Springboard

Capstone Report

Capstone Project 2

Detecting Malaria using Image classification

Vaibhav Agarwal
February 1, 2020

Contents

1. Problem Statement
2. Introduction
3. Methodology
    - Preprocessing
    - Feature Enhancement
4. Deep Learning Model
5. Observations
6. Conclusion

Problem Statement

To solve with the best accuracy by detecting and deploying Image Cells that contain Malaria or not.

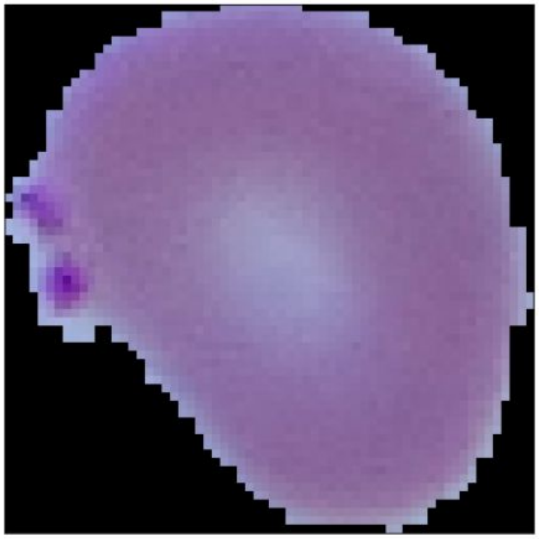
Introduction

Malaria is a life-threatening disease caused by parasites that are transmitted to people through the bites of infected female Anopheles mosquitoes. It is preventable and curable.
Here I am trying to solve with the best accuracy by detecting and deploying Image Cells that contain Malaria or not!
The client for this project is Healthcare Industries. This project will help them as a Healthcare company will be able to classify the malaria-infected and uninfected cells. They will be able to deploy this predictive system even in remote corners of the world.
The data used in this project is taken from kaggle. The dataset is divided into two parts, one containing infected cells images and the other containing uninfected cell images.
The kaggle link for data is: https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria

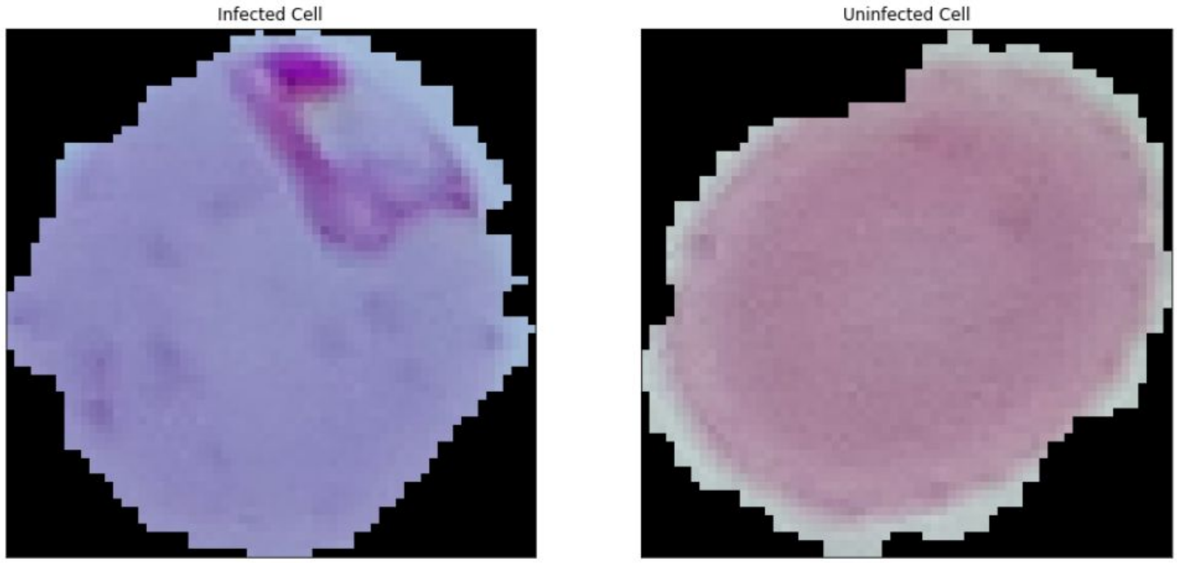
Methodology

The data is in the form of images, this means in the form of 3d array. The images are provided in two folders i.e. infected and uninfected. We have to append the data into a list of arrays while labelling another list of categorical values (0,1). We are reducing the size of images to (64, 64) to make it easier for processing, and to keep the uniformity.
The infected and uninfected cell images can be seen side by side below.

Infected Cell



Uninfected Cell



After reading the images, the shape of the data is (27558, 64, 64, 3). The number means:

- 27558: the length of the list is the number of images
- (64, 64): is the pixel size of image
- 3: contains the values of RGB of the image.

