Machine Learning Engineer Nanodegree

Capstone Project Proposal
Classifying Malaria Cell Images Data set
Puvvula spandana
February 2st, 2019

Proposal:-

Classifying Malaria Cell Images Data set

I. Definition

Project Overview:-

History:-

The mosquito and the fly in this Baltic amber necklace are between 40 and 60 million years old.
The first evidence of malaria parasites was found in mosquitoes preserved in amber from the Palaeogene period that are approximately 30 million years old. Human malaria likely originated in Africa and coevolved with its hosts, mosquitoes and non-human primates. Malaria protozoa are diversified into primate, rodent, bird, and reptile host lineages.[9][10] Humans may have originally caught Plasmodium falciparum from gorillas.[11] P. vivax, another malarial Plasmodium species among the six that infect humans, also likely originated in African gorillas and chimpanzees. Another malarial species recently discovered to be transmissible to humans, P. knowlesi, originated in Asian macaque monkeys. While P. malariae is highly host specific to humans, there is some evidence that low level non-symptomatic infection persists among wild chimpanzees.

About 10,000 years ago, malaria started having a major impact on human survival, coinciding with the start of agriculture in the Neolithic revolution. Consequences included natural selection for sickle-cell disease, thalassaemias, glucose-6-phosphate dehydrogenase deficiency, Southeast Asian ovalocytosis, elliptocytosis and loss of the Gerbich antigen (glycophorin C) and the Duffy antigen on the erythrocytes, because such blood disorders confer a selective advantage against malaria infection (balancing selection). The three major types of inherited genetic resistance (sickle-cell disease, thalassaemias, and glucose-6-phosphate dehydrogenase deficiency) were present in the Mediterranean world by the time of the Roman Empire, about 2000 years ago.
Reference link:- https://en.wikipedia.org/wiki/History_of_malaria

Economic impact:-

Malaria is not just a disease commonly associated with poverty: some evidence suggests that it is also a cause of poverty and a major hindrance to economic development. Although tropical regions are most affected, malaria's furthest influence reaches into some temperate zones that have extreme seasonal changes. The disease has been associated with major negative economic effects on regions where it is widespread. During the late 19th and early 20th centuries, it was a major factor in the slow economic development of the American southern states.

A comparison of average per capita GDP in 1995, adjusted for parity of purchasing power, between countries with malaria and countries without malaria gives a fivefold difference ($1,526 USD versus $8,268 USD). In the period 1965 to 1990, countries where malaria was common had an average per capita GDP that increased only 0.4% per year, compared to 2.4% per year in other countries.
Reference:- https://en.wikipedia.org/wiki/Purchasing_power_parity

Eradication efforts:-

Several notable attempts are being made to eliminate the parasite from sections of the world, or to eradicate it worldwide. In 2006, the organization Malaria No More set a public goal of eliminating malaria from Africa by 2015, and the organization claimed they planned to dissolve if that goal was accomplished. As of 2018 they are still functioning.[183] Several malaria vaccines are in clinical trials, which are intended to provide protection for children in endemic areas and reduce the speed of transmission of the disease. As of 2012, The Global Fund to Fight AIDS, Tuberculosis and Malaria has distributed 230 million insecticide-treated nets intended to stop mosquito-borne transmission of malaria. The U.S.-based Clinton Foundation has worked to manage demand and stabilize prices in the artemisinin market. Other efforts, such as the Malaria Atlas Project, focus on analysing climate and weather information required to accurately predict the spread of malaria based on the availability of habitat of malaria-carrying parasites. The Malaria Policy Advisory Committee (MPAC) of the World Health Organization (WHO) was formed in 2012, "to provide strategic advice and technical input to WHO on all aspects of malaria control and elimination".[186] In November 2013, WHO and the malaria vaccine funders group set a goal to develop vaccines designed to interrupt malaria transmission with the long-term goal of malaria eradication.
Reference link:- https://en.wikipedia.org/wiki/Malaria_Policy_Advisory_Committee

Vaccine:-

A vaccine against malaria called RTS,S, was approved by European regulators in 2015. It is undergoing pilot trials in select countries in 2016.Immunity (or, more accurately, tolerance) to P. falciparum malaria does occur naturally, but only in response to years of repeated infection. An individual can be protected from a P. falciparum infection if they receive about a thousand bites from mosquitoes that carry a version of the parasite rendered non-infective by a dose of X-ray irradiation. The highly polymorphic nature of many P. falciparum proteins results in significant challenges to vaccine design. Vaccine candidates that target antigens on gametes, zygotes, or ookinetes in the mosquito midgut aim to block the transmission of malaria. These transmission-blocking vaccines induce antibodies in the human blood; when a mosquito takes a blood meal from a protected individual, these antibodies prevent the parasite from completing its development in the mosquito. Other vaccine candidates, targeting the blood-stage of the parasite's life cycle, have been inadequate on their own. For example, SPf66 was tested extensively in areas where the disease is common in the 1990s, but trials showed it to be insufficiently effective.
Reference link:- https://en.wikipedia.org/wiki/Malaria_vaccine

Applications:-

- This project Classifying Malaria Cell Images Data set can be applied in any medical hospital to check the person is infected by malaria or not.

- This can be further extend to develop the AI devices that can check the wither the person is infected or not with out any human involvement.

personal motivation:-

- I choose this project because I am really inserted in learning **Convolutional Neural Networks .** I know project some what challenging but still I am taking this project to learn Convolutional neural networks.
- I think in this project I can learn how to use the keras_pretrained models like to load them.
- In this project I can learn how to handle the image data and pre-processes datasets for the training the data.
- In this project I can also learn ,how to use of kaggle kernals.
- In this project I can also learn,how to build a Convolutional neural networks using keras.

Inspiration:-

Save humans by detecting and deploying Image Cells that contain Malaria or not!

Problem Statement:-

The aim of this project is to predict the person is infected by malaria or not by visualizing the cell image of person. In the project, I am going to use various Convolutional Neural Networks to predict the types of images and compare their performance and finally declare my final model.

Evaluation Metrics:-

I want to use accuracy as evaluation metric for cell_type classification. Accuracy is a common metric for categorical classifiers

Accuracy = (images correctly classified) / (all images)

During development, a validation set was used to evaluate the model. I want to use a small set of training images as my validation images.

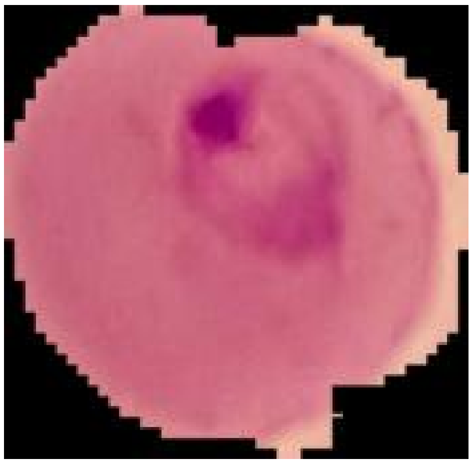
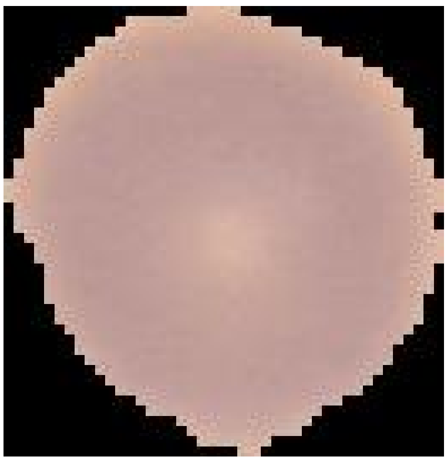
For validation I want to use "categorical_crossentropy" as loss metric for CNN, optimizer as "adam" and also metrics as "accuracy

II. Analysis

Data Exploration:-

Classifying Malaria Cell Images Data set has 27,558

Total 2 categories of images are present in Malaria Cell Images Data set

Parasitized

uninfected

The dataset that I am working is downloaded from

https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria/home This dataset contains 27,558 images of cells

content :-

The pictures are divided into two classes: Infected - Uninfected. For each class there are about 13700 photos. Photos are not high resolution. Photos are not reduced to a single size, they have different proportions

Citation:-

This Dataset is taken from the official NIH Website.anybody trying to start working with this dataset can get started immediately, as to download the dataset from NIH website is quite slow.

https://ceb.nlm.nih.gov/repositories/malaria-datasets/

https://unsplash.com/@ekamelev

The data is open – sourced and can be download for education purpose with no citation..

