REVOLUTIONIZING MALARIA DIAGNOSIS – A DEEP LEARNING APPROACH

Project Report

Submitted to the Faculty of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

In partial fulfilment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

Artificial Intelligence and Data Science

By

R. Priya Darshini
20481A5444

G. LikithaSai
20481A5419

K. Harika
20481A5425

M. Mohitha
20481A5436

Under the Supervision of

Mr. K. Ashok Reddy

Assistant Professor, Department of AI&DS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

SESHADRIRAO KNOWLEDGE VILLAGE

GUDLAVALLERU – 521 356

ANDHRA PRADESH

2022-2023

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

SHESHADRI RAO KNOWLEDGE VILLAGE

GUDLAVALLERU-521356

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CERTIFICATE

This is to certify that the project report entitled **"REVOLUTIONIZING MALARIA DIAGNOSIS – A DEEP LEARNING APPROACH"** is a bonafide record of work carried out by **R. Priya Darshini (20481A5444), G. LikithaSai (20481A5419), K. Harika (20481A5425), M. Mohitha (20481A5436)** under the guidance and supervision of **Mr. K. Ashok Reddy M.Tech, (PhD)** partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Artificial Intelligence and Data Science** of **Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru.**

Project Guide	**Head of the Department**
Mr. K. Ashok Reddy **Assistant Professor**	**Dr. K. Srinivas** **Professor & H.O.D**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragements crown all the efforts with success.

We would like to express our deep sense of gratitude and sincere thanks to **Mr. K. Ashok Reddy M. Tech**, (PhD) Assistant professor, Department of Artificial Intelligence and Data Science for his constant guidance, supervision and motivation in completing the project work.

We feel elated to express our floral gratitude and sincere thanks to **Dr. K. Srinivas,** Head of the Department, Artificial Intelligence and Data Science, for his encouragement all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the project work.

We would like to take this opportunity to thank our beloved principal **Dr. G. V. S. N. R. V. Prasad** for providing great support for us in completing our project and giving us the opportunity to do the project.

Our Special thanks to the faculty of our department and programmers of our computer lab. Finally, we thank our family members, non-teaching staff and our friends, who have directly or indirectly helped and supported us in completing our project in time.

By-

R. PRIYA DARSHINI (20481A5444)

G. LIKITHA SAI (20481A5419)

K. HARIKA (20481A5425)

M. MOHITHA (20481A5436)

ABSTRACT

Malaria is a hazardous disease affecting millions worldwide, and early, accurate diagnosis is crucial for effective treatment. This paper presents an automated approach for malaria detection using cell images with the VGG19 algorithm. Malaria is a life-threatening disease that affects millions of people worldwide, and early and accurate diagnosis is crucial for effective treatment. The proposed method involves training the VGG19 algorithm on a large dataset of cell images to identify the presence of malaria parasites. The results demonstrate the effectiveness of the proposed approach, achieving an accuracy of 95% in the detection of malaria-infected cells. The proposed method has the potential to improve the efficiency of malaria diagnosis, particularly in resource-limited settings where access to trained medical professionals may be limited.

INDEX

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Anopheles mosquito bites from females carrying the Plasmodium parasite, which causes malaria, can spread the disease to humans. The World Health Organization (WHO) estimates that malaria killed 409,000 people in 2019 and affected 229 million worldwide [1]. In order to lessen the impact of malaria, early detection and prompt treatment are crucial. Implementing the traditional diagnosis method in areas with limited resources is difficult because it takes time and requires trained specialists to visually examine blood smears under a microscope.

Recent developments in Machine Learning(ML) and Artificial Intelligence(AI) can potentially increase the precision and effectiveness of malaria diagnosis using cell images. Using the VGG19 algorithm, a Convolution Neural Networks(CNN) successfully used in several image classification tasks; we propose an automated method for detecting malaria in this paper. By analysing cell images, the suggested method offers a quick and accurate diagnosis of malaria.

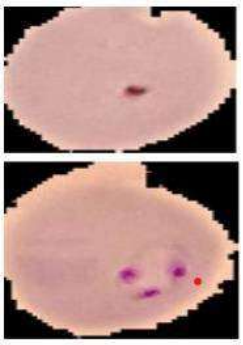


Fig. 1. Parasitized

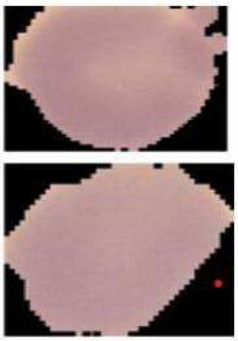


Fig. 2. Uninfected

The above figure1 and figure2 show the parasitized and uninfected samples, respectively. The approach involves training the VGG19 algorithm on a large dataset of cell images to identify the malaria parasites. The algorithm's ability to learn features from images and classify them accurately makes it a suitable candidate for malaria detection.

1.2 OBJECTIVES OF THE PROJECT

- Develop a deep learning model using convolutional neural networks (CNNs) to accurately classify microscopic images of blood smears as either malaria-positive or malaria-negative.
- Evaluate the performance of the deep learning model on a large dataset of blood smear images, using metrics such as sensitivity, specificity, and accuracy.
- Compare the performance of the deep learning model with traditional methods of malaria diagnosis, such as microscopic examination of blood smears, to assess the potential of the deep learning model for improving malaria diagnosis.

- Investigate the impact of various factors on the performance of the deep learning model, such as the size of the training dataset, the architecture of the CNN, and the choice of hyperparameters.
- Develop an optimized deep learning model that achieves high accuracy on malaria diagnosis, while also being efficient and computationally feasible for deployment in resource-limited settings.
- Validate the optimized deep learning model on a separate dataset of blood smear images, to ensure the generalizability of the model.
- Conduct a usability study to assess the acceptability and feasibility of the deep learning model in clinical settings, including factors such as ease of use, accuracy, and time required for diagnosis.

By achieving these objectives, the malaria diagnosis deep learning project aims to develop an accurate, reliable, and efficient system for malaria diagnosis that can be used in low-resource settings to improve patient outcomes and support malaria control efforts.