And the shape of the labels is (27558, 2). Here the number means:

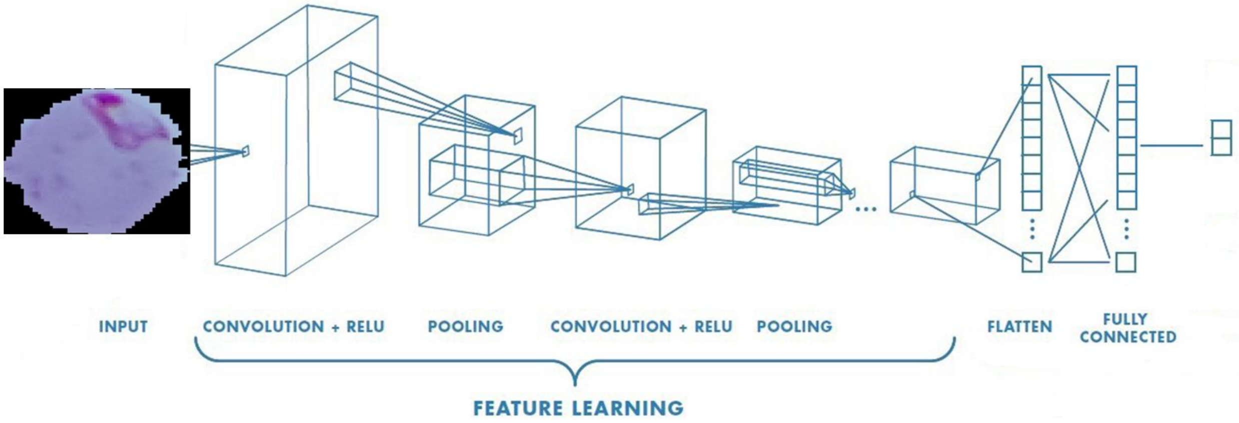
- 27558: the length of the list corresponding to the images data
- 2: labels categorical values of 0 or 1

Following are the data processing steps:

1. The data and labels are converted into numpy array. It can be numpy array before but just to be sure.
2. An array is generated containing numbers zero to the length of data. The array is then randomly shuffled.
3. The array is used to shuffle the data and labels. As they were containing first infected then uninfected images.
4. The data is converted to float data type. This is done so when scaling is done, the numbers after decimal remains.
5. The data is scaled by dividing it by 255, the largest number in representing colors.
6. The data is then split in 70-30 ratio. With 70% of data in train and 30% of data in test.

Now the images has been represented as the list of array(numbers). We can proceed with the model building.

Architecture



Deep Learning Model

We have used the CNN method of neural network for the model. The details of the model are as follows:

- Input shape = (64,64,3)   #the shape of an image as array
- Layers = 4
- Method = SeparableConv2D
- Each layer method
    - Conv2D
    - Filters = 64, 128, 256, 64
    - Kernel size = (3,3)
    - Activation = relu
    - BatchNormalization
    - Max Pooling = (2,2)
    - Dropout = 0.2
- Flatten = Yes
- Pre-final layer = Dense: 256 nodes, activation: relu
- Dropout = 0.2
- Final Layer = 2 output nodes, activation: sigmoid

Data Augmentation Parameters

Train:

- rescale=1./255,
- shear_range=0.2,
- zoom_range=0.2,
- horizontal_flip=True,
- vertical_flip=True

Validation/Test:

- rescale=1./255

Callbacks

Earlystopping:

- Monitor: val_loss
- Min_delta = 0.001
- Patience = 7
- Verbose = 2
- Mode = ‘min’

ModelCheckpoint:

- Monitor = 'val_loss'
- Verbose = 1,
- Save_best_only = True
- Mode = 'min'
- Save_weights_only = False

Compile

- Loss = binary_crossentropy
- Optimizer = adam
- Metrics = accuracy

Fit

- steps_per_epoch = 1000
- verbose = 1
- validation_steps = 200
- epochs = 10
- use_multiprocessing=True