Exploratory Visualization:-

There are 2 total image categories.

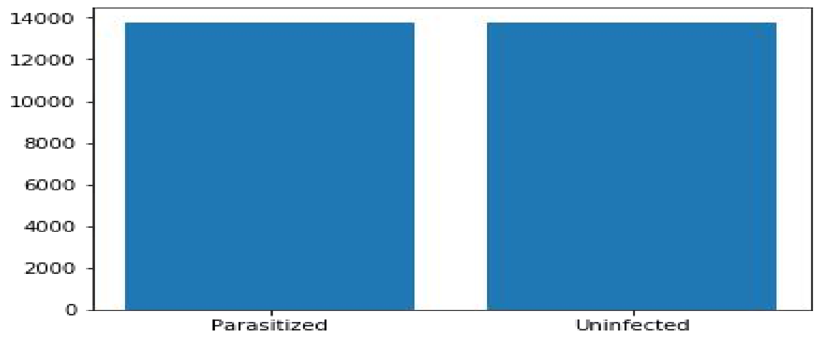
There are 27558 total cell images.

There are 22321 training cell images.

There are 2481 validation cell images.

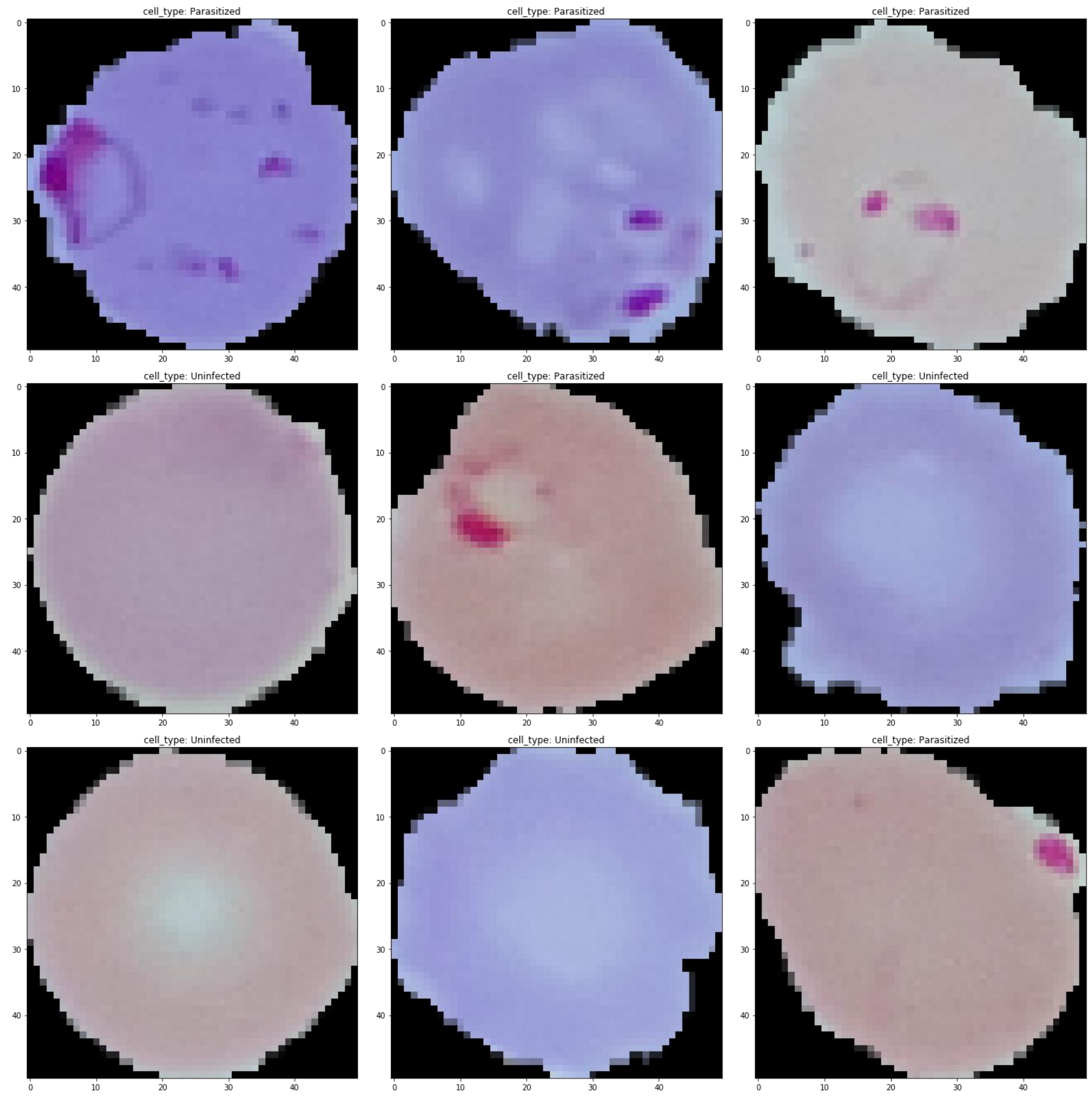
There are 2756 testing cell images.

```
Total dataset image categories.
```



