



NHÓM SINH VIÊN THỰC HIỆN:

1. NGUYỄN MAI HOÀNG ANH - 23001824
2. NGUYỄN THỊ LAN ANH - 23001826
3. BÙI XUÂN CHUNG - 23001838
4. ĐÀO NGỌC CƯỜNG - 23001841
5. NGUYỄN THỊ HUYỀN LINH - 23001899

**NHẬN DIỆN
PHƯƠNG TIỆN
GIAO THÔNG**

Nêu vấn đề



Hình ảnh của camera AI do Cục CSGT công bố vào ngày 26/11/2025
tại nút giao Phạm Văn Bách-Hoàng Quán Chi (Cầu Giấy)

Nêu vấn đề

PHỤ LỤC CAMERA AI GHI NHẬN DỮ LIỆU VI PHẠM TẠI TẠI TUYẾN CAO TỐC NỘI BÀI - LÀO CAI VÀ NỘI ĐÔ HÀ NỘI (từ 12h ngày 24/11/2025 đến 12h ngày 25/11/2025)

Số	Thời gian phát hiện	Biển số	Màu lố	Đối tượng	Loại cảnh báo
1	24/11/2025 - 19:47:37	29G100050	Trắng	Xe mô tô	Vượt đèn đỏ
2	24/11/2025 - 19:21:04	29V740746	Trắng	Xe mô tô	Vượt đèn đỏ
3	24/11/2025 - 16:45:14	35N164050	Trắng	Xe mô tô	Vượt đèn đỏ
4	25-11-2025 07:27:28	29P171002	Trắng	Xe mô tô	Vượt đèn đỏ
5	25-11-2025 06:59:39	29L168888	Trắng	Xe mô tô	Vượt đèn đỏ
6	25-11-2025 06:16:16	29X579452	Trắng	Xe mô tô	Vượt đèn đỏ
7	25-11-2025 06:13:54	29E233791	Trắng	Xe mô tô	Vượt đèn đỏ
8	25/11/2025 - 05:58:49	29M0220767	Trắng	Xe mô tô	Vượt đèn đỏ
9	25/11/2025 - 05:53:31	29S620117	Trắng	Xe mô tô	Vượt đèn đỏ
10	25/11/2025 - 05:49:35	30H83234	Trắng	Xe mô tô	Vượt đèn đỏ
11	25/11/2025 - 05:49:23	20C131070	Trắng	Xe mô tô	Vượt đèn đỏ
12	25/11/2025 - 05:49:12	29X358072	Trắng	Xe mô tô	Vượt đèn đỏ
13	25/11/2025 - 05:41:49	99013986	Trắng	Xe mô tô	Vượt đèn đỏ
14	25/11/2025 - 05:34:54	29T209007	Trắng	Xe mô tô	Vượt đèn đỏ
15	25/11/2025 - 05:10:26	23E105889	Trắng	Xe mô tô	Vượt đèn đỏ
16	25/11/2025 - 05:04:28	29M120026	Trắng	Xe mô tô	Vượt đèn đỏ
17	25/11/2025 - 05:02:51	99AE03095	Trắng	Xe mô tô	Vượt đèn đỏ
18	25/11/2025 - 04:58:44	29X541188	Trắng	Xe mô tô	Vượt đèn đỏ
19	25/11/2025 - 04:55:21	37A33010	Trắng	Xe mô tô	Vượt đèn đỏ
20	25/11/2025 - 04:55:16	29Y23244	Trắng	Xe mô tô	Vượt đèn đỏ

Tại sao Cục
CSGT có thể thu
được những
danh sách này?



Giới thiệu Nhận diện phương tiện giao thông bằng AI

Mục tiêu & bài toán:

- Phát hiện nhanh và chính xác các phương tiện (ô tô, xe máy, xe tải, bus) và người đi bộ trong môi trường giao thông phức tạp.
- Xử lý real-time, chịu được che khuất, vật thể nhỏ, điều kiện ánh sáng thay đổi và góc nhìn khác nhau.

Giải pháp:

- Mô hình: YOLOv8 (one-stage detector, anchor-free)
- Ưu điểm: tốc độ cao (FPS), độ chính xác tốt (mAP), vượt hạn chế của Faster R-CNN.

Ứng dụng:

- Giám sát giao thông thông minh
- Phát hiện vi phạm tự động (vượt đèn đỏ, đi sai làn)
- Đếm và phân loại phương tiện, thu thập dữ liệu lưu lượng
- Kết hợp tracking (DeepSORT, ByteTrack) để theo dõi hành vi phương tiện

Object Detection – Định nghĩa & Ứng dụng

Định nghĩa:

- Object Detection là bài toán xác định vị trí và nhãn của từng đối tượng trong ảnh/video.
- Mục tiêu: phát hiện nhiều đối tượng cùng lúc, ngay cả khi chúng chồng lấn (occlusion) hoặc có kích thước nhỏ.

Đặc điểm chính:

- Vị trí: Mỗi đối tượng được bao quanh bằng bounding box (x, y, w, h).
- Nhãn (class label): Xác định loại đối tượng (xe máy, ô tô, người đi bộ...).
- Độ tin cậy (confidence): Xác suất dự đoán đúng đối tượng.

Ứng dụng:

- Giám sát giao thông thông minh
- Robot tự hành
- Nhận diện khuôn mặt, phân tích hành vi con người

Kiến trúc chung của Object Detection

1. Backbone (Feature Extraction):

- Trích xuất đặc trưng từ ảnh đầu vào: cạnh, màu sắc, hình dạng, các mẫu phức tạp.
- Các kiến trúc phổ biến: ResNet, Darknet, EfficientNet, MobileNet, DenseNet.
- Vai trò: Là nền tảng quyết định chất lượng feature map.

2. Neck (Feature Fusion – tùy chọn):

- Kết hợp đặc trưng từ nhiều tầng (multi-scale feature fusion).
- Giúp nhận diện các đối tượng có kích thước khác nhau.
- Ví dụ: FPN (Feature Pyramid Network), PAN (Path Aggregation Network).

3. Detection Head:

- Dự đoán bounding box, class label, và confidence score.
- Áp dụng Non-Maximum Suppression (NMS) để loại bỏ các dự đoán trùng lặp.
- Là phần cuối cùng quyết định kết quả nhận diện.

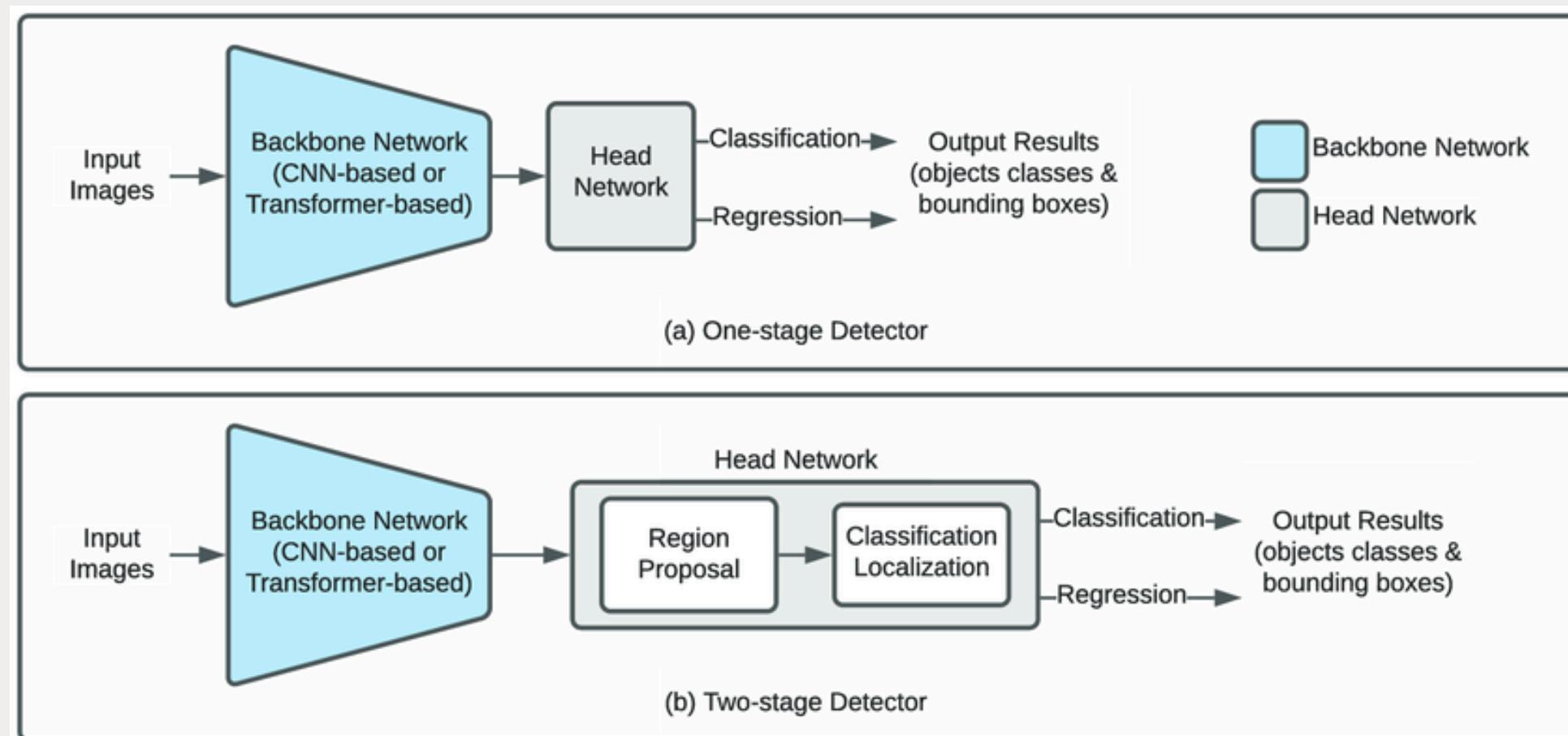
Nhóm mô hình Object Detection

Two-stage detectors: Faster R-CNN, Mask R-CNN

- Ưu điểm: Độ chính xác cao, tốt với small objects
- Nhược điểm: Tốc độ xử lý chậm, khó real-time

One-stage detectors: YOLO (v1→v8), SSD, RetinaNet

- Ưu điểm: Tốc độ cao, dễ triển khai real-time
- Nhược điểm: Đôi khi độ chính xác thấp với small objects hoặc occlusion



Thách thức & Ứng dụng trong giám sát giao thông

Thách thức:

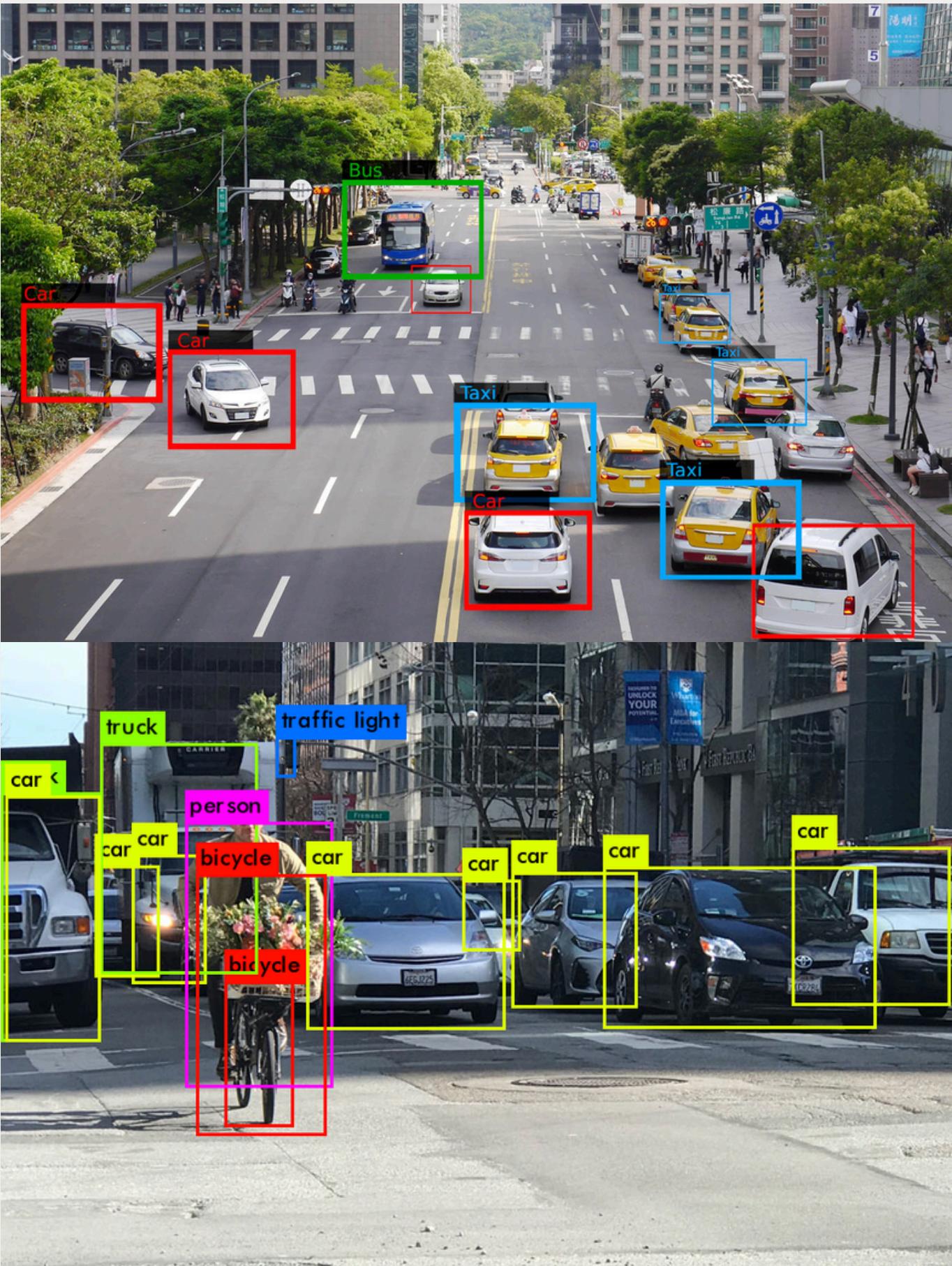
- Vật thể nhỏ (small objects)
- Che khuất (occlusion)
- Đa dạng môi trường: ánh sáng, thời tiết, góc nhìn
- Background phức tạp

Ứng dụng trong giao thông:

- Đếm phương tiện: xe máy, ô tô, bus
- Phát hiện vi phạm: vượt đèn đỏ, đi sai làn
- Phân tích lưu lượng và hỗ trợ camera thông minh

Kỹ thuật cải thiện:

- Augmentation, regularization, fine-tuning → độ chính xác & khả năng generalization tốt



Faster R-CNN

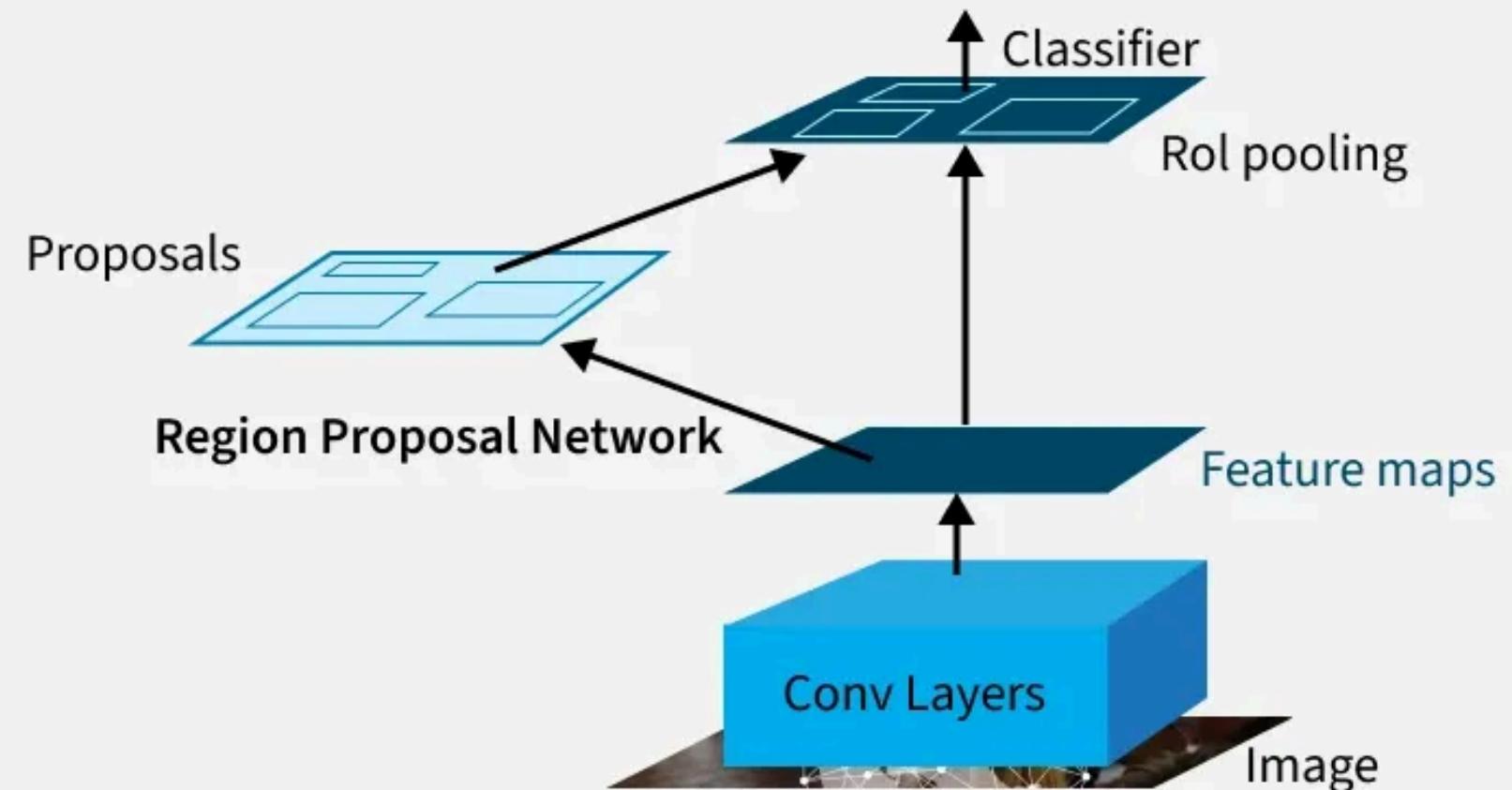
Phát triển bởi Shaoqing Ren et al., 2015

Hai giai đoạn chính:

1. RPN (Region Proposal Network): tạo các đề xuất vùng chứa đối tượng
2. Detection Head (Fast R-CNN): phân loại và tinh chỉnh bounding box

Kết hợp Backbone CNN + Anchor Boxes + Multi-task Loss

Ứng dụng: giám sát giao thông, an ninh, y tế, video analytics



Backbone ResNet

- Backbone là mạng trích xuất đặc trưng từ ảnh đầu vào, tạo feature map cho RPN và ROI Head.
- ResNet sử dụng cơ chế Residual Learning để huấn luyện mạng sâu mà không bị mất gradient.
- Cấu trúc Residual Block:

Input → Conv → BN → ReLU → Conv → BN → Skip Connection (x) → Output
→ Giúp mạng học phần dư $F(x)$ thay vì toàn bộ ánh xạ.
- ResNet50 làm backbone:
 - Gồm 5 stage: conv1 → conv2_x → conv3_x → conv4_x → conv5_x
 - Kết hợp FPN để tạo đa mức đặc trưng (P2–P5) cho cả đối tượng nhỏ – lớn
 - Cho đặc trưng ổn định, mạnh, hiệu quả cao trong detection.

Anchor Boxes

- Anchors: bounding box tham chiếu với nhiều kích thước và tỷ lệ
- Tại mỗi điểm trên feature map, ví dụ 3×3 tỷ lệ $\times 3$ kích thước $\rightarrow 9$ anchor boxes
- Dùng để tính: Objectness score & Bounding box regression
- Công thức bounding box regression:

$$t_x = \frac{x - x_a}{w_a}, \quad t_y = \frac{y - y_a}{h_a}, \quad t_w = \log \frac{w}{w_a}, \quad t_h = \log \frac{h}{h_a},$$



Region Proposal Network (RPN)

Quét feature map, với mỗi anchor, dự đoán:

