Artificial Intelligence Internship

Project Report

Project Title : # Malaria Detection using Deep Learning

Team Members: **Kranthi Kumar**

Jahnavi Behara

Menda Kavya Sampath

Yoga Sai Krishna Ramineni

Shubham Mahapatra

Description : **Malaria remains a major burden on global health, with roughly 200 million cases worldwide and more than 400,000 deaths per year. Besides biomedical research and political efforts, modern information technology is playing a key role in many attempts at fighting the disease. One of the barriers to a successful mortality reduction has been inadequate malaria diagnosis in particular. To improve diagnosis, image analysis software and machine learning methods have been used to quantify parasitemia in microscopic blood slides.**

Introduction:

Malaria is one the deadliest disease afflicting the mankind, with more than 200 million new cases every year, and over 400,000 reported deaths (WHO, 2018). The causative agent of infection, Plasmodium spp. parasites have developed resistance to almost all currently marketed drugs including the current treatment choice artemisinin-based combination therapy (ACT) . This underscores an urgent need to discover next generation anti-malarials. Traditionally, the discovery of new bioactive chemo types relies on cell or target-based screening of natural or synthetic compound libraries. High Throughput Screening (HTS) using either approach entails screening of large library of compounds. This process is often inefficient and not cost effective because of high failure rate at subsequent stages of drug discovery.

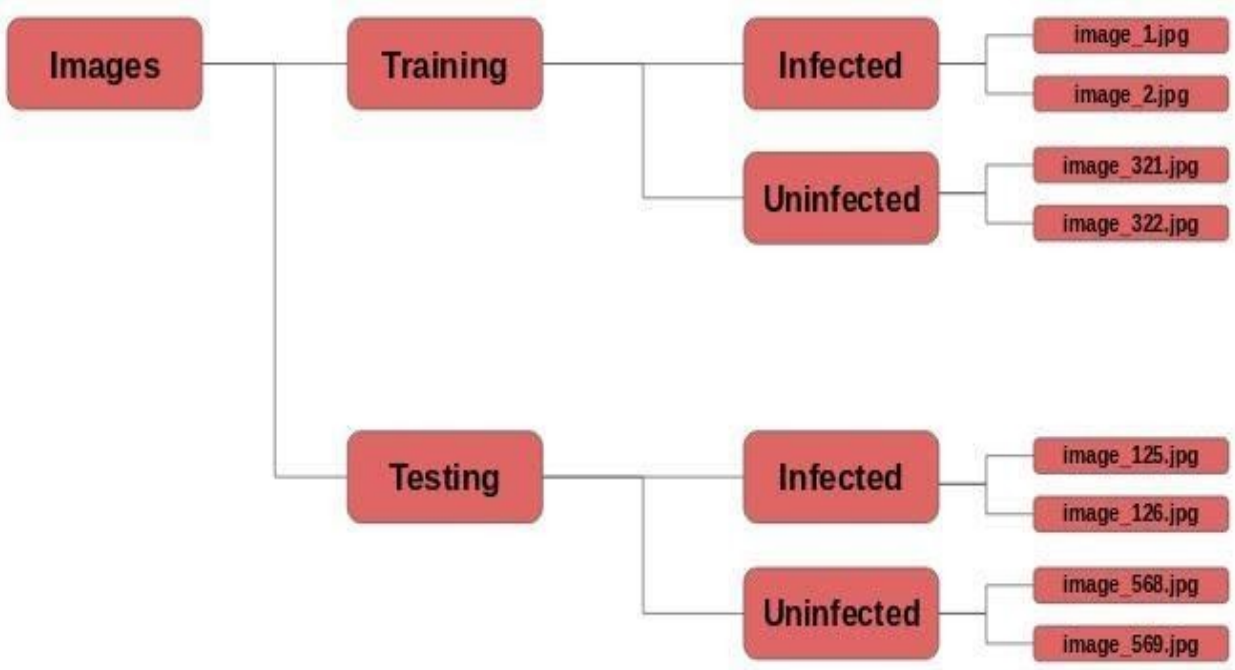
Description:

Malaria remains a major burden on global health, with roughly 200 million cases worldwide and more than 400,000 deaths per year. Besides biomedical research and political efforts, modern information technology is playing a key role in many attempts at fighting the disease. One of the barriers to a successful mortality reduction has been inadequate malaria diagnosis in particular. To improve diagnosis, image analysis software and machine learning methods have been used parasitemia in microscopic blood slides