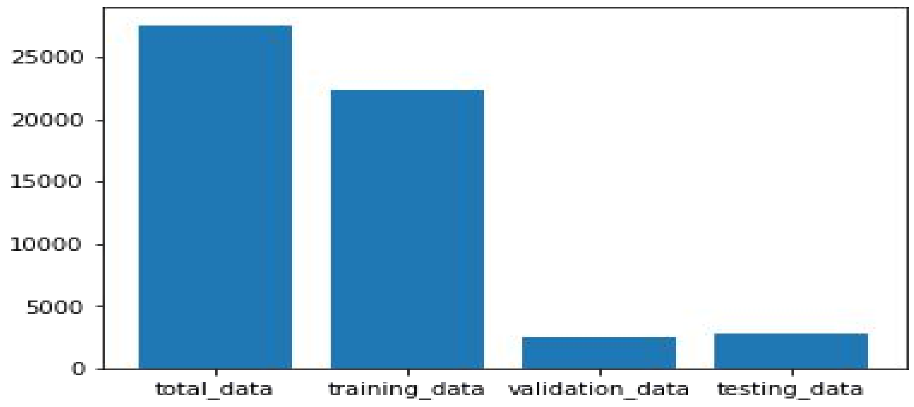
There are total 22321 training dataset

Sample images form the total dataset is



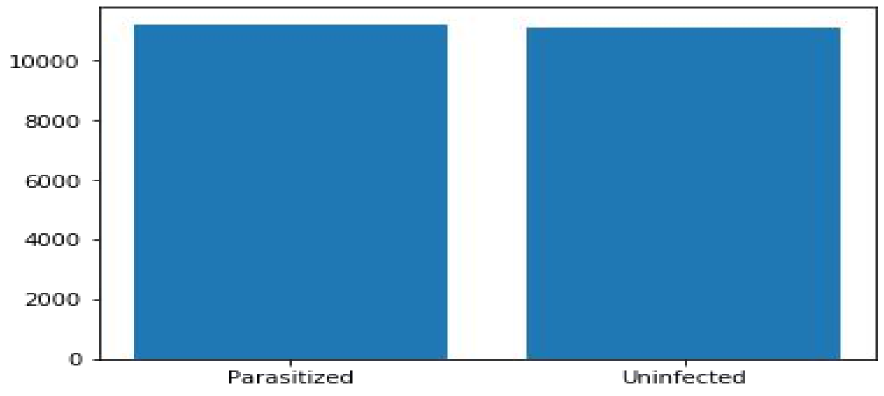
We see that cell image form the dataset the cell with parasitized has some dots in cell representing the infection and cell that is not infected as clear. Form this visualization conclude that we have find the cnn model that finds the dots in the cell representing the infection if the infection is to high the cell color will changes into pink.so our cnn model should find that kind of images to.this visualization provide me a good insight about the parasitized and uninfected cells.one of the challenges may be faced by the cnn model is learning above two future at a time.

Total data is divided into train valid test data.



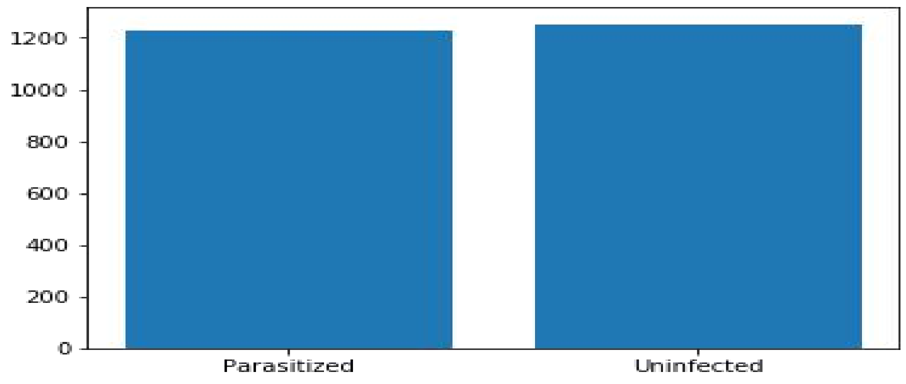
There are 27558 total images ,22321 training images,2481 validation images and 2756 testing images.

Training categories:-



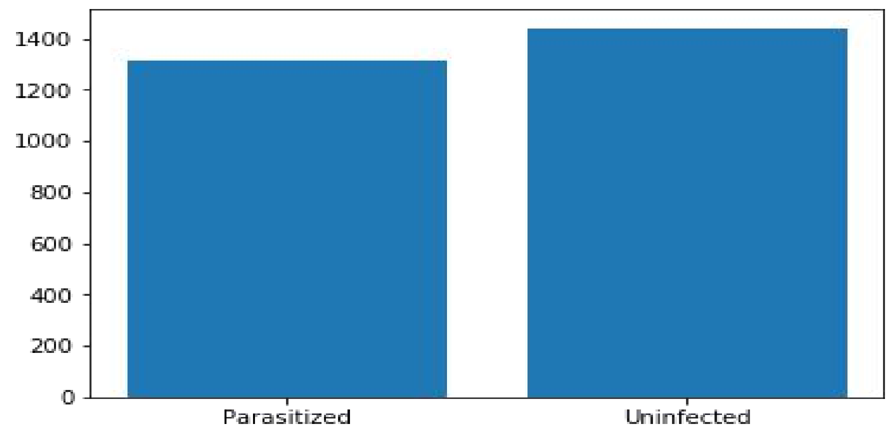
There are around 13779 images per each category.

Validation categories:-



There are around 1240 images per each category.

Testing categories:-



There are around 1378 images per each category.

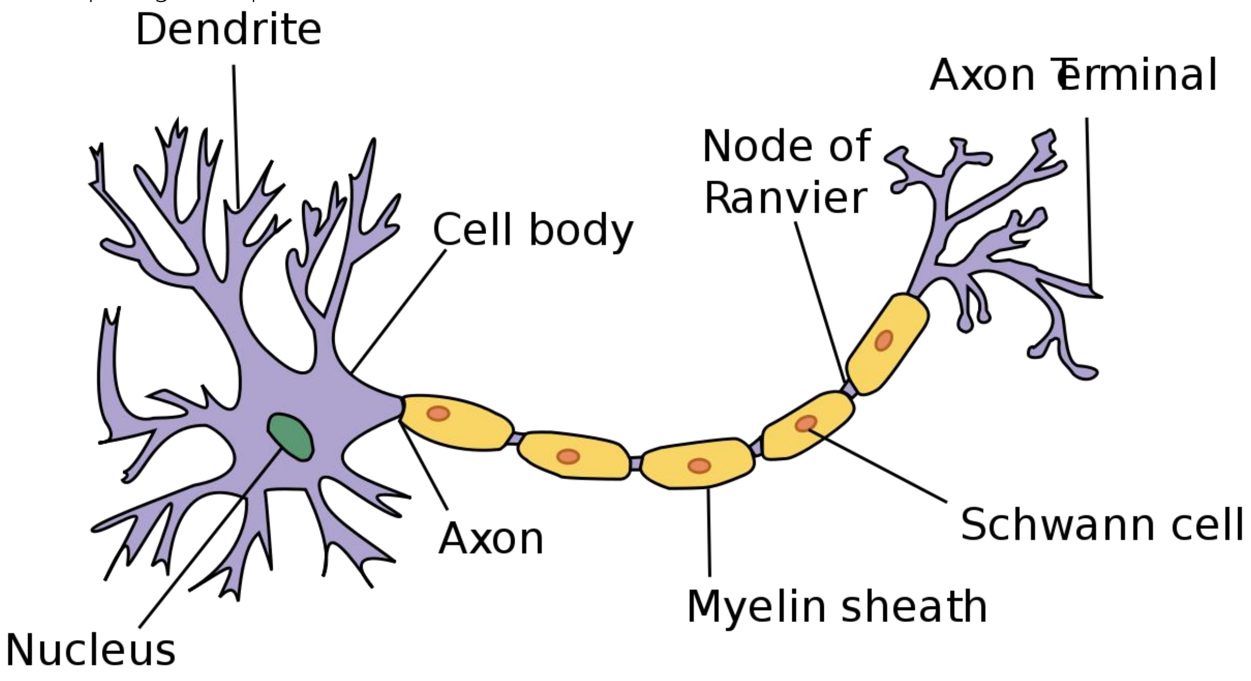
Algorithms and Techniques:-

Theory behind Scenes:-

Artificial neural networks (ANNs): ANNs are computing systems vaguely inspired by the biological neural networks that constitute animal brains and humans. these systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

A typical brain contains something like 100 billion miniscule cells called **neurons** (no-one knows exactly how many there are and estimates go from about 50 billion to as many as 500 billion).

Each neuron is made up of a **cell body** (the central mass of the cell) with a number of connections coming off it: numerous **dendrites** (the cell's inputs—carrying information toward the cell body) and a single **axon** (the cell's output—carrying information away). Neurons are so tiny that you could pack about 100 of their cell bodies into a single millimeter. Inside a computer, the equivalent to a brain cell is a tiny switching device called a transistor. The latest, cutting-edge microprocessors (single-chip computers) contain over 2 billion transistors; even a basic microprocessor has about 50 million transistors, all packed onto an integrated circuit just 25mm square (smaller than a postage stamp)!

