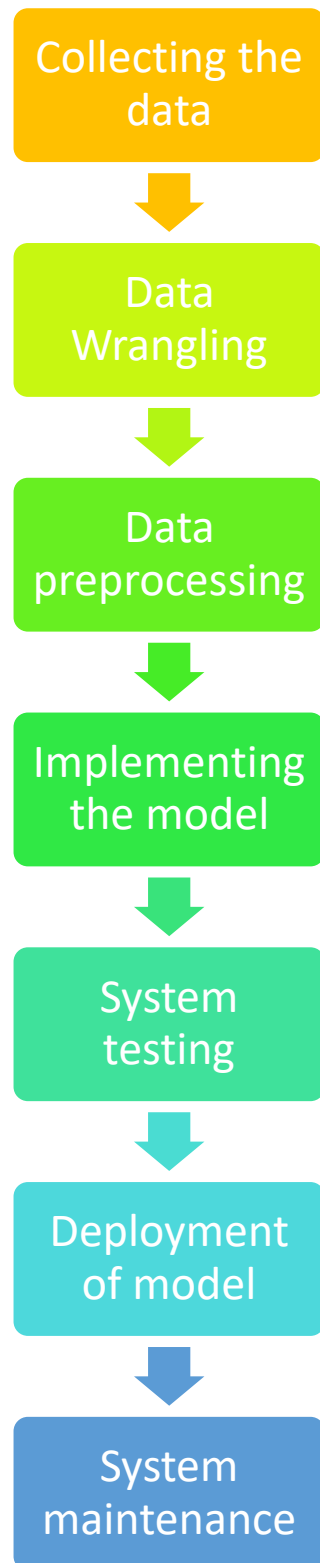


Figure 2: Waterfall Model

3.2) Requirement Analysis

In this section we describe the hardware and software components required for our project.

Software requirement:

- Python interpreter -> Python is a scripting language that has many built-in libraries and can execute thousands of complex machine learning algorithms, and uses libraries like numpy, sklearn, pandas, flask, joblib.
- Anaconda Navigator -> Anaconda Navigator is a desktop GUI that comes with Anaconda Individual Edition. It makes it easy to launch applications and manage packages and environments without using command-line commands.
- Python packages like pandas, numpy, matplotlib, sklearn, seaborn, etc
 - Numpy -> NumPy is a library of the Python programming language. It adds support for large multidimensional arrays and matrices, as well as a large collection of advanced mathematical functions for operations on these arrays.
 - Pandas -> Pandas is a software library for data processing and analysis written for the Python programming language.
 - Matplotlib -> Matplotlib is a plotting library for the Python programming language and its digital math extension NumPy.

- Modern text editors like Sublime Text
- TensorFlow Lite -> TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and IoT devices.

Hardware requirement:

- 16 GB + RAM
- 500 GB + SSD
- Processor speed 1.8 GHz or above
- External GPU

3.3) Design Summary

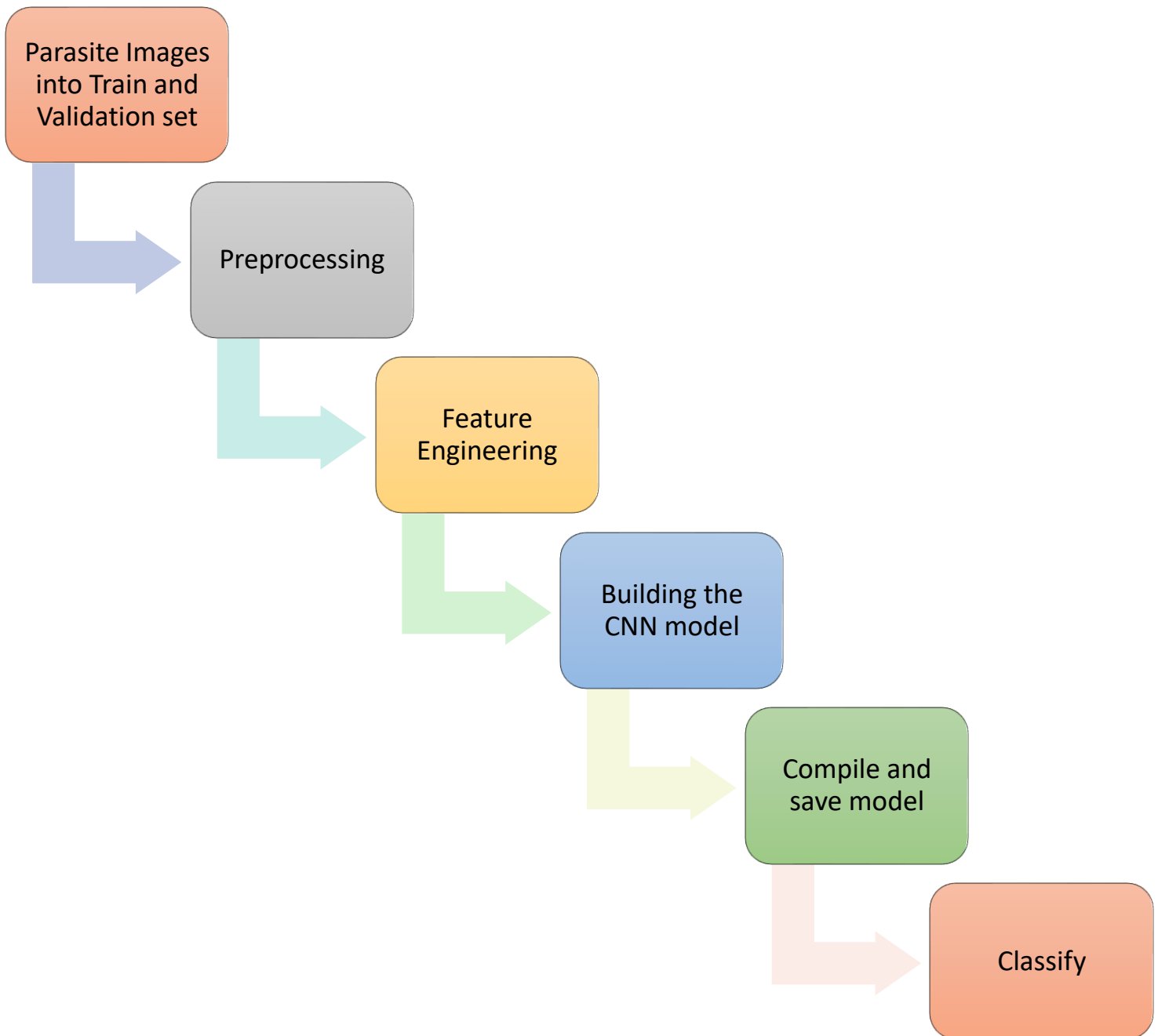


Figure 3: Proposed Model

CHAPTER 4

4. Proposed Methodology

4.1) Base

Computer Vision:

Computer vision is a field of study that focusing of developing technique to help computers see and understand the content of digital images such as photographs and videos. In the context of computer vision there are several methods in which machine are able to understand and sense their surroundings.

Object detection: With object detection the machine is able to sense and elements of image by extracting pixels and running them on a machine learning and deep learning algorithm.

3D scene reconstruction: Computer vision algorithms can reconstruct 3D objects from 2D imagery taken from multiple angles to create as realistic of an image a possible

Image and video pre-processing: Advance computer vision incorporate neural network that can perform images transformation which are not available for traditional image processing algorithms.

Scene segmentation: By making use of computer vision technology, pixel can be isolated and scanned obtaining an image that is very similar to stained glass. This technology will be extensively used in autonomous navigation and radiology to check cancer tissue, X-ray, Malaria cells etc.

Video and Image context indexing: With computer vision can also incorporate machine learning models that are trained to detect objects in photos.

Computer vision really helps to make life easier and is a major technology that can transform the future of automation. This domain will open new areas of development and help to create new industry. This field is already advancing at rapid speed, some of its current application retails shelf analysis, RTG analysis, automatic video tagging, real estate evaluation, security system, ID verification etc.

4.2) Algorithm Description

Neural Networks:

Human neural network: Neurons are cells in the brain. Our brain is packed with full of neurons like these. The neuron has a cell body, like so, and moreover, the neuron has a number of input wires, and these are called the dendrites and a neuron also has an output wire called an Axon. This output wire is what it uses to send signals to other neurons, so to send messages to other neurons.

So, at a simplistic level what a neuron is, is a computational unit that gets a number of inputs through its input wires and does some computation and then it sends outputs via its axon to other nodes or to other neurons in the brain.

The way that neurons communicate with each other is with little pulses of electricity.

A sample representation of Human NN [43]:

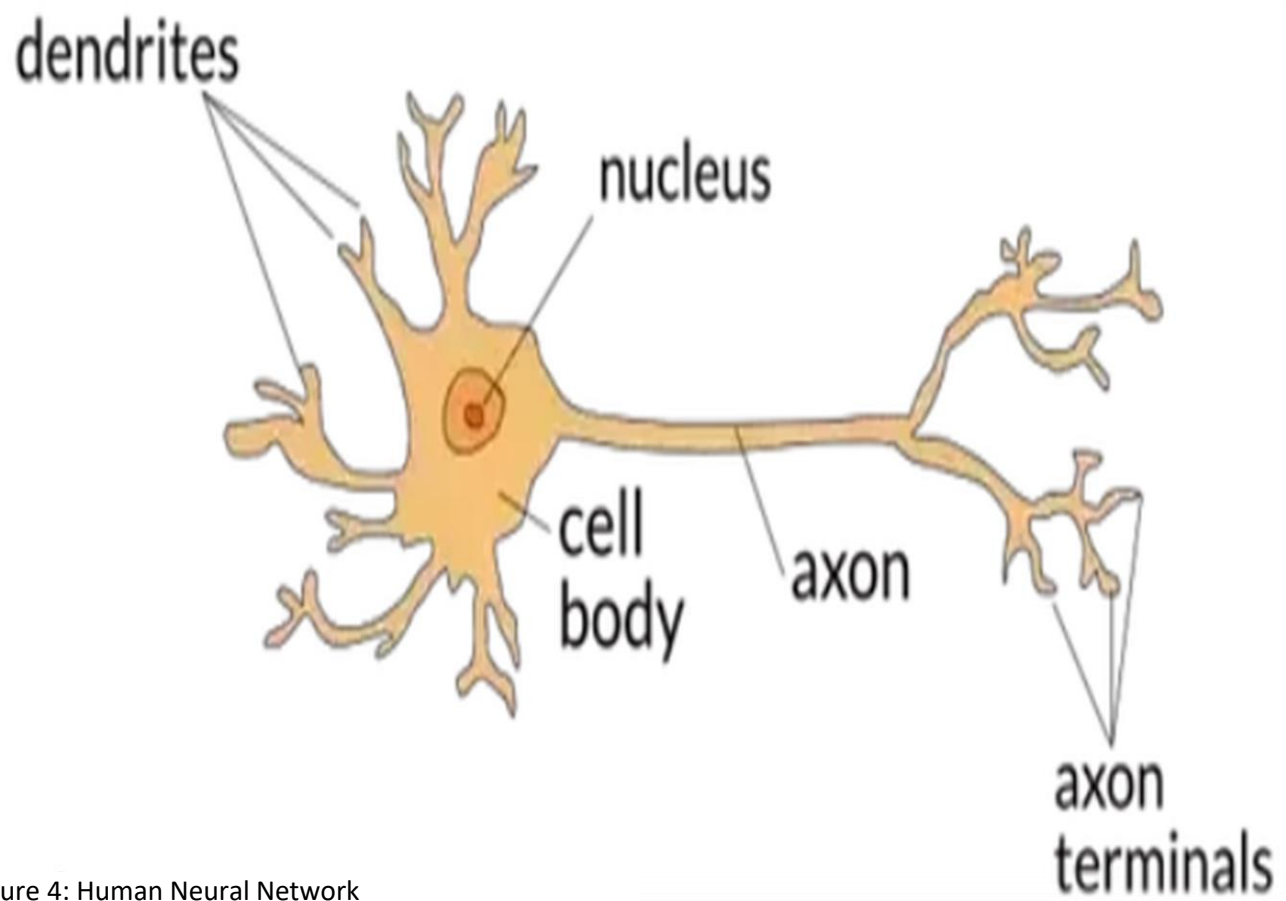


Figure 4: Human Neural Network

Artificial Neural Networks:

Neural network forms a base of deep learning which is a sub field of deep learning where algorithms are inspired by human brain.