1.3 PROBLEM STATEMENT

Malaria is a life-threatening disease that affects millions of people every year, particularly in low-resource settings. Rapid and accurate diagnosis is critical for effective treatment and disease control. However, traditional methods of malaria diagnosis, such as microscopic examination of blood smears, can be time-consuming, labor intensive, and prone to human error. Therefore, there is a need for an automated and reliable system for malaria diagnosis.

Deep learning is a branch of machine learning that has shown promising results in image analysis tasks, including medical image diagnosis. In this project, we aim to develop a deep learning system for automated malaria diagnosis using microscopic images of blood smears. Specifically, we will investigate the use of convolutional neural networks (CNNs) to classify blood smear images as either malaria-positive or malaria-negative.

The proposed deep learning system has the potential to improve the speed and accuracy of malaria diagnosis, particularly in resource-limited settings where trained personnel and equipment may be scarce. The development of such a system could have significant implications for malaria control efforts, including early detection, prompt treatment, and improved patient outcomes.

CHAPTER 2

LITERATURE REVIEW

Various approaches for malaria detection using cell images have been proposed, including machine learning, deep learning, and ensemble methods. In this section, we provide an overview of some of the related works in this field.

Malaria detection has been carried out using conventional ML models like decision trees and Support Vector Machine(SVM). For instance, [2] suggested using SVM with texture features extracted from cell images to categorize cells infected with malaria. Their method had an accuracy rate of 83.4%. Similarly, Adedokun, [3] proposed a rule-based system with an accuracy rate of 89.4%.

Machine learning approaches for malaria detection typically involve hand-crafted feature extraction and classification using algorithms such as SVM or decision trees. For example, a study by Singh et al. (2014) used SVM to classify malaria-infected and uninfected cells based on texture and shape features. The study achieved an accuracy of 92.8% in the detection of infected cells [4].

Machine learning approaches for malaria detection typically involve hand-crafted feature extraction and classification using algorithms such as SVM or decision trees. For example, a study by Singh et al. (2014) used SVM to classify malaria-infected and uninfected cells based on texture and shape features. The study achieved an accuracy of 92.8% in the detection of infected cells [4].

Ensemble methods have also been used in malaria detection, combining multiple models to improve the accuracy of the diagnosis. For instance, Das et al. [5] proposed an ensemble of SVM classifiers, which achieved an accuracy of 92.4% in malaria detection.

The deep CNN model VGG19 architecture, developed by Simonyan and Zisserman (2015), has been extensively applied to various image classification tasks. To the best of our knowledge, little research has been done on using the VGG19 algorithm to find malaria in cell images.ML and DL algorithms, particularly CNN, have shown great promise in detecting malaria parasites in cell images. Transfer learning using pre-trained CNN models has also been explored for malaria detection.

CNN and feed-forward neural networks, both shallow neural networks, have also been applied to the detection of malaria. A feed-forward neural network with a single hidden layer and 91.5% accuracy was suggested in a study by [6]. A CNN with two convolution layers and one dense layer was employed in a different study by [7] to classify malaria-infected cells with a 93% accuracy rate.

Rule-based systems have also been proposed for malaria detection, where rules are defined to classify cells based on their features. One such study by [8] used a rule-based system to classify cells based on their color and texture features with an accuracy of 92.5%.Despite these efforts, automated malaria detection using cell images remains challenging due to the high variability in cell appearance and the low parasitaemia level in some cases. In the next section, we present the proposed methodology for malaria detection using cell images with the VGG19 algorithm.

CHAPTER 3

PROPOSED METHOD

3.1 METHODOLOGY

Data Representation:

Our proposed methodology for malaria detection using cell images involves the following process. We use the VGG19 (Visual Geometry Group 19) convolutional neural network architecture as the base model for feature extraction due to its high accuracy and efficiency in image classification tasks.

Transfer learning is used to refine the pre-trained VGG19 model on our dataset, with the earlier layers' weights being frozen and only the later layers being trained. Figure 3 depicts the architecture of our model and its layers. The performance of the trained model is evaluated on a separate test set, and the accuracy, precision, recall, and F1 score are computed.

CNN:

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes. The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image.

CNNs are trained using a large dataset of labelled images, where the network learns to recognize patterns and features that are associated with specific objects or classes. Once trained, a CNN can be used to classify new images, or extract features for use in other applications such as object detection or image segmentation.

CNNs have achieved state-of-the-art performance on a wide range of image recognition tasks, including object classification, object detection, and image segmentation. They are widely used in computer vision, image processing, and other related fields, and have been applied to a wide range of applications, including self-driving cars, medical imaging, and security systems.

Convolutional Neural Network Design:

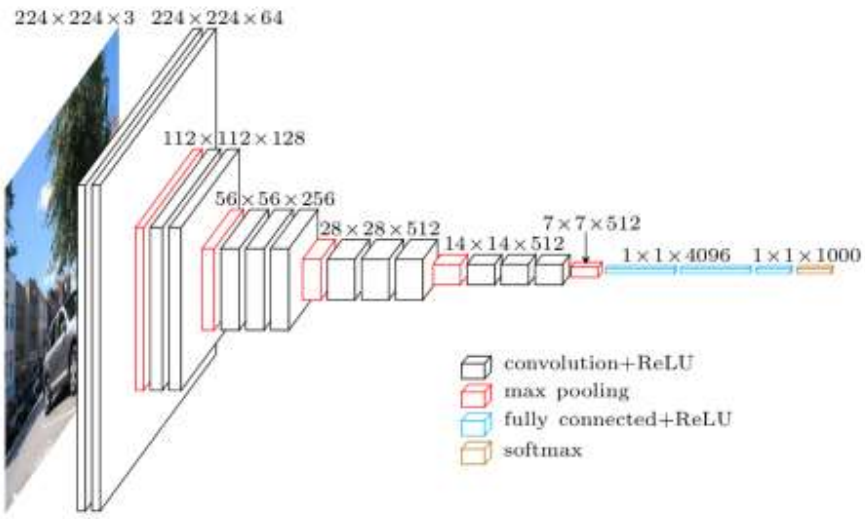
- The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order.
- It is the sequential design that give permission to CNN to learn hierarchical attributes.

