EXECUTIVE SUMMARY:

Most of the people who die of malaria are from South America or South Asia where the resources are scarce and there is socio-economic instability (CDC). Visually identifying the parasitized cells using microscopy depends on the reliability of the laboratory performing the identification. It is a costly, tedious and time-consuming process. The accuracy of rapid diagnostic test results interpreted visually is 82.1% (Kalinga et. al, 2017). Whereas accuracy of our base CNN model is 98.07%. In a country like United States, there are about two thousand cases of malaria diagnosed and reported each year. Hence, an average laboratorian does not perform this test regularly, and may not be too proficient. In order to reduce the time, cost, error and requirement for human supervision, an automated classification technique (CNN) can be implemented. This has a high accuracy as well as recall value. This can help with the early and accurate detection of malaria and hence save lives in both low-resource, as well as high-resource but low-occurrence countries.

SECTION I: PROBLEM DEFINITION

The context:

Malaria is a devastating disease that is transmitted to human beings through infected female Anopheles mosquito bites that carry Plasmodium parasites. The malaria parasites, Plasmodium, enters the bloodstream when the mosquitoes bite a person, in form of sporozites. These sporozites can multiply quickly without causing any symptoms usually for days but stay alive in a human liver up to a year. The parasites then travel to the lungs via the heart and settle in the lung capillaries. That can cause severe respiratory diseases. Hence, an early treatment can avoid further complications or even death due to malaria. According to the Centers for Disease Control and Prevention (CDC), about 627,000 people died of malaria in 2020, most of which are in South America or South Asia where the resources are scarce and there is socio-economic instability. Most vulnerable to malaria are children under five years of age. Every twelfth child in the World that died in 2017, died because of malaria. Traditionally, malaria in diagnosed by experts in the laboratory, where they visually distinguish between infected and uninfected blood cells. It is a costly, tedious and time-consuming process. The diagnostic accuracy depends on human expertise can be adversely impacted by inter-observer variability. In order to reduce the time, cost, error and requirement for human supervision, an automated classification technique can be implemented. This can help with the early and accurate detection of malaria. One can use Machine Learning (ML) and Artificial Intelligence (AI) and obtain higher accuracy than manual classification. Hence, we propose using Deep Learning Algorithm, more specifically, Convolutional Neural Network, for malaria detection.

The objectives:

Build an efficient computer vision model to detect malaria. The model should identify whether the image of a red blood cell is that of one infected with malaria or not, and classify the same as parasitized or uninfected, respectively.

The key questions:

1. Can we develop a cost-effective, automated Deep Learning algorithm to identify the malaria-causing parasitized cells from the uninfected cells that do not cause malaria?
2. Can we do the above with a higher level of accuracy and ease than a manual detection of infected cells?
3. Will our Deep Learning algorithm be scalable for large data sets?

The problem formulation:

We will try to formulate a Deep learning algorithm that will be automated, time and cost effective, and scalable for large data sets. A Deep Learning network will be developed to learn and identify patterns in the parasitized cells and predict them with high accuracy.

SECTION II: DATA EXPLORATION

Data description:

There are a total of **24,958 train** and **2,600 test** images (colored) that we have taken from microscopic images. About 90% of the data is in the train set and 10% in the test set. These images are of the following categories:

- **Parasitized:** The parasitized cells contain the Plasmodium parasite which causes malaria
- **Uninfected:** The uninfected cells are free of the Plasmodium parasites

Preprocessing the data:

1. The images have been **reshaped** to a size of (64, 64, 3), The index 3 represents the RGB color.
2. There are 256 (2^8) colors that can be displayed in 8 bits (when 8 bits is required to display the image). Hence the range of pixel values are (0, 255). We **normalize** the images so that the pixel intensities have a normal distribution and ranges between (0, 1). This ensures that each pixel, which is the input parameter, has the same data distribution. This helps in faster convergence when training the neural network.
3. The **test** data is **balanced**, while the **train** data is **not balanced**. There is a slight discrepancy in the number of values in each level in the training set. A balanced data set is usually preferred while performing statistical tests because then the test with have larger statistical power, i.e., the likelihood of a significance test detecting an effect when there actually is one. The test statistics is unlikely to be affected by homoscedasticity (i.e., variance is likely to remain the same for balanced data).