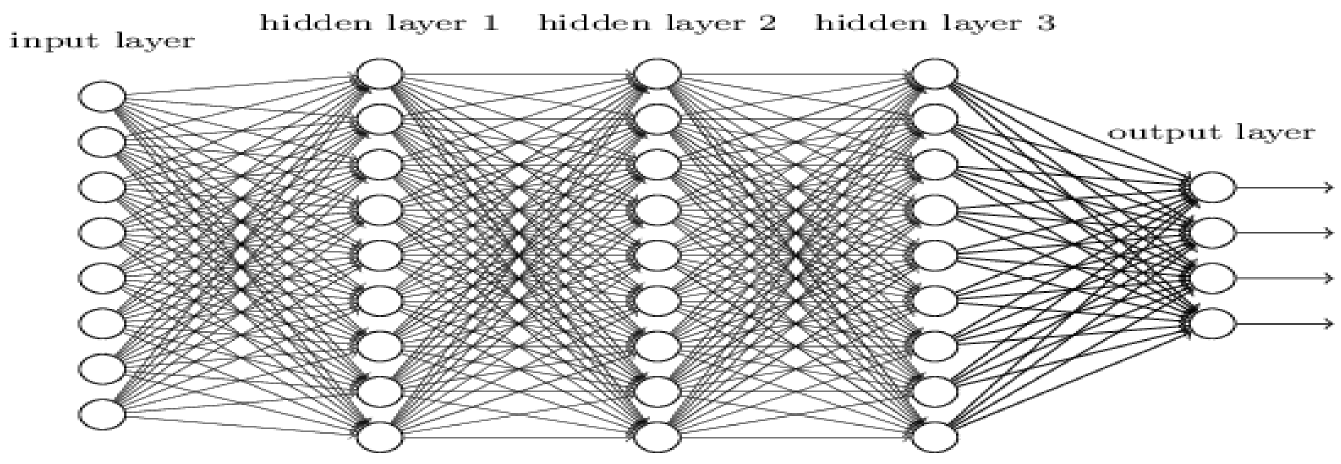


ANN is a set of connected neurons organized in layers:

- **input layer:** brings the initial data into the system for further processing by subsequent layers of artificial neurons. Like dendrites in human neuron
- **hidden layer:** a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function. Like nucleus in human neuron.
- **output layer:** the last layer of neurons that produces given outputs for the program. Like axon in human neuron

NOTE: no one how they process information inside.

A typical ANN will look like:



WORKING PROCEDURE:-

Step 1- Model initialization: -

A random initialization of the model is a common practice. The rationale behind is that from wherever we start, if we are perseverant enough and through an iterative learning process, we can reach the pseudo-ideal model.

Step 2- Forward propagate:-

The natural step to do after initializing the model at random, is to check its performance.
We start from the input we have, we pass them through the network layer and calculate the actual output of the model straightforwardly.

Step 3- Loss function:-

At this stage, in one hand, we have the actual output of the randomly initialized neural network.
On the other hand, we have the desired output we would like the network to learn. Then we define what we call: loss function (for intuition just think loss function like absolute difference or squared difference, but it changes with model to model). Basically, it is a performance metric on how well the NN manages to reach its goal of generating outputs as close as possible to the desired values.

Step 4- Differentiation:-

we can use any optimization technique that modifies the internal weights of neural networks in order to minimize the total loss function that we previously defined.

Step 5- Back-propagation:-

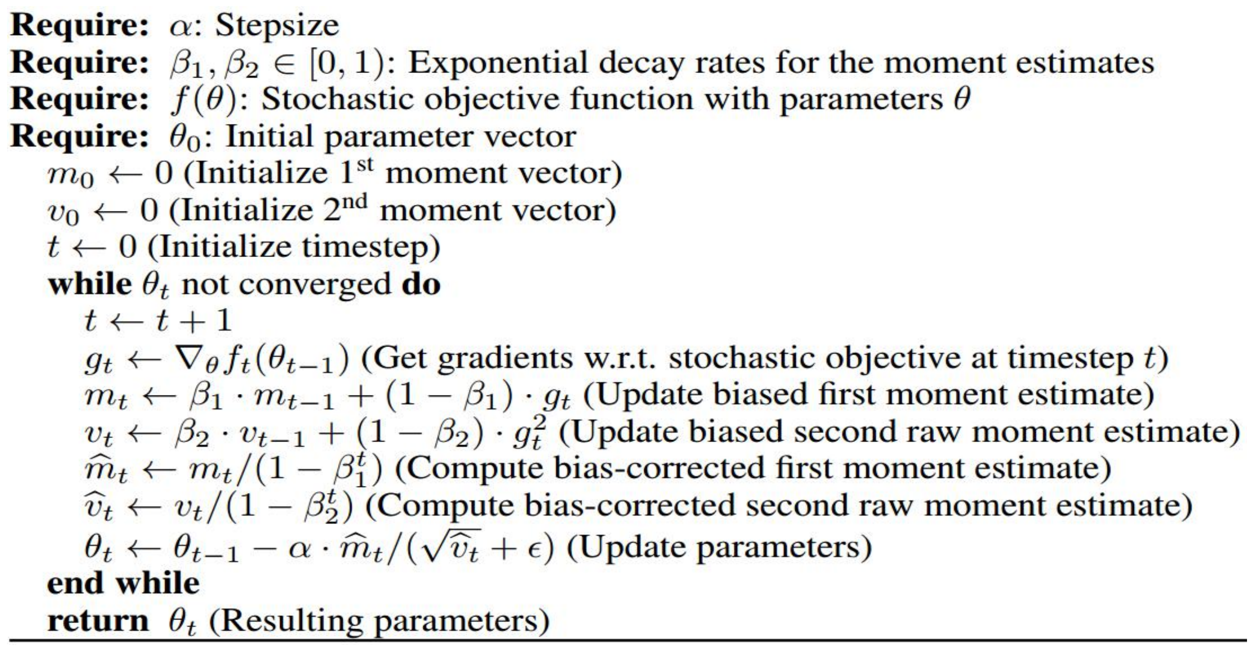
Then error in loss function is propagated from output layer to input layer by using differentiation we solved in step-4

Step 6- Weight update:-

Then corresponding weights in the network are changed in order with learning rate.

