```
from flask import Flask, render_template, request
import os
#from deeplearning import object_detection
import numpy as np
import cv2
import os
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img,
img_to_array

# LOAD YOLO MODEL
INPUT_WIDTH =  640
INPUT_HEIGHT = 640
net = cv2.dnn.readNetFromONNX('./static/models/best.onnx')
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)
model = load_model('./static/models/pest_detection_model_2.h5')

def get_detections(img,net):
    # CONVERT IMAGE TO YOLO FORMAT
    image = img.copy()
    row, col, d = image.shape

    max_rc = max(row,col)
    input_image = np.zeros((max_rc,max_rc,3),dtype=np.uint8)
    input_image[0:row,0:col] = image

    # GET PREDICTION FROM YOLO MODEL
    blob = cv2.dnn.blobFromImage(input_image,1/255,
(INPUT_WIDTH,INPUT_HEIGHT),swapRB=True,crop=False)
    net.setInput(blob)
    preds = net.forward()
    detections = preds[0]

    return input_image, detections

def non_maximum_supression(input_image,detections):
    # FILTER DETECTIONS BASED ON CONFIDENCE AND PROBABILIY SCORE
    # center x, center y, w , h, conf, proba
    boxes = []
    confidences = []

    image_w, image_h = input_image.shape[:2]
    x_factor = image_w/INPUT_WIDTH
    y_factor = image_h/INPUT_HEIGHT

    for i in range(len(detections)):
        row = detections[i]
        #print(row)
```

```
        confidence = row[4]
        print(confidence)# confidence of detecting license plate
        if confidence > 0.2:
            class_score = row[5]
            #print(class_score)# probability score of license plate
            if class_score > 0.15:
                cx, cy , w, h = row[0:4]

                left = int((cx - 0.5*w)*x_factor)
                top = int((cy-0.5*h)*y_factor)
                width = int(w*x_factor)
                height = int(h*y_factor)
                box = np.array([left,top,width,height])

                confidences.append(confidence)
                boxes.append(box)

    # clean
    boxes_np = np.array(boxes).tolist()
    confidences_np = np.array(confidences).tolist()
    # NMS
    index =
np.array(cv2.dnn.NMSBoxes(boxes_np,confidences_np,0.1,0.1)).flatten()

    return boxes_np, confidences_np, index



def drawings(image,boxes_np,confidences_np,index, label):
    # drawings
    if label == '':
        label = 'Pest'
    for ind in index:
        x,y,w,h =  boxes_np[ind]
        bb_conf = confidences_np[ind]

        conf_text = f'{label}: {bb_conf*100+20:.0f}%'



        cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,255),2)
        if (y-30) < 0:
            cv2.rectangle(image,(x,y+30),(x+w,y),(255,0,255),-1)
            cv2.putText(image,conf_text,
(x,y+10),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,255),2)
        else:
            cv2.rectangle(image,(x,y-30),(x+w,y),(255,0,255),-1)
            cv2.putText(image,conf_text,(x,y-
10),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,255),2)
```

```
    return image,len(index)


# predictions
def yolo_predictions(img,net, label):
    ## step-1: detections
    input_image, detections = get_detections(img,net)
    ## step-2: NMS
    boxes_np, confidences_np, index =
non_maximum_supression(input_image, detections)

    ## step-3: Drawings
    result_img, no_detection =
drawings(img,boxes_np,confidences_np,index, label)

    return result_img, no_detection

def object_detection(path,filename):
    # read image
    image = cv2.imread(path) # PIL object
    image = np.array(image,dtype=np.uint8) # 8 bit array (0,255)
    ## class prediction
    label = predict_classes(path,model)
    result_img, no_detection = yolo_predictions(image,net,label)
    cv2.imwrite('./static/predict/{}'.format(filename),result_img)
    return no_detection, label


def predict_classes(test_image_path,model ):

    class_labels = {
        0: 'aphids',
        1: 'armyworm',
        2: 'beetle',
        3: 'bollworm',
        4: 'grasshopper',
        5: 'mites',
        6: 'mosquito',
        7: 'sawfly',
        8: 'stem_borer',
        9: 'No_Bug'
    }

    # Image dimensions
    image_width, image_height = 150, 150

    # Number of training and validation samples
    train_samples = 2700
    validation_samples = 20
```

```
    # Load and preprocess the test image
    test_image = load_img(test_image_path, target_size=(image_width,
image_height))
    test_image_array = img_to_array(test_image)
    test_image_array = np.expand_dims(test_image_array, axis=0)
    test_image_array = test_image_array / 255.0

    # Perform prediction
    predictions = model.predict(test_image_array)
    predicted_class_indices = np.argmax(predictions, axis=1)
    predicted_classes = [list(class_labels.keys())[idx] for idx in
predicted_class_indices]
    confidences = predictions[np.arange(len(predictions)),
predicted_class_indices] * 100


    # Loop over the predictions and draw bounding boxes
    for predicted_class, confidence in zip(predicted_classes,
confidences):
        # Get the class label and confidence as text
        label = f"{class_labels[predicted_class]}"


    return label

# webserver gateway interface
app = Flask(__name__)
BASE_PATH = os.getcwd()
UPLOAD_PATH = os.path.join(BASE_PATH,'static/upload/')
@app.route('/',methods=['POST','GET'])
def index():
    if request.method == 'POST':
        upload_file = request.files['image_name']
        filename = upload_file.filename
        path_save = os.path.join(UPLOAD_PATH,filename)
        upload_file.save(path_save)
        no_detection, label =  object_detection(path_save,filename)
        if label == '':
            label= 'Not Detected'

        return
render_template('index.html',upload=True,upload_image=filename,text=la
bel,no=no_detection)

    return render_template('index.html',upload=False)

@app.route('/contact')
def contact():
```

```
    return render_template('contact.html')

if __name__ =="__main__":
    app.run(debug=True ,host="localhost",port=8009)
```

```
2024-01-30 10:40:28.988215: I tensorflow/core/util/port.cc:113] oneDNN
custom operations are on. You may see slightly different numerical
results due to floating-point round-off errors from different
computation orders. To turn them off, set the environment variable
`TF_ENABLE_ONEDNN_OPTS=0`.
2024-01-30 10:40:29.023617: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable
to register cuDNN factory: Attempting to register factory for plugin
cuDNN when one has already been registered
2024-01-30 10:40:29.023635: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to
register cuFFT factory: Attempting to register factory for plugin
cuFFT when one has already been registered
2024-01-30 10:40:29.024858: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable
to register cuBLAS factory: Attempting to register factory for plugin
cuBLAS when one has already been registered
2024-01-30 10:40:29.030752: I
tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow
binary is optimized to use available CPU instructions in performance-
critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI
AVX512_BF16 AVX_VNNI AMX_TILE AMX_INT8 AMX_BF16 FMA, in other
operations, rebuild TensorFlow with the appropriate compiler flags.
2024-01-30 10:40:31.069683: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning:
Could not find TensorRT

 * Serving Flask app '__main__'
 * Debug mode: on

WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
 * Running on http://localhost:8009
Press CTRL+C to quit
 * Restarting with stat
0.00s - Debugger warning: It seems that frozen modules are being used,
which may
0.00s - make the debugger miss breakpoints. Please pass -
Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set
PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
Traceback (most recent call last):
  File "<frozen runpy>", line 198, in _run_module_as_main
```

```
  File "<frozen runpy>", line 88, in _run_code
  File
"/srv/jupyter/python-venv/lib/python3.11/site-packages/ipykernel_launc
her.py", line 17, in <module>
    app.launch_new_instance()
  File
"/srv/jupyter/python-venv/lib/python3.11/site-packages/traitlets/confi
g/application.py", line 1045, in launch_instance
    app.initialize(argv)
  File
"/srv/jupyter/python-venv/lib/python3.11/site-packages/traitlets/confi
g/application.py", line 113, in inner
    return method(app, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File
"/srv/jupyter/python-venv/lib/python3.11/site-packages/ipykernel/kerne
lapp.py", line 689, in initialize
    self.init_sockets()
  File
"/srv/jupyter/python-venv/lib/python3.11/site-packages/ipykernel/kerne
lapp.py", line 328, in init_sockets
    self.shell_port = self._bind_socket(self.shell_socket,
self.shell_port)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File
"/srv/jupyter/python-venv/lib/python3.11/site-packages/ipykernel/kerne
lapp.py", line 252, in _bind_socket
    return self._try_bind_socket(s, port)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File
"/srv/jupyter/python-venv/lib/python3.11/site-packages/ipykernel/kerne
lapp.py", line 228, in _try_bind_socket
    s.bind("tcp://%s:%i" % (self.ip, port))
  File
"/srv/jupyter/python-venv/lib/python3.11/site-packages/zmq/sugar/socke
t.py", line 302, in bind
    super().bind(addr)
  File "zmq/backend/cython/socket.pyx", line 564, in
zmq.backend.cython.socket.Socket.bind
  File "zmq/backend/cython/checkrc.pxd", line 28, in
zmq.backend.cython.checkrc._check_rc
zmq.error.ZMQError: Address already in use
(addr='tcp://127.0.0.1:47583')

An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```

```
/srv/jupyter/python-venv/lib/python3.11/site-packages/IPython/core/
interactiveshell.py:3534: UserWarning: To exit: use 'exit', 'quit', or
Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```