Advances in Malaria Detection: Synergizing Deep Learning and Traditional Machine Learning

Samuel Yao Sebuabe
Department of Computer Science
Valley View University

Justice Kwame Appati
Department of Computer Science
University of Ghana

Stephen Kofi Dotse
Department of Information Technology Studies
University of Professional Studies

John Yao Akakpo
Department of Computer Science
University of Ghana

Doe Martin
Department of Computer Science
Valley View University

ABSTRACT
Malaria remains one of the leading causes of death globally, necessitating continuous research into novel diagnostic and treatment methods. Despite available treatments, accurately assessing drug efficacy against malaria parasites remains challenging due to the need for precise parasite quantification in blood-smeared images, a task traditionally performed using time-consuming microscopy methods. In this study, we propose a Convolutional Neural Network (CNN)-based deep learning model to enhance malaria parasite detection from Giemsa-stained thin blood smears. The proposed model incorporates advanced preprocessing techniques, including normalization, standardization, and staining, as well as data augmentation methods (e.g., random cropping, flipping, and rotation) and hyperparameter optimization to significantly improve performance. The primary dataset from the National Institutes of Health (NIH), consisting of 27,558 parasitized and uninfected cell images, was used to train and evaluate the model. A custom CNN architecture was compared with pre-trained models like VGG-19, ResNet-50, and MobileNetV2 based on accuracy, precision, recall and AUC. The best-performing model achieved a training accuracy of 96.88%, validation accuracy of 95.55%, and test accuracy of 95.67% after 50 epochs. Performance metrics such as precision (97.37%), recall (97.75%), and AUC (99.19%) demonstrated high sensitivity and specificity, confirming the model's robustness. A secondary dataset from the IEEE repository, containing 43,434 images, was used to validate the model, yielding near-identical performance and further confirming its generalizability across diverse datasets. These findings underscore the proposed model's ability to accurately detect malaria parasites, offering a faster and more reliable alternative to traditional microscopy. Future work will explore integrating mobile-based imaging systems with cloud and edge-based inference for deployment in low-resource settings, aiming to enhance malaria treatment outcomes in underserved regions.

General Terms
Artificial Intelligence, Deep Learning, Pattern Recognition, Computer Vision, Algorithms, Image Processing, Medical Diagnostics, Machine Learning, Health Informatics.

Keywords
Malaria, Convolutional Neural Network, Cell Images, Minutiae, IEEE Dataset, Giemsa-stained Thin Blood Smears.

1. INTRODUCTION
Malaria is a major national health concern around the world. It is transmitted by the parasite plasmodium falciparum, which is responsible for the majority of deaths and serious health problems in impoverished areas of Africa and other tropical regions. According to the latest World malaria report, there were 263 million cases of malaria in 2023 compared to 252 million cases in 2022. The estimated number of malaria deaths stood at 597 000 in 2022 compared to 600 000 in 2022 [1]. The main symptoms of malaria include fever, vomiting, headache, and tiredness. Parasitological studies, including microscopic tests and rapid diagnostic tests (RDT), are necessary to assess whether potential patients are infected with malaria or not. However, in locations where malaria parasitological tests are not easily accessible, the complexities of malaria diagnosis may result in misinterpretation and improper presumed therapy [2]. In the treatment of malaria, it is recommended that anti-malaria drugs be taken upon parasitological confirmation of the condition. A variety of contributions have been put in place to diagnose malaria easily and efficiently [3]. The fundamental approach entails collecting a blood sample from an infected person and sending it to a laboratory where an experienced health professional must distinguish between uninfected and parasitized cells. This diagnostic technique is time-consuming and labor-intensive. To minimize the cost of blood testing and reduce the personnel required for plasmodium parasite detection in blood samples, it is recommended to implement cost-effective automated diagnostic techniques, in one or more modalities, for the identification of infected erythrocytes containing plasmodium parasites [4]. The most critical stage before starting treatment of malaria is the detecting stage, as accurate diagnosis is essential for determining the appropriate treatment [5]. According to [6], several studies have been conducted on diagnosing malaria using machine learning techniques, with the majority focusing on the blood smear image method. They also stated that tons of blood smears are studied for the plasmodium parasite every year, using available methods such as Rapid Diagnostic Tests, Immunofluorescence Antibody Testing (IFA), Polymerase Chain Reaction (PCR) based techniques, Loop-mediated Isothermal Amplification (LAMP) technique, Light Microscopy, and others, which involve manual detection of malaria falciparum parasites and infected red blood cells by trained microscopists. Despite many of these evaluation strategies for detecting infection, the majority of microscopists in resource-limited areas have hurdles in improving diagnostic accuracy [7]. Machine learning has recently been applied in the public medical field for disease detection and prediction. With the availability of extensive healthcare datasets and

advancements in computational techniques, it is now capable of diagnosing a wide range of health disorders. Public health researchers have utilized several machine learning methods which have yielded promising results. [8] introduces the IDTL-MPDC model for malaria detection using deep transfer learning, achieving 95.86% accuracy by integrating median filtering, Res2Net, differential evolution, and KNN. While it outperforms existing methods, its reliance on a single dataset and lack of interpretability and real-world validation highlights areas for improvement. The study by [9] presents TL-SGAN, a model combining transfer learning and semi-supervised GANs for malaria detection with limited labeled data, achieving 96.6% accuracy. It reduces data dependency and training time but focuses only on binary classification and requires validation on diverse datasets. In a similar vein, [10] also proposes a deep learning-based framework (HPTDL-MPDC) for automated malaria detection, combining VGG19 for feature extraction and an LSTM-CNN model for classification, optimized with Adagrad. Achieving 91% accuracy on a large dataset, it outperforms existing methods, though it requires further clinical validation and testing on diverse datasets for broader applicability. [11] develops YOLO-mp models for real-time malaria detection, addressing annotation inconsistencies in datasets and optimizing YOLOv4 for speed and accuracy. The refined Dataset B-centered improves data quality, and the lightweight YOLO-mp-3l model achieves 94.07% mAP, outperforming YOLOv4 while being faster and smaller, making it suitable for low-resource settings. The objectives of this study are:

- To propose a more efficient computational method for the detection of malaria parasites from blood smear images
- To perform an in-depth comparison of the proposed method with other existing approaches in the same domain using secondary data (online repository)

The subsequent sections of the paper are organized as follows. The material and method/algorithm are introduced in Section 2. Section 3 extensively discusses the results recorded. Lastly, the work is concluded in Section 4.

2. THE MATERIALS AND METHODS

This study focuses on comparing the performance of different deep learning methods in detecting Plasmodium parasites from medical images, aiming to improve malaria diagnosis accuracy and efficiency. This section provides an overview of the deep learning techniques considered, the data acquisition strategy adopted, and the algorithms used to achieve the study's objective.

The data acquisition strategy is critical for ensuring high-quality, labeled medical images for training the models [12]. Details about the dataset, including its source, resolution, and annotation process, are discussed. The programming logic and algorithms employed, particularly the use of deep learning models like Convolutional Neural Networks (CNNs), are also outlined.

The following sections cover data preprocessing and augmentation techniques, which are essential for preparing the raw images by enhancing their quality and expanding the dataset. The proposed architecture is then discussed, detailing the layers and components used to detect Plasmodium parasites, with a focus on the integration of pre-trained models and transfer learning strategies. The training process, including loss functions, optimization techniques, and hyperparameter tuning, is briefly explained.

Finally, the section discusses various evaluation metrics, such as accuracy, sensitivity, specificity, precision, recall, F1-score, and the area under the ROC curve, which will be used to assess the effectiveness of the model in detecting malaria. These metrics are critical for evaluating the performance of the deep learning models and ensuring their suitability for medical applications.

2.1 Deep Learning Models

Deep learning is a powerful computational framework that relies on multiple convolutional filters to interpret data at various levels of abstraction. Unlike humans, deep learning algorithms require vast amounts of high-quality annotated data to make accurate predictions. This dependency on large datasets has historically hindered the widespread adoption of deep learning in fields like medicine, where annotated data is scarce and privacy concerns are prevalent [13], Neural networks, which are primarily composed of layers of interconnected neurons, can be employed to uncover fundamental relationships in datasets and extract meaningful features [14] In this context, several deep learning models have been investigated, each with distinct principles, advantages, and limitations. These models include Sequential CNN, pre-trained VGG19, pre-trained ResNet50, and pre-trained MobileNet-v2 classifiers. All models were tested on the same dataset, with their outcomes compared to assess performance. These models were chosen because they are trained on the ImageNet database, which, despite requiring high computational power and large datasets, produces reliable and robust results. Given the computational limitations of the system used in this study, pre-trained models were leveraged to efficiently compare results while avoiding the need for extensive retraining.