*New weight = old weight—learning rate*Derivative Rate*

In this project I am using adam optimizer so the weights are updated as follows



```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Reference link:- https://medium.com/@nishantnikhil/adam-optimizer-notes-ddac4fd7218

Step 7- Iterate until convergence:-

Just iterate the procedure from step2 to step6 until convergence

NOTE: How many iterations are needed to converge?

- This depends on how strong the learning rate we are applying. High learning rate means faster learning, but with higher chance of instability.
- It depends on the optimization method
- It depends as well on the meta-parameters of the network (how many layers, how complex the non-linear functions are

Classifier I used for this data set is VGG16 and RENET50 neural networks.since I have small amount of data I use the data augmentation.

The following parameters can be tuned to optimize the classifier:

❖ Training parameters:
  ➢ Training length (number of epochs)
  ➢ Batch size (how many images to look at once during a single training step)

❖ Neural network architecture:
  ➢ Number of layers
  ➢ Layer types (convolutional , fully-connected , or pooling)

Benchmark:-

Bench mark model: Any CNN model that gives accuracy around 20% is my benchmark model.

My created model

Conv2D layer with 16 filters,kernel_size is equal to 2 ,padding is 'same' and input shape is (50, 50 ,3)

Max Pooling layer with pooling size is equal to 2.

flatten layer and Dense layers with 2 units and activation of "SoftMax"

At compilation phase loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'] are used.

check pointer is used with 'weights. best. from bench. hdf5', as file path and save_best_only is tuned to True. bench model is trained on fit method with 32 as batch size and number epochs are 10

Architecture:-

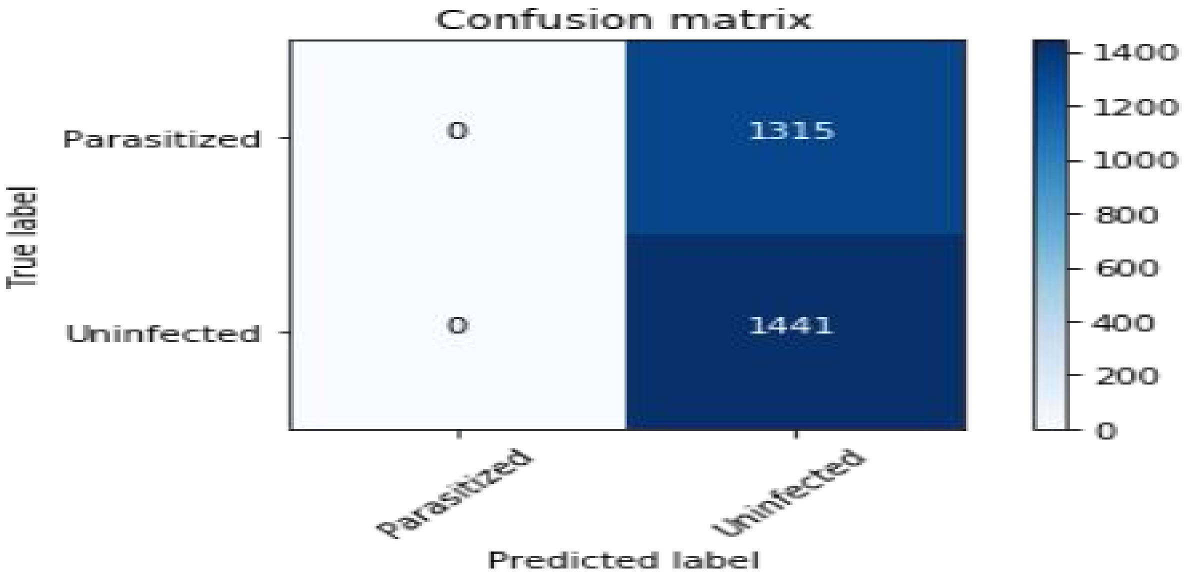
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 50, 50, 16)        208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 16)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 10000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 20002
=================================================================
Total params: 20,210
Trainable params: 20,210
Non-trainable params: 0
_________________________________________________________________
```

I declared bench mark model with single convolutional layer and dense layer.

It gives me the training accuracy: 49.66%

It gives me the testing accuracy: 52.2859%%

Confusion matrix:-



It does not predicting any thing simple gives uninfected as output.

This type of classifier causes very danger in real life.

III. Methodology

Data Preprocessing:-

The preprocessing done to "Prepare data" for the classification task ,the notebook consists of the following steps:

1. The list of images is randomized: to get good redistribution of data when applying split
2. The images are divided into a training set and a validation set (10% of original train set): To avoid overfitting
3. The images are resized into 3D tensor with shape of width = 50, height = 50, channels = 3 shaped arrays.: so that every image will have same dimensions.
5. all images are then normalized by dividing with 255.
6. all labels are converted to 2 categories by using keras.utils.to_categorical: in order use for categorical classification.

Implementation:-

As in implementation model I have created two models.one is using stranded cnn model and another is using ResNet50 model with pretrained weights.

Model1:-

Training phase:-

I choose the a Sequential model contain three convolution layers with filters 16,32,.64 for each layer ,padding is same with activation 'relu' and three maxpooling layers with pool_size equal to 2.At last I add dropout regularization with 0.2 in order to avoid overfitting.then Flatten the input inorder to make it fully connected layer. Then I add a Dense layer with 512 units with activation 'rulu' followed by a dropout layer then Dense layer with 2 units with softmax activation

Architecture:-

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 50, 50, 16)        208
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 16)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 25, 25, 32)        2080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
```

```
conv2d_4 (Conv2D)                (None, 12, 12, 64)          8256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 64)              0
_________________________________________________________________
dropout_1 (Dropout)              (None, 6, 6, 64)              0
_________________________________________________________________
flatten_2 (Flatten)              (None, 2304)                  0
_________________________________________________________________
dense_2 (Dense)                  (None, 512)                 1180160
_________________________________________________________________
dropout_2 (Dropout)              (None, 512)                   0
_________________________________________________________________
dense_3 (Dense)                  (None, 2)                   1026
=================================================================
Total params: 1,191,730
Trainable params: 1,191,730
Non-trainable params: 0
```

Compilation phase:-

Trained model has compiled using “adam” as optimizer and 'categorical_crossentropy' as loss function, used metrics is “accuracy”.

check pointer is used with 'weights.best.from_sequential_model.hdf5', as file path and save_best_only is tuned to True

Model is trained by using fit method with parameters 10 as epochs 32 as batch size

While 90% of training data is using for training purpose remaining 10% of data is used for validation purposes.

Model2:-

Training phase:-

I choose the RESNET50 model and add some more layer to the model which will looks like.

Architecture:-

