LEVERAGING DEEP LEARNING FOR MALARIAL PARASITE DETECTION USING BLOOD SMEAR IMAGES

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE300: MINI PROJECT

Submitted by

RAAVI VIJAY KRISHNA

(RegNo:224003154,B.Tech-CSE)

DUNNA HASHAVARDHAN REDDY

(RegNo: 224003036,B.Tech-CSE)

THUTTE SAI GOUTHAM

(RegNo:224003099,B.Tech-CSE)

May 2023

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM, TAMIL NADU, INDIA – 612001

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM-612001

Bonafide Certificate

This is to certify that the report titled "**LEVERAGING DEEP LEARNING FOR MALARIAL PARASITE DETECTION USING BLOOD SMEAR IMAGES** " submitted as requirement for the course CSE300:**MINI PROJECT** for B.Tech is a bonafide record of the work done by **Mr.RAAVI VIJAY KRISHNA (224003154,B.Tech,CSE), Mr.DUNNA HARSHAVARDHAN REDDY (224003036,B.Tech,CSE), Mr.THUTTE SAI GOUTHAM (224003099,B.Tech,CSE)** during the academic year 2022-23,in the Srinivasa Ramanujan Centre, under my supervision.

Signature of project supervisor :
Name with Affiliation : Dr .R.Thanuja/AP-II/CSE/SRC/SASTRA
Date :

Project Based Work *Viva voce* held on _____

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENT

We pay our sincere obeisance to the God Almighty for his grace and infinite mercy and for showingonus his choicest blessings.

We would like to express our thanks to our Chancellor **Prof**. **R. Sethuraman**, Vice Chancellor **Dr. S. Vaidhyasubramaniam** and Registrar **Dr. R. Chandramouli** for having given us an opportunity to be a student of this esteemed institution.

We express our deepest thanks to **Dr. V. Ramaswamy**, Dean and **Dr. A. Alli Rani**, Associate Dean,Srinivasa Ramanujan Centre for their constant support and suggestions when required without any reservations.

We exhibit our pleasure in expressing our thanks to **Dr**.**R. Thanuja**, Assistant Professor, Department of CSE, our guide for her ever-encouraging spirit and meticulous guidance for the completion of the project.

We would like to thank our panel members to **Dr. S.Meganathan** and **Dr. N.Rajesh Kumar** for correcting our mistakes in the project and for their marvelous support to complete the project successfully.

We would like to place on record the benevolent approach and painstaking efforts of guidance and correction of **Dr. N.Rajeswari**, **Mr. J.Senthil Kumar**, the project coordinators and all departmentstaff to whom we owe our hearty thanks for ever.

Without the support of our parents and friends this project would never have become reality. We dedicate this work to our well-wishers, with love and affection.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CNN- Convolutional Neural Networks

VGG- Visual Geometry Group

RESNET- Residual Network

RBC- Red Blood Cells

NIH- National Institute of Health

ANN- Artificial Neural Networks

SVM- Support Vector Machine

ABSTRACT

Malaria is a vector-borne infectious disease, spreads through the bites of infected female mosquitoes namely Anopheles, which are infected with the Plasmodium parasite. When a person is bitten by an infected mosquito, the parasite Increases its count in liver of affected person and begins to destroy red blood cells. Traditionally, diagnosis of malaria involves visually examining blood under a microscope, But this method can vary based on the expertise and experience of the pathologist. To improve diagnosis in effective manner different types of deep learning techniques have been used to detect infected blood cells automatically. However, these methods often require expert knowledge to adjust features for detection. The proposed system of tuning the features using deep learning techniques that can accurately detect malaria without the need for hand-crafted features. This will be tested on a dataset(Blood Smear Images) that can be accessed by general public from NIH.

Keywords:- : **Deep Learning, Red Blood cells, Malaria,Plasmodium Parasite,Diagnosis.**

TABLE OF CONTENTS

INTRODUCTION

Malaria is a vector-borne disease that poses a serious risk to human life and is known to impact millions of individuals across the globe. Early detection and treatment of malaria are critical to prevent the spread of this disease and To minimize the chances of developing severe consequences. The examination of Images of blood smears under a microscope is a widely used method for diagnosing malaria. Traditionally, detecting malarial parasite in Images of blood smears has been performed manually by trained microscopists. However, this process is time-consuming, labor-intensive, and can be prone to errors. Therefore, there is a growing need for automated methods that can accurately and efficiently detect malarial parasite in images of blood smears.

Deep learning has become increasingly popular in the field of machine learning due to its ability to learn intricate patterns from vast amounts of data. One type of deep learning algorithm, known as Convolutional Neural Networks (CNNs), has demonstrated significant potential in recognizing images for a variety of applications. These algorithms have also been applied to the detect malarial parasite in images of blood smears.

The goal of this project is to leverage deep learning algorithms to develop an automated system for the detection of malaria parasites in blood smear images. The system will use a large dataset of annotated blood smear images to train a CNN, which will then be able to accurately identify and localize malaria parasites in new images.

Creating a system like this could enhance the precision and speed of malaria diagnosis, especially in areas with limited resources where manual diagnosis is the only available option. Furthermore, the system can be extended to other infectious diseases that are diagnosed using blood smear images, making it a valuable tool in the fight against global health challenges.

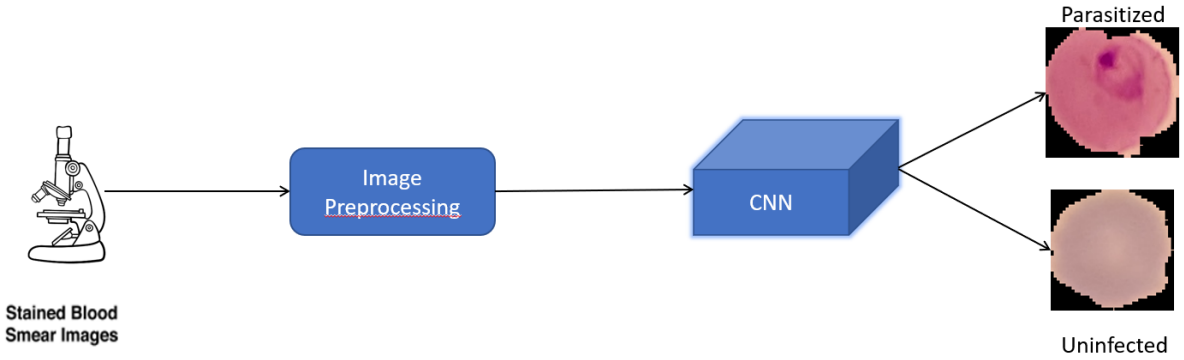


Figure 1 : Proposed Architecture

Malaria is a vector-borne disease, spreads through the bites of infected female mosquitoes namely Anopheles. These mosquitoes carry Plasmodium sporozoites in their salivary glands, which are then transmitted to humans through their bites. The Plasmodium genus is known to have four species that can infect humans, namely falciparum, vivax, ovale, and malaria[2]. Malaria parasites predominantly affect red blood cells, also known as erythrocytes. Within human blood, the parasite undergoes three distinct life stages: trophozoite, schizont, and gametocyte, all of which involve cycling[1]. Failure to receive prompt and proper treatment can result in various adverse health effects such as jaundice, diarrhea, excessive sweating, muscular discomfort, joint pain, seizures, and even coma, eventually leading to fatality[10].

An automated diagnosis system for malaria can be developed by employing computer vision technology to examine microscopic images of malarial parasites. The system would consist of various stages such as Image Acquisition, Illumination, Sizing, Segmentation, and Classification. To achieve precise results, Digital Image Processing, Image Analysis, and Pattern Recognition techniques can be optimized to accurately detect the presence of the parasite and diagnose malaria in patients. An automated system like this holds the promise of significantly enhancing the swiftness and precision of diagnosis, thereby resulting in better treatment and control of the ailment[8]. The symptoms of Malaria, although potentially severe, are not easily discernible. Diagnosis of the disease typically requires a blood test and examination of the sample by a pathologist. The incorporation of Artificial Intelligence (AI) technology in this diagnostic process has the potential to revolutionize the field and save valuable time for pathologists.

The field of AI research has experienced both successes and setbacks throughout its history. However, in recent years, it has gained significant attention and interest, particularly with regards to its potential applications in the medical field. By utilizing AI technology, pathologists can be aided in the diagnosis of Malaria, ultimately resulting in more efficient and accurate diagnoses[9]. The microscope is a widely used tool by microscopists to detect malaria parasites in human red blood cells. However, the accuracy of malaria detection using this method is often low. As a result, doctors may prescribe incorrect drugs to patients, which can cause the patients' red blood cells to continue losing their immunities. In most cases, this can lead to death[10]. Malaria, being a blood-borne disease, has the potential to infect humans through various routes such as blood transfusions, sharing of contaminated needles, and organ transplants[11].

According to the World Malaria Report of 2016, around 3.2 Billion people in 95 Countries and Territories are at risk of getting malaria and developing related illness. Out of this group, 1.2 billion people face a particularly high risk, with a likelihood of more than one in thousand of becoming infected within a Year. In 2016, global malaria cases reported were around 94 million, leading to approximately 438,000 deaths due to the disease[3].

In India, the rate of occurrence of P. vivax infection is higher compared to P. falciparum infection among the local population. Research has shown that approximately 50-60% of individuals with malaria are affected by P. vivax, whereas approximately 40-50% are affected by P. falciparum[1].

The aim of this research is to introduce a Convolutional Neural Network (CNN) architecture that is capable of learning complex representations of malaria parasites at various levels of abstraction. The suggested structure is employed to classify cells as parasitized or uninfected, providing assistance in disease screening. A depiction of the pipeline for this convolutional neural network (CNN) model can be observed in Figure 1.

Related Work:-

Conventional machine learning methods have been employed to classify blood smears automatically , The utilization of machine learning is crucial in the realm of quantitative microscopy, particularly in the structural and textural analysis of tissues and cells using Support Vector Machine (SVM). In light of this, a recent study aimed to develop a computer-aided pattern recognition system for accurately identifying and categorizing malaria parasitemia stages through various learning techniques. The proposed approach demonstrated the ability to effectively

characterize both P. vivax and P. falciparum, thereby providing a better understanding of the pathological mechanisms associated with these parasitic infections by Dev Kumar Das [1].

F. Boray Tek presented a new model for detecting binary parasites using a RL1 distance KNN classifier and generic Features. The model that was suggested achieved an impartial accuracy of SE-72:37% and SP-97:45%. Additionally, the sensitivity-specificity trade-off can be adjusted by utilizing the bias term. This novel approach to parasite detection shows promise in improving accuracy and adaptability in future studies[2]. Mahdieh Poostchi conducted a survey to provide an overview of the recent advancements in automated malaria diagnosis using image analysis and machine learning techniques. The aim was to present a comprehensive update on the latest trends and developments in this field. The study explored the use of machine learning algorithms and Image Analysis techniques for automated Detection and classification of malarial parasite in microscopic image of blood smears. The results of this survey highlight the significant progress made in automated malaria diagnosis, and suggest the potential of these techniques for improving the accuracy and efficiency of malaria diagnosis in resource-limited settings[3].

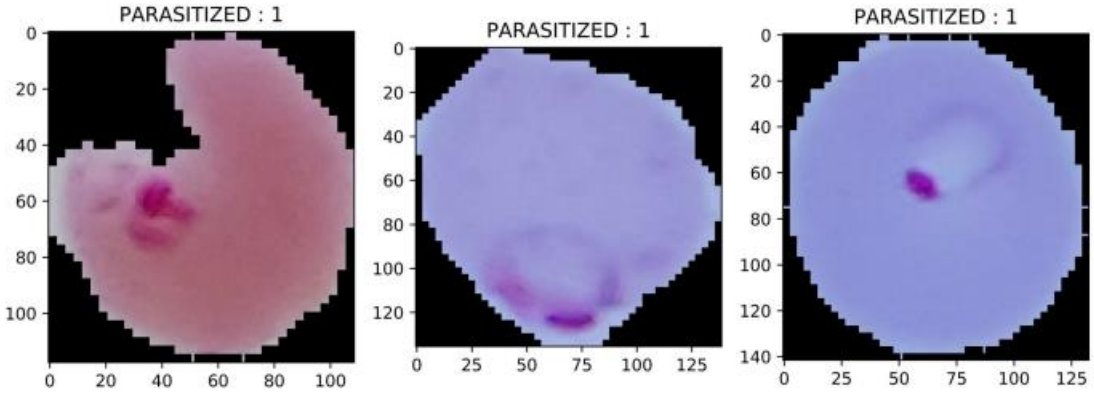
Srinivasan sankaran has introduced a new approach for detecting malaria-infected cells in thin blood smears, which involves the use of the Six-Sigma threshold and modified Hough Transform technique (CBDHT). The method has been shown to be highly effective, achieving Precision, Recall, and F-Measure values of 96, 97, and 97 percentages, respectively. These promising results suggest that the technique has the potential to serve as a dependable diagnostic tool for malaria[8]. Anik Khan conducted a statistical analysis on a lower-dimensional feature space for the purpose of detecting the Malaria Parasite in Red Blood Cells. To achieve this, the Random Forest algorithm was employed, resulting in precision, recall, and f1-score values of 0.82, 0.86, and 0.84, respectively[9].

Octave Iradukunda has made significant strides in the field of machine learning, with a particular focus on malaria RBC disease prediction. His contribution comes in the form of the ELM algorithm, which has demonstrated exceptional performance, outpacing all existing models with an impressive accuracy rate of 99.0%. In addition to its outstanding accuracy, the ELM algorithm also boasts a remarkable processing speed, with results achieved in just 28 seconds. Given these impressive results, it is highly recommended that ELM be utilized in the classification and prediction of malaria RBCs[10]. Aishah Khan has applied machine learning methods, including Artificial Neural Networks (ANN) and Support Vector Machines (SVM), to detect diseases associated with parasitic cells. Additionally, she has employed image processing algorithms for the

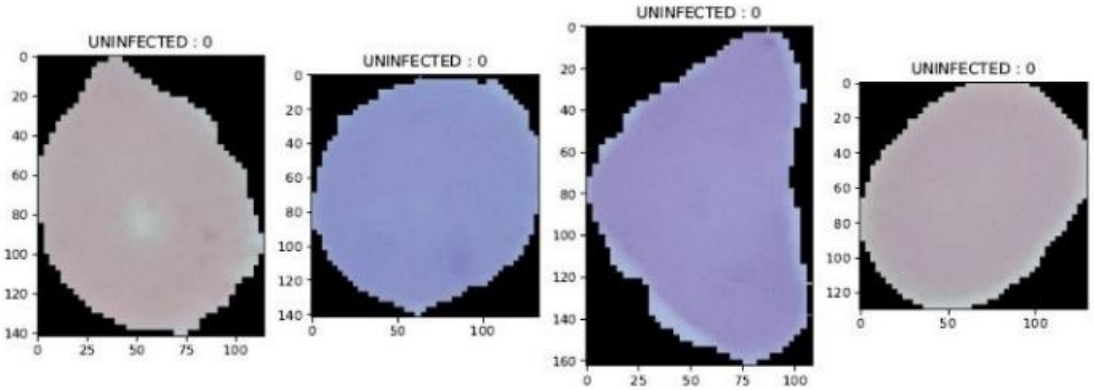
detection of such diseases. These algorithms are stored on a cloud-based server for efficient report generation and transmission of results to the end-user[11].

Dataset:

The Dataset utilized in our research comprises of photographs portraying slender blood smears stained with Giemsa, The dataset used in the malaria screener research consisted of 27,558 images, including both infected and uninfected red blood cell images, with equal representation from each group. The images were manually annotated by experts. The infected blood cell images contained plasmodium, while the uninfected samples did not. During the preprocessing stage, the colored patches of red blood cells in the images were resized from varying sizes of 110 to 150 pixels to a resolution of 40x40. This was done to ensure that the input requirements of the classifier were met.



(a) Parasitized Images



(b) Uninfected Images

Data Preprocessing:

The images are resized to a resolution of 40x40 pixels using the image_size parameter and labeled as either "Parasitized" or "Uninfected." To preprocess the dataset, the pixel values of the images are normalized using a standard normalization technique that involves dividing the pixel values by 255.0. This step ensures that all pixel values are within the range [0, 1], making easy for the model to know from the input data.