One of the core deep learning models, the Convolutional Neural Network (CNN), is widely used for image analysis and classification tasks. CNNs consist of convolutional, pooling, and fully connected layers [15]. which allow them to automatically extract features from images while preserving spatial hierarchies. The performance of a CNN can be mathematically expressed as:

$$C_j^1 = \mu\left(\sum_{i=1}^{m^{l-1}} c_i^{j-1} * K_{i_j}^i + b_j^{l-1}\right) \quad (1)$$

Typically, CNNs are built with three primary types of layers: convolutional layers, pooling layers, and fully connected layers, as illustrated in Fig. 1. Convolution blocks, which combine convolutional and pooling layers, are stacked together to form the network. The fully connected layer, usually found at the end of the network, is often used for segmentation or hypothesis testing. While CNNs are highly effective at processing large image datasets and automatically extracting hierarchical features, they do have limitations. These models require large amounts of labeled data and significant computational resources. Moreover, they may perform poorly on tasks with limited data or non-image inputs, which can pose challenges in certain domains.

In response to these challenges, MobileNet was introduced by Google at CVPR 2017 as a lightweight CNN designed for mobile applications. By using depthwise separable convolutions, MobileNet reduces the computational load compared to traditional CNNs [17]. Building on this, MobileNetv2, introduced by [18], incorporates inverted residual blocks, further optimizing the model with fewer parameters and improved performance when processing images larger than $32px$ x $32px$. Each block in MobileNetv2 includes a $1x1$ convolution with ReLU6 activation, followed

by a depthwise convolution and another $1x1$ convolution. This architecture not only reduces the model's computational footprint but also ensures efficient performance for a variety of tasks, as demonstrated in Fig. 2.

Another well-established deep learning model, VGG19, was developed by the Visual Geometry Group at Oxford and is known for its high accuracy in image recognition tasks. VGG19 consists of 19 layers, including 16 convolutional layers, 3 fully connected layers, and 5 max-pooling layers. It is widely used for visual recognition and is often fine-tuned with transfer learning to adapt to specific datasets, such as ImageNet or more specialized tasks like malaria detection [19]. Despite its simplicity, VGG19 is highly effective at extracting hierarchical features, making it a popular model for many image classification challenges, as shown in Fig. 1.

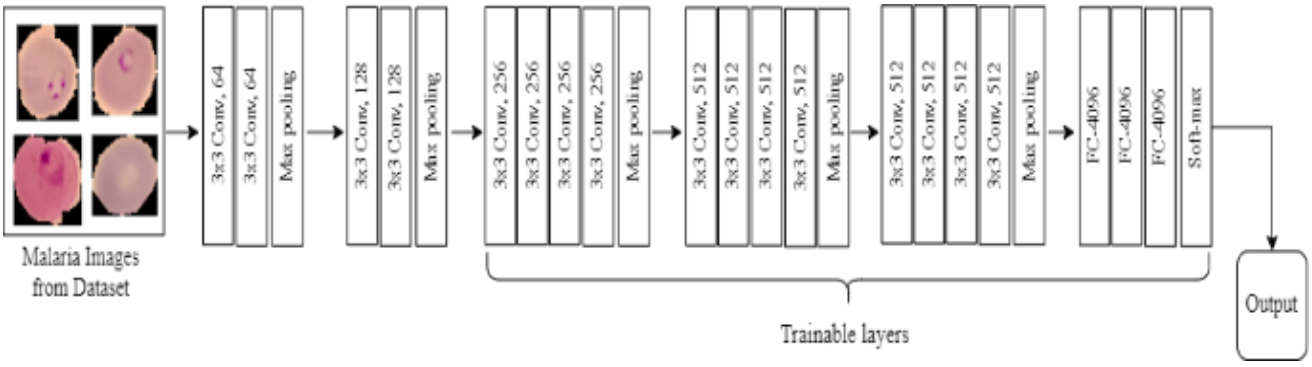

Fig 1: VGG-19 Architecture

In addition to CNN and MobileNet, the ResNet architecture, introduced by [5], addresses performance degradation in deep learning models by employing residual connections. These connections allow the network to bypass certain layers, mitigating the vanishing gradient problem and enabling deeper networks without compromising performance. ResNet50, an improved version of ResNet, uses these residual blocks to maintain high performance even with complex architectures. The residual connection is mathematically expressed as:

$$y = F(x, W + x) \quad (2)$$

where x is the input and y is the output. ResNet50 was used in this study with transfer learning.

ResNet50 improves both training efficiency and classification accuracy, making it particularly effective for complex datasets. For this study, ResNet50 was employed with transfer learning, helping to enhance performance while reducing the need for extensive retraining [20].

Each of the deep learning models tested in this study CNN, MobileNetv2, VGG19, and ResNet50 offers unique advantages suited to different tasks. CNNs excel at feature extraction from large image datasets, MobileNetv2 is optimized for low-resource environments such as mobile applications, VGG19 provides high accuracy in image recognition tasks, and ResNet50 allows for deeper networks with better performance through residual connections [21]. By leveraging pre-trained models, the study makes use of the strengths of each architecture, addressing computational limitations while comparing their performance on a common dataset. Collectively, these models contribute valuable insights into the potential of deep learning for image classification and other machine learning applications.

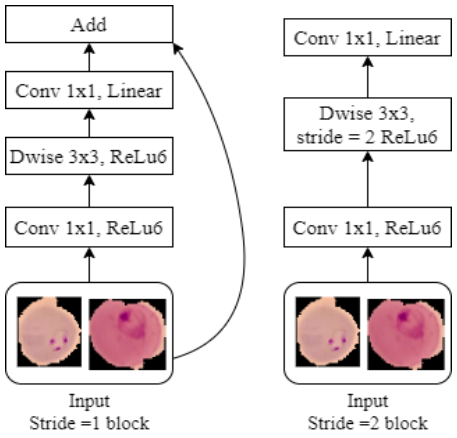

Fig 2: MobileNetv2 Architecture

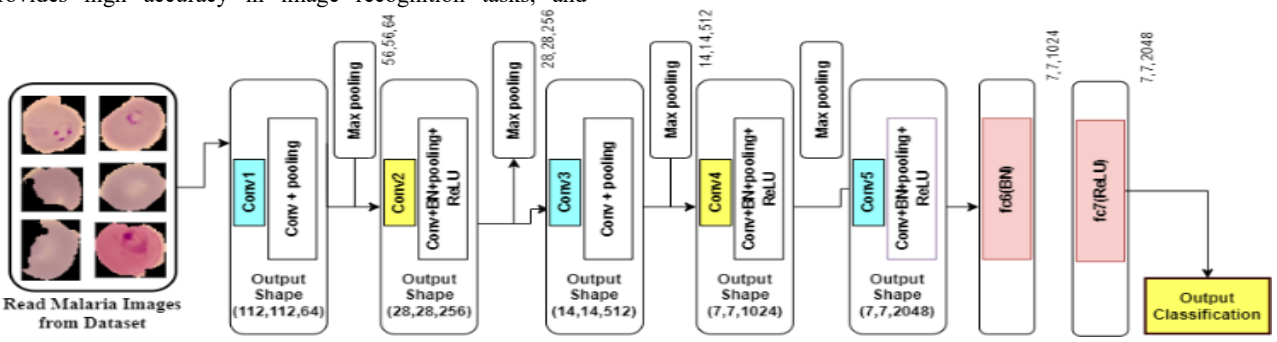

Fig 3: ResNet-50 Architecture

2.2 Data Acquisition

For this research, two different secondary datasets were considered. They are the malaria images (thin blood smear) dataset which was downloaded from the IEEE data repository

and malaria cell images which were also retrieved from the National Institutes of Health (NIH) website. The main reasons for adopting these datasets are that (a) they are used extensively by other researchers in the same or similar domain, and (b) they also produced higher accuracy results based on papers. Further descriptions of these datasets are stated in the sections that follow.

2.1.1 National Institutes of Health Dataset

This dataset was retrieved from the NIH data repository [22]. The dataset contains 27,558 cell images which are organized into two folders labeled as parasitized and uninfected. Each of these subfolders contains 13,779 images as represented in Table I below. The data was taken from 150 P. falciparum and 50 healthy patients and it was photographed at Chittagong Medical College Hospital, Bangladesh using a smartphone by placing it on the conventional light microscope [23].

Table 1: Malaria Image Dataset from NIH Repository

Label	Number of Images
Parasitized	13,779
Uninfected	13,779

This data, however, was further restructured into three units, for training, testing, and validation. A detail of this is explained in section 5 of this report. Figure 5 and 6 are sample displays of the images in the dataset for parasitized and uninfected cells.

2.1.2 IEEE Data

This dataset is retrieved from the IEEE data repository [24]. In total, it contains 43,434 cell images which are organized into three folders labeled training, testing, and single prediction. The training and testing folders have subfolders labeled as parasitized and uninfected. The distribution of the samples in these folders is summarized in Table 2.

Table 2: IEEE Dataset Repository

Set	Labels	
	Parasitized	**Uninfected**
Testing	7,952	7,880
Training	13,800	13,800
Single Prediction	1	1

2.3 Programming Logic and Algorithms

The programming language used for the implementation of this work is python programming language version 3.12.4 and anaconda software which provided access to a coding environment called Jupiter notebook. The following open-source libraries which made it possible to perform certain special functions during the implementation process were used.

