Malaria Detection - Final Report

Suhaib Tariq
30075751

Alina Mansuri
30008370

I. Abstract

A. Succinct description of your project and main outcomes.

This project aimed to improve the accuracy of malaria detection using computer vision as a replacement for microscopy, which is only able to accurately detect malaria in 88 percent of cases. The project used various machine learning algorithms, with a focus on Convolutional Neural Networks (CNNs). The results showed that CNNs were able to detect malaria with an accuracy of 98.58 percent, which is a significant improvement over microscopy. The conclusion is that computer vision-based approaches can be highly effective for malaria detection, especially in regions where access to traditional diagnostic methods is limited.

II. Introduction

A. What is the project's goal and objectives?

Malaria is a life-threatening disease that are transmitted through the bites of infected mosquitos. According to the World Health organization, there were an estimated 229 million cases of malaria worldwide in 2019, leading to approximately 409,000 deaths, with majority of the cases occuring in sub-Saharan Africa. Since human beings are less able to detect the disease, early and accurate diagnosis of malaria is crucial for effective treatment and contain the spread of the disease.

The aim of this project is to develop a computer vision-based system that is able to detect and differentiate accurately, both benign and malignant tumors from the microscopic images of bood samples. In doing this, we have used several evaluation metrics, of which includes F-score, precision, recall amongst many others to help us further ensure our goal is met.

B. Importance in the field of computer vision

Since less-developed countries have less ability to access resources, it is devastating for those individuals who are not able to seek the treatment that they need. Computer vision, has the potential to alleviate this issue by improving the speed, accuracy and scalability of the diagnosis, preventing the risk of human error and increasing the likelihood that the individual will survive, in cases where there is a shortage of trained personnel.

Thus, computer vision in malaria can play a significant role in saving lives and can provide a platform of which many can rely on computer's to predict more accurately in cases where it is even harder to detect other medical issues such as brain tumors where humans can detect only up to 0.1 mm, which can be problematic if not detected early enough.

C. What have others done in this space?

In proceeding forth with our project, we will be looking and reviewing several related work that focuses on existing malaria detection techniques such as traditional diagnosis and other computer- vision related work. This includes work by K.M Fuhud who used three convolutional layers, followed by a max pooling layer for each layer and used a decoder, which consists of 4 convolutional layers and Up-sampling layers to detect malaria, which was looked upon when building our model for malaria detection. This brought the study of Rishika Kapoor's work along who relied on transfer learning of VGG19 and ResNet50 in the detection of malaria. This exploration of other computer-vision related work will enable us to think of various other ways to improve the existing model to help contribute to the early detection of malaria, and prevent the number of deaths that occur worldwide.

III. Materials and Model Architecture

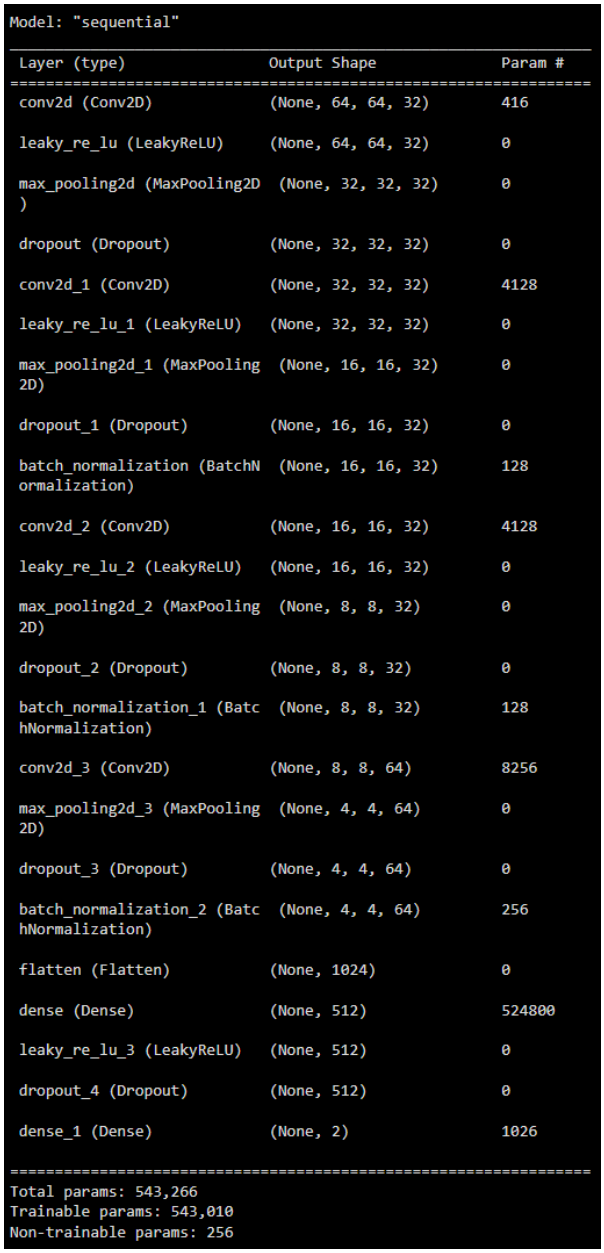
We examined several architectures and found that CNN performed well with a test accuracy of 98.58%. This architecture involved several layers.