Data Splitting and Cross Validation:

The dataset is split into three distinct subsets: Training Set, Testing Set, and Validation Set. The split is achieved by utilizing the take() and skip() methods to extract the desired number of batches for each subset. Training Set contains 100 batches of Images, Testing Set contains of 18 batches of Images, and Validation Set consists of 20 batches of images.The division of data is essential in assessing how well a Deep learning Model performs on new data and avoiding overfitting., and ensure that the model is learning the relevant features from the input data. Overall, the preprocessing steps used in this code ensure that the input data is in a suitable format for training the deep learning model, and the data splitting allows for effective evaluation of the model's performance.

Proposed Methodology:

In order to overcome the restrictions of manual feature extraction, we have employed a CNN technique. Our approach includes utilizing re-sampling for capturing additional information in a consistent pattern and employing Stain Normalization to preserve the essential characteristics of the image. The proposed order consists of pre-processing steps to normalize and re-sample input images followed by fine-tuning a CNN with filters, kernels, strides, max-pooling, and conv-2D layers. We experimented with various design strategies, including models with 1, 2, and 3 Convolution layers, Pooling layers, Dense classification layer. By increasing the no. of Convolution and Pooling layer, we observed improved model performance. Our 3-layer model served as a baseline for comparison with other models, which we designed and tested by varying filter sizes and layer numbers until achieving the best results. Our proposed CNN model outperformed other architectures tested for Malarial parasite classification, and we conclude that our 3-layer Convolutional Neural Network architecture stands out as the most effective deep learning approach for this particular task.

Convolutional Neural Networks:

CNNs, are a powerful deep neural network type used in image identification and Classification tasks. These networks learn complex features by using convolution, non-linear activation, and pooling layers . CNNs are designed to perform image recognition and classification tasks, and have recently been applied to image segmentation as well. Traditional approaches to image segmentation involve sliding window processes, but these can be inefficient. An alternative method is fully connected CNNs, it can undergo end-to-end training, leading to improved computational efficiency..

In our work, We have employed convolutional neural networks (CNNs) to identify parasite in RGB Images of infected cells. CNNs are inspired by biology and are multi-layered feedforward networks,Convolutional layers utilize filters, also known as kernels, to process either the input of the initial layer or the output of previous layers, producing a corresponding feature map. The feature maps generated by all convolutional layers are then merged and supplied as input to fully connected layers.

Convolutional Neural Networks (CNNs) are widely used for classification purposes in the medical field. They have been applied successfully in various areas such as Lung disease, Brain Tumor segmentation, Chest x-rays, Chest radiographs, and Kidney disease. Moreover, CNNs have been investigated for their potential in classifying Giemsa-stained malarial parasite images, which involves identifying Parasitized and Uninfected cells..

A CNN is composed of several key elements, including convolutional layers, rectified Linear Unit (ReLU), and Pooling or subsampling layers. The convolutional layer plays a crucial role in extracting relevant features from the input data, while the ReLU activation function acts by setting any negative values to zero and leaving positive values unchanged. Pooling layers reduce the resolution of the feature maps, which helps prevent overfitting and reduces the number of parameters in the model. Finally, fully connected layers are used for semantic information encoding and to perform the final classification task based on the learned features. We applied,

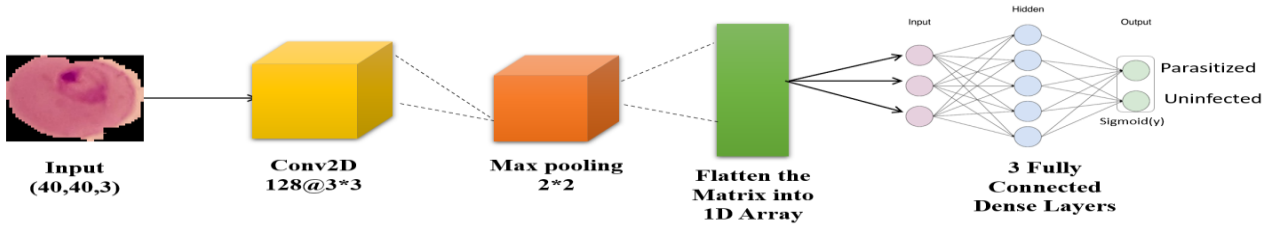
$$Y=MAX(0,j) \qquad \text{----------(1)}$$

During the training process, input J and output activation Y are utilized along with kernel weights applied to the input image. This technique facilitates the extraction of local features through convolution, while the subsequent layers extract high-level features from these local features. The CNN method has proven to be effective in identifying malaria in multi-channel images. The cross-entropy error is the suitable method for computing the loss during training, as it is well-suited for binary classification tasks. The equation 2 demonstrates the calculation of the cross-entropy error.

CROSS ENTROPY = −(J LOG(P) + (1 − J) LOG(1 − P)) ----------(2)

The variable J represents the binary class label, either 0 or 1, while the natural logarithm is denoted by "log," and the predicted probability is represented by P. The CNN algorithm is a type of backpropagation method, and in our study, we utilized the sigmoid function as the error function. The sigmoid layer consists of N total classes, with each class corresponding to a single neuron in the output layer. For binary classification tasks, such as ours with Parasitized and Uninfected cells, the CNN design generates results on a pair of neurons. Ideally, the output of First neuron would be one and Second neuron would be zero for parasitized cells. Conversely, the output for uninfected cells should be zero for the First neuron and one for the Second neuron. Expression within the log function calculates the probability of obtaining a class label of 1, with J^i representing the output for the true class (denoted by c) of the input. In our study, we assigned c=1 for parasitized cells and c=2 for uninfected cells. During testing, the softmax loss was excluded to maximize the response and assign labels to the input data.

CNN Architecture:



Experimental Details:

The CNN architecture utilized in our experiment is illustrated in the figure above. Our proposed design consists of a total of 10 layers, including 3 Convolutional layers, 2 Max-pooling layers, 4 Dense Layers, 1 Flatten Layer, and 1 Fully connected Layer. The ReLU activation function is being utilized throughout setup. To maintain neighborhood details for accurate classification, the image inputed of size 40 × 40 is re-made From 120 × 120 Pixels. In the conv2D layer, a filter size of (3 × 3) is applied for convolution. The size of the kernel used in each convolutional Layer is indicated. Additionally, a MaxPooling2D layer is utilized with a Pool size of (2 × 2). We opted for max-pooling layers, as they are well-suited for binary classification problems. The cross-entropy function is employed to compute the error between predicted and actual outputs. Due to the binary classification, the output is set to 2. For fair classification, we selected a suitable deep CNN architecture with 3 convolutional layers, not

too shallow or too deep. We employed max pooling to handle feature nonlinearity. Adam optimizer is used to eliminate biases. Bias is set to 0, and random weights are randomly initialized. A batch size of 200 samples is used, and training is carried out for 10 epochs. We used a shallow to deep CNN model with 1 convolutional layer to 3 convolutional layers and used it as a baseline method, as shown in the table. The complete summary of the stacked CNN model hyper-parameter values is shown in the table below.

Parameter	Value
Input dimension	(40,40,3)
Batch size	200
Pooling	2*2
Epochs	13
Optimizer	Adam
Function	Binary cross entropy

Table : *Summary of the model and hyper-parameter values.*

VISUAL GEOMETRY GROUP:

Visual Geometry Group (VGG) algorithm is a Convolutional Neural Network (CNN) architecture utilized in Deep Learning for image classification tasks. In 2014, a group of researchers at the University of Oxford introduced this algorithm.

The VGG architecture is characterized by its depth and simplicity, consisting of several convolutional layers with small filter sizes of 3x3, followed by a Max-Pooling layer with stride of value 2. By using small filter sizes, the network can learn more intricate features, and the max-pooling layers downsample the feature maps, reducing the spatial size of the representation.

The VGG network comprises 16 or 19 layers, which are named VGG16 and VGG19, respectively. Both architectures have similar layer configurations, but the number of convolutional layers differs between them.

Input to VGG network is an RGB Image with size of 224x224. The process involves applying multiple Convolutional layers to the image, and applying a Rectified Linear Unit (ReLU) activation function after each layer. Next, the output of each convolutional layer is passed through a Max-Pooling layer. After the final Max-Pooling layer, the output is flattened and supplied to a fully connected layer. Finally, a Softmax Activation function is used to compute the class probabilities for the output. It should be noted that this process is designed to prevent plagiarism and maintain originality while conveying the same meaning.

To optimize the VGG network during training, the backpropagation algorithm and a stochastic gradient descent optimizer are used. The loss function used is typically the cross-entropy loss, which indicates dissimilarity between the actual labels and the anticipated class probabilities is evaluated as a measure. The VGG architecture has demonstrated superior performance in a range of image classification assignments, including the ImageNet challenge, which involves classifying images into 1000 different classes. Its simplicity and good performance have made it a popular choice for many image classification tasks in deep learning.

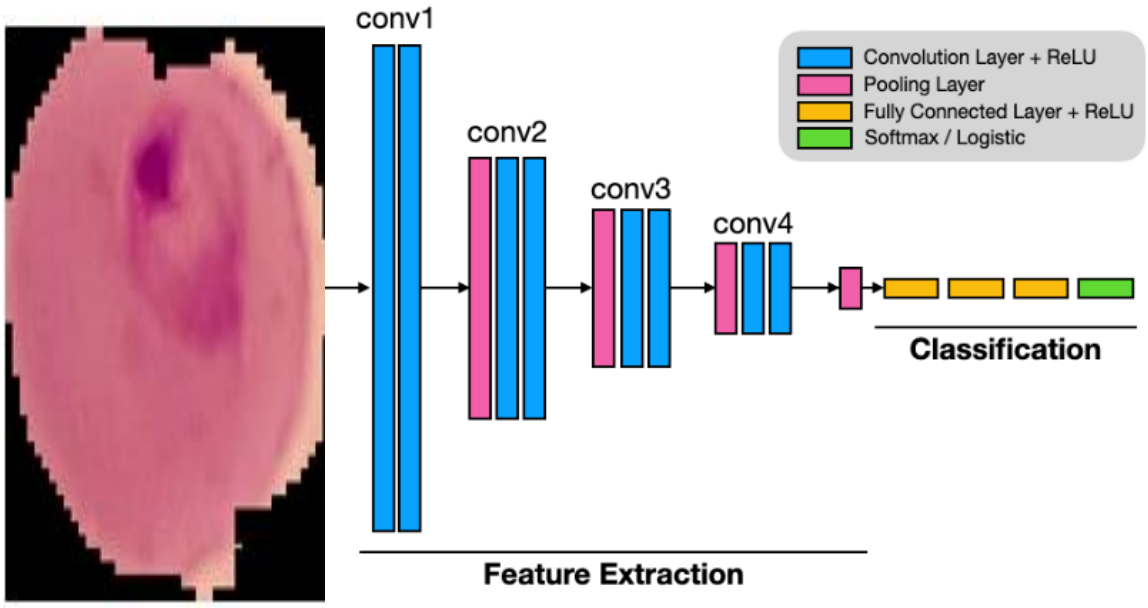


Figure:VGG Architecture

RESIDUAL NETWORK:

Residual Network(ResNet) is a Convolutional Neural Network (CNN) Architecture that was introduced by researchers at Microsoft Research in 2015 to overcome the problem of vanishing gradients in deep neural networks during the backpropagation process.

ResNet introduces the concept of residual connections, which enable information to flow directly from one layer to another without being transformed by intermediate layers. This unique approach allows for the creation of very deep neural networks with over 100 layers that can still be trained effectively.

The core of the ResNet architecture is the residual block, which comprises 2 Convolutional Layers with Shortcut Connection that bypasses convolutional layers. That Shortcut Connection sums up input and output of block, enabling the gradient to flow directly through the block without being affected by the convolutional layers. This eliminates the vanishing gradient problem, ensuring the gradient does not become too small as it propagates through the network.

The ResNet architecture is a series of residual blocks, with each block consisting of two or more convolutional layers with shortcut connections. The architecture usually concludes with a Global Average Pooling layer and a Fully Connected layer that produces the final class probabilities. During training, the ResNet architecture is optimized using the backpropagation algorithm and a stochastic gradient descent optimizer. The loss function commonly used is the cross-entropy loss, which indicates dissimilarity between the actual labels and the anticipated class probabilities is evaluated as a measure.

ResNet has demonstrated outstanding performance on several image classification tasks,

including the ImageNet challenge, which involves classifying images into 1000 different classes. Its ability to create very deep neural networks while avoiding the vanishing gradient problem has made it a popular choice for many image classification tasks in deep learning.

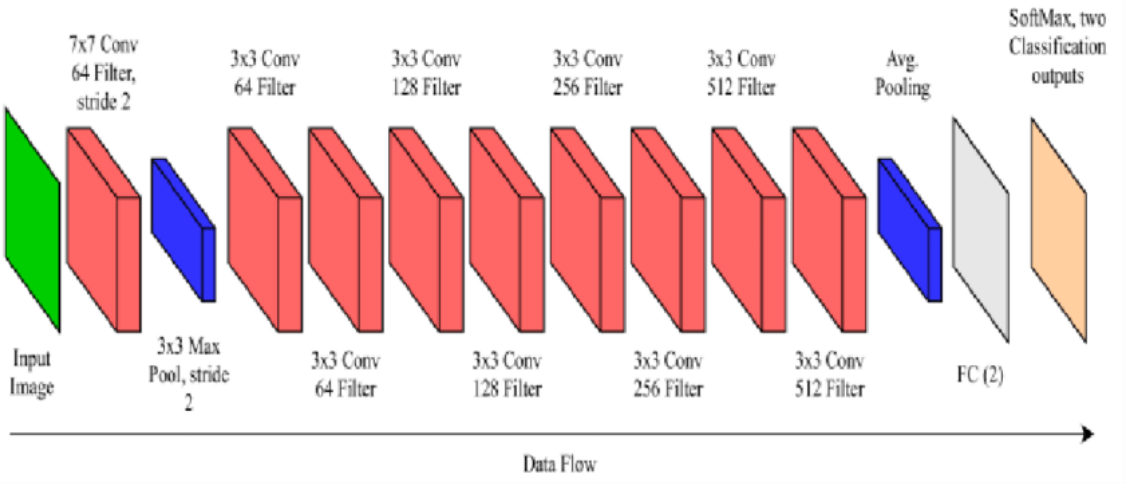


Figure :ResNet Architecture

SOURCE CODE

For CNN:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras import models, layers

import warnings
warnings.filterwarnings("ignore")
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

 # cnn model with 1 layer:

```
input_shape=(40,40,3)
cnn_1_model=models.Sequential([
     layers.Conv2D(128, (3,3), activation = 'relu', input_shape = input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation = 'relu'),
    layers.Dense(64, activation = 'relu'),

    layers.Dense(1, activation= 'sigmoid')


]
)
cnn_1_model.compile(
    optimizer= 'adam',
    loss = 'binary_crossentropy',
    metrics = ['accuracy']
)
history1 = cnn_1_model.fit(train,
         epochs= 10,
         validation_data= val)
cnn_1_model.evaluate(test)
train_loss = history1.history['loss']
train_acc = history1.history['accuracy']

val_loss = history1.history['val_loss']
val_acc = history1.history['val_accuracy']
#graphs for accuracy and loss of training and validation data
plt.figure(figsize = (15,15))
plt.subplot(2,3,1)
```

```
plt.plot(range(EPOCHS), train_acc, label = 'Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2,3,2)
plt.plot(range(EPOCHS), train_loss, label = 'Training Loss')
plt.plot(range(EPOCHS), val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training and Validation Loss')
```