Neural network takes in data and train themselves to recognize pattern in the data and predicts the output for a new set of similar data

Neural network is made of layers of neurons. These neurons are core processing unit of the network. First, we have an input layer which receive the input, and the output layers predicts the final output. In between input and output layers exist the hidden layers which perform most of the computations required by our network.

A sample representation [44]:

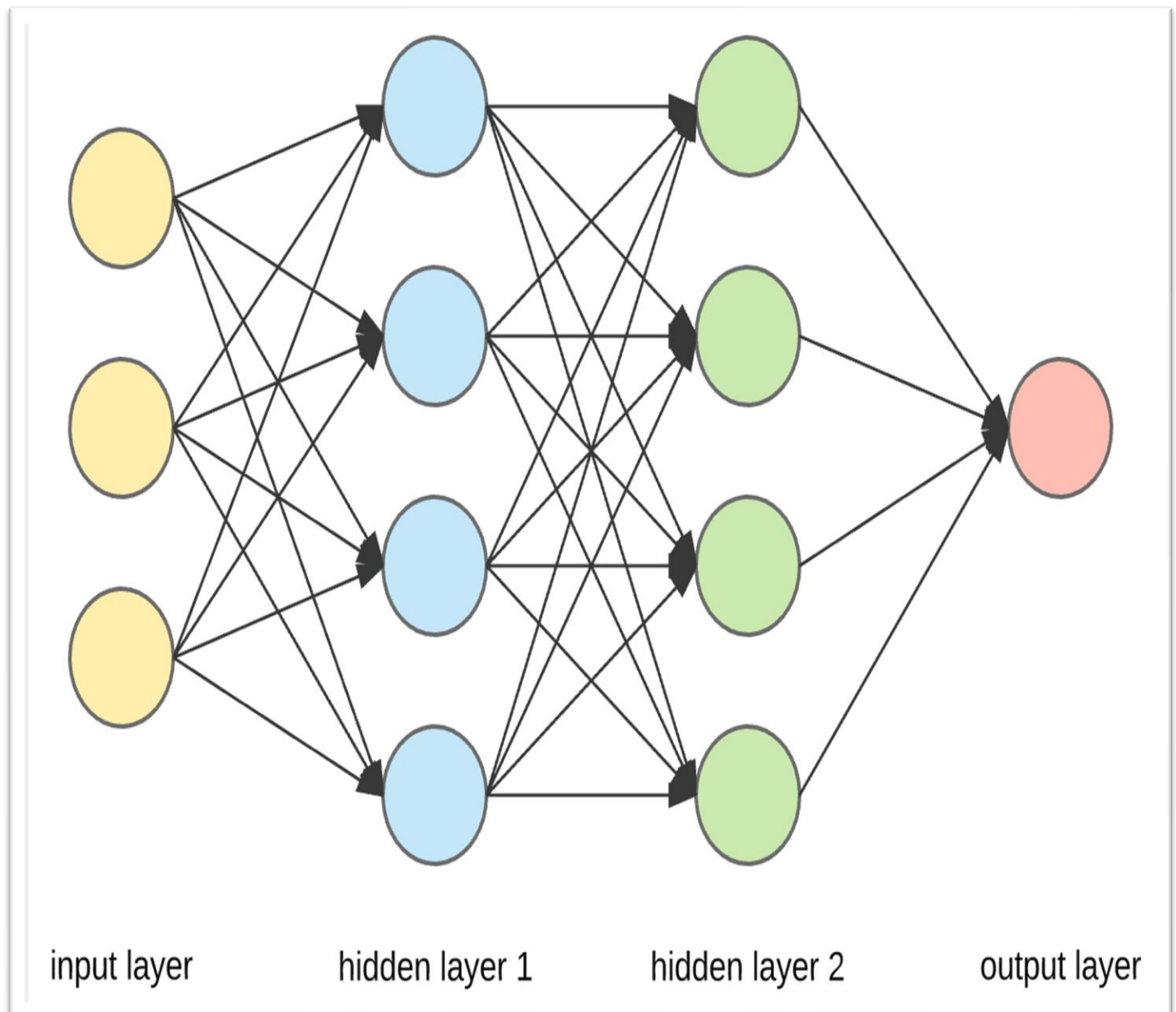


Figure 5: Neural network in deep learning

S

Neurons of one layer is connected to the neuron of another layers through channels. Each of these channels are assigned a numerical value known as weight. The inputs are multiplied with corresponding weight and their sum is sent as input to the neurons in the hidden layers. Each of these neurons is associated with a numerical value called the bias which is then added to the input sum.

Then this value is passed through a threshold function called activation function. The result of the activation function determines if the particular neuron will get activated or not.

Activated neuron transmit data to the neurons of the next layers over the channels. In this manner the data is propagated through the network. This is called **forward propagation**.

In the output layers the neuron with the highest value determines the output. The predicted output is compared to the actual output provided in our data set. If they are incorrect, the network uses a technique called **backpropagation** to change its learning parameters so that it can make right prediction and to get more accuracy.

Structure of Neuron:

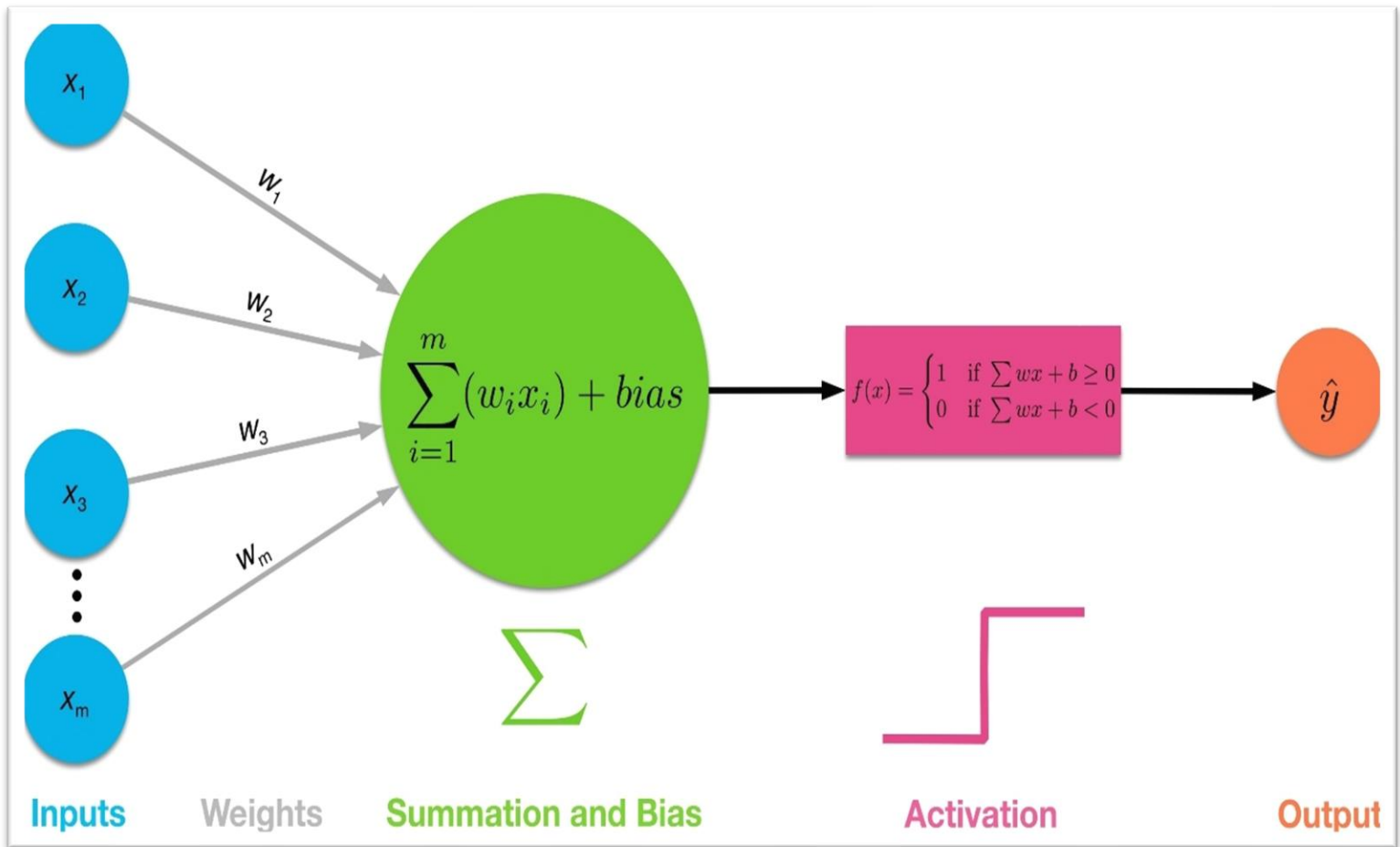


Figure 6: Structure of Neuron

This is a simple structure of neuron. When the input is feed into the neuron it get multiplies with “weight” ($w_1, w_2, w_3, \dots, w_m$). These weights are the only value that will change during the learning process (using backpropagation).

Bias (b) may get added to the total value calculate by the neuron. It is chosen before the learning process and it can be useful for the network.

All these summations then pass through an “**activation function**” to obtain the value.

Backpropagation:

In a neural network training play a very important role. By training we meant to update the weights occur in neural network. These act of actually calculating the gradients in order update weights occurs through a process called backpropagation.

In forward propagation once we reach the output layers, we obtain the resulting output from the model for the given input. Given the output result we then calculate the **loss** on the result. The way the loss is calculated is going to depend on the particular loss function we are using (In our model we used **Binary cross entropy loss function**).

We know that the gradient objective is to minimize this loss function. This is done by taking the derivative of loss function w.r.t to weights in the models. This is where backpropagation comes in picture. Backpropagation is the tool that gradient descent uses to calculate the gradient of the loss function.

In other words, we can say that we have the output generated from the input, the loss then calculated for the that output and then gradient descent starts updating our weights using backpropagation in order to minimize the loss.

We can also say that in backpropagation we are actually working backwards. From the output we are coming back to the provided input.