```
_________________________________________________________________
Layer (type)                     Output Shape                Param #
=================================================================
resnet50 (Model)                 (None, 5, 5, 2048)          23587712
_________________________________________________________________
flatten_2 (Flatten)              (None, 51200)                 0
_________________________________________________________________
dense_2 (Dense)                  (None, 512)                 26214912
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 2565
=================================================================
Total params: 49,805,189
Trainable params: 49,752,069
Non-trainable params: 53,120
```

Pretrained weights for Resnet50 are downloaded form.
https://www.kaggle.com/ishootlaser/resnet50-pretrained-weights

Compilation phase:-

Trained model has compiled using “adam” as optimizer and 'categorical_crossentropy' as loss function, used metrics is “accuracy”.
check pointer is used with 'weights.best.from_resnet50.hdf5', as file path and save_best_only is tuned to True
Model is trained by using fit method with parameters 10 as epochs 32 as batch size
While 90% of training data is using for training purpose remaining 10% of data is used for validation purposes.

Refinement:-

I have no complications occurred during the coding process because I used keras library for build CNN.since it high Level library for deep learning there is no complication during coding process. The main trouble was occurred when I am trying to tune the learning rate and number of Convolution layers etc.

Model1:-

initial I tried with learning rate = 0.1 and with any dropout layers.and three convolution layers with filters 16,32,64 and three max pooling layers.with pool_size 2.it does not learning any thing.I think learning rate is to height and adam optimizer is stock in local minima it always show loss as 8.1136 for every epoch.then used learning rate = 0.1 for same convolution network same problem is repeated finally I used learning rate = 0.001 for the same convolution network.it give me loss for lost epoch is 0.0607 and accuracy is 97.89% I think that it model as been overfitted because I did not use data augmentation so I add 2 dropout layers to build the final model I have not only tried case that I mentioned but also some other sequential model that I wont remember.

Model2:-

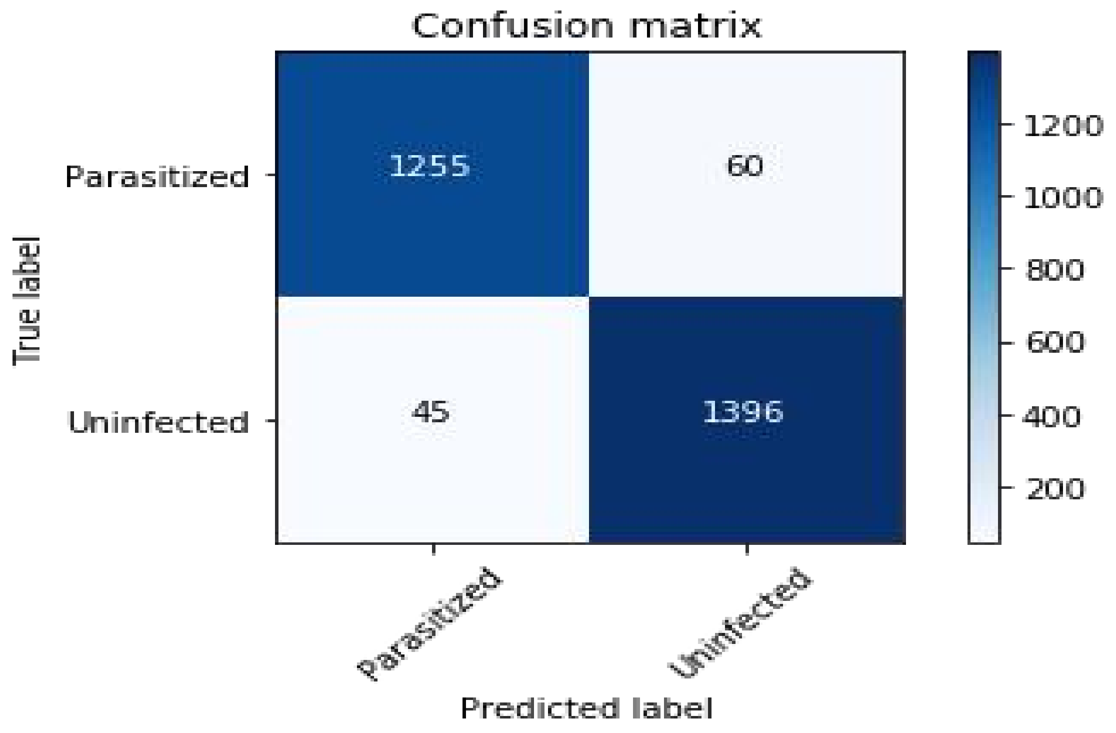
Since I have already tune the learning rate in model1 I just used same learning rate for this resnet50 model.I have issue to build this model in the first attempt I got very good accuracy but training time is very high compare to the model1 because resnet50 have more number of convolution layers .I am proud that I got a model which beats resnet50 model in predict cell_type of give image.

IV. Results

Model1:-

Model Evaluation and Validation:-

Confusion matrix:-



Form above confusion matrix we see that our model is failed to predict 105 images for over all testing images seen it is high recall problem number of false positives should be less because a a person with parasitized should not be predicted as uninfected because it may cause very serious problem In real life.predicting uninfected person as parasitized is not as big problem because he may go to further diagnosis and can know that he is not infected.

Classification report:-

	precision	recall	f1-score	support
Parasitized	0.97	0.95	0.96	1315
Uninfected	0.96	0.97	0.96	1441