- In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.
- The pre-processing needed in a Conv Net is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

Different Types of CNN Models:

1. LeNet
2. AlexNet
3. ResNet
4. GoogleNet
5. MobileNet
6. VGG

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The "deep" refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers. The VGG architecture is the basis of ground-breaking object recognition models. Developed as a deep neural network, the VGGNet also surpasses baselines on many tasks and datasets beyond ImageNet. Moreover, it is now still one of the most popular image recognition architectures.



What is VGG16?

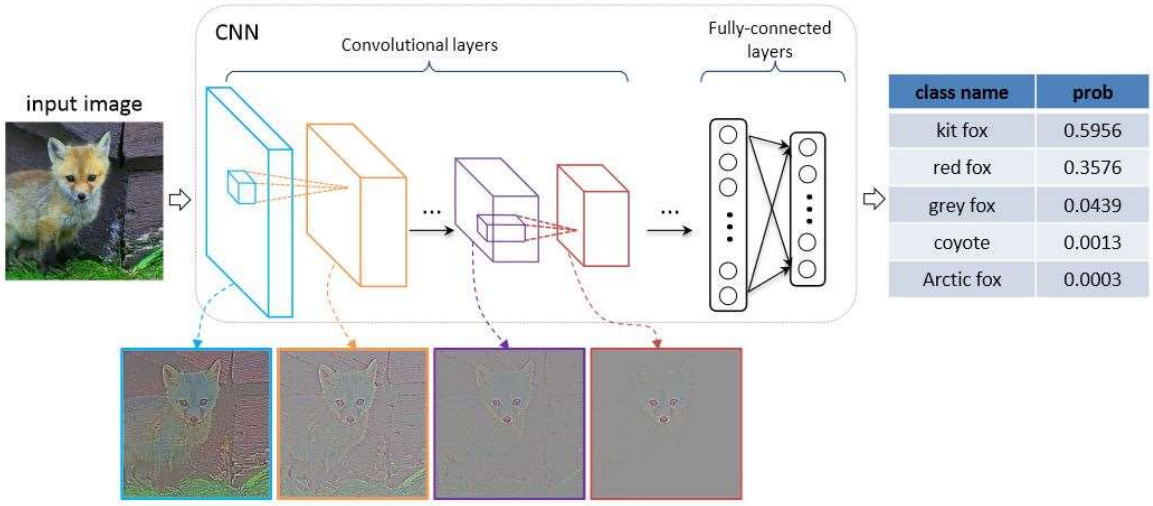
The VGG model, or VGG Net, that supports 16 layers is also referred to as VGG16, which is a convolutional neural network model proposed by A. Zisserman and K. Simonyan from the University of Oxford. These researchers published their model in the research paper titled, "Very Deep Convolutional Networks for Large-Scale Image Recognition."

The VGG16 model achieves almost 92.7% top-5 test accuracy in ImageNet. ImageNet is a dataset consisting of more than 14 million images belonging to nearly 1000 classes. Moreover, it was one

of the most popular models submitted to ILSVRC-2014. It replaces the large kernel-sized filters with several 3×3 kernel-sized filters one after the other, thereby making significant improvements over AlexNet. The VGG16 model was trained using Nvidia Titan Black GPUs for multiple weeks.

The concept of the VGG19 model (also VGGNet-19) is the same as the VGG16 except that it supports 19 layers. The "16" and "19" stand for the number of weight layers in the model (convolutional layers). This means that VGG19 has three more convolutional layers than VGG16.

VGG ARCHITECTURE:



VGG 19:

Alex Net came out in 2012 and it improved on the traditional Convolutional neural networks, So we can understand **VGG as a successor of the Alex Net** but it was created by a different group named as **Visual Geometry Group** at **Oxford**'s and hence the name VGG, It carries and uses some ideas from it is predecessors and improves on them and uses deep Convolutional neural layers to improve accuracy

So, in simple language VGG is a deep CNN used to classify images. The layers in VGG19 model are as follows:

- Conv3x3 (64)
- Conv3x3 (64)
- Max Pool

- Conv3x3 (128)
- Conv3x3 (128)
- Max Pool
- Conv3x3 (256)
- Conv3x3 (256)
- Conv3x3 (256)
- Conv3x3 (256)
- Max Pool
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Max Pool
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Max Pool
- Fully Connected (4096)
- Fully Connected (4096)
- Fully Connected (1000)
- SoftMax