- OS – enabled us to load the image files from the hard disk to Jupiter notebook
- OpenCV and skimage – helped in processing our images
- Keras API and TensorFlow
- Matplotlib and Seaborn – helped in the visualization of the images in the Jupiter notebook
- Pandas – for analysis and normalization of data
- NumPy – for mathematical computations and further computational analysis

2.4 The Proposed Architecture

The architecture of the proposed malaria detection model is systematically outlined, as depicted in the schematic diagram in Fig. 4. The workflow begins with input data acquisition (microscopic images of blood smears), followed by preprocessing to enhance data quality through normalization and augmentation techniques. The processed dataset is then divided into training, validation, and test subsets, ensuring robust model evaluation. The core of the architecture involves model training, where a customed sequential convolutional neural network (CNN) is developed to classify the images as infected or uninfected. Finally, the trained model is saved for deployment, facilitating future use in malaria detection systems.

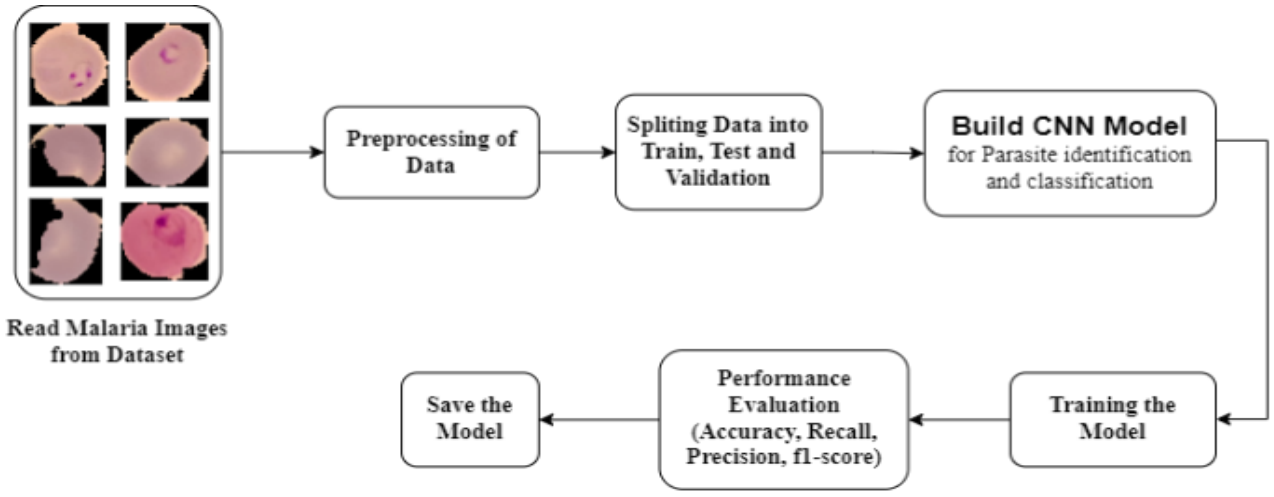

Fig 4: Pipeline for the Proposed method

2.5 Data Preprocessing and Augmentation

Data preprocessing plays a crucial role in improving image quality, as images can be affected by various types of noise, such as camera angle and the position of the image at the time of capture [25]. To address this, several methods are applied to clean the images and reduce noise. In this work, during the preprocessing stage, all malaria images are first transformed to a specified target size, and their pixel values are rescaled to a unit value. This process according to [26] ensures that all images are of uniform size before being passed into the model

for training. To suppress high intensities and enhance the images by detecting their edges, digital image filtering techniques are employed. These filtering techniques are essential in improving the overall quality of the images [27]. One of such technique used in this study is image resizing, which is an important preprocessing technique in the image recognition domain. Typically, deep convolutional learning frameworks are trained more efficiently using smaller-sized images, as larger images require the network to process four times as many pixels, increasing computational costs and training time. In this study, the original dimensions of the parasitized and uninfected images in the dataset are heterogeneous, with variations in size and resolution observed across the samples. Consequently, all images are resized uniformly to a target size of [224, 224], and their pixel values are rescaled to unit values before being input into the model. This resizing technique helps to reduce the training time and conserves computational resources, such as CPU power and memory [28]. In addition to resizing, Gaussian filtering is employed to further enhance the image quality. This technique helps reduce noise and blur areas in the image by applying a filter that focuses on specific areas of the image. The filter passes as a geometric kernel across each pixel in the region of interest, giving greater weight to pixels near the center of the kernel, while those at the periphery have less influence on the final value. The Gaussian filter is essentially an approximation of Gaussian mathematical concepts, and when applied to an image, the dimensions of the matrix used to modify the image are first calculated [29]. These dimensions are typically odd numbers, which ensures that the center pixel is the focal point of the calculation. The kernel itself is squared, with an equal number of rows and columns, and the values within it are computed using the Gaussian function:

$$f(\mathrm{x},\mathrm{y}) = \frac{1}{2\pi\sigma^2} - \mathrm{e}^{\frac{\mathrm{x}^2+\mathrm{y}^2}{2\sigma^2}} \quad (3)$$

where x and y are the coordinates, and σ represents the standard deviation. This function allows for the creation of a Gaussian kernel of any size by adjusting the parameters accordingly. The effect of applying the Gaussian filter is illustrated in Fig. 8, which shows a comparison between the original image and the one after the filter has been applied. In the other hand, the edges of an image represent its most fundamental features, capturing a significant number of internal properties of an image. As such, edge detection is one of the most important tasks in the field of image processing. Among the various approaches used for edge detection, the most common are differential and filtering techniques [30]. However, the more traditional differential and gradient-based edge detection techniques are often complicated and yield unsatisfactory results [31]. Operators such as Prewitt, Roberts, and Laplacian, although widely used, are highly sensitive to noise and tend to perform poorly under noisy conditions [32].

In this study, the Sobel edge detection technique was employed to extract edge features from the images for further analysis. The Sobel operator was chosen for its ability to smooth random noise in images [33], especially since the introduction of median filters. This makes it a reliable method for edge detection in the context of image processing. However, it is worth noting that, despite its advantages, this technique did not yield significant contributions to the work during the experiments. This observation warrants further investigation, and future work will explore the potential causes of this phenomenon in more detail. Figure 9 illustrates a comparison between the original image and the image after the Sobel edge detection technique has been applied. Another preprocessing technique that has been explored in this study is normalization, which as has been defined by [34], as the process of converting image pixels to a common scale across all images in a dataset. This technique is widely used in computer vision applications to accelerate model learning. Specifically, normalization involves dividing the pixel values of an image by the highest value that a pixel can take. In this study, normalization was achieved by dividing each image in the dataset by 255, a method easily implemented using OpenCV and the Python programming language. This process becomes essential because when using raw images and passing them directly to the model, the computation of these pixel values can become complex and computationally expensive.

Therefore, normalizing the pixel values to a range from 0 to 1 helps reduce computational costs and makes the processing faster and more efficient [35], as the resulting values are smaller.

2.6 Splitting Data into Training, Validation and Testing

In the first experiment, the dataset was partitioned into three subsets: training, validation, and testing. To optimize computational efficiency, 70% of the dataset was designated for training, 10% for model validation, and 20% for evaluating the performance of the model on the test set. In the second experiment, the dataset was again divided into three subsets, but with different proportions: 60% for training, 10% for validation, and 30% for testing. The outcomes of these experiments are discussed in the subsequent section.

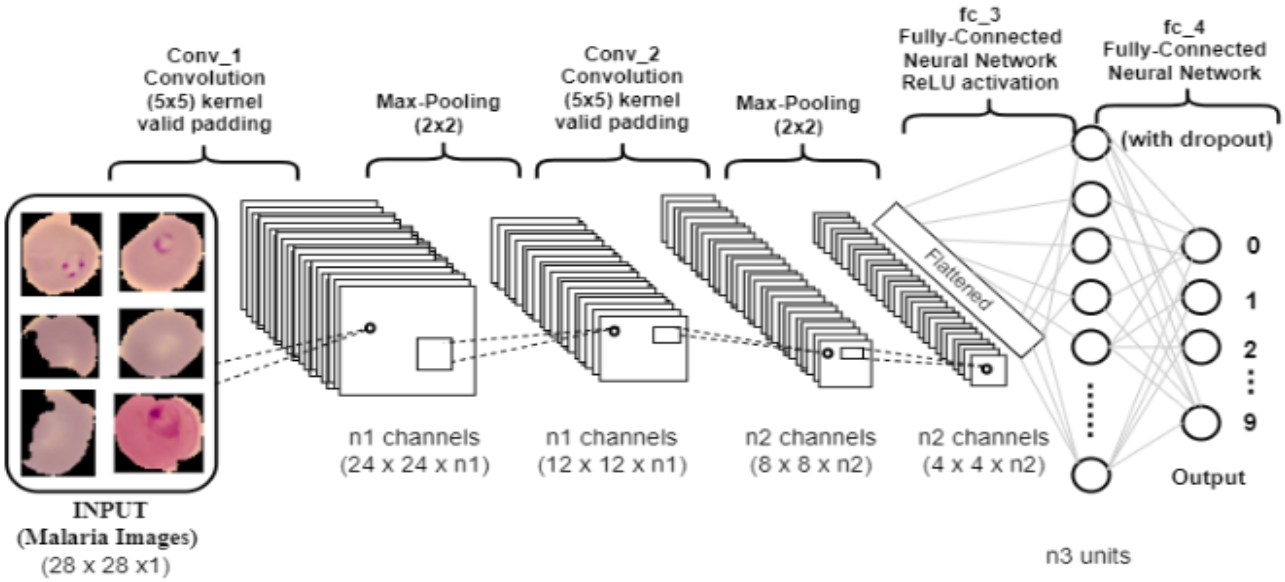

Fig 5: CNN Architecture [16]