Micro avg	0.96	0.96	0.96	2756
Macro avg	0.96	0.96	0.96	2756
Weighted avg	0.96	0.96	0.96	2756

As we can see that all metrics (precision, recall, f1-score) 96 % for unseen data. And can be able to generalize the data pretty well.

We can absolutely trust this model for future improvisation. Not only good but it also exceeded my expected outcome of 80%.

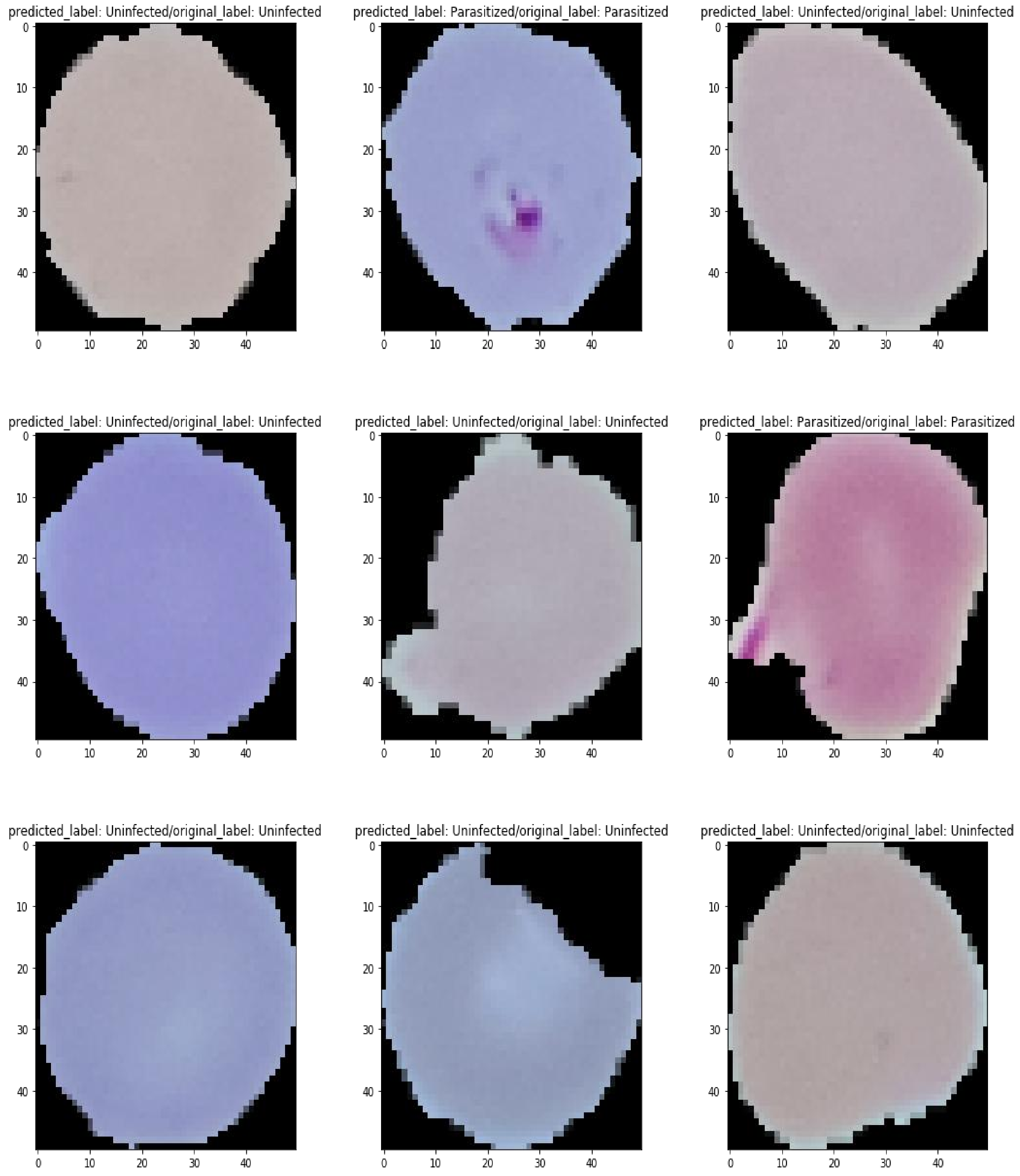
Justification:-

When compared with my benchmark model, my model gives training accuracy up to 96.45%

Whereas my testing accuracy is around 96.19%

The results obtained from my model above satisfactory as both training and testing have accuracy scores are above 95% which is pretty good.
Now I can confidently say that my model1 solution significant to solve the problem.

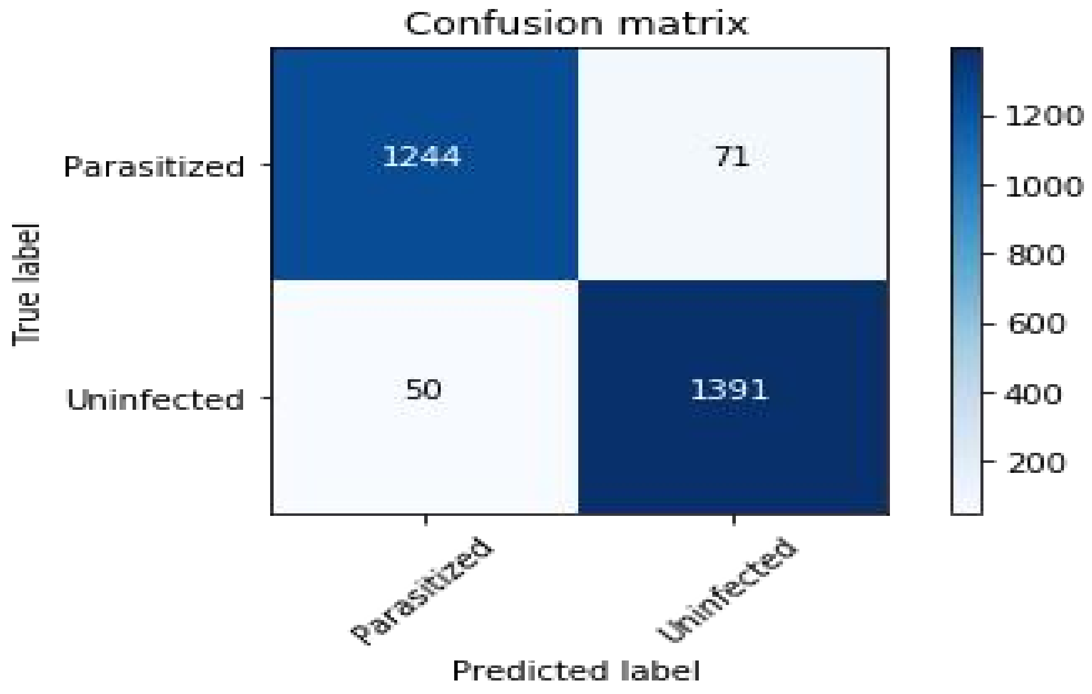
Visualization results:-


predicted_label: Uninfected/original_label: Uninfected
predicted_label: Parasitized/original_label: Parasitized
predicted_label: Uninfected/original_label: Uninfected
predicted_label: Uninfected/original_label: Uninfected
predicted_label: Uninfected/original_label: Uninfected
predicted_label: Parasitized/original_label: Parasitized
predicted_label: Uninfected/original_label: Uninfected
predicted_label: Uninfected/original_label: Uninfected
predicted_label: Uninfected/original_label: Uninfected

Model2:-

Model Evaluation and Validation:-

Confusion matrix:-



Form above confusion matrix we see that our model is failed to predict 121 images for over all testing images seen it is high recall problem number of false positives should be less because a a person with parasitized should not be predicted as uninfected because it may cause very serious problem In real life.predicting uninfected person as parasitized is not as big problem because he may go to further diagnosis and can know that he is not infected.

classification report for RESNET50 model:-

	precision	recall	f1-score	support
Parasitized	0.96	0.95	0.95	1315
Uninfected	0.95	0.97	0.96	1441
Micro avg	0.96	0.96	0.96	2756
Macro avg	0.96	0.96	0.96	2756
Weighted avg	0.96	0.96	0.96	2756

As we can see that all metrics (precision, recall, f1-score) 95% for parasitized and 0.96 for uninfected. From this we can conclude that RESNET50 is robust to unseen data. And can be able to generalize the data pretty well.
We can absolutely trust this model for future improvisation. Not only good but it also exceeded my expected outcome of 80%.

Justification:-

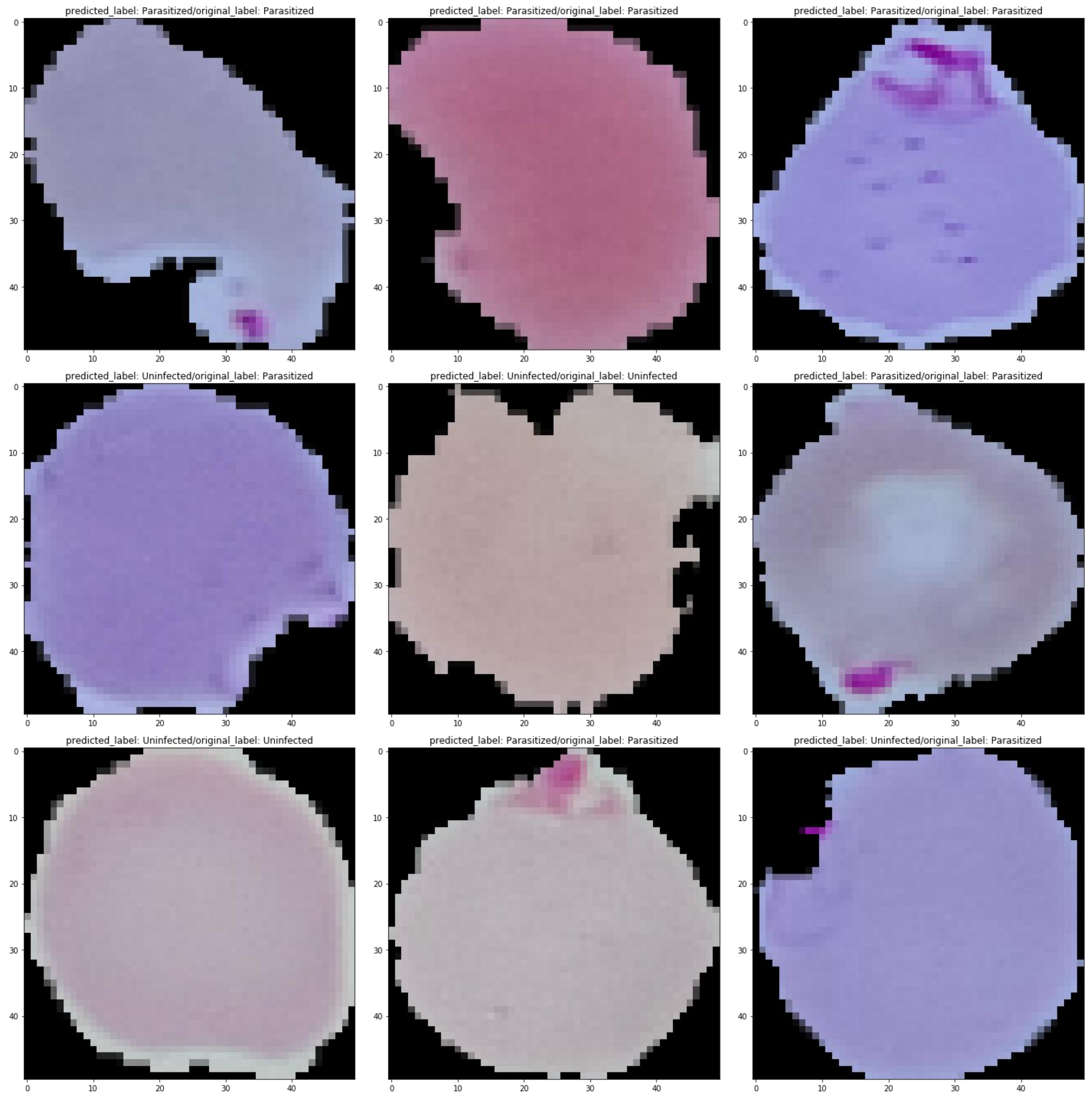
When compared with my benchmark model, my model gives training accuracy up to 94.94%

Whereas my testing accuracy is around 95.60%

The results obtained from my model above satisfactory as both training and testing have accuracy scores are above 95% which is pretty good.

Now I can confidently say that my model2 solution significant to solve the problem.

Visualization results:-

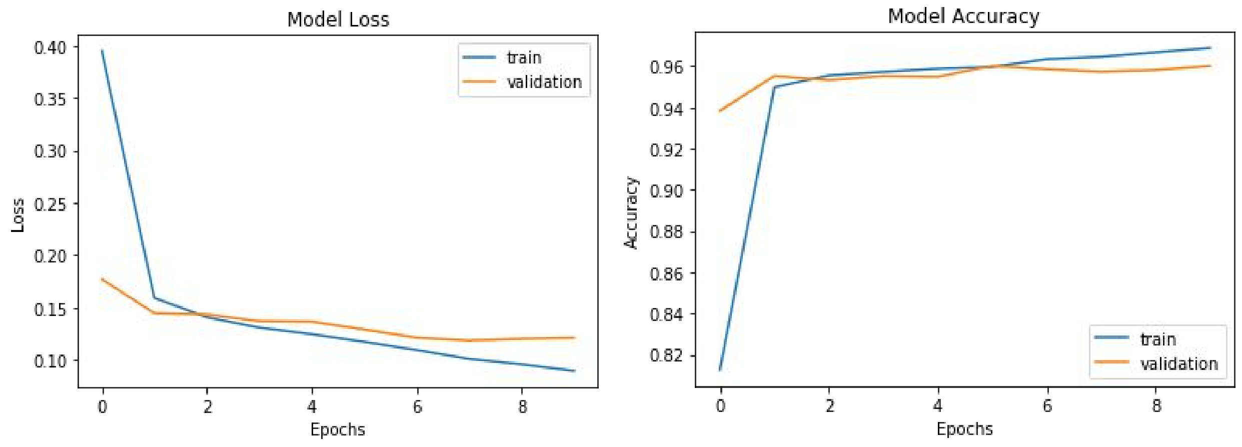

predicted_label: Parasitized/original_label: Parasitized
predicted_label: Parasitized/original_label: Parasitized
predicted_label: Parasitized/original_label: Parasitized
predicted_label: Uninfected/original_label: Parasitized
predicted_label: Uninfected/original_label: Uninfected
predicted_label: Parasitized/original_label: Parasitized
predicted_label: Uninfected/original_label: Uninfected
predicted_label: Parasitized/original_label: Parasitized
predicted_label: Uninfected/original_label: Parasitized

Which model to choose:-

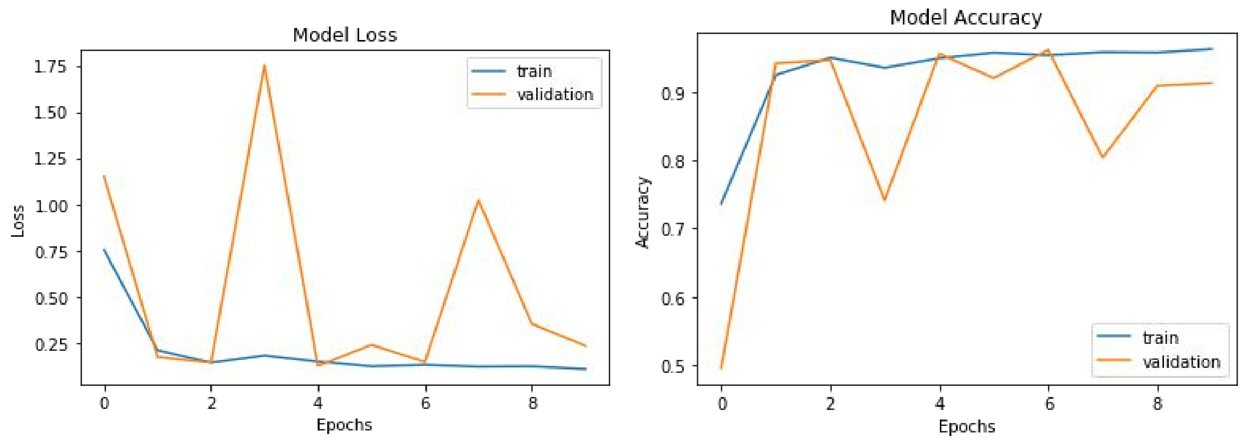
