

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM

KHOA CNTT
NGÀNH KHOA HỌC MÁY TÍNH

COMPUTER VISION

BÁO CÁO ĐỀ TÀI MÔN HỌC

Nhóm 24

Sinh viên thực hiện: Nguyễn Phú Sang - Lưu Chí Tài

XÂY DỰNG HỆ THỐNG ỨNG DỤNG MÔ HÌNH DEEP LEARNING YOLO ĐỂ PHÁT HIỆN ĐỐI TƯỢNG TRONG ẢNH

GVHD: Ths VÕ QUANG HOÀNG KHANG

NỘI DUNG BÁO CÁO

- 1 Giới thiệu về đề tài
- 2 Mô hình nghiên cứu
- 3 Phương pháp tiếp cận
- 4 Kết quả đạt được
- 5 Xây dựng hệ thống
- 6 Kết luận, hạn chế, hướng phát triển

1. GIỚI THIỆU VỀ ĐỀ TÀI

Trong những năm gần đây, trí tuệ nhân tạo (Artificial Intelligence – AI), đặc biệt là học sâu (Deep Learning), đã trở thành một trong những lĩnh vực công nghệ mũi nhọn, mang lại nhiều bước tiến quan trọng trong các ngành công nghiệp như y tế, giao thông, giáo dục, sản xuất, và đặc biệt là lĩnh vực xử lý ảnh và video. Một trong những nhánh quan trọng của AI là thị giác máy tính (Computer Vision), cho phép máy tính có khả năng “nhìn”, phân tích và hiểu được nội dung trong hình ảnh hoặc video tương tự như con người.

Mục tiêu

Mục tiêu chính của đề tài là xây dựng một hệ thống ứng dụng mô hình Deep Learning cụ thể là YOLO – để phát hiện các đối tượng trong ảnh một cách chính xác và nhanh chóng, đáp ứng yêu cầu của các bài toán xử lý thời gian thực trong thực tế

