Image Classification – Milestone Report-2

Objective:

- To reduce the burden for microscopists in resource-constrained regions and improve diagnostic accuracy
- Build an Image Classification model to obtain highest level of accuracy
- Making the model as small and computationally efficient as possible

Introduction:

Overview:

- The dataset contains 2 folders - Infected - Uninfected and it has total of 27,558 images.
- An instance of how the patient-ID is encoded into the cell name is shown herewith: “P1” denotes the patient-ID for the cell labeled “C33P1thinF_IMG_20150619_114756a_cell_179.png”

Source:

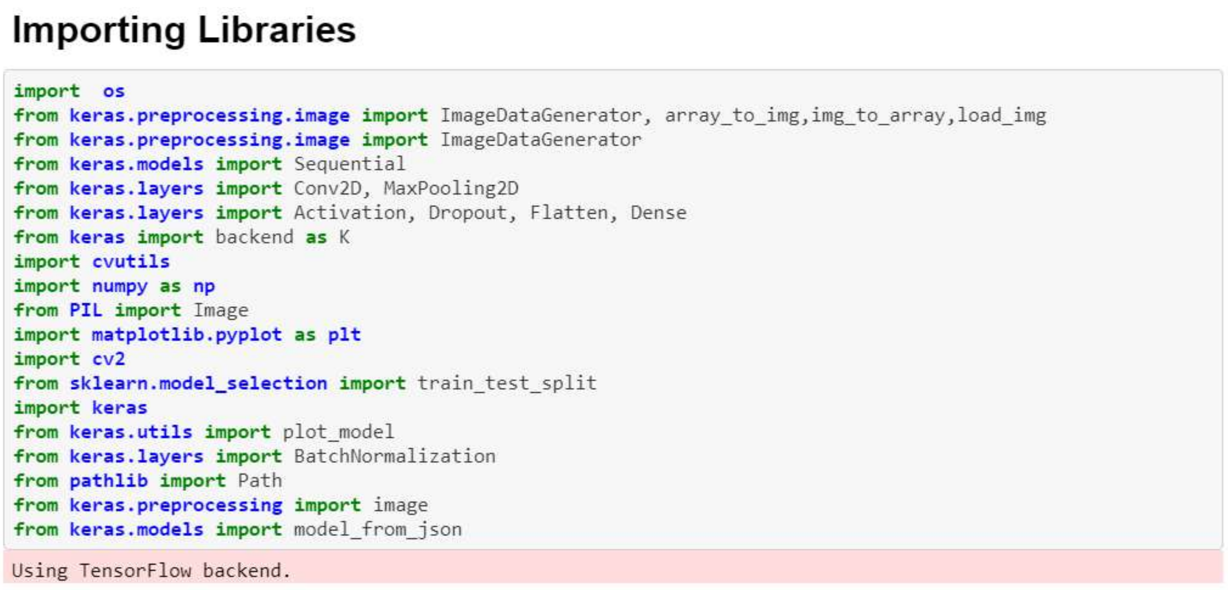
- This Dataset is taken from the official NIH Website: https://ceb.nlm.nih.gov/repositories/malaria-datasets/

Tools:

- Python -3
- Jupyter Notebook
- TensorFlow
- Keras

Approach:

- Import required libraries
- Pre-processing image data
- Train and test data
- Building Model
- Compile & Training Model



Importing Libraries

```
import  os
from keras.preprocessing.image import ImageDataGenerator, array_to_img,img_to_array,load_img
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
import cvutils
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import cv2
from sklearn.model_selection import train_test_split
import keras
from keras.utils import plot_model
from keras.layers import BatchNormalization
from pathlib import Path
from keras.preprocessing import image
from keras.models import model_from_json
```

```
Using TensorFlow backend.
```