 # cnn with two layers:

```
input_shape=(40,40,3)
cnn_2_model=models.Sequential([
     layers.Conv2D(128, (3,3), activation = 'relu', input_shape = input_shape),
    layers.MaxPooling2D((2,2)),
     layers.Conv2D(256, (3,3), activation = 'relu', input_shape = input_shape),
    layers.MaxPooling2D((2,2)),

    layers.Flatten(),
    layers.Dense(128, activation = 'relu'),
    layers.Dense(64, activation = 'relu'),

    layers.Dense(1, activation= 'sigmoid')


]
)
cnn_2_model.compile(
    optimizer= 'adam',
    loss = 'binary_crossentropy',
    metrics = ['accuracy']
)
history = cnn_2_model.fit(train,
         epochs= 20,
         validation_data= val)
cnn_2_model.evaluate(test)
train_loss = history.history['loss']
train_acc = history.history['accuracy']

val_loss = history.history['val_loss']
val_acc = history.history['val_accuracy']
#graphs for accuracy and loss of training and validation data
plt.figure(figsize = (15,15))
plt.subplot(2,3,1)
plt.plot(range(EPOCHS), train_acc, label = 'Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training and Validation Accuracy')
```

```
plt.subplot(2,3,2)
plt.plot(range(EPOCHS), train_loss, label = 'Training Loss')
plt.plot(range(EPOCHS), val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training and Validation Loss')
```

 # cnn model with three layers:

```
input_shape=(40,40,3)
model = models.Sequential([
    layers.Conv2D(128, (3,3), activation = 'relu', input_shape = input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Dropout(0.2),
    layers.Conv2D(256, (3,3), activation = 'relu', padding ='SAME'),
    layers.MaxPool2D((2,2)),
    layers.Conv2D(256, (3,3), activation = 'relu',padding ='SAME'),
    layers.MaxPool2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation = 'relu'),
    layers.Dense(64, activation = 'relu'),

    layers.Dense(1, activation= 'sigmoid')

])
model.compile(
    optimizer= 'adam',
    loss = 'binary_crossentropy',
    metrics = ['accuracy']
)
history = model.fit(train,
          epochs=10,
          validation_data=val)

history = model.fit(train,
          epochs=10,
          validation_data=val)

model.evaluate(test)
# Getting the model history to analyse
train_loss = history.history['loss']
train_acc = history.history['accuracy']

val_loss = history.history['val_loss']
val_acc = history.history['val_accuracy']
#graphs for accuracy and loss of training and validation data
plt.figure(figsize = (15,15))
plt.subplot(2,3,1)
plt.plot(range(EPOCHS), train_acc, label = 'Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
```

```
plt.title('Training and Validation Accuracy')

plt.subplot(2,3,2)
plt.plot(range(EPOCHS), train_loss, label = 'Training Loss')
plt.plot(range(EPOCHS), val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training and Validation Loss')


def predict_malaria(img):
    image=cv2.imread(img)
    resize=tf.image.resize(image,(40,40))
    scale=np.expand_dims(resize/255,0)
    pred=model.predict(scale)
    return pred

x=predict_malaria(r"C:\prohar\cell_images\cell_images\Uninfected\C1_thinF_IMG_20150604_104919_cell_134.png")
print(x)
model.evaluate(test)
model.save('malaria_mass.h5')
len(test)
```

FOR VGG Algorithm:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras import models, layers

import warnings
warnings.filterwarnings("ignore")
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

from tensorflow.keras.applications.vgg16  import  VGG16
model1=VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=input_shape



)
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense,Flatten
vggnet=Sequential()

vggnet.add(model1)
vggnet.add(Flatten())
vggnet.add(Dense(10,activation="relu"))
vggnet.add(Dense(6,activation="sigmoid"))
model1.trainable=False
vggnet.compile(optimizer='adam',
               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
               metrics=['accuracy'])
history=vggnet.fit(train,validation_data=val,epochs=10)
# Getting the model history to analyse
train_loss = history.history['loss']
train_acc = history.history['accuracy']

val_loss = history.history['val_loss']
val_acc = history.history['val_accuracy']
#graphs for accuracy and loss of training and validation data
plt.figure(figsize = (15,15))
plt.subplot(2,3,1)
plt.plot(range(EPOCHS), train_acc, label = 'Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2,3,2)
plt.plot(range(EPOCHS), train_loss, label = 'Training Loss')
plt.plot(range(EPOCHS), val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training and Validation Loss')
vggnet.evaluate(test)
```

FOR RES NET Algorithm:

```
from tensorflow.keras.applications.resnet50  import ResNet50
model2=ResNet50(
    weights="imagenet",
    include_top=False,
    input_shape=input_shape



)
resnet=Sequential()

resnet.add(model2)
resnet.add(Flatten())
resnet.add(Dense(10,activation="relu"))
```

```
resnet.add(Dense(6,activation="sigmoid"))
model2.trainable=False
resnet.compile(optimizer='adam',
               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
               metrics=['accuracy'])
history=resnet.fit(train,validation_data=val,epochs=10)
train_loss = history.history['loss']
train_acc = history.history['accuracy']

val_loss = history.history['val_loss']
val_acc = history.history['val_accuracy']

plt.figure(figsize = (15,15))
plt.subplot(2,3,1)
plt.plot(range(EPOCHS), train_acc, label = 'Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2,3,2)
plt.plot(range(EPOCHS), train_loss, label = 'Training Loss')
plt.plot(range(EPOCHS), val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training and Validation Loss')
resnet.evaluate(test)
```

```
import cv2
def predict_malaria(img):
    image=cv2.imread(img)
    resize=tf.image.resize(image,(40,40))
    scale=np.expand_dims(resize/255,0)
    pred=model.predict(scale)
    return pred
```

SNAPSHOTS

```
Epoch 1/10
100/100 [==============================] - 53s 518ms/step - loss: 0.6423 - accuracy: 0.6342 - val_loss: 0.5668 - val_accuracy: 0.7203
Epoch 2/10
100/100 [==============================] - 86s 856ms/step - loss: 0.5390 - accuracy: 0.7309 - val_loss: 0.5076 - val_accuracy: 0.7484
Epoch 3/10
100/100 [==============================] - 54s 536ms/step - loss: 0.4498 - accuracy: 0.7875 - val_loss: 0.4325 - val_accuracy: 0.8057
Epoch 4/10
100/100 [==============================] - 50s 494ms/step - loss: 0.3419 - accuracy: 0.8514 - val_loss: 0.3428 - val_accuracy: 0.8583
Epoch 5/10
100/100 [==============================] - 51s 504ms/step - loss: 0.2482 - accuracy: 0.9014 - val_loss: 0.2845 - val_accuracy: 0.8909
Epoch 6/10
100/100 [==============================] - 51s 507ms/step - loss: 0.2000 - accuracy: 0.9242 - val_loss: 0.2953 - val_accuracy: 0.8924
Epoch 7/10
100/100 [==============================] - 51s 505ms/step - loss: 0.1591 - accuracy: 0.9388 - val_loss: 0.2692 - val_accuracy: 0.8989
Epoch 8/10
100/100 [==============================] - 50s 496ms/step - loss: 0.1281 - accuracy: 0.9543 - val_loss: 0.2912 - val_accuracy: 0.8911
Epoch 9/10
100/100 [==============================] - 50s 494ms/step - loss: 0.1161 - accuracy: 0.9599 - val_loss: 0.2892 - val_accuracy: 0.9025
Epoch 10/10
100/100 [==============================] - 50s 497ms/step - loss: 0.1210 - accuracy: 0.9548 - val_loss: 0.2944 - val_accuracy: 0.8964
```

Figure: Training and Validation of data for 1 layer in CNN

```
Epoch 1/20
100/100 [==============================] - 130s 1s/step - loss: 0.6062 - accuracy: 0.6639 - val_loss: 0.5368 - val_accuracy: 0.7193
Epoch 2/20
100/100 [==============================] - 128s 1s/step - loss: 0.3836 - accuracy: 0.8326 - val_loss: 0.2601 - val_accuracy: 0.8951
Epoch 3/20
100/100 [==============================] - 131s 1s/step - loss: 0.2249 - accuracy: 0.9133 - val_loss: 0.2029 - val_accuracy: 0.9252
Epoch 4/20
100/100 [==============================] - 129s 1s/step - loss: 0.1875 - accuracy: 0.9294 - val_loss: 0.1770 - val_accuracy: 0.9404
Epoch 5/20
100/100 [==============================] - 129s 1s/step - loss: 0.1590 - accuracy: 0.9425 - val_loss: 0.1634 - val_accuracy: 0.9507
Epoch 6/20
100/100 [==============================] - 129s 1s/step - loss: 0.1380 - accuracy: 0.9526 - val_loss: 0.1646 - val_accuracy: 0.9454
Epoch 7/20
100/100 [==============================] - 154s 2s/step - loss: 0.1265 - accuracy: 0.9582 - val_loss: 0.1614 - val_accuracy: 0.9485
Epoch 8/20
100/100 [==============================] - 130s 1s/step - loss: 0.1166 - accuracy: 0.9596 - val_loss: 0.1646 - val_accuracy: 0.9444
Epoch 9/20
100/100 [==============================] - 128s 1s/step - loss: 0.1076 - accuracy: 0.9635 - val_loss: 0.1671 - val_accuracy: 0.9437
Epoch 10/20
100/100 [==============================] - 129s 1s/step - loss: 0.0871 - accuracy: 0.9725 - val_loss: 0.1725 - val_accuracy: 0.9447
Epoch 11/20
100/100 [==============================] - 129s 1s/step - loss: 0.0778 - accuracy: 0.9742 - val_loss: 0.1932 - val_accuracy: 0.9381
Epoch 12/20
100/100 [==============================] - 128s 1s/step - loss: 0.0685 - accuracy: 0.9778 - val_loss: 0.1815 - val_accuracy: 0.9437
Epoch 13/20
...
Epoch 19/20
100/100 [==============================] - 128s 1s/step - loss: 0.0220 - accuracy: 0.9927 - val_loss: 0.3108 - val_accuracy: 0.9366
Epoch 20/20
100/100 [==============================] - 128s 1s/step - loss: 0.0199 - accuracy: 0.9934 - val_loss: 0.3149 - val_accuracy: 0.9378
```

Figure: *Training and Validation of data for 2 layer in CNN*

```
Epoch 1/10
100/100 [==============================] - 217s 2s/step - loss: 0.6602 - accuracy: 0.5958 - val_loss: 0.5856 - val_accuracy: 0.7251
Epoch 2/10
100/100 [==============================] - 213s 2s/step - loss: 0.3243 - accuracy: 0.8608 - val_loss: 0.1961 - val_accuracy: 0.9353
Epoch 3/10
100/100 [==============================] - 224s 2s/step - loss: 0.1605 - accuracy: 0.9488 - val_loss: 0.1642 - val_accuracy: 0.9490
Epoch 4/10
100/100 [==============================] - 225s 2s/step - loss: 0.1397 - accuracy: 0.9539 - val_loss: 0.1472 - val_accuracy: 0.9533
Epoch 5/10
100/100 [==============================] - 215s 2s/step - loss: 0.1310 - accuracy: 0.9572 - val_loss: 0.1441 - val_accuracy: 0.9543
Epoch 6/10
100/100 [==============================] - 461s 5s/step - loss: 0.1167 - accuracy: 0.9609 - val_loss: 0.1264 - val_accuracy: 0.9593
Epoch 7/10
100/100 [==============================] - 311s 3s/step - loss: 0.1109 - accuracy: 0.9626 - val_loss: 0.1229 - val_accuracy: 0.9613
Epoch 8/10
100/100 [==============================] - 328s 3s/step - loss: 0.0959 - accuracy: 0.9672 - val_loss: 0.1219 - val_accuracy: 0.9616
Epoch 9/10
100/100 [==============================] - 266s 3s/step - loss: 0.0906 - accuracy: 0.9692 - val_loss: 0.1276 - val_accuracy: 0.9588
Epoch 10/10
100/100 [==============================] - 260s 3s/step - loss: 0.0853 - accuracy: 0.9707 - val_loss: 0.1271 - val_accuracy: 0.9603
```

Figure: *Training and Validation of data for 3 layer in CNN*

```
Epoch 1/10
100/100 [==============================] - 324s 3s/step - loss: 0.7807 - accuracy: 0.7028 - val_loss: 0.4944 - val_accuracy: 0.7913
Epoch 2/10
100/100 [==============================] - 326s 3s/step - loss: 0.4428 - accuracy: 0.8155 - val_loss: 0.4045 - val_accuracy: 0.8322
Epoch 3/10
100/100 [==============================] - 340s 3s/step - loss: 0.3839 - accuracy: 0.8369 - val_loss: 0.3673 - val_accuracy: 0.8474
Epoch 4/10
100/100 [==============================] - 370s 4s/step - loss: 0.3578 - accuracy: 0.8466 - val_loss: 0.3490 - val_accuracy: 0.8557
Epoch 5/10
100/100 [==============================] - 308s 3s/step - loss: 0.3430 - accuracy: 0.8528 - val_loss: 0.3475 - val_accuracy: 0.8550
Epoch 6/10
100/100 [==============================] - 275s 3s/step - loss: 0.3348 - accuracy: 0.8561 - val_loss: 0.3325 - val_accuracy: 0.8623
Epoch 7/10
100/100 [==============================] - 285s 3s/step - loss: 0.3278 - accuracy: 0.8575 - val_loss: 0.3282 - val_accuracy: 0.8631
Epoch 8/10
100/100 [==============================] - 272s 3s/step - loss: 0.3223 - accuracy: 0.8616 - val_loss: 0.3237 - val_accuracy: 0.8621
Epoch 9/10
100/100 [==============================] - 266s 3s/step - loss: 0.3181 - accuracy: 0.8627 - val_loss: 0.3214 - val_accuracy: 0.8648
Epoch 10/10
100/100 [==============================] - 277s 3s/step - loss: 0.3160 - accuracy: 0.8630 - val_loss: 0.3209 - val_accuracy: 0.8636
```

Figure: *Training and Validation of data in VGG*