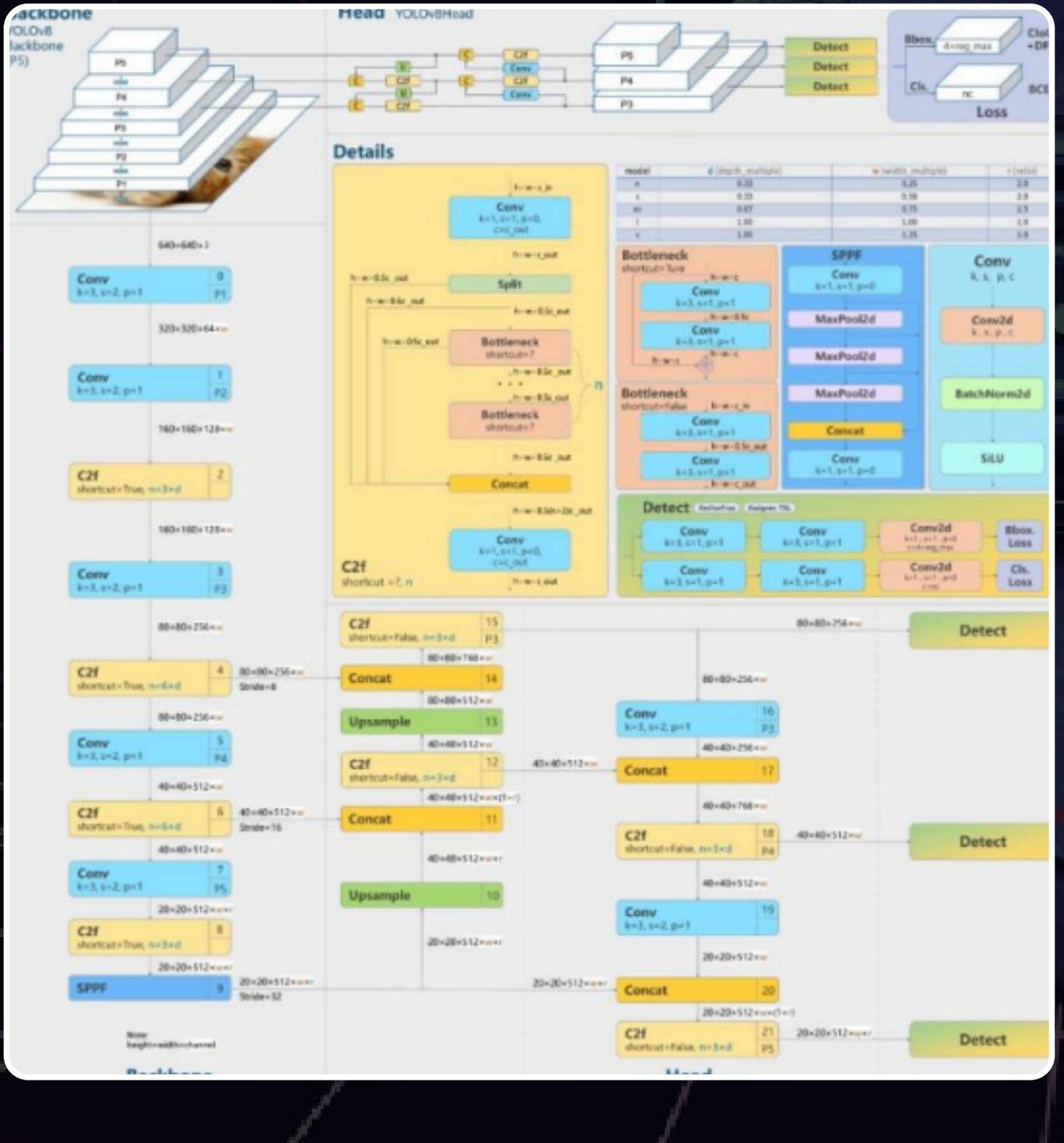
Lý do chọn đề tài

Trong bối cảnh đó, YOLO (You Only Look Once) là một trong những mô hình phát hiện đối tượng nổi bật, với ưu điểm vượt trội về tốc độ xử lý và độ chính xác cao, phù hợp với các ứng dụng yêu cầu phản hồi thời gian thực. Không giống như các mô hình trước đây sử dụng hai giai đoạn (two-stage detection như R-CNN), YOLO sử dụng kiến trúc một giai đoạn (one-stage detection), giúp giảm thiểu độ trễ và tối ưu hiệu suất

2. MÔ HÌNH NGHIÊN CỨU

Tổng quan về mô hình Yolo

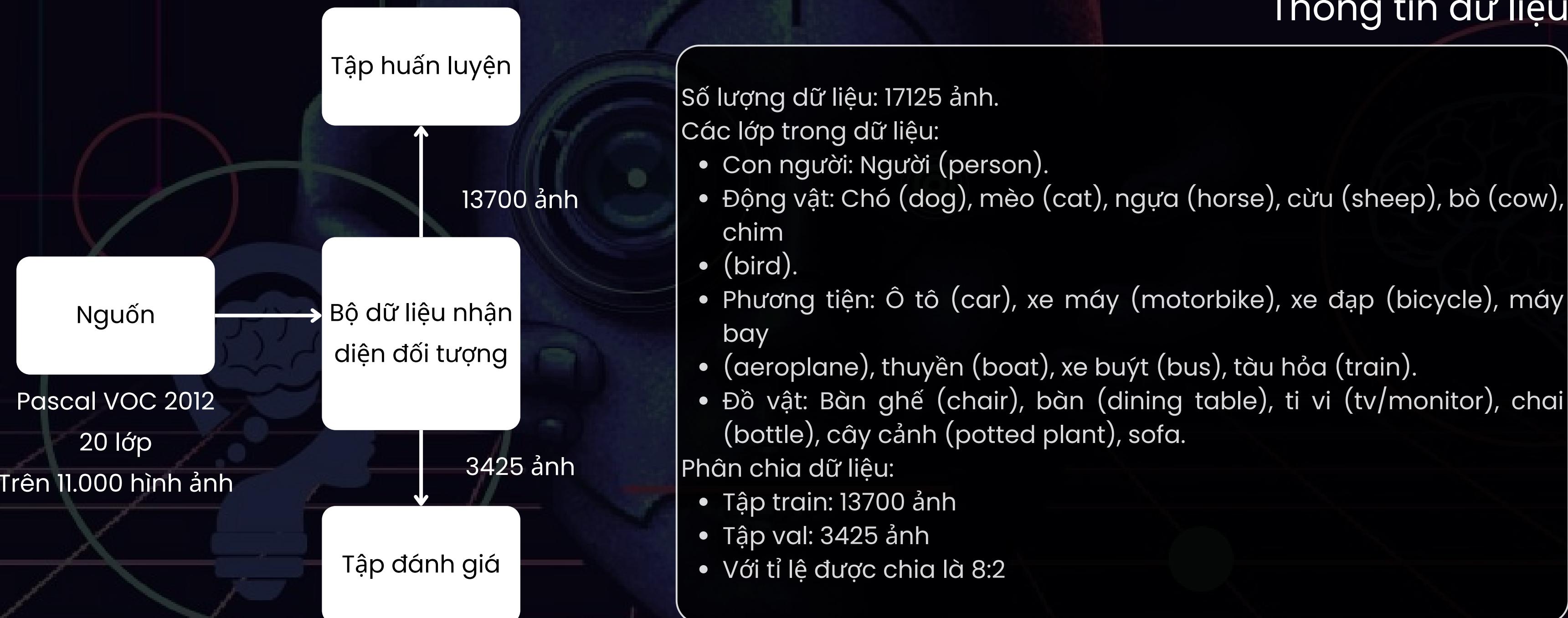
Sử dụng YOLO để phát hiện các đối tượng trong ảnh.
Phiên bản YOLO được sử dụng: YOLOv8n.
Các tiêu chí so sánh: Tốc độ, độ chính xác, và hiệu năng trên tập dữ liệu thực nghiệm.