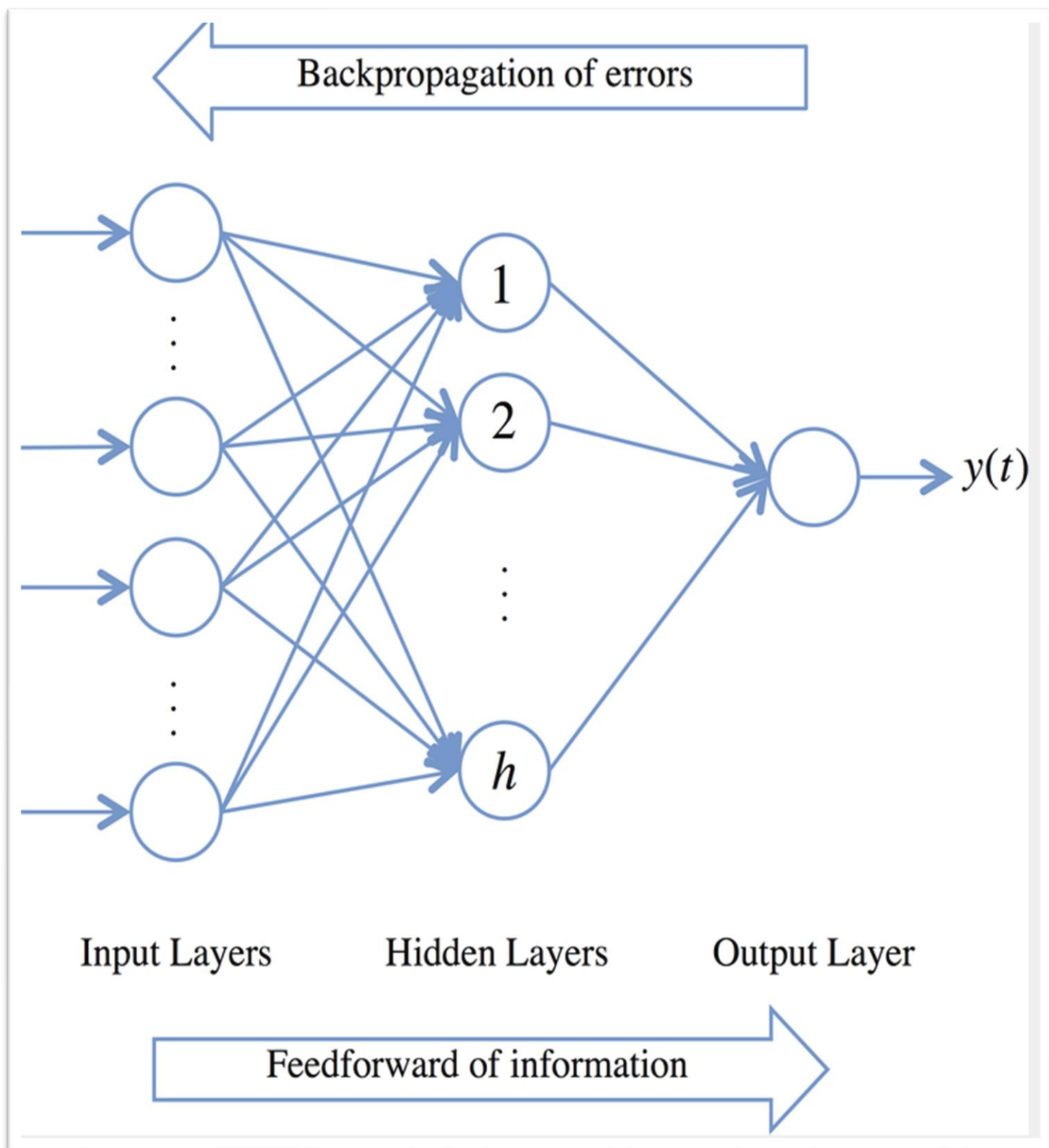


Figure 7: Backpropagation

4.3) Implementation Details

Introduction and Insights of data:

Cell classification is a challenging task for computer vision. In recent years there have been several attempts to build a classifier that will classify these cellular images, using label-free cellular images obtained from a microscope. These are then labelled by humans and feed into a classifier. Yet, classifying different types of cell with high precision has remained a challenging task, given difference in shape and size, and also some impacts caused by external environment.

Here we take this data set from NIH (national institute of health) that has cell images of malaria. [46]

This page hosts a repository of segmented cells from the thin blood smear slide images from the Malaria Screener research activity. To reduce the burden for macroscopics in resource-constrained regions and improve diagnostic accuracy, researchers at the Lister Hill National Centre for Biomedical Communications (LHNCBC), part of National Library of Medicine (NLM), have developed a mobile application that runs on a standard Android smartphone attached to a conventional light microscope.

The dataset contains a total of 27,558 cell images with equal instances of parasitized and uninfected cells.

Parasitized sample images:

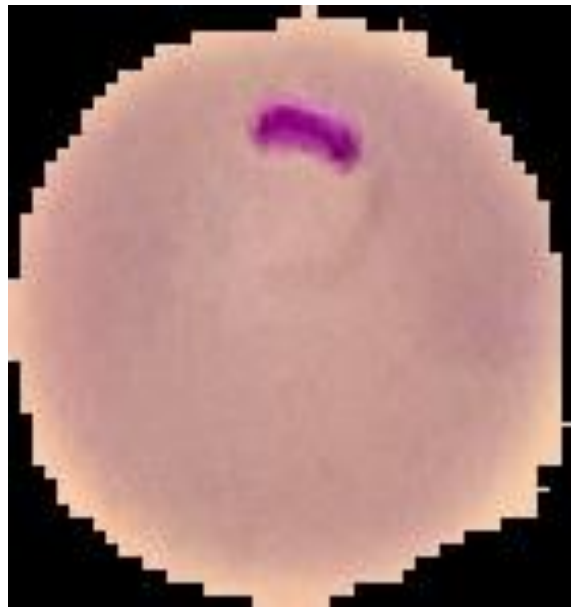
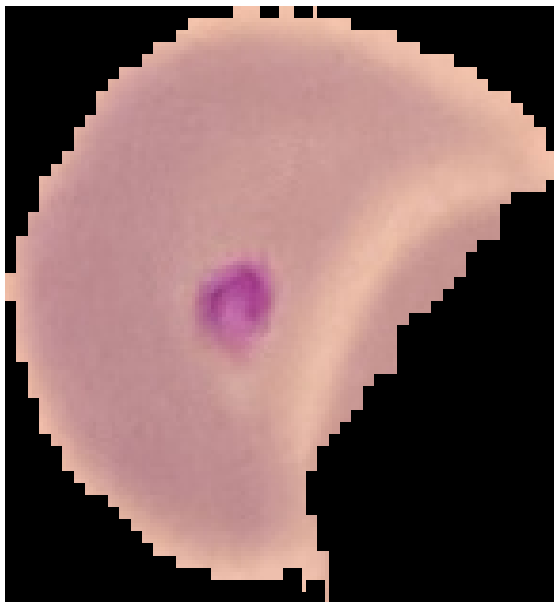


Figure 8: Parasitized Cells

Uninfected sample images:

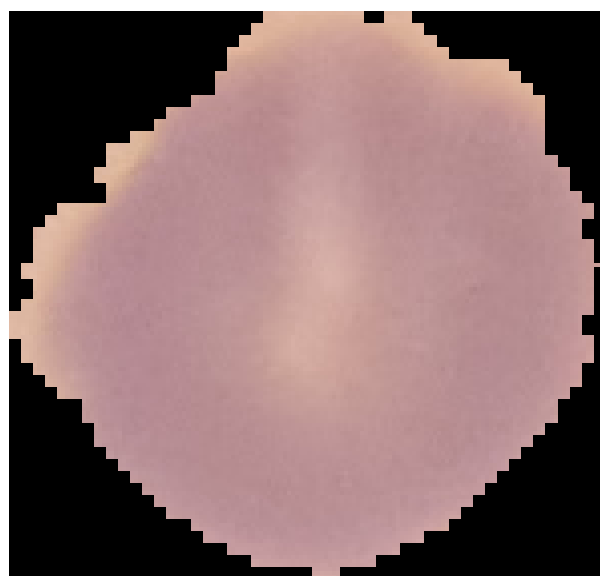
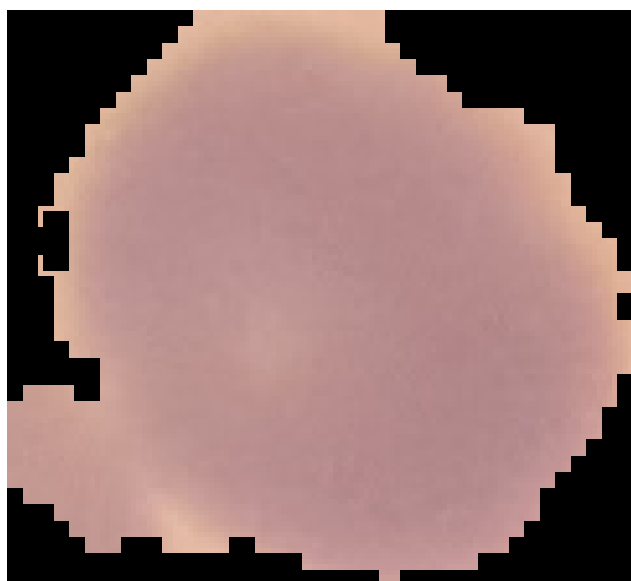
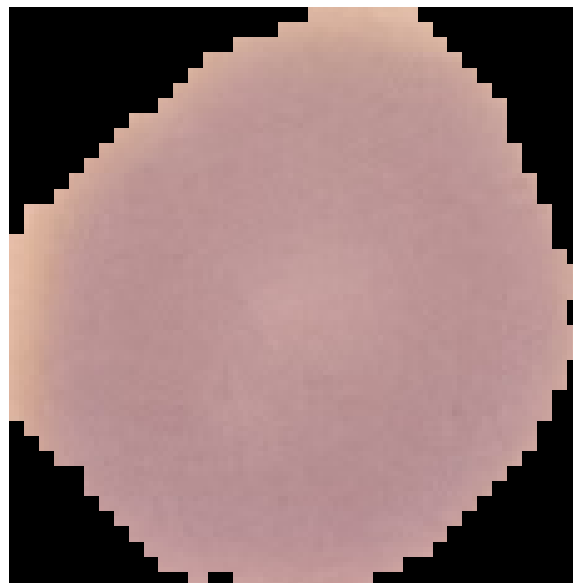
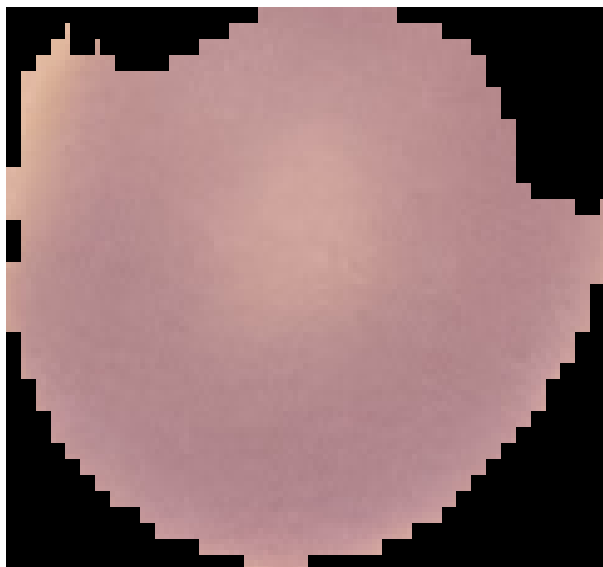


Figure 9: Uninfected Cells

Parasite images train and validation split:

The dataset set contain all the images in a directory cell images. These images are categories in 2 parts Parasitized and Uninfected. In this directory there are two more directories Parasitized and Uninfected. Both directories contain 13,780 images according to their category.

We have taken 11,024 of each category to train our model (22048 training images) and 2,255 of each category for validation (5510 validating images).

That means we split our data set in 80% training and 20% validation.

Data pre-processing:

Data augmentation: Data augmentation is technique to increase the size of data set by artificially creating new training data from the same data. It is believed that data augmentation can improve accuracy and help to reduce overfitting. Also, it has been seen that data augmentation is crucial for Convolutional neural network (CNN) model. So, we applied similar approach and our study by performing different rotations, cropping, skewing, and flipping on each image in the training set, resulting in a significant increase in the number of the images. More training data result in better model.

We selected the image size to be 64 x 64.

For data augmentation we use a function called `ImageDataGenerator`.