The first layer is convolutional layer, which uses a 2D filter with 32 output channels and a kernel size of 2. The padding is set to "same", which allowed the output to have the same size as the input. The input shape is (64,64,3) which means that the input images have a size of 64 * 64 pixels and 3 color channels (RGB). The output of this layer is passed through a LeakyRelU activation function with a slope of 0.1, which helps the model avoid the vanishing gradient problem. Next, a max pooling layer with a pool size of 2 was applied which was needed to reduce the spatial dimensions of the output, followed by a dropout layer with a rate of 0.2, which was applied to reduce overfitting.

The same set of layers(convolutional, Leaky RelU activation ,max pooling and dropout) is repeated for the second and third layers, each with a batch normalization layer added after the max pooling layer. The output of the final dropout layer is flattened into a 1D vector, and then passed through a fully connected layer with 512 neurons and a LeakyRelU activation function. Another dropout layer with a rate of 0.4 was applied to reduce overfitting. Finally, the final layer is a dense layer with 2 neurons and a softmax activation function, which is used for binary classification tasks.

This choice of activation functions, namely Leaky RelU and RelU, along with batch normalization layers helps to avoid the vanishing gradient problem and improve performance. Dropout layers was used to prevent overfitting.

This architecture, CNN was used because it provided high accuracy and is commonly used for image classification and object recognition tasks. The convolutional layers are responsible for learning features in an image, activation was needed to add non-linearity of the model, pooling to reduce dimensions of the feature whilst the parameter used was to ensure important features were not lost, and fully connected layers were for classification. This architecture was used because it was able to extract features from images effectively and thus, this state-of-the-art model was specifically designed for image processing and thus, it is suitable for our goal of malaria detection. A visual overview of the visual representation of architecture is provided below:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 32)        416

 leaky_re_lu (LeakyReLU)     (None, 64, 64, 32)        0

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

 dropout (Dropout)           (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        4128

 leaky_re_lu_1 (LeakyReLU)   (None, 32, 32, 32)        0

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 dropout_1 (Dropout)         (None, 16, 16, 32)        0

 batch_normalization (BatchN  (None, 16, 16, 32)       128
 ormalization)

 conv2d_2 (Conv2D)           (None, 16, 16, 32)        4128

 leaky_re_lu_2 (LeakyReLU)   (None, 16, 16, 32)        0

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 dropout_2 (Dropout)         (None, 8, 8, 32)          0

 batch_normalization_1 (Batc  (None, 8, 8, 32)         128
 hNormalization)

 conv2d_3 (Conv2D)           (None, 8, 8, 64)          8256

 max_pooling2d_3 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 dropout_3 (Dropout)         (None, 4, 4, 64)          0

 batch_normalization_2 (Batc  (None, 4, 4, 64)         256
 hNormalization)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 512)               524800

 leaky_re_lu_3 (LeakyReLU)   (None, 512)               0

 dropout_4 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 2)                 1026