Kiến trúc mô hình

Backbone: Sử dụng CSPDarknet53 kết hợp C2f module để tối ưu trích xuất đặc trưng.
Neck: Sử dụng PANet (Path Aggregation Network) và SPPF (Spatial Pyramid Pooling Fast) để tổng hợp các đặc trưng.
Head: Decoupled Head giúp dự đoán xác suất đối tượng, phân loại, và định vị tọa độ hộp.

3. PHƯƠNG PHÁP TIẾP CẬN



CHUẨN BỊ DỮ LIỆU

Load annotation (XML) và tạo danh sách thông tin ảnh

```
def parse_voc_annotation(xml_file):
    tree = ET.parse(xml_file)
    root = tree.getroot()

    filename = root.find("filename").text
    size = root.find("size")
    width = int(size.find("width").text)
    height = int(size.find("height").text)

    objects = []
    for obj in root.findall("object"):
        label = obj.find("name").text
        bbox = obj.find("bndbox")
        xmin = int(float(bbox.find("xmin").text))
        ymin = int(float(bbox.find("ymin").text))
        xmax = int(float(bbox.find("xmax").text))
        ymax = int(float(bbox.find("ymax").text))
        objects.append({
            "label": label,
            "bbox": [xmin, ymin, xmax, ymax]
        })

    return {
        "filename": filename,
        "width": width,
        "height": height,
        "objects": objects
    }
```

```
def load_annotations(annotation_dir):
    annotation_files = glob(os.path.join(annotation_dir, "*.xml"))
    data = []

    for xml_file in annotation_files:
        parsed_data = parse_voc_annotation(xml_file)
        data.append(parsed_data)

    return data
```

```
[ ] # Load dữ liệu
annotations_list = load_annotations(annotations_dir)
pprint(annotations_list[0])
```

```
→ {'filename': '2008_000435.jpg',
     'height': 335,
     'objects': [{'bbox': [1, 237, 447, 335], 'label': 'diningtable'},
                 {'bbox': [78, 36, 218, 240], 'label': 'person'},
                 {'bbox': [200, 12, 389, 243], 'label': 'person'},
                 {'bbox': [164, 174, 255, 246], 'label': 'person'}],
     'width': 447}
```

CHUẨN BỊ DỮ LIỆU

Kiểm tra số lượng ảnh và tệp annotation có trong bộ dữ liệu

```
num_images = len(os.listdir(images_dir))
num_annotations = len(os.listdir(annotations_dir))

print(f"Số lượng ảnh: {num_images}")
print(f"Số lượng annotation: {num_annotations}")

→ Số lượng ảnh: 17125
Số lượng annotation: 17125
```

```
# Tạo một danh sách chứa tất cả các label
all_labels = []
for annotation in annotations_list:
    for obj in annotation['objects']:
        all_labels.append(obj['label'])

# Đếm số lần xuất hiện của mỗi label
label_counts = Counter(all_labels)

# In ra các label và số lần xuất hiện
print("Các label trong dữ liệu:")
for label, count in label_counts.items():
    print(f"- {label}: {count}")
```