We use following command to import that function:

```
"from tensorflow.keras.preprocessing.image import ImageDataGenerator"
```

```
In [6]: img_shape = (64, 64, 3)
        image_gen = ImageDataGenerator(rotation_range=45,
                                         width_shift_range=0.1,
                                         height_shift_range=0.1,
                                         rescale=1/225,
                                         shear_range=0.2,
                                         zoom_range=0.1,
                                         vertical_flip=True,
                                         horizontal_flip=True,
                                         fill_mode='nearest',
                                         validation_split=0.2)
```

Figure 10: Preprocessing code snippet

Convolutional neural network (CNN):

Convolution neural network: Convolutional neural network also known as CNN or comp net is an artificial neural network that is so far been most popularly used for analysing images. Although image analysis has been the most widespread use of CCN's. They can also be used for other data analysis or classification problem as well.

We can think of a CNN as artificial neural network that has some types of specialization for being able to pick out and detect patterns. This pattern detection is what makes CNN so useful for image analysis.

In CNN hidden layers are called convolutional layers, and these layers are precisely what differentiate a simple neural network and CNN.

These convolutional layers are able to detect patterns. With each convolutional layer we need to specify the numbers of **filters** the layers should have.

These filters help in finding the pattern in the image. They can detect multiple edges, shapes texture, objects. Some may detect circles, squares and others. The deeper our network goes the more sophisticated these filters become and in deep layers rather than edges and simple shape our filter will be able to detect objects like eyes, ears, hair etc. With more deeper layers the filters may able to detect cats, dogs, human.

A filter is technically a small matrix for which we decide the numbers of rows and columns and the value in the matrix are initialize with random numbers. In our model we decide the size for filter is 3 x 3.

Now, when this convolution layer receives input the filter will slide over each 3 x 3 set of pixels from the input until slid over every 3 x 3 block of pixel from the entire image. This sliding is known as convolving

Also, in our model we use **Relu** (Rectified Linear Unit) as an activation function and **padding** in our model. Our model consists of 4 convolution layers followed by **batch normalization** layer and **max-pooling** layer. After that we flatten our output with the help of flatten layer and also, we use dropout layer in between.

The final output layer shows our result where we used **Sigmoid** as an activation function.

```

In [19]: model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', padding='same', input_shape=(64, 64, 3)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(32, (3,3), activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

```

Figure 11: Model in code

Model summary

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
===		
conv2d (Conv2D)	(None, 64, 64, 16)	448
<hr/>		
batch_normalization (Batch Normalization)	(None, 64, 64, 16)	64
<hr/>		
max_pooling2d (MaxPooling2D)	(None, 32, 32, 16)	0
<hr/>		
conv2d_1 (Conv2D)	(None, 32, 32, 32)	4640
<hr/>		
batch_normalization_1 (Batch Normalization)	(None, 32, 32, 32)	128

max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32) 0

conv2d_2 (Conv2D) (None, 16, 16, 64) 18496

batch_normalization_2 (Batch Normalization (None, 16, 16, 64)) 256

max_pooling2d_2 (MaxPooling2 (None, 8, 8, 64)) 0

conv2d_3 (Conv2D) (None, 8, 8, 128) 73856

batch_normalization_3 (Batch Normalization (None, 8, 8, 128)) 512

max_pooling2d_3 (MaxPooling2 (None, 4, 4, 128)) 0

flatten (Flatten) (None, 2048) 0

dropout (Dropout)	(None, 2048)	0
--------------------------	---------------------	----------

dense (Dense)	(None, 1024)	2098176
----------------------	---------------------	----------------

batch_normalization_4 (Batch Normalization)	(None, 1024)	4096
--	---------------------	-------------

dropout_1 (Dropout)	(None, 1024)	0
----------------------------	---------------------	----------

dense_1 (Dense)	(None, 512)	524800
------------------------	--------------------	---------------

batch_normalization_5 (Batch Normalization)	(None, 512)	2048
--	--------------------	-------------

dropout_2 (Dropout)	(None, 512)	0
----------------------------	--------------------	----------

dense_2 (Dense)	(None, 1)	513
------------------------	------------------	------------

=====
===

Total params: 2,728,033

Trainable params: 2,724,481

Non-trainable params: 3,552

Model representation:

Here we try to use a flow chart to show how our models work and how layers are connected to each other.



Figure 12: Model representation

Batch normalization:

In batch normalization we normalize the activation in layers. In this way we can increase the speed of model training, it decreases the importance of weights initialization and it regularizes the model.

Activation function ReLu (Rectified Linear Unit):

ReLu is a non-linear activation function. It is a piecewise linear function that will give output only if the input provided is positive and it will give zero if the input is negative or zero. Those models who use ReLu as an activation function are easier to train.

It is defined by the equation:

$$Y = \max(0, X)$$

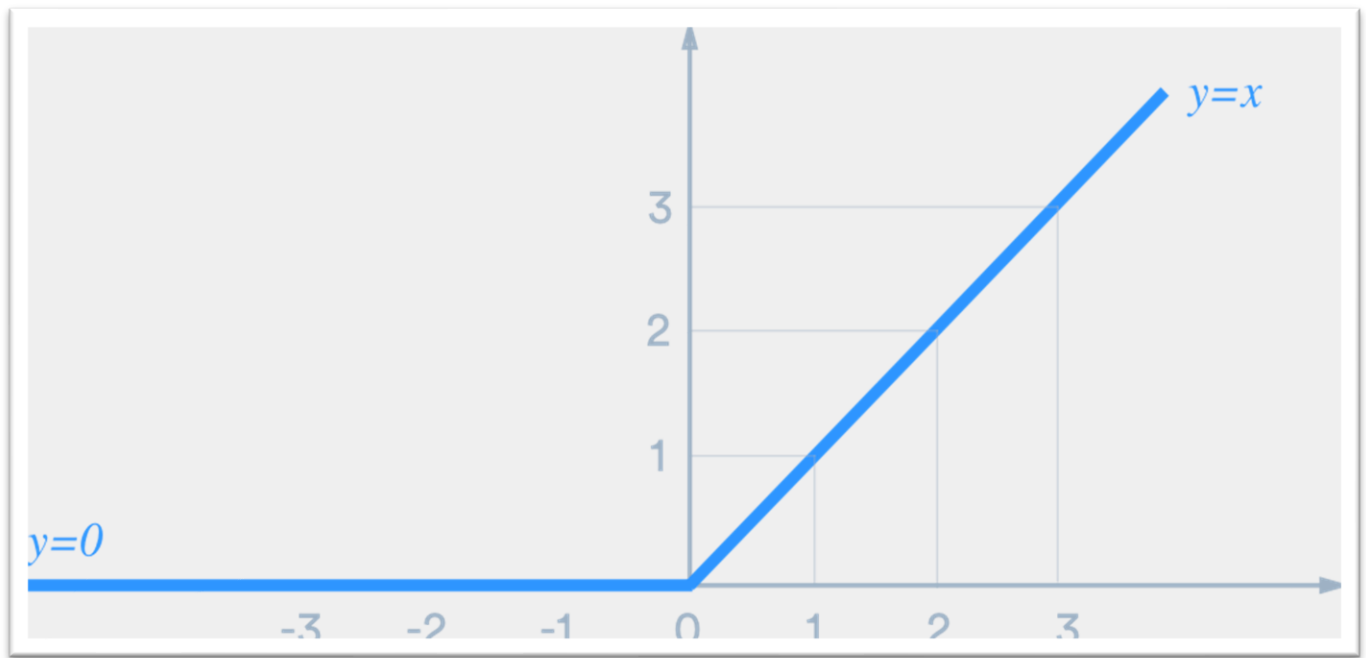


Figure 13: ReLU activation function

Padding:

The image shrinks every time a convolution operation is performed and the pixels at the corners and the edges are used much less than those in the middle.

Padding is a process to add layers of zeros to the output images after a convolution layer so that pixel at the corner are used properly.

Max pooling layers:

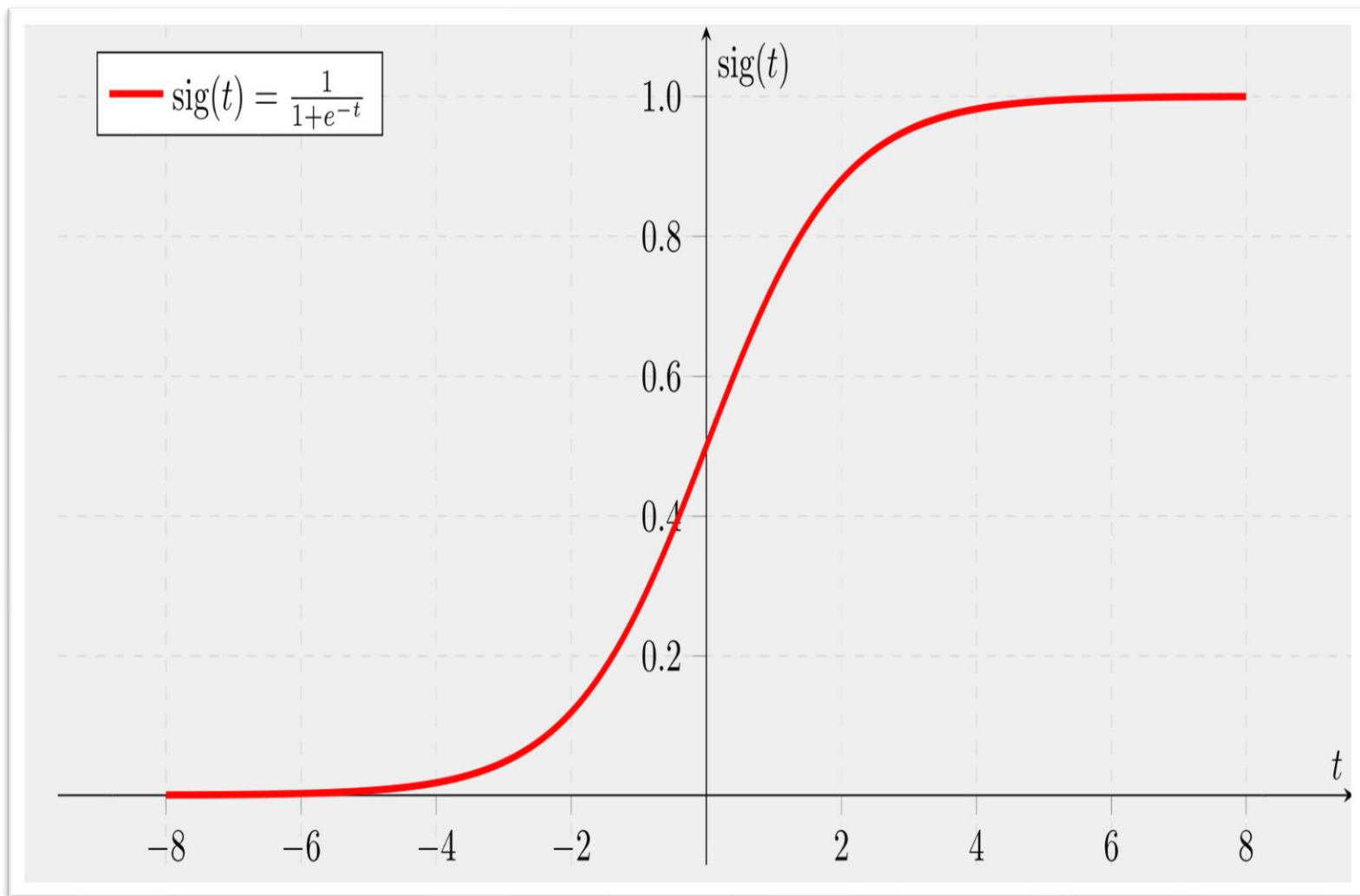
Max pooling is a type of operation that's typically added to the CNN's following individual convolutional layers. Max pooling reduces the dimensionality of images by reducing the numbers of pixels in the output from the previous convolutional layers. It generally used and more effective when the images have black background and our image has more features in the brighter pixels.