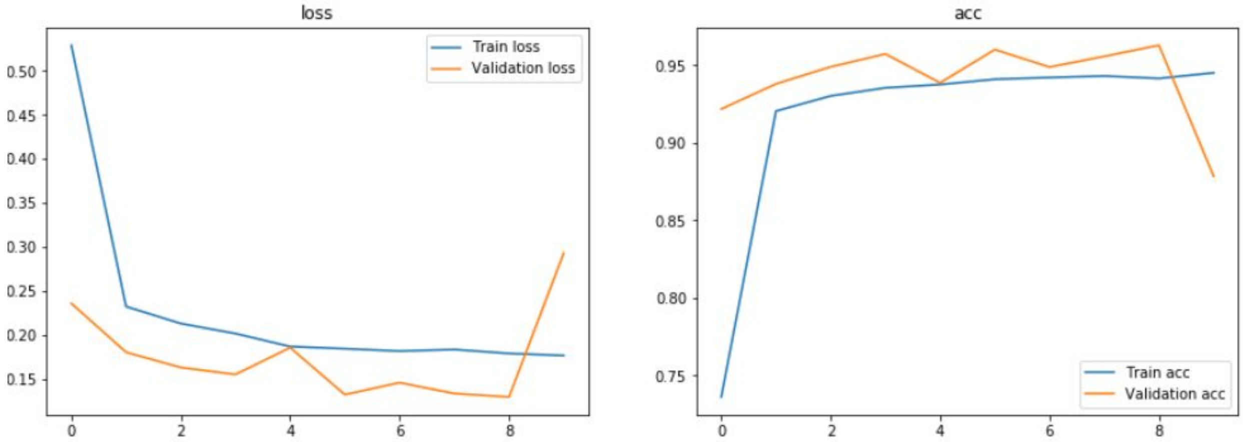
Note: We have loaded the model weights for the lowest validation loss while training.

Observations:

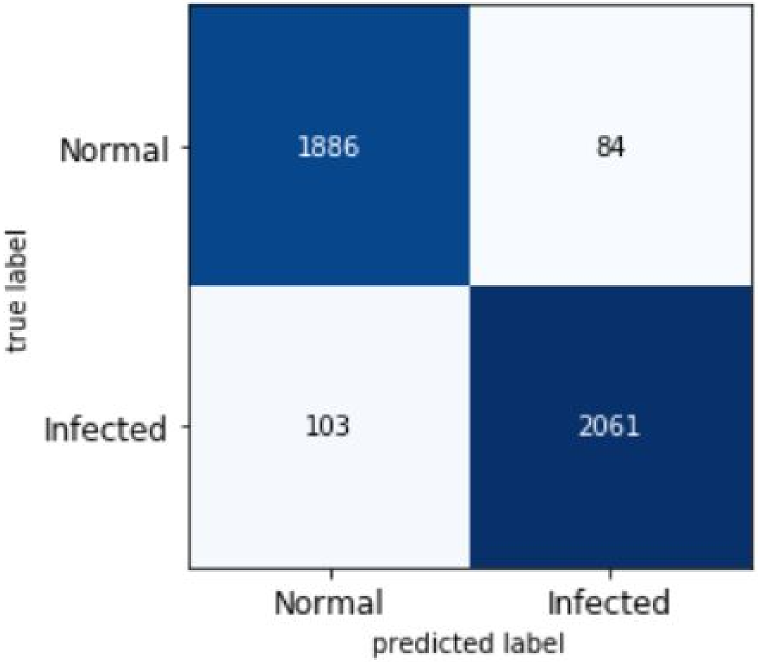
The results are:

- Training Accuracy = 0.9416
- Training Loss = 0.1792
- Validation Accuracy = 0.9630
- Validation Loss = 0.1298
- Testing Accuracy = 0.9506
- Testing Loss = 0.1489

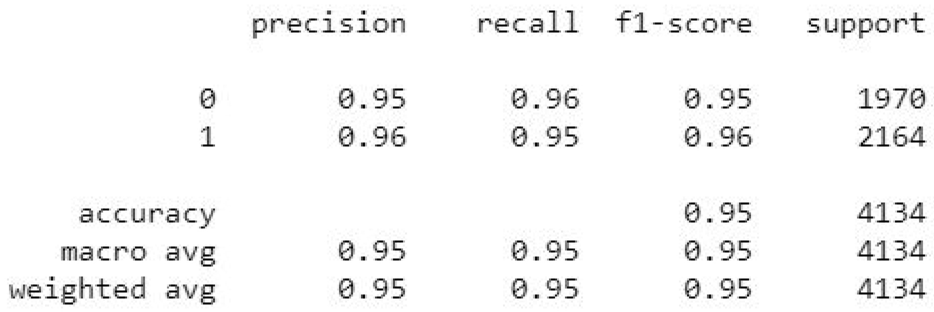
Training History:



Confusion Matrix



Classification Report



```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      1970
           1       0.96      0.95      0.96      2164

    accuracy                           0.95      4134
   macro avg       0.95      0.95      0.95      4134
weighted avg       0.95      0.95      0.95      4134
```

Conclusion

We have got a good test accuracy of 0.94. The model is with good precision able to successfully predict from the blood cell image if it is infected or not. The f1-score of 0.95 shows the capability of model in having a good recall and precision.

The only assumption we took here is that the infected cells are only infected by malaria, no other disease is affecting the cells. We have trained malaria infected cells against healthy cells only.

The model needs to be updated regularly with new images. This model can be used in integration with other models when other diseases might be present.