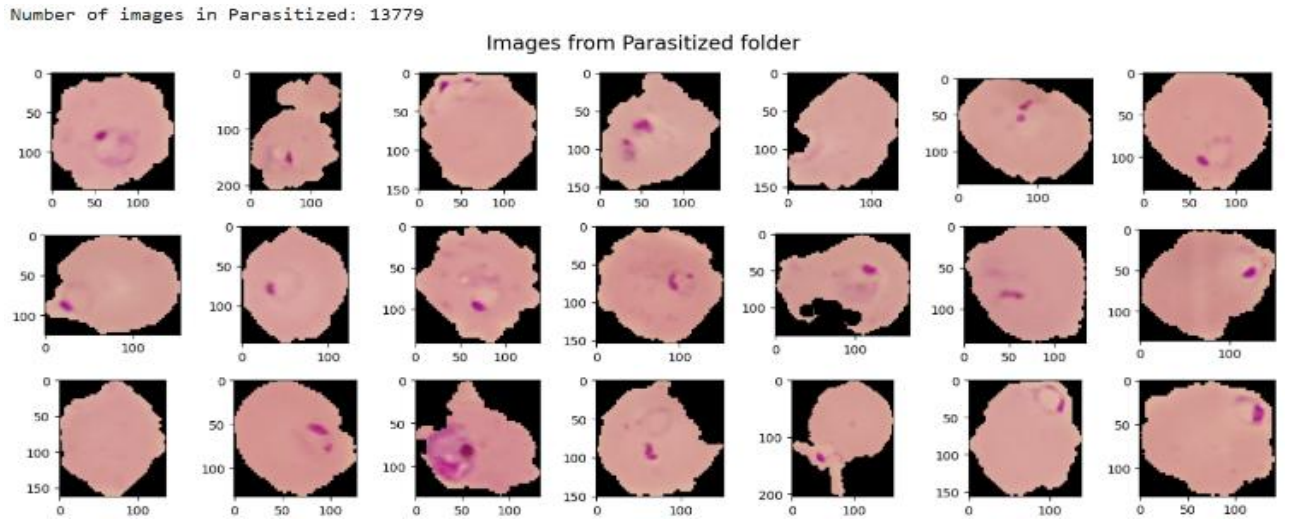

Fig 6: Some samples of parasitized malaria cells

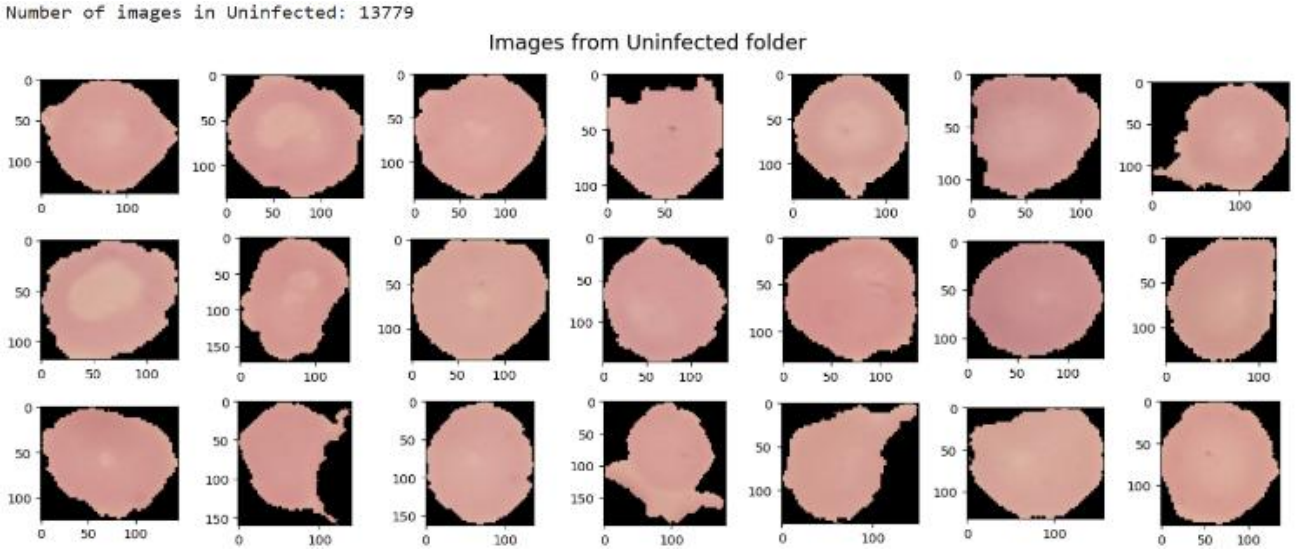

Fig 7: Some samples of uninfected malaria cells

2.7 Activation Functions

The activation function is a critical component of a neural network, providing the necessary nonlinear transformation that enables the network to model complex relationships within data. It defines the output of each neuron in response to a set of inputs, thereby determining the neural network's ability to approximate intricate functions. Activation functions are inspired by biological neural networks, particularly the firing patterns of neurons in the human brain when activated by various stimuli [36]. In the present study, two distinct activation functions were employed to optimize the model during the training phase. The Rectified Linear Unit (ReLU) is a widely used nonlinear activation function in deep neural networks, particularly effective in multi-layer architectures. ReLU is advantageous because its derivative is constant (equal to 1) for positive input values, which allows for faster convergence during training relative to conventional activation functions. Additionally, the simplicity of ReLU eliminates the need for extra computational overhead during training, as it operates on a constant value for positive inputs. ReLU is mathematically defined as:

$$\mathrm{f(x)} = \max\ (0, \mathrm{x}) \tag{4}$$

where x represents the input value. However, ReLU is generally unsuitable for use in the output layer of a neural network. This is due to the fact that it does not activate all neurons simultaneously, meaning neurons will only be activated if the output of the nonlinear transformation exceeds zero. This characteristic makes ReLU less appropriate for tasks requiring probability outputs, such as classification tasks. In such cases, the output layer must generate probabilities for each class based on the input data.

In contrast, the sigmoid activation function is characterized by its ability to map any real-valued input to an output within the range of 0 to 1. This function was originally introduced by Pierre François Verhulst in the mid-19th century as a model for population growth, in which the function adjusted an exponential model to fit observed data between 1838 and 1847. The sigmoid function is mathematically expressed as:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

where x represents the input to the sigmoid function, and e denotes $Euler's\ constant$ (2.71828). The sigmoid function is particularly suitable for binary classification tasks, as its output can be interpreted as a probability value between 0 and 1. In this study, the sigmoid function was selected for the output layer because the task involves binary classification, specifically determining whether a blood smear contains malaria parasites. The probabilistic nature of the sigmoid function aligns well with the requirement to output class probabilities.

