**MALARIA CELL DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**



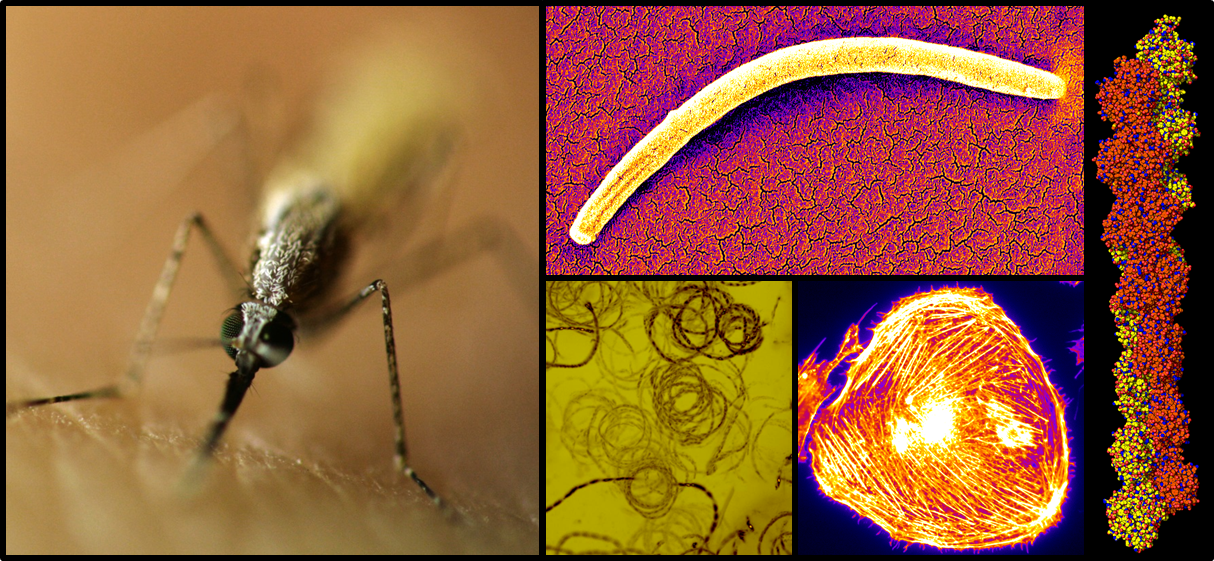
**Submitted to- *iQGateway*  Presented by- Hamzah Jamal**

**Table of Contents:**

1. **Introduction**
2. **Convolution Neural Network**
3. **Methodology**

* **Data acquisition and Pre-Processing**
* **CNN Classifier Architecture**
* **Training CNN Classifier**
* **Evaluation of CNN Architecture**

1. **Create a Streamlit Web App**
2. **Deploying Web App with Heroku**

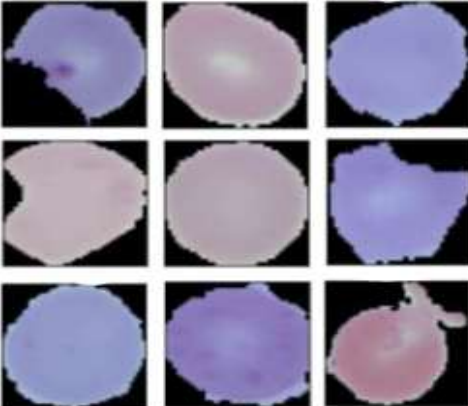


**Introduction**

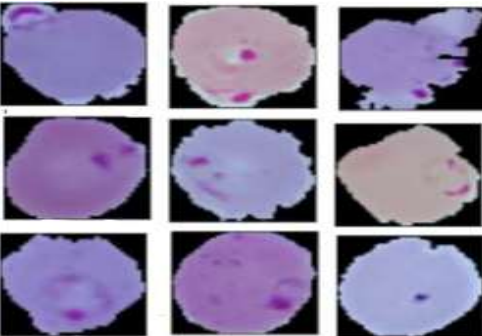
All over the world malaria is a prevalent disease that has sued many lives. These diseases can affect humans or other species. The disease is usually transmitted by mosquitoes of female anopheles. The mosquito bite injects the parasite into the blood of the infected individual, which then goes to the liver for maturation and reproduction.