Pre-processing image data

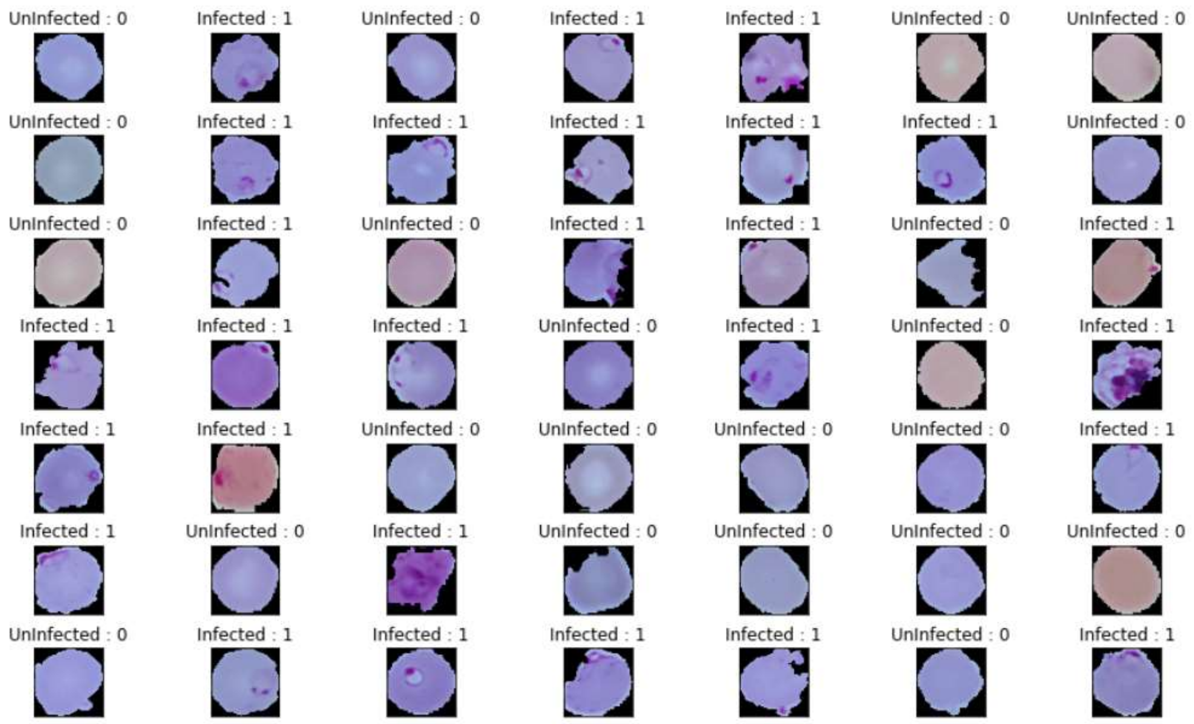
- Read the Image data using os.listdir
- Loop through all the images in both folders (Parasitized & Uninfected)
- Resize all the images with 50 x 50 pixel and convert the image into array
- Add all the resized image array to list

Train and test data

- Split the data in to train and test using sklearn.model_selection
- 70% Training, 10% Validation and 20% testing
- Normalize the data so that CNN process the data quickly and efficiently

Visualization:

