Malaria Classification Project Report

Introduction

This project focuses on the development of a deep learning model for the classification of malaria cell images. Malaria, a life-threatening disease caused by Plasmodium parasites, is a major global health concern. In 202 alone, there were approximately 247 million cases of malaria worldwide, leading to around 619 000 deaths, with the majority occurring in sub-Saharan Africa. - WHO

Objective

The goal of this project is to create a model capable of distinguishing between two key classes:

- **Uninfected Cells:** Healthy red blood cells without malaria infection.
- **Parasitized Cells:** Red blood cells infected with Plasmodium parasites.

Dataset

Link - https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria/data

The dataset comprises a collection of images depicting both uninfected and parasitized cells. Sourced from the official NIH Malaria Datasets, it contains a total of 27,558 images.

Methodology

We will leverage deep learning techniques, specifically Convolutional Neural Networks (CNNs), to train a model that can automatically learn and differentiate between features associated with infected and uninfected cells. The model will be trained on a labeled dataset and evaluated on a separate test set to assess its classification performance.

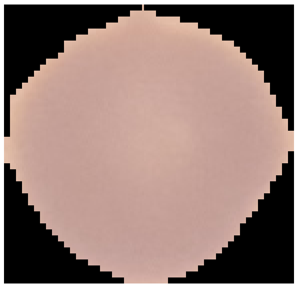
Significance

Automated malaria cell classification has the potential to expedite the diagnostic process, especially in regions with limited access to healthcare resources. The successful implementation of this model could contribute to more efficient and accurate malaria diagnosis, aiding healthcare professionals in providing timely and targeted interventions.

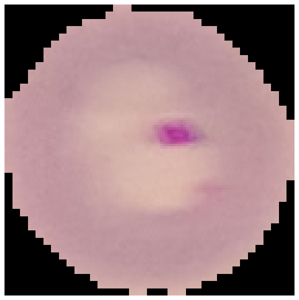
Let's embark on this journey to develop a robust deep learning model for malaria cell classification!

Sample Images

Uninfected Cell



Parasitized Cell



Workout Steps:

Import requirement Libraries

From Tensorflow
- Layers
- Models
- Preprocessing
  - Image
- Application
  - VGG19
  - Resnet50

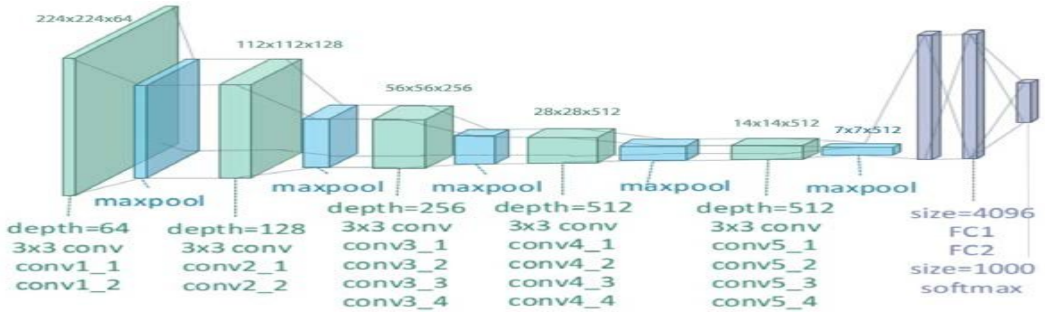
Tensorflow version - 2.14.0

WHY VGG19?

VGG-19 is a convolutional neural network that is 19 layers deep. We can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories.