=================================================================
Total params: 543,266
Trainable params: 543,010
Non-trainable params: 256
```

Fig. 1. Architecture of the CNN Model

The research conducted by Fuhad and his team used a CNN-KNN model,in conjunction with a smartphone application in which our project did not utilize KNN and no smartphone was used but instead CNN was used where images were resized to 64, which was also done in their project, along with the same learning rate of 0.001 and a momentum of 0.9 with a stochastic gradient descent(SCD) which different from ours as we used Adams as well as a early stopping mechanism so that it is able to learn effectively and apply what it is able to learn to new unseen,test images and stops before it overfits. In addition to this, the format was the same but they used different parameter size for example 2 * 2 pooling, 8 layers of convolutional of 28 * 28. In addition to this, they used an autoencoder model in which they had much more layers for example, a convolutional layer had 3 16*16, which is a lot compared to our model but in addition to this, it decoded the information in the dataset. The total system constituted of three convolutional layers, with a Max Pooling Layer that follows each layer. Within the convolutional layers, kernel of 3 *3 was used with same padding and 1 pixel stride. The kernel number for first, second and third convolutional layers were respectively 16,8 and 4.This encoding, was followed by decoding, which consists of 4 deconvolutional layers and 3 up-sampling layers.These layers of decoding had same stride size of 1, same padding and kernel numbers of 4,8,16 and 3 were used respectively. RelU was used in the hidden layers to introduce non-linerarity. During the procedure of testing, they had replaced the decoder with the flattened and fully connected layers(two). This was followed my softmax layer to help with the prediction of two classes: uninfected and parasitized.

Another paper by Rishika Kapoor used label-encoding where malaria was used for 1 and uninfected as 0, along with normalization and standardization, as well as data augmentation. This VGG19 model, consisted of “16 convolutional layers and 3 fully connected layers with five max-pool layers”. The activation function used was RelU along, with Adams optimization strategy, which was similar to our project. The last two blocks were unfrozen but for us, we froze block five, where we applied preprocessing techniques such as data augmentation and normalization along with softmax unlike this project which used sigmoid. The second model that was used was from pre-trained frozen layers of CNN, which was not discussed in much detail. In fine-tuning their third model, initial three blocks were frozen using transfer learning trained block four and five using a trainable dataset. This was then applied to the VGG model, where the last two blocks (four and five) were unfrozen so that their weights get updated in each iteration. In classifying our problem, the sigmoid activation function was used.

However, their ResNet50 performed best, in which they had “freezing initial blocks to get the output without updating the weights”. This architecture consisted of 5 stages,” each with a convolution block and identity block. Each convolutional block has three convolutional layers, and each identity block also has three convolutional layers”, with sigmoid being their activation function. In our architecture, the input images were

resized to 64 * 64 with 3 color channels (RGB) and the output of the last convolutional layer was obtained, which is the cov5_block3_out, and weights weren't updated unlike Rishika Kapoor and the output was flattened, passing through a dense layer with 256 units and a RelU activation. Furthermore, softmax activation was used to output the probabilities of the input image. In addition to this, the CNN they used were of great depth as well e.g. a CNN layer had 128 3*3.

IV. Dataset

For this project, we will be using the "Cell Images for Detecting Malaria" dataset, which can be found through Kaggle, which consists of a total of 27,558 cell images, which consist of both infected and uninfected cells. The source of the dataset is the National Institutes of Health(NIH), which provides an open-access repository of biomedical images.

The images in this dataset were preprocessed by converting them to grayscale followed by the conversion of the images to a uniform size of 224 * 224 pixels of which data augmentation techniques were soon applied such as random rotations, flips and zooms to artificially increase the diversity and thus, the size of the dataset which was essential to help reduce overfitting during the training of the model.

A bias inherent in this dataset is that these biomedical images were collected from laboratory cultures, namely from a hospital in Bangladesh and thus, may not accurately reflect the wide range of malaria that is found in the real-world settings and thus, can affect the accurate prediction of those suffering from malaria, preventing their early treatment. This leads to further discussion of two types of cells that are found: infected and uninfected, since there is no in-between, this may also be problematic in certain cases. In addition to this, they were collected from a single type of microscope, which can also further block the details as some microscopes have better features and thus, we may not be able to detect using computer vision as what comes out is as good as what comes in. In addition to this, there is also an imbalance of data where 150 individuals are infected and 50 samples are uninfected. This is a problem as the network is not trained on enough uninfected samples and this may cause an issue in detecting an uninfected cell as a infected cell, leading to a false positive.

V. Training Procedure

A. Description of the training process, including hardware, software, and frameworks used.

The training process for this CNN model is done using the Keras library in Python. The imports used in model_3 include libraries for data preprocessing (e.g., MinMaxScaler), deep learning model building (e.g., Sequential, Conv2D, Dense, Dropout, BatchNormalization, etc.), optimization algorithms (e.g., RMSprop, Adam, SGD), and training callbacks (e.g., EarlyStopping, ModelCheckpoint).

MinMaxScaler is used to normalize the pixel values of the input images. Sequential is the model-building API used to define a sequential deep learning model. The different layers of the model are defined using Conv2D, Dense, Dropout, BatchNormalization, and MaxPooling2D. RMSprop, Adam, and SGD are optimization algorithms used to minimize the loss function during training. EarlyStopping is a regularization method to prevent overfitting, and ModelCheckpoint is used to save the best model weights during training. Finally, load_img is used to load the input images, and cv2_imshow is used to display images in the Google Colab notebook environment.

We trained our model on the dataset described above, in this data set we applied the following augmentations :

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

# Using ImageDataGenerator to generate images
# Added vertical flipping, noise, lighting conditions, perspective transformation
train_datagen = ImageDataGenerator(horizontal_flip = True,
                                   vertical_flip  = True,
                                   zca_whitening = True,
                                  zoom_range = 0.5,shear_range = 0.2,
                                   channel_shift_range=0.8)
```