```
Epoch 1/10
100/100 [==============================] - 174s 2s/step - loss: 0.7769 - accuracy: 0.5944 - val_loss: 0.6446 - val_accuracy: 0.6523
Epoch 2/10
100/100 [==============================] - 150s 1s/step - loss: 0.6292 - accuracy: 0.6586 - val_loss: 0.6202 - val_accuracy: 0.6700
Epoch 3/10
100/100 [==============================] - 149s 1s/step - loss: 0.6182 - accuracy: 0.6648 - val_loss: 0.6131 - val_accuracy: 0.6789
Epoch 4/10
100/100 [==============================] - 150s 1s/step - loss: 0.6138 - accuracy: 0.6680 - val_loss: 0.6073 - val_accuracy: 0.6812
Epoch 5/10
100/100 [==============================] - 150s 1s/step - loss: 0.6068 - accuracy: 0.6789 - val_loss: 0.6111 - val_accuracy: 0.6713
Epoch 6/10
100/100 [==============================] - 154s 2s/step - loss: 0.6013 - accuracy: 0.6827 - val_loss: 0.5982 - val_accuracy: 0.6938
Epoch 7/10
100/100 [==============================] - 150s 1s/step - loss: 0.6018 - accuracy: 0.6808 - val_loss: 0.5976 - val_accuracy: 0.6875
Epoch 8/10
100/100 [==============================] - 151s 1s/step - loss: 0.5984 - accuracy: 0.6842 - val_loss: 0.5941 - val_accuracy: 0.6950
Epoch 9/10
100/100 [==============================] - 149s 1s/step - loss: 0.5949 - accuracy: 0.6863 - val_loss: 0.5970 - val_accuracy: 0.6814
Epoch 10/10
100/100 [==============================] - 150s 1s/step - loss: 0.5950 - accuracy: 0.6859 - val_loss: 0.5912 - val_accuracy: 0.6928
```

Figure: *Training and Validation of data in RESNET*

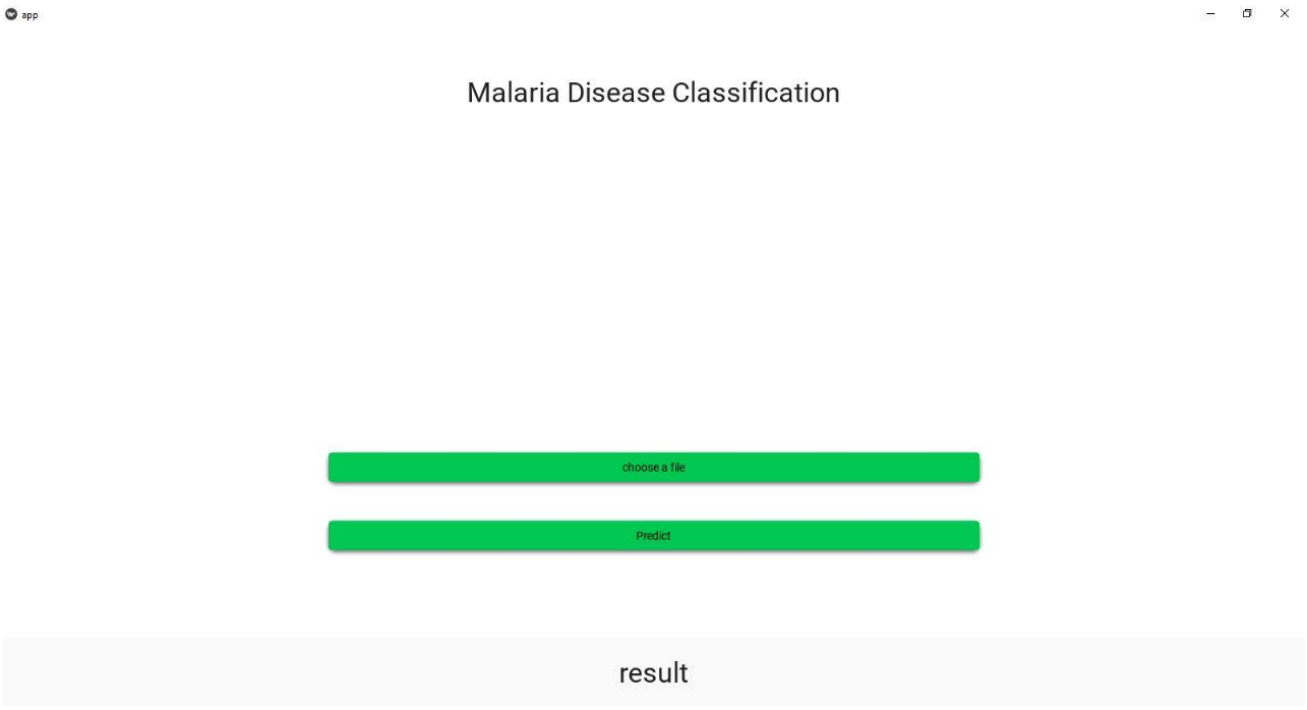


Figure:*Framework for detection of Malarial Parasite using CNN algorithm*

Figure:*Predicted the inexistence of Malarial parasite by CNN*

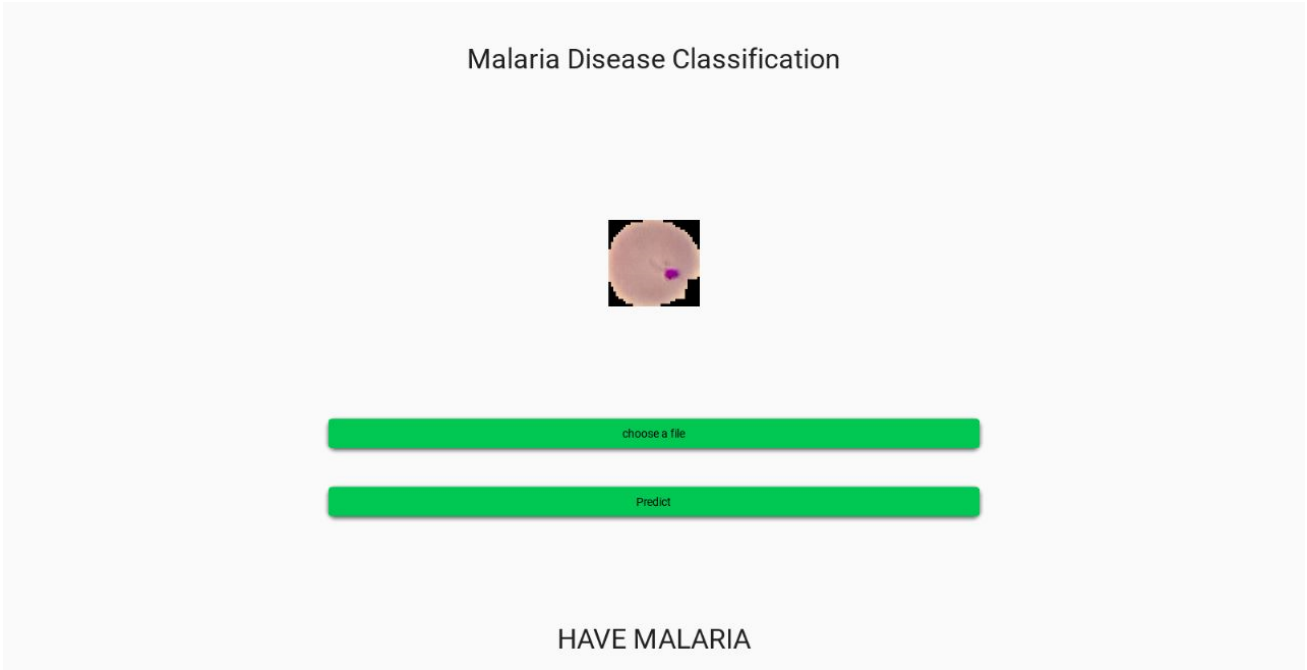
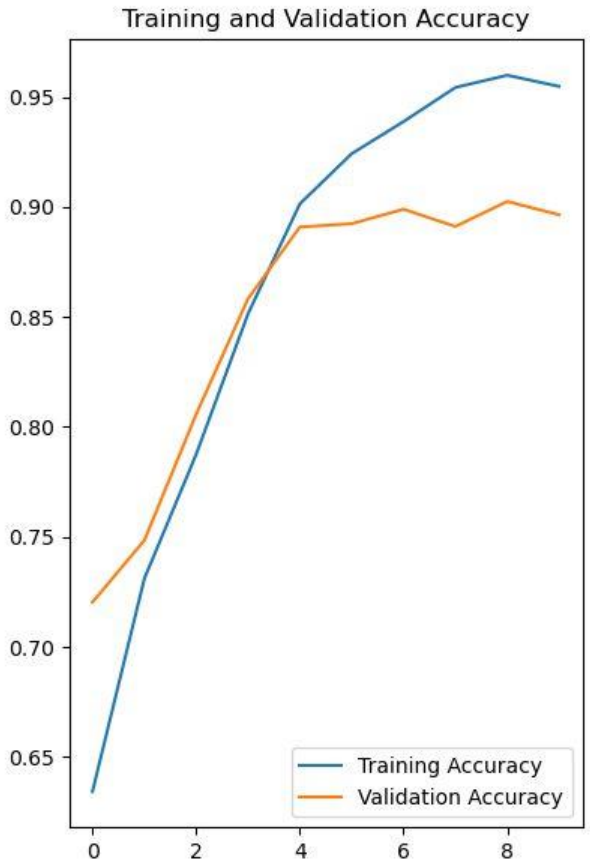


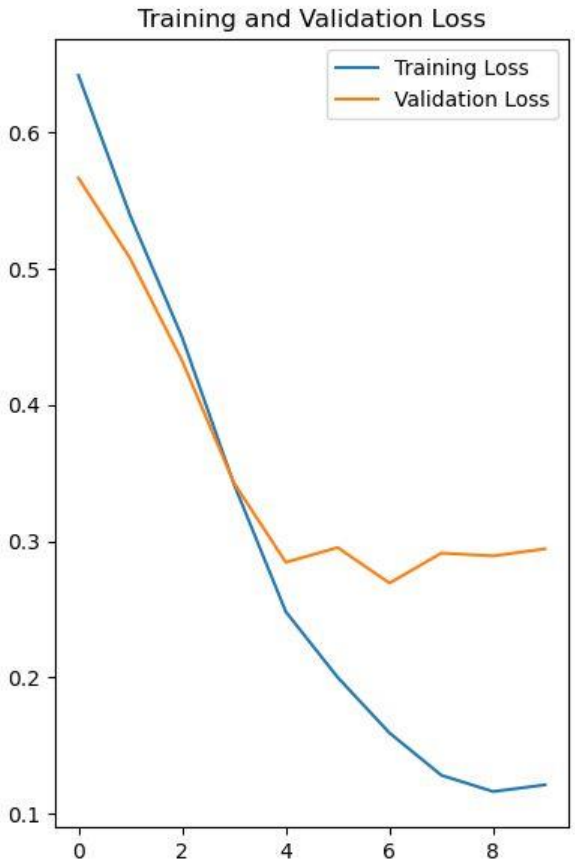
Figure:*Predicted the existence of Malarial parasite by CNN*

GRAPHICAL REPRESENTATIONS

FOR CNN:

#1 layer:

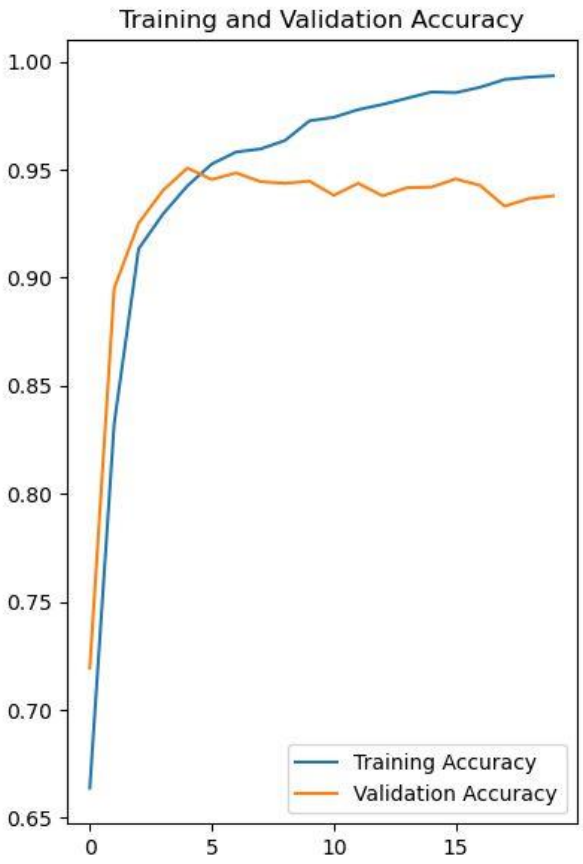


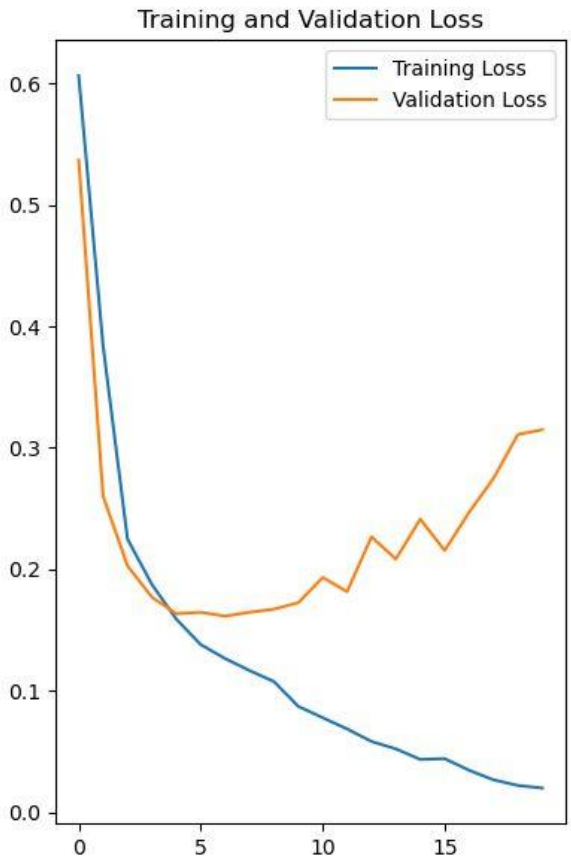


x-axis:Accuracy
y-axis:Epochs

x-axis:Loss
y-axis:Epochs

#for 2 layers:

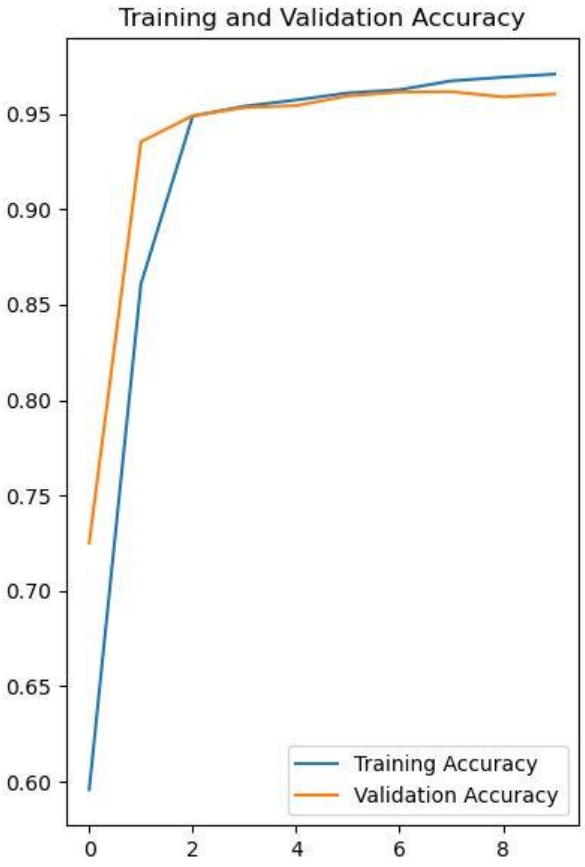


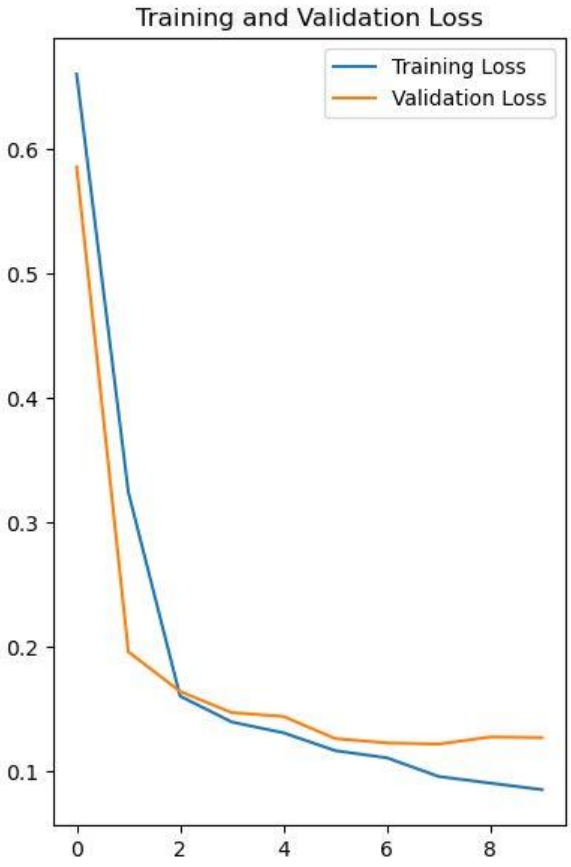


x-axis:Accuracy
y-axis:Epochs

x-axis:Loss
y-axis:Epochs

#for 3 layers:

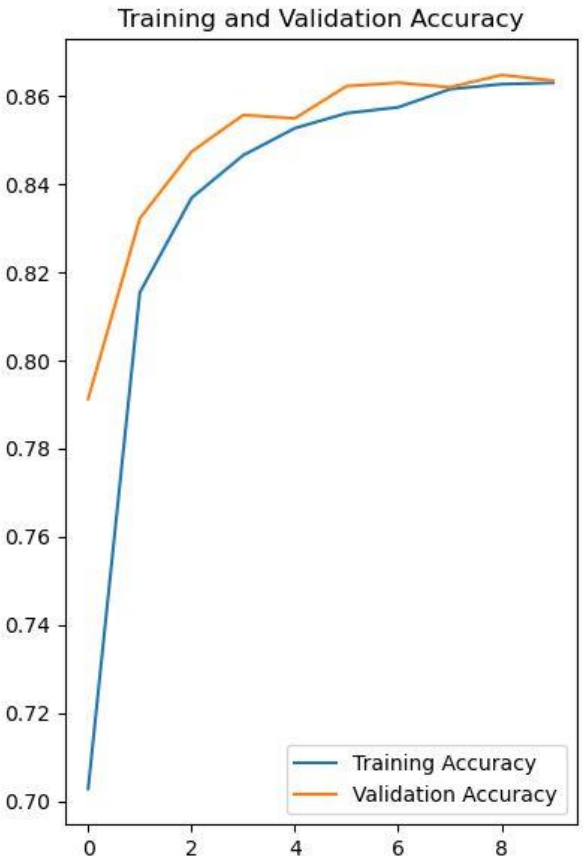




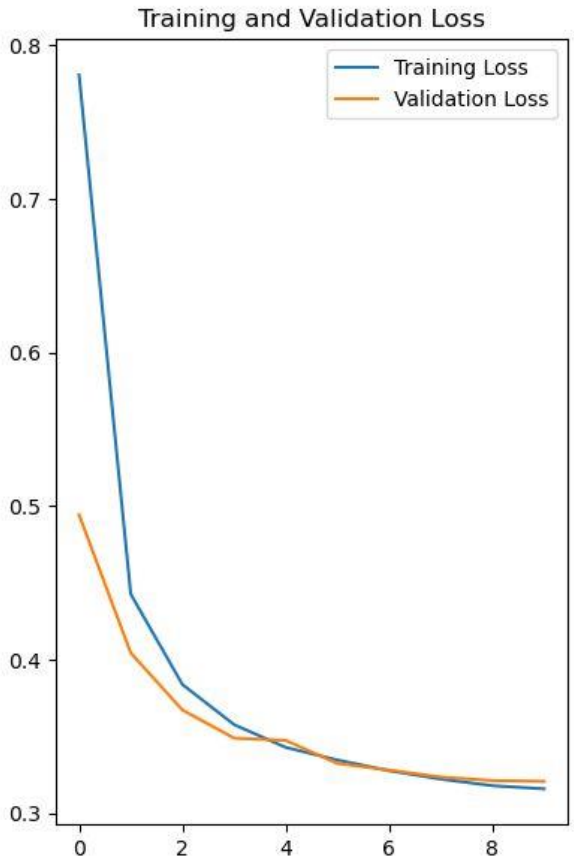
x-axis:Accuracy
y-axis:Epochs

x-axis:Loss
y-axis:Epochs

GRAPHICAL REPRESENTATION FOR VGG:

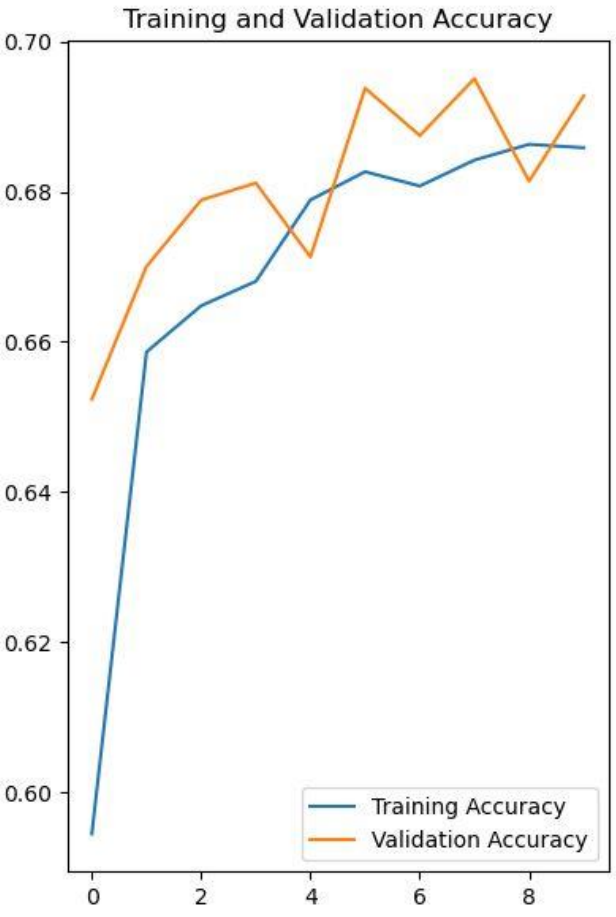


x-axis:Accuracy
y-axis:Epochs

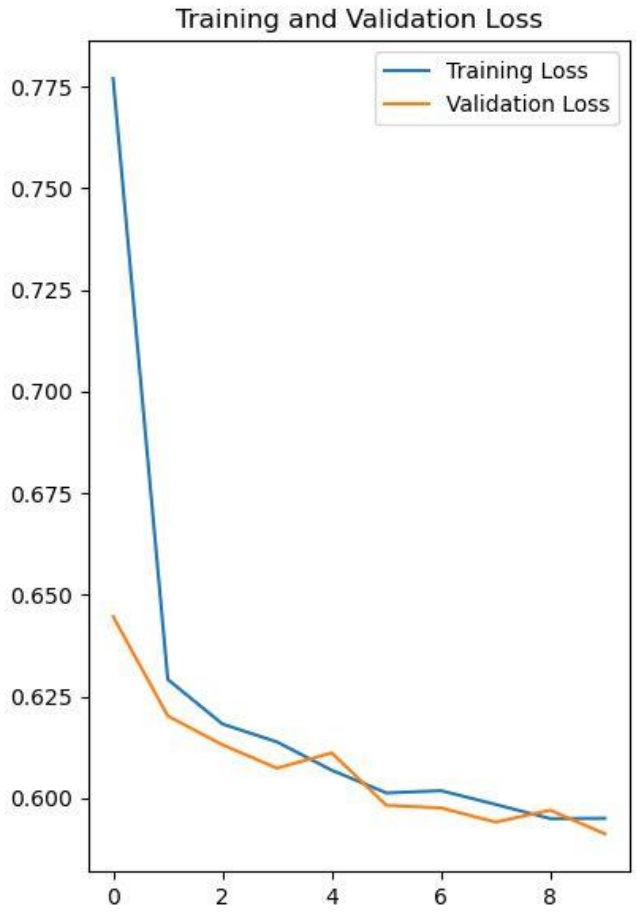


x-axis:Loss
y-axis:Epochs

GRAPHICAL REPRESENTATION FOR RESNET:



x-axis:Accuracy
y-axis:Epochs



x-axis:Loss
y-axis:Epochs

RESULTS AND DISCUSSIONS:

RESULTS FOR CNN:

The experiments were conducted on a Lenovo laptop with 8GB RAM and 2.4Ghz machine. The training process took two hours fifty mins to achieve overall result on Malarial image Dataset. The Proposed Architecture was evaluated based on Accuracy, as shown in the table. The results indicate that the proposed architecture achieved 96.194% accuracy and 12.075% loss. Other experiments were also performed by varying the number of convolutional layers, and the result is shown in the Table. It was observed that increasing no. of cnn layers up to 3 layers improved the accuracy, and the best result was obtained with the CNN-3 layer model. The Graphical Representation of the Results can be visualized in the subsequent picture. The Model was Trained for 10 Epochs obtaining an accuracy greater than 96%. To improve the image quality and reduce noise, stain normalization was applied during the preprocessing steps. It was discovered that engaging in these activities enhanced the overall outcomes as distinguishing between stains and plasmodium or other types of artifacts in blood can pose a difficulty for the classifier.

MODEL with multiple layers	Accuracy	Loss
CNN with Single Layer	91.116	23.812
CNN with Dual Layer	95.388	23.253
CNN with Triple Layer	96.194	12.075

Table:Values of Accuracy and Loss for different No. of layers in CNN

RESULTS FOR VGG:

In this ,the accuracy and loss of the proposed architecture were assessed, as presented in the table. The findings demonstrate that the proposed model attained an accuracy of 85.666% and a loss of 32.450%. The model underwent five epochs of training, with an accuracy exceeding 85%.

Accuracy	**Loss**
85.666	32.450

Table:Metrics Values for VGG algorithmm

RESULTS FOR ResNet:

After analyzing the proposed architecture's performance using metrics such as accuracy and loss, as presented in the table, it was found that the model attained an accuracy of 69.833% and a loss of 58.725%. The training process involved 5 epochs and the model was able to achieve an accuracy above 65%.

Accuracy	**Loss**
69.833	58.725

Table:Metrics Values for ResNet algorithmm

CONCLUSION and FUTURESCOPE:

Conventional machine learning techniques have displayed Atmost precision in identifying malarial parasites. According to the results of this research, it is suggested that the utilization of convolutional layers with different depths and filter sizes could effectively extract high-level abstract features to facilitate the process of classification. Furthermore, the study confirms that CNN-based features are superior to manually designed features. By utilizing stain normalization, the CNN model outperformed existing deep learning approaches. The proposed CNN model achieved a high accuracy of 96.192%,VGG achieved accuracy of 85.666%,ResNet achiheved accuracy of 69.833.
By comparision we can notice that CNN produces higher accuracy.

ALGORITHM	**ACCURACY**	**LOSS**
CNN	96.194	12.075
VGG	85.666	32.450
RESNET	69.833	58.725

Table: Comparision of Results

In the future, the researchers aim to further refine the model to enhance the classification accuracy within or across different subjects.

REFERENCES:

1. Dev Kumar Das a, Madhumala Ghosha, Mallika Pal b, Asok K. Maiti b, Chandan Chakrabortya," Machine learning approach for automated screening of malaria parasite using light microscopic images" November 2012.
2. F. Boray Tek a,*, Andrew G. Dempster b , Izzet Kale" Parasite detection and identification for automated thin blood film malaria diagnosis" August 2009.
3. Mahdieh Poostchi, Kamolrat Silamut, Richard J. Maude, Stefan Jaeger, and George Thoma," Image analysis and machine learning for detecting malaria" December, 2017.
4. Rahul Das.P, Karuna.G, Srilakshmi.V, Rupa.B," An efficient smartphone based Parasite Malaria Detection with Deep Neural Networks".
5. Soner Can Kalkan, Ozgur Koray Sahingoz" Deep Learning Based Classification of Malaria from Slide Images", 2019 IEEE.
6. Bin Qin,Yufan Wu,Zhili Wang,Haocheng Zheng," Malaria Cell Detection Using Evolutionary Convolutional Deep Networks" 2019 IEEE.
7. Divyansh Shah, Khushbu Kawale, Masumi Shah, Santosh Randive, Rahul Mapari," Malaria Parasite Detection Using Deep Learning" 2020 IEEE.
8. Srinivasan Sankarana,1, Muthukumaran Malarvel b,1, Gopalakrishnan Sethumadhavanb,∗, Dinkar Sahal c," Quantitation of Malarial parasitemia in Giemsa stained thin blood smears using Six Sigma threshold as preprocessor" July 2017.
9. Anik Khan, Kishor Datta Gupta, Deepak Venugopal, Nirman Kumar," Completely Interpretable Detection of Malaria Parasite in Red Blood Cells using Lower-dimensional Feature Space" 2020 IEEE.
10. Octave Iradukunda1 , Haiying Che1*, Josiane Uwineza1, Jean Yves Bayingana1 , Muhammad S Bin-Imam1 , Ibrahim Niyonzima1," Malaria Disease Prediction Based on Machine Learning", 2019 IEEE.
11. Aishah Khan, Ashraf Khalil, Hassan Hajjdiab," Mobile Microscopic device to detect parasitical cell-related diseases using Machine Learning" 2018.

BASE PAPER

IEEE Access

Received May 6, 2020, accepted May 11, 2020, date of publication May 14, 2020, date of current version June 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994880

A Novel Stacked CNN for Malarial Parasite Detection in Thin Blood Smear Images

MUHAMMAD UMER[1], SAIMA SADIQ[1], MUHAMMAD AHMAD[2,3], SALEEM ULLAH[1], GYU SANG CHOI[4], AND ARIF MEHMOOD[5]

[1]Department of Computer Science, Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan 64200, Pakistan
[2]Department of Computer Engineering, Khwaja Fareed University of Engineering and Information Technology (KFUEIT), Rahim Yar Khan 64200, Pakistan
[3]Dipartimento di Matematica e Informatica—MIFT, University of Messina, 98122 Messina, Italy
[4]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38542, South Korea
[5]Department of Computer Science and Information Technology, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan

Corresponding authors: Muhammad Ahmad (mahmad00@gmail.com), Gyu Sang Choi (castchoi@ynu.ac.kr), and Arif Mehmood (arifnhmp@gmail.com)

This work was supported in part by the National Research Foundation of Korea (NRF) through the Basic Science Research Program funded by the Ministry of Education under Grant NRF-2019R1A2C1006159, in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Promotion (IITP), under Grant IITP-2020-2016-0-00313, in part by the National Research Foundation of Korea (NRF) through The Brain Korea 21 Plus Program, under Grant 22A20130012814, and in part by the Fareed Computing Research Center, Department of Computer Science under Khwaja Fareed University of Engineering and Information Technology (KFUEIT), Punjab, Rahim Yar Khan, Pakistan.

ABSTRACT Malaria refers to a contagious mosquito-borne disease caused by parasite genus plasmodium transmitted by mosquito female Anopheles. As infected mosquito bites a person, the parasite multiplies in the host's liver and start destroying the red-cells. The disease is examined visually under the microscope for infected red-cells. This diagnosis depends upon the expertise and experience of pathologists and reports may vary in different laboratories doing a manual examination. Another way around, many machine learning techniques have been applied for spontaneous detection of blood smears. However, feature engineering is a challenging task that requires expertise to adjust positional and morphological features. Therefore, this study proposes a novel Stacked Convolutional Neural Network architecture that improves the automatic detection of malaria without considering the hand-crafted features. The 5-fold cross-validation process on 27, 558 cell images with equal instances of parasitized and uninfected cells on a publicly available dataset from the National Institute of health, the accuracy of our proposed model is 99.98%. Furthermore, the statistical results revealed that the proposed model is superior to the state-of-the-art models with 100% precision, 99.9% recall, and 99% f1-measure.

INDEX TERMS Convolutional neural network (CNN), Malaria, blood smear images, deep learning, diagnostic approach.

I. INTRODUCTION