The diagnosis of malaria is done by manual method to examine the microscopic images of red blood slide. In order to have trustworthy diagnosis of the disease, many years of experience and training are required to examine the slide to give decision. Inappropriately, such experience pathologist and resources are very less in countryside areas where malaria has a striking prevalence. The problems related to manual diagnosis results into the state of art for automation detection of the malaria from the digitized microscopic image of red blood cell. The automation in detection process will make sure precise detection of the disease. Change in computer-aided systems uses profound learning algorithms to interpret medical images.

Automatic diagnosis of malaria implicates understanding of conventional microscopy to computerized system.Recently, deep learning models have greatly exceeded performance of human beings in identification of complex images. The latest trend going on in artificial intelligence is deep learning that has increased the performance in several medical areas. It is multilayer neural network classifier which is trained by back propagation and many added layers are also used for the classification and detection process of medical imaging.



*Figure 1: Samples Drawn from Kaggle Dataset which are Uninfected Red Blood Cells*

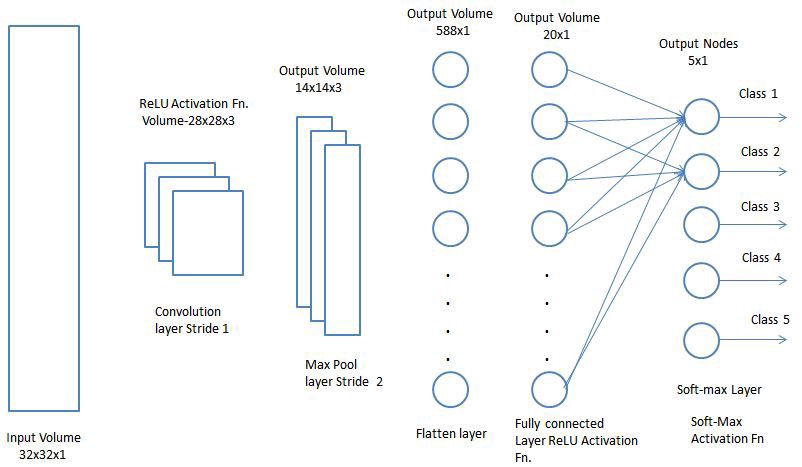


*Figure 2: Samples Taken from Kaggle Dataset which Red Blood Cells are Infected with Malaria*

**Convolution Neural Network**

A Convolution Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNN have the ability to learn these filters/characteristics.

The architecture of a CNN is analogous to that of the connectivity pattern of neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area. Convolution layer, pooling layer, and fully connection layer are the three main types of layers used to construct the CNN architecture.



*Basic Architecture of CNN*

Compared to traditional neural networks, CNNs can extract features without losing much spatial correlations of the input. Each layer consists of neurons that have learnable weights and biases. The optimal model is achieved after feeding data into the network and minimizing the loss function at the top layer.

**Methodology**

The proposed technique practices deep learning methodology which takes features from pixels and classify straight from the segmented patched of the red blood cell. The dataset used for this study is taken from Kaggle, online database. The evaluation metric accuracy is used to compute the best performing architecture. The frame work of this project is divided into different pipelines which are:

* Data acquisition and Pre-Processing
* CNN Classifier Architecture
* Training CNN classifier
* Evaluation of CNN Architecture

**Data acquisition and Pre-Processing:**

The dataset is taken from kaggle which contains infected and uninfected cell images for Detecting Malaria. There are total of 27,558 images. We have used 26000 each for different class for training the model and the rest 1000 for the testing. We have performed techniques which are as following:

1. ***Data loading and visualization***

First step of any problem is to load the data, we have carefully loaded the data of each class. Once the data is loaded we visualize the images of the infected and uninfected class

1. ***Re-scaling (Min-Max Normalization)***

Re-scaling has been done on the image patches for mapping the feature range between zero and one. This is done to obtain a faster convergence.