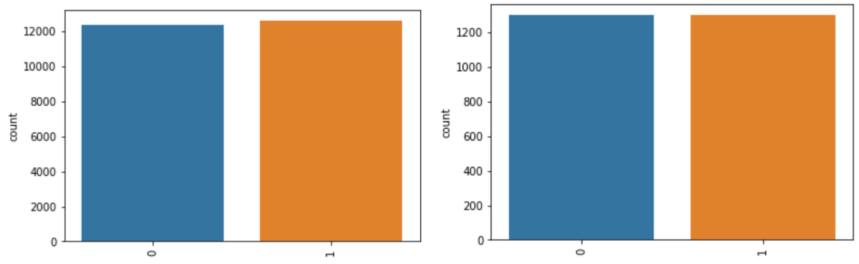


Figure1: *Count of uninfected (0) and parasitized (1) images in the train set (**Left figure**). Count of uninfected (0) and parasitized (1) images in the test set (**Right figure).***

Observations & Insights:

1. Key patterns in the data:

- We have visualized the above processed data for figure size (16, 16) and (12, 12) respectively.
- The parasitized cells have one or more **dark purple patches** on them, which are missing in the uninfected cells.
- These patches will be our **most important identifying features** to differentiate between parasitized and uninfected cells.

- Even with a smaller figure size (12, 12), the images capture the purple patches effectively.

2. **Why do we want the images to be blurred/smoothened?**
   - We want the input images to the CNN such that they are easier to process without losing the most important features that are required for accurately predicting the outcome. This is particularly important for large data sets.
   - Image smoothening takes away the noise and unnecessary details of an image and makes it easy to find the features we are most interested in.
   - Image processing like edge detection and thresholding work better if the image is first smoothened.
   - The parasitized cells show a dark purple patch, whereas the uninfected cells do not show any such darker patches.
   - We need to make sure that any image processing conserves and highlights these patches so that our algorithm can identify them easily, while removing noise and unnecessary details.
3. **We have performed three different smoothing techniques:**

(i) Mean image filtering: Mean images consider the pixels around a central pixel and calculate the average of these pixels, and then replaces the central pixel with the average. This averaging smoothens the data and removes the noise and the details. However, doing this takes away the identifying feature of the parasitized cells which is the dark purple patch. We see no difference between the uninfected and the infected cells if we use mean image filtering. Hence, this method of image smoothing is not useful for our purpose.

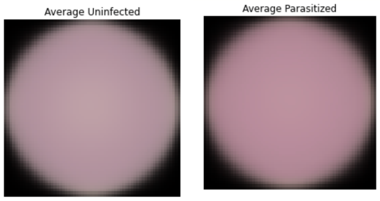


Figure2:** Mean image filtering for uninfected blood cell. (**Left figure**). Mean image filtering for parasitized blood cell. (**Right figure).

(ii) Median image filtering: We see a similar effect for median smoothing. We see no difference between the uninfected and the infected cells if we use median image filtering. Hence, we cannot use this method for this data as well.

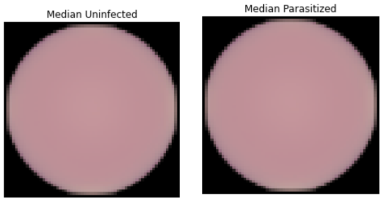


Figure3:** Median image filtering for uninfected blood cell. (**Left figure**). Medan image filtering for parasitized blood cell. (**Right figure).

(iii) Processing Images using Gaussian Blurring: Gaussian Smoothing uses the gaussian function. It blurs the image to reduce the details and noise from the image. It also produces a rotationally symmetric image. However, it doesn't remove the identifying dark patches in the original images of parasitized cells, unlike the mean and median image filtering. Hence, Gaussian blurring can be used to smoothen our data and reduce noise and unnecessary details in our images.

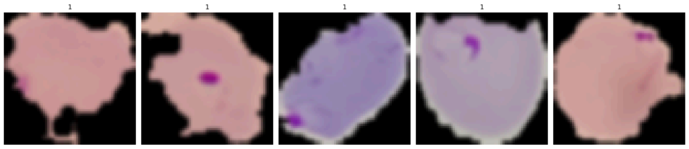


Figure4: Gaussian blurring with noise removed. The patches showing the parasitized blood cells are clearly visible in the filtered images.