- Objectness score `obj_ipi`
- Bounding box offsets `tit_itd`

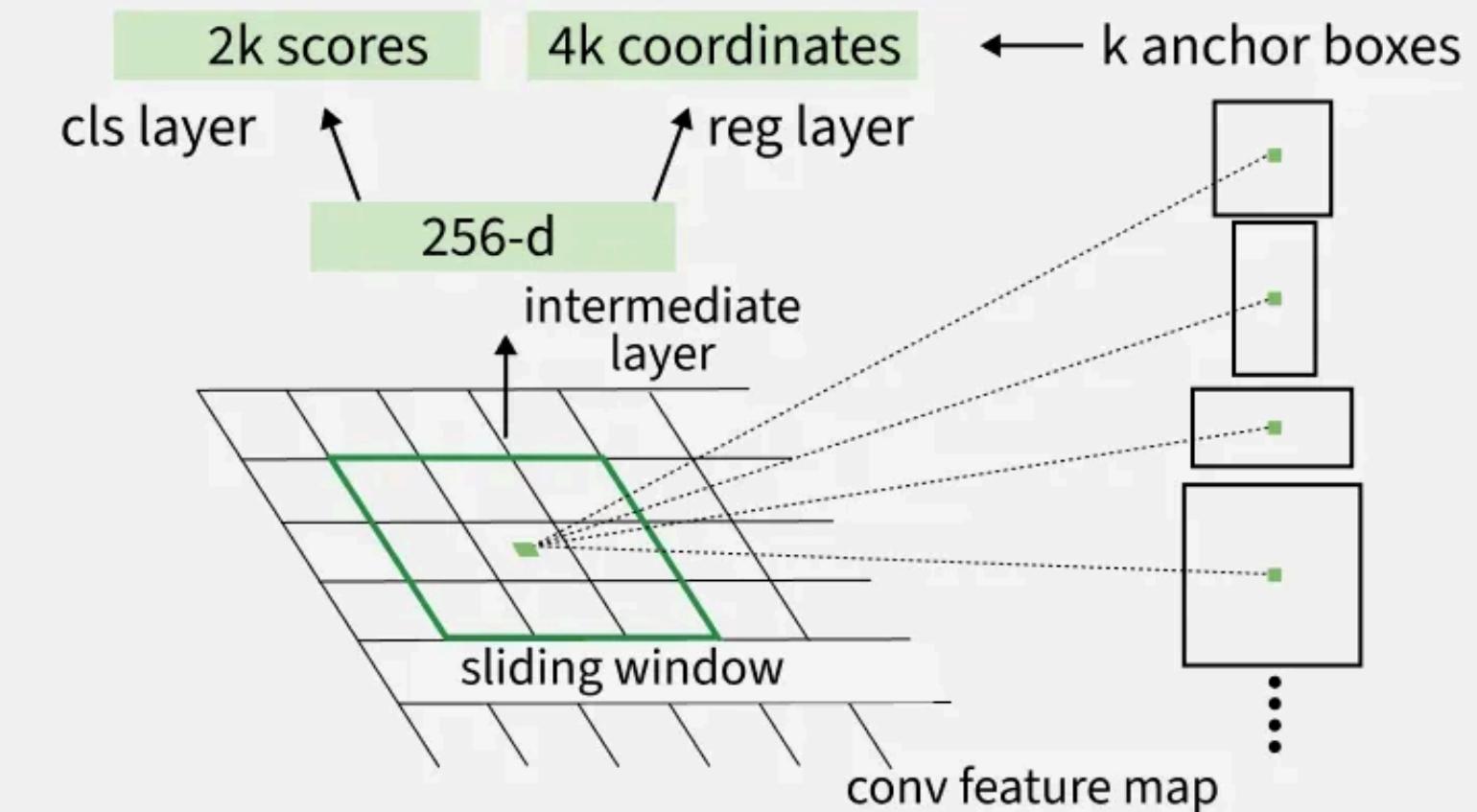
Gán nhãn dựa trên IoU với ground-truth:

$$\text{label} = \begin{cases} 1 & \text{nếu } \text{IoU} \geq 0.7 \\ 0 & \text{nếu } \text{IoU} \leq 0.3 \\ \text{ignore} & \text{trường hợp còn lại} \end{cases}$$

Loại bỏ trùng lắp bằng NMS

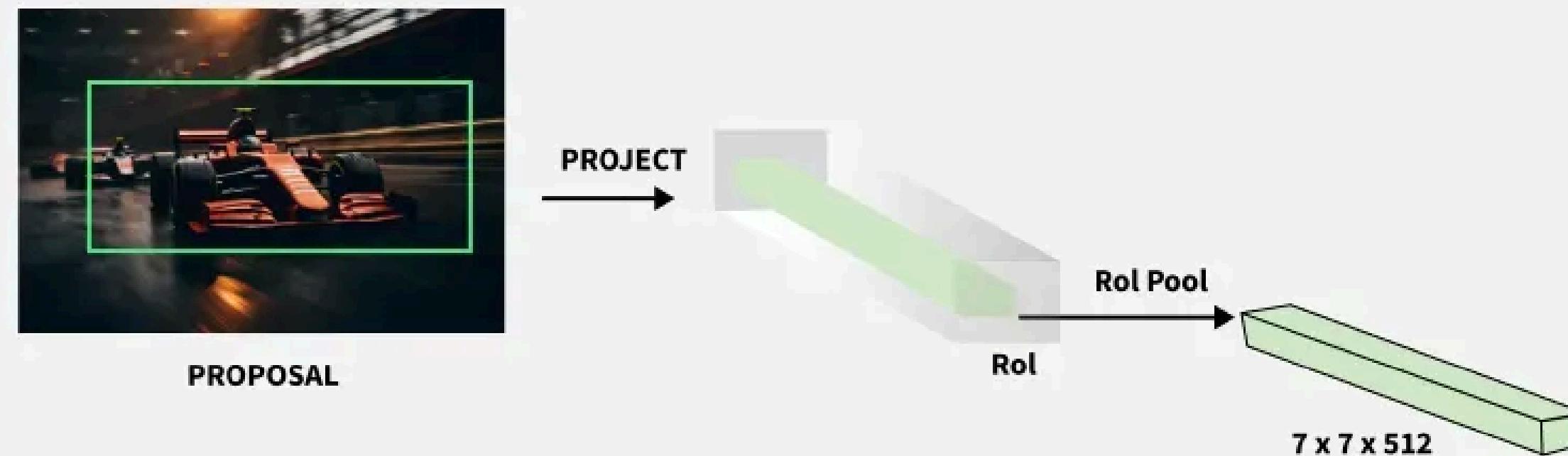
Công thức IoU:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



ROI Pooling / ROI Align

- ROI Pooling: chia proposal thành lưới cố định, max pooling
- ROI Align: cải tiến, dùng bilinear interpolation → giữ thông tin không gian chính xác



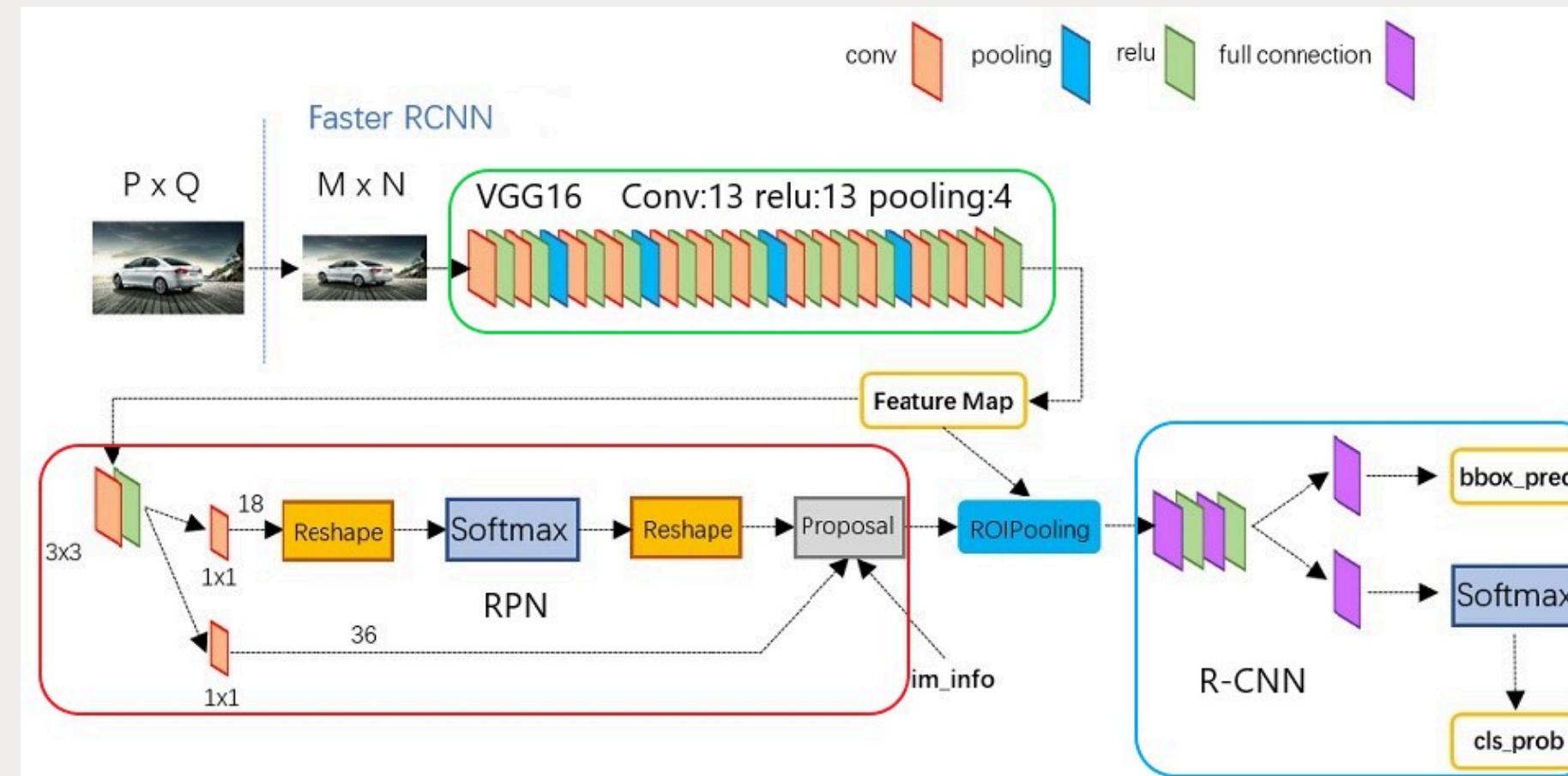
Detection Head & Multi-task Loss

- Classification: dự đoán nhãn lớp
- Bounding Box Regression: tinh chỉnh tọa độ:

$$(\hat{x}, \hat{y}, \hat{w}, \hat{h}) = (x_a + t_x w_a, y_a + t_y h_a, w_a e^{t_w}, h_a e^{t_h})$$

- Multi-task Loss:

$$L = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$



Ưu/Nhược điểm & Ứng dụng

Ưu điểm:

- Độ chính xác cao, tốt với small objects & occlusion
- Huấn luyện end-to-end
- Kiến trúc mô-đun, dễ thay backbone & tuning anchor

Nhược điểm:

- Tốc độ chậm, khó real-time
- Yêu cầu GPU mạnh, bộ nhớ lớn
- Cấu hình anchor phức tạp

Ứng dụng:

- Nhận diện phương tiện & biển số
- Hệ thống giám sát an ninh
- Ảnh y tế, video analytics

Thực nghiệm & Phân tích

Kết quả thực nghiệm với mô hình Faster R-CNN

Công cụ và thư viện sử dụng

- Python 3.10 – ngôn ngữ lập trình chính.
- PyTorch & Torchvision – xây dựng mô hình Faster R-CNN, huấn luyện và suy luận.
- FastAPI (trong các thử nghiệm triển khai) – phục vụ kiểm thử mô hình trực quan.
- NumPy, Matplotlib – xử lý số liệu và vẽ biểu đồ.
- PIL (Pillow) – đọc và xử lý ảnh

Xây dựng lớp TrafficDataset để thực hiện:

Chuẩn bị và nạp dữ liệu

- Đọc ảnh từ thư mục images/;
- Đọc nhãn từ thư mục labels/;
- Chuyển nhãn sang tensors (boxes, labels);
- Áp dụng các phép biến đổi dữ liệu (transforms):
 - Chuyển ảnh sang tensor,
 - Chuẩn hóa theo chuẩn COCO,
 - Resize về kích thước phù hợp với mô hình.