From the above two models, F-score is almost same but our problem is a high recall problem due to this RESNET 50 model classify 71 persons as uninfected although they are parasitized .This is a serious problem that I have already explained in the confusion matrix but our standard CNN model classify only 60 persons as uninfected although they are parasitized.Due to this I will prefer to choose standard CNN model instead of RESNET 50 as my classifier.

V. Conclusion

History of loss and accuracy for stranded cnn model.



History of loss and accuracy for the RESNET50 model.



Reflection:-

In this capstone project I have taken image classification as my thought of interest inspired from dog breed classifier. In this process I have learned many things.

1) First thing I have learned about data retrieval processes. When I am doing research about retrieval process, I have come across a lot of surprising methods try. Out of all I decided to use load files method in cv2
2) I have also learn how to use kaggle kernals and how to commit it and reproduce my work.
3) Then I used my skills on representation of overall number of images, and also category wise in both training and testing images by using Matplotlib library. I have realized that its most informative in beginning in understanding size of your project.

4) When I stated loading images, I have found many useful methods like image. load_img in keras. preprocessing, cv2.load_img, plt.imread etc., there are lots of them.
5) I am also learn how to handle the corrupted images in dataset
6) Then I have learned about using tqdm for reading a list of images.
7) I learn that Image resizing is more important in image classification because we can’ t expect every image of same size.
8) Then image normalization is used to standardize the all image’ s (division by 255). I came to know about how numbers in image array changes with brightness. How dark areas will replicate zeros, bright areas replicate 225 etc.,
9) Then I learned that How we divide the data into training and validation using traint_test_split method.
10) Here comes the heart of project, creating a CNN model, fitting it to training data and testing on test data evaluating validation curves, learn from confusion matrix doing modifications on models and there’ s more going on.
11) Last but not least without visualizing results we can’ t trust the robustness of a model.

Improvement:-

One thing, that can be improved from my models is we can use Kfold method for splitting data while fitting model, when we have enough time.
And also use grid search for tuning hyper parameters but it takes so much time.
And also data augmentation can be applied.

FINAL NOTE:-

But, when I trying to improve my two models, I have done two things.
Instead of train_test_split method, I have used Kfold split method, soon I came to observe that, both models are taking infinite amount of time to fit training data, so to speak I kind of drop the idea of using it.
When I started this project I used RESNET50 model with out pre-trained weights I build the models manually in keras tried this takes days to train the model. So I quit the idea.

Though Both models are giving same performances on training and testing data,
I would preferred standard CNN model, because, its training time is far less than RESNET50 model,
Resnet50 model training time exponential to VGG16 training time.
This is a high recall problem,standard CNN has high recall than Resnet 50.
So, I consider standard CNN model as my final model.