COMPUTER VISION : RECOGNIZE OBJECTS IN PHOTOS

Supervisor: DANG TUAN HOANG
Member: NGUYEN TRUONG KHAI

Ha Noi, 10 August 2023

TABLE OF CONTENTS

I. INTRODUCTION

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital image, video and visual inputs and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognizes images. For example, to train a computer to recognize cars, it must be fed vast quantities of car images to learn differences and recognize a car, especially one with no defects.

In this project, I used the image detection model of YOLOv8 ultralytics combined to carry out the project of recognizing car objects in photos and created 1 separate model so that I could run a test of it to recognize all car objects in every photo that the user provided.

As I mentioned, YOLOv8 is slowly gaining popularity in the world of computer vision. YOLO's popularity is for its high precision. YOLO models can be trained on the same 1 single GPU and it has also used a lot of algorithms such as anchor boxes and anchor free detection reduces the number of box predictions, which speeds up Non-Maximum Suppression (NMS), a complicated post processing step that sifts through candidate detections after inference.

So in this project I combined using it to create my own model to identify cars objects in any photo

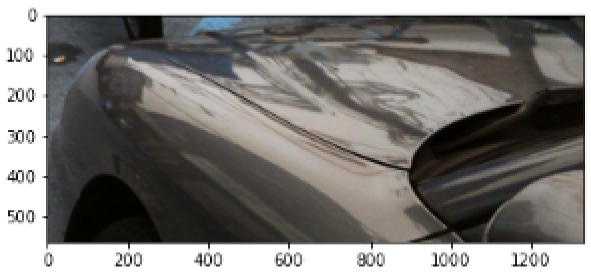
II. DATA PROCESSING

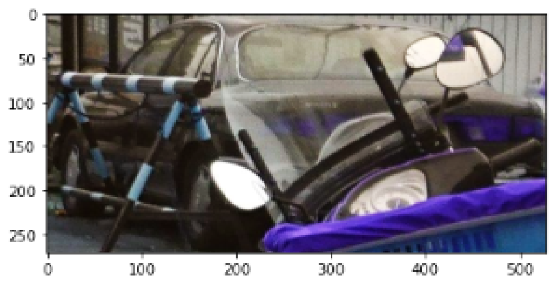
2.1 Dataset Description

A dataset we need to process will include 1 JSON file and 1 image file respectively, the JSON file will contain information of many objects, and each object will have 1 id and 1 separate polygon used to identify where the object is in the image but it is the surrounding border of 1 object but I will turn to Change it to 1 type of boundary box used to identify each object.

First, we will have to isolate all the objects that are needed to identify in the provided JSON file and determine the standard size of the image by extracting the width and height variables in the JSON file from which we have obtained the original width and height of the image corresponding to it will be 2 x and y coordinate axes.

The polygon will be the place to store all the coordinates surrounding each object in the image, here I will have many methods to crop the individual image of each object in the image like Fig(2.1) and Fig (2.2)


fig(2.1)


fig(2.2)

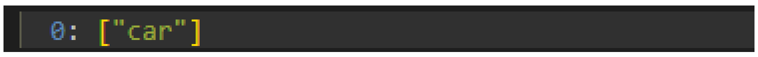
As the 2 images above follow x(max), x(min) and y(max) coordinates, y (min) have been filtered throughout the data of the car objects that I selected in the JSON file by that method we will isolate the box containing the object in it. Suppose I would treat the variables x1, x2 and y1, y2 respectively by the values x (max), x (min) and y (max), y (min) where (x1,y1) will represent the bottom left corner and (x2,y2) represent the right corner like the minimum coordinates and maximum coordinates of the image that contains that object. so that the machine can learn faster and more efficiently in object detection

+ Bounding boxes: bounding boxes are rectangular boxes used to determine the position of objects. They can be identified by the x and y-axis coordinates at the left angle and the x and y-axis coordinates at the right corner of the rectangle

+ Normalization: this is a step that we use to normalize coordinates to a certain range [0,1] or [1,1] to improve convergence in the training process and the effect that can bring. (Assuming the original image will have a value of 255 pixels, to normalize the range [0.1], we must divide each pixel value by 255)