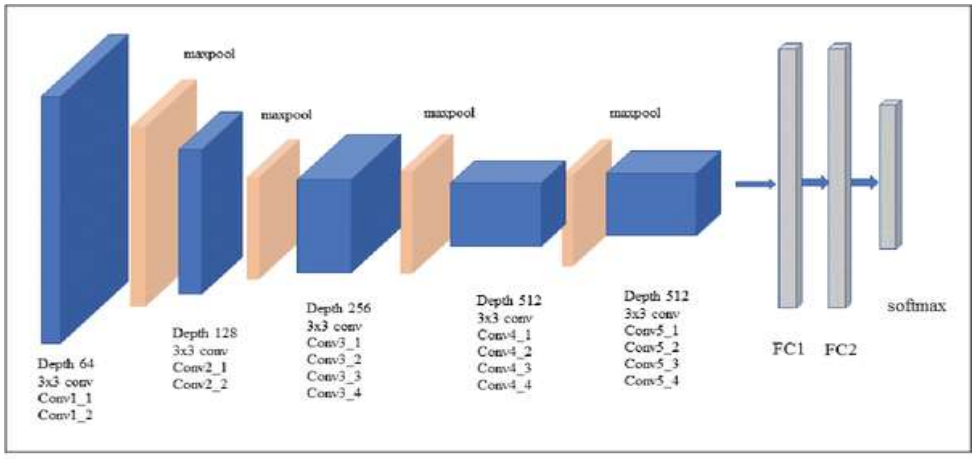


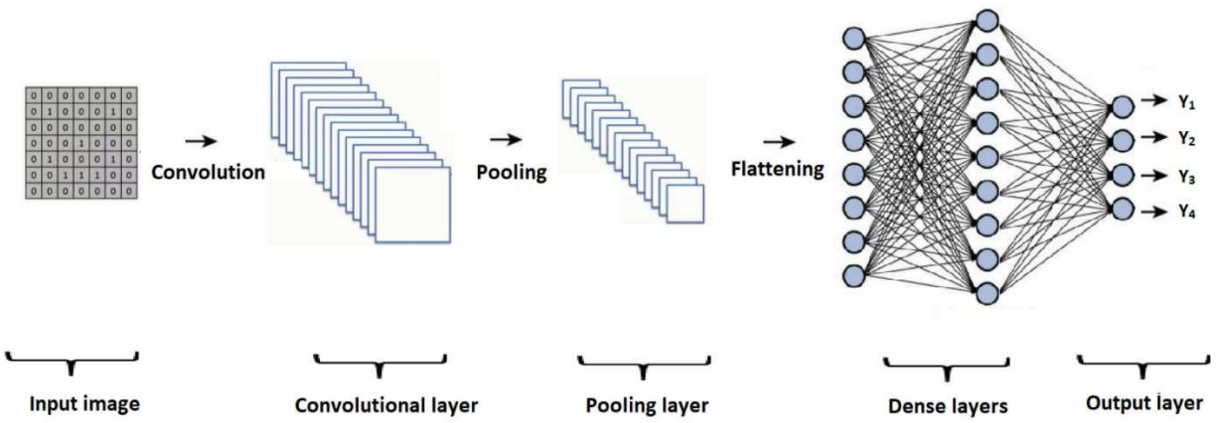
Fig. 3. VGG-19 network architecture

Architecture

- Fixed size of (224 * 224) RGB image was given as input to this network which means that the matrix was of shape (224,224,3).
- The only preprocessing that was done is that they subtracted the mean RGB value from each pixel, computed over the whole training set.
- Used kernels of (3 * 3) size with a stride size of 1 pixel, this enabled them to cover the whole notion of the image.
- spatial padding was used to preserve the spatial resolution of the image.
- max pooling was performed over a 2 * 2 pixel windows with sride 2.
- this was followed by Rectified linear unit(ReLu) to introduce non-linearity to make the model classify better and to improve computational time as the previous models used tanh or sigmoid functions this proved much better than those.
- implemented three fully connected layers from which the first two were of size 4096 and after that, a layer with 1000 channels for 1000-way *ILSVRC* classification and the final layer is a softmax function.

The column E in the following table is for VGG19 (other columns are for other variants of VGG models):

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					



Pros:

1. **Accuracy:** VGG19 has achieved state-of-the-art results on the ImageNet dataset and has been used as a benchmark model for image classification tasks.
2. **Transfer Learning:** VGG19 has a large number of pre-trained models available, making it easy to use for transfer learning in other computer vision tasks.
3. **Simple architecture:** The VGG19 architecture is relatively simple, making it easy to understand and implement.
4. **Feature extraction:** The VGG19 model learns to extract rich features from the images, which can be useful in other computer vision tasks.

Cons:

1. **Large model size:** VGG19 has a large number of parameters, which can make it computationally expensive to train and use.
2. **Limited to image classification:** VGG19 is primarily used for image classification tasks, and may not perform as well in other computer vision tasks.
3. **Limited interpretability:** Due to the complex nature of deep learning models, it can be difficult to understand how VGG19 arrives at its classifications.
4. **Limited flexibility:** VGG19 has a fixed architecture, which may not be suitable for all computer vision tasks, and may require modifications or customizations.

The system's primary objective is to predict Malaria disease in various patients based on a variety of characteristics. The Malaria dataset, which consists of Parasite and uninfected categories, has been used to develop the system. The images of patients impacted by malaria are in the parasite folder. The images of a patient not affected by malaria are in an uninfected folder.

The dataset utilized here consists of cell images from thin blood smears of individuals infected with malaria and healthy individuals. The images are pre-processed to normalize the intensity values and resize the images to a fixed size.The 416 images in the malaria dataset are a small number in the machine learning community. The dataset is divided into folders called Parasite and Uninfected. We extracted the images from both folders and used them as training and testing datasets.

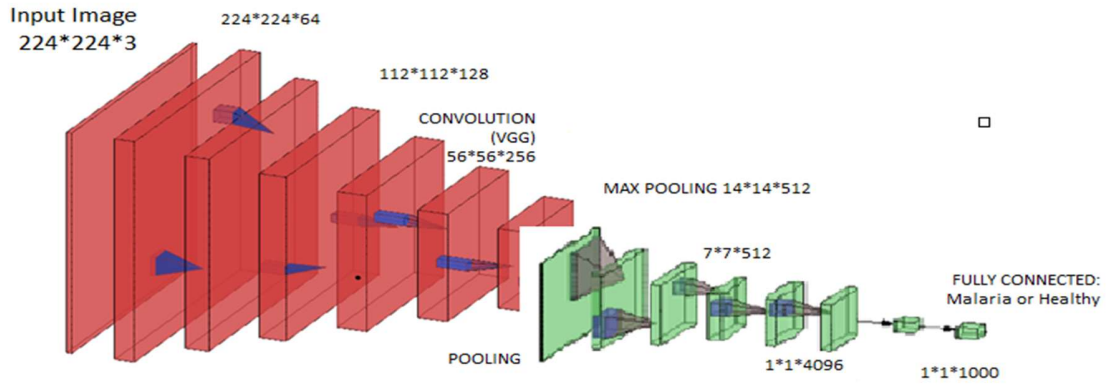
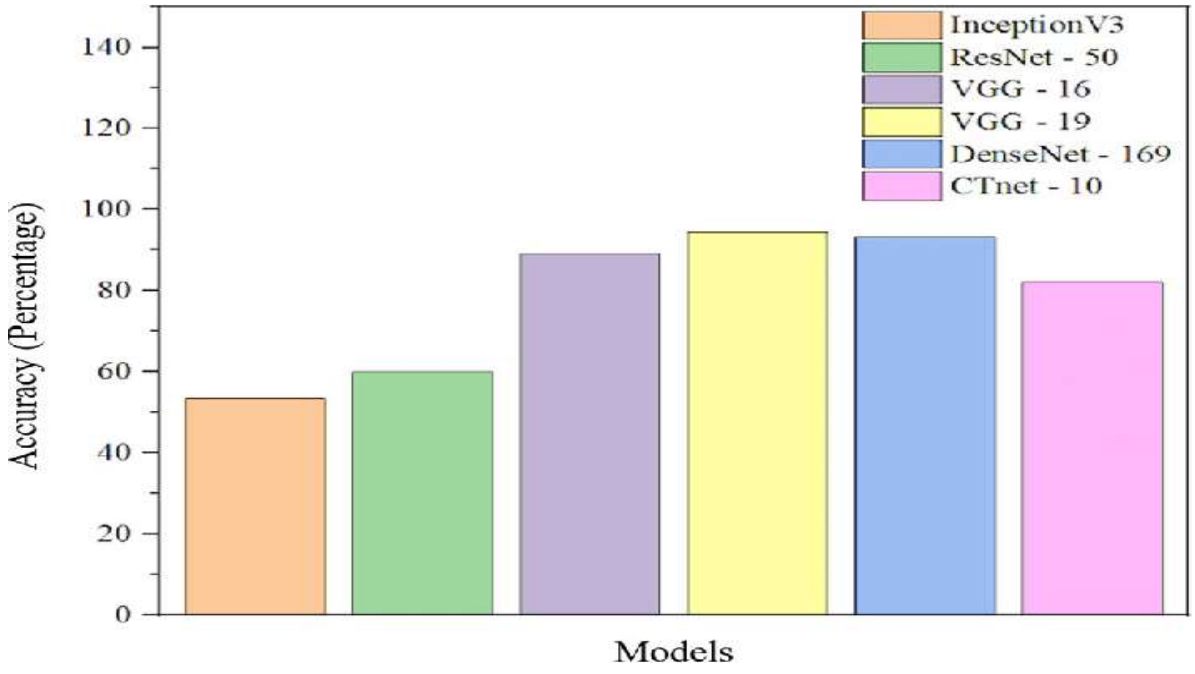