Thống kê các đối tượng có trong ảnh

Các label trong dữ liệu:

- diningtable: 800
- person: 17401
- chair: 3056
- boat: 1059
- car: 2492
- bottle: 1561
- motorbike: 801
- sofa: 841
- cat: 1277
- bird: 1271
- horse: 803
- dog: 1598
- tvmonitor: 893
- train: 704
- aeroplane: 1002
- pottedplant: 1202
- cow: 771
- bus: 685
- sheep: 1084
- bicycle: 837

CHUẨN BỊ DỮ LIỆU

Hiển thị ảnh kèm Bounding Boxes

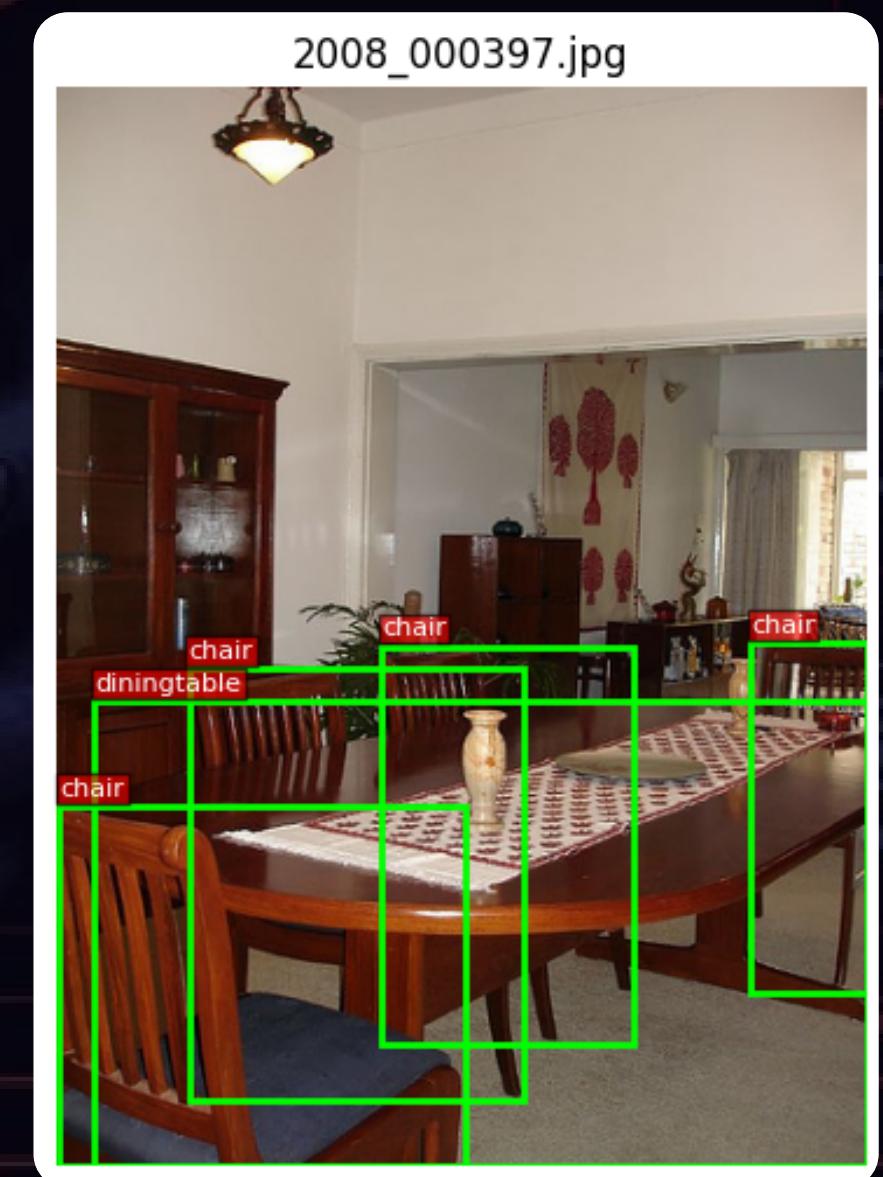
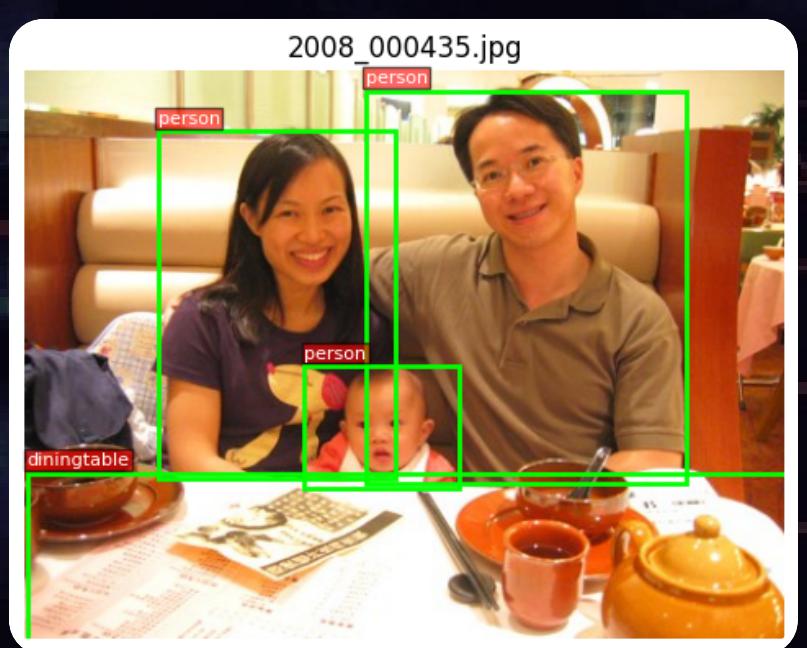
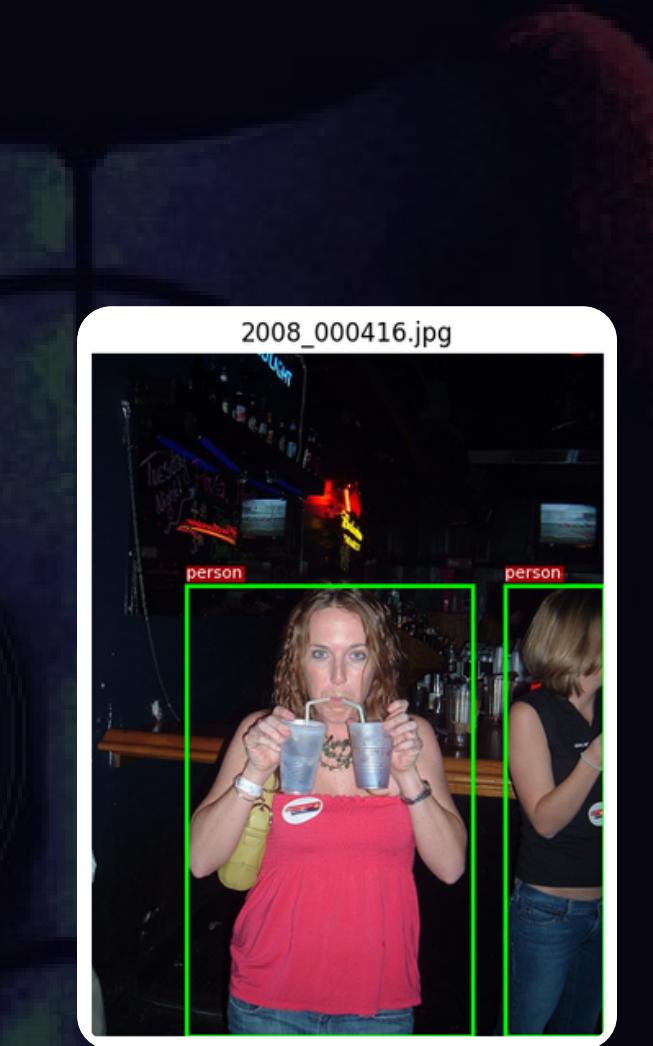
```
def show_image_with_boxes(annotation):
    image_file = os.path.join(images_dir, annotation['filename'])
    image = Image.open(image_file)

    plt.figure(figsize=(6,6))
    plt.imshow(image)
    ax = plt.gca()

    for obj in annotation['objects']:
        xmin, ymin, xmax, ymax = obj['bbox']
        ax.add_patch(plt.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin,
                                  edgecolor='lime', facecolor='none', lw=2))
        ax.text(xmin, ymin - 5, obj['label'], color='white',
                fontsize=8, bbox=dict(facecolor='red', alpha=0.6, pad=1))

    plt.axis('off')
    plt.title(annotation['filename'])
    plt.show()

# Hiển thị 3 ảnh đầu
for i in range(3):
    show_image_with_boxes(annotations_list[i])
```