Fig. 2. Augemntations applied to cell images

- The horizontal flip randomly flips the image horizontally, creating a mirror image.
- The vertical flip randomly flips the image vertically, creating an upside-down version of the original image.
- ZCA whitening is a transformation that reduces the correlation between adjacent pixels in the image by rotating the image's pixel values along the principal component axes of the data.
- Zooming randomly applies a zoom-in or zoom-out effect to the image, giving the impression of the image appearing closer or further away.
- Shearing is a transformation that tilts the image, creating a slanted effect.
- Channel shifting is a transformation that randomly shifts the pixel values of one or more color channels in the image, creating a color shift effect.

We have probability applied for these so each image has a certain chance of these augmentations being applied. All these augmentations ensure that our model can learn from new and unseen data and not just the data we have provided from the data-set. This reduces over-fitting and increases the chances of a higher accuracy for our model.

B. Explanation of the optimization algorithm, learning rate, and other relevant hyperparameters

The model uses the Adam optimizer, which is an adaptive learning rate optimization algorithm that is well-suited for training deep neural networks. The Adam optimizer adjusts the learning rate of each weight parameter adaptively during training, based on estimates of the first and second moments of the gradients.

- The learning rate is set to the default value of 0.001.
- The batch size is set to 32. Batch size of 32 means that the model processes 32 training examples at a time before updating the weights of the neural network. Larger batch

sizes may be more efficient in terms of processing time, but can lead to overfitting and decreased generalization performance

