AI Enabled
Weed Recognition System

Submitted by:-

U.TALIN SAI

V S S R RAJA VARMA

ANAND MOULI

SYED WASIM

1.Introduction

1.1 Overview

Weeds are considered to be one of the biggest problems in agronomy. Their adverse effect is widely known—they reduce crop yields, serve as hosts for crop diseases and also produce toxic substances. Manual removal of weeds is labour-intensive, while the use of chemicals has long-term environmental consequences. Currently, approximately 23,000 tonnes of chemical herbicide at a cost of around £400million are used annually in the UK on weed prevention (Marchant, 1996). Reducing this quantity would potentially lead to reduced herbicide residues in water, food crops and the environment. One possible method of achieving this is to improve the selectivity with which herbicidal agents are applied to fields, by automated visual discrimination between crops and weeds for spot application of herbicide .Computer vision has been shown to provide a viable option in inspection of agricultural products, particularly when colour and shape need to be analyzed at high speed (Batchelor and Searcy, 1989).

1.2 Purpose

Weed detection systems are important solutions to one of the existing agricultural problems—unmechanized weed control. Weed detection also helps provide a means of reducing or eliminating herbicide use, mitigating agricultural environmental and health impact, and improving sustainability.

2. Literature Survey

2.1 Existing Problem

Weed picking is one of the laborious job in fields. Weeds are the plants growing in a wrong place which compete with crop for water, light, nutrients and space, causing reduction in yield and effective use of machinery and can cause a disturbance in agriculture. Weeds can also host pests and diseases that can spread to cultivated crops.In olden days weed detection was done by employing some people, especially for that purpose. They will detect the weed by checking each and every place of the field. Then they will pluck them out manually using their hands.