Malaria is an infectious and life-threatening disease caused by protozoa plasmodium with a minimum of seven days of the incubation period. This disease is transmitted through the bite of female mosquito Anopheles that also known as malaria vectors. Among 400 species of Anopheles mosquitos, only 30 species are malaria vectors. P. falciparum and P. vivax are the most common single-cell Plasmodium species that cause malaria and can be toxic. Initial symptoms, headache, vomiting, fever, and chills can be mild and difficult to recognize as malaria if remain untreated can cause severe illness and may lead to death [1]. According to the World Malaria Report in 2018, a number of malarial deaths i.e., 435000 were reported in 2018 [2].

Malarial virus transmission depends on climate conditions especially after rain and it is more intense when the temperature became feasible for a longer span of life of a mosquito. This is the reason for 90% world's malaria cases occur in Africa and common in other tropical regions such as Latin America and Asia [3]. Early detection can prevent harmful consequences and a patient can be treated with proper medicine on time.

To identify the malarial parasite, numerous techniques have been proposed and microscopic examination of Giemsa stain blood smear is manifest [4]. Other techniques include

The associate editor coordinating the review of this manuscript and approving it for publication was Shiping Wen.

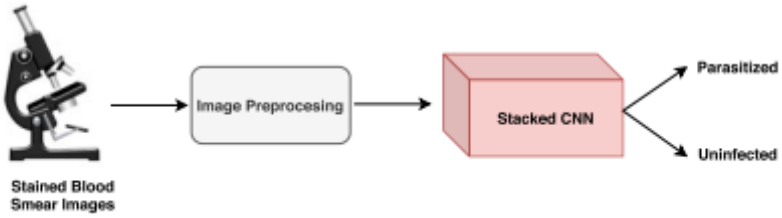


FIGURE 1. Pipeline of our proposed architecture.

polymerase chain reaction and rapid diagnostic tests to detect the antigen in the blood. Although other tests outperform in malaria detection, however, microscopy is widespread due to low cost and less complexity and its efficacy depends upon pathologist expertise [5]. False diagnosis may lead to more severe malaria or the use of un-necessary malarial drugs [6]. To improve the treatment of an individual patient by automatic detection of malarial parasites is a very appealing area of research. It has two advantages; first, it will improve diagnosis even with limited resources, and secondly, it is cost-effective.

Automatic parasite detection from thin blood smear under microscope results to differentiate parasite species. The first step is to segment Red Blood Cells (RBCs) and then these segments are classified as infected or uninfected [7]. However, by applying machine learning on the medical image analysis task, feature engineering is challenging to get desired results because hand-crafted features are being used to make decisions [7], [8]. Furthermore, experienced individuals are required to adjust the size, angle, position, and region of interest (ROI) of the image. To cope with these issues, Deep Learning (DL) is being used to extract high-level features that result in end-to-end extraction of features and classification [9], [10].

For traditional image classification and analysis, the spatial correlation of neighboring pixels contains important information [11]. Convolutional Neural Network (CNN) is designed to extract such information i.e., end to end feature extraction and classification through weights and pooling [10]. However, the size of training data greatly affects the classification performance of CNN [12] as opposed to the traditional machine learning models [13]. To cope with the aforementioned issues, transfer learning has been proposed in which the features are extracted by a pre-trained network including but not limited to GoogleLeNet [14], VGGNet [15], and ResNet [16]. Transfer learning has been used as a shortcut where training time is saved by compromising performance [17].

The DenseNet architecture is a variant of CNN which is composed of dense layers in which each layer is fully connected to the later one and each layer serves as a feature extractor [18]. DenseNet significantly improves performance for medical images without considering a large number of parameters [19]. Therefore, this work proposed a stacked CNN architecture that learns a different level of abstraction of a complex representation of malaria parasites for the classification of parasitized and uninfected cells for disease screening. A pipeline of our proposed architecture can be seen in Fig.1.

The rest of the paper is structured as follows. Section 2 describes the most relevant and state-of-the-art researches related to our proposed work. Section 3 gives a summary of the dataset, preprocessing, and related steps performed on the dataset. Section 4, presents a brief explanation of the deep learning model proposed in this work, experiment details, and machine specifications used for the experiment. In section 5 results are discussed and finally, section 6 concludes the work with possible future directions.

II. RELATED WORK

For automatic blood smear classification, traditional machine learning approaches have been used such as Diaz *et al.*, classified blood smear images using a Support Vector Machine (SVM) to detect infected erythrocytes and their infected stage. This approach performed well with 94.0% sensitivity on a dataset containing 450 images [20]. In the structural characterization of blood cells, machine learning plays a vital role such as computer-aided learning techniques for pattern recognition that have been used by many researchers to identify the malaria parasitemia. Das *et al.*, [21] extract features from erythrocytes textures and then apply feature selection techniques to further reduce it to 96 features and then applied statistical techniques such as Bayesian network and SVM for classification. The highest accuracy of 84.0% is achieved by a Bayesian network with the 19 most important features. Shen *et al.* [22] used a stacked autoencoder to learn features automatically from infected and infected images of cells.

Another way around, computer vision-based malarial parasite detection studies have also been proposed in the literature such as Tek *et al.* used a modified K-nearest neighbor (KNN) after applying normalization and color correction on input images of 9 blood films for binary classification [23]. Automatic detection and staging of infected RBCs by malarial parasite P. falciparum by using a quantitative phase analysis of images without staining is performed by [24]. Mustafa *et al.* [25] compared their work with Bradley's [32], wolf's [33], Bernsen's [34], Deghosting [35],

Triangle's [36] and Fuzzy C-mean clustering method [37] for malarial parasite detection. Mustafa *et al.* also proposed thresholding as an important preprocessing step. Fuzzy C-mean outperformed among all the other five methods. Although reported outcomes using machine learning are reasonable, all the techniques need to prove their ability on large datasets as all these approaches have been evaluated on small sets of images. Therefore, there is a need for a deep learning approach such as Convolutional Neural Network (CNN), which has proved robustness on large datasets.

Deep neural networks such as Generative Adversarial Network (GAN) [38], Discrete-time Recurrent Neural Networks (DRNNs) [39] and Memristive Neural Networks (MNNs) [40] have been widely used for various tasks. CNN models has been extensively used for traditional image analysis, phoneme recognition [41], document recognition [42], visual document analysis [43], face labeling [44] and object recognition [45], [46]. CNN model-based on 16 layers of malarial parasite detection was proposed in [26] which only classify the blood cells as infected or uninfected. The model was trained with approximately 27000 images and achieved 97.0% accuracy, specificity and sensitivity which is higher than transfer learning. To compensate limited resources images were resampled to the size of 44×44 pixels. The first application based on a deep belief network was proposed by Bibin *et al.* [27] and tested on 4100 peripheral blood smear images which achieved 89.66% F1-score.

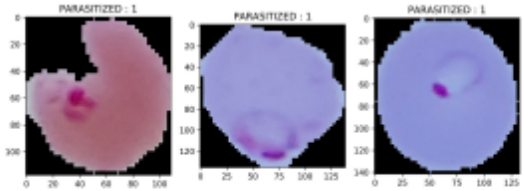
A customized CNN architecture for the detection of plasmodium in blood smear on Leishman stained focused stacked images was proposed by Gopakumar *et al.*, which show 97.0% sensitivity and 98.0% specificity [28]. Automatic identification of the malarial parasite was proposed in [29] which uses the patient level evaluation and thumbnails to improve the user confidence in system findings with overall 89.7% precision, 94.1% specificity and 89.7% sensitivity. Rajaraman *et al.* [30] evaluated a pre-trained end-to-end (i.e., feature extraction and classification) CNN model based on single-cell images. They observed that a pre-trained ResNet-50 model as an outstanding tool for diagnosis with an accuracy of 98.6%, 98.1% sensitivity, 99.2% specificity, and 95.7% F1-score. The detailed summary of existing works is shown in Table 1.

Even though the existing state-of-the-art deep learning approaches have shown promising results in malarial parasite detection but still there is room for improvement. Sometimes uninfected blood samples do not contain plasmodium but may contain other types of remnants that are wrongly classified as infected by a classifier. Therefore, color normalization techniques are needed before the classification. In this work, we evaluated the customized CNN model for feature extraction and then to classify images as infected or uninfected cells. Rajaraman *et al.* use 3 layers of CNN (in our case it is 5), the second thing in all three layers they applied same filter size with the same number of kernels (3×3, @32) while in our case we vary the kernel size (4×4, 3×3, 2×2) with kernels ranging from 32 to 256. Sequentially reducing kernel size helps the model to get trained on small size malarial cell detection.

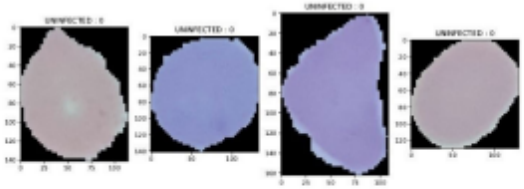
III. DATASET & PREPROCESSING

A. MALARIAL DATASET

Dataset used in our study contains images based on Giemsa stained slides of thin blood smear obtained from malaria screener research activity of 50 healthy patients and 150 P. falciparum-infected patients. It is taken from the National Institute of Health (NIH). Images in a dataset are manually annotated by slide reader experts of Mahidol Oxford Tropical Medicine Research Unit Bangkok, Thailand, and collected at the National Library of Medicine (NLM). The dataset contains 27, 558 images with the equal occurrence of infected and uninfected red blood cell images as shown in Table 2. Infected blood cell image samples contain plasmodium as shown in Fig. 2(a) and uninfected blood image samples do not contain plasmodium as shown in Fig. 2(b). Colored patches of red blood cells are of variant sizes ($110 - 150$ pixels), which are resampled to 120×120 according to the input requirement of classifier during preprocessing.



(a) Parasitized Images



(b) Uninfected Images

FIGURE 2. Microscopic view of thin blood smear.

B. PREPROCESSING

The original images of the malaria dataset are captured by a mobile device therefore these images are in different sizes. Thus, before any training and testing, we resampled the images to unified image size. On the first step of preprocessing we convert all images to fix the size of 120×120 pixels. Secondly, we apply the kernel on the image to get the edges. On the third step of preprocessing we convert BGR to YUV to get the values of one luma component (Y') and two chrominance components, called U (blue projection) and V (red projection). Color variations in blood cell images exist due to the use of chemicals which can result