- The model uses a categorical cross-entropy loss function. It measures the dissimilarity between the predicted probability distribution and the actual probability distribution.
- The model is trained for 40 epochs.
- The early stopping callback is used to prevent overfitting, with a patience of 2. This monitors the validation loss of the model and stops training if the validation loss does not improve after a certain number of epochs, which in our case is 2 (patience)
- EarlyStopping was used to monitor the validation loss and stop the training process when there is no improvement in the validation loss.
- ModelCheckpoint was used to save the weights of the best model during the training process. The monitor parameter specifies the metric to be monitored, and the save_best_only parameter ensures that only the weights of the best model are saved.
- The model checkpoint callback is used to save the weights of the best model based on validation loss.
- The LeakyReLU activation function is used with a slope of 0.1. The slope of the LeakyReLU function controls how much the function "leaks" by specifying the gradient of the function for negative input values. In this model, a slope of 0.1 is used, which means that for negative input values, the function will have a small negative gradient of -0.1.
- Dropout regularization with a rate of 0.2 and 0.4 is used in different parts of the model. A dropout rate of 0.2 means that 20% of the neurons in that layer will be randomly dropped out during training, while a dropout rate of 0.4 means that 40% of the neurons will be dropped out. By using dropout at different rates in different parts of the network, the model can better balance the trade-off between overfitting and underfitting.This forces the remaining neurons to learn more robust and generalized representations of the data, instead of relying on specific combinations of neurons that may be more susceptible to overfitting.
- Batch normalization is used in multiple layers to improve the training stability and performance. Batch normalization normalizes the input data to each layer, which means that the input data is scaled to a similar range. It also helps prevent overfitting in our model by reducing the effect of outliers or noise in the input data

C. Discussion of any techniques used to prevent overfitting (e.g., dropout, regularization, early stopping)

Several techniques are used to prevent overfitting. One of the most common techniques is dropout, which randomly drops out nodes in the neural network during training to prevent the model from relying too heavily on any one node. In this model, dropout is used after each max pooling layer, with a rate of 0.2.

Another technique used in this model is batch normalization, which normalizes the inputs of each layer to reduce the internal covariate shift and improve training stability. In the model_3, batch normalization is used after the second and third convolutional layers.

Finally, early stopping is also used in this model to prevent overfitting. Early stopping stops training when the validation loss stops improving, which prevents the model from overfitting to the training data. In this code block, early stopping is used with a patience of 2 epochs, which means that training will stop after two epochs of no improvement in the validation loss.

D. Presentation of training and validation loss curves and/or other performance metrics.

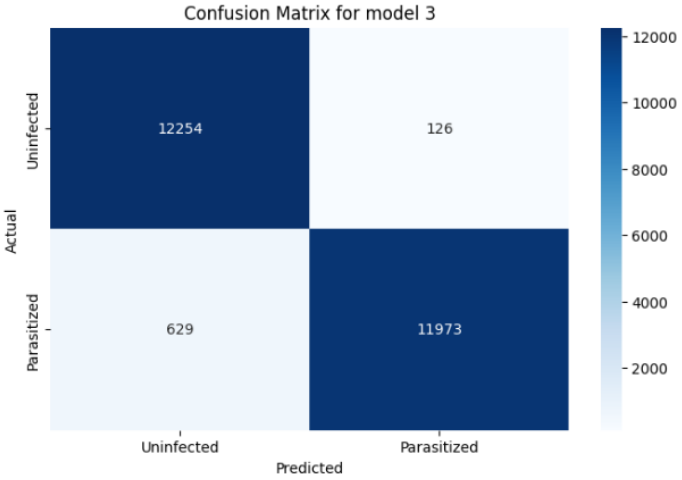


Fig. 3. Confusion matrix for the CNN Model (Model 3)

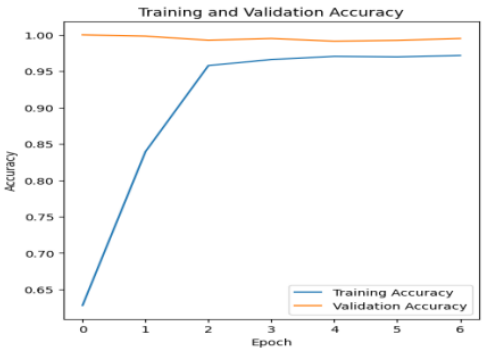


Fig. 4. Training and Validation accuracy for the CNN Model (Model 3)

VI. Evaluation

The evaluation metrics used to assess the performance of the model include precision, recall, and F1 score, as well

as a confusion matrix that provides the number of correct and incorrect predictions made by the model for both the uninfected and parasitized samples. Precision refers to the ratio of true positives to the total number of predicted positives, while recall refers to the ratio of true positives to the total number of actual positives. F1-score is the harmonic mean of precision and recall and is used to balance the two metrics when evaluating the performance of the model.