4. Converting RGB to HSV of Images:

HSV color space is consists of 3 matrices: 'hue' (i.e., color), 'saturation' (i.e., amount to which that respective color is mixed with white) and 'value' (i.e., the amount to which that respective color is mixed with black. HSV is effective for color-based image segmentation which is required to identify the darker patches in our images.

On the other hand, RGB color space, the red, green and blue colors are correlated with the luminance/intensity of light. Hence it is more difficult to perform image segmentation in RGB color space. As we can see in the picture below, the features (the darker patches) are preserved in the HSV color images. We will convert our images to HSV color space.

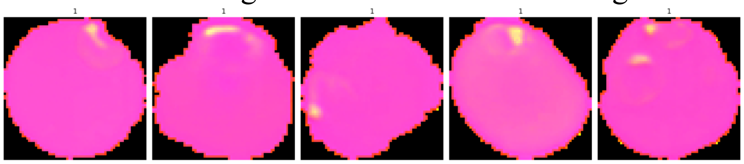


Figure5: Images converted to HSV. The patches showing the parasitized blood cells are clearly visible in the HSV images.

SECTION III: PROPOSED APPROACH

Potential techniques:

I will reshape and normalize the data, apply Gaussian smoothing and HSV color conversion.

I will then start with a basic Convolutional Neural Network (CNN) for solving this problem. I will then fine tune the hyperparameters to increase the model efficiency. The reason for choosing this method over others is as follow:

- Machine Learning methods require hand-engineered features which is time-consuming.
- CNN on the other hand, contains convolutions layers, which form the heart of the CNN algorithm. These layers Learn the patterns based on the previous input.
- Hence, the convolution layers learn the features within the algorithm itself and hence, feature-learning is automated, i.e., no manual intervention is required to identify the features. This is time effective, computationally effective, as well as cost effective.
- Pooling layers within CNN are used to downsize the sample and reduce the dimension of the input without losing the critical information that is required for a prediction with high accuracy.
- Artificial Neural Networks (ANN) do not contain the convolution and pooling layers. Hence, it cannot learn the features within the algorithm automatically. The features need to be explicitly provided as an input to the neural network. Hence it required human

supervision. Also, a 2-D image will be required to be converted to a 1-D vector for an ANN. This increases the number of trainable parameters and the process is not very time, cost or computationally effective any more.

- Recurrent Neural Networks (RNN) can handle images. However, RNNs are better at handling temporal or sequential information. Hence, they are more effective in time series problems, speech, audio and video related inputs. However, an image is a spatial data and not a temporal data. RNNs are less effective than CNN when it comes to resolving different features in a given image. CNN is also faster than RNN.

I would also like to implement Transfer learning. Neural Networks require a large number of labelled input data which are manually, as well as time and cost intensive. Hence, they are often difficult to obtain. Also, if a Neural Network is created for a specific task, it may be highly accurate for that set of tasks, but not so for other types of tasks. In Transfer learning, the knowledge gained from one task can be used to solve similar tasks. The knowledge gained can be the 'features' or 'weights'. Hence, the algorithm is no longer an isolated single task solver. Also, by Transfer learning when applicable, we can speed up the process, may require less training data, and also predict more accurately.

- I will use pre-trained models that performs Transfer knowledge: Very Deep Convolutional Networks/VGG-16 and/or VGG-19 (Visual Geometry Group).
- These models are used train computer vision/images and they contain millions of parameters/weights that have been obtained by training more than a million images.
- VGG-16 and VGG-19 have 16 and 19 learnable parameters layers respectively, that can be fine-tuned.

Overall solution design:

Step 1: Split the data into training set and test set. Keep the test data separate. Now split the training data into training and validation sets.
Step 2: Create a CNN model.
Step 3: Train the model on the training set and test it on the validation set and check for accuracy and minimized the loss.
Step 4: Fine-tune the hyperparameters and check again if it has improved accuracy and minimized the loss.
Step 5: Try VGG-16 and/or VGG-19 model and check if accuracy has improved and the loss minimized.
Step 6: Fine-tune VGG-16 and/or VGG-19 model with image augmentation and if it has improved accuracy and minimized the loss.

