MALARIA INFECTED CELL CLASSIFICATION AND DETECTTION

Dataset Description

- The dataset consists of 27,558 cell images; 13,780 images of Parasitized and uninfected cells each and is taken from the official Kaggle website.
- Link: https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria
- We divide the dataset into two parts training_data and validation_data inside of this two folders again two folders with Parasitized and Uninfected.validation_data is $1/4^{th}$ of the dataset where as training_data is $3/4^{th}$ of the dataset
- Fig 1 shows visualization of the dataset used.

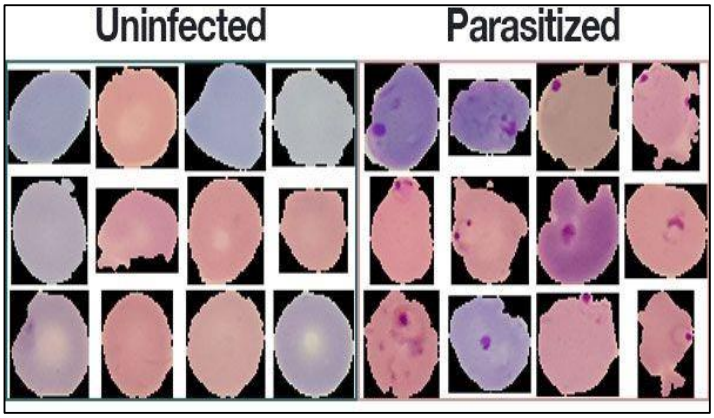

Fig 1. Dataset of uninfected and Malaria infected cells. (Source: Google Images)

Cnn.py

The CNN.py code defines and trains a Convolutional Neural Network (CNN) for image classification using the Keras library. Here's a brief explanation of each part:

Model Architecture:

The CNN model is built using the Sequential API of Keras.

It consists of convolutional layers with ReLU activation, followed by max-pooling layers for feature extraction.

Global Average Pooling is used to reduce the spatial dimensions of the feature maps.

The output layer has two units (binary classification) with softmax activation.

Importing Libraries:

The code imports necessary libraries, including TensorFlow, Keras, and data preprocessing tools.

Initializing the CNN:

A Sequential model is created using Keras, representing a linear stack of layers.

Two initial convolutional layers with ReLU activation are added, followed by a max-pooling layer.

Two more sets of convolutional layers and max-pooling layers are added for feature extraction.

A global average pooling layer is added to reduce the spatial dimensions of the feature maps.

Adding Fully Connected Layers:

A dense layer with a softmax activation function is added for classification. The output has two units, representing the classes "Parasitized" and "Uninfected."

Compiling the Model:

The model is compiled using the Adam optimizer and categorical cross-entropy loss function.

Data Augmentation and Loading:

Image data generators are created for both training and testing data.

Training images undergo data augmentation (e.g., shear, zoom, horizontal flip).

The flow_from_directory function is used to load the training and testing datasets.

Training the CNN:

The fit_generator function is used to train the model on the training dataset.

The training process involves 32 epochs and 250 steps per epoch.

Saving the Model:

The trained model is saved as a Keras model file (malaria-cnn-v1.keras).

Test Accuracy Function:

A function test_accuracy is defined to evaluate the accuracy of the trained model on a test set.

It iterates through batches of the test set, makes predictions, and compares them to the ground truth.

Overall, this code demonstrates the process of building, training, and saving a CNN for the classification of malaria-infected and uninfected cells using a provided dataset.

Hotspotdetection.py

In this hotspotdetection.py, Streamlit app combines image classification and hotspot detection using a pre-trained model. Here's a brief explanation of each part:

Haar Cascade Classifier:

A Haar Cascade Classifier is loaded to identify regions of interest (ROI) in the image.

Load Trained Model:

The pre-trained CNN model for malaria classification (malaria-cnn-v1.keras) is loaded using Keras.

Class Names Dictionary:

A dictionary (class_names) is created to map class indices to class names for interpretation of the model predictions.

Hotspot Detection Function:

The detect_hotspots function takes an image array as input and performs the following:

Converts the image to the HSV color space.

Defines a range for magenta color in HSV.

Creates a magenta mask using cv2.inRange.

Performs morphological operations to clean up the binary image.

Finds contours in the magenta mask.

Draws rectangles around the identified contours.

Streamlit UI:

The Streamlit app is initialized with a title and a file uploader for image input.