In addition to these classification metrics, the evaluation also included the use of validation and test loss to monitor the performance of the network during training. Validation loss was used to prevent overfitting by monitoring the performance of the model on a separate validation set while test loss was used to evaluate the performance on unseen data. The difference between validation and test loss was also used to detect underfitting or overfitting; a lower validation loss compared to the test loss indicates overfitting, while high values for both suggest underfitting. To provide a comprehensive evaluation, the achieved accuracy values were also compared to other studies in the literature as the accuracy for their studies was provided. In our study, we achieved an accuracy of 98.58%, which is somewhat comparable to other studies that used similar datasets, where they had obtained an accuracy of 99.12% however, this was due to removing of mislabelled samples by experts permitting them to obtain the high accuracy. Thus, the reduced dataset was not a limitation but was in fact a limitation for our study as there was incorrect labels that incorrectly classified uninfected as parasitized and vice versa.

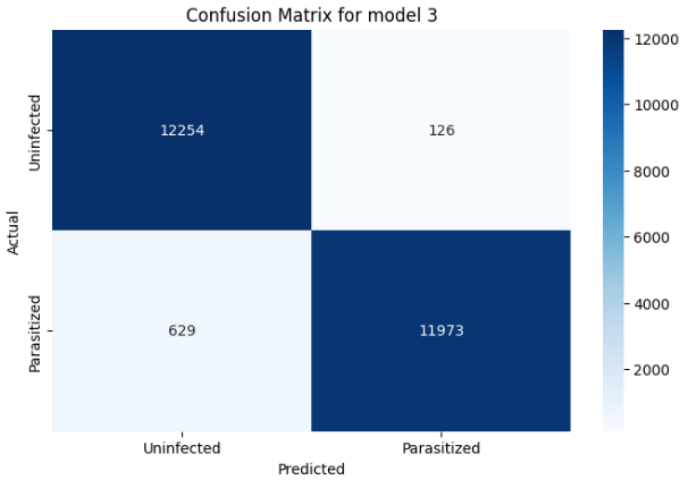


Fig. 5. Confusion matrix for the CNN Model (Model 3)

Figure 5&6 : We can see it performed well on uninfected and uninfected as well as detecting parasitized and parasitized in the confusion matrix (fig. 5) and therefore, the scores for precision and recall are relatively the same. Given the classification report (fig. 6), our goal of high recall are met and thus, we have strong evidence for classifying malaria accurately.

Figure 7: This shows the validation and training accuracy

781/781 [==============================] - 45s 57ms/step

	precision	recall	f1-score	support
0	0.95	0.99	0.97	12380
1	0.99	0.95	0.97	12602
accuracy			0.97	24982
macro avg	0.97	0.97	0.97	24982
weighted avg	0.97	0.97	0.97	24982

Fig. 6. Precision, recall and f1-score for CNN Model (Model 3)

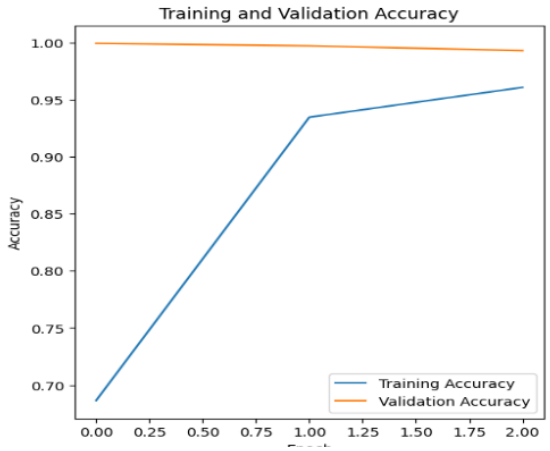


Fig. 7. Training and Validation Accuracy for CNN model (Model 3)

of over 2 epochs. From this figure, we see no signs of overfitting and generalizes well, where we can see the validation accuracy at its optimum of 1.00 epoch and thus, this model is recommended to be saved at 1.00 epoch, which is our final epoch.