Measures of success:

In order to compare the models, we will evaluate accuracy, recall, precision and F1 scores for each model and compare their values. If we are interested in the model accuracy, then the accuracy and f1 scores will be of particular interest in deciding the best model for this task.

SECTION IV: REFINED INSIGHTS

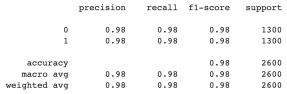
The images have been reshaped to a size of (64, 64, 3), The index 3 represents the RGB color. There are 256 (2^8) colors that can be displayed in 8 bits (when 8 bits is required to display the image). Hence the range of pixel values are (0, 255). We normalize the images so that the pixel intensities have a normal distribution and ranges between (0, 1). This ensures that each pixel, which is the input parameter, has the same data distribution. This helps in faster convergence when training the neural network. The test data is balanced, while the train data is not balanced. There is a slight discrepancy in the number of values in each level in the training set. A balanced data set is usually preferred while performing statistical tests because then the test with have larger statistical power, i.e., the likelihood of a significance test detecting an effect when there actually is one. The test statistics is unlikely to be affected by homoscedasticity (i.e., variance is likely to remain the same for balanced data).

We will build several models that will train the train image data and identify these patches in the infected blood cells caused by the parasites. We will consider metrics like accuracy, as well as the time required to execute each epoch. If the accuracies are comparable between two or more models, then the model that requires least time to execute each epoch will be the best model for this task. A lower execution time means less memory and time, and hence more cost effective.

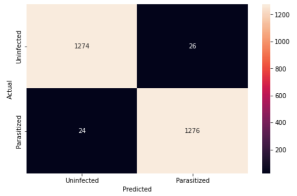
SECTION V: Comparison of various techniques and their relative performance

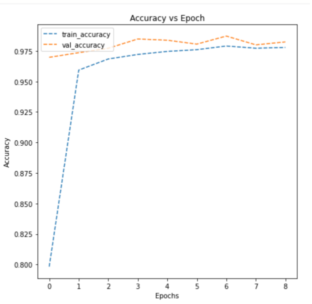
Base Model: (Test accuracy = 98.07%)

- *Description of the CNN model:* The base model has 3 convolution layers, 3 maxpooling layers, 4 dropout layers. It uses ReLU as the activation function. The final layer has an activation function of softmax which helps in classification problems. The epoch is 20. Callbacks monitors loss function, val_loss in this case. We want to minimize the loss function. The 'mode = min' by default. 'Patience' delays the triggering of the callback. The count of patience is the number of epochs on which we would like to see no improvement. We have set the patience value at 2.
- *Results:*



```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1300
           1       0.98      0.98      0.98      1300

    accuracy                           0.98      2600
   macro avg       0.98      0.98      0.98      2600
weighted avg       0.98      0.98      0.98      2600
```





  - The model has a **98% recall, 98% precision and 98% F1-score**, which is a good performance. We already see a very high accuracy for the test dataset without any image augmentation and smoothing (**98.07%**).
  - The **model does not overfit the train data set**. We can conclude this from comparable accuracy in the training set (blue dotted line) and the validation set (orange dotted line). Here we can clearly observe that the training and validation accuracy are increasing. And we can also see that validation accuracy is slightly higher than the train accuracy. The loss function for the validation set decreases overall, which means that the error in the predicted outcome has decreased overall, although some minor fluctuations are observed.
  - The **accuracy** for both the train and validation data sets **increases sharply** with increase in epoch till about 2 to 3 epochs. However, the accuracy begins to stabilize after approximately 2 to 3 epochs. The accuracy of the validation **can be improved by further hyperparameter tuning and image processing**. From the confusion matrix, we

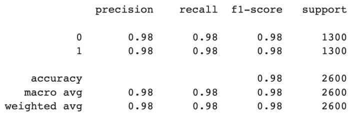
can see that most of the actual Vs predicted outcomes are along the diagonal, i.e., **most of the observations are correctly identified** by the model.

- In 20 test images out of 2600 test images, uninfected cells have been incorrectly identified as parasitized and in 23 test images, parasitized cells have been incorrectly identified as uninfected.