Dữ liệu được chia thành tập huấn luyện và kiểm tra theo chỉ số indexes

Cấu hình huấn luyện

- Kích thước batch: 4 (lựa chọn dựa trên giới hạn bộ nhớ GPU và đảm bảo tính ổn định của gradient).
- Optimizer: SGD với các tham số:
 - learning rate = 0.005,
 - momentum = 0.9,
 - weight decay = 0.0005.
- Scheduler: giảm learning rate theo epoch.
- Số epoch huấn luyện: thực nghiệm chạy 40 epoch.
- Theo dõi loss: lưu giá trị vào danh sách train_loss_values.
- Mỗi batch trả về bốn thành phần loss:
 - loss_classifier
 - loss_box_reg
 - loss_objectness
 - loss_rpn_box_reg

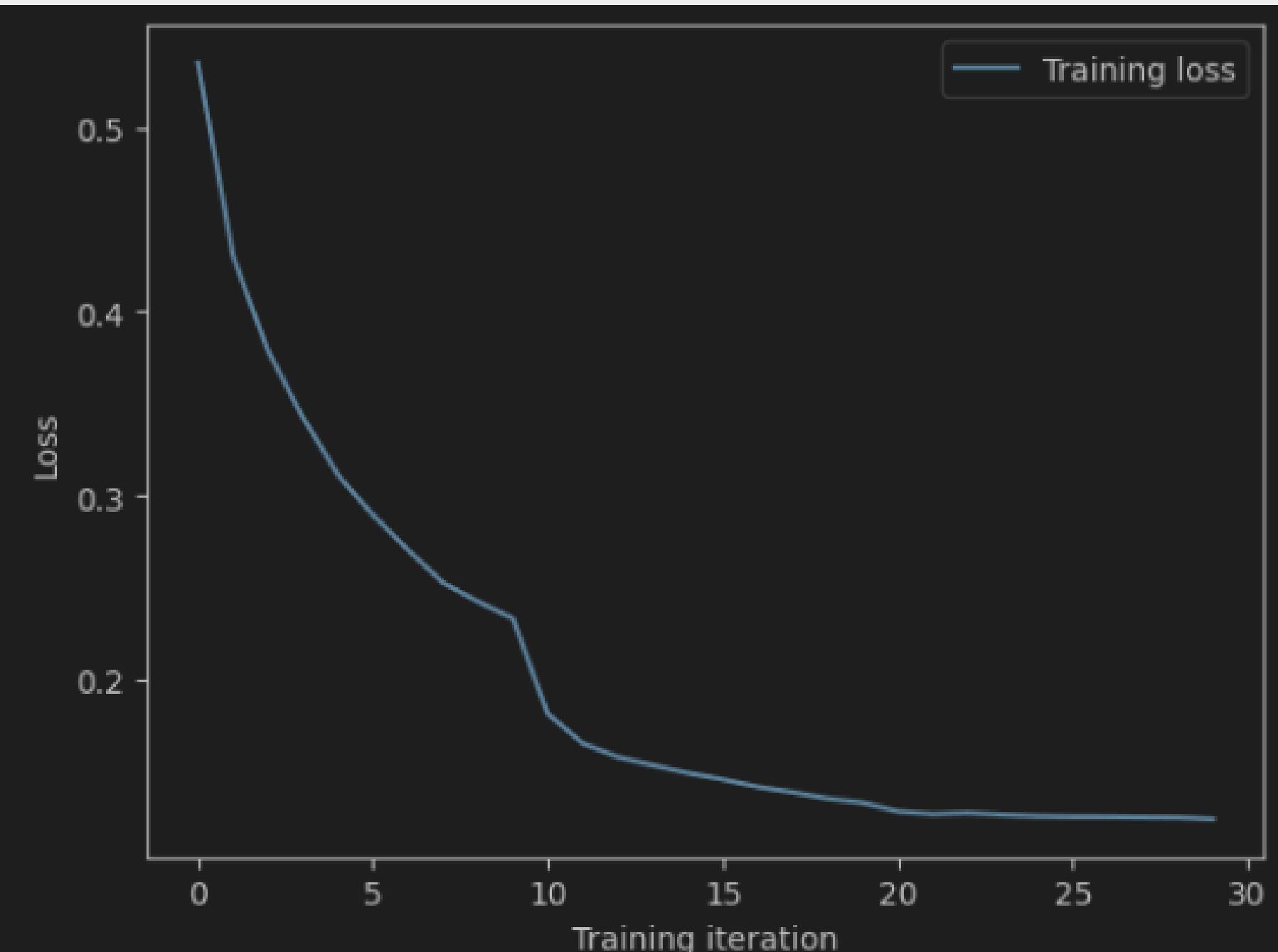
Loss tổng được sử dụng để tối ưu mô hình.

Biểu đồ hàm mất mát

Sau khi huấn luyện, trực quan hóa hàm mất mát bằng lệnh:

```
plt.plot(train_loss_values, label='Training loss')
```

Biểu đồ thể hiện loss giảm đều, chứng tỏ mô hình hội tụ tốt.



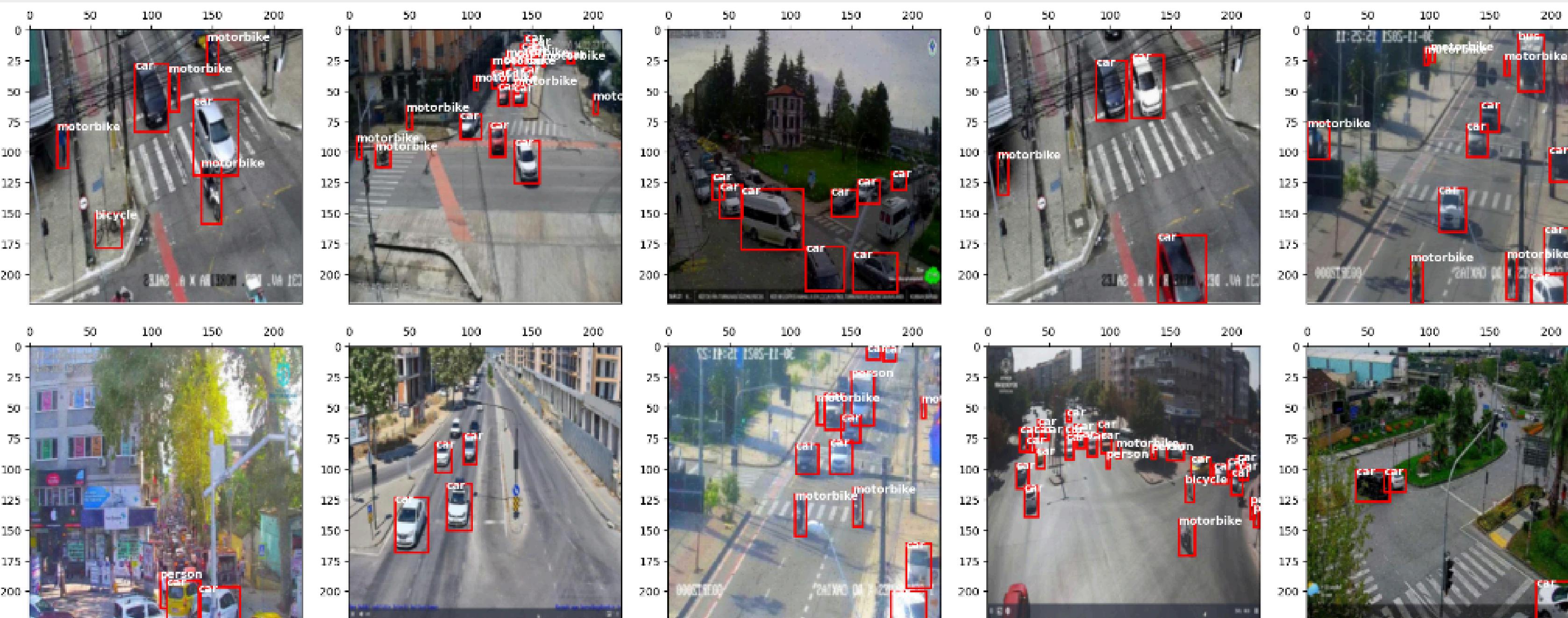
Trực quan hóa dữ liệu ground truth

Để đảm bảo dữ liệu nạp đúng, cần trực quan hóa một số ảnh bất kỳ từ tập kiểm tra:

plot_img_bbox(tensorToPIL(img), target)

Hình hiển thị bounding boxes
chuẩn giúp kiểm tra:

- Nhãn và vị trí box được đọc đúng,
- Ảnh không bị biến dạng,
- Dữ liệu phù hợp với định dạng đầu vào của mô hình.



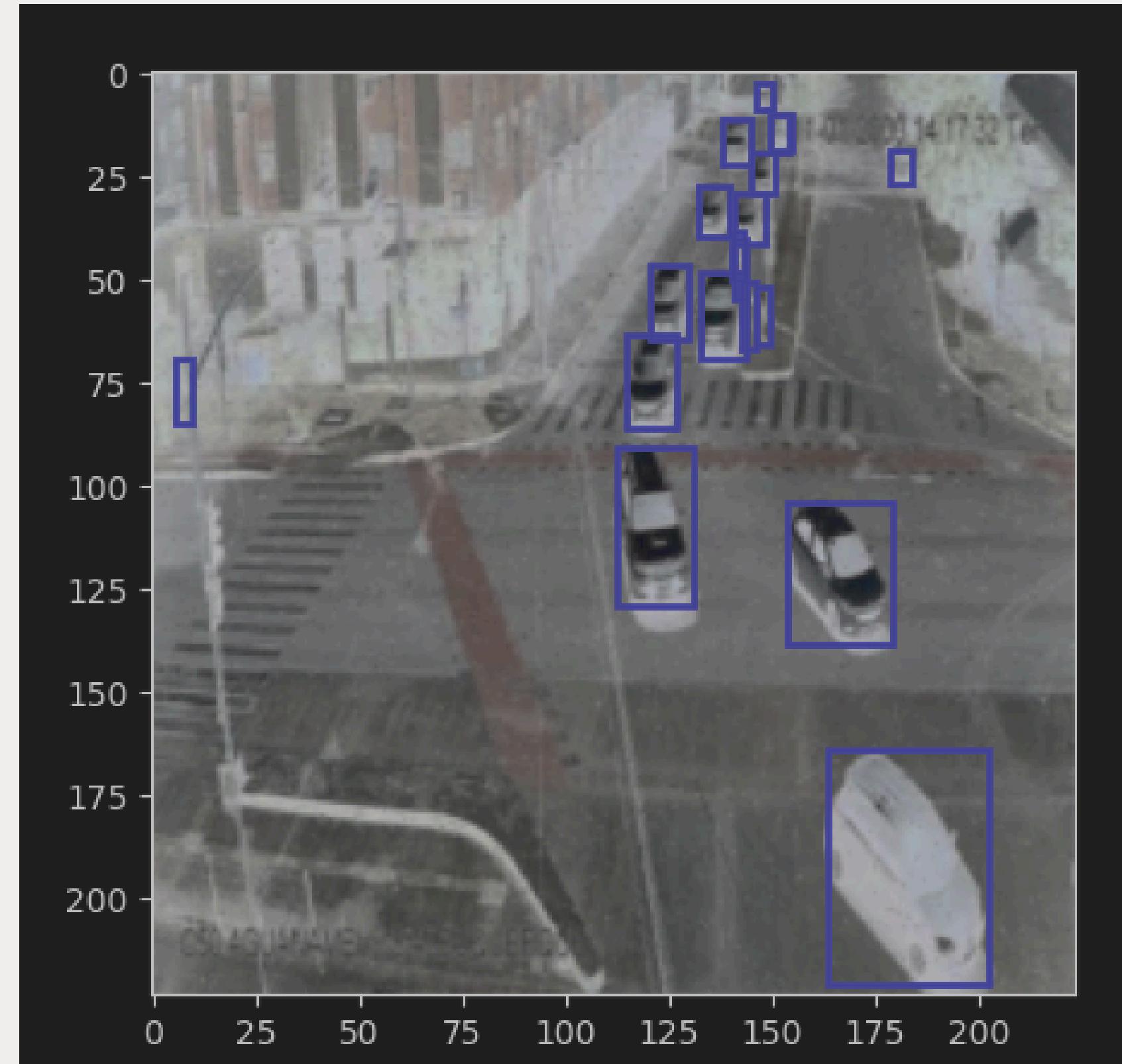
Kết quả dự đoán của mô hình (Before NMS)

Ảnh kiểm tra được đưa qua mô hình trong chế độ suy luận:

```
prediction = model([img.to(device)])[0]  
plot_img_bbox(tensorToPIL(img), prediction)
```

Trước khi áp dụng NMS, mô hình tạo ra nhiều bounding box chồng lấp nhau do:

- RPN sinh nhiều vùng đề xuất (proposals).
- Mỗi vùng được mô hình phân loại và hồi quy box.
- Các box có IoU cao nhưng chưa được lọc.