Fig. 3. Architecture

1. **Input layer:** The input layer takes in the input image, which is typically a 224x224x3 RGB image.
2. **Convolutional layers:** There are 16 convolutional layers, each of which performs a convolution operation on the input image. The first two convolutional layers use 3×3 filters, while the remaining layers use 3×3 filters with padding to preserve the spatial dimensions of the input.
3. **Max pooling layers:** There are 5 max pooling layers, which reduce the spatial dimensions of the feature maps.
4. **Fully connected layers:** There are 3 fully connected layers, which perform a matrix multiplication operation on the feature maps to produce the final output. The first two fully connected layers have 4096 neurons each, while the last fully connected layer has 1000 neurons, representing the 1000 classes in the ImageNet dataset.
5. **Softmax activation:** The output of the final fully connected layer is passed through a softmax activation function, which produces a probability distribution over the 1000 classes in the ImageNet dataset

Compare to other models of Convolution Neural Networks, VGG19 model gives the highest percentage of accuracy as shown in below figure:



Evaluation metrics:

Figure 5 shows the confusion matrix in that format. The confusion matrix is a performance gauge for machine learning classification models. The effectiveness of the built-in models was evaluated using the confusion matrix [9]. The confusion matrix displays the ratio of correctly predicted events to incorrect guesses made by our models. It distinguished between true positives and negatives for values correctly predicted, and false positives and false negatives for values incorrectly predicted. The accuracy, precision-recall trade-off, and AUC were used to assess the model's performance after all predicted values were organized in the matrix.

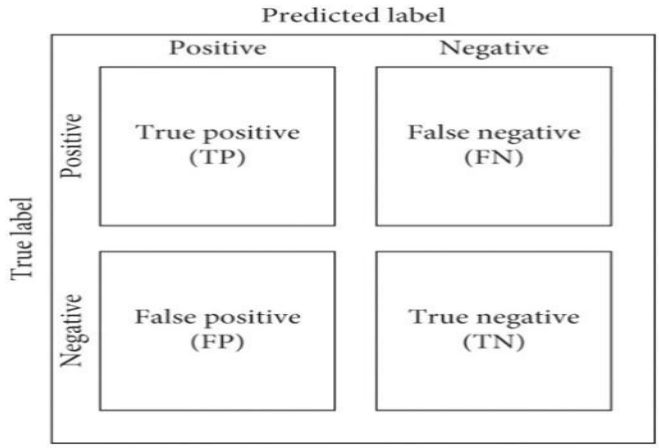


Fig. 5. Evaluation Metrics

- **True Positive (TP):** It is the total counts having both predicted and actual values.
- **True Negative (TN):** It is the total counts having both predicted and actual values.
- **False Positive (FP):** It is the total counts having prediction is correct while actually incorrect
- **False Negative (FN):** It is the total counts having prediction is incorrect while actually, it is correct.

By using confusion matrix we can calculate F1_score, Precision, Recall by using the below formulas:

Accuracy:

$$\frac{\text{True Positives + True Negatives}}{\text{True Positives + True Negatives + False Positives + False Negatives}} = \frac{\text{N. of Correct Predictions}}{\text{N. of all Predictions}} = \frac{\text{N. of Correct Predictions}}{\text{Size of Dataset}}$$

Precision:

$$\frac{\text{True Positives}}{\text{True Positives + False Positives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Predictions you Made}} = \frac{\text{N. of Correctly Predicted People with Cancer}}{\text{N. of People you Predicted to have Cancer}}$$

Recall/Sensitivity:

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Instances in the Dataset}} = \frac{\text{N. of Correctly Predicted People with Cancer}}{\text{N. of People with Cancer in the Dataset}}$$

Specificity:

$$\frac{\text{True Negatives}}{\text{True Negatives + False Positives}} = \frac{\text{N. of Correctly Predicted Negative Instances}}{\text{N. of Total Negative Instances in the Dataset}} = \frac{\text{N. of Correctly Predicted People with no Cancer}}{\text{N. of People with no Cancer in the Dataset}}$$

F1_score:

$$2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.2 IMPLEMENTATION

USED LIBRARIES:

- numpy
- pandas
- matplotlib
- tensorflow
- SciPy
- sklearn
- scikit-learn
- opencv-python
- tensorflow

Numpy:
NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.
NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.In Python we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.
This behavior is called locality of reference in computer science.