In comparison to Rishika's study, our CNN with reduced layers achieved an accuracy of 98.58% while their resNet model achieved an accuracy of 95.7% which is higher than our achieved accuracy of 50.44% for ResNet. On further analysis of the confusion matrix provided of the VGG19 model, which provides quantitative information on the performance of the model, it can also be used as a qualitative result as it provides insight into the types of errors the model is making. By examining the confusion matrix, it enables us to gain an understanding of how the model is performing on uninfected and parasitized classes and help us identify potential areas for improvement. In light of this, our model correctly classified 11,795 uninfected images as uninfected, incorrectly classified 585 parasitized images as uninfected, correctly classified 12,044 parasitized images as parasitized and incorrectly classified 958 uninfected images as parasitized. The second model, which was conducted under the study of Rishika correctly classified 4,063 uninfected images as uninfected, incorrectly classified 81 parasitized images as uninfected, correctly classified 3,929 parasitized images as parasitized, and incorrectly classified 195 uninfected images as parasitized, which showed that the second model achieved lower counts due to access to better resources to remove classification error and could also be due to the fact that more

layers was used to filter out unnecessary information enabling more accurate detection of malaria.

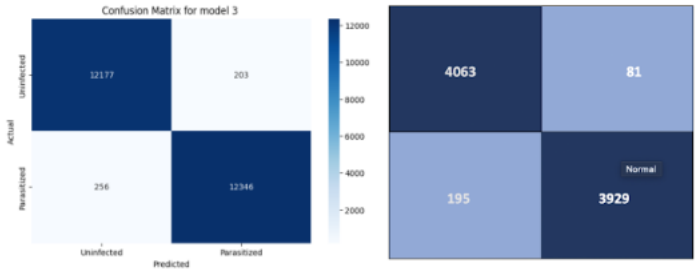


Fig. 8. Results on left of our study in comparison with the results from confusion matrix from VGG-19, obtained from Rishika Kapoor

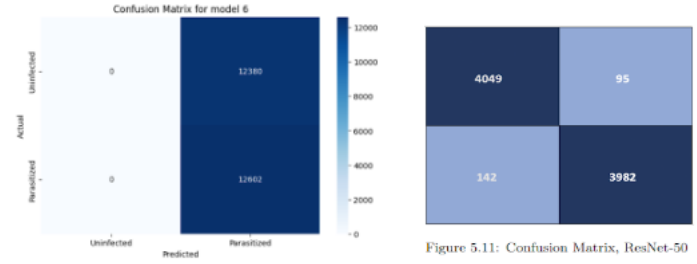


Figure 5.11: Confusion Matrix, ResNet-50

Fig. 9. Results on left of our study in comparison with the results from confusion matrix from ResNet-50, obtained from Rishika Kapoor

In discussing about resNet50, it was able to correctly classify 12,380 parasitized images as parasitized, incorrectly classified 585 uninfected images as parasitized and did not classify the uninfected images correctly. Under Rishikas study, it correctly classified 142 uninfected images as uninfected, correctly classified 3,982 parasitized images as parasitized, incorrectly classified 95 uninfected images as parasitized, and incorrectly classified 4,049 uninfected images as parasitized. From this result of the confusion matrix, it seems to suggest that Rishika's study was able to perform much better in comparison to ours but this could be due to the same factors that affected the performance of the VGG.

VII. Discussion

To provide a comprehensive evaluation, the achieved accuracy values were also compared to other studies in the literature as the accuracy for their studies was provided. In our study, we achieved an accuracy of 98.58%, which was found through different variations of the CNN model (adjusting various hyperparameters such as number of layers, batch size) but this high accuracy was obtained using model 3, which involved 4 convolutional layers, and using a leakyRelU with a slope of 0.1 along with a batch normalization.