Skills Required:

Python,Python Web Frameworks,CNN,Project Development

Flow Chart:



The following steps are:

1.Data Collection

- Create Train and Test sets

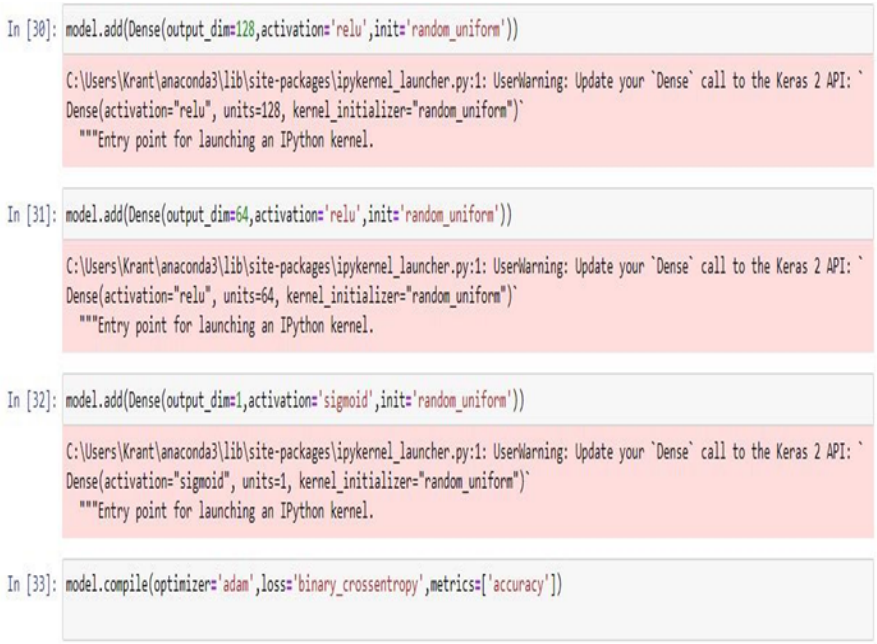
2.Importing libraries and data sets
3. Model Architecture
4. Image Data Generator
5. Visual Accuracy and Loss
6. Model Generator
7. Training and Accuracy
8. Application Building

Model Architecture:

Now, let's define the model architecture. Since our images are not very large and complex, we can use just a few layers of CNN and get good results. Hence our model will be as follows:

First layer with input shape defined as the shape of the images, followed by max pooling

Second layer followed by max pooling
Third layer followed by max pooling
Flatten layer to flatten the output from third layer and feed to the Dense layer of 128 activation/neuron nodes
Output Dense layer with a single unit/neuron because we only have two classes; binary classification problem. We'll use sigmoid activation function since we only need 0,1 as outputs to classify into two classes of infected and uninfected.



```
In [34]: model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_2 (MaxPooling2	(None, 31, 31, 32)	0
flatten_2 (Flatten)	(None, 30752)	0
dense_8 (Dense)	(None, 128)	3936384

We can print the model summary to display output shape at each level of the architecture and the number of parameters to learn at each stage.

Image Data Generator:

Rescale: all images will be rescaled by 1./255
Training _directory: this is the source directory for training images
Target_size : all images will be resized to 80x80
Batch_size : the number of images in one batch of optimizer loss cycle, the Train_generator will flow training images in batches of 256
Class_mode: we'll keep this as binary since we'll use Binary_crossentropy loss, we need binary labels

```
In [2]: from keras.preprocessing.image import ImageDataGenerator

In [3]: train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
        test_datagen= ImageDataGenerator(rescale=1./255)

In [23]: x_train=train_datagen.flow_from_directory(r'E:\Smart Bridge intern\project\data collection\cell_images\train set',target_size=(64
         x_test= test_datagen.flow_from_directory(r'E:\Smart Bridge intern\project\data collection\cell_images\test set',target_size=(64,
         Found 22046 images belonging to 2 classes.
         Found 5512 images belonging to 2 classes.