In max pooling we define some $N \times N$ region, as a corresponding filter for the *max* pooling operation. In our model we have taken 2×2 . So we will take max value of 2×2 matrix and slide over by 2 again. We do that until we reach edge on the far right. We then move down by the 2 because that's our stride size and then we do the same exact thing of calculating the max value for all 2×2 blocks in this room. We can think of these 2×2 block as pools of numbers and we are taking the max of these pools, so we can see where the name max pooling came from.

So, in this way we can see the dimension of our matrix is reduce by 2(Follow the model summary).

Activation function Sigmoid:

A sigmoid function plays an important role in the context of logistic regression. Here the sigmoid function plays the role of activation function. It takes the weighted sum of the input feature as an input and outputs the probability value of the outcome.



Sample figure [45] :

Figure 14: Sigmoid activation function

Model compile:

The compilation is the final step in building a model. Once we compile our model, we move to training phase. Also, the loss function, the optimizer and the metrics are decided on this phase.

Optimizer (Adam):

The objective of all the optimizers is to reach the global minima where, the cost function attains the minimum possible value. Adam optimizer is basically taking gradient descent with momentum and RMSProp and putting them together.

CHAPTER 5

5. Experimental Result and Analysis

5.1) Introduction and Terminologies

We ran our code on in-built Jupyter Notebook provided by Kaggle, as well as free GPU and TPU provided by them.

From our algorithms, we managed to conclude the outcomes like:

True Positive (TP):

True Positive is the case when the output predicted by our algorithm as well as the actual result for a particular training data are both TRUE.

True Negative (TN):

True Negative is the case when the output predicted by our algorithm is FALSE and the actual result for a particular training data is FALSE too.

False Positive (FP):

False Positive is the case when the output predicted by our algorithm is TRUE but the actual result is FALSE.

False Negative (FN):

False Negative is the case when the output predicted by our algorithm is FALSE but the actual result is TRUE.

Accuracy, Sensitivity/ Recall, Specificity, Precision and F1-Score can be calculated using these four outcomes.

Accuracy:

Accuracy is the ratio of the true values in dataset to overall values in dataset:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Sensitivity (Recall):

Sensitivity is given by the ratio of values which are true and predicted true too, to the values which are true and predicted true along with other values which are false and predicted false:

$$Sensitivity = \frac{TP}{TP + FN}$$

Specificity:

Specificity is given by the ratio of true negatives to sum of true negatives and false positives:

$$Specificity = \frac{TN}{TN + FP}$$

Precision:

Precision is an important metric which is given by the ratio of true positives to sum of true positives and false positives:

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

F1-score:

F1-score is the harmonic mean of precision and recall.

F1-score is given by the ratio of product of twice the precision and recall to the sum of precision and recall, which can be further simplified as:

$$F1 - score = \frac{2 * TP}{2 * TP + FP + FN}$$

Importance of these metrics:

It is intuitive to use accuracy as the defining indicator of the model, but in most cases, it is always recommended to use both Precision and Recall. In other cases, our accuracy may be high, but the accuracy or recall rate is low. Ideally, for our model, we want to completely avoid any situation where the patient has heart disease, but our model classifies it as having no heart disease, that is, for high memory.

On the other hand, for patients without heart disease and our model predicts other conditions, we also want to avoid treating patients without heart disease (this is important in cases where the input parameters may indicate other diseases, but eventually he was treated for his malaria).

Although our goal is high precision and high recovery rate, it is impossible to achieve both at the same time. For example, if we change the model to a model with a higher recall rate, we can detect all patients who actually have heart disease, but may eventually treat many patients without heart disease.

Similarly, if we seek high precision to avoid errors and unnecessary treatments, many patients who actually suffer from malaria will not be treated.

In many cases, accuracy is as important as precision. For example, for our model, if the clinician tells us that patients who are misclassified as malaria are equally important because they may indicate other diseases, then our goal is not only a high recall rate, but also a high recall rate. The accuracy is the same. In this case, F1-score bails us out!

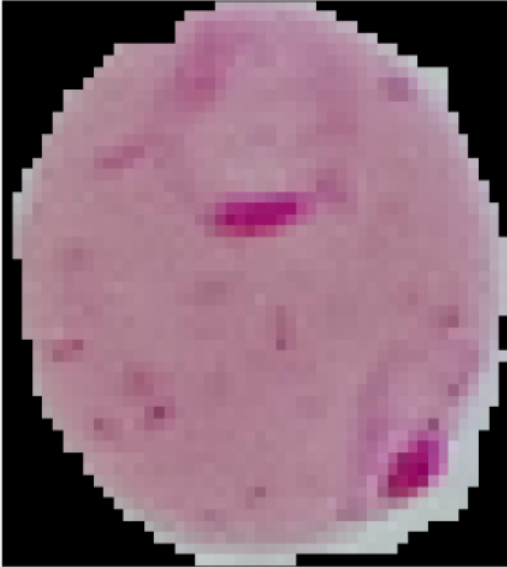
5.2) Analysis of Malaria Cell Image dataset:

Dataset description:

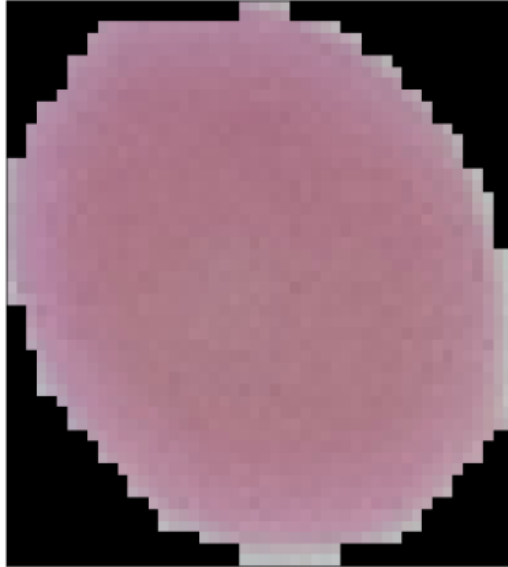
Kaggle's Malaria Cell Image Dataset [46] is used in this study. The dataset consists of total 27560 images categorized into two parts of 13780 each, i.e. Parasitized Cell (or we can say Infected Cell) and Unparasitized Cell (or we can say Uninfected Cell). The dataset is divided into training and validation sets.

The split is done in 4:1 ratio, i.e. the validation set has 20% of shuffled images from the whole dataset, i.e. 2756 images are present in our validation set.

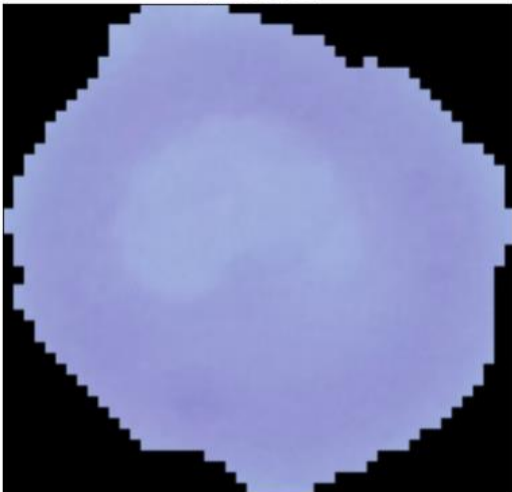
Infected Cell



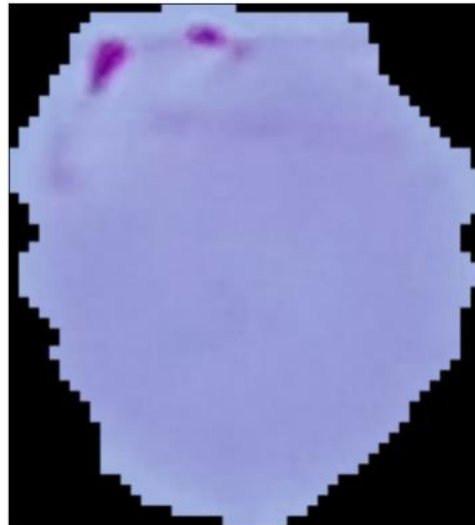
Uninfected Cell



Uninfected Cell



Infected Cell



5.3) Experimental Results

A confusion matrix helps us gain an insight into how correct our predictions were and how they hold up against the actual values.

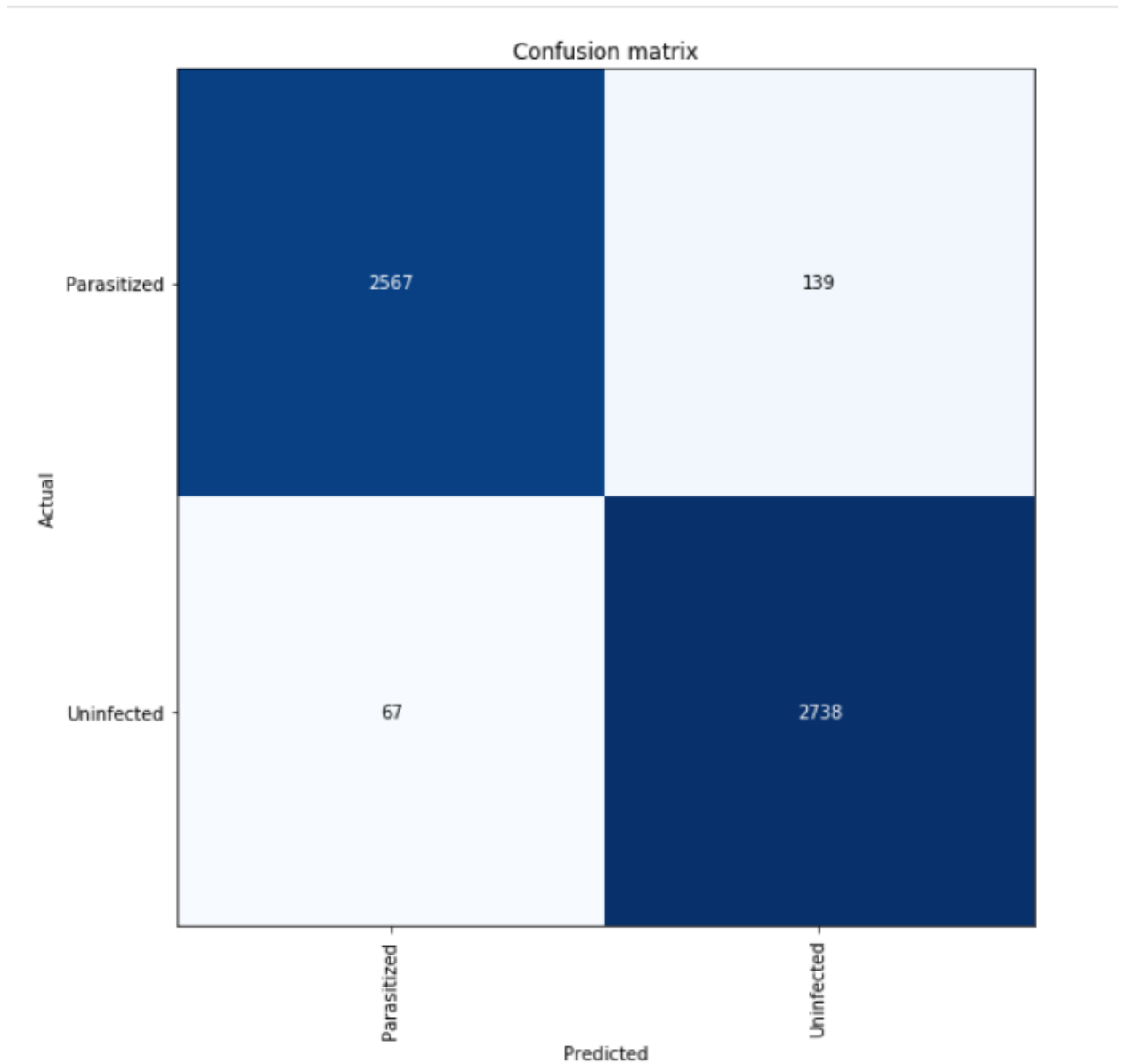


Figure 15: Confusion Matrix

The confusion matrix containing data of Precision, Recall, F1-score and accuracy is as follows:

From this confusion matrix, we can infer that our proposed model has an accuracy of 96.39%, with precision being 95%, recall being 98% and F1-score being 96%.

	precision	recall	f1-score	support
Parasitized	0.97	0.95	0.96	2706
Uninfected	0.95	0.98	0.96	2805
accuracy			0.96	5511
macro avg	0.96	0.96	0.96	5511
weighted avg	0.96	0.96	0.96	5511

Figure 16: Accuracy Chart

Plotting the accuracy curve, we can see:

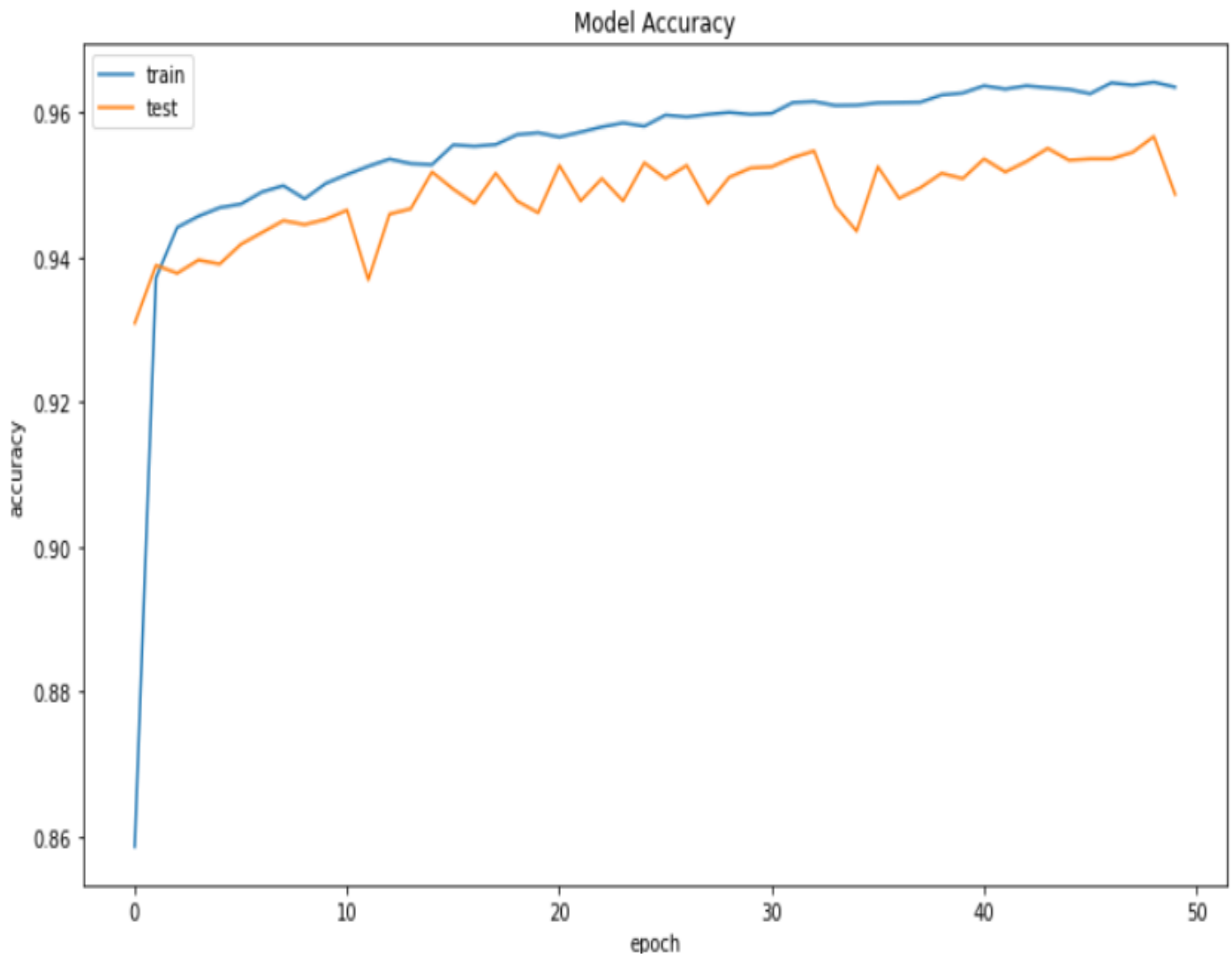


Figure 17: Model Accuracy

The loss can be visualized as:

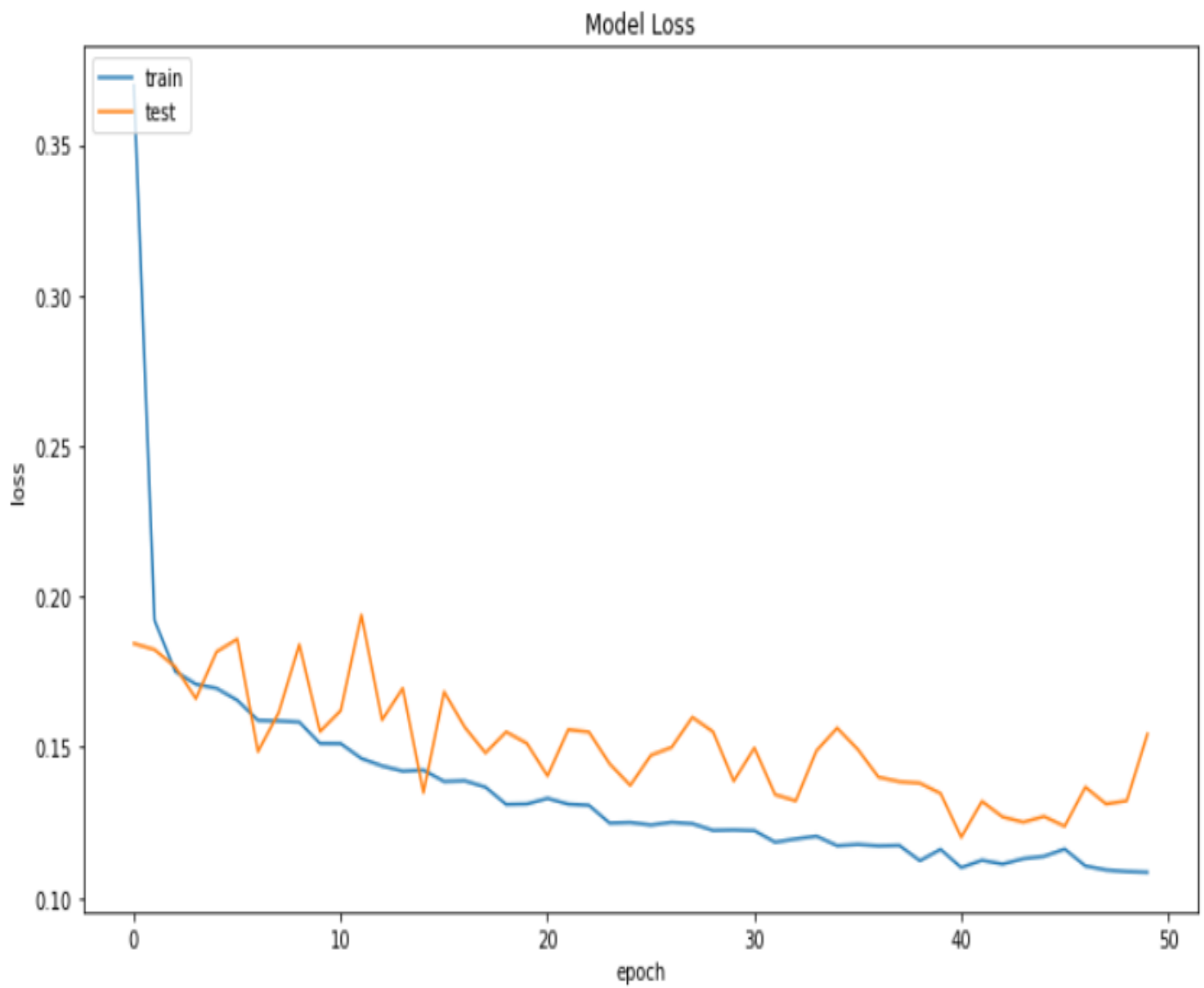


Figure 18: Loss graph

CHAPTER 6

6.Conclusion

When doing the performance measure of our proposed work for malaria detection using classical machine learning algorithms, we can see that our model performs superiorly than other classical ones.

Algorithm	Accuracy	Precision	Recall	F-Score
Proposed Model	0.9636	0.95	0.98	0.96
Ada Boost	0.962	0.729	0.526	0.611
Decision Tree	0.946	0.526	0.529	0.527
KNN	0.940	0.465	0.278	0.348
Linear Regression	0.943	0.375	0.000	0.000
Naïve Bayes	0.858	0.271	0.873	0.414
Random Forest	0.965	0.775	0.553	0.645
Extra Trees	0.956	0.837	0.298	0.440
SVM	0.7889	-	-	-

CHAPTER 7

7. Future Works

There are way too many manual works in medicine. Even for physicians, it is difficult to determine any disease from some raw data, which is why many healthcare industries are turning towards machine learning and deep learning technology to determine disease. With these types of advanced analytics, we can provide physicians with better information while caring for the patient.

Deep learning is going to benefit the practitioner or internist at the bedside. Machine learning can provide objective feedback to improve efficiency, reliability, and accuracy.

Models like the one proposed in our project will serve as tools that medical practitioners will use to improve the ongoing facilities in the malaria detection domain, as well as other similar domains.

Since the dataset is ever increasing in real time, our model can be further trained on large volumes of data and on better systems with high computational powers and thus accuracy and efficiency can be further boosted.

REFERENCES

1. LeCun Y. Lenet-5, Convolutional Neural Networks. [(accessed on 20 May 2015)]; URL. Available online: <http://yann.lecun.com/exdb/lenet>.
2. Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks; Proceedings of the Advances in Neural Information Processing Systems; Lake Tahoe, Nevada. 3–6 December 2012; pp. 1097–1105.
3. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. Going deeper with convolutions; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; Boston, MA, USA. 7–12 June 2015; pp. 1–9.
4. Dong Y., Jiang Z., Shen H., Pan W.D., Williams L.A., Reddy V.V., Benjamin W.H., Bryan A.W. Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells; Proceedings of the 2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI); Orlando, FL, USA. 16–19 February 2017; pp. 101–104.
5. Linder N., Turkki R., Walliander M., Mårtensson A., Diwan V., Rahtu E., Pietikäinen M., Lundin M., Lundin J. A malaria diagnostic tool based on computer vision screening and visualization of plasmodium falciparum candidate areas in digitized blood smears. PLoS ONE. 2014;9:e104855. [PMC free article] [PubMed]
6. Opoku-Ansah J., Eghan M.J., Anderson B., Boampong J.N. Wavelength markers for malaria (plasmodium falciparum) infected and uninfected red blood cells for ring and trophozoite stages. Appl. Phys. Res. 2014;6:47.
7. Wold S., Esbensen K., Geladi P. Principal component analysis. Chemom. Intell. Lab. Syst. 1987;2:37–52.