This result of 98.58In comparison to Rishika's study, our CNN with reduced layers achieved an accuracy of 98.58In discussing about resNet50, it was able to correctly classify 12,380 parasitized images as parasitized, incorrectly classified 585 uninfected images as parasitized and did not classify the uninfected images correctly. Under Rishikas study, it correctly classified 142 uninfected images as uninfected, correctly classified 3,982 parasitized images as parasitized, incorrectly classified 95 uninfected images as parasitized, and incorrectly classified 4,049 uninfected images as parasitized. From this result of the confusion matrix, it seems to suggest that Rishika's study was able to perform much better in comparison to ours but this could be due to the same factors that affected the performance of the VGG. Based on the results obtained in this study, it can be concluded that the CNN model with reduced layers achieved a reasonably high accuracy of 98.30

To further improve the accuracy of the CNN model in detecting malaria in microscopic blood images, future research could explore the use of deeper architectures such as resNet50 or other advanced techniques. Additionally, using a larger dataset with fewer mislabeled samples could improve the accuracy of the model. Another possible direction for future research is to explore the use of transfer learning to reduce the amount of training data required and improve the accuracy of the model. This study has implications for the field of computer vision in the medical domain, as accurate detection of diseases from medical images can aid in early diagnosis and treatment, potentially saving lives. Furthermore, future directions can include the use of different types of microscopy images such as fluorescence microscopy, which provides a higher resolution of the images can increase the accuracy of the CNN algorithm as more important features can be seen. This integration with a smartphone app can increase accessibility to malaria diagnosis, especially in regions with limited resources, enabling individuals to take images of their blood samples using their smartphone cameras and receive a diagnosis within minutes, without the need for specialized laboratory equipment or trained personnel. However, when pursuing this future project, it is vital that the accuracy of the algorithm is not altered when integrated with such tools. Expanding the study to include a larger and more diverse population, as well as having access to several experts that can correctly identify the parasitized and uninfected cells, is also crucial in validating the accuracy of the model. This would help address the issue of mislabelling that has been identified in the current study and ensure that the model is effective in detecting malaria parasites in different populations.

VIII. Conclusion

The main finding of this project was that computer vision-based approaches, particularly the use of Conventional Neural Networks(CNN) was highly effective in detecting malaria, achieving an accuracy of 98.58

The impact of this project can be significant, as it provides a potential alternative to traditional microscopy for malaria detection, which can improve diagnosis rates and potentially save lives. Future directions of this project could lie in further improvement and refinement of the optimization of the CNN algorithm by exploring different types of layers and adjusting the parameters of the existing layers. and expanding the study to include a larger and more diverse population, as well as

having access to several experts that can correctly identify the parasitized and uninfected as the mislabelling has increased the likelihood of the inability to detect malaria as well as it could have. This is essential to ensure accuracy of the model is validated and increases the effectiveness of detecting malaria parasites in different populations. This diversity of dataset can prevent the development of a specific algorithm, especially in regards to population-specific variations in disease detection and presentation. Thus, having the diversity of data is essential to counteract this problem. Additionally, the exploration of several tools such as integration with smartphone or any other computer-based resource can be valuable in regions where geographic, political and economic issues arise, leading to a lack of resources that could help aid in detecting malaria early.In doing this, it will be valuable to assess the feasibility and cost-effectiveness of implementing computer-vision malaria detection in different healthcare settings, evaluating impact on patient outcomes and ensuring that patient privacy and confidentiality is met, whilst obtaining informed consent.

Addressing of these limitations and biases in future projects will lead to improved health outcomes for those who have no access to traditional diagnostic methods or in the case where traditional diagnostic methods are accurate only 88

REFERENCES

[1] Kapoor, R. (2020). Malaria detection using Deep Convolution Neural Network. OhioLINK ETD: Kapoor, Rishika. Retrieved April 12, 2023, from https://etd.ohiolink.edu/apexprod/rws_olink/r/1501/10?clear=10p10_accession_num=ucin1613749143868579

[2] Fuhad, K. M. F., Tuba, J. F., Sarker, M. R. A., Momen, S., Mohammed, N., Rahman, T. (2020, May 20). Deep learning based automatic malaria parasite detection from blood smear and its smartphone based application. MDPI. Retrieved April 12, 2023, from https://www.mdpi.com/2075-4418/10/5/329

[3] Tangpukdee, N., Duangdee, C., Wilairatana, P., Krudsood, S. (2009, June). Malaria diagnosis: A brief review. The Korean journal of parasitology. Retrieved April 12, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2688806/