TIỀN XỬ LÝ

Mã hóa nhãn

```
labels = sorted(list(label_counts.keys()))
label_dict = {label: i for i, label in enumerate(labels)}

print("Từ điển ánh xạ nhãn sang số nguyên:")
for label, idx in label_dict.items():
    print(f"- {label}: {idx}")

Từ điển ánh xạ nhãn sang số nguyên:
- aeroplane: 0
- bicycle: 1
- bird: 2
- boat: 3
- bottle: 4
- bus: 5
- car: 6
- cat: 7
- chair: 8
- cow: 9
- diningtable: 10
- dog: 11
- horse: 12
- motorbike: 13
- person: 14
- pottedplant: 15
- sheep: 16
- sofa: 17
- train: 18
- tvmonitor: 19
```

```
def create_yolo_annotation(xml_file_path, yolo_label_path, label_dict):
    tree = ET.parse(xml_file_path)
    root = tree.getroot()
    annotations = []

    img_width = int(root.find('size/width').text)
    img_height = int(root.find('size/height').text)

    for obj in root.findall('object'):
        label = obj.find('name').text
        if label not in label_dict:
            continue
        label_idx = label_dict[label]
        bndbox = obj.find('bndbox')
        xmin = float(bndbox.find('xmin').text)
        ymin = float(bndbox.find('ymin').text)
        xmax = float(bndbox.find('xmax').text)
        ymax = float(bndbox.find('ymax').text)

        x_center = ((xmin + xmax) / 2) / img_width
        y_center = ((ymin + ymax) / 2) / img_height
        width = (xmax - xmin) / img_width
        height = (ymax - ymin) / img_height

        annotations.append(f"{label_idx} {x_center:.6f} {y_center:.6f} {width:.6f} {height:.6f}")

    with open(yolo_label_path, 'w') as f:
        f.write("\n".join(annotations))
```

CHUYỂN ĐỔI ANNOTATIONS PASCAL_VOC SANG ĐỊNH DẠNG YOLO

TIỀN XỬ LÝ

PHÂN TÁCH DỮ LIỆU TRAIN, VAL VỚI TỈ LỆ 8:2

```
random.seed(42)
random.shuffle(image_ids)
split_index = int(0.8 * len(image_ids))
train_ids = image_ids[:split_index]
val_ids = image_ids[split_index:]

print(f"Số lượng ảnh train: {len(train_ids)}")
print(f"Số lượng ảnh validation: {len(val_ids)}")
```

Số lượng ảnh train: 13700
Số lượng ảnh validation: 3425

```
for image_set, ids in [('train', train_ids), ('val', val_ids)]:
    for img_id in ids:
        img_src_path = os.path.join(images_dir, f'{img_id}.jpg')
        label_dst_path = os.path.join(yolo_dataset_path, 'labels', image_set, f'{img_id}.txt')

        xml_file_path = os.path.join(annotations_dir, f'{img_id}.xml')
        if not os.path.exists(xml_file_path):
            print(f"Warning: Annotation {xml_file_path} not found, skipping.")
            continue
        create_yolo_annotation(xml_file_path, label_dst_path, label_dict)

        img_dst_path = os.path.join(yolo_dataset_path, 'images', image_set, f'{img_id}.jpg')
        shutil.copy(img_src_path, img_dst_path)
```

LẮP QUA DATASET ĐỂ CHIA DỮ LIỆU THÀNH TẬP ĐÀO TẠO VÀ XÁC THỰC

```
yaml_content = """
train: {os.path.join(yolo_dataset_path, 'images/train')}
val: {os.path.join(yolo_dataset_path, 'images/val')}

nc: {len(label_dict)}
names: {list(label_dict.keys())}
"""

with open(os.path.join(yolo_dataset_path, 'data.yaml'), 'w') as f:
    f.write(yaml_content)
```