In [24]: print(x_train.class_indices)
         {'Parasitized': 0, 'Uninfected': 1}
```

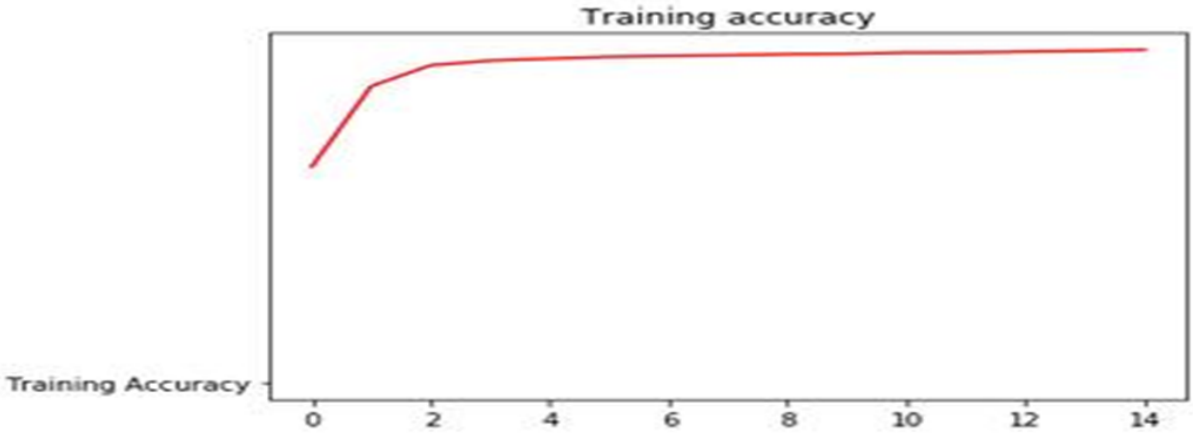
In [35]:
```
model.fit_generator(x_train,samples_per_epoch=689,epochs=25,validation_data=x_test,nb_val_samples=172)
```

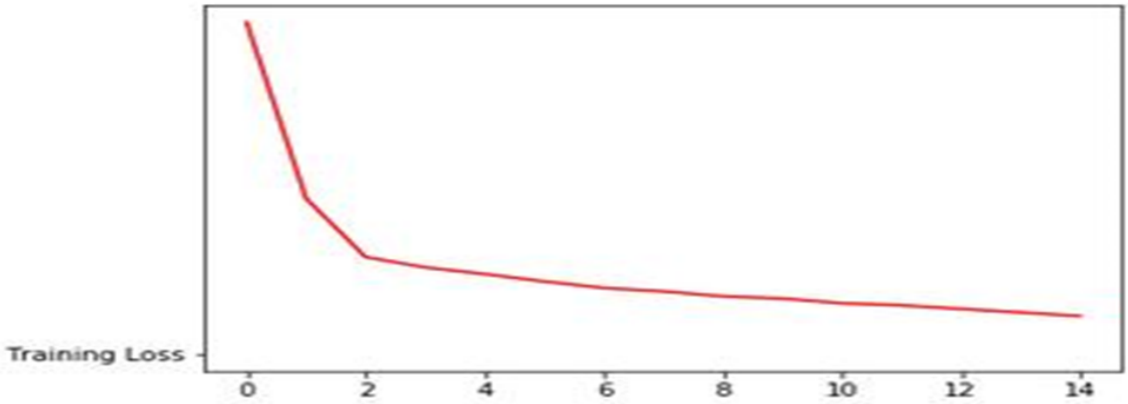
```
C:\Users\Krant\anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: The semantics of the Keras 2 argument `steps_p
er_epoch` is not the same as the Keras 1 argument `samples_per_epoch`. `steps_per_epoch` is the number of batches to draw from
the generator at each epoch. Basically steps_per_epoch = samples_per_epoch/batch_size. Similarly `nb_val_samples`->`validation_
steps` and `val_samples`->`steps` arguments have changed. Update your method calls accordingly.
  """Entry point for launching an IPython kernel.
C:\Users\Krant\anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `fit_generator` call to the Keras
2 API: `fit_generator(<keras_pre..., epochs=25, validation_data=<keras_pre..., steps_per_epoch=10, validation_steps=172)`
  """Entry point for launching an IPython kernel.
```

