MALARIA CELL DETECTION USING DEPTHWISE SEPARABLE CNN

A PROJECT REPORT

Submitted By

Niel Parekh 185001100

Sai Charan 185001131

Satya Bharati 185001138

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Rajiv Gandhi Salai (OMR), Kalavakkam - 603110

June 2022

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report titled **"Malaria Cell Detection using Depthwise Separable CNN"** is the *bonafide* work of "**Niel Parekh (185001100)**, **Sai Charan (185001131)**, and **Satya Bharathi (185001138)**" who carried out the project work under my supervision.
Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. T.T. Mirnalinee
Head of the Department
Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Dr. Jansi Rani S.V
Supervisor
Associate Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Place:
Date:

Submitted for the examination held on. . . . . .

Internal Examiner **External Examiner**

ACKNOWLEDGEMENTS

I thank GOD, the almighty for giving me strength and knowledge to do this project.

I would like to thank and deep sense of gratitude to my guide **Dr. JANSI RANI S.V**, Associate Professor, Department of Computer Science and Engineering, for her valuable advice and suggestions as well as her continued guidance, patience and support that helped me to shape and refine my work.

My sincere thanks to **Dr. T.T. MIRNALINEE**, Professor and Head of the Department of Computer Science and Engineering, for her words of advice and encouragement and I would like to thank our project Coordinator **Dr.B. BHARATHI**, Associate Professor, Department of Computer Science and Engineering for her valuable suggestions throughout this project.

I express my deep respect to the founder **Dr. SHIV NADAR**, Chairman, SSN Institutions. I also express my appreciation to our **Dr. V.E. Annamalai**, Principal, for all the help he has rendered during this course of study.

I would like to extend my sincere thanks to all the teaching and non-teaching staffs of our department who have contributed directly and indirectly during the course of my project work. Finally, I would like to thank my parents and friends for their patience, cooperation and moral support throughout my life.

Niel Parekh **Sai Charan** **Satya Bharati**

ABSTRACT

Malaria is a deadly virus that affects lots of impoverished regions in developing countries and has the capability of proving fatal. Some of these regions don't have access to devices capable of performing complex computations. A challenge faced by these the doctors of these regions is correctly diagnosing the disease with the hardware accessible to them. In order to run a malaria cell detection algorithm on less efficient hardware, the model must be less computationally expensive. Our solution aims at reducing the amount of computational power required and hence reducing the time taken to correctly identify a malaria infected cell. This was to be achieved without it reducing the accuracy. We have achieved this using Convolutional Neural Networks (CNNs) along with Depthwise Separable Layers. These layers reduce the number of multiplications required during training significantly which thereby reduces the amount of time taken to identify an affected cell. The dataset used consists of 27,000 segmented images belonging to 2 classes (Infected and Uninfected). Our solution was also compared to existing pre-trained models such as ResNet50, InceptionV3 and a stacked CNN in order to prove it's efficacy. The time taken by the ResNet50, InceptionV3 and the Stacked CNN are 179, 133 and 76 seconds respectively. Our depthwise separable CNN (DS CNN) runs 4 times than the other models taking

just 39 seconds. The models achieve accuracies of 90%, 91% and 90% respectively. Our model achieves an accuracy of 93% which is 2% more than what the other models achieve. This system will help healthcare officials in developing nations by enabling them to run an accurate malaria cell detection algorithm both quickly and on the hardware accessible to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Blood smears are the most common and accurate test for the diagnosis of malaria. The attending physician will take a sample of the blood and the sample is sent to a lab where it is stained and observed (manually / by a human) under a microscope. While the above method has proven to be effective, it can be seen that there is a lot of room for error. There is always the risk that the test will return a false positive or a false negative, both of which engender their own negative consequences. Automating this process will have greater precision and would definitely help reduce the number of false positives or negatives. Malaria can be fatal within 24 hours after symptoms appear. A false negative result that is not quickly identified can cost a life. Moreover, during these hard times, doctors already have their hands full with COVID running rampant. Taking any load off of technicians or physicians would go a long way. Also, most malaria stricken regions are severely underdeveloped and do not have access to proper healthcare facilities. Hence, using lightweight CNNs to detect malaria would increase the rate of diagnosis and help curb the disease. The current smear tests have an accuracy of 85% in detecting infected cells. We hope to achieve a larger number using CNNs.

1.1 MALARIA

Malaria is a serious and sometimes fatal disease. The Plasmodium parasite initially infects a certain species of mosquito, the Anopheles mosquito, which

then can transmit the parasite to a human by biting them. The parasite is released into the bloodstream and travels to the liver, where the parasite matures. Upon maturing, they re-enter the bloodstream and infect Red Blood Cells. The infected human can transmit the parasite to another anopheles mosquito and hence the cycle continues. There are four kinds of Malaria parasites known to man: Plasmodium falciparum, P. vivax, P. ovale, and P. malariae. Plasmodium falciparum is the most common cause of malaria accounting for up to 50% of all malaria cases. Symptoms of malaria include but are not limited to, chills, headaches, muscle aches, tiredness, fevers, nausea, and vomiting. Malaria may also cause anemia and jaundice. In more serious cases, it is also known to cause renal failure, seizures, coma, and death. There are multiple antimalarial drugs that should be taken as early as possible after the initial infection. It is of paramount importance to catch the disease as early as possible, to visit the doctor immediately upon noticing any of the aforementioned symptoms or upon receiving the knowledge of an outbreak of malaria in any place you have recently visited. In 2019 there were 299 million cases of malaria, a number that increased by 1 million from that of one year ago. Malaria has been responsible for 410,000 deaths in 2019 alone. Its effects are more prominent in Africa. The region was home to 94% of all malaria cases and deaths. India also has been a key site for the Anopheles vector to transmit the parasite to humans according to the CDC.

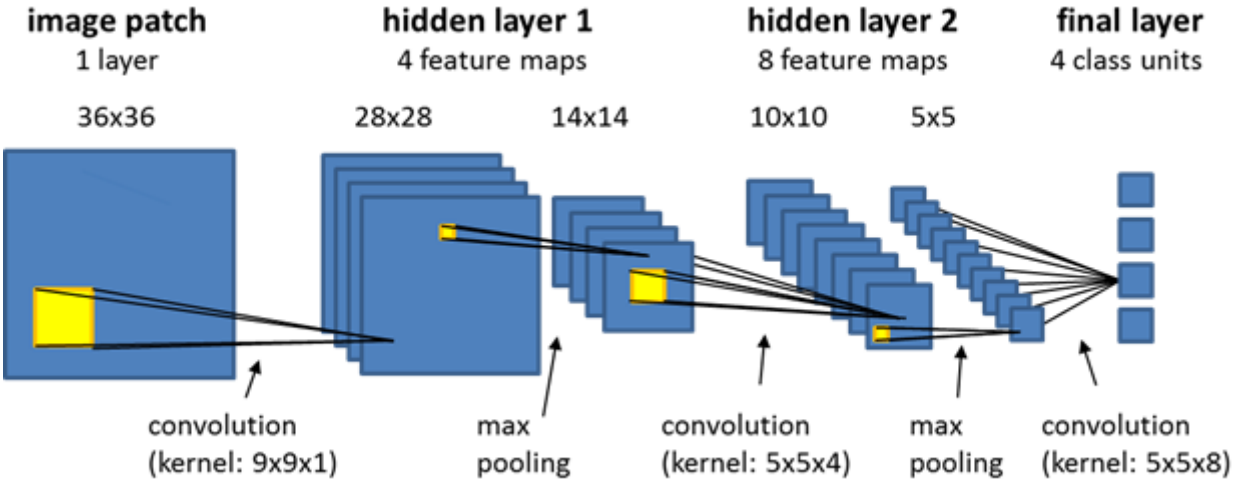

FIGURE 1.1: Convolutional Neural Network Architecture

CHAPTER 2

CONVOLUTIONAL NEURAL NETWORKS

Out of the various types of neural networks that exist, the one that is focused on is the Convolutional Neural Network (CNN), which has image processing as its focal point. Digital images can be defined as binary representations of visual data. It comprises the arrangement of pixels in a matrix-like structure, with the pixel values indicating the brightness and the colour of the pixel. A convolution network highly resembles the human brain. This is intentional as these networks were modelled after the ocular system present in animals. The connection patterns of neurons mimic these systems in the way they are structured.

CNNs do not require much data augmentation or pre processing as do other machine or deep learning models. It essentially means that while in traditional algorithms the filters are hand-engineered, the network, on the other hand, learns to optimize these kernels or filters via automated learning, which in fact proves to be advantageous since it provides independence from prior knowledge as well as human intervention in feature extraction.

2.1 CONVOLUTIONAL LAYER

A mathematical function is applied to an input. The filters that have been defined in the convolution layer act as a sliding window on the input, extracting its features which allows us to obtain feature maps. After having passed through the convolutional layer, the image is then abstracted to a feature map, also known as

an activation map.

Next, the convolution layer folds the input and then passes the result to the succeeding layer. This process mimics the neurons in the ocular system of an animal to stimuli. These convolutional neural networks are used to learn utility in addition to classifying data. However, this would not work in every scenario. For example, using CNNs for higher resolution images is not practical and other machine learning methods would be more optimal since the input size of the image and hence the features of the input associated with each pixel are of a large size. Instead, the number of free parameters is reduced in convolution, thus allowing the network to be deeper. The use of regularized weights with fewer parameters evades the vanishing gradient and the explosion gradient problems which occur during back-propagation in the traditional neural networks. Additionally, convolutional neural networks prove to be ideal for data that have a grid-like topology (like images) because they consider the semantic relationships between individual features during convolution an / or pooling.