8. S. Rajaraman, S. K. Antani, M. Poostchi et al., "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, article e4568, 2018.
9. Liang Z., Powell A., Ersoy I., Poostchi M., Silamut K., Palaniappan K., Guo P., Hossain M.A., Sameer A., Maude R.J., et al. Cnn-based image analysis for malaria diagnosis; *Proceedings of the 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*; Shenzhen, China. 15–18 December 2016; pp. 493–496.
10. Deng J., Dong W., Socher R., Li L.J., Li K., Fei-Fei L. Imagenet: A large-scale hierarchical image database; *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*; Miami, FL, USA. 20–25 June 2009; pp. 248–255.
11. Hung J., Carpenter A. Applying faster r-cnn for object detection on malaria images; *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*; Honolulu, HI, USA. 21–26 July 2017; pp. 56–61.
12. J.A. Quinn, A. Andama, I. Munabi, F.N. Kiwanuka Automated blood smear analysis for mobile malaria diagnosis *Mobile Point Care Monitors Diagno Device Design*, 31 (2014), p. 115
13. L. Rosado, J.M.C. da Costa, D. Elias, J.S. Cardoso Automated detection of malaria parasites on thick blood smears via mobile devices *Proc Comput Sci*, 90 (2016), pp. 138-144
14. S. Herrera, A.F. Vallejo, J.P. Quintero, M. Arévalo-Herrera, M. Cancino, S. Ferro Field evaluation of an automated RDT reader and data management device for *Plasmodium falciparum*/*Plasmodium vivax* malaria in endemic areas of Colombia *Malar J*, 13 (2014), p. 87
15. M. Cesario, M. Lundon, S. Luz, M. Masoodian, B. Rogers Mobile support for diagnosis of communicable diseases in remote locations *Proceedings of the 13th international conference of the NZ chapter of the ACM's special interest group on human-computer interaction*, ACM (2012), pp. 25-28
16. C. Dallet, S. Kareem, I. Kale Real time blood image processing application for malaria diagnosis using mobile phones *International conference on circuits and systems*, IEEE (2014), pp. 2405-2408

17. A. Skandarajah, C.D. Reber, N.A. Switz, D.A. Fletcher Quantitative imaging with a mobile phone microscope PLoS ONE, 9 (2014) e96906
18. C.W. Pirnstill, G.L. Coté Malaria diagnosis using a mobile phone polarized microscope Sci Rep, 5 (2015), p. 13368
19. D.N. Breslauer, R.N. Maamari, N.A. Switz, W.A. Lam, D.A. Fletcher Mobile phone based clinical microscopy for global health applications PLoS ONE, 4 (2009) e6320
20. H. Shen, W. D. Pan, Y. Dong, and M. Alim, “Lossless compression of curated erythrocyte images using deep autoencoders for malaria infection diagnosis,” in 2016 Picture Coding Symposium (PCS), pp. 1–5, Nuremberg, Germany, December 2016.
21. I. Mohanty, P. A. Pattanaik, and T. Swarnkar, “Automatic detection of malaria parasites using unsupervised techniques,” in International Conference on ISMAC in Computational Vision and Bio-Engineering, pp. 41–49, Springer, Cham, 2018.
22. H. Jane and A. Carpenter, “Applying faster R-CNN for object detection on malaria images,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 56–61, Honolulu, HI, USA, 2017.
23. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: a large-scale hierarchical image database,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255, Miami, FL, USA, June 2009.
24. D. Bibin, M. S. Nair, and P. Punitha, “Malaria parasite detection from peripheral blood smear images using deep belief networks,” IEEE Access, vol. 5, pp. 9099–9108, 2017.
25. X. Yang, T. Zhang, C. Xu, S. Yan, M. S. Hossain, and A. Ghoneim, “Deep relative attributes,” IEEE Transactions on Multimedia, vol. 18, no. 9, pp. 1832–1842, 2016.

26. M. I. Razzak and S. Naz, "Microscopic blood smear segmentation and classification using deep contour aware CNN and extreme machine learning," in 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 801–807, Honolulu, HI, USA, July 2017.
27. V. Kantorov, M. Oquab, M. Cho, and I. Laptev, "Contextlocnet: context-aware deep network models for weakly supervised localization," in Computer Vision – ECCV 2016, pp. 350–365, Springer, 2016.
28. G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.
29. D. Anggraini, A. S. Nugroho, C. Pratama, I. E. Rozi, A. A. Iskandar, and R. N. Hartono, "Automated status identification of microscopic images obtained from malaria thin blood smears," in Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, Bandung, Indonesia, July 2011.
30. MOMALA," <https://momala.org/malaria-diagnosis/>.
31. "This New App Helps Doctors Diagnose Malaria in Just 2 Minutes," <https://www.globalcitizen.org/en/content/app-diagnose-malaria-uganda>.
32. N. E. Ross, C. J. Pritchard, D. M. Rubin, and A. G. Dusé, "Automated image processing method for the diagnosis and classification of malaria on thin blood smears," *Medical & Biological Engineering & Computing*, vol. 44, no. 5, pp. 427–436, 2006.
33. D. K. Das, M. Ghosh, M. Pal, A. K. Maiti, and C. Chakraborty, "Machine learning approach for automated screening of malaria parasite using light microscopic images," *Micron*, vol. 45, pp. 97–106, 2013.
34. M. Poostchi, K. Silamut, R. J. Maude, S. Jaeger, and G. R. Thoma, "Image analysis and machine learning for detecting Malaria," *Translational Research*, vol. 194, pp. 36–55, 2018.

35. M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic fruit classification using deep learning for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1027–1034, 2019.
36. M. Usama, B. Ahmad, J. Wan, M. S. Hossain, M. F. Alhamid, and M. A. Hossain, "Deep feature learning for disease risk assessment based on convolutional neural network with intra-layer recurrent connection by using hospital big data," *IEEE Access*, vol. 6, pp. 67927–67939, 2018.
37. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
38. Y. Zhang, Y. Qian, D. Wu, M. Shamim Hossain, A. Ghoneim, and M. Chen, "Emotion-aware multimedia Systems security," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 617–624, 2019.
39. Y. Zhang, X. Ma, J. Zhang, M. S. Hossain, G. Muhammad, and S. U. Amin, "Edge intelligence in the cognitive internet of things: improving sensitivity and interactivity," *IEEE Network*, vol. 33, no. 3, pp. 58–64, 2019.
40. X. Ma, R. Wang, Y. Zhang, C. Jiang, and H. Abbas, "A name disambiguation module for intelligent robotic consultant in industrial internet of things," *Mechanical Systems and Signal Processing*, vol. 136, article 106413, 2020.
41. Liang Z., Powell A., Ersoy I., Poostchi M., Silamut K., Palaniappan K., Guo P., Hossain M.A., Sameer A., Maude R.J., et al. Cnn-based image analysis for malaria diagnosis; *Proceedings of the 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*; Shenzhen, China. 15–18 December 2016; pp. 493–496.
42. Gopakumar G.P., Swetha M., Sai Siva G., SaiSubrahmanyam G.R.K. Convolutional neural network-based malaria diagnosis from focus stack of blood smear images acquired using custom-built slide scanner. *J. Biophotonics*. 2018;11:e201700003. doi: 10.1002/jbio.201700003.

43. <https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7>
44. <https://medium.com/@sergioalves94/deep-learning-in-pytorch-with-cifar-10-dataset-858b504a6b54>
45. <https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e>
46. <https://lhncbc.nlm.nih.gov/LHC-downloads/downloads.html#malaria-datasets>

Appendix

CODE

```
# This Python 3 environment comes with many helpful analytics li  
braries installed  
# It is defined by the kaggle/python Docker image: https://githu  
b.com/kaggle/docker-python  
# For example, here's several helpful packages to load  
  
import numpy as np # linear algebra  
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)  
  
# Input data files are available in the read-only "../input/" di  
rectory  
# For example, running this (by clicking run or pressing Shift+E  
nter) will list all files under the input directory  
  
import os  
  
# You can write up to 20GB to the current directory (/kaggle/wor  
king/) that gets preserved as output when you create a version u  
sing "Save & Run All"  
# You can also write temporary files to /kaggle/temp/, but they  
won't be saved outside of the current session  
  
# Setting paths and showing the number of images  
infected = os.listdir("cell_images/cell_images/Parasitized")  
infected_path = "cell_images/cell_images/Parasitized"  
print("Length of infected data = ", len(infected), 'images')  
uninfected = os.listdir("cell_images/cell_images/Uninfected")
```

```

uninfected_path = "cell_images/cell_images/Uninfected"
print("Length of uninfected data = ", len(uninfected), 'images')

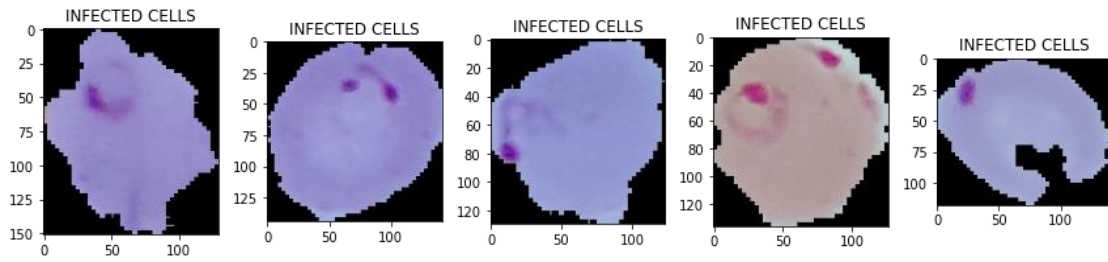
Length of infected data = 13780 images
Length of uninfected data = 13780 images

from glob import glob
from PIL import Image
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import fnmatch
import tensorflow as tf
from time import sleep
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Dropout, Flatten, BatchNormalization, MaxPooling2D, Activation
from tensorflow.keras.optimizers import RMSprop, Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras import backend as k
import matplotlib.image as mpimg
import os
from tensorflow.keras import layers
from tensorflow.keras import Model

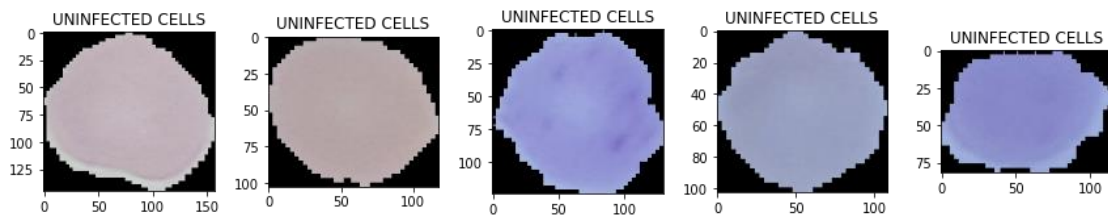
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator
plt.rcParams['figure.figsize'] = (12,7)

for i in range(5):
    plt.subplot(1, 5, i+1)
    plt.imshow(cv2.imread(
infected_path + '/' + infected[i]))
    plt.title('INFECTED CELLS')
    plt.tight_layout()
plt.show()

```



```
for i in range(5):
    plt.subplot(1, 5, i+1)
    plt.imshow(cv2.imread(uninfected_path + '/' + uninfected[i])
    )
    plt.title('UNINFECTED CELLS')
    plt.tight_layout()
plt.show()
```



```
img_shape = (64, 64, 3)
image_gen = ImageDataGenerator(rotation_range=45,
                                width_shift_range=0.1,
                                height_shift_range=0.1,
                                rescale=1/225,
                                shear_range=0.2,
                                zoom_range=0.1,
                                vertical_flip=True,
                                horizontal_flip=True,
                                fill_mode='nearest',
                                validation_split=0.2)

train_generator = image_gen.flow_from_directory('cell_images/cel
l_images',
                                                target_size = img_shape[:2]
,
                                                color_mode = 'rgb',
                                                batch_size = 32,
                                                class_mode = 'binary',
                                                subset = 'training',
```

```

shuffle = True,
classes=['Uninfected', 'Parasitized'])

validation_generator = image_gen.flow_from_directory('cell_images/cell_images',
                                                    target_size = img_shape[:2],
                                                    color_mode = 'rgb',
                                                    batch_size = 32,
                                                    class_mode = 'binary',
                                                    subset = 'validation',
                                                    shuffle = False,
                                                    classes=['Uninfected', 'Parasitized'])

```

```

Found 22048 images belonging to 2 classes.
Found 5510 images belonging to 2 classes.