Kết quả dự đoán trước khi áp dụng NMS.
Total number of rectangle: 19

Áp dụng Non-Maximum Suppression

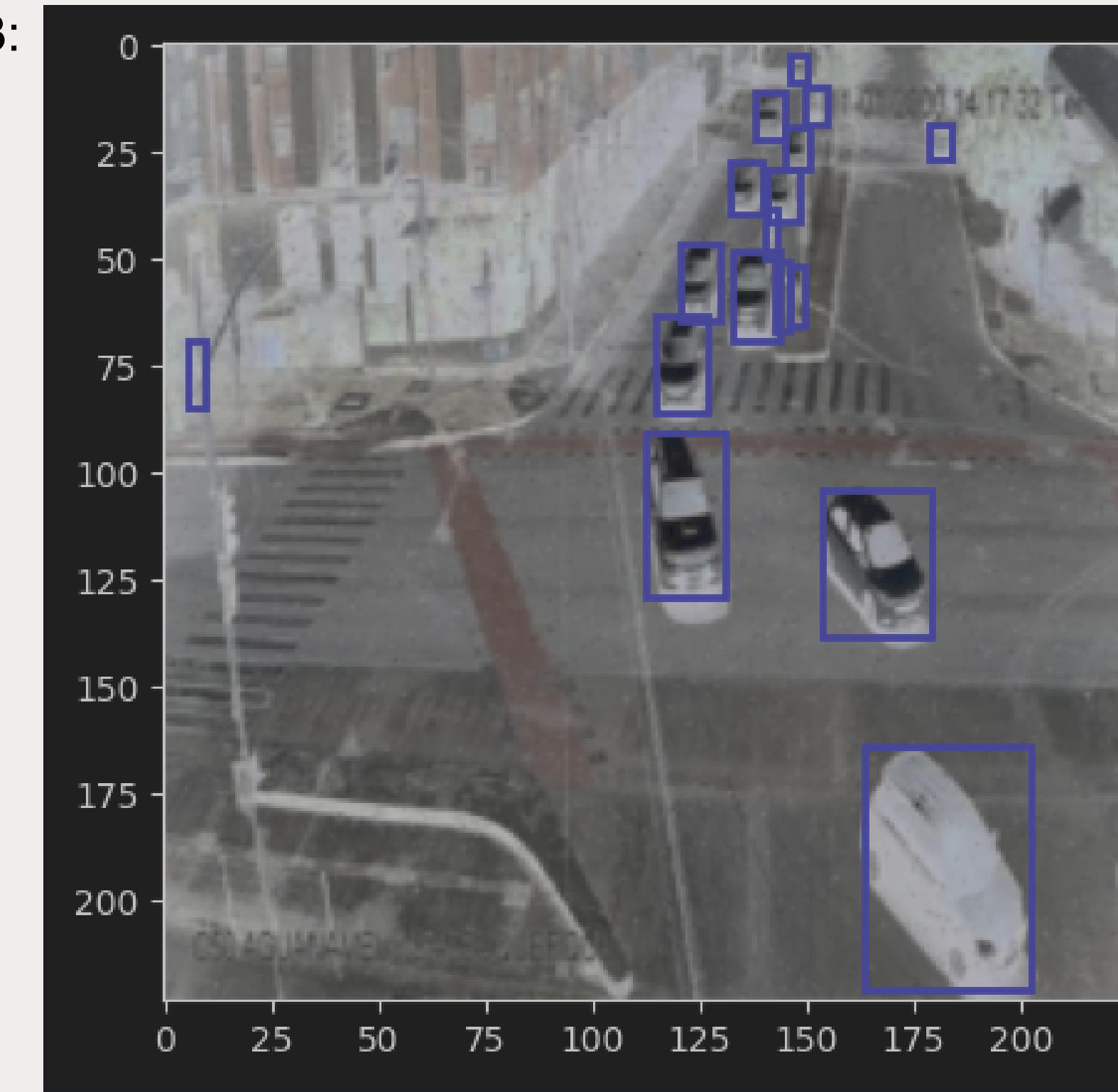
Để loại bỏ các box trùng lặp, cần áp dụng NMS với ngưỡng 0.3:

```
nms_prediction = apply_nms(prediction,  
threshold=0.3)  
  
print('predicted post nms #boxes2:',  
len(nms_prediction['labels']))  
  
plot_img_bbox(tensorToPIL(img), nms_prediction)
```

NMS giữ lại box có độ tin cậy cao nhất và loại bỏ các box có IoU lớn hơn ngưỡng.

Kết quả cho thấy:

- Số box giảm mạnh,
- Các detection còn lại sát với ground truth,
- Hình ảnh sạch và trực quan hơn.



Kết quả dự đoán sau khi áp dụng NMS.
Total number of rectangle: 17

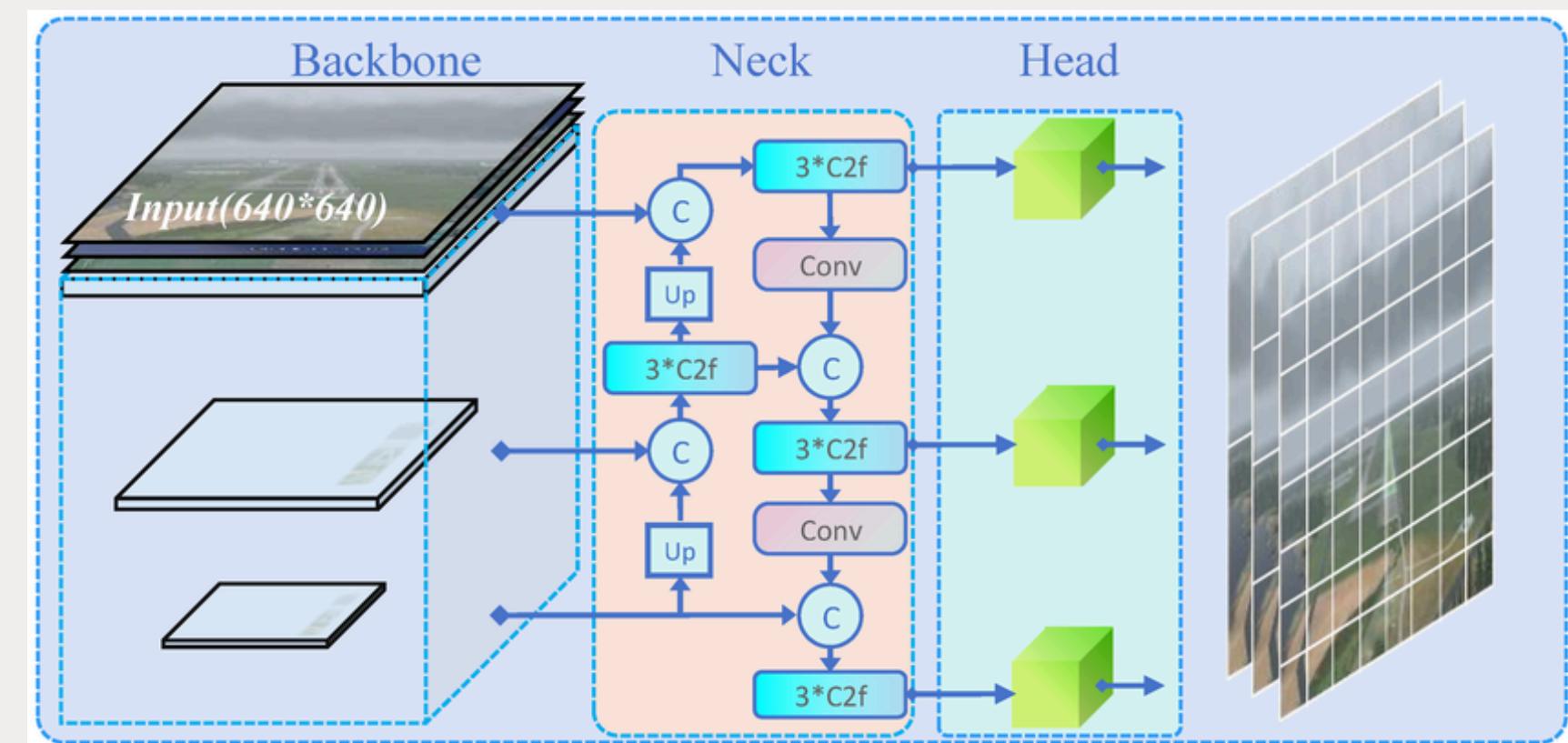
Đánh giá chung

Kết quả trực quan và biểu đồ loss cho thấy:

- Mô hình học ổn định, không bị overfitting đột ngột;
- Bounding box dự đoán có độ chính xác cao sau NMS;
- Số lượng box hợp lý và ít bị phát hiện sai lệch;
- Mô hình có thể áp dụng cho bài toán phát hiện phương tiện giao thông.

YOLOv8

- Phát triển bởi Ultralytics (2023), kế thừa YOLOv5 với nhiều cải tiến.
- One-stage → tốc độ cao, phù hợp real-time.
- Pipeline gồm 3 phần chính:
 - a. Backbone (C2f modules) – trích xuất đặc trưng
 - b. Neck (FPN+PAN cải tiến) – multi-scale fusion
 - c. Head decoupled – regression & classification tách riêng
- Anchor-free → đơn giản hơn anchor-based đời cũ.