```
Epoch 1/25
10/10 [==============================] - 39s 4s/step - loss: 0.6874 - acc: 0.5344 - val_loss: 0.7136 - val_acc: 0.5006
Epoch 2/25
10/10 [==============================] - 34s 3s/step - loss: 0.6887 - acc: 0.5641 - val_loss: 0.6934 - val_acc: 0.5183
Epoch 3/25
10/10 [==============================] - 34s 3s/step - loss: 0.6674 - acc: 0.6094 - val_loss: 0.6876 - val_acc: 0.5453
Epoch 4/25
10/10 [==============================] - 34s 3s/step - loss: 0.6556 - acc: 0.6297 - val_loss: 0.6820 - val_acc: 0.5731
Epoch 5/25
10/10 [==============================] - 34s 3s/step - loss: 0.6370 - acc: 0.6422 - val_loss: 0.6916 - val_acc: 0.5731
Epoch 6/25
10/10 [==============================] - 34s 3s/step - loss: 0.6616 - acc: 0.6219 - val_loss: 0.7263 - val_acc: 0.5343
Epoch 7/25
10/10 [==============================] - 35s 3s/step - loss: 0.6349 - acc: 0.6609 - val_loss: 0.6824 - val_acc: 0.5566
Epoch 8/25
10/10 [==============================] - 34s 3s/step - loss: 0.6350 - acc: 0.6594 - val_loss: 0.6812 - val_acc: 0.5873
Epoch 9/25
10/10 [==============================] - 34s 3s/step - loss: 0.6166 - acc: 0.6906 - val_loss: 0.6948 - val_acc: 0.5744
Epoch 10/25
10/10 [==============================] - 35s 3s/step - loss: 0.6202 - acc: 0.6625 - val_loss: 0.6987 - val_acc: 0.5897
Epoch 11/25
10/10 [==============================] - 35s 3s/step - loss: 0.6232 - acc: 0.6734 - val_loss: 0.7175 - val_acc: 0.5671
Epoch 12/25
10/10 [==============================] - 37s 4s/step - loss: 0.6184 - acc: 0.6562 - val_loss: 0.6622 - val_acc: 0.5947
Epoch 13/25
10/10 [==============================] - 41s 4s/step - loss: 0.6158 - acc: 0.6828 - val_loss: 0.6686 - val_acc: 0.6002
Epoch 14/25
10/10 [==============================] - 43s 4s/step - loss: 0.6102 - acc: 0.6688 - val_loss: 0.7019 - val_acc: 0.6044
Epoch 15/25
10/10 [==============================] - 39s 4s/step - loss: 0.5960 - acc: 0.6969 - val_loss: 0.7779 - val_acc: 0.5801
Epoch 16/25
10/10 [==============================] - 39s 4s/step - loss: 0.6000 - acc: 0.6859 - val_loss: 0.7192 - val_acc: 0.5984
Epoch 17/25
10/10 [==============================] - 38s 4s/step - loss: 0.5873 - acc: 0.6734 - val_loss: 0.7564 - val_acc: 0.5916
Epoch 18/25
10/10 [==============================] - 39s 4s/step - loss: 0.6002 - acc: 0.6953 - val_loss: 0.7271 - val_acc: 0.5928
Epoch 19/25
10/10 [==============================] - 39s 4s/step - loss: 0.5872 - acc: 0.6969 - val_loss: 0.6772 - val_acc: 0.6319
Epoch 20/25
10/10 [==============================] - 39s 4s/step - loss: 0.5935 - acc: 0.6984 - val_loss: 0.6791 - val_acc: 0.6142
Epoch 21/25
10/10 [==============================] - 39s 4s/step - loss: 0.5937 - acc: 0.6781 - val_loss: 0.6516 - val_acc: 0.6332
Epoch 22/25
10/10 [==============================] - 41s 4s/step - loss: 0.5993 - acc: 0.6813 - val_loss: 0.6744 - val_acc: 0.6189
Epoch 23/25
10/10 [==============================] - 41s 4s/step - loss: 0.5921 - acc: 0.6859 - val_loss: 0.7019 - val_acc: 0.6236
Epoch 24/25
10/10 [==============================] - 38s 4s/step - loss: 0.6078 - acc: 0.6797 - val_loss: 0.6694 - val_acc: 0.6396
Epoch 25/25
10/10 [==============================] - 39s 4s/step - loss: 0.5801 - acc: 0.7016 - val_loss: 0.6747 - val_acc: 0.6263
```

Visualize Accuracy and Loss:

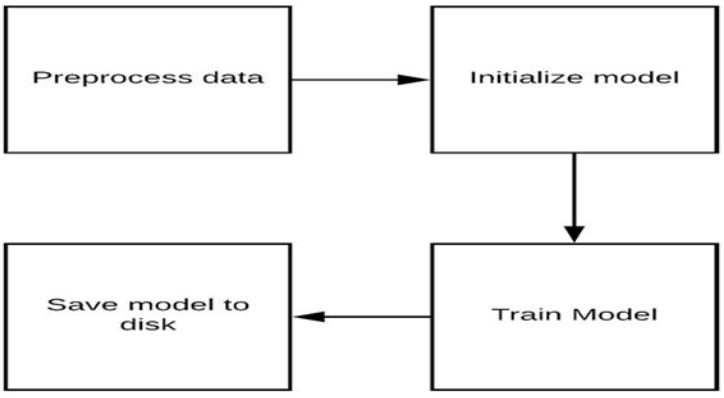
```
<Figure size 432x288 with 0 Axes>
```