```

```

print(train_generator.class_indices)
print(validation_generator.class_indices)

```

```

{'Uninfected': 0, 'Parasitized': 1}
{'Uninfected': 0, 'Parasitized': 1}

```

```

from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

```

```

#classifier = Sequential()

```

```

#classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))

```

```

#classifier.add(MaxPooling2D(pool_size = (2, 2)))

```

```

#classifier.add(Conv2D(32, (3, 3), activation = 'relu'))

```

```

#classifier.add(MaxPooling2D(pool_size = (2, 2)))

```

```

#classifier.add(Conv2D(32, (3, 3), activation = 'relu'))

```

```

#classifier.add(MaxPooling2D(pool_size = (2, 2)))

```



```

#classifier.add(Flatten())

#classifier.add(Dense(units = 128, activation = 'relu'))
#classifier.add(Dense(units = 1, activation = 'sigmoid'))

#classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

#classifier.fit_generator(train,
#steps_per_epoch = 5000,
#epochs = 10,
#validation_data = validation,
#validation_steps = 2000)

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', padding
='same', input_shape=(64, 64, 3)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(32, (3,3), activation='relu', padding
='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding
='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu', padding
='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.5),

```

```

tf.keras.layers.Dense(512, activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.5),
tf.keras.layers.Dense(1, activation='sigmoid')
])

```

```

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['acc'])

```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 64, 64, 16)	448

batch_normalization (Batch Normalization)	(None, 64, 64, 16)	64

max_pooling2d (MaxPooling2D)	(None, 32, 32, 16)	0

conv2d_1 (Conv2D)	(None, 32, 32, 32)	4640

batch_normalization_1 (Batch Normalization)	(None, 32, 32, 32)	128

max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0

conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496

batch_normalization_2	(Batch (None, 16, 16, 64)	256
max_pooling2d_2	(MaxPooling2 (None, 8, 8, 64)	0
conv2d_3	(Conv2D) (None, 8, 8, 128)	73856
batch_normalization_3	(Batch (None, 8, 8, 128)	512
max_pooling2d_3	(MaxPooling2 (None, 4, 4, 128)	0
flatten	(Flatten) (None, 2048)	0
dropout	(Dropout) (None, 2048)	0
dense	(Dense) (None, 1024)	2098176
batch_normalization_4	(Batch (None, 1024)	4096
dropout_1	(Dropout) (None, 1024)	0
dense_1	(Dense) (None, 512)	524800
batch_normalization_5	(Batch (None, 512)	2048
dropout_2	(Dropout) (None, 512)	0

—

dense_2 (Dense)	(None, 1)	513
-----------------	-----------	-----

=====

=

Total params: 2,728,033
Trainable params: 2,724,481
Non-trainable params: 3,552

set early stopping, to avoid overtraining

```
early_stop = EarlyStopping(monitor='val_loss',patience=3)
```

```
history = model.fit_generator(
    train_generator,
    epochs=50,
    validation_data=validation_generator,
    callbacks=[early_stop]
)
```

/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

warnings.warn("`Model.fit_generator` is deprecated and "

Epoch 1/50

689/689 [=====] - 254s 363ms/step - loss: 0.6039 - acc: 0.7576 - val_loss: 0.2383 - val_acc: 0.9134

Epoch 2/50

689/689 [=====] - 67s 97ms/step - loss: 0.1970 - acc: 0.9341 - val_loss: 0.2033 - val_acc: 0.9310

Epoch 3/50

689/689 [=====] - 69s 100ms/step - loss: 0.1787 - acc: 0.9419 - val_loss: 0.1624 - val_acc: 0.9414

Epoch 4/50

689/689 [=====] - 68s 98ms/step - loss: 0.1687 - acc: 0.9430 - val_loss: 0.1677 - val_acc: 0.9410

Epoch 5/50

689/689 [=====] - 67s 97ms/step - loss: 0.1560 - acc: 0.9517 - val_loss: 0.1701 - val_acc: 0.9401

```
Epoch 6/50
689/689 [=====] - 68s 98ms/step - loss:
0.1632 - acc: 0.9466 - val_loss: 0.1714 - val_acc: 0.9407
```

```
model.evaluate(validation_generator)
```

```
173/173 [=====] - 13s 76ms/step - loss:
0.1714 - acc: 0.9423
```

```
[0.1713898479938507, 0.9422867298126221]
```

```
173/173 [=====] - 13s 76ms/step - loss:
0.1697 - acc: 0.9430
```

```
[0.16965334117412567, 0.9430127143859863]
```

```
pred_probabilities = model.predict(validation_generator)
```

```
predictions = pred_probabilities > 0.5
```

```
predictions
```

```
array([[False],
       [False],
       [False],
       ...,
       [ True],
       [ True],
       [ True]])
```

```
print(classification_report(validation_generator.classes, predic
tions))
```

	precision	recall	f1-score	support
0	0.91	0.97	0.94	2755
1	0.97	0.91	0.94	2755
accuracy			0.94	5510
macro avg	0.94	0.94	0.94	5510

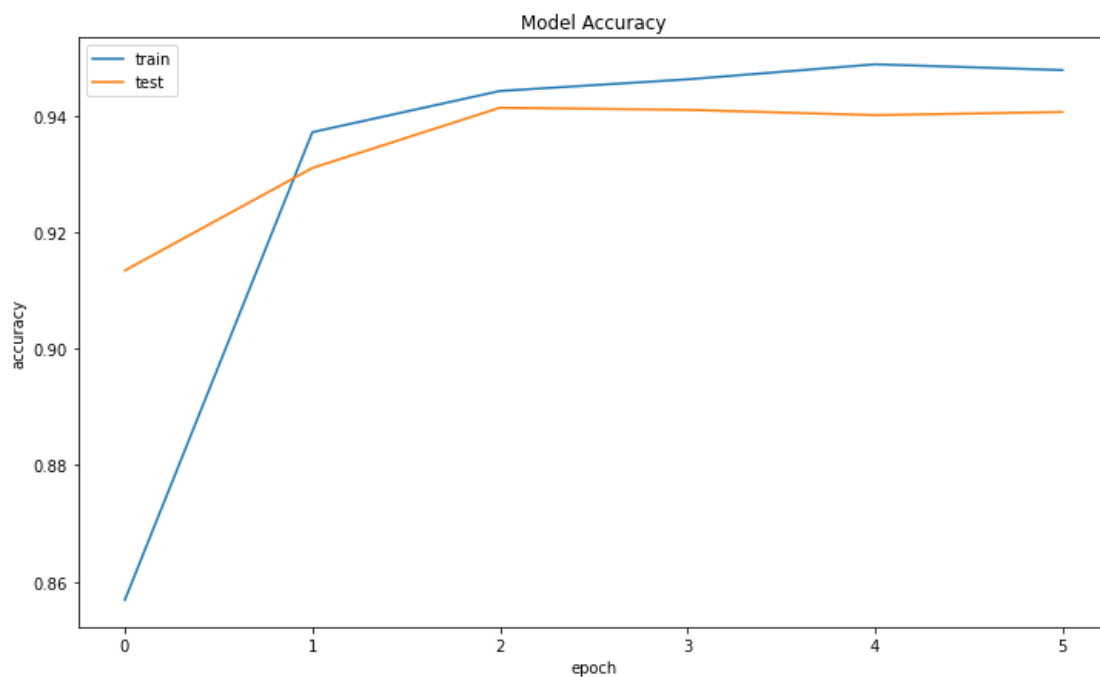
weighted avg	0.94	0.94	0.94	5510
--------------	------	------	------	------

```
confusion_matrix = pd.DataFrame(confusion_matrix(validation_generator.classes, predictions))
confusion_matrix
```

	0	1
0	2679	76
1	259	2496

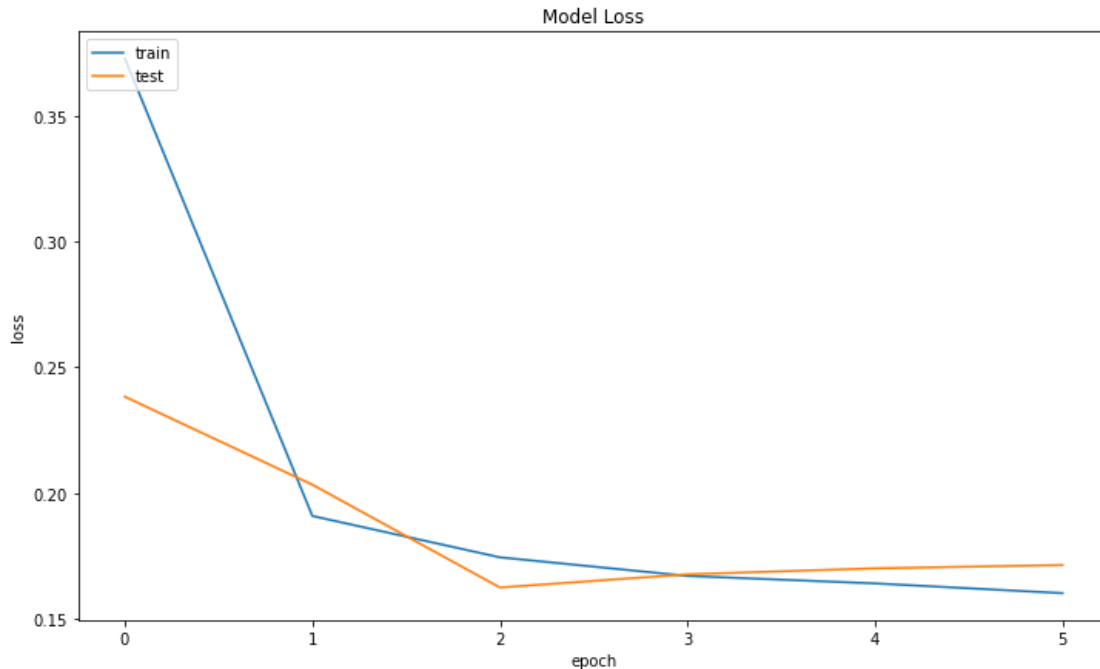
```
from sklearn.metrics import classification_report, confusion_matrix
```

```
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
test_ = image_gen.flow_from_directory('cell_images/cell_images',
                                     target_size = img_shape[
:2],
                                     color_mode = 'rgb',
                                     batch_size=32,
                                     class_mode='binary',
                                     subset='validation',
                                     shuffle = True
                                     )
```

Found 5510 images belonging to 2 classes.

```
model.evaluate_generator(test_, verbose=1)
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate`, which supports generators.
```

```
warnings.warn("`Model.evaluate_generator` is deprecated and "
```

```
173/173 [=====] - 13s 77ms/step - loss:
4.1213 - acc: 0.0593
[4.12133264541626, 0.05934664234519005]
model.save('malaria_detection.h5')
```