Backbone (C2f Module)

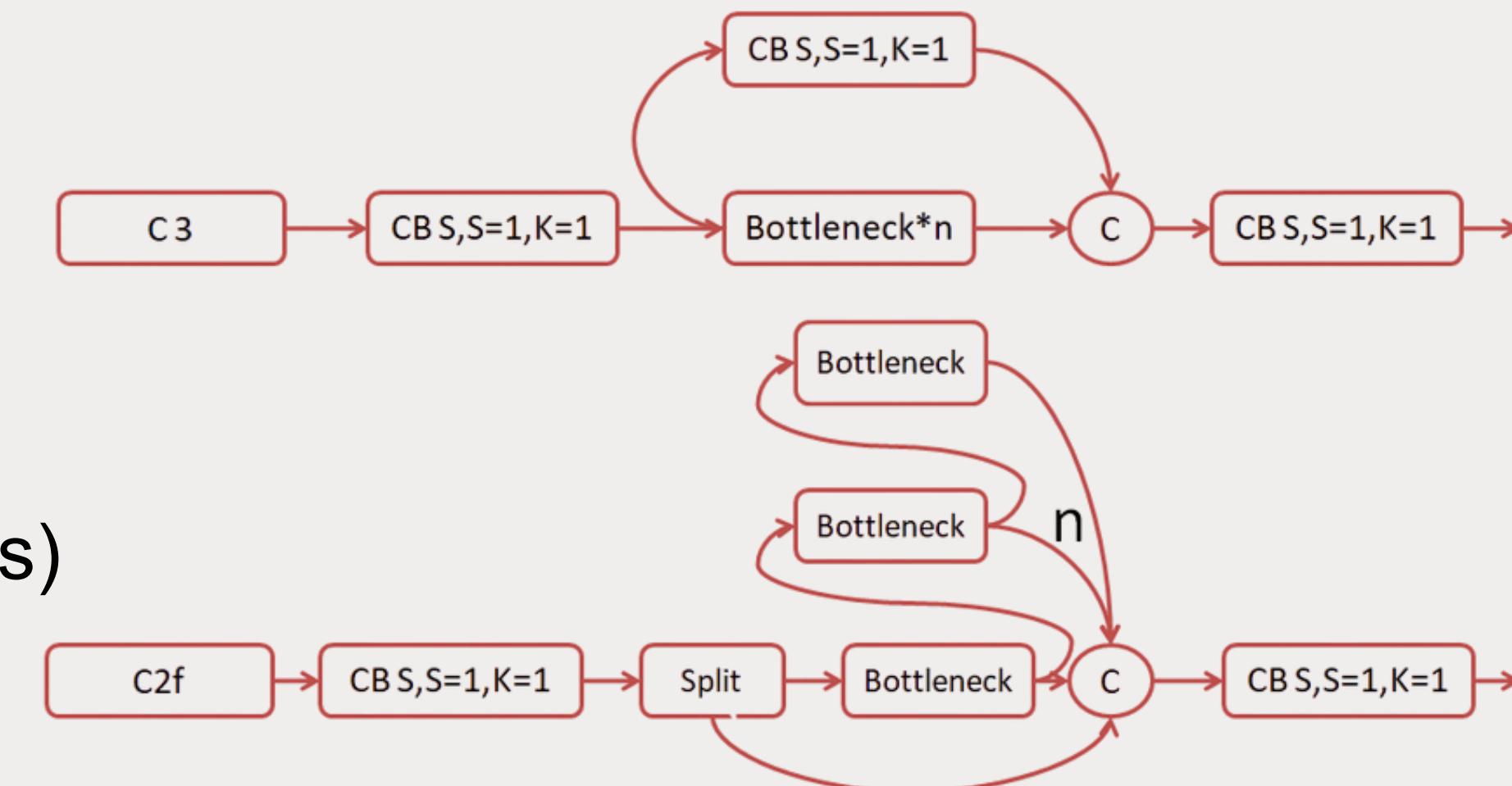
C2f là khối trích xuất đặc trưng chính của YOLOv8, được thiết kế để:

- Giữ lại thông tin gốc (nhờ shortcut)
- Học thêm đặc trưng mới (nhờ chuỗi Bottleneck)
- Giảm tính toán (so với nhiều Conv 3×3 dày đặc)

Nó là phiên bản tối ưu hơn của CSP + Bottleneck trong YOLOv5/YOLOv7.

Ưu điểm chính

- Giảm mất thông tin (nhờ shortcut)
- Tăng khả năng học đặc trưng (nhờ nhiều bottleneck)
- Gradient ổn định → train nhanh & mượt
- Nhẹ hơn các block truyền thống (ít FLOPs)



Backbone (SPPF)

SPPF : Khối trích xuất đặc trưng đa tỉ lệ giúp mô hình “nhìn” vật thể ở nhiều kích thước khác nhau.

Cơ chế hoạt động

Bước 1: MaxPool lồng nhau (stacked)

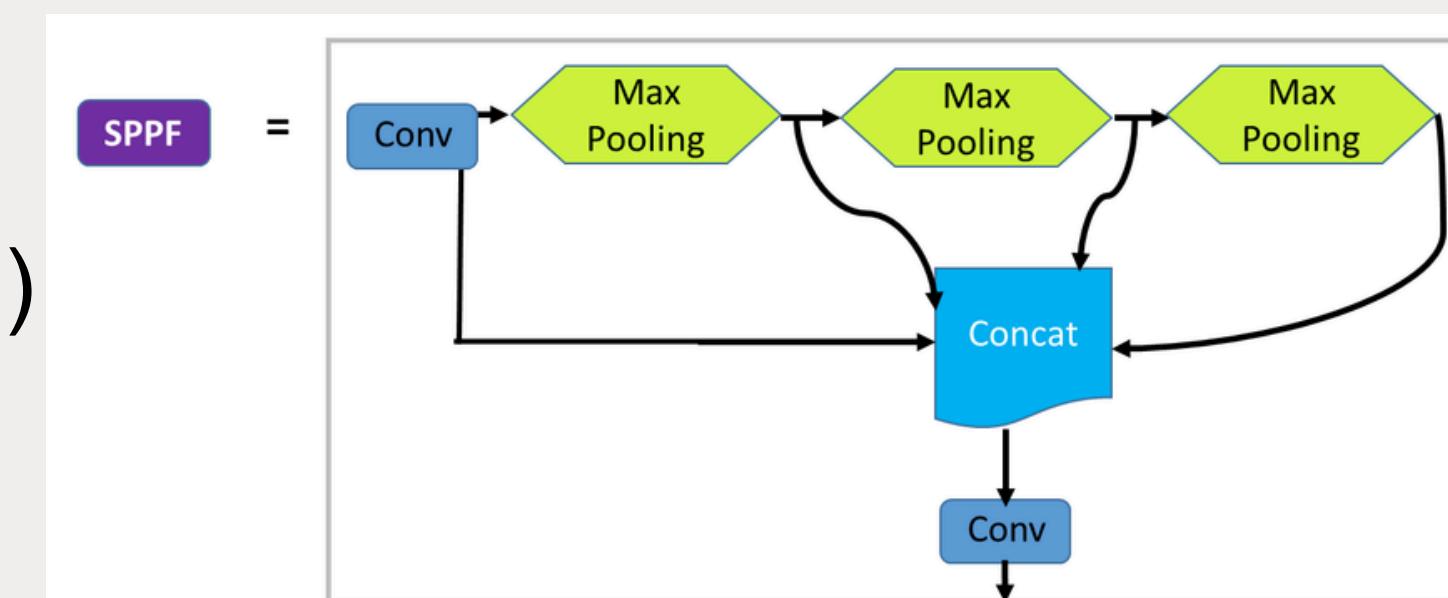
- Mỗi lần pooling → receptive field tăng dần
- Tương đương scale khác nhau (nhỏ → vừa → lớn)

Bước 2: Concat với feature map gốc

- Mỗi pixel giờ chứa thông tin đa tỉ lệ
- Giữ cùng $H \times W$ nhưng kênh tăng → chứa thông tin đa scale

Bước 3: Conv 1×1 cuối

- Trộn thông tin giữa các scale
- Giảm số kênh → gọn, hiệu quả cho layer tiếp theo



Neck (FPN + PAN)

Mục tiêu của Neck

- Kết hợp đa scale feature map từ backbone (P3, P4, P5)
- Giúp mô hình hiểu vật thể từ nhỏ → lớn
- Chuẩn bị dữ liệu tốt cho Head (dự đoán bounding box)

FPN (Feature Pyramid Network) — top-down

- Ý tưởng: đặc trưng sâu (P5) → mang ngũ cảnh → phóng to → trộn vào đặc trưng trung bình (P4)
- Tiếp tục: P4 → phóng to → trộn P3 → tạo P3'
- Giúp truyền thông tin tổng quát xuống các layer chi tiết

PAN (Path Aggregation Network) — bottom-up

- Ý tưởng: đặc trưng chi tiết (P3') → nâng lên P4' → P5'
- Giúp:
 - Bổ sung chi tiết bị mờ
 - Hiểu vật thể nhỏ, local information tốt hơn
- Kết hợp ngũ cảnh + chi tiết → feature map đa scale hoàn thiện

Head

Mục tiêu:

- Dự đoán vật thể xuất hiện ở đâu, kích thước ra sao, thuộc loại nào
- Đầu ra: bounding box (x, y, w, h), objectness score, class probability

Input:

- Feature maps đa scale từ Neck:
 - $P3'$ → small objects
 - $P4'$ → medium objects
 - $P5'$ → large objects

Ý tưởng:

- Head nhận feature map giàu thông tin → trả về dự đoán anchor-free cho mỗi pixel/cell

Bounding Box Regression (Anchor-free)

- Dự đoán tọa độ và kích thước box (x, y, w, h) dựa vào cell + stride
- Công thức:
 - $x = (\text{cell_}x + dx) * \text{stride}$
 - $y = (\text{cell_}y + dy) * \text{stride}$
 - $w = \exp(dw) * \text{stride}$
 - $h = \exp(dh) * \text{stride}$
- Giải thích:
 - $\text{cell_}x, \text{cell_}y$: vị trí ô trên feature map
 - dx, dy, dw, dh : giá trị mạng dự đoán
 - stride: tỉ lệ từ feature map → ảnh gốc
- Anchor-free:
 - Không cần anchor predefined → mô hình linh hoạt với nhiều kích thước vật thể

Bounding Box Regression (Loss)

- Objectness Prediction (Logistic Loss)

$$\mathcal{L} = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})] \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

- Class Prediction (Softmax + Cross-Entropy)

$$\mathcal{L} = -\sum_{i=1}^C y_i \log(\hat{y}_i) \quad \hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}$$

- Bounding Box Loss (CIOU)

$$\mathcal{L}_{\text{box}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}_{gt})}{c^2} + \alpha v$$

$\mathbf{b} = (x, y, w, h)$ = box dự đoán

\mathbf{b}_{gt} = box thật

ρ^2 = khoảng cách tâm 2 box

c = đường chéo bao quanh 2 box

$$v = \frac{4}{\pi^2} \left(\arctan\left(\frac{w_{gt}}{h_{gt}}\right) - \arctan\left(\frac{w}{h}\right) \right)^2$$

α = hệ số điều chỉnh

Ưu điểm, Nhược điểm & Ứng dụng

Ưu điểm

- Tốc độ cao: One-stage detector → phù hợp real-time.
- Anchor-free: Giảm độ phức tạp, generalization tốt hơn.
- Small objects: Nhận diện các đối tượng nhỏ tốt.
- Triển khai dễ dàng: Hỗ trợ PyTorch, ONNX, TensorRT, edge devices.
- Độ chính xác cao (mAP): Hội tụ nhanh nhờ decoupled head + C2f module.

Nhược điểm

- Hiệu suất giảm với objects rất nhỏ hoặc bị che khuất mạnh.
- Cần GPU mạnh cho các biến thể L/X.
- Giai đoạn đầu huấn luyện dễ không ổn định.
- Phụ thuộc vào data augmentation (Mosaic, MixUp, HSV...).
- Tốc độ giảm khi tăng kích thước ảnh input → trade-off tốc độ vs. mAP.
- Hiệu năng giảm trong điều kiện ánh sáng/ thời tiết phức tạp.