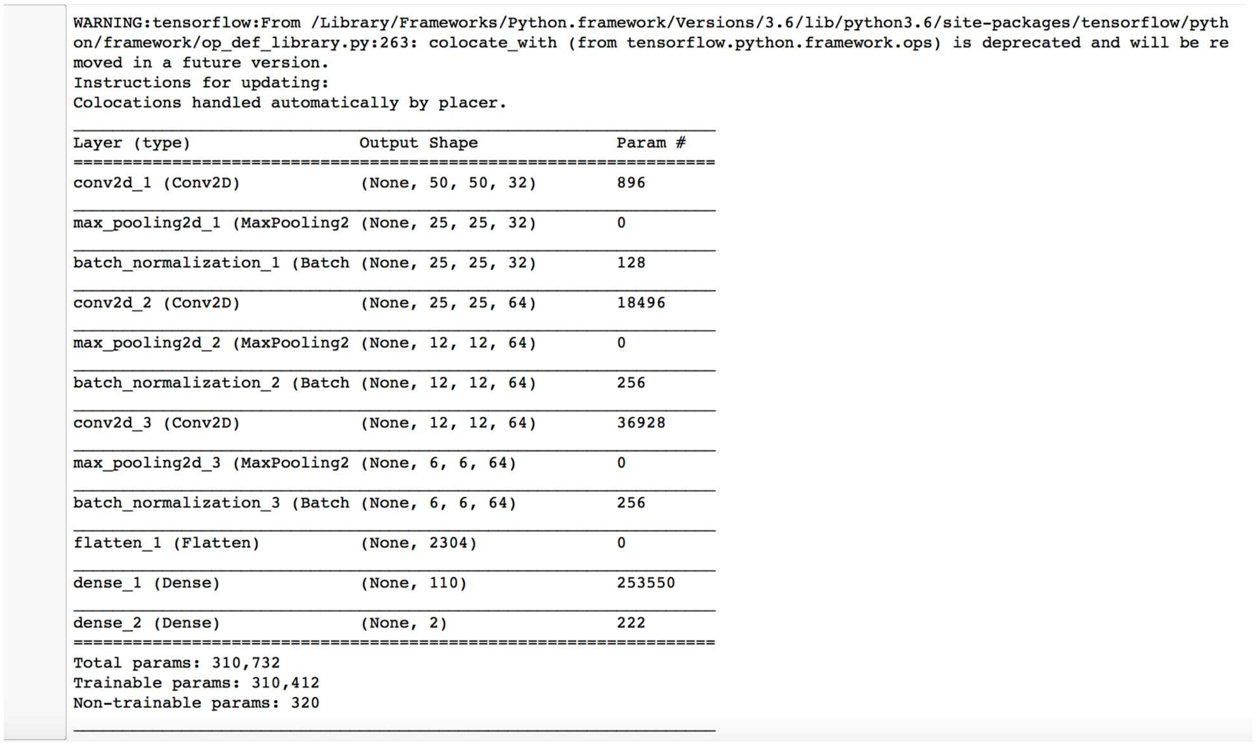
Sample resized image Data



Building Model:

- Call keras Sequential model
  - Model.Sequential()
- Add convolution layer (Conv2D) to process image files
  - Params:
    - Kernel_size = 3 ,3 (window size)
    - Input shape = 50, 50, 3
    - Activation = relu
- Add max pooling so that only the higher weights will be passed on to next layer
  - Maxpooling2D: pool_size = 2,2
- Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass
  - Dropout(0.25)

Model Summary



```
WARNING:tensorflow:From /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/tensorflow/pyth
on/framework/op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be re
moved in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 50, 50, 32)        896       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 32)        0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 25, 25, 32)        128       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 64)        18496     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0         
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 64)        256       
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0         
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 64)          256       
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 110)               253550    
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 222       
=================================================================
Total params: 310,732
Trainable params: 310,412
Non-trainable params: 320
_________________________________________________________________
```

Parameter Details:

- Total params: 310,732
- Trainable params: 310,412
- Non-trainable params: 320

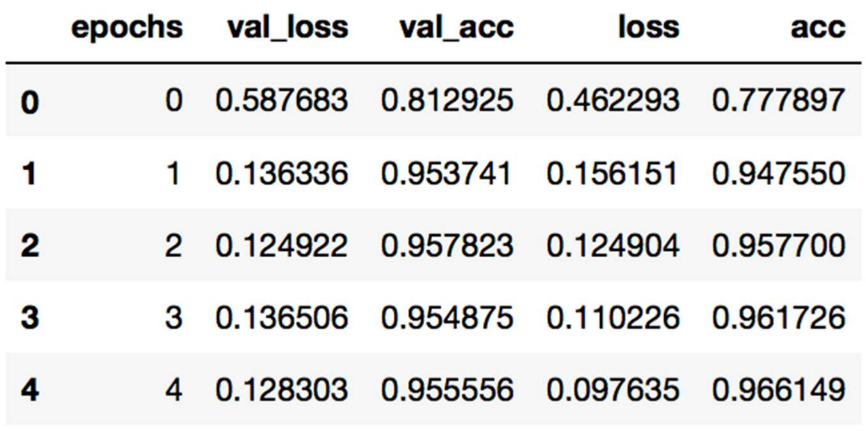
Compile & Train:

- Compile the Model
- Fit the data to Model
- Specify
    - Batch Size
    - Epochs
    - validation data

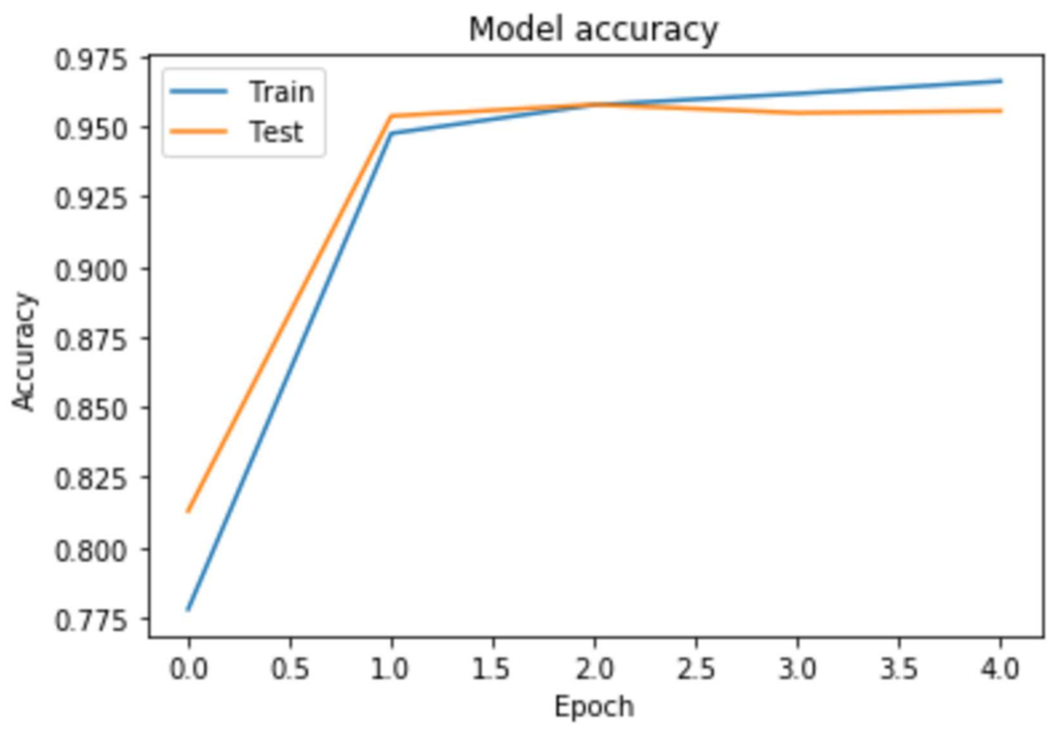
Training Results

```
WARNING:tensorflow:From /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/tensorflow/pyth
on/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Train on 17636 samples, validate on 4410 samples
Epoch 1/5
17636/17636 [==============================] - 74s 4ms/step - loss: 0.4623 - acc: 0.7779 - val_loss: 0.5877 - val_acc
: 0.8129
Epoch 2/5
17636/17636 [==============================] - 71s 4ms/step - loss: 0.1562 - acc: 0.9476 - val_loss: 0.1363 - val_acc
: 0.9537
Epoch 3/5
17636/17636 [==============================] - 72s 4ms/step - loss: 0.1249 - acc: 0.9577 - val_loss: 0.1249 - val_acc
: 0.9578
Epoch 4/5
17636/17636 [==============================] - 71s 4ms/step - loss: 0.1102 - acc: 0.9617 - val_loss: 0.1365 - val_acc
: 0.9549
Epoch 5/5
17636/17636 [==============================] - 69s 4ms/step - loss: 0.0976 - acc: 0.9661 - val_loss: 0.1283 - val_acc
: 0.9556
```