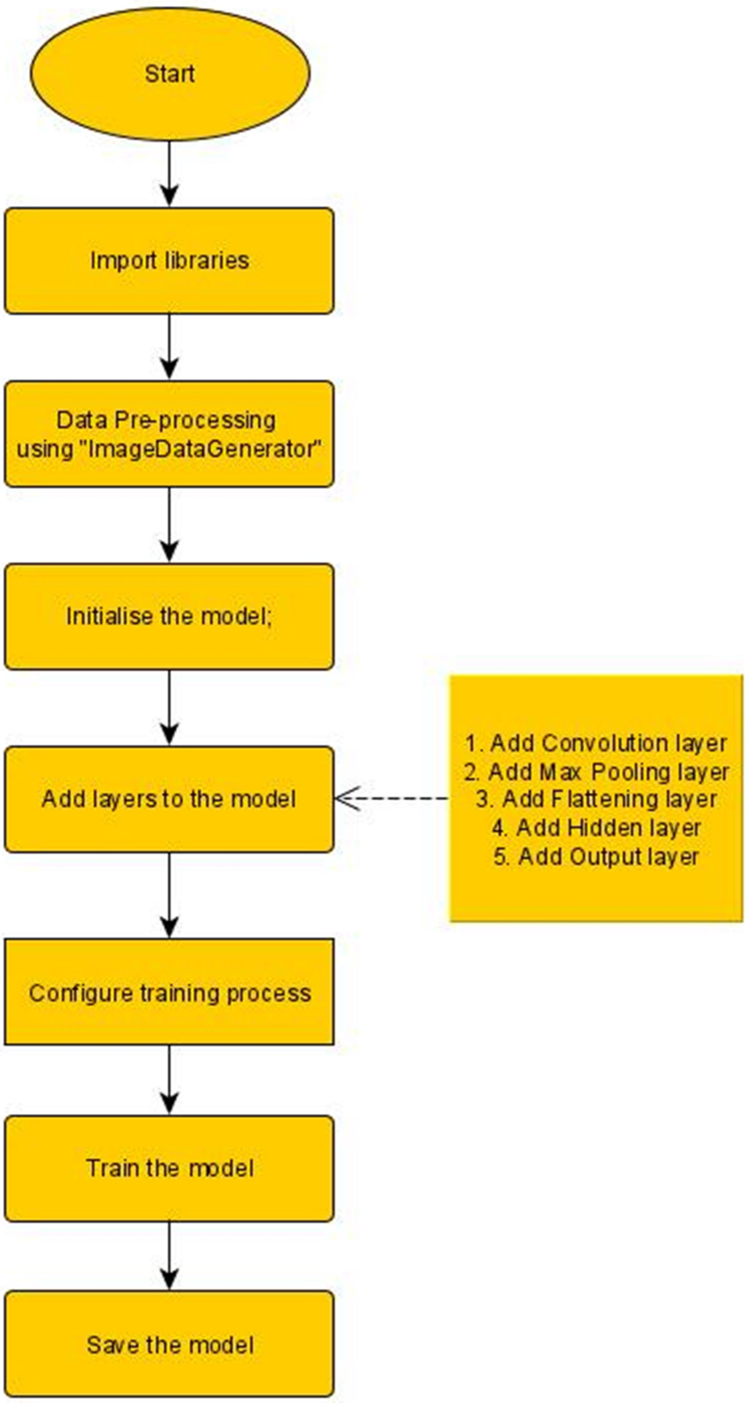
2.2 Proposed Solution

We are Proposing a solution in the device is integrated with camera and there will be a live video streaming in that it will detect the weed in the crop by using image processing. This system will distinguish the crop and weed. Our system will use a Convolution neural network algorithm to extract the features from image and train them by using neural network.

3. Theoretical Analysis

3.1 Block Diagram

The following is the block diagram of our proposed solution of using the Conolution Neural Networks to detect the weed whether present in the soil or not.