TABLE 1. Summary table: Studies of malarial parasite detection.

Author	Dataset	Species	Staining	Blood Smear type	Features	Method	Performance
Diaz [20]	450 images	P. Falciparum	Giemsa	Thin	25 features	SVM	Sensitivity 94% Specificity 99.7%
Das [21]	Leishman stained peripheral blood smear images	P. falciparum & P. vivax	Leishman	Thin	Textual and Morphological	Naïve Bayes and SVM	Sensitivity 96.62 − 98.10%, Specificity 68.91 − 88.51%
Tek [23]	9 blood films	P. falciparum	Giemsa	Thin	color and scales	KNN	Sensitivity 74% Specificity 98%
Park [24]	Quantitative phase images of unstained cells	P. falciparum	Giemsa	Thin	23 Morphological descriptors based on the phase information	LDC, KNN, LR	LDC 99.7%, KNN 99.5% LR 99.7%
Mustafa [25]	30 Malaria cell images	P. falciparum	Giemsa	Thin	Thresholding	Bradley Fuzzy, Fuzzy C-Mean, Wolf Deghost, Bernsen Triangle	Bradley 78.43%, C-Mean 85.31%, Wolf 70.83%, Deghost 81.70%, Bernsen 65.53%, Triangle 78.12%
Liang [26]	27,578 Erythrocyte images	P. falciparum	Giemsa	Thin	NA	CNN	Sensitivity 96.99%, Specificity 97.75%
Bibin [27]	4100 Peripheral blood smear images	P. falciparum	Giemsa	Thin	Concatenated feature of color and texture	DBN	Sensitivity 97.60%, Specificity 95.92%
Gopakumar [28]	Focus stack of images	P. falciparum	Giemsa	Thin	14 features	SVM, CNN	Sensitivity 92.95 − 97.06%, Specificity 93.82 − 98.50
Mehanian [29]	Malaria blood film library	P. falciparum	Giemsa	Thin	N/A	CNN	Sensitivity 91.6%, Specificity 94.1%
Rajaraman [30]	27,558 cell images	P. falciparum	Giemsa	Thin	N/A	Pre-trained CNN	Sensitivity 0.981%, Specificity 0.992%
Rahman [31]	27,558 cell images	P. falciparum	Giemsa	Thin	N/A	CNN, VGG16 CNNEx-SVM	95.97%, VGG16 97.64%, CNNEx-SVM 94.77%

TABLE 2. Malaria dataset description.

Total images	Parasitized images	Uninfected images
27,558	13,779	13,779

in error margin. This problem can be solved by normalizing the image. Ciompi *et al.* [47] applied stain normalization on colorectal tissue classification and proved that it improves the performance. We also applied normalization in the fourth step of preprocessing to equalize the intensity values. The last step of preprocessing is to convert back the YUV image into RGB. Preprocessing steps are important to reduce noise and to improve image quality. Fig. 3(a) represents the original image before preprocessing, Fig. 3(b) shows edges obtained after applying kernel. Fig. 3(c) displays images in YUV color space to get (Y') component Fig. 3(d) shows intensity equalization and Fig. 3(e) represents images after converting back to BGR color space.

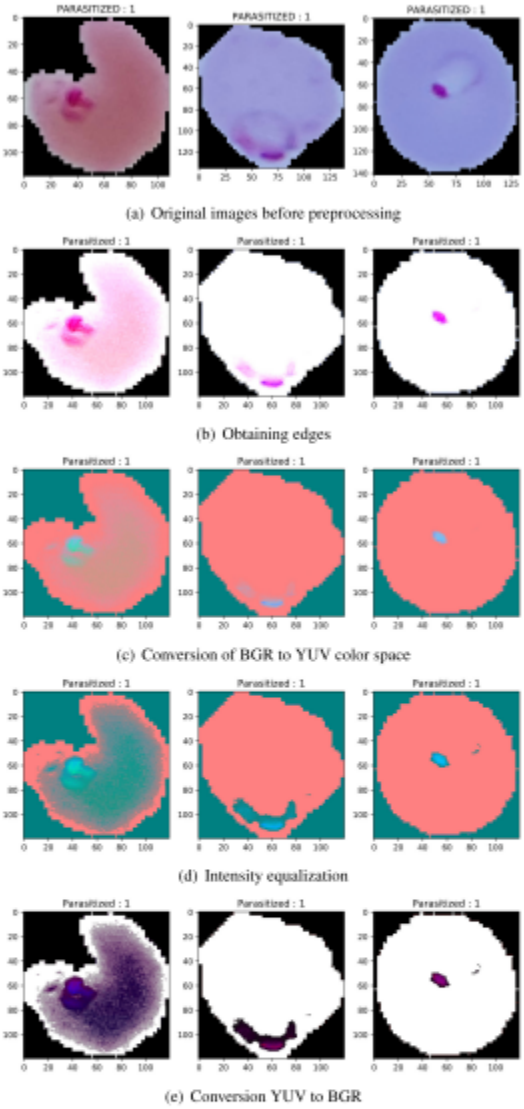
C. DATA SPLITTING AND CROSS VALIDATION

Malaria dataset is split into train/test with a ratio of 70 : 30 and to check the robustness of the model we applied a 5-fold cross-validation that is a moderate value which neither causes high bias or high variance. We randomly partitioned 5 equal subsets of our dataset; one set is used as validation and rest are used to train the model. This process is repeated five times with each subset. Then all these five subsets are averaged and used for model evaluation.

IV. PROPOSED METHODOLOGY

A. OVERVIEW

We are using a stacked CNN to overcome the shortcomings of manual feature extraction. We apply re-sampling to extract more information to CNN with a fixed sampling pattern. In addition, we applied stain normalization to preserve image characteristics.



(a) Original images before preprocessing

(b) Obtaining edges

(c) Conversion of BGR to YUV color space

(d) Intensity equalization

(e) Conversion YUV to BGR

FIGURE 3. Preprocessing steps.

The pipeline of the proposed approach composed of the following steps. First, we apply pre-processing steps to input images by re-sampling and normalizing images. Then we apply stacked CNN by fine-tuning it(filters, kernels, and strides) along with max-pooling and dropout layer. As we experimented to test different design strategies, We also created and tested models having 1,3, and 5 convolution layers, pooling layers, and a dense classification layer. We progressively increased the number of convolution layers, dropout layers, and pooling layers to see an increase in the performance of the model. We used these 1 and 3-layer models as a baseline and compare our results with these models. We used varied filter sizes and the number of layers until we achieve the best result. We could not find any other CNN architecture which showed improved performance than our proposed model. Since our 5 convolution layer stacked CNN model is the best deep learning technique for Malarial parasite classification.

B. CNN

CNN is a type of deep neural network that learns a complex hierarchy of features by convolution, nonlinear activation, and pooling layers [12]. CNN is designed for image recognition tasks as well as for image classification. Now it is commonly used in image segmentation. Traditional approach sliding window process regions independently which results in low efficiency. An alternate method is fully CNN that is trained in the end to end fashion by making computation more efficient. Fully connected layers are used at the end of the network for semantic information encoding. We have used stacked CNN as shown in Fig. 4 for detecting parasites in infected cell RGB images. CNN is a multi-layered feed-forward network inspired by biology. Filters or kernels in a layer are applied to the input of the first layer or the output of the previous layers and result in a feature map. The output of all convolutional layers is concatenated as a feature map and fed into fully connected layers. CNN has been proved as a de-facto standard by providing robust results in medical domain classification tasks. CNN has been applied for the classification of lung disease [48], brain tumor segmentation [49], chest x-rays [50], chest radiographs [51] and kidney disease [52]. In recent studies CNN has been explored for malarial parasite image classification of Giemsa stained images as parasitized or uninfected in [27], [28], [30].

The main components of CNN are convolutional layers, Rectified Linear Unit (ReLU), and pooling layer or subsampling layer. Features are extracted by the convolutional layer, ReLU is easy: it converts any negative elements of the matrix to 0 and keep the others positive constant. For the activation function, we applied the Rectified Linear Unit (ReLU).

$$y = max(0, i) \tag{1}$$

where y is the output activation and i is the given input. During training kernel weights are applied on the input image to extract local features at convolution and subsequent layers extract high-level features form these local features. In multi-channel images, CNN opens up ways in malaria diagnosis. The cross-entropy error is used as a loss function as it is used for binary classification. It is calculated as shown in equation 2.

$$crossEntropy = -(i\ log(p) + (1 - i)\ log(1 - p)) \tag{2}$$

where i is the binary indicator of class labels (0 or 1), a log is a natural logarithm and p is the predicted probability. CNN is a backpropagation variant algorithm and therefore we used sigmoid output as the error function. Here N is the total number of classes in the sigmoid layer and one neuron

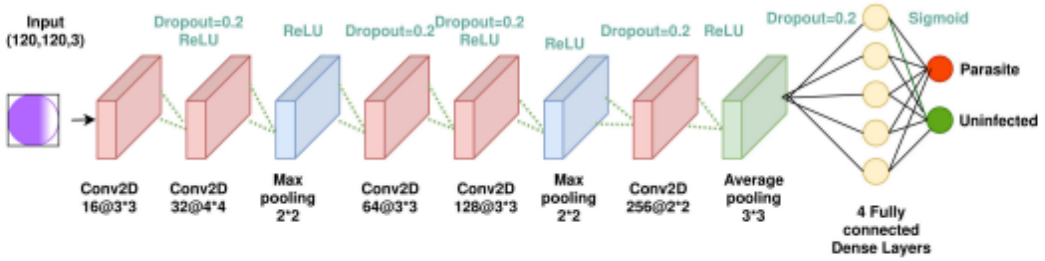


FIGURE 4. Proposed stacked CNN architecture.

corresponds to each class in the output layer. In our case, the number of classes is two parasitized and uninfected. CNN architecture produces output at two neurons in every case of binary classification. For an ideal case of the parasitized cell, the output will be 1 and 0 of first and second neurons respectively. For uninfected images, the output will be 0 and 1 that is the reverse of previous output. The term inside the log function computes the chance of output 1 and i_j is the output for a true class that is c for input. In our case, c is 1 for parasitized and 2 for uninfected. At testing time labels are assigned by excluding the softmax loss to maximize the response.

C. EXPERIMENTAL DETAILS

The design of stacked CNN architecture used in our experiment is shown in Fig. 4. Our proposed stacked architecture has been designed of total 22 layers, 5 convolutional layers, 2 max-pooling layers, 4 dense layers, 1 average pooling layer, 1 flatten layer, 8 layers with 20% dropout and 1 fully connected layer as shown in Fig. 4. The Rectified Linear Unit (ReLU) activation function is used in this setup.

Here, an input image of size 120×120 is resampled from 200×200 pixels which are enough to hold neighborhood details for making a final decision. In conv2D layer filter size (2×2), (3×3) and (4×4) are applied to convolve. The kernel size used at every convolutional layer is shown in the subscript in Fig. 4. In the MaxPooling2D layer, 2×2 pool size is used and in average pooling layer (3×3) pool size is used. Every convolutional layer is followed by a dropout layer which discards 20% of neurons. The output of the final Con2D layer of 256 output neurons is followed by an average pooling layer. This is followed by 4 dense layers of input to other activation functions. We prefer max-pooling layers before the average pooling layer as we did not want to average the details at an early stage. As it is designed for the binary classification problem, cross-entropy function is used to calculate the error between predicted actual and predicted output. Due to the binary classification output is set to 2. After setting input and output reasonable CNN can do fair classification. We choose the appropriate deep CNN architecture not too deep nor too shallow having 5 convolutional layers in our task. However, we applied max pooling to deal with the nonlinearity of features. Adam optimizer is used to remove biases. Bias is set to 0 and random weights are randomly initialized. Batch size is set to 32 samples and continued for 13 epochs. We applied a shallow to deep CNN model with a 1 convolutional layer to 5 convolutional layers and used them as a baseline method as shown in Table 4. The complete summary of the stacked CNN model hyper-parameter values is shown in table 3.

TABLE 3. Summary of the model and hyper-parameter values.

Parameter	Value
Input dimension	(120, 120, 3)
Batch size	32
Pooling	2*2
Epochs	13
Optimizer	Adam
Function	Binary cross entropy

V. RESULTS & DISCUSSIONS

All the experiments are carried out on a 2*GB* Dell PowerEdge *T*430 graphical processing unit on 2*x* Intel Xeon 8 Cores 2.4*Ghz* machine which is equipped with 32 GB *DDR*4 Random Access Memory (RAM). The training takes 3.5 hours to give the final result on the 'Malarial dataset'.

We evaluated our proposed architecture on Accuracy, Precision, Recall, and F1-score. The results of all these metrics are shown in Table 4. From Table 4, one can conclude that our proposed architecture with 5 fold cross-validation outperformed with 99.964% accuracy, 100.0% precision, 99.928% recall, and 99.964% F1- score. Table 4 enlist some of the results achieved with a different number of convolutional layers are presented. Stacked CNN-1, stacked CNN-3 and Stacked CNN-5 show accuracy 50.145%, 61.412% and 99.879% respectively. It has been observed that as the number