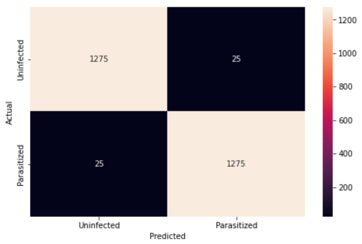
Model 1: Adding extra convolution layer, maxpooling layer, increasing epoch

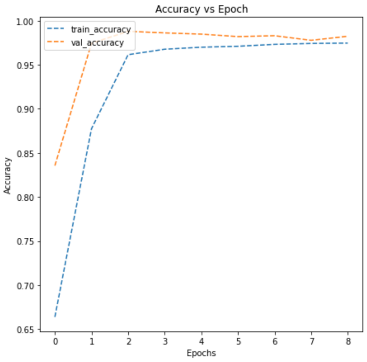
(Test accuracy = 98.11%)

- *Description of the CNN model:* One convolution layer has been added to the previous model. An extra maxpooling layer has been added. Increasing maxpooling layers progressively reduces the spatial size of the input image. This reduces the number of computations in the network. It sends only the important data to next layers in CNN. We see that our computational time has decreased after adding two maxpooling layers, although we have increased the number of hidden layers and the number of epochs. This apparently tiny difference in the time can manifest itself in a significant manner if we have a large dataset to process. An extra dropout layer has been added. Adding dropout layers prevent overfitting on the training data. These layers mask the contribution of some features towards the next layer and leaves the rest of the features unmodified. This ensures that the features that are learnt later in the samples are not overridden by the features that are learnt earlier in the training of the batches of training data. Increasing maxpooling layers progressively reduces the spatial size of the input image. This reduces the number of computations in the network. It sends only the important data to next layers in CNN. We see that our computational time has decreased after adding two maxpooling layers, although we have increased the number of hidden layers and the number of epochs. This apparently tiny difference in the time can manifest itself in a significant manner if we have a large dataset to process. We have also increased the number of epochs from 20 to 30.
- *Results:* The model has a **98% recall, 98% precision and 98% F1-score**, which is a good performance. However, **there is no improvement in the accuracy even after increasing the layers, number of epoch and patience. Hence there is no benefit in increasing the convolution layers and epochs.** The loss function for the validation set decreases and the accuracy for both the train and validation data sets increase sharply with increase in epoch.
- From the confusion matrix, we can see that most of the actual Vs predicted outcomes are along the diagonal, i.e., most of the observations are correctly identified by the model.



```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1300
           1       0.98      0.98      0.98      1300

    accuracy                           0.98      2600
   macro avg       0.98      0.98      0.98      2600
weighted avg       0.98      0.98      0.98      2600
```



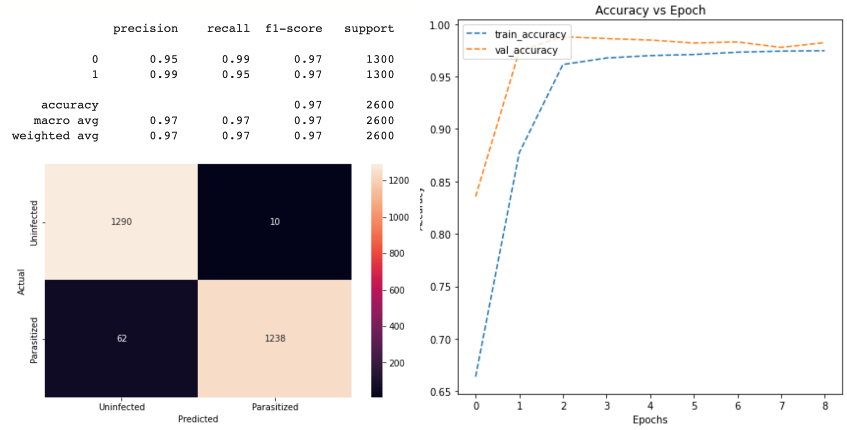


Model 2: Adding batch normalization and leaky ReLU

(Test accuracy = 97.23%.)