1. ***Data Augmentation***

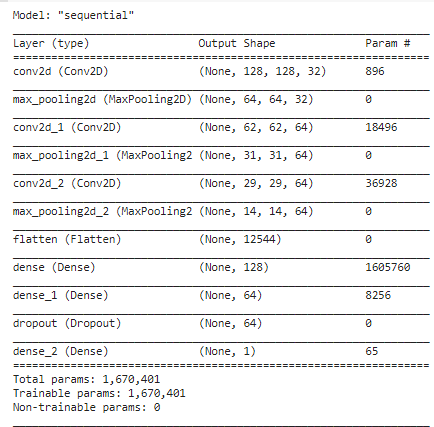
Automatically generated a flow of batches from a directory (and also do some data augmentation in images) with the help of Keras <\_flow\_fromdirectory> and Image Data Generator. As manipulation, we are going to rotate, resize and scale the images, so the model becomes more robust to different image settings.

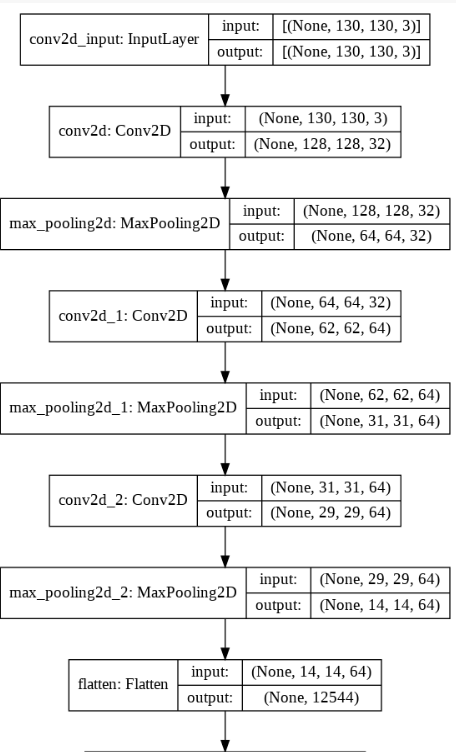
Also, splited our data in train and validation data.

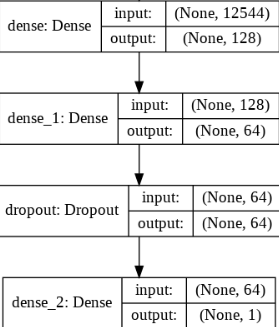
**CNN Classifier Architecture:**

A Convolution Neural Network (CNN) take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

The detail information of the architecture used to build the CNN classifier is-





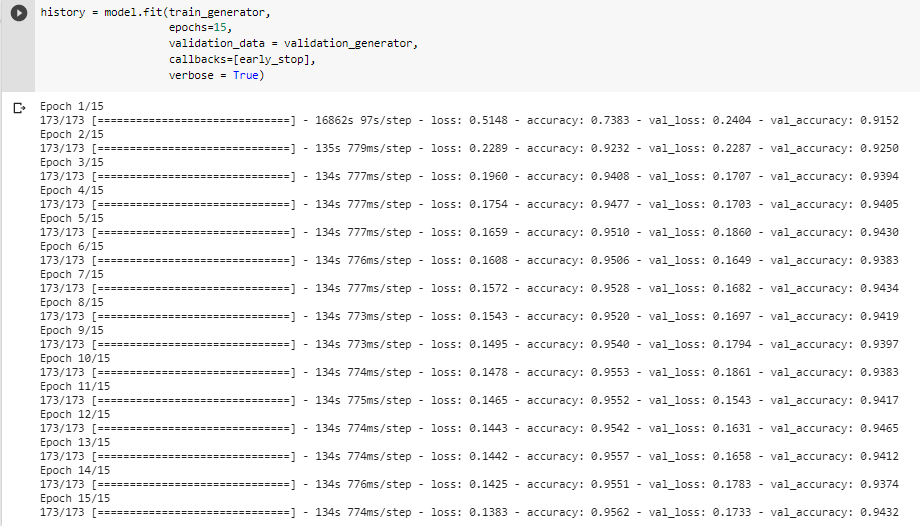
**

*The flowchart of the architecture of the classifier*

**Training CNN Classifier:**

Once the CNN architecture is defined, our next step is to feed the data into the model. In this stage, we feed our data to the model by fitting the model on the dataset.

NOTE:- We are not using the original image from the dataset to feed in the model, but instead we are using augmented image which we have generated in the previous stage.