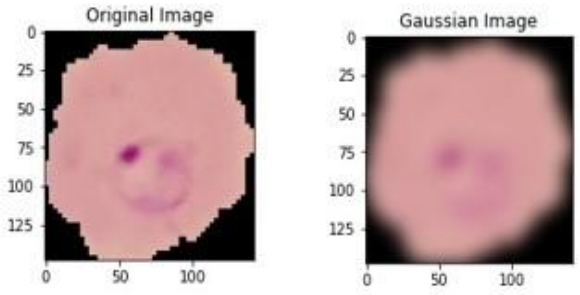

Fig 8: Sample of Original Image of RBC and Image after Gaussian Filtering

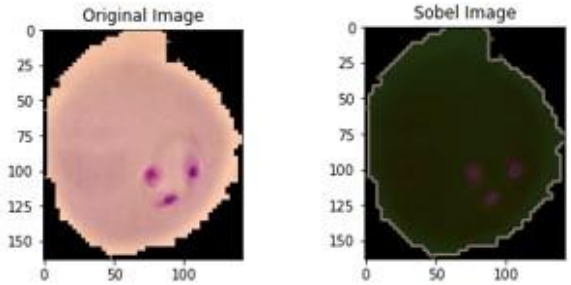

Fig 9: Sample of original image of RBC and image after Sobel edge detection

2.8 Training the Model

The number of trainable and nontrainable parameters of the proposed model are 11,169,089 and 0, respectively, which accounts for a total of 11,169,089 parameters to be trained in the model. Three callback functions were employed during the training process to make the model more resilient, ensuring that training does not continue without improvement in results. These callback functions are ModelCheckPoint, EarlyStopping, and LearningRateScheduler. The learning rate is dynamically adjusted based on 50 epochs. The model was trained using the binary_crossentropy loss function, which is typically used for binary classification models. The binary_crossentropy function computes the cross-entropy loss between true labels and predicted labels. This loss function was chosen because our focus was to perform binary classification of whether a cell is infected with malaria or not. The binary_crossentropy function is mathematically represented as:

$$Loss\ (\widehat{y,p}) = -log\ (P(y \mid x) \tag{6}$$

where x and y are input values. To optimize the weights and biases of the model, an ADAM optimizer was used. ADAM optimizer was chosen because it produces better results than other optimization algorithms [25], has a shorter computation time, and requires fewer tuning parameters.

3. TRANSFER LEARNING

Transfer Learning according to [8] is a feature that enables users to transfer the knowledge of pre-trained models and use it in their own problem set. Instead of creating a model from scratch to compare results in this work, models that are trained on large datasets such as ImageNet with 100,000 data points and explored the power of transfer learning were used which according to [37] is proven to be significant in many image classification types of studies. In this research, it was identified that CNNs modelled on relatively large datasets could serve as feature extractors for a wide range of image recognition tasks

to aid in enhanced performance, as compared to state-of-the-art approaches. Transfer learners on the VGG19, ResNet50, and MobileNetv2 were used, and evaluating their performances which are discussed in further sections.

4. EVALUATION METRICS

A metric helps in evaluating the performance of any designed model. In machine learning, classification problems involve two or more alternative outputs [38]. The performance of the model is evaluated using four metrics: accuracy, AUC, precision, and recall. Given that the distribution of instances across each target class in the dataset is balanced, accuracy is employed as the primary metric to assess model performance. Accuracy is defined as the ratio of correctly classified instances (both positive and negative) to the total number of instances in the dataset. Additionally, precision, recall, and accuracy are derived from the confusion matrix, which is composed of four components: False Positives (FP), True Positives (TP), False Negatives (FN), and True Negatives (TN). A classification report is used to compute precision and recall for each target class. Equations 6, 7, 8, and 9 provide the mathematical formulations for these metrics.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \quad (6)$$

$$\text{AUC} = \frac{1 + \text{TPR} - \text{FPR}}{2} \quad (7)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (8)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (9)$$

where $TP =$ True Positive prediction result, $TN =$ True Negative prediction result, $FP =$ False Positive prediction results, $FN =$ False Negative prediction result, TPR = True Positive Rate and FPR = False Positive Rate.

5. RESULTS AND DISCUSSIONS

To evaluate the performance of the model, the entire NIH dataset was split into three distinct sets: training, validation, and testing. The main idea behind this process is to ensure that the model generalizes well, to tune it appropriately during training, and to evaluate its performance accurately on unseen data. Without such a split, the model risks performing well on known data but failing to perform well when exposed to new, real-world data. In the first experiment, the dataset was split using the following proportions: 70% for training the model, 10% for validation and 20% for testing. This was achieved with the help of split-folders library in python. Table 3 presents a summary of this experiment.

Table 3: Split Proportion for Experience I

Training	Found 20252 images belonging to 2 classes	70%
Validation	Found 3410 images belonging to 2 classes	10%
Testing	Found 5922 images belonging to 2 classes	20%

Following the partitioning of the dataset into training, validation, and testing subsets, the model was subjected to iterative optimization using the training set for parameter updates via backpropagation and gradient descent. Concurrently, the validation set was leveraged to monitor performance metrics, enabling hyperparameter optimization and the implementation of regularization strategies such as early stopping to mitigate overfitting. Upon convergence of the training process, the testing set was utilized to conduct a comprehensive evaluation of the generalization of the model and robustness. The outcomes of this experimental workflow are systematically summarized in Table 4, providing insights into the performance metrics and behavioral trends observed.

Table 4: Results of Experiment I

Splitting Type	Accuracy	Precision	Recall	AUC
Training	0.9688	0.9355	1.0000	0.9778
Validation	0.9548	0.9477	0.9626	0.9871
Testing	0.9552	0.9885	0.9659	0.9457

In the first experiment, the model, trained for 50 epochs with a 70% training, 20% testing, and 10% validation data split, demonstrated strong performance across multiple metrics. It achieved a training accuracy of 96.88%, test accuracy of 95.52%, and validation accuracy of 95.48%, indicating effective convergence and robust generalization, with minimal overfitting, as shown in Fig. 10. The training loss stabilized at 15.73%, and the validation loss at 14.33%, reflecting effective model calibration, as depicted in Fig. 11. Precision, recall, and AUC scores further highlighted the model's effectiveness: training precision of 93.55%, recall of 100%, and AUC of 97.78%; validation precision of 94.77%, recall of 96.26%, and AUC of 98.71%; and test precision of 98.85%, recall of 96.59%, and AUC of 94.57%, as represented in Fig. 12, Fig. 13, and Fig. 14. These results confirm the model's ability to generalize well across different data partitions, achieving high accuracy, precision, recall, and AUC, while minimizing false positives and overfitting.

In the second experiment, the data was modified with split proportions to assess the model's performance under different conditions. Specifically, the dataset was divided into 60% for training, 20% for validation, and 20% for testing, using the same split-folders library to ensure consistency in data handling. As detailed in Table V, the training set contained 16,534 images across two classes, while the validation and testing set each contained 5,514 and 5,510 images, respectively, also distributed across the same two classes. This adjustment in the split ratio allowed us to investigate the impact of varying the amount of training data on the model's ability to generalize, as well as to assess its performance across both validation and testing subsets. The results from this experiment, including accuracy, precision, recall, and AUC, are summarized in Table VI, providing insights into how these changes in data partitioning influence the model's behavior and overall efficacy in different evaluation settings. This variation in data split helps us understand the trade-offs between using more data for training and ensuring sufficient data for validation and testing, contributing to a more comprehensive evaluation of the model's performance.

Table 5: Results of Experiment II

Splitting Type	Accuracy	Precision	Recall	AUC
Training	0.9688	0.9737	0.9737	0.9919
Validation	0.9555	0.9364	0.9775	0.9882
Testing	0.9567	0.9884	0.9688	0.9458

In the second experiment, the dataset was partitioned into 60%

for training, 20% for validation, and 20% for testing, ensuring balanced representation across all subsets. The model was trained for 50 epochs using identical hyperparameters to optimize its parameters effectively. The performance metrics are illustrated in Fig. 15 (Accuracy), Fig. 16 (Loss), Fig. 17 (Precision), Fig. 18 (Recall), and Fig. 19 (AUC). The model achieved a training accuracy of 96.88%, with validation and testing accuracies stabilizing at 95.55% and 95.67%, respectively (Fig. 15), indicating strong generalization with minimal overfitting. The training loss converged to 11.17%, while validation and testing losses settled at 13.77% and 13.22%, respectively, signifying efficient optimization and minimal performance degradation across splits (Fig. 16). Precision scores reached 97.37% (training), 93.64% (validation), and 98.84% (testing), highlighting the model's ability to minimize false positives (Fig. 17). Recall values of 97.37% (training), 97.75% (validation), and 96.88% (testing) demonstrate high sensitivity in identifying parasitized samples (Fig. 18). The Area Under the Curve (AUC) was 99.19% (training), 98.82% (validation), and 94.58% (testing), reflecting exceptional discriminative capability across all datasets (Fig. 19). These results confirm the model's robust performance in malaria parasite detection, achieving high accuracy, sensitivity, and specificity, with reliable generalization to unseen data.

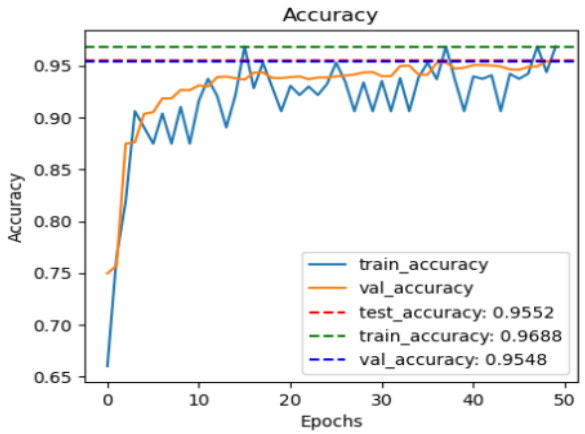

Fig 10: Learning Curve for Experiment I, depicting the accuracy of the model across successive training epochs.