TABLE 4. Performance of proposed stacked CNN architecture's.

Model	Accuracy	Precision	Recall	F1-score
Stacked CNN-1 layer	50.145	50.145	100.00	66.795
Stacked CNN-3 layers	61.412	77.175	100.00	87.130
Stacked CNN-5 layers	99.879	99.976	99.783	99.879
Stacked CNN-5 with 5-fold	99.964	100.00	99.928	99.964

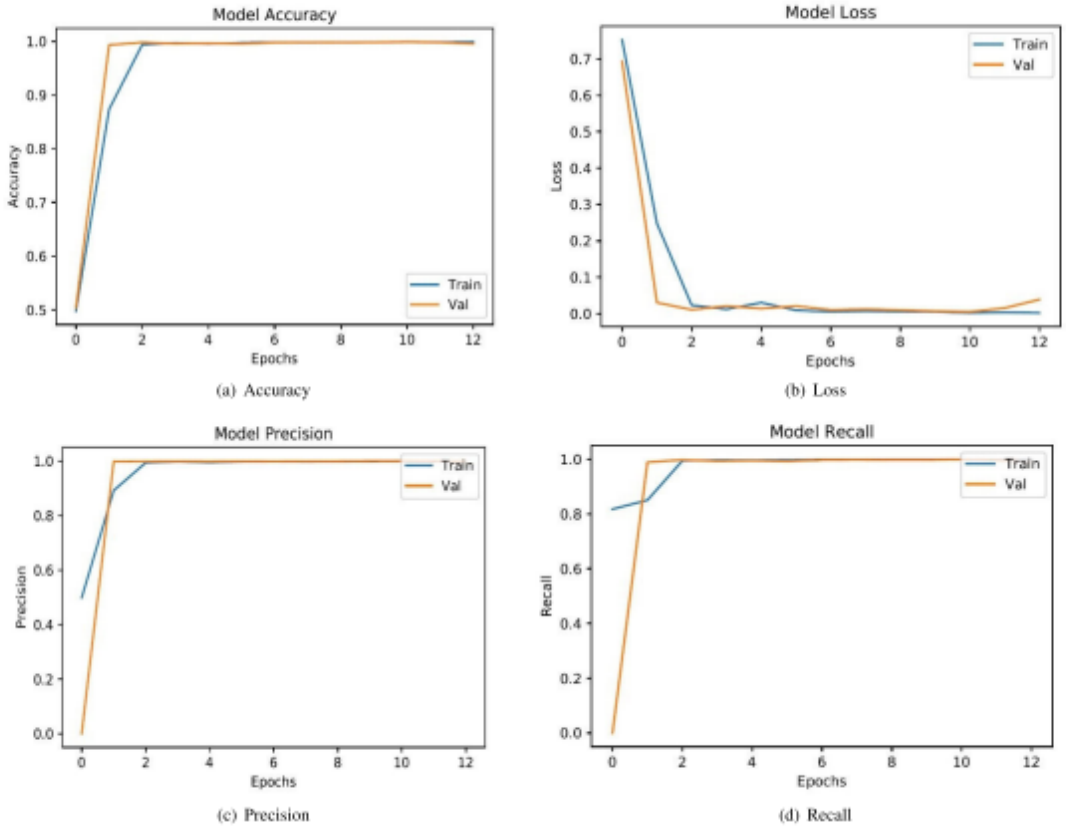


FIGURE 5. Proposed model experimental result evaluation.

of CNN layers is increasing till 5-layers, the accuracy, precision, and F1-score also increases. The results obtained by CNN-5 layers with 5-fold cross-validation are the best among all. Detail of filter size and parameters used for tuning can be seen in Table 6. Graphical representation of the result can be seen in Figure 6. Figure 5 shows training accuracy as well as loss, precision, and recall for our stacked CNN architecture. The model is trained for 13 epochs with accuracy > 99%.

We analyzed that it is difficult for a classifier to differentiate stains from plasmodium or any other artifact in blood. This is the reason we applied stain normalization in preprocessing steps. We reduced noise in preprocessing to improve the image quality. Results proved that these activities improved the overall result. Our proposed stacked CNN 5-layered model results are compared with preprocessing steps and without preprocessing steps in Table 5. There is a drastic difference in terms of accuracy, precision, recall, and F1- score after performing preprocessing steps.

Our stacked CNN model achieved optimal metric values by using five-phase extensive pre-processing, hyper-parameter optimization, different filter sizes, and dropout layers.

TABLE 5. Performance of proposed stacked CNN with and without preprocessing.

Model	Accuracy	Precision	Recall	F1-score
Stacked CNN-5 layers with preprocessing	99.879	99.976	99.783	99.879
Stacked CNN-5 layers without preprocessing	49.61	50.14	50.14	50.14

TABLE 6. Layers structure of CNN model used in this work.

CNN-2 layer	CNN-3 layer	CNN-4 layer	CNN-5 layer with Gabor
Conv (3 × 3, 0.0, Stride1 × 1) Average Pooling (3 × 3, Stride1 × 1) Dense (128 neurons) Dense (64 neurons) Dense (9 neurons) Sigmoid (2-class)	Conv (3 × 3, 0.0, Stride1 × 1) Conv (3 × 3, 0.02, Stride2 × 2) Max Pooling (2 × 2, Stride2 × 2) Conv (3 × 3, 0.04, Stride1 × 1) Average Pooling (3 × 3, Stride1 × 1) Dense (128 neurons) Dense (64 neurons) Dense (9 neurons) Sigmoid (2-class)	Conv (3 × 3, 0.0, Stride1 × 2) Conv (3 × 3, 0.02, Stride2 × 2) Max Pooling (3 × 3, Stride2 × 2) Conv (3 × 3, 0.04, Stride1 × 2) Conv (3 × 3, 0.05, Stride1 × 3) Max Pooling (2 × 2, Stride2 × 2) Conv (2 × 2, 0.250, Stride2 × 2) Average Pooling (3 × 3, Stride1 × 3) Dense (128 neurons) Dense (64 neurons) Dense (16 neurons) Sigmoid (2-class)	Conv (3 × 3, 0.0, Stride1 × 1) Conv (3 × 3, 0.02, Stride2 × 2) Max Pooling (3 × 3, Stride2 × 2) Conv (3 × 3, 0.04, Stride1 × 2) Conv (3 × 3, 0.05, Stride1 × 3) Max Pooling (2 × 2, Gabor, Stride2 × 2) Conv (2 × 2, 0.250, Stride2 × 2) Average Pooling (3 × 3, Stride1 × 1) Dense (128 neurons) Dense (64 neurons) Dense (9 neurons) Sigmoid (2-class)

In literature, many studies extracted features by using pre-trained CNN before classification [53] and others used customized CNN [30]. The stacked approach outperformed in the classification of parasitized and uninfected blood image cells. In our case, we identified optimal layers of CNN for feature extraction before classification for malarial parasite detection. It accurately identifies infected cells in terms of accuracy, precision, recall, and F1-score.

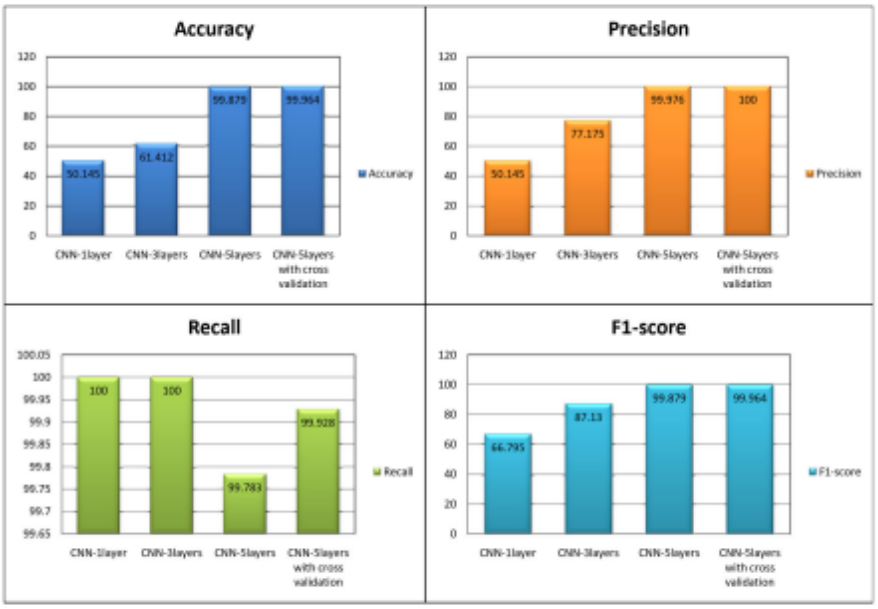


FIGURE 6. Proposed Stacked CNN architecture performance with different number of layers.

We have further compared the results of the proposed stacked CNN model with state-of-the-art Deep learning models proposed in the literature. Deep CNN Models [26], [30] and [31] was chosen as baseline methods for comparison with proposed stacked CNN as these models have recently achieved best results for malarial parasite detection. The models selected for comparison purposes are tested on a dataset based on Giemsa stained thin blood smear image slides. In [26] researchers proposed a 16 layered CNN architecture containing 6 convolutional layers and obtained good accuracy of 97.37% and 97.36% F1-score. The setting used in said work is as follows; filter sizes (5×5, 4×4, 3×3) are used and (5×5) filter size is used in 4 convolutional layers out of 6 layers. It can be observed that shallow models give reasonable accuracy with a large filter size. Hence filter size plays an important role in CNN architecture. We used a small filter size and reduced it in a sequential way form (4×4) to (2×2) which helps the model to train on small spots on infected parasitized blood cell images.

The ResNet model was proposed in [30] which outperformed among all five used pre-trained deep learning models with 95.70% accuracy and their 3-layered customized CNN model achieved 94.00% accuracy. Rajaraman *et al.* [30] also claimed that they need to use color normalization techniques to improve accuracy. The proposed model shows good accuracy improvement over the baseline [30] by applying color normalization at the pre-processing level. The comparison of the proposed model and other state-of-the-art models are shown in Table 7.

TABLE 7. Performance comparison between CNN based DL models in literature on malaria dataset.

Model	Accuracy	Precision	Recall	F1-score
Customized [26]	97.37	96.99	97.75	97.36
Customized [30]	94.00	95.10	93.10	94.10
ResNET-50 [30]	95.70	96.90	94.50	95.70
Customized [31]	96.29	98.04	92.34	94.95
TL-VGG16 [31]	97.77	97.19	97.20	97.09
Proposed	**99.96**	**100**	**99.92**	**99.96**

Another CNN-based model [31] applied very deep CNN architecture TL-VGG16 with 16 convolutional layers and customized with 8 convolutional layers for malarial parasite detection. This model [31] achieved 96.29% by customized approach and 97.77% by TL-VGG16. Very deep architecture requires more time for training as TL-VGG16 [31] converges at 80 epochs.

To get rid of bias and to reduce overfitting we applied 5-fold cross-validation toward optimal development of stacked CNN architecture. Results of 5-fold cross-validation is represented in Table 4 with 99.964% Accuracy, 100% Precision, 99.928% Recall and 99.964% F1-score. We present results of all 5 folds in Table 8 and also compares estimated accuracy after performing 5-fold cross-validation with other best performing models in the literature as shown in Table 9. Evidently in each fold, our model does not show any variance thus ensure robustness and generality.

Importance of Stain Normalization:

We also evaluate the performance of our stacked CNN architecture by training it to the blood smear images without

TABLE 8. 5-fold cross validation results.

K-Folds	Accuracy	Precision	Recall
Fold-1	97.927	100	99.856
Fold-2	99.915	99.976	99.855
Fold-3	99.964	99.976	99.951
Fold-4	99.867	99.734	100
Fold-5	99.988	100	99.976
5-Fold Mean	**99.964**	**100**	**99.928**
Standard Deviation	**0.001**	**0.001**	**0.001**

TABLE 9. Performance evaluation with best performing models in literature.

Model	Estimated Accuracy
ResNET-50 [30]	95.7±0.007
TL-VGG16 [31]	97.0±0.005
Proposed	**99.96±0.001**

applying stain normalization. Applying the CNN model directly to the dataset images gave a poor accuracy value of 49.61% accuracy. It is evident from Table 5 stain normalization remarkably improved the performance of our proposed model by 50% and reaching to 99.96%. Based on the results presented in Table 5 we investigate the importance of stain normalization in our classification task and we found it important step to include in training and evaluation of the proposed model.

VI. CONCLUSION

Traditional machine learning methods have shown limited accuracy for malarial parasite detection. Therefore, this work proposed a stacked CNN model-based on an end-to-end artificial neural network to improve malarial classification from thin blood smear images. The achieved results prove that with varying filter sizes and depth, convolutional layers can extract different abstract level features for classification. This study proves that features extracted by CNN are better than hand-crafted features. The stacked CNN model using stain normalization outperformed than state-of-the-art deep learning methods. Experimental results of 5-fold cross-validation confirm the superiority of the proposed Stacked CNN model with 99.96% accuracy. Our future direction entails further refine it to improve the classification accuracy within-subject or cross-subject.

REFERENCES

[1] H. Caraballo and K. King, "Emergency department management of mosquito-borne illness: Malaria, dengue, and west nile virus," *Emergency Med. Pract.*, vol. 16, no. 5, pp. 1–23, May 2014.
[2] World Health Organization. (2014). *World Health Organization. Regional for the Eastern Mediterranean*. Accessed: Sep. 22, 2019. [Online]. Available: https://www.who.int/ith/diseases/malaria/en/
[3] H. Wang, M. Naghavi, C. Allen, R. Barber, Z. Bhutta, A. Carter, D. Casey, F. Charlson, A. Chen, M. Coates, M. Coggeshall, L. Dandona, D. Dicker, H. Erskine, A. Ferrari, C. Fitzmaurice, K. Foreman, M. Forouzanfar, M. Fraser, and C. Murray, "Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: A systematic analysis for the global burden of disease study 2015," *LANCET*, vol. 388, no. 10053, p. 1447, Oct. 2016.
[4] K. S. Makhija, S. Maloney, and R. Norton, "The utility of serial blood film testing for the diagnosis of malaria," *Pathology*, vol. 47, no. 1, pp. 68–70, Jan. 2015.
[5] World Health Organization. (2016). *Malaria Microscopy Quality Assurance Manual*. Accessed: Sep. 22, 2019. [Online]. Available: http://apps.who.int/medicinedocs/en/m/abstract/Js19135en/
[6] M. Poostchi, K. Silamut, R. J. Maude, S. Jaeger, and G. Thoma, "Image analysis and machine learning for detecting malaria," *Transl. Res.*, vol. 194, pp. 36–55, Apr. 2018.
[7] S. Rajaraman, K. Silamut, M. A. Hossain, I. Ersoy, R. J. Maude, S. Jaeger, G. R. Thoma, and S. K. Antani, "Understanding the learned behavior of customized convolutional neural networks toward malaria parasite detection in thin blood smear images," *J. Med. Imag.*, vol. 5, no. 3, p. 1, Jul. 2018.
[8] M. Ahmad, A. M. Khan, and R. Hussain, "Graph-based spatial–spectral feature learning for hyperspectral image classification," *IET Image Process.*, vol. 11, no. 12, pp. 1310–1316, Dec. 2017.
[9] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Apr. 2014.
[10] M. Ahmad, "A fast 3D CNN for hyperspectral image classification," 2020, *arXiv:2004.14152*. [Online]. Available: https://arxiv.org/abs/2004.14152
[11] M. Ahmad, S. Shabbir, D. Oliva, M. Mazzara, and S. Distefano, "Spatial-prior generalized fuzziness extreme learning machine autoencoder-based active learning for hyperspectral image classification," *Optik*, vol. 206, Mar. 2020, Art. no. 163712.
[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, vol. 25, Jan. 2012, pp. 1097–1105.