➔ faster training speed
➔ fewer training samples per time
➔ higher accuracy

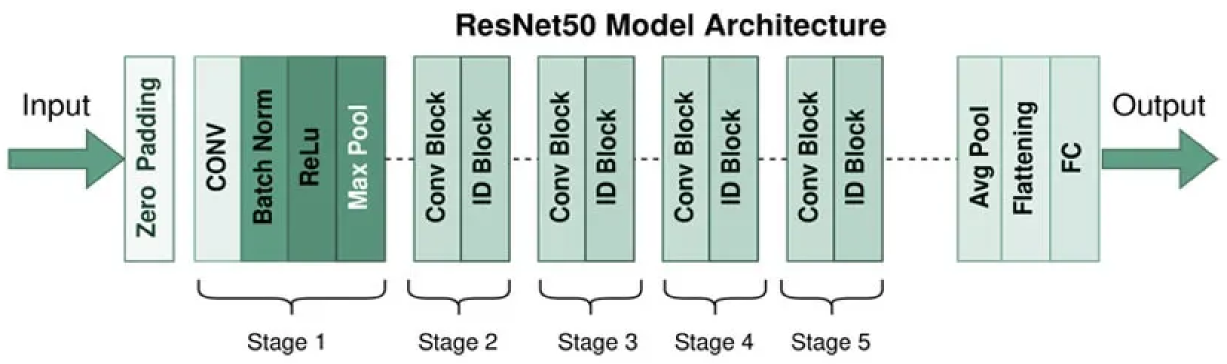
Architecture dig conv - convolution FC - Fully connected



WHY Resnet50?

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the neural network trained on more than a million images from the ImageNet database.

- ➔ without running into the vanishing gradient problem
- ➔ using the concept of shortcut connections.



VGG parameters:

```
VGG19(weights='imagenet', input_shape=IMAGE_SIZE + [3], include_top
=False)
```

Make Constant image size as

- hight - 225px
- width - 225px

Depth of the image RGB chanel by giving [3]
include_top =False, Remove first in vgg arch
Imagenet - visual object recognition

Then take final layer output values, and layer.trainable = False because no need to train layers again.

Set training folder path [‘dataset/train/Parasite’, ‘dataset/train/Uninfected’]

VGG19 layer values need to be flatten here because of result 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector.

Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer. We will passing (length of folder)output layer.
Output have **2 nodes** so im using **‘softmax’** function activation.
Sigmoid function used only binary (1 node output)

From scratch creating Sequential model

Optimization method

2nodes - categorical_crossentropy
1node - binary_crossentropy

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metric
s=['accuracy'])
```

ImageDataGenerator to import the images from the dataset.
use the library read the image form dataset

Apply image augmentation on both train and test data (process of making or becoming greater in size or amount.)Rescaling(size), shear_image(angle), zooming, flipping image(horizandal, vertical)

Based augmentation dataset variable read image data from dataset folder and assign into training_variable.

Training

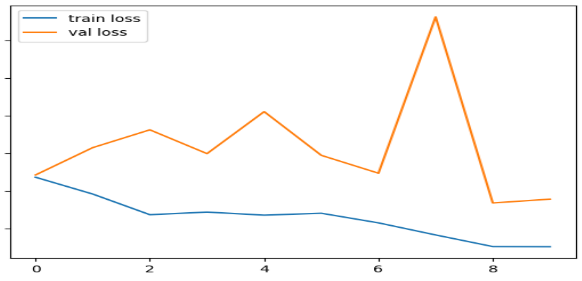
This model training, im passing the parameters to build a model