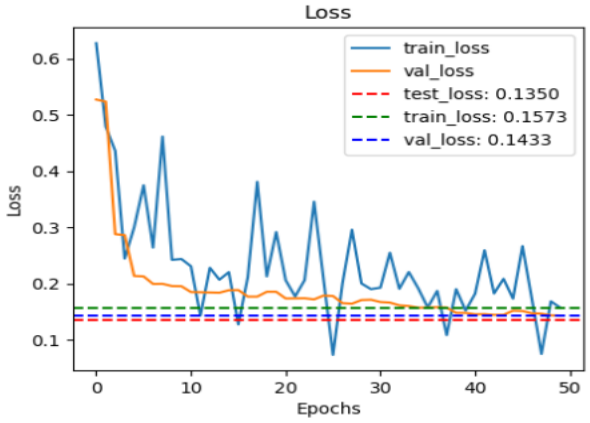

Fig 11: Learning Curve for Experiment I, showing the model's loss across training epochs.

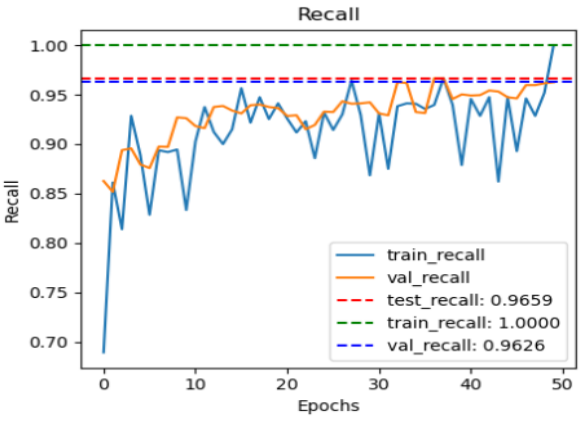

Fig 12: Learning Curve for Experiment I, illustrating the progression of the recall metric across training epochs.

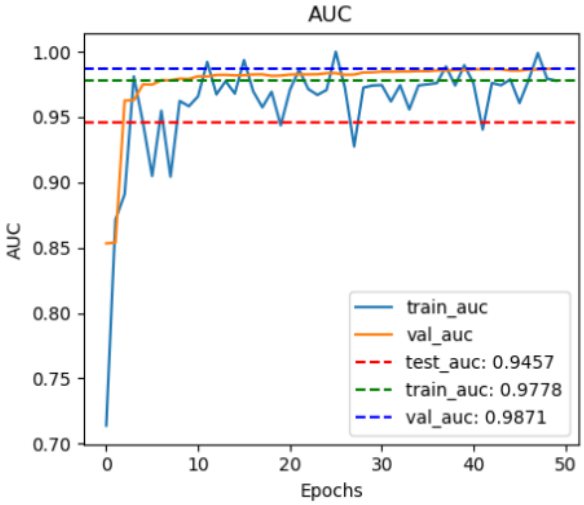

Fig 13: AUC Learning Curve for Experiment I, presenting the evolution of the Area Under the Curve (AUC) over successive training epochs.

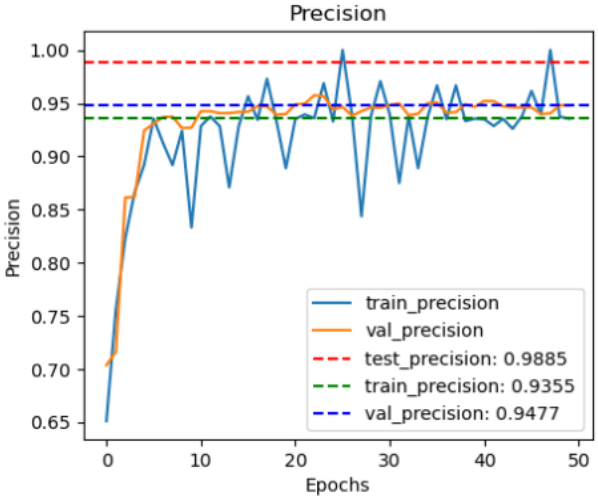

Fig 14: Learning Curve for Experiment I, representing the precision metric across training epochs.

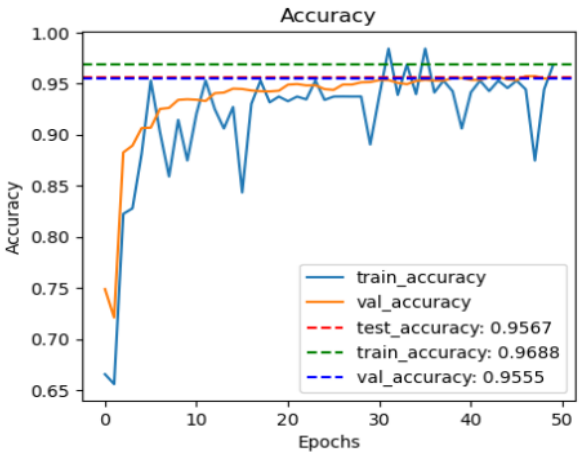

Fig 15: Learning curve for Experiment II, showing model accuracy across training epochs and its performance trends during optimization.

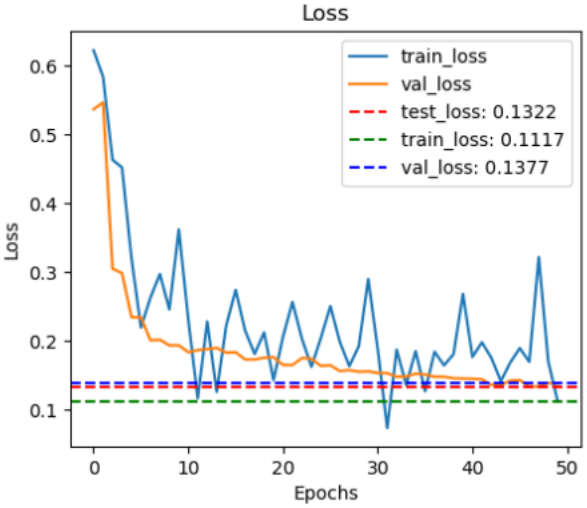

Fig 16: Learning Curve for Experiment II, showing the model's loss across training epochs.

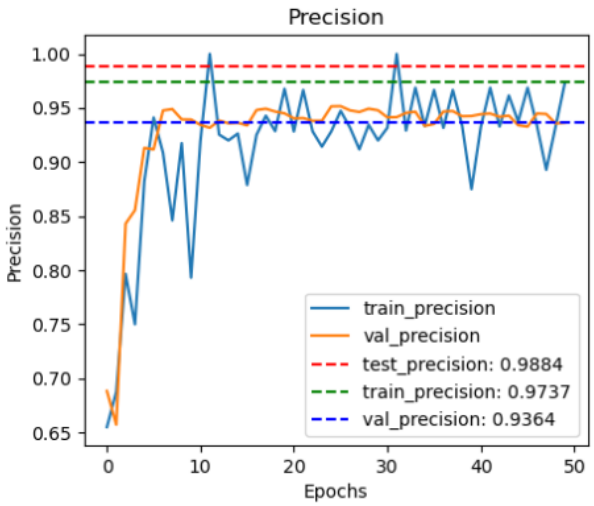

Fig 17: Learning Curve for Experiment II, depicting the precision of the model across successive training epochs.

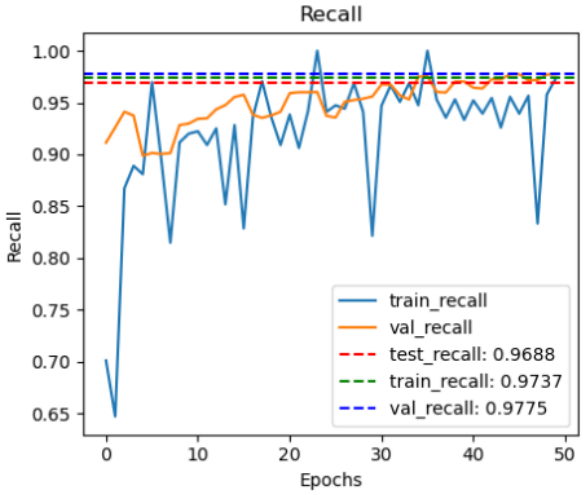

Fig 18: Recall Metric for Experiment II, showing the model's recall values across successive training epochs.

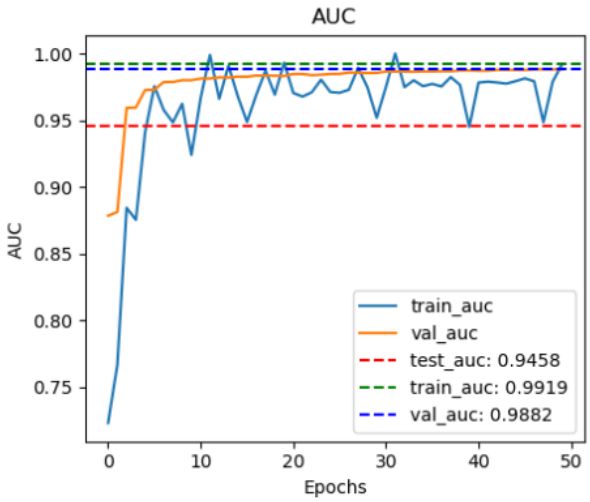

Fig 19: AUC Learning Curve for Experiment II, showing the model's AUC (Area Under the Curve) across successive training epochs.