Data Augmentation is the process of augmenting the images by techniques which included vertical and horizontal flips and shifting Gaussian blur,etc. In this stage we also use early stopping which helps the model to stop training when we got the least validation loss i.e. after a certain number of epochs when our model has been trained early stopping will stop further training once we have the least validation loss.

**Evaluation:**

When we have trained the CNN classifier by feeding the augmented data, now our goal is see how well the model performs. First, we start by visualizing the training and validation accuracy as well as training and validation loss. Following are the graphs of the accuracy and score.

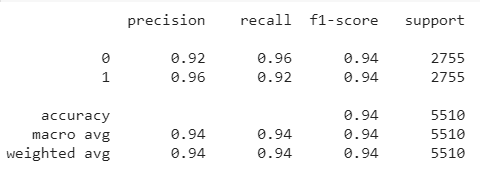


We can see after every epoch the loss is decreasing whereas the accuracy is increasing.

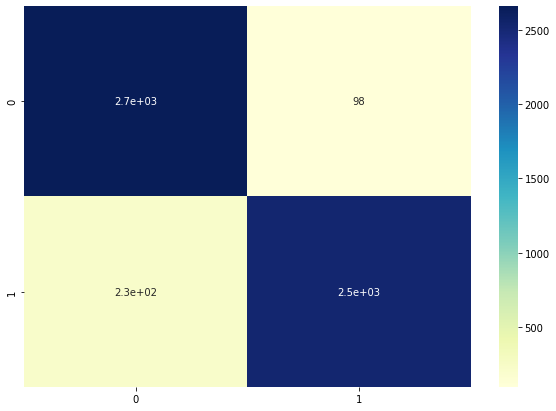
Next step is to evaluate the model on validation dataset which we have augmented. Once, we evaluate the model by feeding validation data the model will give us the values of probabilities of each class.

After setting the threshold value to 0.5, we are all set to check the performance of the model through different metrics.

* **Classification Report:** The classification report of the model is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below.



* **Confusion Matrix:** A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.



**Create a Streamlit Web App**

Creating a web app is one of the solutions such that other people can make use of our machine learning model. We can use Streamlit library to create a simple machine learning web app in your local machine. To deploy the web app to be accessible to other people, then we can use Heroku or other cloud platforms.

To create a web app with Streamlit, first thing that we need to do is creating a new Python file, In the Python file, first we need to load the trained model that we have saved before.

The next step is to write a header and any other texts we want to put into your web app. To let the user upload their own image to your web app, simply used file\_uploader attribute from Streamlit library.

The next important step is to process the image the user has uploaded. The processing step including resizing the image to the same size as training and validation images. After resizing the image, then the loaded model should predict in which category this image belongs.

After that, you need to save the Python file in the same directory as your previous Python file. Once, our files are arranged mentined as above our web app is ready to launch.

**Deploy Your Web App with Heroku**

So far, we’ve built the web app locally on our computer. In order for other people to be able to use your web app, we can utilize Heroku. Heroku is a cloud platform which helps us to deploy any web app. Following are the steps to deploy an app:-

**Additional Files for Deployment**

Before we deploy the web app, we need to create three additional files in addition to Python files that we have created to build the app. These three files are:

* **requirements.txt**: This is the text file that we need to create to tell Heroku to install the necessary Python packages needed to deploy our machine learning model. Hence, we need to specify the relevant version of those libraries in this text file.
* **setup.sh:**  This file is necessary to handle the server and port number of our app on Heroku.
* **Procfile:** this is the file of your configuration to tell Heroku how and which files to be executed.

Now our three files are ready, place them in the same directory as your Python files.

**Create Heroku Account and Deploy the Model with Heroku Git**

Our next step is to create a Heroku account. We can do this simply by signing off to the account.

After the account is created, the next thing is to install Heroku CLI (command-line interface) and then using it we can direct you to login with Heroku account. Next, move the directory of your app files with the command prompt and need to initiate an empty git repository.

Next step is to submit all of the app files to the empty repository that was created.

Now the app files will be deployed to Heroku with Heroku Git and after the process is completed, we will see the URL of your web app that can be accessed via the internet.

Below is the example of a simple web app that has been deployed with Heroku.