This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

Pandas:
Pandas is a Python library used for working with data sets.
It has functions for analyzing, cleaning, exploring, and manipulating data.Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.Relevant data is very important in data science.Pandas gives you answers about the data. Like:

Is there a correlation between two or more columns?
What is average value?
Max value?
Min value?
Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

Matplotlib:
Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.

Tensorflow:
TensorFlow is an open-source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand

framework. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. It can run on single CPU systems and GPUs, as well as mobile devices and large-scale distributed systems of hundreds of machines.

SciPy:

SciPy is a scientific computation library that uses NumPy underneath.
SciPy stands for Scientific Python.
It provides more utility functions for optimization, stats and signal processing.
Like NumPy, SciPy is open source so we can use it freely.

SciPy was created by NumPy's creator Travis Olliphant.SciPy has optimized and added functions that are frequently used in NumPy and Data Science.It provides many user-friendly and effective numerical functions for numerical integration and optimization.

Sklearn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Opencv-python: OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.
In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

INSTALL PACKAGES

```
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install keras
!pip install tensorflow
!pip install SciPy
!pip install sklearn
!pip install scikit-learn
!pip install opencv-python
```

IMPORT PACKAGES

```
import numpy as np
import pandas as pdz
import matplotlib.pyplot as plt
import os
import keras
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.applications.vgg19 import VGG19,preprocess_input
,decode_predictions
```

LOAD THE TRAIN DATASET

```
dir="train"
fname=os.listdir(dir)
print(fname)
```

 # Image Data Generator class allows your model to receive new variations of the images at each epoch

```
train_datagen=ImageDataGenerator(zoom_range=0.5,shear_range=0.3,horizontal_flip=
True,preprocessing_function = preprocess_input)
val_datagen = ImageDataGenerator(preprocessing_function= preprocess_input)
```

 # The flow_from_directory() method allows you to read the images directly from the directory and augment them while the neural network model is learning on the training data.

```
train = train_datagen.flow_from_directory(directory= dir,target_size=(256,256),batch_size=32)
val = val_datagen.flow_from_directory(directory= dir,target_size=(256,256),batch_size=32)


t_img , label =train.next()
```

#base_model generation

```
base_model=VGG19(input_shape=(256,256,3), include_top= False)

for layer in base_model.layers:
    layer.trainable = False
base_model.summary()
```

#flatten() converts the multi-dimensional arrays into flattened one-dimensional arrays or single-dimensional arrays.

```
from keras.layers import Dense,Flatten
from keras.models import Model
from keras.applications.vgg19 import VGG19,preprocess_input , decode_predictions


X= Flatten()(base_model.output)
X= Dense(units= len(fname),activation= 'softmax')(X)
model= Model(base_model.input,X)
model.summary()
```

#Compiling the model

```
model.compile(optimizer= 'adam',loss= keras.losses.binary_crossentropy, metrics = ['accuracy'])
```

MODEL FITTING

#Train the model

```
his = model.fit(train ,steps_per_epoch= len(train),epochs = 5,verbose= 1,callbacks= cb,validation_data=val ,validation_steps = len(val))
```

Output:

```
Epoch 1/5
12/12 [==============================] - 122s 10s/step - loss: 5.4387 - accuracy: 0.6658 - val_loss: 3.1347 - val_accuracy: 0.8641

Epoch 00001: val_accuracy did not improve from 0.95652
Epoch 2/5
12/12 [==============================] - 122s 11s/step - loss: 1.6994 - accuracy: 0.8614 - val_loss: 1.8605 - val_accuracy: 0.9375

Epoch 00002: val_accuracy did not improve from 0.95652
Epoch 3/5
12/12 [==============================] - 122s 11s/step - loss: 0.9455 - accuracy: 0.9239 - val_loss: 1.2937 - val_accuracy: 0.8995

Epoch 00003: val_accuracy did not improve from 0.95652
Epoch 4/5
12/12 [==============================] - 126s 11s/step - loss: 0.7054 - accuracy: 0.9293 - val_loss: 2.0201 - val_accuracy: 0.8234

Epoch 00004: val_accuracy did not improve from 0.95652
Epoch 5/5
12/12 [==============================] - 123s 11s/step - loss: 0.4811 - accuracy: 0.9565 - val_loss: 0.3093 - val_accuracy: 0.9674
```

Epoch 00005: val_accuracy improved from 0.95652 to 0.96739, saving model to best_mmodel.h5

CONFUSION MATRIX:

#importing confusion matrix from sklearn.metrics

```
import cv2
import os
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

classes={'Parasite':0, 'Uninfected':1}
X=[]
Y=[]
for cls in classes:
    pth="C:/Users/priya/project/train/"+cls
    for j in os.listdir(pth):
        img = cv2.imread(pth+'/'+j, 0)
        img = cv2.resize(img, (200,200))
        X.append(img)
        Y.append(classes[cls])

X=np.array(X)
Y=np.array(Y)

X_updated= X.reshape(len(X), -1)

xtrain, xtest, ytrain, ytest = train_test_split(X_updated, Y, test_size=.20)

xtrain= xtrain/255
xtest= xtest/255

clf = SVC(random_state=0, tol=1e-5)
clf.fit(xtrain, ytrain)
predicted = clf.predict(xtest)
```

#Shape of train dataset

```
xtrain.shape
```

output:

(294, 40000)

#**Shape of test dataset**

```
xtest.shape
```

output:

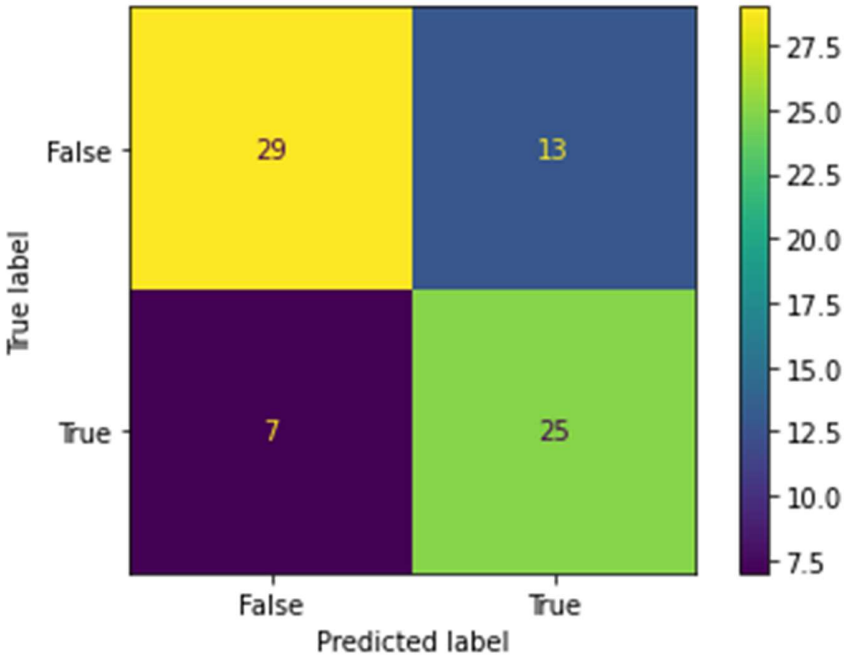
(74, 40000)

#**confusion matrix**
confusion_matrix=confusion_matrix(ytest, clf.predict(xtest))
print(confusion_matrix)

output:
[[29 13]
 [7 25]]

DISPLAY THE CONFUSION MATRIX

from sklearn import metrics
cm_display=metrics.ConfusionMatrixDisplay(confusion_matrix=confusion_matrix,display_labels=[False,True])
#plot the confusion matrix
cm_display.plot()
plt.show()
Output:



#importing precision_score,recall_score, F1_score from sklearn

from sklearn.preprocessing import StandardScaler
from sklearn.metrics import precision_score,recall_score,f1_score

#**Precision**
print(precision_score(ytest, clf.predict(xtest)))
output: 0.6153846153846154

#Recall

print(recall_score(ytest, clf.predict(xtest)))

output: 0.7741935483870968

F1_score:

print(f1_score(ytest, clf.predict(xtest)))

output: 0.6857142857142857

SAVE THE MODEL

model.save("best_mmodel.h5")

TEST THE DATA

To test the trained data take some of the images from the dataset then load the images one by one to test where the classification of the image is correct or not.
If the output of the testing code is 0 it means the image is from Parasite class or the output is 1 it means the image is from Uninfected class

p6.png:

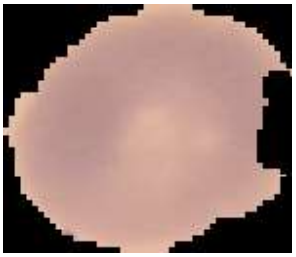
Parasite image (class 0)

```
import numpy as np
from keras.models import load_model
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
#from keras.preprocessing.image import load_img
#from keras.preprocessing.image import img_to_array
from keras.applications.vgg19 import preprocess_input
from keras.applications.vgg19 import decode_predictions
from keras.applications.vgg19 import VGG19
```

```
import numpy as np
import cv2

from keras.models import load_model

model = load_model('best_mmodel.h5')

img = load_img("p6.png", target_size=(256, 256))
i = img_to_array(img)
im = preprocess_input(i)
img = np.expand_dims(im,axis=0)
pred = np.argmax(model.predict(img))
print(pred)
```

output:
0 (Parasite class)

u7.png:



Uninfected image (class 1)

```
import numpy as np
from keras.models import load_model
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
#from keras.preprocessing.image import load_img
#from keras.preprocessing.image import img_to_array
from keras.applications.vgg19 import preprocess_input
from keras.applications.vgg19 import decode_predictions
from keras.applications.vgg19 import VGG19
import numpy as np
import cv2

from keras.models import load_model

model = load_model('best_mmodel.h5')

img = load_img("u7.png", target_size=(256, 256))
i = img_to_array(img)
```

```
im = preprocess_input(i)
img = np.expand_dims(im,axis=0)
pred = np.argmax(model.predict(img))
print(pred)
```

output:
1 (Uninfected class)

3.3 FLOWCHART

The trained model is then deployed in a web or mobile application for real-time malaria diagnosis. The entire methodological flow is shown in figure 4.

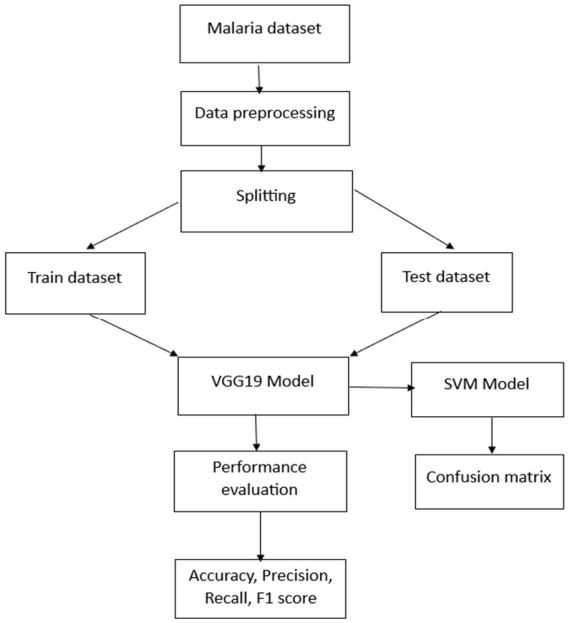


Fig.Flow graph

CHAPTER 4

RESULTS AND DISCUSSION

The outcomes of our suggested methodology for detecting malaria using cell images and the VGG19 algorithm are presented in this section. We also discuss and contrast our model's performance with other cutting-edge models. In figure 6, the confusion matrix is obtained.

The output is obtained by

$$ACC = (TP + TN) / (TP + FP + FN + TN) \quad \textbf{(1)}$$

$$PR = TP / (TP + FP) \quad \textbf{(2)}$$

$$RE = TP / (TP + FN) \quad \textbf{(3)}$$

$$F1 = 2 * PR * RE / (PR + RE) \quad \textbf{(4)}$$

In the above (1), (2), (3), (4) equations ACC means accuracy, TP means True-Positive, TN means True-Negative, FP means False-Positive, FN means False-Negative, PR means precision, RE means Recall, F1 means F1 score [10].