5.1 Validating Model performance with IEEE Dataset

In In addition to the NIH dataset, which has been widely employed in prior research and utilized in this study, the model underwent further validation using an additional dataset sourced from the IEEE data repository to assess its generalization and robustness across distinct datasets. This evaluation was aimed at understanding how well the model adapts to datasets with different statistical distributions, feature representations, and inherent complexities, thereby providing a more comprehensive validation. Both the NIH and IEEE datasets were processed under identical conditions, with the model trained using the same hyperparameters for 50 epochs to ensure consistency in training protocols and eliminate confounding variables. The use of the IEEE dataset, known for its diverse set of images and complex class distributions, offered a more challenging environment for the model, testing its ability to generalize beyond the NIH dataset's characteristics. The results from the cross-validation experiments, including precision, recall, accuracy and AUC, are summarized and compared in Table VII, which provides detailed performance metrics for both datasets. These

comparative results allow for a rigorous assessment of the model's adaptability and its efficacy across different datasets, offering a more nuanced understanding of its potential deployment in varied real-world applications where data diversity is prevalent.

Table 6: Comparison of Model Performance on NIH and IEEE Dataset

Dataset	Accuracy	Precision	Recall	AUC
NIH	0.9688	0.9737	0.9737	0.9919
IEEE	0.9667	0.9724	0.9739	0.9993

To validate the efficacy of the model, three state-of-the-art transfer learning architectures were evaluated: MobileNetV2, ResNet50, and VGG19. The models were trained using varying splits of the training and validation datasets to assess their generalization capabilities. Computational efficiency, particularly the training time and resource utilization, was also quantified as part of the evaluation. The experimental results, including accuracy, loss, and other relevant metrics, are presented in Table 7.

Table 7: Comparison with Transfer Learners

Model	Accuracy	Precision	Recall	AUC
ResNet50	0.8564	0.7770	0.9646	0.9976
MobileNetv2	0.9348	0.9085	0.9628	0.9669
VGG19	0.8906	0.8316	0.9755	0.9782
Proposed Model	0.9688	0.9737	1.0000	0.9919

It can be observed that the proposed model outperformed other transfer learning architectures, as demonstrated in Table 8. This performance enhancement can be attributed to the sophisticated preprocessing pipelines and advanced data augmentation strategies employed, which significantly improved the model's capacity to generalize by exploiting a more heterogeneous and diverse training set. To further validate the superior performance of the model, it was benchmarked against results from prior methodologies within the same domain. A comprehensive comparative analysis of these works is summarized in Table 8, which underscores the advantages of the approach in terms of accuracy, robustness, and computational efficiency over existing techniques.

Table 8: Comparison with Related Works

Model	Dataset	Precision	Accuracy	Recall
HPTDL-MPDC [10]	NIH	0.8900	0.9100	0.9300
TL-SGAN [9]	NIH	N/A	0.9660	N/A
IDTL-MPDC [8]	NIH	0.9586	0.9586	0.9598
CNN [7]	NIH	0.9791	0.9792	0.9792
Random Forest [12]	NIH	0.9000	0.9000	0.9000
Proposed Model	**NIH**	**0.9737**	**0.9688**	**1.0000**

Table 8 above provides a comprehensive comparison of several machine learning models on their performance evaluated using the NIH dataset. The comparison is structured around three key performance metrics: Precision, Accuracy, and Recall, which provide insights into the reliability and effectiveness of each model.

6. CONCLUSIONS

In this study, experiments leveraging Convolutional Neural Network (CNN)-based deep learning algorithms were conducted to enhance the classification performance of malaria parasite detection from cell images. The primary dataset used for training and evaluation was the NIH malaria dataset, which contains 27,588 of parasitized and uninfected cell images. Advanced preprocessing techniques, including normalization, standardization, and staining, significantly contributed to improving the model's feature representation and overall performance. Additionally, techniques such as data augmentation (e.g., random cropping, flipping, rotation) and hyperparameter optimization algorithms were employed, yielding substantial improvements in model generalization and robustness. Various pre-trained architectures VGG-19, ResNet-50, MobileNetV2, and a custom-designed CNN were evaluated based on their accuracy, precision, recall, F1-score, and computational efficiency (in terms of training time and resource consumption). In the first experiment, the model was trained for 50 epochs with a 70% training, 20% testing, and 10% validation data split from the NIH dataset. The model demonstrated strong performance across multiple metrics, achieving a training accuracy of 96.88%, test accuracy of 95.52%, and validation accuracy of 95.48%, indicating effective convergence and robust generalization with minimal overfitting. The training loss stabilized at 15.73%, and the validation loss at 14.33%, reflecting efficient model calibration. Precision, recall, and AUC scores highlighted the model's effectiveness: training precision of 93.55%, recall of 100%, and AUC of 97.78%; validation precision of 94.77%, recall of 96.26%, and AUC of 98.71%; and test precision of 98.85%, recall of 96.59%, and AUC of 94.57%. These results confirm the model's ability to generalize well across different data partitions, achieving high accuracy, precision, recall, and AUC, while minimizing false positives and overfitting. In the second experiment, the dataset was partitioned into 60% for training, 20% for validation, and 20% for testing, ensuring balanced representation across all subsets. The model was trained for 50 epochs using identical hyperparameters to optimize its parameters effectively. The model achieved a training accuracy of 96.88%, with validation and testing accuracies stabilizing at 95.55% and 95.67%, respectively, indicating strong generalization with minimal overfitting. The training loss converged to 11.17%, while validation and testing losses settled at 13.77% and 13.22%, respectively, signifying efficient optimization and minimal performance degradation across splits. Precision scores reached 97.37% (training), 93.64% (validation), and 98.84% (testing), highlighting the model's ability to minimize false positives. Recall values of 97.37% (training), 97.75% (validation), and 96.88% (testing) demonstrated high sensitivity in identifying parasitized samples. The Area Under the Curve (AUC) was 99.19% (training), 98.82% (validation), and 94.58% (testing), reflecting exceptional discriminative capability across all datasets. These results confirm the model's robust performance in malaria parasite detection, achieving high accuracy, sensitivity, and specificity, with reliable generalization to unseen data. Additionally, the performance of the proposed model was validated using an independent dataset retrieved from the IEEE data repository. The model was trained with the same hyperparameters and configuration as in the initial experiments. Remarkably, the results on this new dataset were almost identical, with training accuracy of 96.67%, validation accuracy of 95.51%, and test accuracy of 95.53%. The precision, recall, and AUC scores also exhibited near-identical values, further confirming the robustness and generalizability of the model across different datasets. These consistent results

validate the model's capacity for reliable malaria parasite detection in diverse environments.

In conclusion, the highest performing metrics were observed in the second experiment, where the model achieved exceptional values across key metrics: a training accuracy of 96.88%, a precision of 97.37%, recall of 97.75%, and an outstanding AUC of 99.19% for training data. These results demonstrate the model's superior ability to detect malaria parasites with high sensitivity and specificity, indicating its potential for reliable, real-world deployment in malaria detection. Future research will focus on leveraging mobile-based imaging systems for the analysis of thin blood smear images captured with smartphones. The objective is to develop an efficient pipeline for detecting and quantifying various malaria parasite pairings capable of infecting humans. Ultimately, the goal is to design a cutting-edge, end-to-end diagnostic system that integrates with cloud computing and edge-based inference to enable accurate pre-diagnosis of malaria in low-resource, underserved environments.

7. ACKNOWLEDGMENTS

National Institute of Health (NIH) and IEEE are duly acknowledged for making their data publicly available, which was instrumental in training and validating the model.

8. REFERENCES

[1] World Malaria Report 2023. World Health Organization, 2023.

[2] E. Guemas et al., "Automatic patient-level recognition of four Plasmodium species on thin blood smear by a real-time detection transformer (RT-DETR) object detection algorithm: a proof-of-concept and evaluation," Microbiol Spectr, vol. 12, no. 2, Feb. 2024, doi: 10.1128/spectrum.01440-23.

[3] L. Zedda, A. Loddo, and C. Di Ruberto, "A deep architecture based on attention mechanisms for effective end-to-end detection of early and mature malaria parasites," Biomed Signal Process Control, vol. 94, Aug. 2024, doi: 10.1016/j.bspc.2024.106289.

[4] E. Hassan, M. Y. Shams, N. A. Hikal, and S. Elmougy, "A Novel Convolutional Neural Network Model for Malaria Cell Images Classification," Computers, Materials and Continua, vol. 72, no. 3, pp. 5889–5907, 2022, doi: 10.32604/cmc.2022.025629.