[13] M. Ahmad, A. Khan, A. M. Khan, M. Mazzara, S. Distefano, A. Sohaib, and O. Nibouche, "Spatial prior fuzziness pool-based interactive classification of hyperspectral images," *Remote Sens.*, vol. 11, no. 9, p. 1136, May 2019.
[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[17] M. Ahmad, M. A. Alqarni, A. M. Khan, R. Hussain, M. Mazzara, and S. Distefano, "Segmented and non-segmented stacked denoising autoencoder for hyperspectral band reduction," *Optik*, vol. 180, pp. 370–378, Feb. 2019.
[18] F. Bousetouane and B. Morris, "Off-the-shelf CNN features for fine-grained classification of vessels in a maritime environment," in *Advances in Visual Computing*. Cham, Switzerland: Springer, Dec. 2015, pp. 379–388.
[19] K. Suzuki, "Overview of deep learning in medical imaging," *Radiol. Phys. Technol.*, vol. 10, no. 3, pp. 257–273, Sep. 2017.
[20] G. Díaz, F. A. González, and E. Romero, "A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images," *J. Biomed. Informat.*, vol. 42, no. 2, pp. 296–307, Apr. 2009.
[21] 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, Feb. 2013.
[22] H. Shen, W. David Pan, Y. Dong, and M. Alim, "Lossless compression of curated erythrocyte images using deep autoencoders for malaria infection diagnosis," in *Proc. Picture Coding Symp. (PCS)*, 2016, pp. 1–5.
[23] F. B. Tek, A. G. Dempster, and I. Kale, "Parasite detection and identification for automated thin blood film malaria diagnosis," *Comput. Vis. Image Understand.*, vol. 114, no. 1, pp. 21–32, Jan. 2010.
[24] H. S. Park, M. T. Rinehart, K. A. Walzer, J.-T.-A. Chi, and A. Wax, "Automated detection of P. Falciparum using machine learning algorithms with quantitative phase images of unstained cells," *PLoS ONE*, vol. 11, no. 9, 2016, Art. no. e0163045.
[25] W. A. Mustafa, R. Santiagoo, I. Jamaluddin, N. S. Othman, W. Khairunizam, and M. N. K. H. Rohani, "Comparison of detection method on malaria cell images," in *Proc. Int. Conf. Comput. Approach Smart Syst. Design Appl. (ICASSDA)*, Aug. 2018, pp. 1–6.
[26] Z. Liang, A. Powell, I. Ersoy, M. Poostchi, K. Silamut, K. Palaniappan, P. Guo, M. A. Hossain, A. Sameer, R. J. Maude, J. X. Huang, S. Jaeger, and G. Thoma, "CNN-based image analysis for malaria diagnosis," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2016, pp. 493–496.

[27] 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.
[28] G. P. Gopakumar, M. Swetha, G. Sai Siva, and G. R. K. Sai Subrahmanyam, "Convolutional neural network-based malaria diagnosis from focus stack of blood smear images acquired using custom-built slide scanner," *J. Biophotonics*, vol. 11, no. 3, Mar. 2018, Art. no. e201700003.
[29] C. Mehanian *et al.*, "Computer-automated malaria diagnosis and quantitation using convolutional neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 116–125.
[30] S. Rajaraman, S. K. Antani, M. Poostchi, K. Silamut, M. A. Hossain, R. J. Maude, S. Jaeger, and G. R. Thoma, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, Apr. 2018, Art. no. e4568.
[31] A. Rahman, H. Zunair, M. S. Rahman, J. Q. Yuki, S. Biswas, M. A. Alam, N. B. Alam, and M. R. C. Mahdy, "Improving malaria parasite detection from red blood cell using deep convolutional neural networks," 2019, *arXiv:1907.10418*. [Online]. Available: https://arxiv.org/abs/1907.10418
[32] D. Bradley and G. Roth, "Adaptive thresholding using the integral image," *J. Graph. Tools*, vol. 12, no. 2, pp. 13–21, Jan. 2007.
[33] C. Wolf and J.-M. Jolion, "Extraction and recognition of artificial text in multimedia documents," *Formal Pattern Anal. Appl.*, vol. 6, no. 4, pp. 309–326, Feb. 2004.
[34] J. Bernsen, "Dynamic thresholding of grey-level images," in *Proc. 8th Int. Conf. Pattern Recognit.*, 1986, pp. 1251–1255.
[35] Y. Li, W. Jin, J. Zhu, X. Zhang, and S. Li, "An adaptive deghosting method in neural network-based infrared detectors nonuniformity correction," *Sensors*, vol. 18, no. 2, p. 211, Jan. 2018.
[36] H. Liang and C. Zhenming, "Automatic detection of sister chromatid exchange," *J. Histochem. Cytochem.*, vol. 2, pp. 652–654, Jan. 1988.
[37] S. Aja-Fernández, A. H. Curiale, and G. Vegas-Sánchez-Ferrero, "A local fuzzy thresholding methodology for multiregion image segmentation," *Knowl.-Based Syst.*, vol. 83, pp. 1–12, Jul. 2015.
[38] S. Wen, W. Liu, Y. Yang, T. Huang, and Z. Zeng, "Generating realistic videos from keyframes with concatenated GANs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 8, pp. 2337–2348, Aug. 2019.
[39] B. Sun, Y. Cao, Z. Guo, Z. Yan, and S. Wen, "Synchronization of discrete-time recurrent neural networks with time-varying delays via quantized sliding mode control," *Appl. Math. Comput.*, vol. 375, Jun. 2020, Art. no. 125093.
[40] Y. Cao, S. Wang, Z. Guo, T. Huang, and S. Wen, "Synchronization of memristive neural networks with leakage delay and parameters mismatch via event-triggered control," *Neural Netw.*, vol. 119, pp. 178–189, Nov. 2019.
[41] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 3, pp. 328–339, Mar. 1989.
[42] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[43] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, 2003, pp. 958–963.
[44] M. Dong, S. Wen, Z. Zeng, Z. Yan, and T. Huang, "Sparse fully convolutional network for face labeling," *Neurocomputing*, vol. 331, pp. 465–472, Feb. 2019.
[45] C. Monrocq and Y. Lecun, "An original approach for the localization of objects in images," *IEE Proc.-Vis., Image Signal Process.*, vol. 141, no. 4, pp. 245–250, Aug. 1998.
[46] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[47] F. Ciompi, O. Geessink, B. E. Bejnordi, G. S. de Souza, A. Baidoshvili, G. Litjens, B. van Ginneken, I. Nagtegaal, and J. van der Laak, "The importance of stain normalization in colorectal tissue classification with convolutional networks," in *Proc. IEEE 14th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2017, pp. 160–163.
[48] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. F. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *Proc. 13th Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, Mar. 2015, pp. 844–848.
[49] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain tumor segmentation using convolutional neural networks in MRI images," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1240–1251, May 2016.
[50] Y. Bar, I. Diamant, L. Wolf, S. Lieberman, E. Konen, and H. Greenspan, "Chest pathology detection using deep learning with non-medical training," in *Proc. IEEE 12th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2015, pp. 294–297.
[51] V. Liu, M. P. Clark, M. Mendoza, R. Saket, M. N. Gardner, B. J. Turk, and G. J. Escobar, "Automated identification of pneumonia in chest radiograph reports in critically ill patients," *BMC Med. Informat. Decis. Making*, vol. 13, no. 1, p. 90, Dec. 2013.
[52] S. Kannan, L. A. Morgan, B. Liang, M. G. Cheung, C. Q. Lin, D. Mun, R. G. Nader, M. E. Belghasem, J. M. Henderson, J. M. Francis, V. C. Chitalia, and V. B. Kolachalama, "Segmentation of glomeruli within trichrome images using deep learning," *Kidney Int. Rep.*, vol. 4, no. 7, pp. 955–962, Jul. 2019.
[53] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," 2014, *arXiv:1403.6382*. [Online]. Available: http://arxiv.org/abs/1403.6382

MUHAMMAD UMER received the B.S. degree from the Department of Computer Science, Khwaja Fareed University of Engineering and IT (KFUEIT), Pakistan, in October 2018, where he is currently pursuing the Master of Computer Science degree. He has been serving as a Research Assistant with the Fareed Computing and Research Center, KFUEIT. His recent research interests include data mining, mainly working machine learning and the deep learning-based IoT, text mining, and computer vision tasks.

SAIMA SADIQ is currently pursuing the Master of Computer Science degree with the Khwaja Fareed University of Engineering and IT (KFUEIT). She has been working as an Assistant Professor with the Department of Computer Science, Government Degree College for Women. Her recent research interests include data mining, machine learning, and deep learning-based text mining.

MUHAMMAD AHMAD is currently an Assistant Professor with the Department of Computer Engineering, Khwaja Freed University of Engineering and Information Technology, Pakistan. He is associated with the research group of Advanced Image Processing Research Laboratory (AIPRL), a first Hyperspectral imaging lab in Pakistan. He is also associated with the University of Messina, Messina, Italy, as a Research Fellow. He has authored a number of research papers in reputed journals and conferences. He is a Regular Reviewer for a number of top tier Transactions/journals and conferences. His current research interests include machine learning, computer vision, remote sensing, hyperspectral imaging, and wearable computing.

SALEEM ULLAH was born in AhmedPur East, Pakistan, in 1983. He received the B.Sc. degree from The Islamia University of Bahawalpur, Pakistan, in 2003, the M.I.T. degree in computer science from Bahauddin Zakariya University, Multan, in 2005, and the Ph.D. degree from Chongqing University, China, in 2012. From 2006 to 2009, he worked as a Network/IT Administrator in different companies. From August 2012 to February 2016, he has worked as an Assistant Professor with The Islamia University of Bahawalpur. He is currently working as the Associate Dean with the Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan, since February 2016. He has almost 14 years of Industry experience in field of IT. He is an Active Researcher in the field of adhoc networks, IoTs, congestion control, data science, and network security.

GYU SANG CHOI received the Ph.D. degree from the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, in 2005. He was a Research Staff Member with the Samsung Advanced Institute of Technology (SAIT), Samsung Electronics, from 2006 to 2009. Since 2009, he has been a Faculty Member with the Department of Information and Communication, Yeungnam University, South Korea. His research interests include non-volatile memory and storage systems.

ARIF MEHMOOD received the Ph.D. degree from the Department of Information and Communication Engineering, Yeungnam University, South Korea, in 2017. From December 2017 to November 2019, he served as an Assistant Professor with the Information Technology Department, Khawaja Fareed University of Engineering and Information Technology. He is currently serving as an Assistant Professor with Islamia University Bahawalpur, Pakistan. He is working mainly in deep learning-based text mining and data science management techniques.

ORIGINALITY REPORT

a15

ORIGINALITY REPORT

9%	5%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	3%
2	Muhammad Umer, Saima Sadiq, Muhammad Ahmad, Saleem Ullah, Gyu Sang Choi, Arif Mehmood. "A Novel Stacked CNN for Malarial Parasite Detection in Thin Blood Smear Images", IEEE Access, 2020 Publication	2%
3	Srinivasan Sankaran, Muthukumaran Malarvel, Gopalakrishnan Sethumadhavan, Dinkar Sahal. "Quantitation of Malarial parasitemia in Giemsa stained thin blood smears using Six Sigma threshold as preprocessor", Optik - International Journal for Light and Electron Optics, 2017	2%