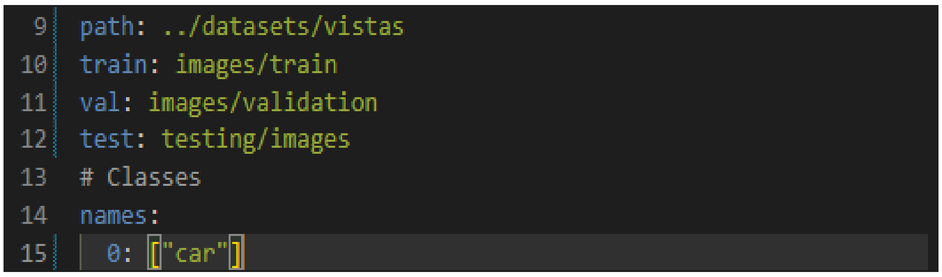
$$\text{Normalised x codinate} = \frac{center\ poit}{width}$$
$$\text{Normalised y codinate} = \frac{center\ poit}{height}$$
$$\text{Normalised width} = \frac{width\ of\ object}{width\ of\ image}$$
$$\text{Normalised height} = \frac{height\ of\ object}{height\ of\ image}$$

+ Class labels: because we have a lot of objects in the same image and have to set them a certain number and specify which object it is here I assigned the number 0 as car



```
0: ["car"]
```

+ Create YAML file: to create a YAML file, the first thing is to pip install pyyaml



```
path: ../datasets/vistas
train: images/train
val: images/validation
test: testing/images
# Classes
names:
  0: ["car"]
```

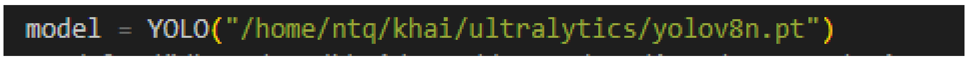
path: will be the path to the folder address containing 2 train and validation files

train: will be the path to the image file used for training

val: path to validation image file

In the steps above we have to create 1 data processing file this step helps us normalize and transform the data so that it is suitable for model training. Next, I will divide the data into 2 sets of train (to train the model) and validation (to test the predictability of the model on the dataset).

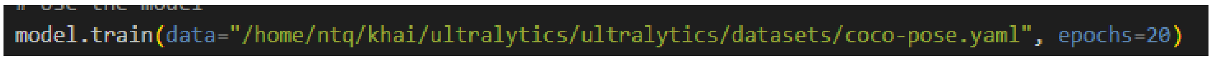
Select the model here, but I mentioned the yolov8 model and I will use the model as a foundation to create the model just to predict the car in many photos.



```
model = YOLO("/home/ntq/khai/ultralytics/yolov8n.pt")
```

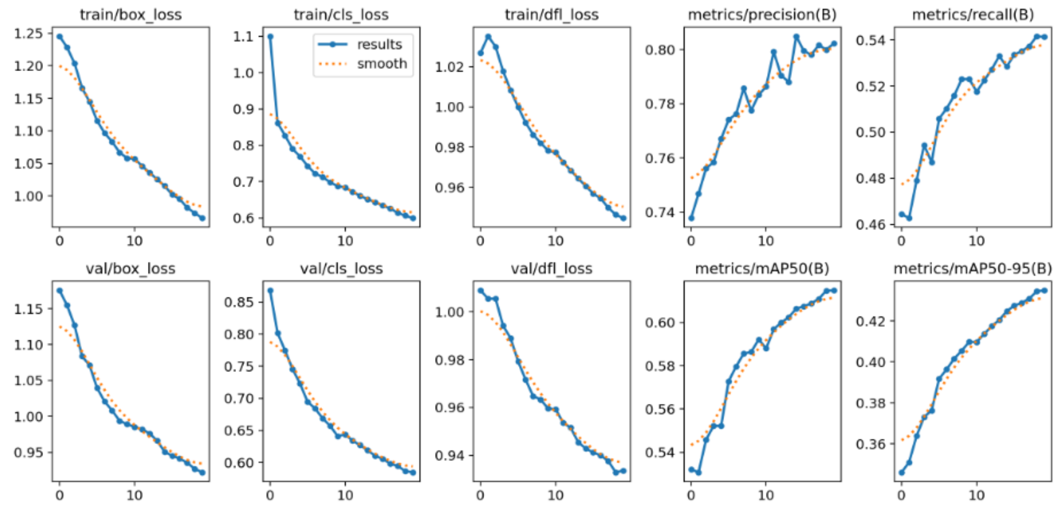
Model training is the process of optimizing parameters to minimize errors between prediction and reality such as and we will evaluate the model after each learning to adjust accordingly, during epoch training is a factor that greatly affects the output of learning results, if epochs are too low, underfitting will occur.

Thus, we must understand what underfitting is. Underfitting is a phenomenon that occurs when a machine learning model is trained on 1 dataset that is too small and cannot generalize well to real world situations, so here I have chosen epochs = 20

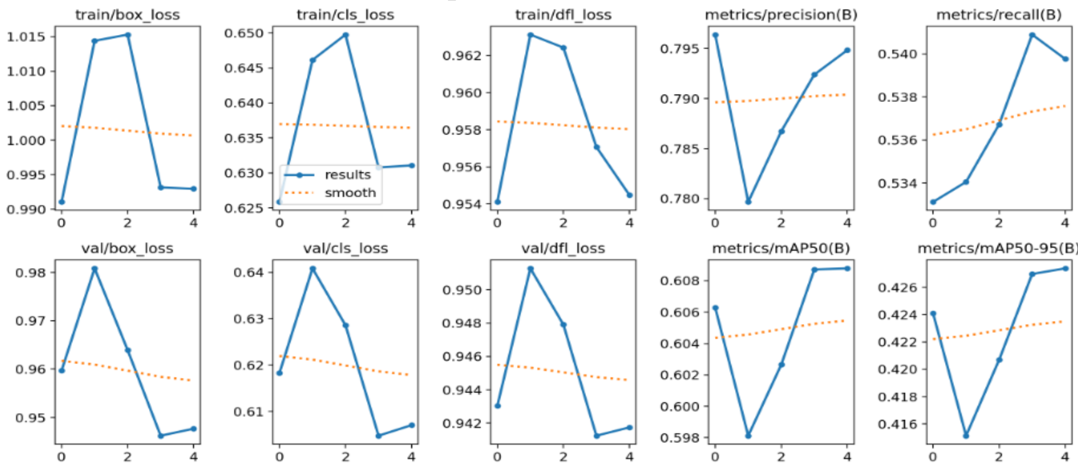


```
model.train(data="/home/ntq/khai/ultralytics/ultralytics/datasets/coco-pose.yaml", epochs=20)
```

Let's say here we can compare my 2 attempts with epochs=20 and epochs=4



(epochs = 20)

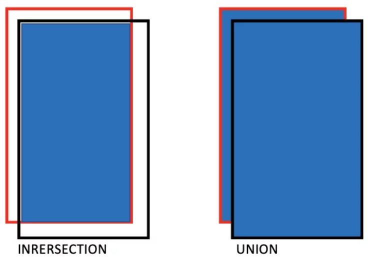


(epochs = 4)

Its accuracy has changed and the errors have been significantly minimized.

2.2 What is IoU(Intersection over Union)?

IoU stands for Intersection over Union, IoU is used in evaluating the ratio of two overlapping areas, especially used in object detection models such as R-CNN, Faster R-CNN, YOLO



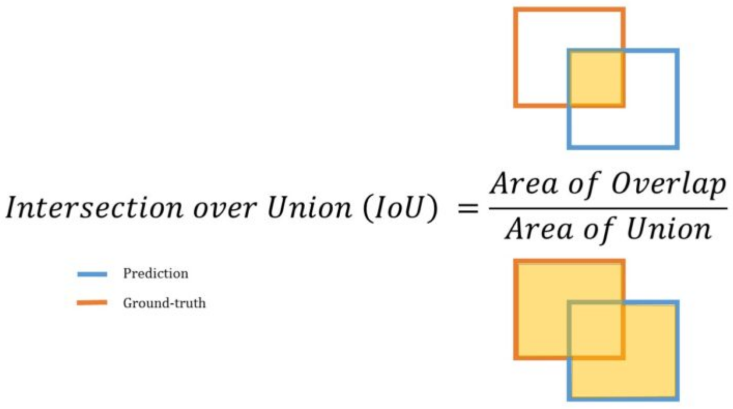
The bounding box is the set of coordinates used to define the position in the image. Different datasets can have different formats of this set.

Unlike measures like *accuracy* or *f1* above that, we do not require an absolute match, but based on the similarity of the two bounding frames. IoU allows us to calculate that number

Suppose we have two sets, S1 and S2 the Jaccard index is calculated by:

$$J(S1, S2) = |S1 \cap S2| / |S1 \cup S2|$$

The IoU algorithm calculates based on the coordinates of 2 rectangles in the Cartesian coordinate system. The IoU formula is calculated as the area of the intersection divided by the area of the union, as shown below:

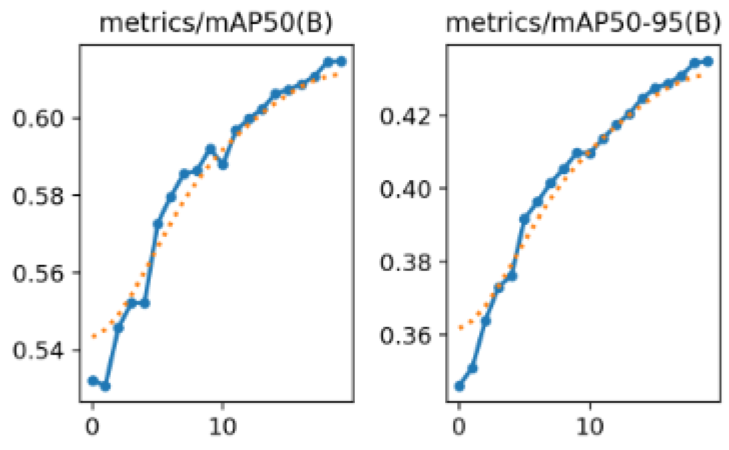

Intersection over Union (IoU) = Area of Overlap / Area of Union
Prediction
Ground-truth

+Area of overlap: is the area of intersection between 2 boxes

+Area of Uinon: is the area of the union of the two boxes

IoU ≥0.5 is considered to have a high degree of similarity.

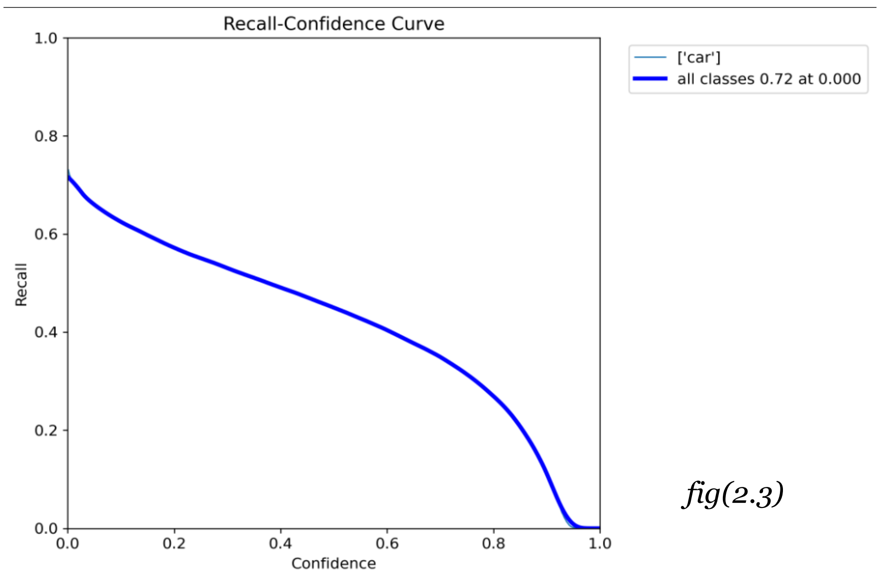
There are my results *fig(2.5)* after experimenting with epochs=20 and you can see every single blue dot is As a result of one test and after 20 times, my model accuracy has gradually increased dramatically and the entire prediction accuracy rate is over 70%



fig(2.5)

2.3 Precision, Recall, and F1-Score

Next, I will so you the graph "Recall-Confidence". It's like the value of all classes on each confidence and if the confidence is higher, the sensitivity will decrease.

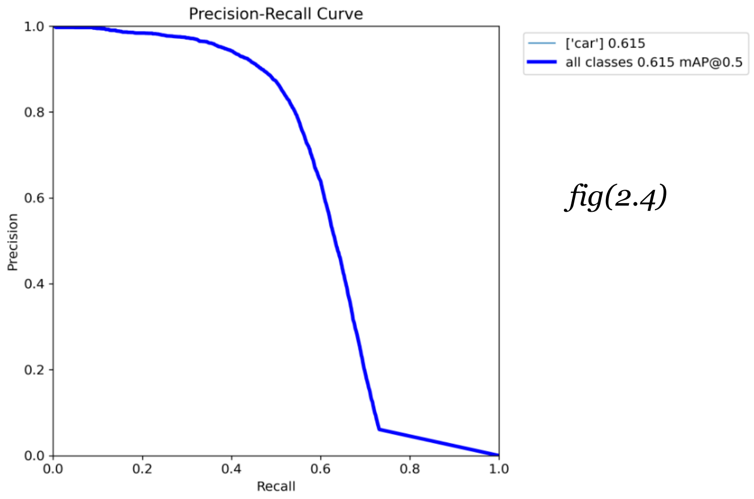


fig(2.3)

In the chart above that *fig(2.3)* we need to pay attention to 2 trouble recall, confidence, and decrease of recall.

Recall means is a measure of the relevance of the relevant object in the photo, it is also understood as the rate of the number of discoveries Subject compared to the total number of actual objects on the image, high sensitivity will search for subjects better Conversely, low sensitivity will miss the subject. And confidence is the bounding box of each object it was found to be accompanied by high confidence score accuracy indicating that the model is definitely true for each object.

There will be a comparison of precision and recall of all classes with objects the car



fig(2.4)

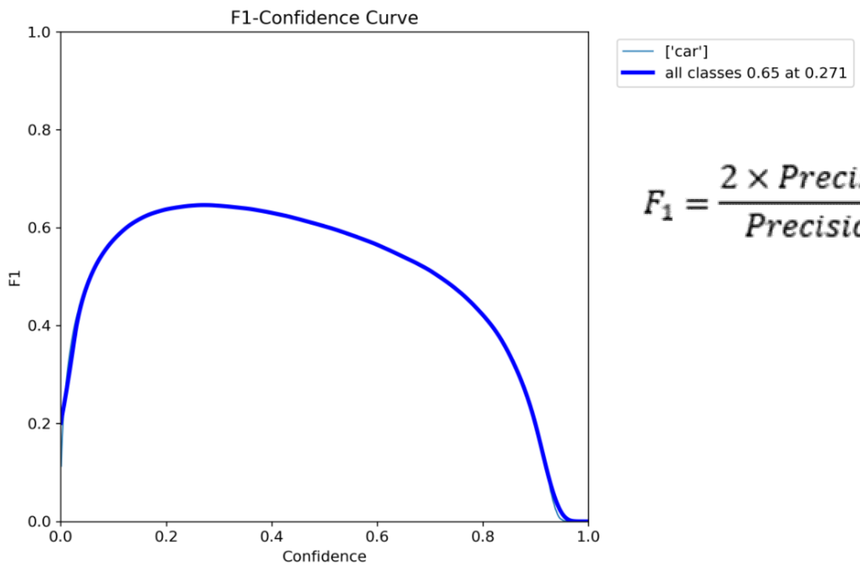
Precision means the focus is on measuring the model's ability to avoid mispredicting patterns belonging to the positive class, i.e. minimizing the number of false positives. A high precision value indicates that the model tends to accurately predict the positive class, is less confusing, and is capable of identifying an actual sample as belonging to the positive class with high confidence.

The formula for calculating precision is: $\frac{TP}{TP + FP}$

- True Positives (TP) is the number of samples belonging to the positive class to which the correct prediction model belongs.

- False Positives (FP) are the number of patterns that the model predicts as belonging to the positive class but in fact belonging to the negative class.

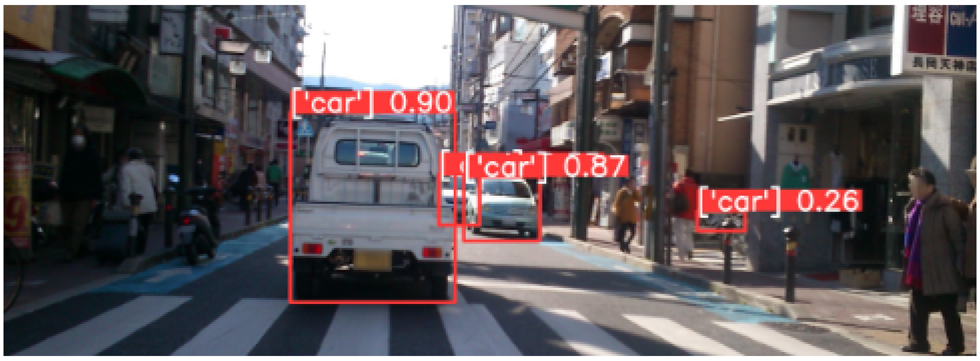
To evaluate the model more comprehensively, we will need to use the F1-score base to evaluate one most accurately



$$F_1 = \frac{2 \times Precision \times Recall}{Precision \ \times Recall}$$

- F1 means a combination of both recall and precision F1-scores range from 0 to 1, with 1 being the best possible score (perfect accuracy and recall) and 0 being the worst (accuracy or recall is 0).

At the end of the training process, I tested again with my new model, and here are the results.



III. API AND DEMO

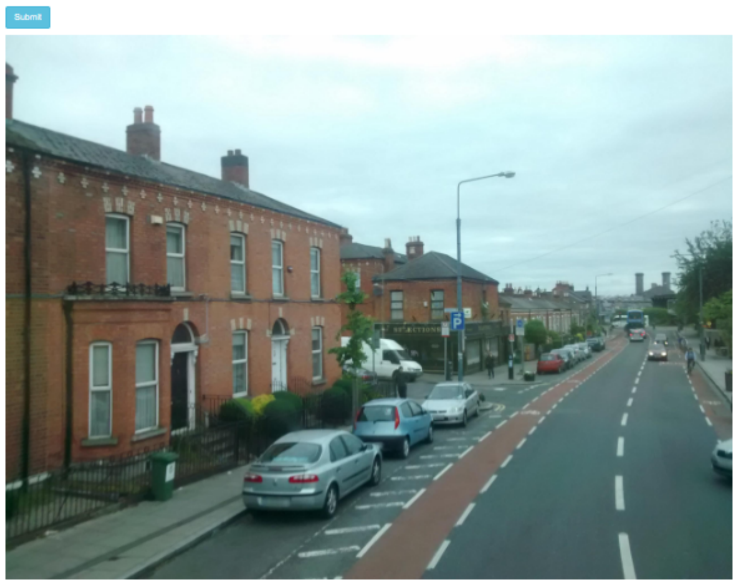
3.1 How to create API?

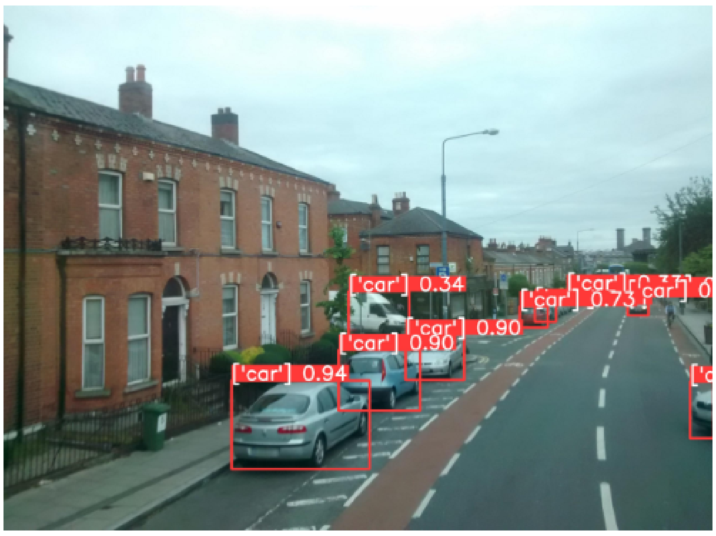
API is an indispensable thing in testing and simplifying the interface so that we can easily test, so here I have created an API so that people can train any image they want to be able to find all objects that are cars.

This time I used Flask, Flask is a Python web framework used to build web applications. It allows users to create their own HTTP so that users can upload files. I used Werkzeug which is a Python library that supports Flask to handle data files.

There will be 1 key function I used to recognize the object (predict_image) after successfully creating a model used just to test the car puzzle above and in this function I will call that trained model to return the result after the user uploads the image. The following are my results after a successful test.

3.2 Training





V. CONCLUSION

The report has presented the process of developing a model for object recognition in images, particularly for objects that are cars. After data processing and model training, we used YOLOv8 ultralytics to create a car object recognition model. We conducted tests and evaluated the performance of the model using metrics such as precision, recall, and F1-score.

The test results demonstrated that the model is capable of accurately identifying car objects in images with high precision. Additionally, we built a simple API using Flask to enable users to upload images and perform object recognition on those images.

However, in the future, further enhancements and improvements to the model may be necessary to increase the object recognition performance, as well as to improve the precision and recall of the model. Additionally, the scope of the model could be expanded to recognize objects other than cars, but this would require additional training data and appropriate adjustments.

Overall, the project has achieved its goal of object recognition for cars in images and has produced an effective model. Further extensions and optimizations can be applied to enhance the practical application capabilities of the model.