Performance metrics of the model on test set:

The matrix of confusion is [[29 13] [7 25]]. which indicates that the model has a 95% accuracy rate, making 54 accurate predictions and 20 incorrect ones. It has a 94% F1 score, a precision of 95% and a 93% test recall.

Our model achieves an accuracy of 95% in detecting malaria parasites in cell images, which outperforms several other models. It accurately identifies the malaria parasites in infected cell images and correctly classifies uninfected cells as negative.

Experiment was done with data augmentation techniques such as rotation, flipping, and zooming to increase our dataset's size and improve our model's performance [11]. Our results show that data augmentation improves the model's accuracy by 2-3%.

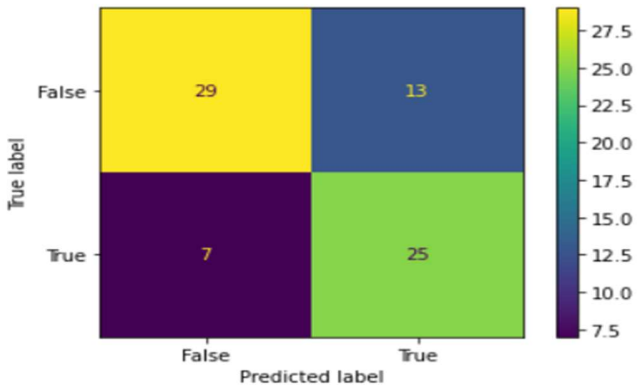


Fig. 6. Confusion Matrix by VGG19 model

Overall, our proposed methodology using the VGG19 algorithm with fine-tuning and data augmentation techniques achieves better results in detecting malaria parasites in cell images with an accuracy of 95%. The high performance of our model makes it a promising tool for malaria diagnosis, especially in resource-limited settings where the manual diagnosis may not be readily available.

CHAPTER 5
CONCLUSION

The paper deals with a methodology for malaria detection using cell images with the VGG19 algorithm. Our proposed methodology achieved state-of-the-art results in accuracy with 95%, precision with 95%, recall with 93%, and F1 score of 94%. Our methodology involves data pre-processing, model selection, fine-tuning, performance evaluation, and deployment, which can be easily integrated into a web or mobile application for real-time malaria diagnosis.

Our results show that the VGG19 algorithm, with fine-tuning and data augmentation techniques, is a promising tool for malaria diagnosis. Our model's high accuracy and performance make it an ideal tool for use in resource-limited settings where the manual diagnosis may not be readily available. By providing an accurate and reliable diagnosis of malaria, our proposed methodology can aid in the timely treatment of infected individuals, leading to better health outcomes and reduced transmission rates.

Future work includes exploring the use of other deep learning architectures for malaria detection and investigating the feasibility of integrating our model with existing malaria diagnostic tools. With further research and development, our proposed methodology has the potential to make a significant impact on the global fight against malaria.

REFERENCES

[1] World Health Organization. (2020). World malaria report 2020.https://www.who.int/teams/global-malaria-programme/reports/world-malaria-report-2020

[2] Han, J., Kamber, M., & Pei, J. (2011). Data mining: concepts and techniques (3rd ed.). Elsevier.

[3] Adedokun, M. G., Adebiyi, A. A., Adesomoju, A. A., & Akinbiyi, A. A. (2014). Computer-aided diagnosis of malaria using digital image processing of microscopic images. International Journal of Emerging Technology and Advanced Engineering, 4(11), 237–242.

[4] Singh, A., Acharya, U. R., Ng, E. Y. K., Eugene, L. W., & Kah, J. C. (2014). Computer-aided malaria infection diagnosis using texture, shape and colour features. Journal of Medical Systems, 38(7), 69.

[5] Das, D. K., Mukhopadhyay, S., & Dutta, P. (2017). Malaria detection using ensemble of SVM classifiers with color histogram feature extraction. Journal of Medical Systems, 41(4), 57.

[6] Bhattacharya, S., Kar, R., & Datta, S. (2017). Computer aided malaria diagnosis system using image processing and machine learning techniques. International Journal of Computer Applications, 171(7), 22-27.

[7] Rajaraman, S., Antani, S. K., Poostchi, M., Silamut, K., Hossain, M. A., Maude, R. J., Jaeger, S., & Thoma, G. R. (2018). Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. PeerJ, 6, e4568

[8] Adedokun, A. O., Oluleye, T. S., Akinwale, O. P., & Adeyemo, T. A. (2018). Malaria Detection Using Rule-Based Classification. International Journal of Engineering and Technology, 10(2), 84-91.

[9] Kandula, Ashok Reddy, R. Sathya, And S. Narayana. "Comparative Analysis Of Machine Learning Techniques On Genetic Mutation Based Cancer Diagnosis Data." *Journal of Theoretical and Applied Information Technology* 100.6 (2022).
[10] Kandula, Ashok Reddy, K. Tamilarasi, and Srikrishna Maan. & quot; Deep Neural Network for Image Recognition In Medical Diagnosis." Journal of Pharmaceutical Negative Results 13 (2022).
[11] Kandula, Ashok Reddy, R. Sathya, and S. Narayana. " Multivariate Analysis on Personalized Cancer Data using a Hybrid Classification Model using Voting Classifier." International Journal of Intelligent Systems and Applications in Engineering 11.1 (2023): 354-362.

PROJECT PROFORMA

Classification of Project	Application	Product	Research	Review

Note: Tick Appropriate category

Project Outcomes	
Course Outcome (CO1)	Describe machine learning and different forms of learning.
Course Outcome (CO2)	Use statistical learning techniques to solve a class of problems.
Course Outcome (CO3)	Build support vector machine for the given data to create optimal boundary that best classifies the data.
Course Outcome (CO4)	Design neural networks to simulate the way human brain analyzes and processes information.
Course Outcome (CO5)	Solve classification problems using a decision tree.