Ứng dụng trong giao thông

- Đếm và phân loại phương tiện: Ô tô, xe máy, xe buýt...
- Phát hiện vi phạm: Đi ngược chiều, vượt đèn đỏ.
- Phân tích lưu lượng giao thông: Dữ liệu từ camera AI.
- Camera giám sát an ninh: Cảnh báo nguy hiểm, tracking phương tiện.

Thực nghiệm & Phân tích

Kết quả thực nghiệm với mô hình YOLOv8

Công cụ và thư viện sử dụng

- Python 3.11 – môi trường lập trình chính.
- Ultralytics YOLOv8 – xây dựng mô hình, huấn luyện, đánh giá và suy luận.
- PyTorch 2.x – nền tảng tính toán cho mô hình YOLO.
- NumPy, Pandas – xử lý và phân tích dữ liệu.
- OpenCV và PIL – trực quan hóa ảnh và bounding boxes.
- Matplotlib – vẽ biểu đồ quá trình huấn luyện và đánh giá.
- ONNX Runtime – phục vụ triển khai mô hình sau khi xuất sang ONNX.

Mô hình chính được sử dụng là YOLOv8m (Medium), tối ưu giữa tốc độ và độ chính xác.

Chuẩn bị và nạp dữ liệu

Dữ liệu được tổ chức theo cấu trúc chuẩn YOLO:

train/

 images/

 labels/

valid/

 images/

 labels/

test/

 images/

 labels/

File data.yaml mô tả bộ dữ liệu như sau:

train: ../train/images

val: ../valid/images

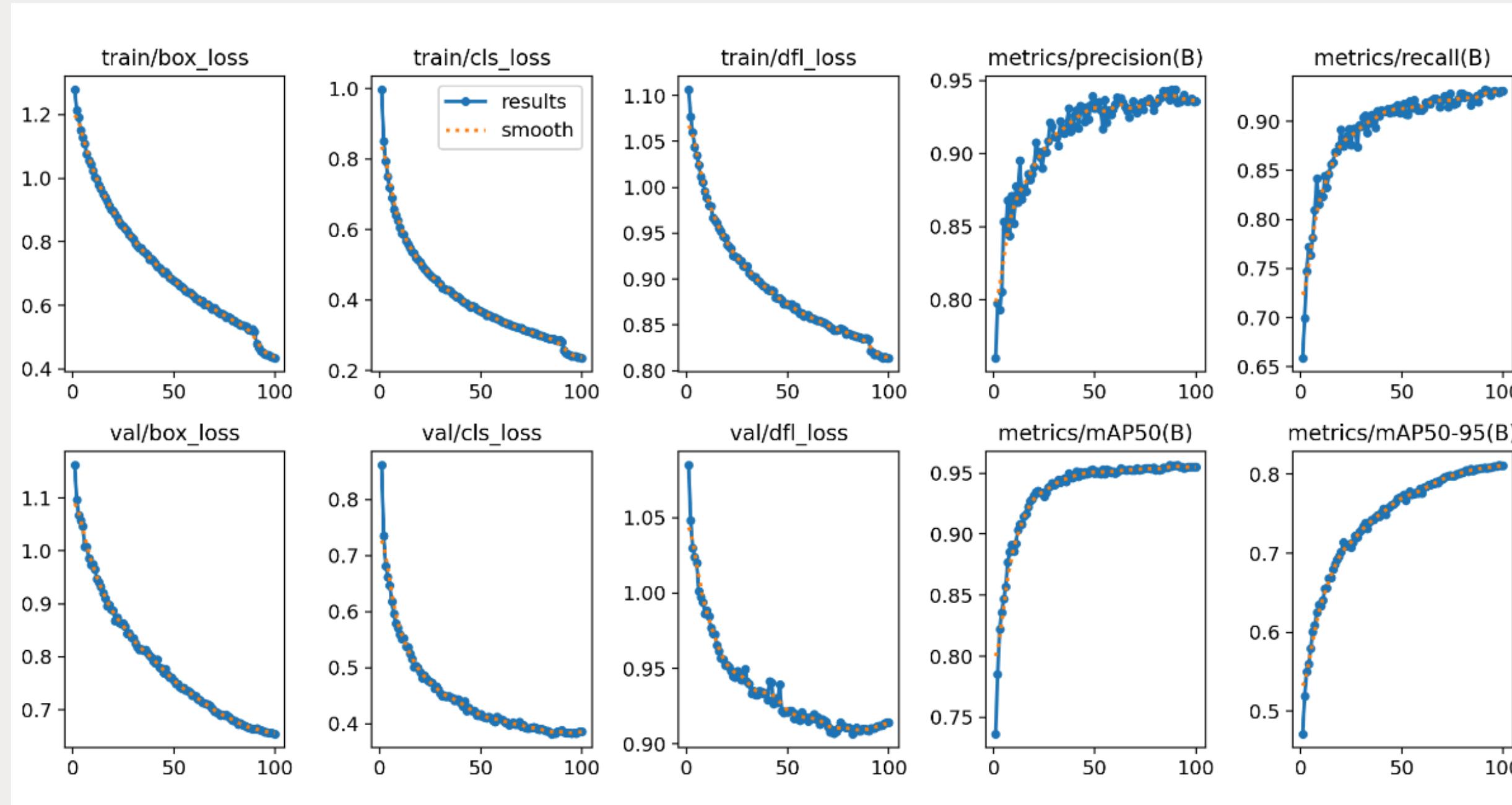
test: ../test/images

Cấu hình huấn luyện

- Model: YOLOv8m (yolov8m.pt).
- Batch size: 8.
- Image size: 640×640.
- Epochs: 100.
- Optimizer: AdamW với:
 - learning rate = 0.001,
 - weight decay = 0.0005.
- Augmentation:
 - RandAugment,
 - Random Erasing (0.4),
 - Augmentation mặc định của Ultralytics.
- Early stopping: patience = 15.
- Mixed Precision: FP16 (half = True)

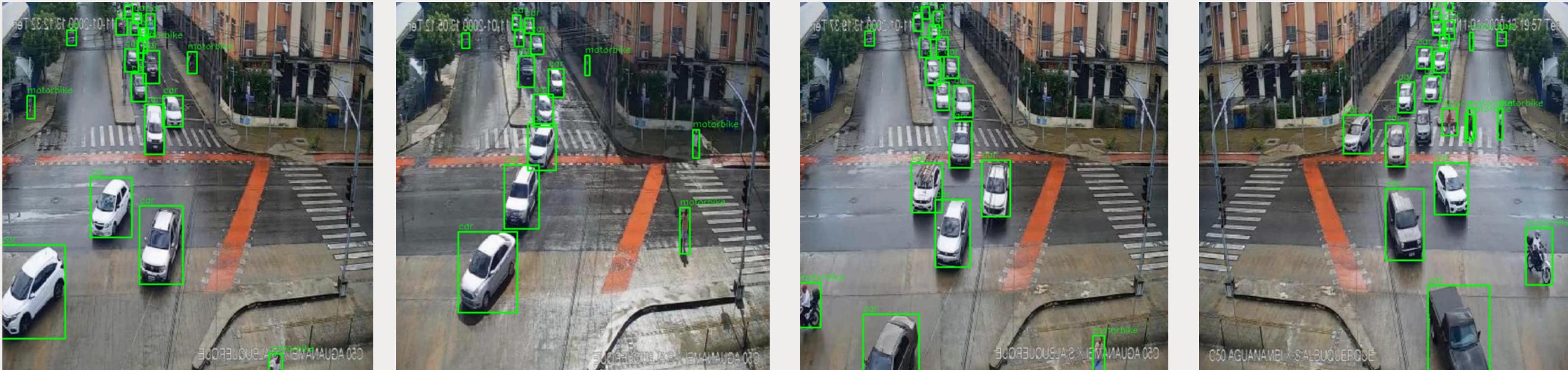
Biểu đồ hàm mất mát

- Box Loss: đánh giá sai lệch bounding box theo IoU Loss.
- Cls Loss: sai lệch phân loại.
- DFL Loss: Distribution Focal Loss, giúp ổn định dự đoán box.



Trực quan hóa dữ liệu ground truth

Để kiểm tra chất lượng dữ liệu và annotation, một số ảnh từ tập kiểm tra được trực quan hóa với bounding boxes chuẩn.



Việc trực quan hóa cho thấy nhãn và vị trí bounding box chính xác, ảnh không bị méo và phù hợp với định dạng đầu vào của mô hình.

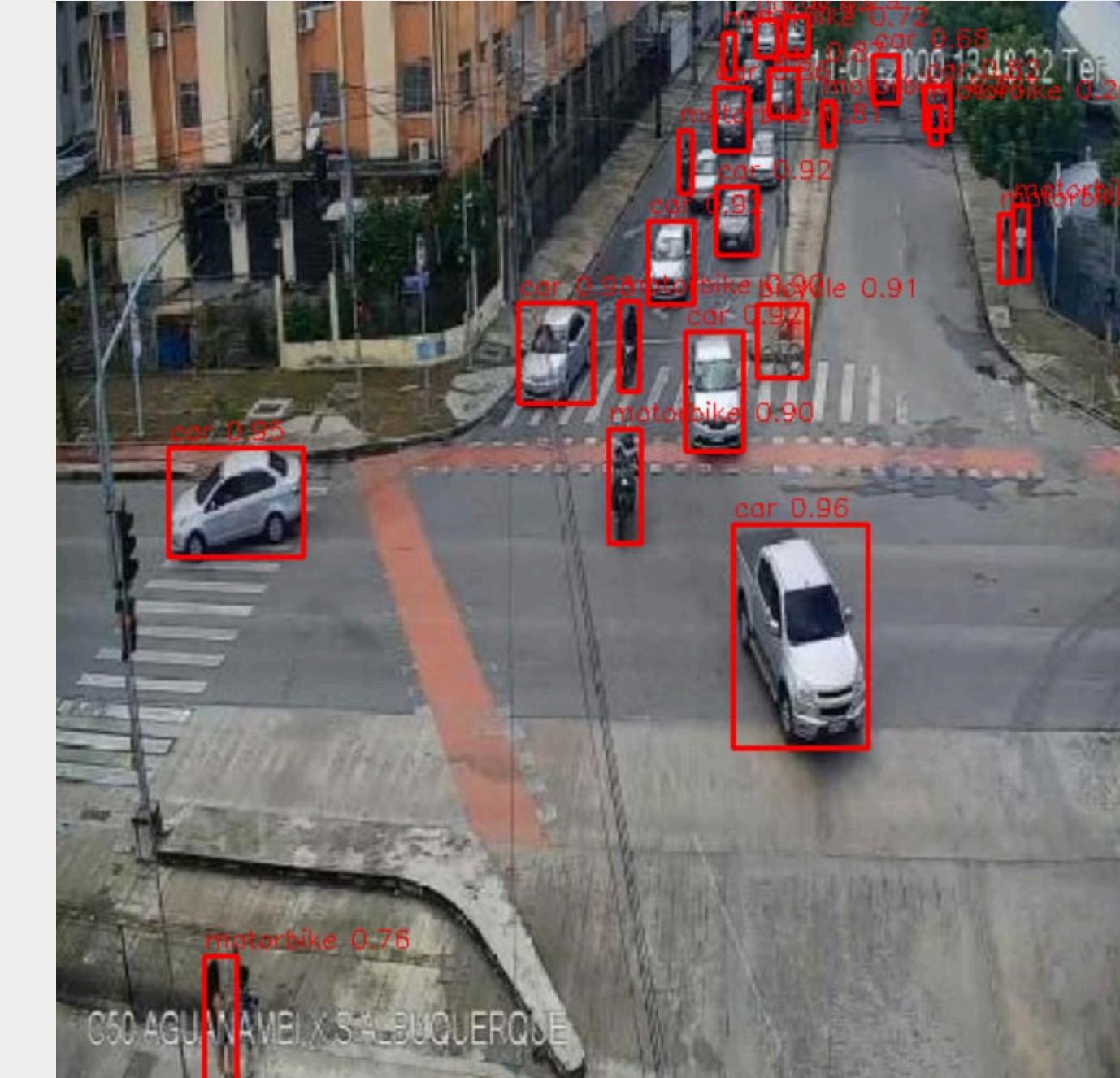
Kết quả dự đoán YOLOv8 và NMS

Before NMS

Trước khi áp dụng NMS, mô hình sinh ra nhiều bounding box trùng lặp do:

- Mỗi feature map dự đoán nhiều bounding box,
- Confidence chưa được lọc,
- Các box có IoU cao chưa được loại bỏ.