Description of the CNN model: We have used the same layers as the base model, since increasing the layers and epoch in model 1 didn't yield a better model accuracy. However, we have **added batch normalization** and changed the activation function to **leakyReLU.** Batch normalization normalizes the contributions to a layer for every mini-batch. This reduces the number of training epochs required to train deep neural networks. We replaced ReLU activation function by leakyReLU activation function because it fixes the dying ReLU problem (since it has a non-zero slope below zero). It also helps train the neural network faster.

Results:



```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1300
           1       0.99      0.95      0.97      1300

    accuracy                           0.97      2600
   macro avg       0.97      0.97      0.97      2600
weighted avg       0.97      0.97      0.97      2600
```

-The test accuracy is still 97 %. Hence **no improvement in accuracy even after adding batch normalization and using leakyReLU as activation function.**

- Also, the accuracy converges withing the first few epochs. We can **reduce the number of epochs to 10 to save computational time and memory.**

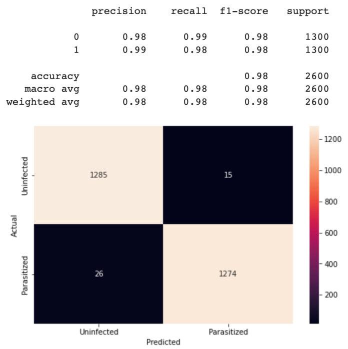
- The precision of parasitized cells is 99%, whereas that of the uninfected cell is 95%. The percentages flip for 'recall'. This means that the parasitized cells have a very low chances of being misidentified as uninfected. However, an uninfected

Model 3: Can we improve the model with Image Data Augmentation? **(Test accuracy = 98.42%)**

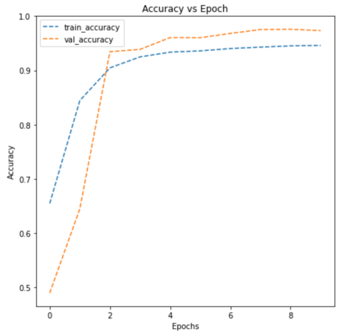
Description: Data augmentation synthesizes new data from existing data.

Results: We created three sets of augmented images: train_datagen1(zoom_range = 0.5, rotation_range = 30°), train_datagen2 (zoom_range = 0.75, rotation_range = 60°) and train_datagen3 (zoom_range = 0.85, rotation_range = 45°). We ran the model2 for each of them. The test accuracy for the three models is 98.42%

The augmented images resulted in an accuracy for the test set images that is slightly better but comparable to the previous models. However, the time required to execute each epoch is between 4-6 times that for the non-augmented data sets.



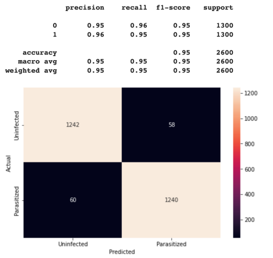
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      1300
           1       0.99      0.98      0.98      1300

    accuracy                           0.98      2600
   macro avg       0.98      0.98      0.98      2600
weighted avg       0.98      0.98      0.98      2600
```



Model 4: Adding pre-trained model (VGG16) (Test accuracy = 96.69%)

Description: One can use a pre-trained model that has been created for a similar task instead of creating a model from scratch. This is known as transfer learning. One advantage of transfer learning is that we can build on it and introduce additional layers according to our requirements for the task at hand. We have used VGG16 for our task. We have used 10 epochs for this model since the accuracy converges pretty quickly within a few epochs. Patience for callback is 5. We flattened the output from the 3rd block of the VGG16 model. Added two dense layers with 128 and 64 neurons. The final dense layer has two neurons representing the two classes (uninfected and parasitized). There is also a dropout layer and a batch-normalization layer added to the VGG-16 model. We have used ReLU activation function in all additional layers except the final dense layer, which uses softmax because this is a classification task.

Results: The VGG-16 pre-trained model didn't perform as well as the other models for this task.

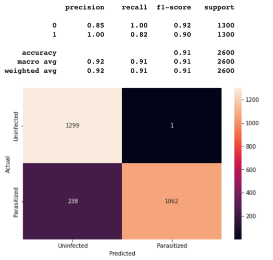