```
r = model.fit(                          #function call from lib
  Training_set,                         #train_data
  validation_data=test_set,             #validate with test_data
  epochs=10,                            #epochs batchs
  steps_per_epoch=len(training_set), #epochs_steps
  validation_steps=len(test_set)    #valid_steps
)
```

The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset. One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters.

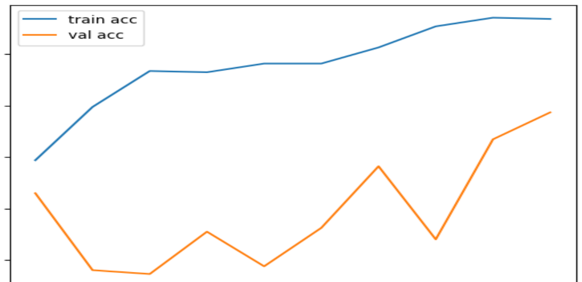
Plot values
Loss:

Trian_loss:	reduced
Valadiating_loss:	reduced



Accuracy:

Trian_accuracy:	increaded



Valadiating_accurasy: increased

Saved model!!!

Deploy

Here im using Flask server.

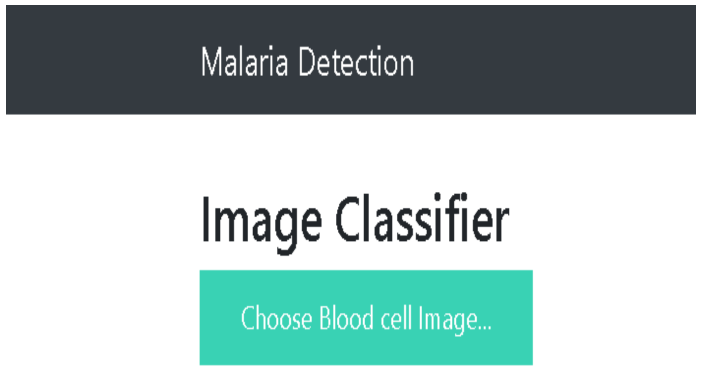
app.py is main file it have the flask server

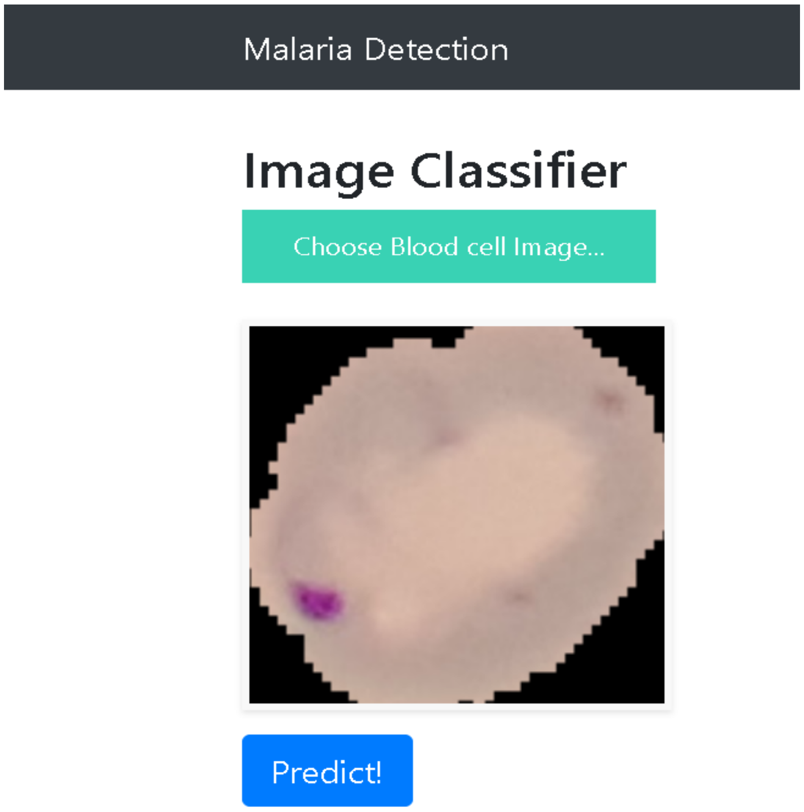
- Home page api
- Model_predict API call my model
- Prediction function

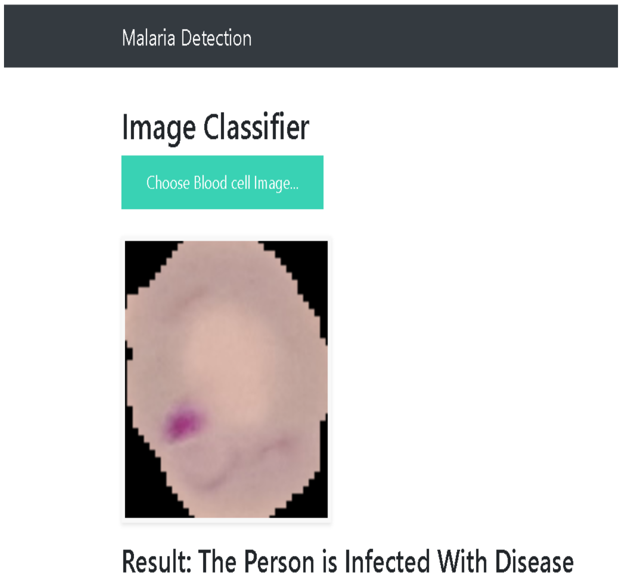
malaria_detection.py have my model script

- Image Oggumantion
- Making training and test function
- Model training and validation

View pages [3 steps]







Result: The Person is Infected With Disease