To form the entire output, the activation maps from the filters are put on top of each other. Hence, they can be viewed as neuron outputs of a small region that have similar parameters to the activation map.

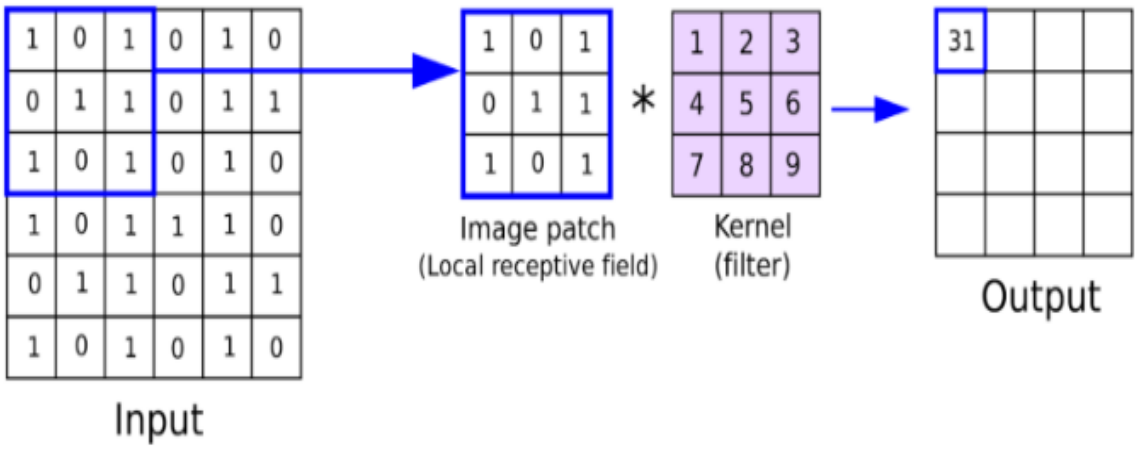

FIGURE 2.1: Working of a convolutional layer.

2.2 POOLING LAYER

The trained filters to the input image are systematically applied by the convolutional layer of the convolutional neural network to build a feature map which summarizes the presence of those particular filters in the input. The stacking of the convolution layers in a deep model with layers near the input which learn low-level features like lines etc., and deeper layers in the model which learns shapes and specific objects proves their high effectiveness. One would be able to learn higher or more abstract features like.

One limitation of the feature map output of the convolution is the noting down of the precise location of features given in the input, meaning that even small shifts in the position of feature positions in the inputs can impact the feature maps. This can happen even if one makes even a minor change to the input. A common practice followed to address this problem is using the concept of downsizing. Downsizing

creates a version of the input by lowering the resolution however maintaining the important structural elements excluding details that are not be very useful.

Each feature map interacts with the pooling layer individually, generating sets of

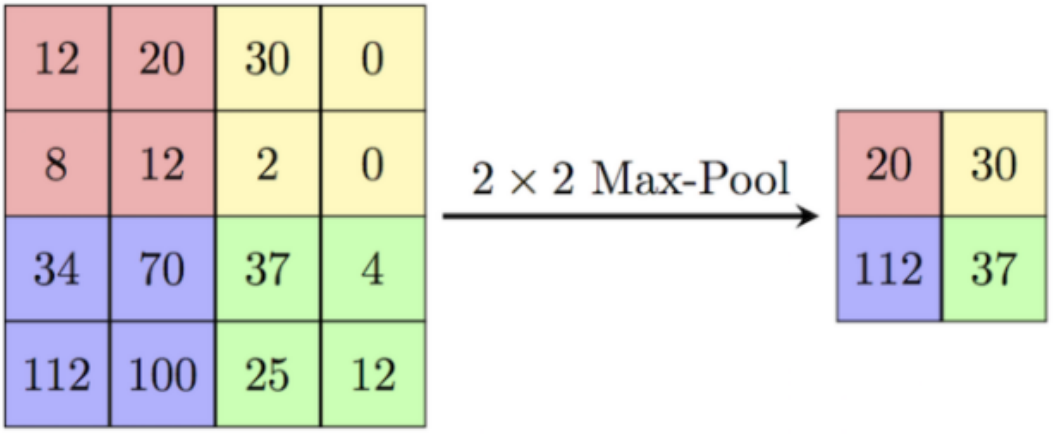

FIGURE 2.2: MaxPooling operation

the feature maps. Pooling implements the usage of a pooling operation which can be compared to filters applied to filter maps.

The pooling operation size is always smaller when compared to the feature map size. Generally, it is of 2×2 pixels and the stride is of 2 pixels. But this can vary. This indicates that it will reduce the feature map size by 2 in all cases.

2.3 FULLY CONNECTED LAYER

In this layer, all the neurons are fully connected with all the neurons in the previous and next layers.

Fully Connected Layer (also known as Hidden Layer) is the last layer in the

convolutional neural network. This layer is a combination of Affine function and Non-Linear function. Fully Connected layers take inputs from Flatten Layer which is a one-dimensional layer (1D Layer). The data coming from Flatten Layer is passed first to Affine function and then to Non-Linear function. The combination of 1 Affine function and 1 Non-Linear Function is called as 1 FC (Fully Connected) or 1 Hidden Layer. We can add multiple such layers based on the depth to which we want to take our classification model.

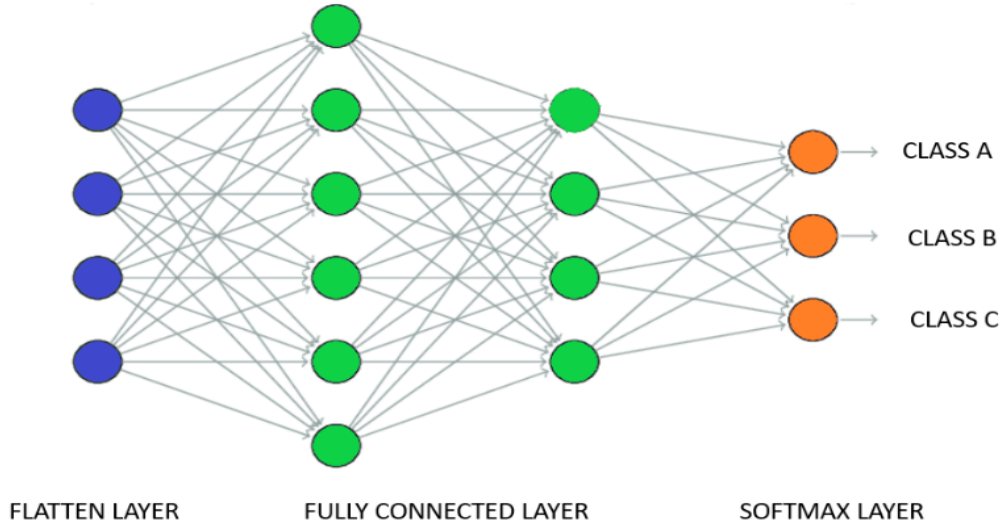

FIGURE 2.3: Fully connected layer

2.4 OPTIMIZERS

A deep learning model uses an algorithm to read and comprehend the data fed to them and will be able to make predictions as per the requirements. Broadly, there are 2 algorithms required to make these predictions: an algorithm that maps input examples to output examples; and an optimization algorithm. The optimization algorithm is used to modify certain properties of a neural network in order to

minimize a loss function or maximize the efficiency of the model. The optimizer determines how the weights/learning rates of the neural network are changed. Essentially, they are mathematical functions that are dependent on the learnable parameters of the model, such as Weights and Biases. They are also vital in minimizing the losses and therefore they play a huge role in determining the accuracy of the model. By changing the weights, optimizers shape and mould your model into the most accurate form possible. When the optimizer is heading in the correct or wrong path, the loss function directs it. There is a wide range of optimizers available to make changes to these weights and rates. Choosing one from them is dependent on its application. Since even a single epoch can take a large amount of time for larger samples of data, the optimizers should be chosen carefully.

2.4.1 ADAM (ADAPTIVE MOMENT ESTIMATION)

Adam is a first-order gradient-based algorithm for stochastic objectives. Adam is one of the more modern optimization algorithms being used in Deep Learning. Adaptive Moment Estimation is an optimization algorithm for gradient descent. It is extremely useful when a lot of data or parameters are involved in the problem. Less memory is required and hence Adam is known to be efficient. Adam also utilizes the average of the second moments of the gradients in contrast to adapting the parameter learning rates based on the mean first moment as in RMSProp. It is simple to implement and is efficient in terms of computational power. As mentioned earlier it's few memory requirements also make it favorable. .

2.4.2 ROOT MEAN SQUARE

RMSprop is a gradient-based technique for optimization in training neural networks. Gradients of very complex functions, such as neural networks, generally disappear or burst as the data propagates through the function. RMSprop solves this problem by implementing a moving average of the squares of the gradient to normalize said gradient. This normalization balances the step size, reduces steps with large gradients to prevent explosions, and increases steps with small gradients to prevent them from disappearing. Simply put, RMSprop uses adaptive learning rates. This means that the learning rate changes over time. RMSProp automatically adjusts the learning rate and selects a different learning rate for each parameter, but the learning rate is slower. It is advantageous as the rate of learning is adjusted automatically. It also chooses different learning rated for different parameters. However, one drawback would be its slow learning.