```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      1300
           1       0.96      0.95      0.95      1300

    accuracy                           0.95      2600
   macro avg       0.95      0.95      0.95      2600
weighted avg       0.95      0.95      0.95      2600
```

Model 5: Adding pre-trained model (VGG19) (Test accuracy = 90.8%)

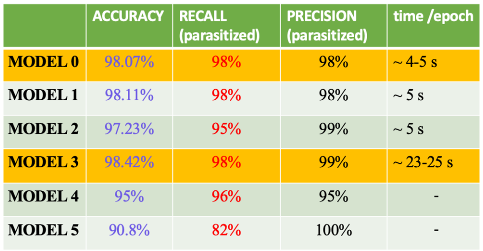
Description: Used **pre-trained model with 19 convolution layer** that has been already created for a similar task. Additional layers, activation function and number of epochs is the same as Model 4.

Results: The VGG-19 pre-trained model also didn't perform as well as the other models for this task.



```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      1300
           1       1.00      0.82      0.90      1300

    accuracy                           0.91      2600
   macro avg       0.92      0.91      0.91      2600
weighted avg       0.92      0.91      0.91      2600
```

Model comparison:



	ACCURACY	RECALL (parasitized)	PRECISION (parasitized)	time /epoch
MODEL 0	98.07%	98%	98%	~ 4-5 s
MODEL 1	98.11%	98%	98%	~ 5 s
MODEL 2	97.23%	95%	99%	~ 5 s
MODEL 3	98.42%	98%	99%	~ 23-25 s
MODEL 4	95%	96%	95%	-
MODEL 5	90.8%	82%	100%	-

SECTION VI: *Proposal for the final solution design*

The base model has a fairly good accuracy for test image classification. However, in order to improve the accuracy, we have applied several different techniques like increasing layers, changing activation functions, introducing batch-normalization, applying image augmentation and transfer learning method using VGG-16 and VGG-19. For a complex task, we need more learnable parameters. And for more learnable parameters we need more data. In order to deal with limited data in real life tasks, we use data augmentation. Data augmentation synthesizes new data from existing data. The best model in terms of accuracy in prediction of the test images is the one with data augmentation (Model3, Model3_2, Model3_3). All the three models have image augmentation applied by using different hyperparameters. These models have an accuracy of 98.42% for the test images. However, one flip side to this model is that the run time is almost 4-6 times longer than the other models. Also, it is important to note that the probability that a sick patient is detected by the classifier is determined by 'recall'. Note that Recall minimized false negatives for parasitized cells because we want to identify correctly as many infected cells as possible. False positives are better than false negatives in this case because the patient can undergo further diagnosis to figure out if they are infected. Although the recall for VGG-19 is excellent for uninfected people, it is not so for parasitized people. The accuracy of the model with augmented images and the base model are comparable (within the error limit). But the time required to run the model with augmented images is 4-6 times longer than that required for the base model. Hence, I propose using the base model for the classification of infected and parasitized malaria blood cells. I also propose using an epoch of about 10, because the accuracy converges pretty quickly within the first few epochs.

SECTION VII: RISKS AND SOLUTIONS

Risk1: Major processing lag in training the models when run on CPUs. For training large number of parameters over increased epochs, the code ran very slowly and/or crashed.
Solution: Using GPU (Graphics Processing Unit) one can increase the performance of the code. It is a specialized processor with dedicated memory for rendering graphics. I bought access to premium GPU from Google in order to run the codes successfully on Google Colab.

Risk2: Callbacks are used for early stopping of the training process when the validation loss doesn't improve any longer. However, sometimes this value can get stuck in a local minima.
Solution: We can use the 'ReduceLROnPlateau' to change the learning rate if the validation loss doesn't improve after the number of epochs as specified by 'patience.

Risk3: Transfer learning, although easier to use, do not yield good results for this particular task, even with added layers.

Solution: Use CNN models built from scratch and tune the hyperparameters to improve the performance.

SECTION VIII: EXECUTIVE BUSINESS SOLUTION

Convolutional Neural Network (CNN) must be used instead of visually inspecting the blood cells of the patients for malaria parasites in order to increase

- accuracy,
- cost-effectiveness (by eliminating the need of skilled personnel required for visual inspection),
- ease,
- scalability,
- and for early detection of malaria and saving lives.