[5] A. Singh, M. Mehra, A. Kumar, M. Niranjannaik, D. Priya, and K. Gaurav, "Leveraging hybrid machine learning and data fusion for accurate mapping of malaria cases using meteorological variables in western India," Intelligent Systems with Applications, vol. 17, Feb. 2023, doi: 10.1016/j.iswa.2022.200164.

[6] P. A. Pattanaik, M. Mittal, M. Z. Khan, and S. N. Panda, "Malaria detection using deep residual networks with mobile microscopy," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 5, pp. 1700–1705, May 2022, doi: 10.1016/j.jksuci.2020.07.003.

[7] M. Bhuiyan and M. S. Islam, "A new ensemble learning approach to detect malaria from microscopic red blood cell images," Sensors International, vol. 4, Jan. 2023, doi: 10.1016/j.sintl.2022.100209.

[8] A. Alassaf and M. Y. Sikkandar, "Intelligent Deep Transfer Learning Based Malaria Parasite Detection and Classification Model Using Biomedical Image," Computers, Materials and Continua, vol. 72, no. 3, pp. 5273–5285, 2022, doi: 10.32604/cmc.2022.025577.

[9] I. Amin, S. Hassan, S. B. Belhaouari, and M. H. Azam, "Transfer Learning-Based Semi-Supervised Generative Adversarial Network for Malaria Classification," Computers, Materials and Continua, vol. 74, no. 3, pp. 6335–6349, 2023, doi: 10.32604/cmc.2023.033860.

[10] T. K. Kundu, D. K. Anguraj, and S. V. Sudha, "Modeling a Novel Hyper-Parameter Tuned Deep Learning EnabledMalaria Parasite Detection and Classification," Computers, Materials and Continua, vol. 77, no. 3, pp. 3289–3304, 2023, doi: 10.32604/cmc.2023.039515.

[11] A. Koirala et al., "Deep Learning for Real-Time Malaria Parasite Detection and Counting Using YOLO-mp," IEEE Access, vol. 10, pp. 102157–102172, 2022, doi: 10.1109/ACCESS.2022.3208270.

[12] D. Crossed D Signumic, D. Keco, and Z. Mašetic, "Automatization of Microscopy Malaria Diagnosis Using Computer Vision and Random Forest Method," in IFAC-PapersOnLine, Elsevier B.V., 2022, pp. 80–84. doi: 10.1016/j.ifacol.2022.06.013.

[13] I. Jdey, G. Hcini, and H. Ltifi, "Deep learning and machine learning for Malaria detection: overview, challenges and future directions," Sep. 2022, [Online]. Available: http://arxiv.org/abs/2209.13292

[14] Y. Alraba'nah and W. Toghuj, "A deep learning based architecture for malaria parasite detection," Bulletin of Electrical Engineering and Informatics, vol. 13, no. 1, pp. 292–299, Feb. 2024, doi: 10.11591/eei.v13i1.5485.

[15] O. S. Zhao et al., "Convolutional neural networks to automate the screening of malaria in low-resource countries," PeerJ, vol. 8, 2020, doi: 10.7717/peerj.9674.

[16] "Convolutional Neural Network: A Complete Guide." Accessed: Dec. 20, 2024. [Online]. Available: https://learnopencv.com/understanding-convolutional-neural-networks-cnn/

[17] S. Shambhu, D. Koundal, P. Das, V. T. Hoang, K. Tran-Trung, and H. Turabieh, "Computational Methods for Automated Analysis of Malaria Parasite Using Blood Smear Images: Recent Advances," Comput Intell Neurosci, vol. 2022, 2022, doi: 10.1155/2022/3626726.

[18] E. Prasetyo, R. Purbaningtyas, R. D. Adityo, N. Suciati, and C. Fatichah, "Combining MobileNetV1 and Depthwise Separable convolution bottleneck with Expansion for classifying the freshness of fish eyes," Information Processing in Agriculture, vol. 9, no. 4, pp. 485–496, Dec. 2022, doi: 10.1016/j.inpa.2022.01.002.

[19] D. T. Rademaker et al., "Quantifying the deformability of malaria-infected red blood cells using deep learning trained on synthetic cells," iScience, vol. 26, no. 12, Dec. 2023, doi: 10.1016/j.isci.2023.108542.

[20] S. Sawant and A. Singh, "Malaria Cell Detection Using Deep Neural Networks," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.20005

[21] M. Mujahid et al., "Efficient deep learning-based approach for malaria detection using red blood cell smears," Sci Rep, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-63831-0.

[22] "Malaria Datasheet." Accessed: Dec. 20, 2024. [Online]. Available: https://lhncbc.nlm.nih.gov/LHC-research/LHC-projects/image-processing/malaria-datasheet.html

[23] F. Yang et al., "Cascading YOLO: automated malaria parasite detection for Plasmodium vivax in thin blood smears," p. 58, Mar. 2020, doi: 10.1117/12.2549701.

[24] "Malaria Thick Blood Smears | IEEE DataPort." Accessed: Dec. 20, 2024. [Online]. Available: https://ieee-dataport.org/documents/malaria-thick-blood-smears

[25] F. Yang et al., "Deep Learning for Smartphone-Based Malaria Parasite Detection in Thick Blood Smears," IEEE J Biomed Health Inform, vol. 24, no. 5, pp. 1427–1438, May 2020, doi: 10.1109/JBHI.2019.2939121.

[26] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," Global Transitions Proceedings, vol. 3, no. 1, pp. 91–99, Jun. 2022, doi: 10.1016/J.GLTP.2022.04.020.

[27] "Image Filtering Techniques in Image Processing." Accessed: Dec. 20, 2024. [Online]. Available: https://www.imageprovision.com/articles/understanding-image-filtering-techniques-in-image-processing.

[28] P. V. Dantas, W. Sabino da Silva, L. C. Cordeiro, and C. B. Carvalho, "A comprehensive review of model compression techniques in machine learning," Applied Intelligence 2024 54:22, vol. 54, no. 22, pp. 11804–11844, Sep. 2024, doi: 10.1007/S10489-024-05747-W.

[29] M. Mafi, H. Martin, M. Cabrerizo, J. Andrian, A. Barreto, and M. Adjouadi, "A comprehensive survey on impulse and Gaussian denoising filters for digital images," Signal Processing, vol. 157, pp. 236–260, Apr. 2019, doi: 10.1016/J.SIGPRO.2018.12.006.

[30] S. Elmi and Z. Elmi, "A robust edge detection technique based on Matching Pursuit algorithm for natural and medical images," Biomedical Engineering Advances, vol. 4, p. 100052, Dec. 2022, doi: 10.1016/J.BEA.2022.100052.

[31] X. Li, "Image Texture Analysis and Edge Detection Algorithm Based on Anisotropic Diffusion Equation," Advances in Mathematical Physics, vol. 2021, no. 1, p. 9910882, Jan. 2021, doi: 10.1155/2021/9910882.

[32] K. Muntarina, R. Mostafiz, F. Khanom, S. B. Shorif, and M. S. Uddin, "MultiResEdge: A deep learning-based edge detection approach," Intelligent Systems with Applications, vol. 20, Nov. 2023, doi: 10.1016/j.iswa.2023.200274.

[33] S. S. Bao, Y. R. Huang, J. C. Xu, and G. Y. Xu, "Pixel Difference Unmixing Feature Networks for Edge Detection," IEEE Access, vol. 11, pp. 52370–52380, 2023, doi: 10.1109/ACCESS.2023.3279276.

[34] S. Seoni et al., "All you need is data preparation: A systematic review of image harmonization techniques in multi-center/device studies for medical support systems," Comput Methods Programs Biomed, vol. 250, p. 108200, Jun. 2024, doi: 10.1016/J.CMPB.2024.108200.

[35] M. Salvi, F. Branciforti, F. Molinari, and K. M. Meiburger, "Generative models for color normalization in digital pathology and dermatology: Advancing the learning paradigm," Expert Syst Appl, vol. 245, p. 123105, Jul. 2024, doi: 10.1016/J.ESWA.2023.123105.

[36] N. Schiess, A. Villabona-Rueda, K. E. Cottier, K. Huether, J. Chipeta, and M. F. Stins, "Pathophysiology and neurologic sequelae of cerebral malaria," Malar J, vol. 19, no. 1, Jul. 2020, doi: 10.1186/S12936-020-03336-Z.

[37] A. M. Qadri, A. Raza, F. Eid, and L. Abualigah, "A novel transfer learning-based model for diagnosing malaria from parasitized and uninfected red blood cell images," Decision Analytics Journal, vol. 9, p. 100352, Dec. 2023, doi: 10.1016/J.DAJOUR.2023.100352.

[38] M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," Computers 2023, Vol. 12, Page 91, vol. 12, no. 5, p. 91, Apr. 2023, doi: 10.3390/COMPUTERS12050091.