Tất cả box dự đoán, chưa lọc → nhiều box chồng nhau, thậm chí nhiều box cùng class trên 1 object.



Kết quả dự đoán YOLOv8 và NMS

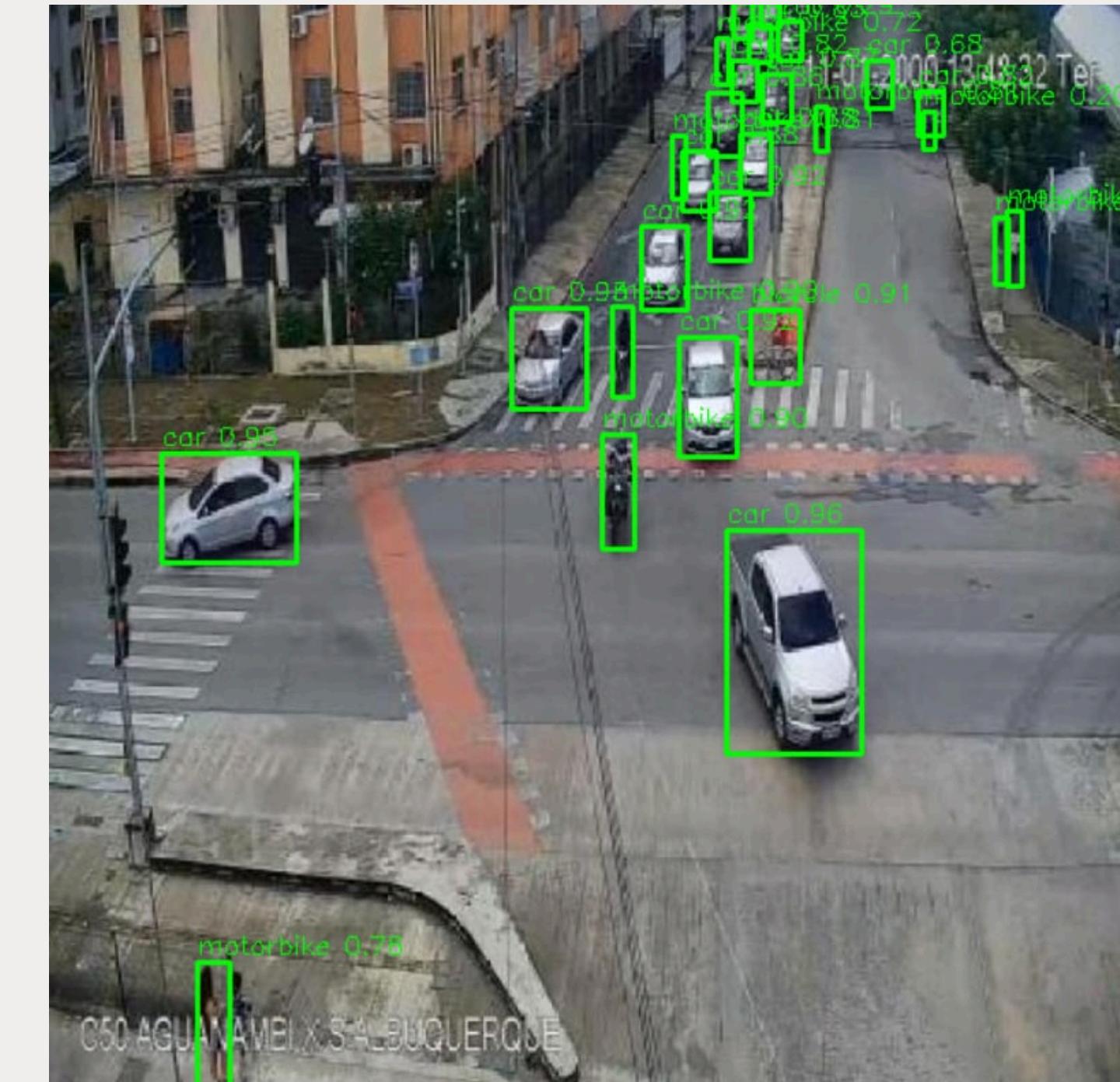
After NMS

Áp dụng NMS tự động sau khi sinh ra prediction nhằm loại bỏ các bounding box dư thừa. NMS giữ lại box có độ tin cậy cao nhất và loại bỏ:

- Các box có IoU lớn hơn ngưỡng,
 - Các detection nhiễu.

Kết quả cho thấy:

- Số lượng box giảm mạnh,
 - Bounding box trực quan hơn,
 - Kết quả sát hơn với ground truth.



Đánh giá chung

- Mô hình hội tụ ổn định, loss giảm đều, không overfitting.
- Bounding box sau NMS rõ ràng, ít nhiễu, chính xác.
- Tốc độ suy luận nhanh, phù hợp bài toán thời gian thực.
- Phát hiện tốt các phương tiện: xe đạp, xe máy, ô tô, xe buýt, người đi bộ.



YOLOv8m phù hợp cho phát hiện giao thông với độ chính xác cao và triển khai linh hoạt.

So sánh và đánh giá mô hình Faster R-CNN và YOLOv8

Các tiêu chí so sánh:

1. Accuracy (Độ chính xác)

- Faster R-CNN: hai-stage → chính xác nhưng bỏ sót vật thể nhỏ/che khuất.
- YOLOv8: YOLOv8: one-stage, backbone hiện đại → bounding box sát annotation, nhạy với vật thể nhỏ.

2. Speed (Tốc độ suy luận)

- Faster R-CNN: 300–500 ms/ảnh, không phù hợp real-time.
- YOLOv8: YOLOv8: 30–50 ms/ảnh → nhanh gấp 6–10 lần, phù hợp giám sát trực tuyến.

3. Stability (Ôn định huấn luyện & suy luận)

- Faster R-CNN: gradient dễ biến động, batch size nhỏ → overfitting.
- YOLOv8: C2f residual + SPPF + AdamW + LR scheduler → loss giảm đều, hội tụ ổn định, bounding box hợp lý.

So sánh và đánh giá

Bảng so sánh tổng quan

Mô hình	Accuracy	Speed	Stability
Faster R-CNN	Thấp	Chậm	Trung bình
YOLOv8	Cao	Nhanh	Ổn định

Nhận xét tổng quan:

- YOLOv8: vượt trội về tốc độ và độ chính xác, phù hợp real-time trên bài toán phát hiện giao thông.
- Faster R-CNN: phù hợp bài toán cần độ chính xác cao, không yêu cầu thời gian suy luận nhanh (dataset lớn, tinh).
- Khả năng triển khai: YOLOv8 hỗ trợ export sang ONNX, TensorRT, CoreML, thuận tiện cho deployment trên camera giao thông hoặc thiết bị nhúng.

Kết luận và hướng phát triển

Kết luận

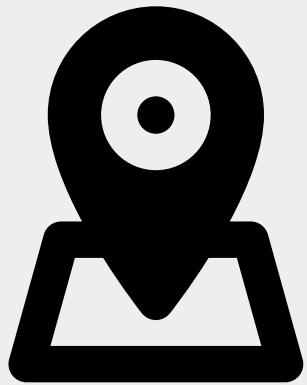
- Tốc độ và hiệu năng cao: YOLOv8 suy luận nhanh, phù hợp real-time, khắc phục nhược điểm tốc độ của Faster R-CNN.
- Độ chính xác vượt trội: Backbone C2f, SPPF, PAN giúp bounding box sát ground truth, nhạy với vật thể nhỏ và che khuất.
- Ổn định trong huấn luyện & suy luận: Loss giảm đều, mô hình hội tụ tốt, hiệu năng ổn định trên tập kiểm tra.
- Khả năng triển khai linh hoạt: Hỗ trợ ONNX, TensorRT, CoreML, dễ tích hợp camera giao thông hoặc phần mềm giám sát.



YOLOv8 tối ưu cho phát hiện phương tiện giao thông, cân bằng accuracy – speed – stability, dễ mở rộng triển khai thực tế.

Kết luận và hướng phát triển

Hướng phát triển



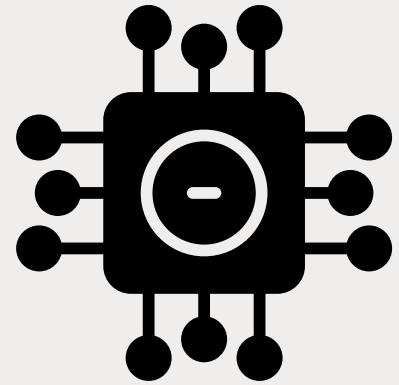
Tracking đối tượng

Kết hợp DeepSORT / ByteTrack để theo dõi hành trình phương tiện → phân tích lưu lượng, vận tốc, hành vi



Phát hiện VPGT

Nhận diện vượt đèn đỏ, đi sai làn, dừng đỗ sai quy định → kết hợp detection + rule-based hoặc deep learning.



Nhúng

Export sang TensorRT / edge device → giảm độ trễ, đáp ứng real-time.



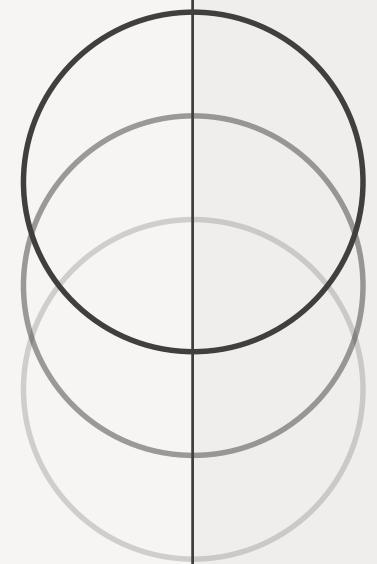
Mở rộng dataset & augmentation

Thêm dữ liệu đa môi trường, áp dụng Mosaic, MixUp → tăng khả năng generalization.



Phân tích thông minh

Liên kết detection/tracking + Big Data → dự đoán lưu lượng, phát hiện nguy hiểm, điều phối giao thông tự động.



**THANK YOU
FOR WATCHING**