Model Accuracy:



	epochs	val_loss	val_acc	loss	acc
0	0	0.587683	0.812925	0.462293	0.777897
1	1	0.136336	0.953741	0.156151	0.947550
2	2	0.124922	0.957823	0.124904	0.957700
3	3	0.136506	0.954875	0.110226	0.961726
4	4	0.128303	0.955556	0.097635	0.966149



Saving Model Weights

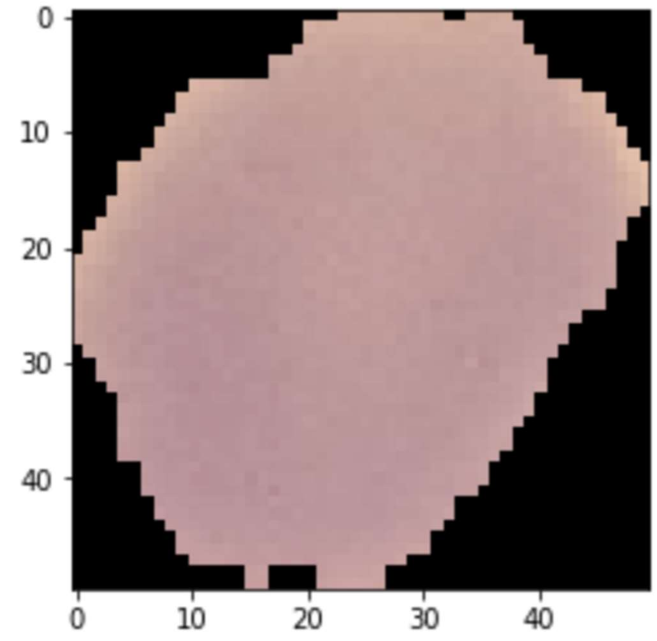
- Save the Model structure so that it can reused, this will save lot time
- Model structure is stored in JSON file format
- Weights are stored in .h5 format

Model Testing with Weights

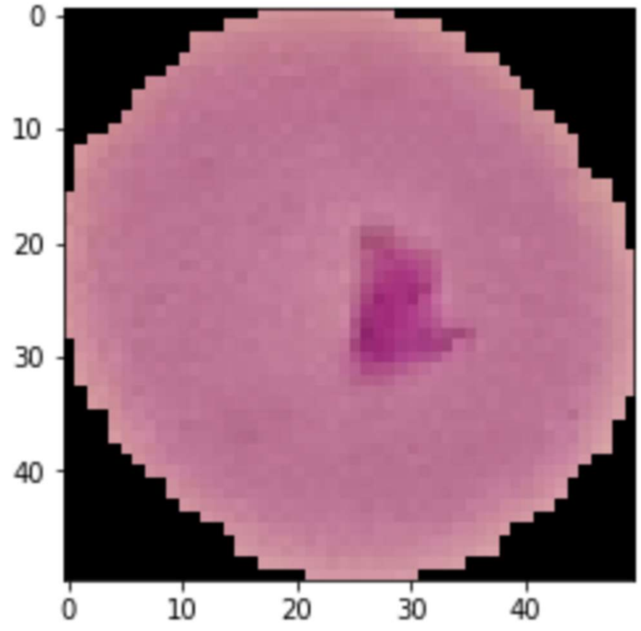
- Load the saved Model structure and Weight
- Load the Image that we want to validate
- Convert the image with target size
- Call model.predict to predict the likelihood of the image

Predictions:

```
This is image is a Uninfected - Likelihood: 0.904207
```



```
This is image is a Infected - Likelihood: 0.999745
```



Conclusion:

With this model we can predict the likelihood of the given image that it is Infected or Uninfected.

This Model used only 2 Convolution layers with activation as relu and dense as 110, and this model requires less computational power