```
<Figure size 432x288 with 0 Axes>
```

Saving the model:



The first thing that we'll have to do if we wish to load our Keras model is adding a few extra imports. Firstly, add load_model to your tensorflow.keras.models import
Also make sure to import numpy, as we'll need to compute an argmax value for our Softmax activated model prediction later

HTML code:

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Malaria Detecting System</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet"
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Candal' rel='stylesheet'>
    <style>
    .bg-dark {
        background-color: #fa2f20!important;
    }
    #result {
        color: #0a1c4ed1;
    }
    </style>
</head>

<body>
    <nav class="navbar navbar-dark bg-dark">
        <div class="container">

            <a class="navbar-brand" style="text-align: center; font-weight:bold ; font-family:
          </div>
    </nav>
    <div class="container">
        <div id="content" style="margin-top:2em">
        <div class="container">
          <div class="row">
            <div class="col-sm-6 bd" >
              <h3>Malaria Detection</h3>
              <br>
              <p>Malaria remains a major burden on global health, with roughly 200 million case
                <img src="https://media.gettyimages.com/photos/mosquito-on-water-picture-id157
            </div>
            <div class="col-sm-6">
```

BuildingApplication :

```
import numpy as np
import os
from keras.models import load_model
from keras.preprocessing import image
import tensorflow as tf
global graph
graph = tf.get_default_graph()
from flask import Flask , request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer

app = Flask(__name__)
model = load_model(r"E:\Smart Bridge intern\project\CNN-Malaria\Model\Detecting_Malaria_cnn.h

@app.route('/')
def index():
    return render_template('base.html')

@app.route('/predict',methods = ['GET','POST'])
def upload():
    if request.method == 'POST':
        f = request.files['image']
        print("current path")
        basepath = os.path.dirname(__file__)
        print("current path", basepath)
        filepath = os.path.join(basepath,'uploads',f.filename)
        print("upload folder is ", filepath)
        f.save(filepath)

        img = image.load_img(filepath,target_size = (64,64))
        x = image.img_to_array(img)
        x = np.expand_dims(x,axis =0)

        with graph.as_default():
            preds = model.predict_classes(x)
            4

            print("prediction",preds)

        index = ['Parasitized','Uninfected']

```

Model Prediction:

In [41]:
```
model=load_model('Detecting_Malaria_cnn.h5')
```

In [42]:
```
from skimage.transform import resize
```

In [43]:
```
def detect(frame):
    try:
        img= resize(frame,(64,64))
        img= np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img= img/255.0
        prediction= model.predict(img)
        print(prediction)
        print(model.predict_classes(img))
    except AttributeError:
        print("Shape not found")
```

In [44]:
```
frame= cv2.imread(r'E:\Smart Bridge intern\project\cell image.jpg')
data= detect(frame)
```

```
[[0.1218617]]
[[0]]
```

In [45]:
```
frame= cv2.imread(r'E:\Smart Bridge intern\project\unin.png')
data= detect(frame)
```

```
[[0.61681247]]
[[1]]
```

- **Flask** is a web structure .It is used to develop a web application using python. It is a flexible framework. Using Flask we can develop some basic web applications easily
- **HTML** is a hypertext markup language used to render webpages. So in our application we have used templates for the frontend to render HTML which will be displayed in the user’s browser

Model Output pictures: