



Final Project: Object Detection using YOLO

Môn: Học Thống Kê

Họ tên: Phan Đặng Anh Khôi

MSSV: 21127325

Lớp 21CNTT

Giảng viên phụ trách

Ngô Minh Nhựt

Lê Long Quốc



Contents

I.	Mức độ hoàn thành	2
II.	Về yêu cầu 1	2
1.	Object detection là gì?	2
2.	Ứng dụng của object detection	2
3.	Các mô hình object detection.....	2
4.	YOLO (You Only Look Once) [1, 2].....	3
5.	Yêu cầu 1	4
III.	Về Yêu cầu 2	5
1.	Chủ đề thực hiện	5
2.	Về dữ liệu.....	5
3.	Huấn luyện mô hình.....	6
4.	Kiểm tra mô hình	6
5.	Kết quả phát hiện đối tượng.....	12
IV.	Reference	15

I. Mức độ hoàn thành

Yêu cầu	Hoàn thành
1/ Tải và sử dụng mô hình YOLO có sẵn để xây dựng một trang web cho phép người dùng tải ảnh lên và trả về kết quả phát hiện đối tượng.	100%
2/ Áp dụng YOLO để xây dựng một ứng dụng phát hiện đối tượng bao gồm việc hiểu mô hình – 2 giai đoạn huấn luyện và suy luận.	100%

II. Về yêu cầu 1

1. Object detection là gì?

- Trong thị giác máy tính, đây là một kỹ thuật dùng để phát hiện đối tượng trong một video hoặc hình ảnh. Bao gồm việc xác định đối tượng và đóng bounding box cho vật đó.

2. Ứng dụng của object detection

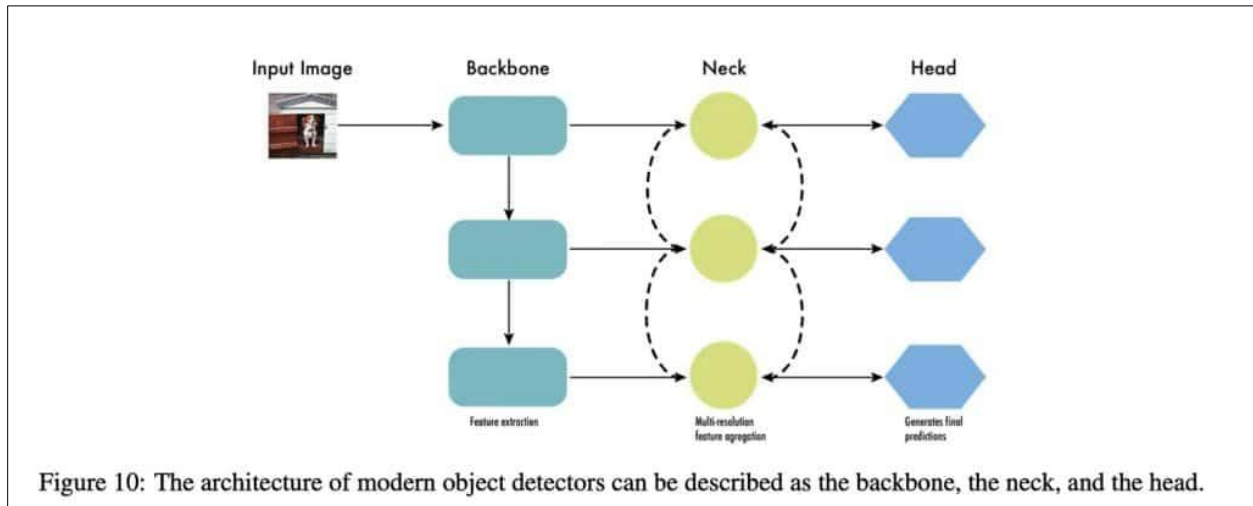
- Phát hiện khuôn mặt người, hành vi bất thường (đánh nhau,...) trong camera giám sát.
- Nhận diện biển số xe.
- Phát hiện bất thường trong y tế (khối u,...)
- Ô tô tự hành (phát hiện và né tránh vật cản)
- Phát hiện bất thường trong cây trồng (sâu bệnh)

3. Các mô hình object detection

- Được chia thành 2 mô hình chính: one-stage (một giai đoạn) và two-stage (hai giai đoạn).
- Mô hình một giai đoạn thực hiện phát hiện đối tượng trong một bước duy nhất, nhanh hơn nhưng có độ chính xác thấp hơn so với mô hình hai giai đoạn.
- Các mô hình một giai đoạn: YOLO, Single Shot Multibox Detector...
- Các mô hình hai giai đoạn: R-CNN và các biến thể của nó (Fast, Faster R-CNN)

4. YOLO (You Only Look Once) [1, 2]

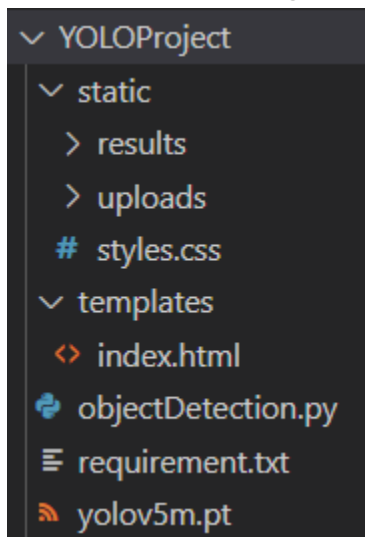
- Mô hình chung hiện nay



- Phần Backbone dùng để trích xuất các đặc trưng sử dụng CNN (mạng tích chập) đã được huấn luyện. Các đặc trưng thấp như cạnh được trích xuất ở các lớp tích chập đầu tiên. Các đặc trưng cấp cao như ngữ nghĩa của vật thể được trích xuất ở các tầng sau.
- Phần Neck: kết nối giữa Backbone và Head.
- Phần Head: chịu trách nhiệm cho việc dự đoán.
- Đây là một trong những mô hình nổi tiếng dùng để phát hiện đối tượng hiện nay.
- Tính đến tháng 5 năm 2024, YOLOv10 là mô hình mới nhất với tốc độ lẫn độ chính xác được cải thiện so với các mô hình trước đó.
- YOLO là một thuật toán supervised (học có giám sát), nghĩa là cần dữ liệu đã được gán nhãn để huấn luyện, “nhãn” ở đây là các thông tin về bounding box và class từ ảnh đầu vào.
- Cách hoạt động của YOLO [2]:
 - (1) Chia ảnh thành lưới (5x5, 7x7, tùy thuộc)
 - (2) Dự đoán bounding box và xác suất đối tượng (class) trong bounding box đó. Mỗi bb được xác định bởi: (x, y) tọa độ phía trên bên trái của bounding box; (w, h) chiều rộng và cao của bb; p(obj) xác suất đối tượng tồn tại trong bb này.
 - (3) Tính confidence score cho mỗi bounding box (xác suất mà bounding box thuộc vào một lớp)
 - (4) Non-max supression: Loại bỏ các bounding box bị chồng lên nhau.

5. Yêu cầu 1

- Được xây dựng sử dụng micro web framework Flask trong python. Sử dụng mô hình yolov5 [4] để thực hiện phát hiện đối tượng.
- Về cấu trúc của ứng dụng:



Flask App (YOLOProject):

- **Folder static – thư mục tĩnh, chứa các file không thay đổi trong quá trình chạy:**
 - + Folder *results* lưu trữ hình ảnh kết quả phát hiện.
 - + Folder *uploads* lưu trữ hình ảnh tải lên.
 - + *styles.css* kiểm soát bố cục, font chữ,... của trang web
- **Folder templates – thư mục templates, chứa các file .html:**
 - + *index.html* cấu trúc của một trang web.
- *objectDetection.py* là file chính để khởi tạo chương trình flask.
- *requirement.txt* là file chứa các thư viện cần để chạy chương trình.
- *yolov5m.pt* (file pretrained) là bộ trọng số đã được huấn luyện.
- Các hàm trong *objectDetection.py*
 - + *getFirstImagePath(folder_path)*: lấy ra đường dẫn ảnh đầu tiên trong folder.
 - + *clrFolder(folder_path)*: xóa tất cả file trong folder.
 - + *resizeImagecv2(image_path)*: Thay đổi kích thước của ảnh về 800, 600 nếu lớn hơn nó.
 - + *@app.route('/') index*: định nghĩa một route tới *index.html* (mặc định)

+ @app.route('/upload', method=['POST']) upload_file: xử lý khi người dùng upload (POST) một hình ảnh, kết quả trả về là render_template(index.html với tham số result_img là hình ảnh kết quả phát hiện đối tượng)

app chính là ứng dụng Flask đã được khởi tạo: app = Flask(__name__)

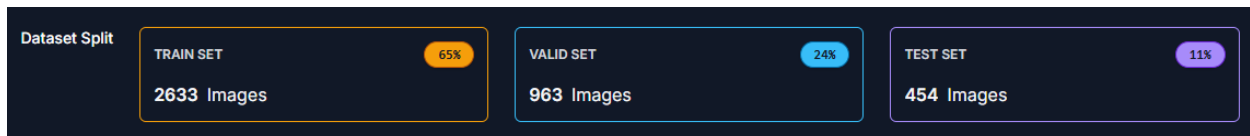
III. Về Yêu cầu 2

1. Chủ đề thực hiện

- Vehicle Detection.
- Nhằm phát hiện các loại xe trên đường, có thể ứng dụng để làm xe tự hành hay được sử dụng trong hệ thống thu phí ETC.

2. Về dữ liệu

- Được lấy từ roboflow (một trang web chuyên về các tác vụ Thị Giác Máy Tính).
- Venom Dataset YOLOv7 [5].
- Bộ dữ liệu này gồm 4050 ảnh, chia thành 3 folder: train – valid – test.



- Trong đó tập train gồm 2633 ảnh, valid gồm 963 ảnh và test gồm 454 ảnh.
- Mỗi folder có thêm 2 folder con là images (gồm ảnh) và labels (nhãn), trong đó các nhãn được đánh dấu như sau [6]:
ClassID – x_center – y_center – width – height
+ ClassID: ID của class.
+ x_center, y_center: tọa độ x, y của bounding box ở trung tâm.
+ width, height: chiều ngang và cao của bounding box.
- Ngoài ra có file data.yaml đi kèm như sau:

```

train: ../train/images
val: ../valid/images
test: ../test/images

nc: 12
names: ['bus-l-', 'bus-s-', 'car', 'extra large truck', 'large bus',
        'medium truck', 'small bus', 'small truck', 'truck-l-', 'truck-m-',
        'truck-s-', 'truck-xl-']

roboflow:
  workspace: ilham-winar
  project: venom
  version: 8
  license: CC BY 4.0
  url: https://universe.roboflow.com/ilham-winar/venom/dataset/8

```

- Có tổng cộng 12 class trong dữ liệu này, gồm: bus-l-, bus-s-, car, extra large truck, large bus, medium truck, small bus, small truck, truck-l-, truck-m-, truck-s-, truck-xl-.

3. Huấn luyện mô hình

- Sử dụng mô hình YOLOv7 có sẵn từ github [7].
- Các bước chuẩn bị đã được liệt kê trong file jupyter notebook đính kèm.
- Thực hiện huấn luyện 50 lần x3 (vì sau mỗi 50 lần sẽ bị giới hạn GPU) tổng cộng 150 lần (epochs). Lần đầu (0-50) huấn luyện sử dụng bộ trọng số đã huấn luyện trước (pretrained weight). Lần 2 (51-100) sử dụng bộ trọng số cuối mà lần đầu cho (last.pt) tương tự với lần 3 (101-150).

4. Kiểm tra mô hình

- Thực hiện kiểm tra mô hình với bộ trọng số cuối của lần 3, thu được kết quả sau:

```

val: Scanning 'data/valid/labels.cache' images and labels... 963 found, 0 missing, 0 empty, 0 corrupted: 100% 963/963 [00:00<?, ?it/s]

```

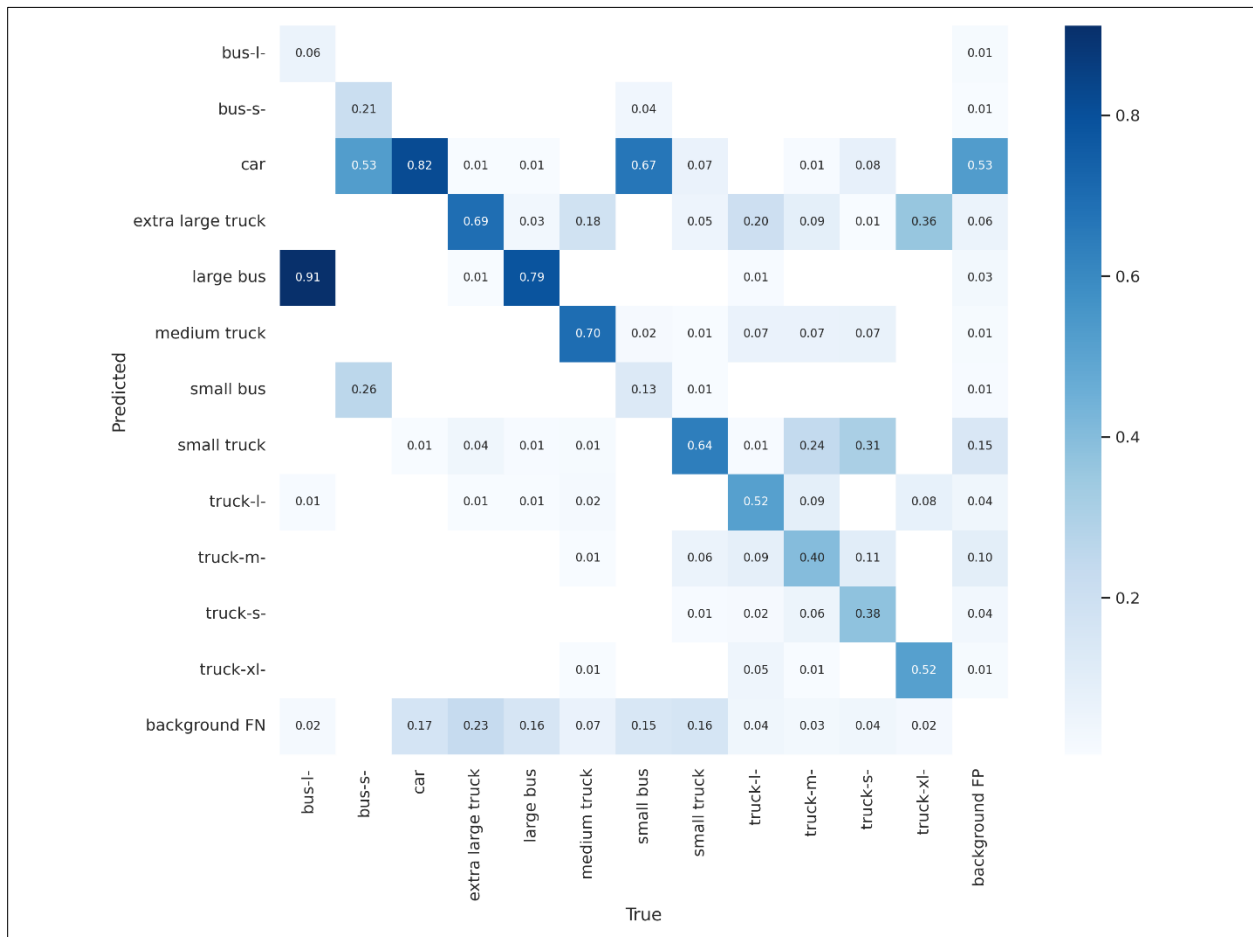
Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	963	13450	0.486	0.601	0.436	0.29
bus-l-	963	8	0.0463	0.75	0.058	0.0226
bus-s-	963	12	0.264	0.75	0.2	0.155
car	963	8537	0.814	0.777	0.827	0.481
extra large truck	963	1162	0.8	0.488	0.638	0.385
large bus	963	273	0.763	0.591	0.727	0.497
medium truck	963	257	0.696	0.409	0.411	0.306
small bus	963	49	0.135	0.188	0.0566	0.0343
small truck	963	1721	0.645	0.595	0.603	0.361
truck-l-	963	433	0.466	0.672	0.515	0.375
truck-m-	963	629	0.352	0.676	0.363	0.257
truck-s-	963	221	0.351	0.579	0.29	0.187
truck-xl-	963	148	0.496	0.743	0.548	0.424

```

Speed: 499.0/1.8/500.8 ms inference/NMS/total per 640x640 image at batch-size 64
Results saved to runs/test/yolov7_valid_best

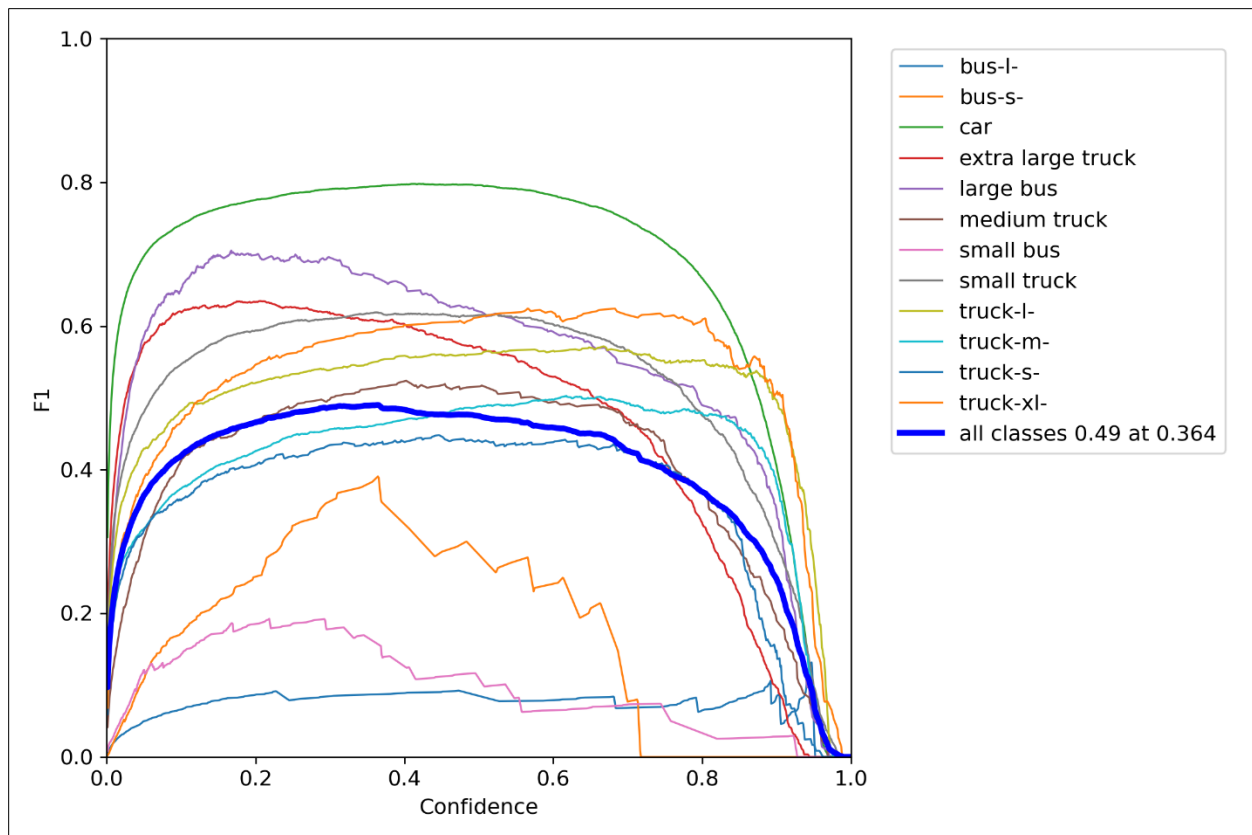
```

- Trong folder yolov7_valid_best ta được một vài file



Confusion Matrix

- Có thể thấy class bus-l bị nhầm lẫn với class large bus, có thể là do bus-l cũng là large bus. Tương tự với bus-s và small bus. Class car cũng nhầm lẫn với bus-s và small bus.



F1-score curve

- Nhìn chung F1-score có hình dạng như cây cầu, có một vài class có F1 không cao.
- Lớp Car có F1-score cao nhất là 0.8 với độ tin cậy tối ưu.
- Lớp bus-l có F1-score thấp nhất, có nhiều dao động không ổn định.
- Các lớp còn lại có F1-score dao động khá ổn định, từ 0.6-0.7.
- Biến động từ 0 đến 0.2-0.4 (confidence) làm cho F1-score của các lớp tăng đều sau đó giảm dần -> Nghĩa là có một ngưỡng tin cậy tối ưu cho tất cả các class.
- All Class có F1-score là 0.49 tại ngưỡng tin cậy 0.364 (ngưỡng tin cậy tối ưu cho tất cả các lớp).
- Một số lớp có khả năng phân loại kém khi đạt ngưỡng confidence cao hơn (ví dụ như truck-xl, small bus)

Về đánh giá F1-score sau các epoch [8, 9]

- **Precision:** độ chính xác; Precision cao nghĩa là độ chính xác của các điểm tìm được là cao.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- **Recall:** độ nhạy; Recall cao nghĩa là mô hình ít bỏ sót mô hình các điểm positive (tỷ lệ bỏ các điểm positive) thấp.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- **F1-score:** là điều hòa giữa Precision và Recall. Thực tế thì khi mô hình học, precision càng cao sẽ dẫn đến recall càng thấp và ngược lại -> F1-score sẽ điều chỉnh đại lượng này. F1-score = 1 đồng nghĩa với Precision và Recall đều = 1.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- Ví dụ:

		Predicted				Total
		Cat	Dog	Tiger	Wolf	
Actual	Cat	6	0	3	1	10
	Dog	2	4	0	4	10
	Tiger	3	3	3	0	9
	Wolf	1	4	1	2	8
Total		12	11	7	7	

- Class Dog có:
 - + Precision(Dog) = 4/11 = 0.364
 - + Recall(Dog) = 4/10 = 0.4
 - + F1-score(Dog) = 2 * (0.364 * 0.4) / (0.364 + 0.4) = 0.381
- Class Tiger có:
 - + Precision(Tiger) = 3/7 = 0.428

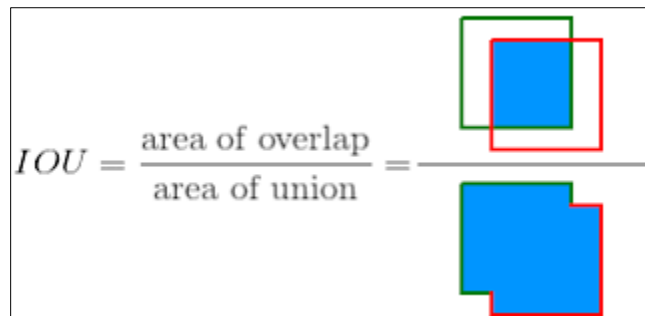
$$+ \text{Recall}(\text{Tiger}) = 3/9 = 0.333$$

$$+ \text{F1-score}(\text{Tiger}) = 2 * (0.428 * 0.333) / (0.428 + 0.333) = 0.375$$

- F1-score tại một vài epochs trong huấn luyện (lấy từ file results_101-150):

0/49	200	640	0.475	0.5224	0.4213	0.2796
1/49	114	640	0.4782	0.5116	0.4233	0.2802
2/49	153	640	0.489	0.5046	0.4252	0.2801
3/49	98	640	0.4331	0.5606	0.4213	0.2771
4/49	101	640	0.4924	0.4934	0.4189	0.2735
5/49	127	640	0.4188	0.565	0.4136	0.2699
6/49	141	640	0.4541	0.5314	0.4182	0.2756
7/49	184	640	0.4617	0.5295	0.4172	0.2691
8/49	230	640	0.4714	0.4907	0.4189	0.2632
9/49	148	640	0.4567	0.4941	0.3989	0.2547

- Các cột từ trái qua phải: Epochs, Labels, Image Size, Precision, Recall, mAP@.5, mAP@.5:.95. Trong đó mAP (mean Average Precision) trong object detection thì là một độ đo được tính theo trung bình cộng giá trị Average Precision của tất cả các class. Về khái niệm IoU (intersection over union)



$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$

- IoU là tỉ lệ giữa giao và hợp của 2 vùng màu xanh dương, với màu xanh lá là Bounding box Predicted và màu đỏ là Bounding box Groundtruth.
- mAP@.5 (mean Average Precision at IoU = 0.5) nghĩa là IoU cố định ở mức 0.5 có giá trị 0.42 tại epoch 0/49.
- mAP@.5:.95 nghĩa là IoU khắt khe hơn, có mức 0.5 đến 0.95 có giá trị 0.27 tại epoch 0/49.
- Hai độ đo này càng cao thì hiệu suất mô hình cũng càng cao
- Quay trở lại với F1-score, áp dụng công thức tại một số epoch:

$$F1 - score_{0/49} = 2 * \frac{0.475 * 0.5224}{0.475 + 0.5224} = 0.4976$$

$$F1 - score_{9/49} = 2 * \frac{0.4567 * 0.4941}{0.4567 + 0.4941} = 0.4747$$

- F1-score tại một số epoch cuối:

45/49	140	640	0.4704	0.5951	0.4339	0.2896
46/49	138	640	0.4941	0.5431	0.4151	0.278
47/49	187	640	0.4878	0.5615	0.4236	0.2832
48/49	131	640	0.493	0.5507	0.4214	0.2813
49/49	123	640	0.4779	0.5633	0.4159	0.2797

$$F1 - score_{48/49} = 2 * \frac{0.493 * 0.5507}{0.493 + 0.5507} = 0.5202$$

$$F1 - score_{49/49} = 2 * \frac{0.4779 * 0.5633}{0.4779 + 0.5633} = 0.5171$$

- Các độ đo F1-score này được tính theo Precision và Recall của tất cả các class tại một epoch.

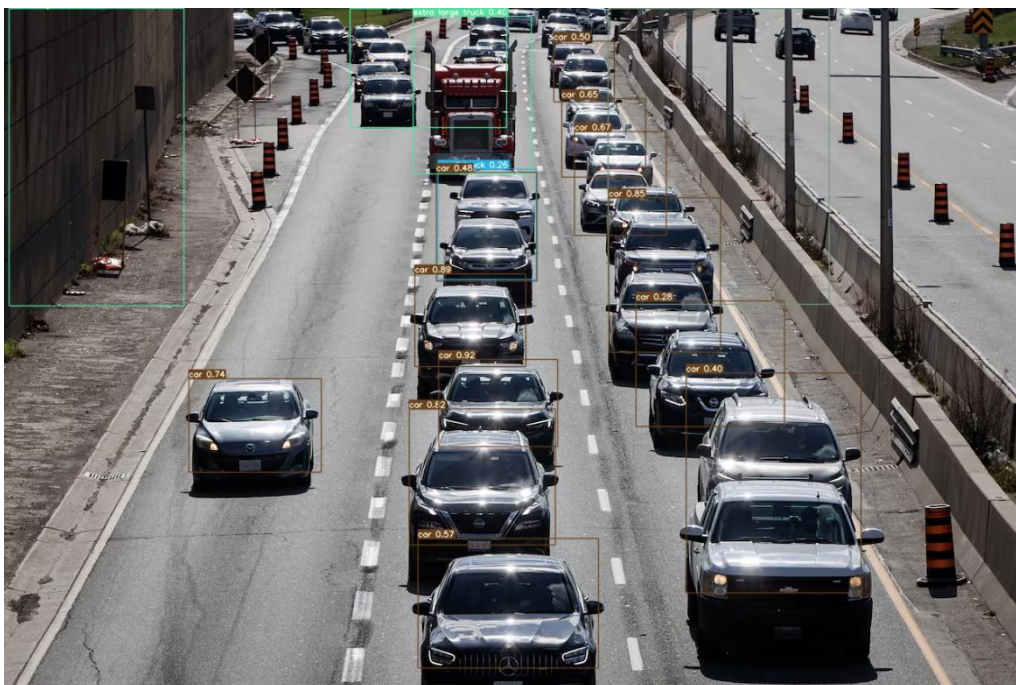
Nhận xét tổng thể

- Mô hình này học bất ổn với dữ liệu, lúc tăng, lúc giảm. Để cải thiện mô hình, ta có thể thêm số lần huấn luyện, thay đổi learning rate và bộ tối ưu (đã sử dụng bộ tối ưu SGD với lr = 0.01). Ngoài ra trước huấn luyện, chưa áp dụng tăng cường dữ liệu (data augmentation) nên cũng có thể thực hiện tăng cường dữ liệu như các phép biến đổi ảnh xoay, scale ảnh, thay đổi độ sáng của ảnh... có thể sẽ giúp mô hình hoạt động tốt hơn.
- Ở một vài epoch, khi precision tăng thì recall giảm và ngược lại (tại epoch 23/49 trong file results_101-150).
- Mô hình hoạt động tốt nhất trên class car và có hiệu suất tương đối ổn định với các class extra large truck, large bus và medium truck.
- Nhưng lại gặp khó khăn trong việc phân loại các lớp như bus-s và small bus.

5. Kết quả phát hiện đối tượng







- Mô hình chỉ phát hiện các xe trong ảnh với kích thước đủ to như không quá to như hình trên, đây cũng là một nhược điểm của YOLO – vấn đề về chia lưới, nếu lưới quá nhỏ thì không phát hiện được kích thước lớn và ngược lại. Dù vậy, mô hình vẫn đạt tốc độ cao, hầu hết cho kết quả đều dưới 1s (kích thước ảnh càng lớn thì càng cần nhiều thời gian để xử lý).

IV. Reference

- Các bài giảng từ đầu năm môn Học Thống Kê [HKIII, 2024].

Yêu cầu 1 (tính đến ngày 13 tháng 8, 2024)

- [1] [What is YOLO? The Ultimate Guide \[2024\] \(roboflow.com\)](#)
- [2] [A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS](#)
- [3] [YOLO Algorithm: Real-Time Object Detection from A to Z \(kili-technology.com\)](#)
- [4] [ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite \(github.com\)](#)

Yêu cầu 2 (tính đến ngày 13 tháng 8, 2024)

- [5] [Venom Object Detection Dataset by ilham winar \(roboflow.com\)](#)
- [6] [What is the YOLOv7 PyTorch TXT Annotation Format? \(roboflow.com\)](#)
- [7] [WongKinYiu/yolov7: Implementation of paper - YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors \(github.com\)](#)
- [8] [Đánh giá model AI với Precision, Recall và F1 Score - Mì AI \(miai.vn\)](#)
- [9] [Thử tìm hiểu về mAP - đo lường Object Detection model - Mì AI \(miai.vn\)](#)