3.2 Hardware/Software Designing

Hardware Requirements:- Laptop/Desktop with webcam

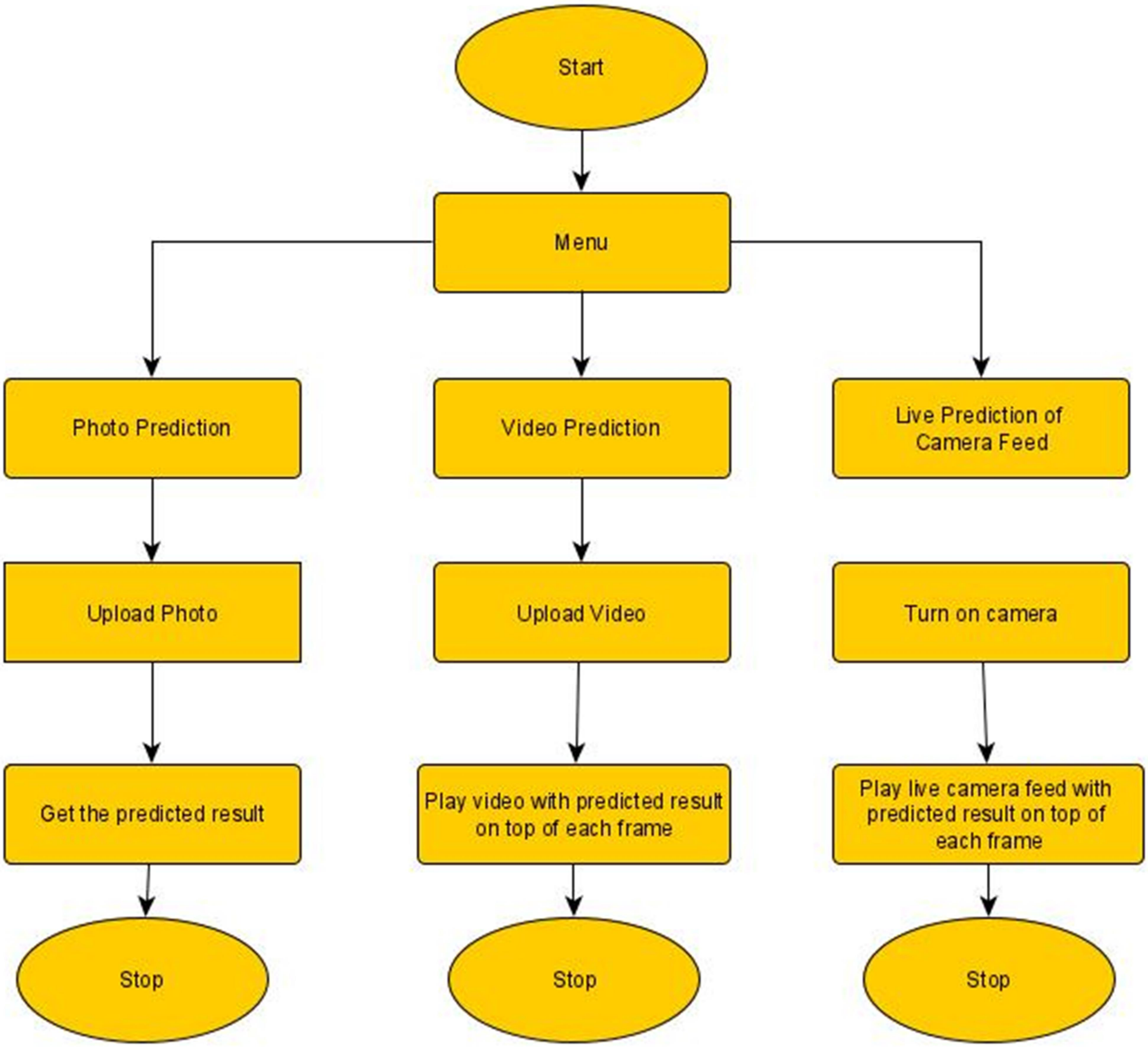
Software Requirements:-

- Python
- Keras
- Spyder
- Tensorflow
- Jupyter Notebook

4. Experimental Investigations

- The crops in the field very much look alike, to achieve a practically good accuracy larger dataset is needed.
- The dataset need to be more distributed and unbiased to achieve to a accurate model.
- Edge Detection is quite harder with the crops dataset because of similarity of the crops with their background.

5. Flowchart


Start
Menu
Photo Prediction
Video Prediction
Live Prediction of Camera Feed
Upload Photo
Upload Video
Turn on camera
Get the predicted result
Play video with predicted result on top of each frame
Play live camera feed with predicted result on top of each frame
Stop
Stop
Stop

6. Result

The following images show the screenshots of our application of the AI Based Weed Recognition System

Main Menu

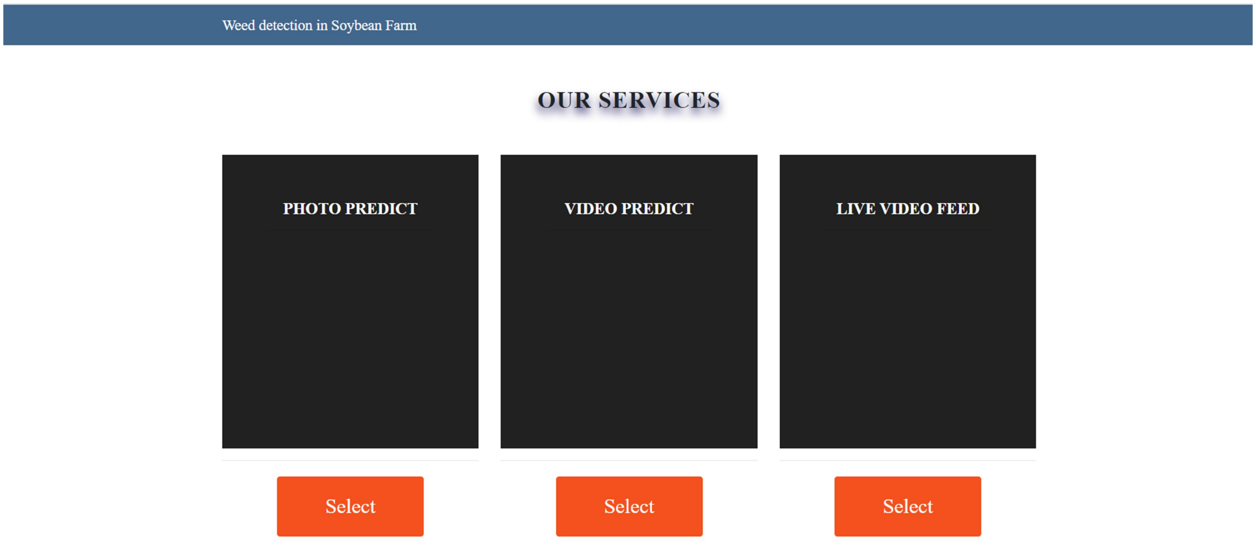
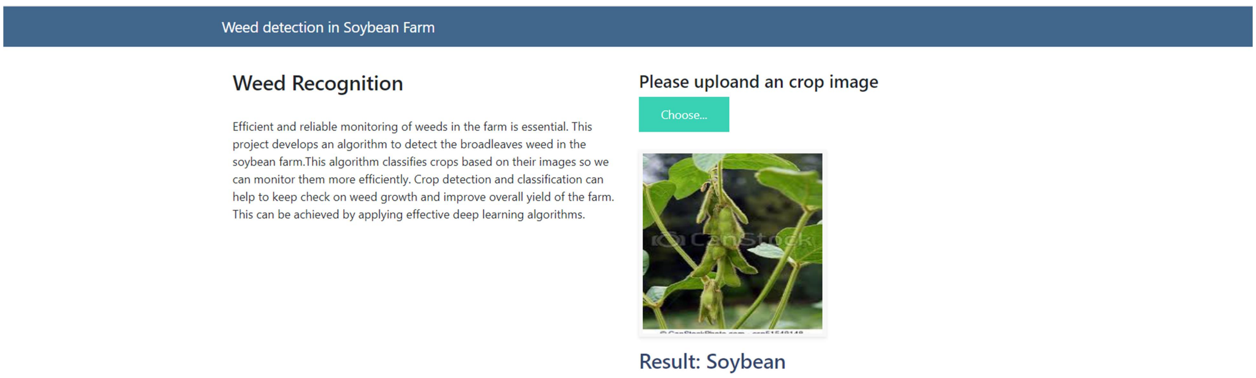
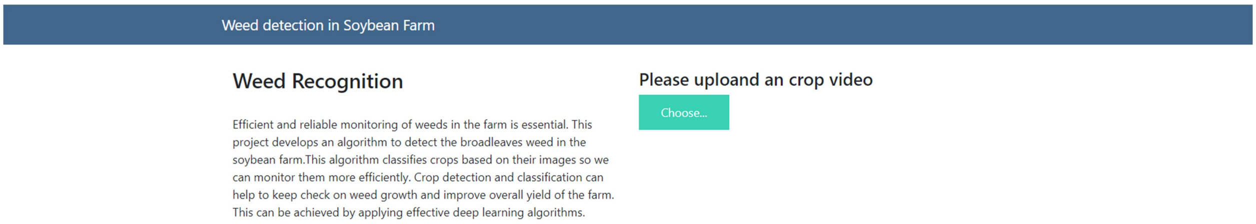


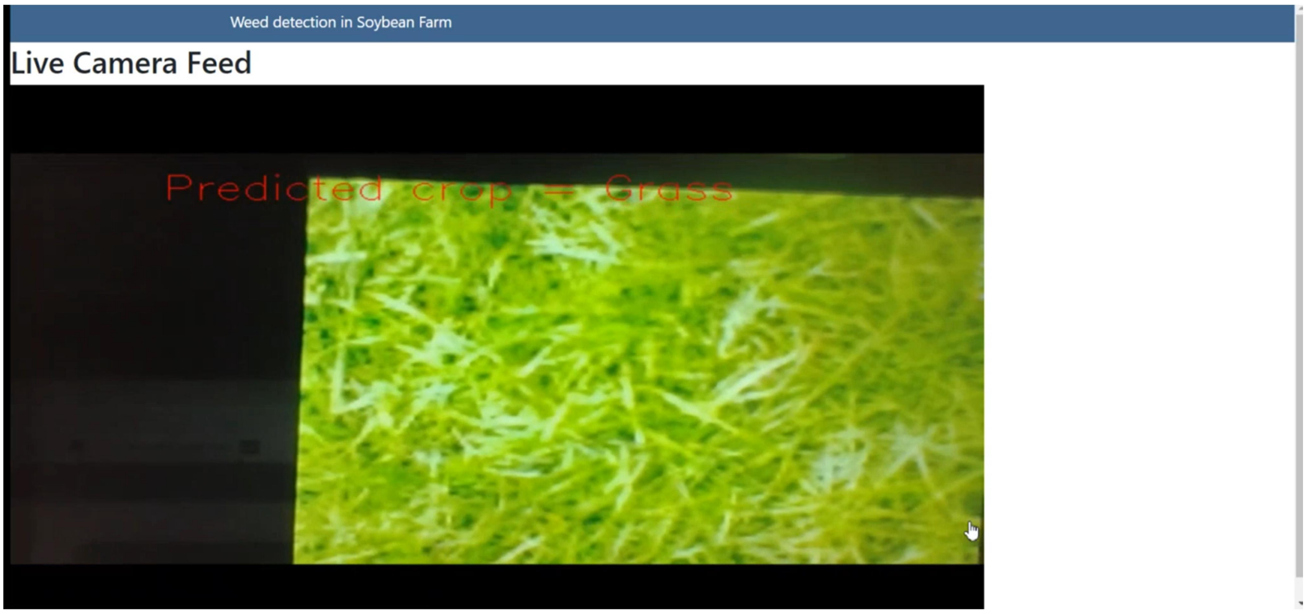
Photo Predictor



Video Predictor



Live Camera Feed Predictor



7.Advantages and Disadvantages

Advantages	Disadvantages
Easy Model building with less formal statistical knowledge required	Sharing an existing CNN model is difficult
Capable of capturing non linearties between predictors and outcome	Prone to overfitting due to the complexity of model structure

8.Applications

The AI enabled weed recognition system using Convolution neural networks is currently being practiced at many agricultural sectors.

This application of CNN in weed recognition system helps many farmers as it is time saving, less work compared to the previous practices being done.

9. Conclusion

We would like to conclude that the developed model if trained with more data and better algorithms using more computation power can be deployed in a real world to help farmers and decrease the use of excessive herbicides , mitigating agricultural, environmental and health impact, and improving sustainability

10. Future Scope

We can integrate this model to robot where the robot recognise the weed and pick automatically.

11. Bibilography

- https://www.kaggle.com/fpeccia/weed-detection-in-soybean-crops
- https://thesmartbridge.com/documents/spsaimldocs/CNNcollection.pdf
- https://thesmartbridge.com/documents/spsaimldocs/CNNprep.pdf
- https://thesmartbridge.com/documents/spsaimldocs/CNNflow.pdf
- https://opencv-python tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html

12.Appendix

Model Training

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

x_train = train_datagen.flow_from_directory(r"D:\Weed Project\dataset\trainset",target_size=(64,64),batch_size=32,class_mode='categorical')
x_test = train_datagen.flow_from_directory(r"D:\Weed Project\dataset\testset",target_size=(64,64),batch_size=32,class_mode='categorical')
```

Found 10837 images belonging to 4 classes.
Found 4499 images belonging to 4 classes.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten

model = Sequential()
model.add(Conv2D(32,3,3,input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(output_dim=128,activation='relu',init='random_uniform'))
model.add(Dense(output_dim=4,activation='sigmoid',init='random_uniform'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

print(x_train.class_indices)
```

{'broadleaf': 0, 'grass': 1, 'soil': 2, 'soybean': 3}

```
model.fit_generator(x_train,steps_per_epoch = 339,epochs=25,validation_data=
x_test,validation_steps = 141 )
```

```
Epoch 1/25
339/339 [==============================] - 55s 163ms/step - loss: 0.
5572 - accuracy: 0.7720 - val_loss: 0.9070 - val_accuracy: 0.8371
Epoch 2/25
339/339 [==============================] - 56s 164ms/step - loss: 0.
3995 - accuracy: 0.8477 - val_loss: 0.3365 - val_accuracy: 0.8860
Epoch 3/25
339/339 [==============================] - 61s 179ms/step - loss: 0.
3318 - accuracy: 0.8761 - val_loss: 0.1398 - val_accuracy: 0.8904
Epoch 4/25
339/339 [==============================] - 62s 182ms/step - loss: 0.
2766 - accuracy: 0.8955 - val_loss: 0.1585 - val_accuracy: 0.8969
Epoch 5/25
339/339 [==============================] - 60s 177ms/step - loss: 0.
2572 - accuracy: 0.9013 - val_loss: 0.3267 - val_accuracy: 0.8864
Epoch 6/25
339/339 [==============================] - 61s 180ms/step - loss: 0.
2416 - accuracy: 0.9051 - val_loss: 0.5447 - val_accuracy: 0.8128
Epoch 7/25
339/339 [==============================] - 64s 189ms/step - loss: 0.
2280 - accuracy: 0.9115 - val_loss: 0.3420 - val_accuracy: 0.8675
Epoch 8/25
339/339 [==============================] - 65s 192ms/step - loss: 0.
2288 - accuracy: 0.9143 - val_loss: 0.1986 - val_accuracy: 0.9235
Epoch 9/25
339/339 [==============================] - 63s 186ms/step - loss: 0.
2053 - accuracy: 0.9228 - val_loss: 0.3976 - val_accuracy: 0.9182
Epoch 10/25
339/339 [==============================] - 65s 190ms/step - loss: 0.
2061 - accuracy: 0.9195 - val_loss: 0.3038 - val_accuracy: 0.9164
Epoch 11/25
339/339 [==============================] - 62s 182ms/step - loss: 0.
1861 - accuracy: 0.9290 - val_loss: 0.0540 - val_accuracy: 0.9242
Epoch 12/25
339/339 [==============================] - 61s 179ms/step - loss: 0.
1742 - accuracy: 0.9318 - val_loss: 0.0575 - val_accuracy: 0.9233
Epoch 13/25
339/339 [==============================] - 61s 179ms/step - loss: 0.
1749 - accuracy: 0.9326 - val_loss: 0.0283 - val_accuracy: 0.9262
```

```
Epoch 14/25
339/339 [==============================] - 62s 183ms/step - loss: 0.
1616 - accuracy: 0.9383 - val_loss: 0.2013 - val_accuracy: 0.9222
Epoch 15/25
339/339 [==============================] - 62s 184ms/step - loss: 0.
1590 - accuracy: 0.9371 - val_loss: 0.0789 - val_accuracy: 0.9235
Epoch 16/25
339/339 [==============================] - 63s 187ms/step - loss: 0.
1585 - accuracy: 0.9404 - val_loss: 0.1998 - val_accuracy: 0.9224
Epoch 17/25
339/339 [==============================] - 63s 186ms/step - loss: 0.
1600 - accuracy: 0.9372 - val_loss: 0.2660 - val_accuracy: 0.9284
Epoch 18/25
339/339 [==============================] - 64s 189ms/step - loss: 0.
1485 - accuracy: 0.9450 - val_loss: 0.3198 - val_accuracy: 0.9231
Epoch 19/25
339/339 [==============================] - 62s 183ms/step - loss: 0.
1450 - accuracy: 0.9423 - val_loss: 0.2404 - val_accuracy: 0.9262
Epoch 20/25
339/339 [==============================] - 63s 186ms/step - loss: 0.
1457 - accuracy: 0.9428 - val_loss: 0.0649 - val_accuracy: 0.9178
Epoch 21/25
339/339 [==============================] - 62s 184ms/step - loss: 0.
1329 - accuracy: 0.9492 - val_loss: 0.1291 - val_accuracy: 0.9186
Epoch 22/25
339/339 [==============================] - 63s 186ms/step - loss: 0.
1270 - accuracy: 0.9516 - val_loss: 0.2807 - val_accuracy: 0.9255
Epoch 23/25
339/339 [==============================] - 62s 183ms/step - loss: 0.
1203 - accuracy: 0.9549 - val_loss: 0.6384 - val_accuracy: 0.9213
Epoch 24/25
339/339 [==============================] - 63s 185ms/step - loss: 0.
1373 - accuracy: 0.9480 - val_loss: 0.2804 - val_accuracy: 0.9255
Epoch 25/25
339/339 [==============================] - 63s 186ms/step - loss: 0.
1140 - accuracy: 0.9570 - val_loss: 0.0202 - val_accuracy: 0.9240

<keras.callbacks.callbacks.History at 0x2622d522988>
```

```
model.save("weed.h5")
```

Flask app.py

```
import numpy as np
import time
import os
import cv2
import tensorflow as tf
from keras.models import load_model
from keras.preprocessing import image
from flask import Flask , request, render_template, Response
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer

app = Flask(__name__)
model = load_model("weed.h5")

@app.route('/photo',methods = ['GET','POST'])
def index():
    if request.method == 'POST':
        f = request.form["action"]
        if(f=="photo"):
            return render_template('base.html')
        elif(f=="video"):
            return render_template('video.html')
```

```
        elif(f=="live"):
            return render_template('live.html')
@app.route('/')
def options():
    return render_template('index.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)
        preds = model.predict_classes(x)
        4
```

```
        print("prediction",preds)

        index = ['Broadleaf','Grass','Soil','Soybean']

        text = str(index[preds[0]])

    return text




def uploadVideo(filepath):



        video = cv2.VideoCapture(filepath)
        fourcc = cv2.VideoWriter_fourcc(*'MP4V')

        name = ["Broadleaf","Grass","Soil","Soybean"]

        while(video.isOpened()):

            success, frame = video.read()
            if success==True:
                cv2.imwrite("image.jpg",frame)
                img = image.load_img("image.jpg",target_size = (64,64))
```

```
                x = image.img_to_array(img)

                x = np.expand_dims(x,axis=0)

                pred = model.predict_classes(x)

                p = pred[0]

                print(pred)

                cv2.putText(frame, "Predicted crop = "+ str(name[p]), (100,100),
cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)

                #print(frame.shape)

                #cv2.imshow("image",frame)

                frame = cv2.imencode('.jpg', frame)[1].tobytes()

                yield (b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n')


                if(cv2.waitKey(1) & 0xFF == ord('a')):

                    break

            else:

                break





@app.route('/video1',methods=['GET','POST'])

def video():

    if request.method == 'POST':

        f = request.files['video']

        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)

        return Response(uploadVideo(filepath),mimetype='multipart/x-mixed-
replace; boundary=frame')




def gen():
    """Video streaming generator function."""
    cap = cv2.VideoCapture(0)

    # Read until video is completed
    name = ["Broadleaf","Grass","Soil","Soybean"]
    while(cap.isOpened()):
      # Capture frame-by-frame
        ret, frame = cap.read()
        if ret == True:
            cv2.imwrite("image.jpg",frame)
            img = image.load_img("image.jpg",target_size = (64,64))
            x = image.img_to_array(img)
```

```
            x = np.expand_dims(x,axis=0)
            pred = model.predict_classes(x)
            p = pred[0]
            print(pred)
            cv2.putText(frame, "Predicted crop = "+ str(name[p]), (100,100),
cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),1)
            frame = cv2.resize(frame, (0,0), fx=2, fy=1.5)
            frame = cv2.imencode('.jpg', frame)[1].tobytes()
            yield (b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
            time.sleep(0.1)
        else:
            break


@app.route('/video_feed')
def video_feed():
    """Video streaming route. Put this in the src attribute of an img tag."""
    return Response(gen(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
    app.run(debug = True, threaded = False)
```