2.4.3 STOCHASTIC GRADIENT DESCENT

A typical gradient descent optimization technique requires that all samples in the dataset be used to complete one iteration while performing the Gradient Descent. This should be done on each iteration until the minima is reached. Therefore, the computational cost is very high for implementation. Stochastic gradient descent, a simple yet highly efficient approach to fitting linear classifiers as well as regressors under a convex loss function, is used to solve the problem. Under SGD, the calculation does not make use of the sum of the gradients of the cost function of all the samples. Instead, it calculates the gradient of the cost function

of a simple sample at each iteration. Since there is random selection of only one sample from the dataset at each iteration, the path taken by the algorithm to reach minima is usually much noisier than the normal Gradient Descent algorithm. It updates the model parameters one by one. However, the frequency creates a noisy gradient, which can increase the error rather than decrease it. There are certain advantages to a stochastic gradient descent. It frequently updates the model parameter. It also requires less memory. It permits the use of larger datasets. The drawbacks of using the stochastic gradient descent optimizer are: noisy gradients that are caused by the frequency and the higher computational costs

CHAPTER 3

DEPTHWISE SEPERABLE LAYERS

In our CNN, we use depthwise separable convolutional layers. We use depthwise separable convolutional layers instead of normal convolutional layers as they use fewer computations than traditional convolutional layers. Depthwise separable convolution splits a kernel into 2 separate kernels that do two convolutions. One is depthwise convolution. Another is pointwise convolution. This tends to take into account each channel, giving a higher accuracy while also being efficient.

3.1 DEPTHWISE CONVOLUTION

First, depthwise convolution is performed. In this case, the kernel is of the dimensions 5x5x1. The image is of the dimensions 12x12x3. Each kernel iterates 1 channel of the image. There are 3 channels in total. Hence we get an 8x8x1 image from each kernel. Stacking the images from each kernel, we get an 8x8x3 image.

3.2 POINTWISE CONVOLUTION

After this, we perform a pointwise convolution. The kernel is of the dimensions 1x1x3 in this case. Each kernel iterates through all 3 channels of the image. The

FIGURE 3.1: Depthwise convolution

output is an 8x8x1 image. There are 256 of these kernels used. Stacking them all on top of each other results in an 8x8x256 image.

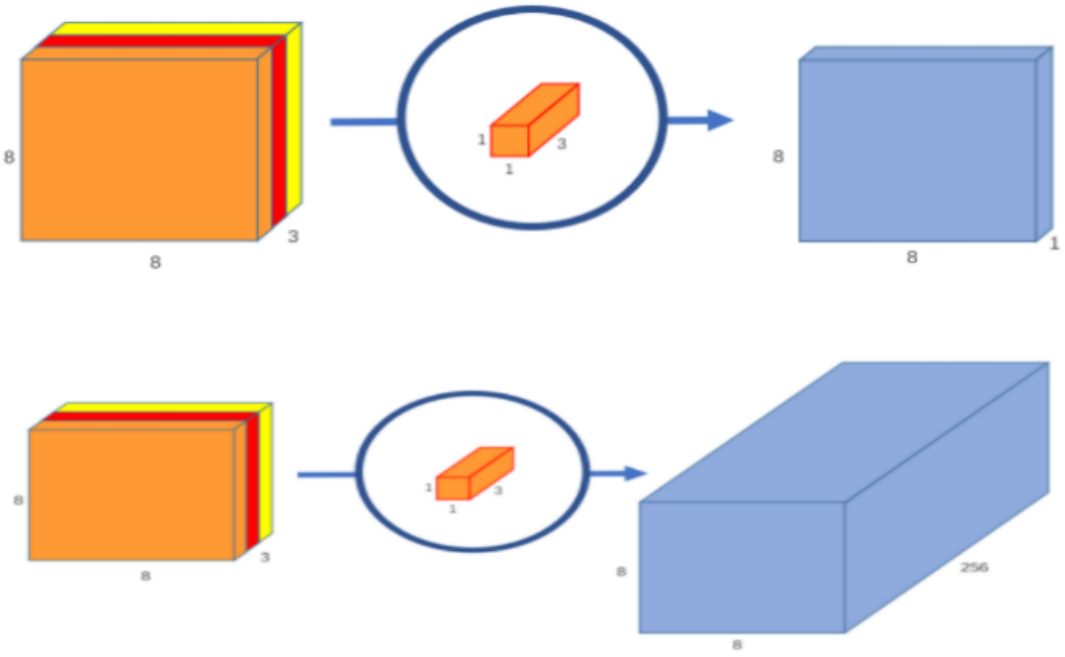

FIGURE 3.2: Pointwise convolution

This way, we attain the same 8x8x256 images, while performing significantly less computations.

3.3 COMPARISON TO REGULAR CNNs

In a normal convolution, assuming a kernel size of 5x5x3 and image size of 12x12x3, the kernel moves 8x8 times and there are 256 of them. Combining these, we get a total of 1,228,800 multiplications.

Depthwise separable convolutions can be divided in two parts : Depthwise convolution and pointwise convolution.

In depthwise convolution, the kernel size would be 5x5x1. The image size will remain the same (12x12x3). It would move 8x8 times and there are 3 of them. Combining these, we get a total of 4,800 multiplications.

In pointwise convolution, the kernels are 1x1x3 in size. They move 8x8 times and there are 256 of them. Combining these we get 49,152 multiplications.

Adding them, we get a total of 53,952 multiplications. So in the end, with the normal convolutional layers, we get 1,228,800 multiplications and with depthwise separable layers, we get 53,952.

As we can see, fewer computations are required for the depthwise separable

Type of convolution	No. of multiplications
Depthwise Convolution	4,800
Pontwise Convolution	49,152

TABLE 3.1: Depthwise and pointwise multiplications

Type of convolution	Total No. of multiplications
Normal Convolutional Layer	1,228,800
Depthwise Separable Layer	53,952

TABLE 3.2: Comparision of number of multiplications in normal and depthwise separable layers

layers by a significant amount. This helps us achieve our goal of developing a lightweight yet accurate model.

We also observe a significant decrease in the number of trainable parameters. The regular CNN having 77,025 and the depthwise layers bringing it down to 28,737.

CHAPTER 4

LITERATURE SURVEY

Applying faster R-CNN for object detection on malaria images [1] , utilizes models used for identifying brightfield image of malaria-infected blood. Faster R-CNN model, pretrained on the ImageNet is used on an imbalanced dataset to obtain optimal results. The results are then compared to a baseline model that uses traditional methods. The dataset consists of over 100,000 individual cell specimens. The Faster R-CNN performs better than baseline with an average F1 score of 70% across all cells.

Clustering-based dual deep learning architecture for detecting red blood cells in malaria diagnostic smears [2] The proposal, named RBCNet, is a method to detect red blood cells and their counts. Then, faster R-CNN is used to find small cell objects. The cell detection accuracy provided by the use of RBCNet was more than 97%. The reason why the dual cascade architecture of RBCNet provided better detection of cells is because it was guided by masks generated by the U-Net model.

A Malaria Diagnostic Tool Based on Computer Vision Screening and Visualization of Plasmodium falciparum Candidate Areas in Digitized Blood Smears [3] Manual blood smear analysis is a time-consuming, error-prone, iterative process that relies heavily on skilled personnel. In this study the proposed method is considering only related areas in the sample of blood smear images and then implementing computer vision detection on it. Infected areas are detected using parameters such as size of the object and color of the object. The

model has been trained on digital images from 10 people and validated on 6 images.

A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images [4]. In this study, a three phase approach is employed. The images are corrected for difference in brightness by using a filter that was adaptive in nature. The second phase is that of the erythrocyte recognition phase which is divided into 3 separate steps. In the first step, pixels are classified based on their color. This is then used as a look up table to classify pixels as 'foreground' or 'background'.
Finally, clumped shapes are split. This approach uses the Expectation-Maximization algorithm to find a a contour and an ideal erythrocyte. The final stage is classifying the four potential classes. This classification takes into account the color, texture, brightness and edges. There are two stages in this process, the first process is using a classifier to decide whether it is healthy or not, and then the second stage uses a multi-class classifier to predict the precise class.

Parasite detection and identification for automated thin blood film malaria diagnosis [5). The model is based on a modified K nearest neighbour (KNN) classifier. Three different classification schemes for identification are compared. The conclusion states that detection, and species and lifecycle-stage identification tasks can be performed successfully by a single multi-class classification. It also states that the necessity of seeking a high-level generalization in two assumed categories can be removed. The study implements a total of 8 stages for the detection and classification: Stages 1-5: Preprocessing; Stage 6: Object Extraction; Stage 7: Feature Extraction; Stage 8 : Classification using K nearest neighbor clustering algorithm.

Malaria Cell Detection Using Evolutionary Convolutional Deep Networks [6] uses evolutionary convolutional deep networks. It can work with keras, it automatically generates a good neural architecture. It can be called a sub domain of AutoML. The data used here is by NIH. Around 28000 images of both infected and uninfected cells. Pre processing of the dataset included sample purification, image rescaling and data enhancement. The final accuracy of the model is 99.98 percent. Further work includes testing accuracy of model on different datasets, improving mobility and testing with better network architectures.

CNN-Based Image Analysis for Malaria Diagnosis [7] looks into normal CNN models and transfer learning models. The CNN model is 17 layers and gives an average accuracy of 97.37%. The transfer learning model gives an accuracy of 91.99% on the same dataset. The dataset used here has been acquired from Chittagong Medical College Hospital, Bangladesh. It contains 27578 images of infected and uninfected cell. The dataset is perfectly balanced. The training-testing split used here is 90%-10%. The results indicate that the new CNN model is more accurate than the transfer learning model (by around 7%). The limitation is that accuracy is relatively lower than models in other papers.