Image Processing and Classification:

When an image is uploaded, it is displayed, and its size is adjusted to fit the model input size (128x128).

The image is converted to a numpy array of type uint8 and normalized to the range [0, 1].

The pre-trained model predicts the class of the image (Parasitized or Uninfected).

The predicted class is displayed as a success message.

Hotspot Detection and Display:

The detect_hotspots function is called to identify and highlight magenta-colored hotspots in the image.

The image with hotspots is displayed using Streamlit.

This app allows users to upload an image, performs malaria classification using a pre-trained CNN, and identifies magenta-colored hotspots within the image. The Haar Cascade Classifier provides additional capabilities for detecting specific regions of interest.

The below images gives Image classification and Hotspot Detection app:

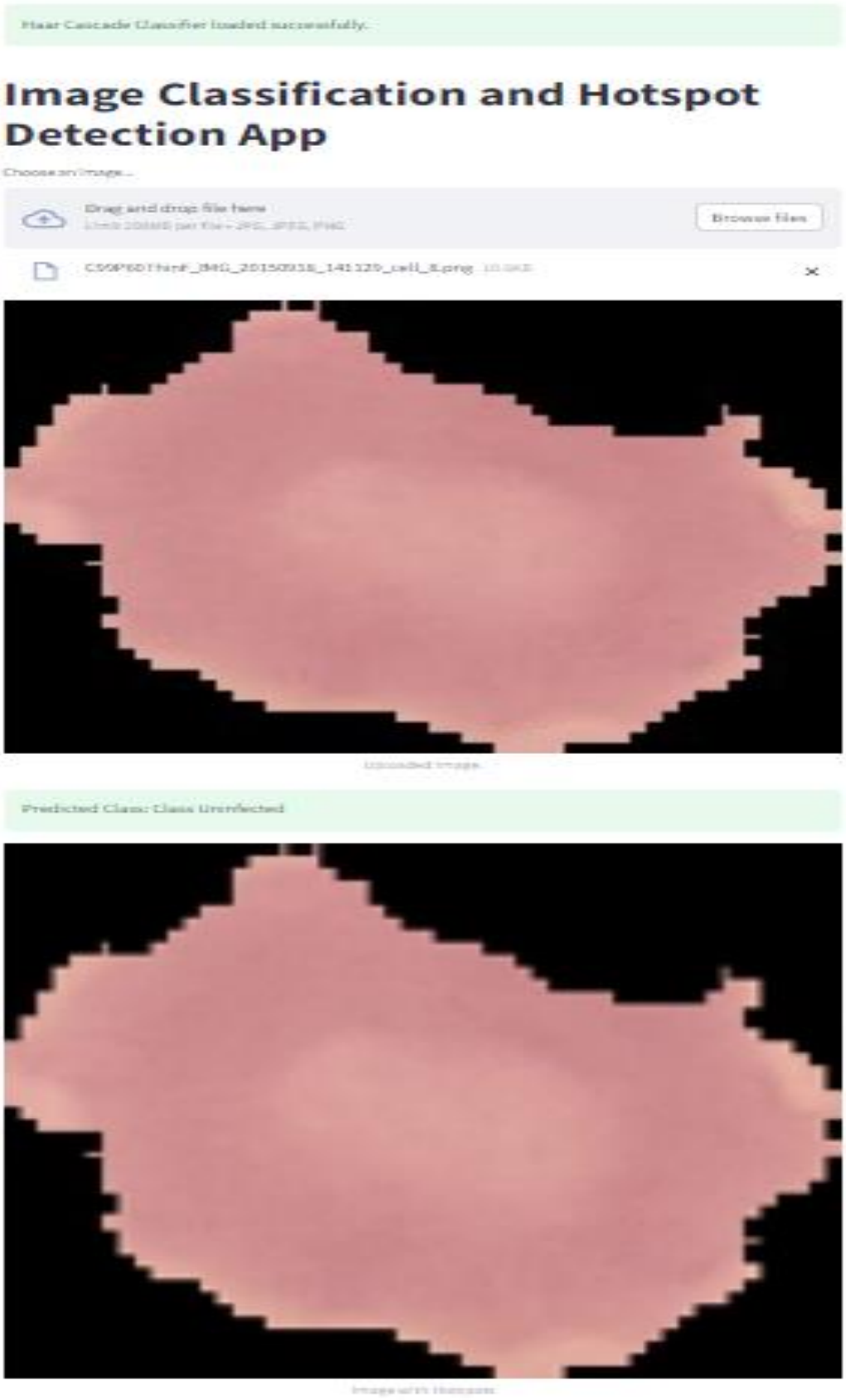

Fig2:Image classification and detection of hotspot -Uninfected

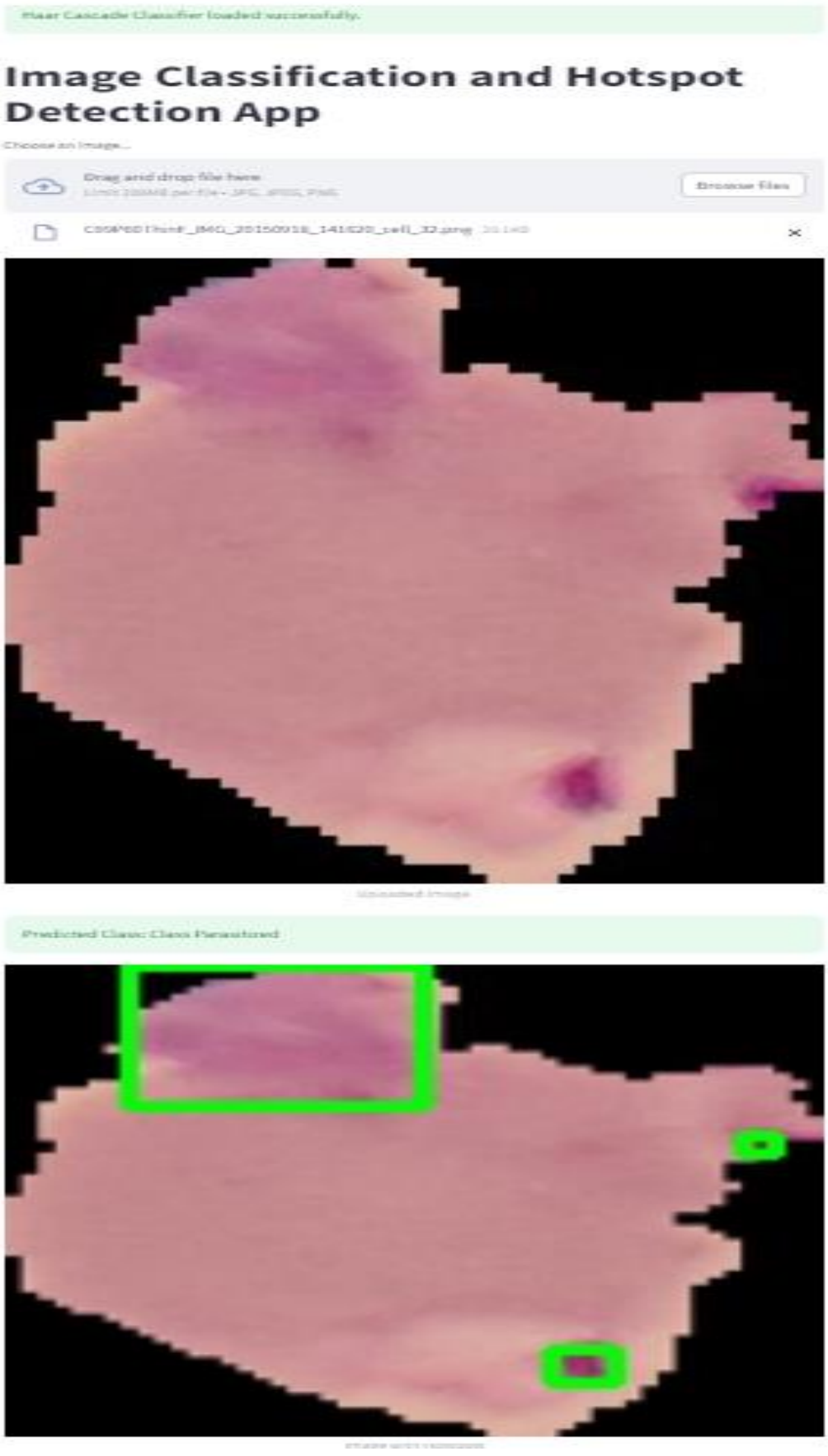

Fig3:Image classification and detection of hotspot -Parasitized