In Image Classification of Unlabeled Malaria Parasites in Red Blood Cells [8], HOGs features extracted and classifier trained offline. Viola-Jones object detection is implemented. Model out-performs PCA feature classification by 50% and Hugh transform algorithms by 24%. Accuracy achieved with model is 93%. Limitations are that red blood cells which were shriveled up were not detected. Also, the processing time is very high at 3.3 seconds with a margin of error of 0.4 seconds.

FPGA Implementation of CNN Algorithm for Detecting Malaria Diseased Blood Cells [9] implements FGPA (Field Programmable Gate Array) for CNN. The average computation time is 174 microseconds. Dataset was taken from the US National Library of Medicine site. Testing contained 200 images. 90 diseases, 90 healthy and 20 invalid. Experimental accuracy was 94.76% (189 of the 200 were correctly classified). Uses VHDL language for high efficiency (low energy usage and low use of embedded circuit platform hardware). Limitations are that the dataset used is very small having only 200 images. Also, accuracy is quite low when compared to other models at 94.76%.

Automatic White Blood Cell Detection and Identification Using Convolutional Neural Network [10] uses image segmentation for detection of white blood cells. These cells are then classified using a CNN into 5 different classes. Dataset used for this was attained by medical staff of the Faculty of Medicine at Clinical Medical Center Osijek, Croatia. The dataset contains 412 images. Resolution is 2560x1920. Dataset is extremely unbalanced. The classifications are 'Eosinophils', 'Lymphocytes', 'Monocytes', 'Neutrophils' and 'Unknown'. Modified LeNet-5CNN was used. It is a 7 layer CNN. ReLU is used as the activation function. Accuracy attained was 81.11%. Limitation is that accuracy is low and the dataset used is highly imbalanced.

In A Novel Stacked CNN for Malarial Parasite Detection in Thin Blood Smear Images[20], a stacked CNN architecture has been designed for the binary classification of the cell images. The dataset used contains blood smear images of both infected and uninfected cell images that were procured from 50 uninfected people and 150 infected people. The dataset contains 27, 558 images. Infected and uninfected red blood cell images occur in equal frequency. The BGR image is

converted to YUV in the first step of preprocessing. Subsequently, the image is Normalized to equalize the intensity values. In the final step, the YUV images are converted back into RGB. This model utilizes a 5 fold cross validation. The loss function that is used is cross entropy error as it was ideal for binary classification. The proposed stacked architecture consists of 22 layers. The Rectified Linear Unit (ReLU) is the activation function used. The paper reported accuracy of 99.964%, a precision of 100%, recall of 99.928% and an F1 score of 99.964%. It is compared with other models such as VGG, RESNET, etc. to quantify superior accuracy.

There have been several attempts at solving this problem of detecting malaria cells from a given cell sample. There are 2 main approaches. The first one is segmenting the image to extract the cells first and the second would be to run the CNN directly on the image. However, the first step to both these approaches is heavy pre-processing. As the images are of cell slides we need to get a clear image of the cells present. To achieve this a variety of techniques have been employed. A few of them are passing a low pass filter to adjust the difference in luminescence and even using a pixel classifier to determine whether a pixel belongs to the foreground or background. The first method we found to be used in previous papers was to first run segmentation algorithms on the images. Various methods were deployed for this, a few of them being binary thresholding and specialized mathematical functions. A unique method we found was using an open source module called CellProfiler which identified the cell boundaries of various types of cells from erythrocytes to white and blood cells. These identified cells are then run through a CNN or a KNN algorithm to identify any malaria cells. The second method is to run a classification algorithm directly on the image. The CNNs had several layers with most of them having RELU and

SIGMOID as activation functions. Another method being used was the Faster R-CNN architecture. Overall this method lowered the accuracy as there was a higher presence of false positives. All the work that we have reviewed is focused on developed countries that have access to the premium hardware. These regions are already equipped to combat malaria to their fullest capabilities. As mentioned above, Africa was home to 94% of the total world's cases. Taking in other factors (such as technological infrastructure and Healthcare facilities), it is of paramount importance to ensure that the model to be run is computationally efficient.

We notice that none of the prior work done in the field of malaria cell detection has focused on reducing the computational power required and the time taken for the model to process an image. This is important as Malaria stricken regions often don't have access to the best hardware facilities. Hence, it is important for the model to be light weight and computationally inexpensive. Our work aims to target this gap in the literature. Using Depthwise Separable convolutions, we aim to create a CNN which can run on limited hardware, while also yielding high accuracy.

Title	Dataset	Methodology	Pre-trained models used	Results
Applying faster R-CNN for object detection on malaria images	Dataset of 1300 fields from 1,00,000 individual cell specimens	Stage 1 : run segmentation with open source CellProfiler Stage 2 : RPN and then faster R-CNN	No pre-trained model used	Average F1 score of 70% across all types of malarial cells.
Clustering-based dual deep learning architecture for detecting red blood cells in malaria diagnostic smears	Dataset consisted of 2,00,000 labelled cells	U-Net followed by Faster R-CNN	U-Net	F1 score : 97.76, accuracy of over 97%
A Malaria Diagnostic Tool Based on Computer Vision Screening and Visualization of Plasmodium falciparum Candidate Areas in Digitized Blood Smears	1,00,000 windows of positive and negative cases each.	It is a 2 stepped process. First, a region is segmentation and then the cells are detected Detection. Segmentation was done using a variety of mathematical functions followed by an SVM classifier for classification.	No pre-trained model used	Sensitivity and specificity of 95% and 100%

A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images	Dataset of 12,577 images was used	The image analysis and recognition process include three main components:image preprocessing,recognition of erythrocytes, and classification. A low pass filter was applied to adjust the luminescence and the colors of the sample images.	No pre-trained model used	Accuracy and specificity were 96.2% and 99.9%.
Parasite detection and identification for automated thin blood film malaria diagnosis	336 and 1786 parasite and non-parasite objects were used for the training.	An FLP and BPNN were also implemented.	No pre-trained model used	Accuracy KNN:93.3 FLP:90.1 BNPP:92.2(+-4)

Computing Communications and IoT Applications (ComComAp) 2019 EvolutionaryCNN	Dataset provided by NIH. 27558 images total. Ratio of training to testing is 6:4	Evolutionary CNN using AutoML. Pre processing of dataset included sample purification, image rescaling and data enhancement.	No pre-trained model used	Precision ranges from 0.9986 to 0.9996 Accuracy ranges from 0.9987 to 0.9997 F1 score ranges from 0.9986 to 0.9998 Recall ranges from 0.9986 to 0.9998
IEEE international conference on bioinformatics and biomedicine (BIBM) 2016 MalariaDiagnosis	27578 images. Training testing split was 9:1	Normal CNN + transfer learning. CNN model is 17 layers and gives better accuracy than transfer learning model.Data processing done on Matlab. Increasing local brightness and contrast	AlexNet based on the CIFAR-100 dataset	Normal CNN performs better than transfer learning model by around 7% CNN Model - Accuracy : 97.37%Precision : 97.73%F1 score:97.36% and the Transfer learning model - Accuracy : 91.99% Precision : 95.12% F1 score : 90.24%

38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2016, MalariaParasites	Three datasets were created : infected cells, uninfected cells, regions without a complete cell	HOGs features extracted and classifier trained offline. Viola-Jones object detection is implemented. Split into two stages : Detecting all cell regions, detecting infected cells from regions. Drawback was that red blood cells which were shriveled up were not detected. The processing time is very high at 3.3 seconds with a margin of error of 0.4 seconds.	None (but ViolaJones object detection framework used)	Model outperforms PCA feature classification by 50% and Hugh transform algorithms by 24%. Accuracy achieved with model is 93%
International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2019, FGPA	Dataset was taken from US National Library of Medicine site. Testing contained 200 images:90 diseased,90 healthy and 20 invalid	FGPA (Field Programmable Gate Array) implementation of CNN is used. The average computation time is 174 microseconds. Uses VHDL language for high efficiency. Also, accuracy is quite low when compared to other models at 94.76%	No pre-trained model used	Experimental accuracy was 94.76% (189 of the 200 were correctly classified)

International Conference on Smart Systems and Technologies (SST), 2018, WBC	Dataset used for this was attained by medical staff. Dataset is extremely unbalanced.	In this paper, image is segmented for detection of white blood cells. These cells are then classified using a CNN into 5 different classes. Modified LeNet-5CNN was used. It is a 7 layer CNN. ReLU is used as the activation function.	No pre-trained model used	Accuracy attained was 81.11%
International conference on communication, control, computing and electronics engineering (ICCCCEE) Detection and Classification of Malaria in Thin Blood Slide Images	The dataset consisted of 160 images obtained from the CDC	Pre-processing on these images, through grayscale conversion, resulted in a negative presentation and contrast enhancement. The identified RBCs are analyzed and are classified as infected or not infected.The malaria parasite is also classified using normalized cross correlation function.	No pre-trained model used	This model had an accuracy of 94.97%

TABLE 4.1: Literature review

CHAPTER 5

ALGORITHMS FOR PRE-TRAINED MODELS

We implemented two seperate pre-trained models :

- RESNET50 [22]
- INCEPTIONV3 [21]

The results obtained from these models were noted to compare with those generated by the depthwise CNN.

5.1 TRANSFER LEARNING

Transfer learning employs an architecture like ResNet. This is the result of many architectures and extensive hyperparameter tuning based on what these models have already learned. This knowledge is then applied to new tasks/models rather than starting from scratch. This is called transfer learning

Simply put, it's a machine learning technique that reuses a model developed for one task as a starting point for a model for a second task.

Given the sheer computational and time resources required to develop neural network models for these problems, this is a popular approach in deep learning and is used as the starting point for computer vision and natural language processing

5.1.1 APPROACH

- A pre-trained model is selected from available models released by many research institutions on large and challenging datasets.
- The pre-trained model can then act as a starting point for a model. Using all or just parts of the model, the second task of interest can be tackled.
- If needed, the model can be modified or fine-tuned on the data that is available for the problem.

5.2 RESNET50

The full form of ResNet is Residual Network. It is a neural network that is extremely innovative. A research paper on computer vision in 2015 titled 'Deep Residual Learning for Image Recognition gave the introduction to RenNet and its uses.

ResNet [22] has many variants that work with the same concept but with different numbers of layers. The ResNet50 is a variant of the ResNet model with 48 convolution layers plus one MaxPool layer and one average pool layer. This is the widely used ResNet model and we have explored the ResNet 50 architecture in detail. The model has been trained on more than 1,000,000 images which have been sourced from the ImageNet database. Pre-trained networks can categorize images into a thousand object categories. Hence, the network has learned a wide feature representation. The image input size is 224x224.

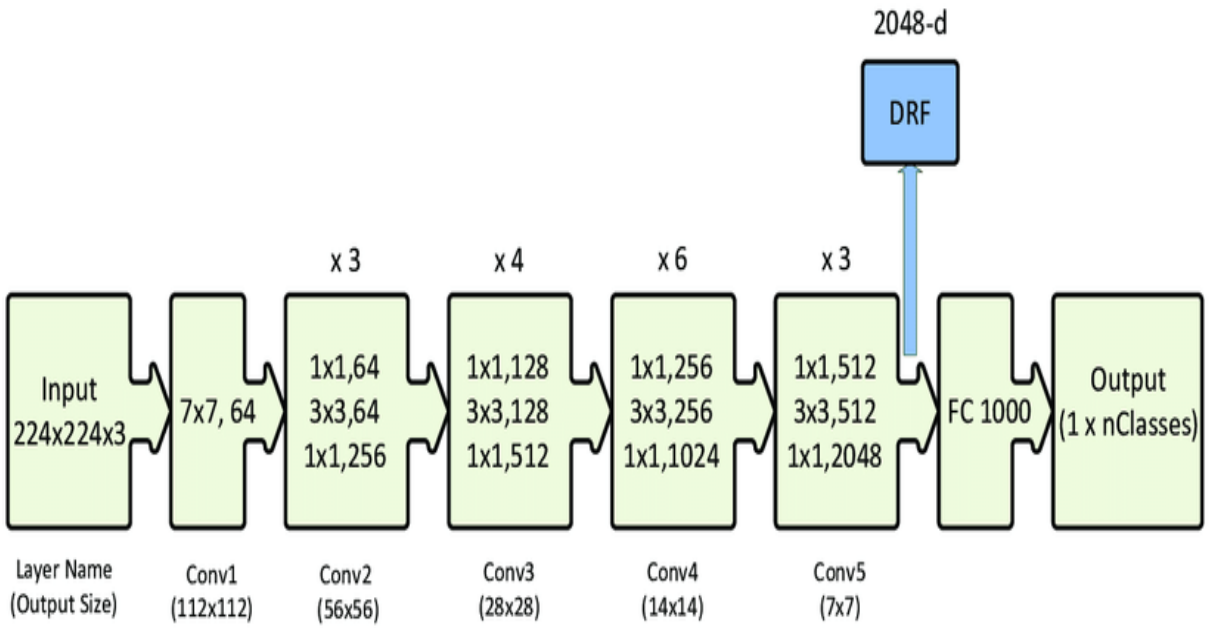

FIGURE 5.1: RESNET50 architecture

5.3 INCEPTIONV3

Inception v3 [21] is a model that was designed for image recognition. It has achieed an accuracy in excess of 78.1% on ImagNet Datasets.

The model consists of many elements such as convolutions, average pooling, maximum pooling, concatenations, dropouts, and fully connected layers. Stack normalization is implemented in the model. The loss value is calculated by Softmax.

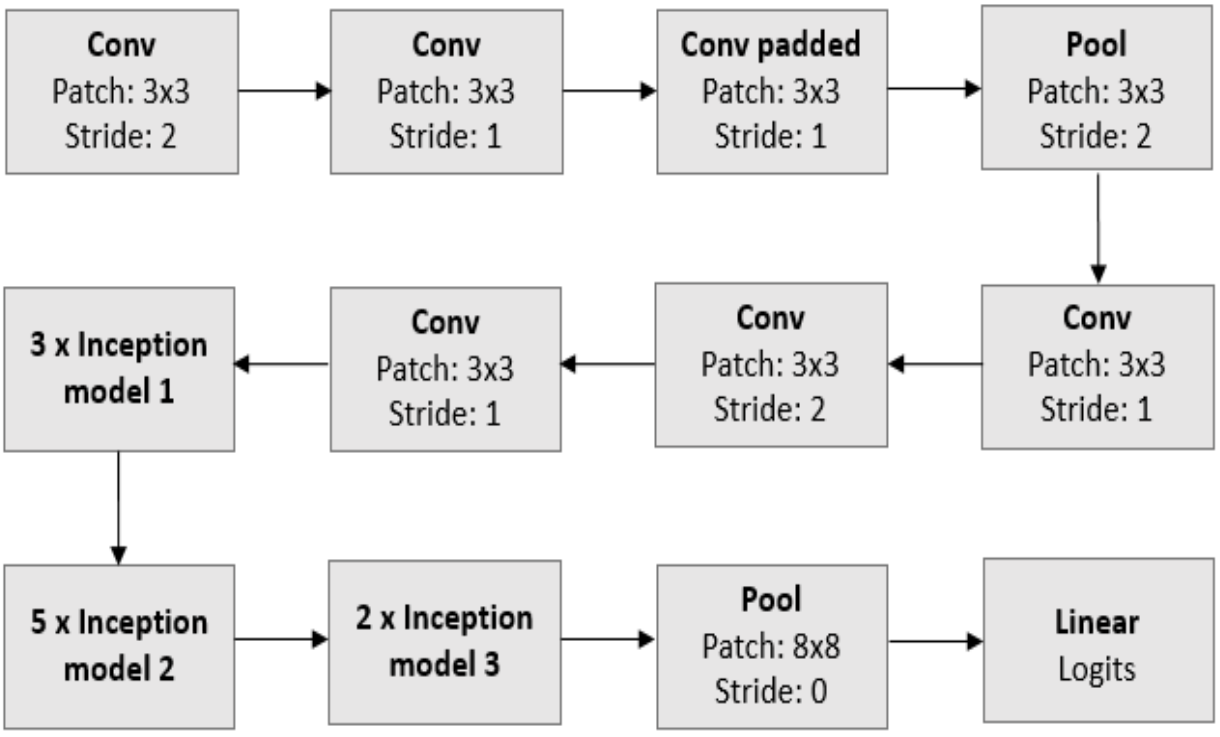

FIGURE 5.2: INCEPTIONV3 architecture

CHAPTER 6

PROPOSED SYSTEM

Our primary goal was to maintain the same level of accuracy as the above-mentioned models as well as keep the model computationally cheap. We achieved this by running a simple CNN on the dataset. The first step consisted of pre-processing our images in particular, converting them to grey-scale and adjudicating the pixels. On these images, we ran a simple CNN to detect the malaria cells. In addition, we used the concept of depthwise separable convolutions in order to achieve our goal.

The architecture implemented was a rather simple one with two separate blocks. The first one being responsible for the pre-procesing of the data and the second one being the CNN itself.

6.1 PRE-PROCESSING

Several pre-processing techniques including grayscale conversion, normalization of pixel values to a predefined range, flipping (both horizontal and vertical), rotation, cropping, shearing, and image standardization techniques were employed. The image pixels were shifted both horizontally and vertically to provide a wider range of images. We also increased and decreased the brightness, contrast and other features of the image to increase the randomness of the dataset.

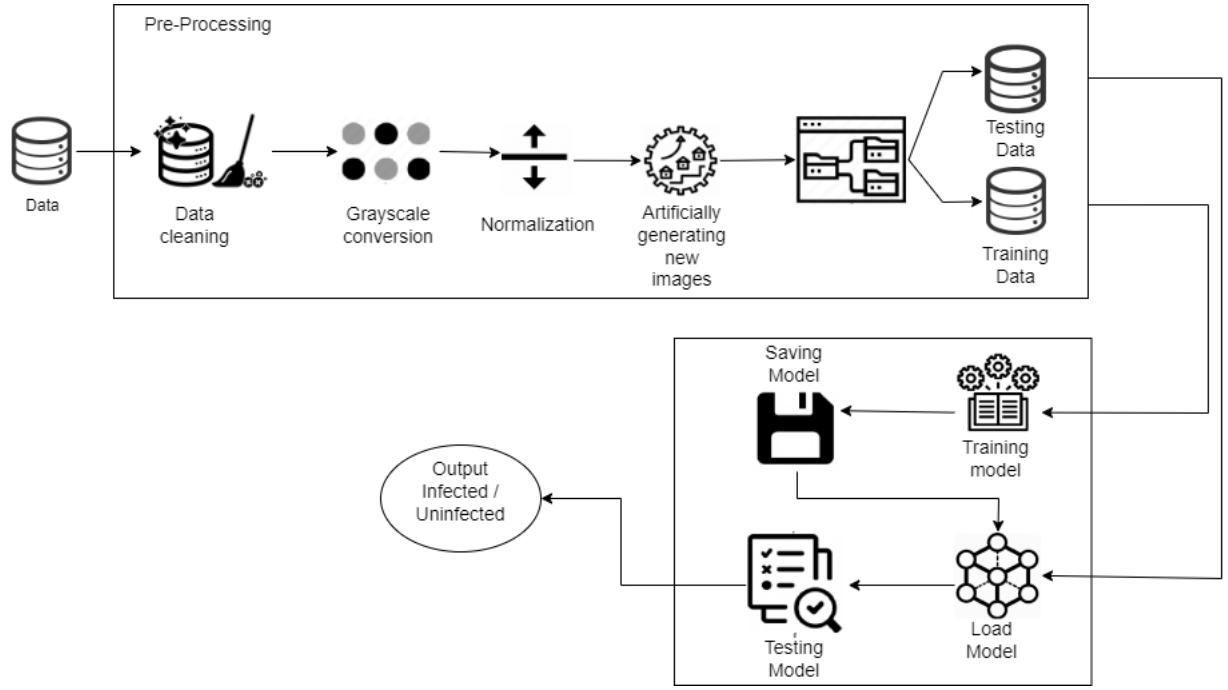

FIGURE 6.1: Architecture diagram

This was done to improve the performance of the models by generating new images similar to the train dataset and adding variety. Increased variety will allow us to train our models to a wider range of images, hence improving its performance.

6.2 TESTING WITH PRE TRAINED MODELS

We implemented the concept of transfer learning as we worked on pre-trained models such as ResNet50, and InceptionV3. We explored the structure of these CNNs and the algorithms behind them. The accuracy, precision, recall and F1 score and the time taken was recorded for every model in order to make a quantitative comparison between the pre-trained models and our depthwise

separable CNN. We also noted down the number of trainable parameters of each of these models. We trained the models on a varying amount of epochs to attain the best performing models. The accuracy flattened out after 30 epochs for all the pre-trained models.

6.3 DESIGNING DEPTHWISE SEPARABLE CNN

We designed our own CNN with the aim to match the accuracy of the pre-trained models while keeping it lightweight and keeping the runtime to a minimum. The purpose of this is so that the model can run on simple and lightweight hardware since the regions that are most affected by malaria are impoverished regions. We used depth-wise separable convolutions in order to achieve this goal. Then we train and test our model on the dataset. We record the accuracy, precision, recall, F1 score and the time taken.

Figure 6.2 shows the architecture of our DS CNN.

6.3.1 TRAINING DEPTHWISE SEPARABLE CNN

While training our DS CNN, we tested with various optimizers such as ADAM, Standard Gradient Descent (SGD), Root Mean Square (RMS). It is noted that the best accuracy was attained with the ADAM optimizer. Hence, we proceed with

the ADAM optimizer.

We trained our model for a varying amount of epochs while recording the accuracy at each epoch. It is noted that our Depthwise Separable CNN reached its maximum accuracy of 0.93 at 950 epochs.

6.3.2 TESTING DEPTHWISE SEPARABLE CNN

We test our Depthwise Separable CNN on the test set and record the Accuracy, Precision, Recall, F1 score and other metrics. We also plot the Precision recall curve and ROC curve. We also calculate the AUC score from the ROC curve. On top of this, we also record the time taken for the model to run. This is our most important metric. Finally, we also note down the number of trainable parameters. It is seen that our Depthwise Separable CNN takes the least amount of time to run compared to all the other models with the least number of trainable parameters.

6.4 COMPARISON WITH STACKED CNN

The results obtained by the Depthwise CNN were compared with those obtained by the normal CNN which had an identical architecture. The normal CNN used normal convolutional layers instead of the Depthwise Separable layers. We record all the essential metrics from the stacked CNN such as Accuracy, Precision, Recall, F1, time taken and number of trainable parameters. It is seen that the Stacked CNN takes longer than the Depthwise Separable CNN to run even

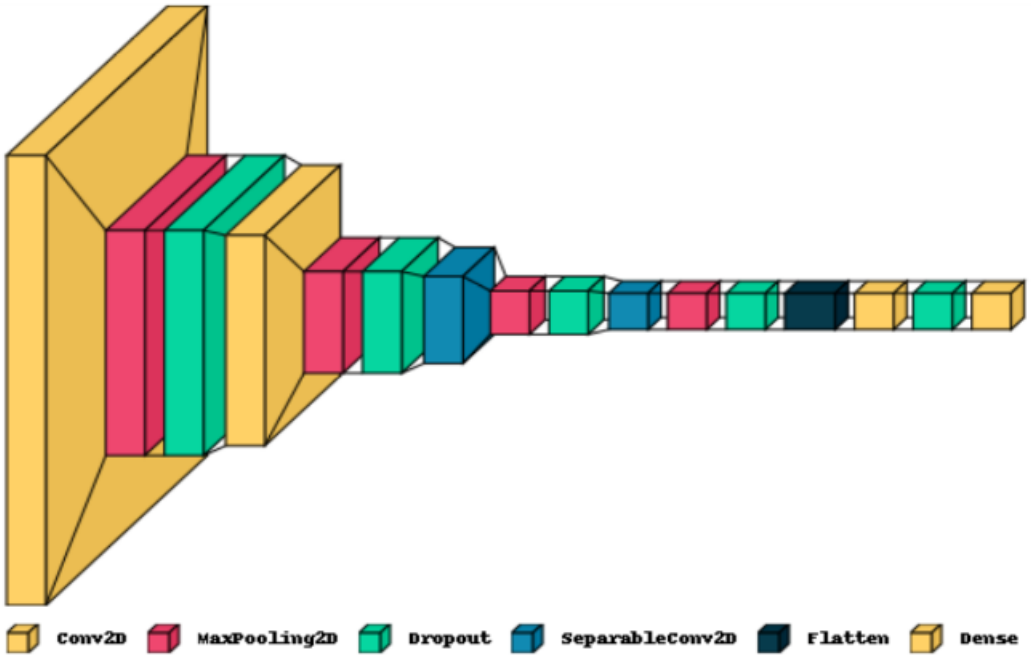

FIGURE 6.2: 3-D representation of layers of DS-CNN

though they have a similar architecture. It is also noted that the stacked CNN has over twice the number of trainable parameters as the depthwise separable CNN.

CHAPTER 7

RESULTS

The aim of this project was to reduce the run time and computational requirements needed to identify a malaria cell from the picture of a cell. This was to be done without compromising on the accuracy, precision, recall, F1-score and other such metrics. The results we obtained during the course of this project have been discussed below.

7.1 Metrics of pre-trained models, Stacked CNN and DS CNN

Table 7.1 shows the Accuracy, Precision, Recall, and F1 score of the 2 pre-trained models (ResNet50 and InceptionV3), the Stacked CNN and also our own Depthwise Separable CNN model. As we can see, our DS CNN performs the best, closely followed by InceptionV3. The Stacked CNN's performance is close to ResNet50's performance. ResNet50 performs the worst compared to all the other models.

Model	Accuracy	Precision	Recall	F1 score
ResNet50	0.90	0.90	0.89	0.89
InceptionV3	0.91	0.91	0.91	0.91
Stacked CNN	0.90	0.90	0.90	0.90
Depthwise Separable CNN	0.93	0.92	0.92	0.92

TABLE 7.1: Performance comparison

7.2 Accuracy vs Epochs

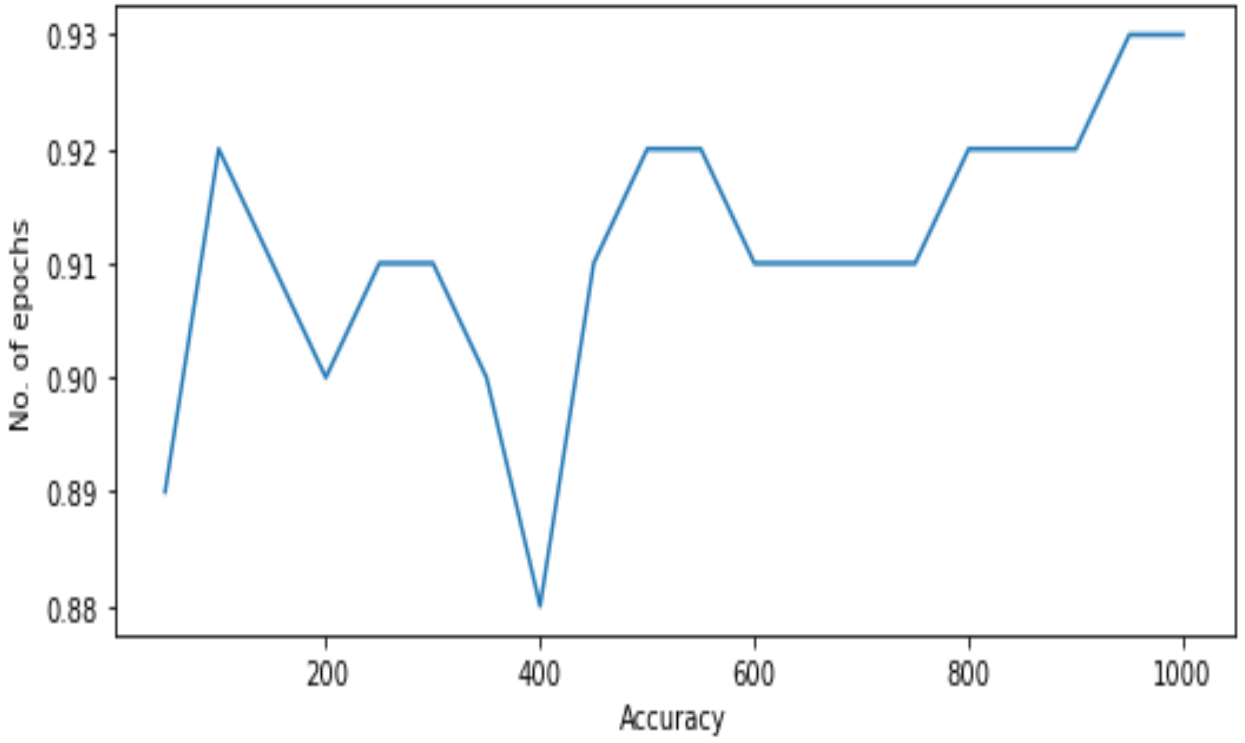

FIGURE 7.1: Accuracy vs Epochs

Figure 7.1 shows the accuracy for the number of epochs the DS CNN model was trained for. It is seen that it reaches its peak accuracy around 950 epochs (0.93).

7.3 Optimizers

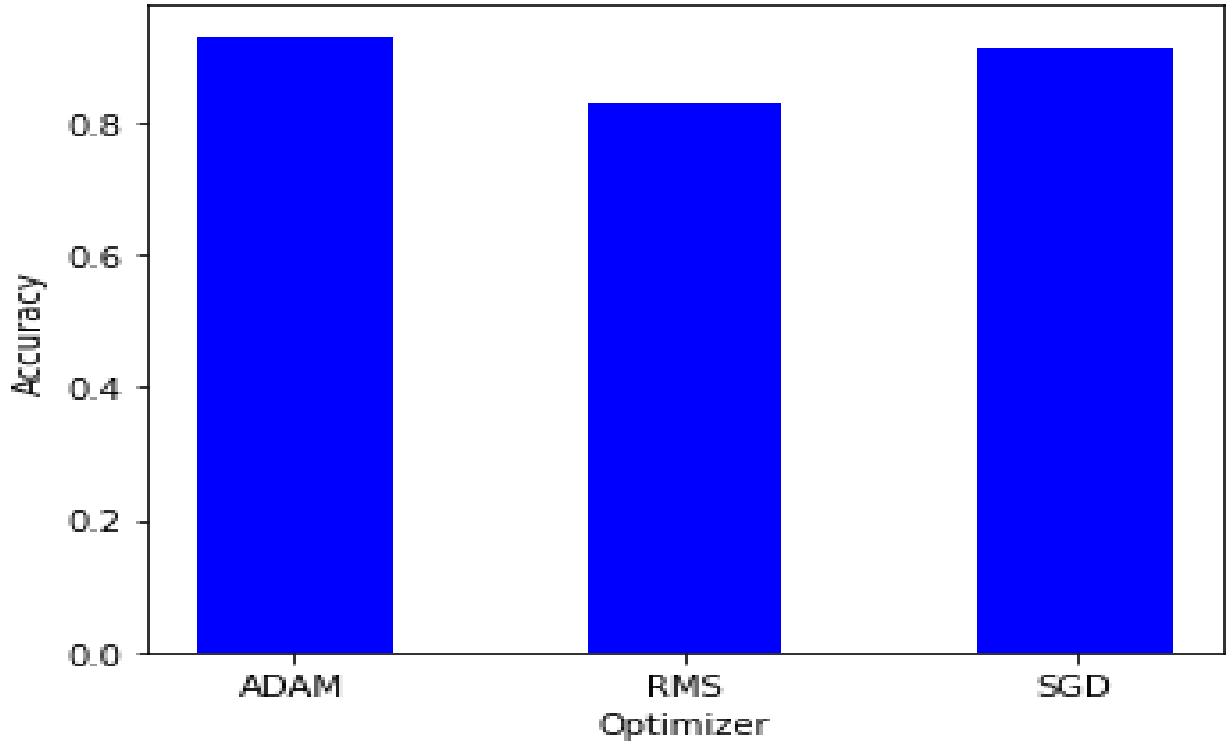

FIGURE 7.2: Accuracy for different optimizers

Figure 7.2 shows the performance of different optimizers. We compare ADAM, RMS and SGD. It is seen that ADAM optimizer performs the best. Closely followed by SGD. RMS performs the worst. Hence, we go forward with the ADAM optimizer.

7.4 Training and Validation metrics

Figures 7.3 and 7.4 show the Training vs Validation for Accuracy and Loss respectively. It is optimal for loss to be minimum.

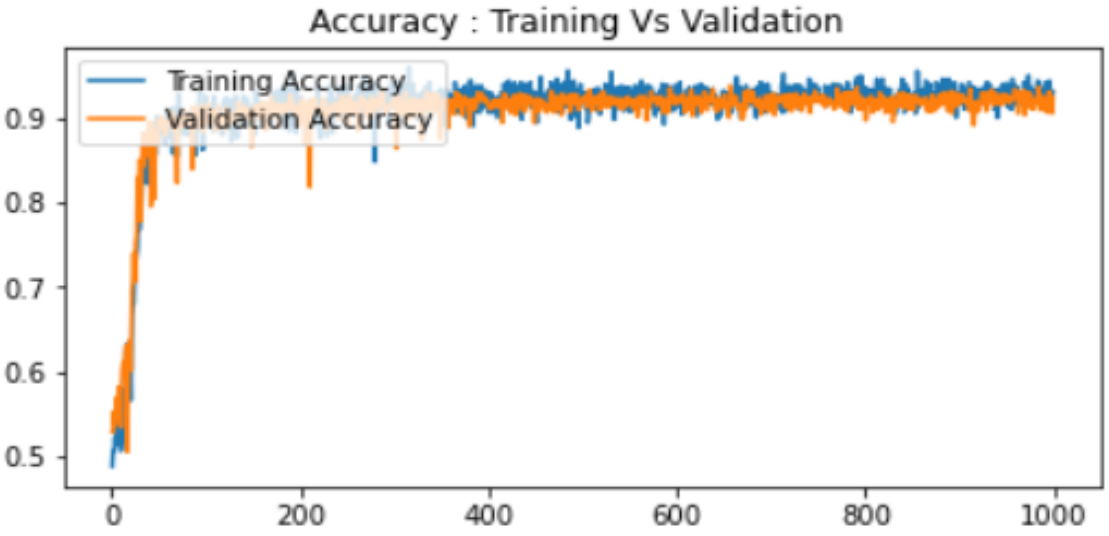

FIGURE 7.3: Training vs Validation Accuracy

FIGURE 7.4: Training vs Validation Loss

7.5 Metrics of DS CNN

Figure 7.5 shows the ROC curve for the DS CNN. The AUC score is calculated from this and is noted to be 0.926.

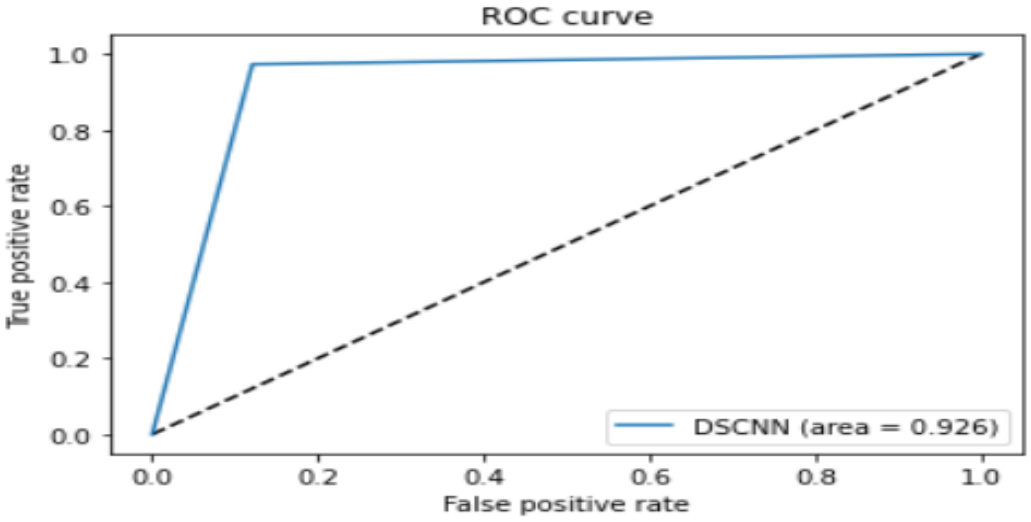

FIGURE 7.5: ROC curve for depthwise CNN

Figure 7.6 shows the precision recall curve for the DS CNN. This shows the tradeoff between precision and recall. High precision relates to a low false positive rate and a high recall rate relates to a low false negative rate. Hence, it is optimal if both recall and precision scores are high.

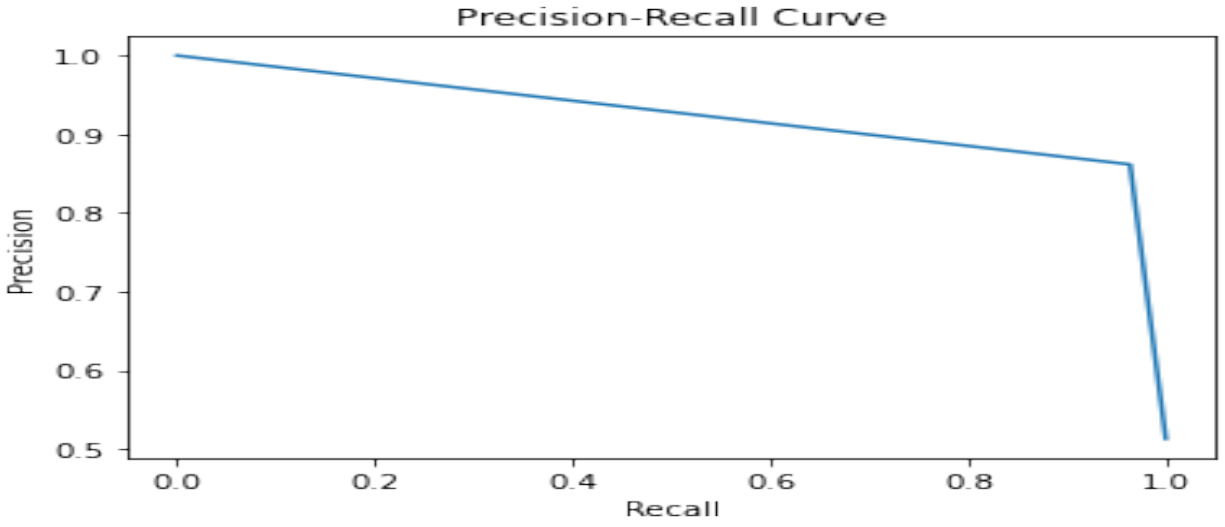

FIGURE 7.6: Precision recall curve for depthwise CNN

Metric	Value
Specificity	0.88755
AUC	0.926

TABLE 7.2: Metrics of depthwise separable model

Table 7.2 shows other metrics of the DS CNN such as Specificity and Area under the ROC curve. The AUC score is calculated from the ROC curve (Figure 7.5).

7.6 Trainable parameters

Table 7.5 shows the number of trainable parameters for the 3 pretrained models, the stacked CNN and our DS CNN. As we can see, the number of trainable parameters is above 20 million for all the pretrained models. The stacked CNN has 77025 trainable parameters. Our DS CNN has the least number of trainable parameters with 28737 trainable parameters. This is an advantage of our DS CNN as it can run on low performance hardware.

Model	No of parameters
ResNet50	27,783,041
InceptionV3	21,073,985
Stacked CNN	77,025
Depthwise Separable CNN	28,737

TABLE 7.3: Comparison of number of trainable parameters

Model	Time Taken
LeNet [24]	5
AlexNet [24]	5
GoogLeNet [24]	19
Depthwise Separable CNN	38.97639
Stacked CNN	76.84733
ResNet50	133.09627
InceptionV3	179.77231

TABLE 7.4: Execution time comparison in seconds

Model	Time Taken
Depthwise Separable CNN	1.414
Stacked CNN	2.788
ResNet50	4.824
LeNet [24]	6.443
AlexNet [24]	6.443
InceptionV3	6.516
GoogLeNet [24]	24.48

TABLE 7.5: Execution time per image comparison in milliseconds

7.7 Time Comparison

Table 7.4 shows the time taken for each model to run on the entire dataset after training. The unit used is seconds.
We have compared our DS CNN with the pre-trained models, Stacked CNN, GoogLeNet [24], LeNet [24] and AlexNet [24]. It is to be noted that the GoogLeNet, LeNet and AlexNet models were run on only 776 images whereas the rest of the models were run on over 27,000 images. InceptionV3 was found to take the most time (180 seconds) whereas our depthwise separable model completed the task in under a minute (39 seconds). All the pretained models took over 2 minutes to run. ResNet50 took the least time among the pre-trained models (133 seconds). The stacked CNN performed much better than the pretrained models but still lags behind our DS CNN (77 seconds).

Table 7.5 shows the time taken for each model to run on a single image from the dataset. The unit used is milliseconds. This is necessary as some models are run on fewer images in Table 7.4. Using 7.5 we can compare the results accordingly.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This work proposes to develop a model that is time efficient and would not compromise on the accuracy of the classification. The depthwise separable CNN was developed and after improvements and fine tuning, an accuracy of 93% was achieved. To establish a frame of reference, 2 pre-trained models were implemented and tested on the same datasets. The accuracy of these models were lower than the custom model. Furthermore, the custom model had far fewer parameters as compared to the pre-trained models; the direct correlation between number of parameters and the time taken to run the model lead to our model outperforming the pre trained models with respect to time taken. Traditional models for this problem did not acknowledge the time taken to run. The time taken by the ResNet50, InceptionV3 and the Stacked CNN are 179, 133 and 76 seconds respectively. Our depthwise separable CNN (DS CNN) runs 4 times than the other models taking just 39 seconds. The models achieve accuracies of 90%, 91% and 90% respectively. Our model achieves an accuracy of 93% which is 2% more than what the other models achieve. Our model would help pathologists even with suboptimal hardware to be able to rapidly and effectively identify infected cells.

There is further scope for this project in multiple aspects. The limitation is that our model only works on cells that have already been segmented, and will not work on entire blood smears. The accuracy, precision and recall of the model, while respectable, could be furthered improved. It should be taken care so as to have little to no effect on the time taken as that should be the primary focus of this

problem statement. The accessibility of this software could also be improved upon by developing a mobile application. This would make it easier for doctors and pathologists to use the model.

REFERENCES

1. Hung, J., Carpenter, A. (2017). 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).

2. Kassim, Y. M., Palaniappan, K., Yang, F., Poostchi, M., Palaniappan, N., Maude, R. J., ... Jaeger, S. (2020). Clustering-based dual deep learning architecture for detecting red blood cells in malaria diagnostic smears. IEEE Journal of Biomedical and Health Informatics, 25(5), 1735-1746.

3. Linder, N., Turkki, R., Walliander, M., Mårtensson, A., Diwan, V., Rahtu, E., ... Lundin, J. (2014). A malaria diagnostic tool based on computer vision screening and visualization of Plasmodium falciparum candidate areas in digitized blood smears. PLoS One, 9(8), e104855.

4. Díaz, G., González, F. A., Romero, E. (2009). A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images. Journal of biomedical informatics, 42(2), 296-307.

5. Tek, F. B., Dempster, A. G., Kale, I. (2010). Parasite detection and identification for automated thin blood film malaria diagnosis. Computer vision and image understanding, 114(1), 21-32.

6. Qin, B., Wu, Y., Wang, Z., Zheng, H. (2019, October). Malaria Cell Detection Using Evolutionary Convolutional Deep Networks. In 2019 Computing, Communications and IoT Applications (ComComAp) (pp. 333-336). IEEE.

7. Liang, Z., Powell, A., Ersoy, I., Poostchi, M., Silamut, K., Palaniappan, K., ... Thoma, G. (2016, December). CNN-based image analysis for malaria

diagnosis. In 2016 IEEE international conference on bioinformatics and biomedicine (BIBM) (pp. 493-496). IEEE.

8. Zhang, Z., Ong, L. S., Fang, K., Matthew, A., Dauwels, J., Dao, M., Asada, H. (2016, August). Image classification of unlabeled malaria parasites in red blood cells. In 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) (pp. 3981-3984). IEEE.

9. Sağlam, S., Tat, F., Bayar, S. (2019, November). Fpga implementation of cnn algorithm for detecting malaria diseased blood cells. In 2019 International Symposium on Advanced Electrical and Communication Technologies (ISAECT) (pp. 1-5). IEEE.

10. Novoselnik, F., Grbić, R., Galić, I., Dorić, F. (2018, October). Automatic white blood cell detection and identification using convolutional neural network. In 2018 International Conference on Smart Systems and Technologies (SST) (pp. 163-167). IEEE.

11. Shah, D., Kawale, K., Shah, M., Randive, S., Mapari, R. (2020, May). Malaria Parasite Detection Using Deep Learning:(Beneficial to humankind). In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 984-988). IEEE.

12. Nayak, S., Kumar, S., Jangid, M. (2019, September). Malaria Detection Using Multiple Deep Learning Approaches. In 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT) (pp. 292-297). IEEE.

13. Mohammed, H. A., Abdelrahman, I. A. M. (2017, January). Detection and classification of malaria in thin blood slide images. In 2017 international conference on communication, control, computing and electronics engineering (ICCCCEE) (pp. 1-5). IEEE.

14. Pattanaik, P. A., Swarnkar, T., Sheet, D. (2017, November). Object detection technique for malaria parasite in thin blood smear images. In 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 2120-2123). IEEE.

15. Peñas, K. E. D., Rivera, P. T., Naval, P. C. (2017, July). Malaria parasite detection and species identification on thin blood smears using a convolutional neural network. In 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE) (pp. 1-6). IEEE.

16. Xia, X., Xu, C., Nan, B. (2017, June). Inception-v3 for flower classification. In 2017 2nd International Conference on Image, Vision and Computing (ICIVC) (pp. 783-787). IEEE.

17. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

18. Wen, L., Li, X., Gao, L. (2020). A transfer convolutional neural network for fault diagnosis based on ResNet-50. Neural Computing Applications, 32(10).

19. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

20. Umer, M., Sadiq, S., Ahmad, M., Ullah, S., Choi, G. S., Mehmood, A. (2020). A novel stacked CNN for malarial parasite detection in thin blood smear images. IEEE Access, 8, 93782-93792.

21. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

22. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

23. Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

24. Dong, Y., Jiang, Z., Shen, H., Pan, W. D., Williams, L. A., Reddy, V. V., ... Bryan, A. W. (2017, February). Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells. In 2017 IEEE EMBS international conference on biomedical health informatics (BHI) (pp. 101-104). IEEE.

25. Sella Veluswami, J. R., Prakash, S., Parekh, N. (2021). Face Mask Detection Using SSDNET and Lightweight Custom CNN.