CẤU HÌNH CẤU TRÚC BỘ DỮ LIỆU TRONG TỆP
data.yaml

HUẤN LUYỆN MÔ HÌNH

```
▶ from ultralytics import YOLO  
  
# Tải mô hình yolov8  
model = YOLO('yolov8n.pt')  
  
→ Creating new Ultralytics Settings v0.0.6 file ✓  
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'  
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://d...  
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8n.pt to 'yolov8n.pt'...  
100%|██████████| 6.25M/6.25M [00:00<00:00, 17.9MB/s]
```

Tải mô hình yolov8

HUẤN LUYỆN MÔ HÌNH

```
import time  
start_time = time.time()  
  
model.train(  
    data=os.path.join(yolo_dataset_path, 'data.yaml'),  
    epochs=5,  
    imgsz=640,  
    batch=32,  
    name='yolov8_pascal_voc2012'  
)  
end_time = time.time()  
print(f"Thời gian huấn luyện: {end_time - start_time:.2f} giây")
```

Thời gian huấn luyện

Sau khi training với 5 epoch, mỗi epoch mất khoảng 3.540 – 3.780 giây với tổng thời gian đào tạo là 22254.33 giây (tương đương 6 giờ 18 phút)

ĐÁNH GIÁ MÔ HÌNH

```
best_model = YOLO('/content/drive/MyDrive/Computer-vision-(TGMT & UD)/Final (CK)/Model-save/yolov8n_pascal_voc2012_pro.pt')
metrics = best_model.val()

→ Ultralytics 8.3.137 🚀 Python-3.11.12 torch-2.6.0+cu124 CPU (Intel Xeon 2.20GHz)
Model summary (fused): 72 layers, 3,009,548 parameters, 0 gradients, 8.1 GFLOPs
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100%|██████████| 755k/755k [00:00<00:00, 4.13MB/s]
val: Fast image access ✓ (ping: 0.8±0.3 ms, read: 0.2±0.1 MB/s, size: 91.4 KB)
val: Scanning /content/drive/MyDrive/Computer-vision-(TGMT & UD)/Final (CK)/Dataset-voc2012/YOLO-DATASET/labels/val.cache... 6154 images, 0 backgrounds, 0 corrupt: 100%|██████████| 6154/6154 [00:00<?, ?it/s]
      Class   Images Instances   Box(P)      R    mAP50  mAP50-95: 100%|██████████| 385/385 [10:50<00:00,  1.69s/it]
        all     6154    14355    0.712    0.594    0.653    0.489
      aeroplane    253     346    0.878    0.725    0.808    0.655
      bicycle     235     329    0.624    0.584    0.604    0.469
      bird       270     432    0.817    0.546    0.643    0.447
      boat       182     350    0.697    0.411    0.494    0.325
      bottle      282     571    0.604    0.342    0.394    0.264
      bus        154     229    0.808    0.755    0.801    0.686
      car        470     925    0.814    0.472    0.608    0.449
      cat        434     479    0.787    0.834    0.881    0.746
      chair       486    1050    0.533    0.423    0.454     0.3
      cow         120     278    0.565    0.687    0.655    0.461
      diningtable   247     297    0.601    0.619    0.618    0.454
      dog         509     606    0.722    0.774    0.793    0.628
      horse        190     291    0.648    0.551    0.625    0.491
      motorbike     193     264    0.702    0.696    0.738    0.544
      person       3425    6200    0.793    0.723    0.781    0.57
      pottedplant   212     410     0.61    0.337    0.401    0.239
      sheep        139     430    0.696    0.598    0.655    0.452
      sofa         254     288    0.733    0.543    0.654    0.526
      train        211     254    0.835    0.748    0.834    0.633
      tvmonitor     227     326    0.778    0.512    0.622    0.442
Speed: 0.6ms preprocess, 53.4ms inference, 0.0ms loss, 0.7ms postprocess per image
Results saved to runs/detect/val
```

ĐÁNH GIÁ MÔ HÌNH

Chỉ số đánh giá mô hình

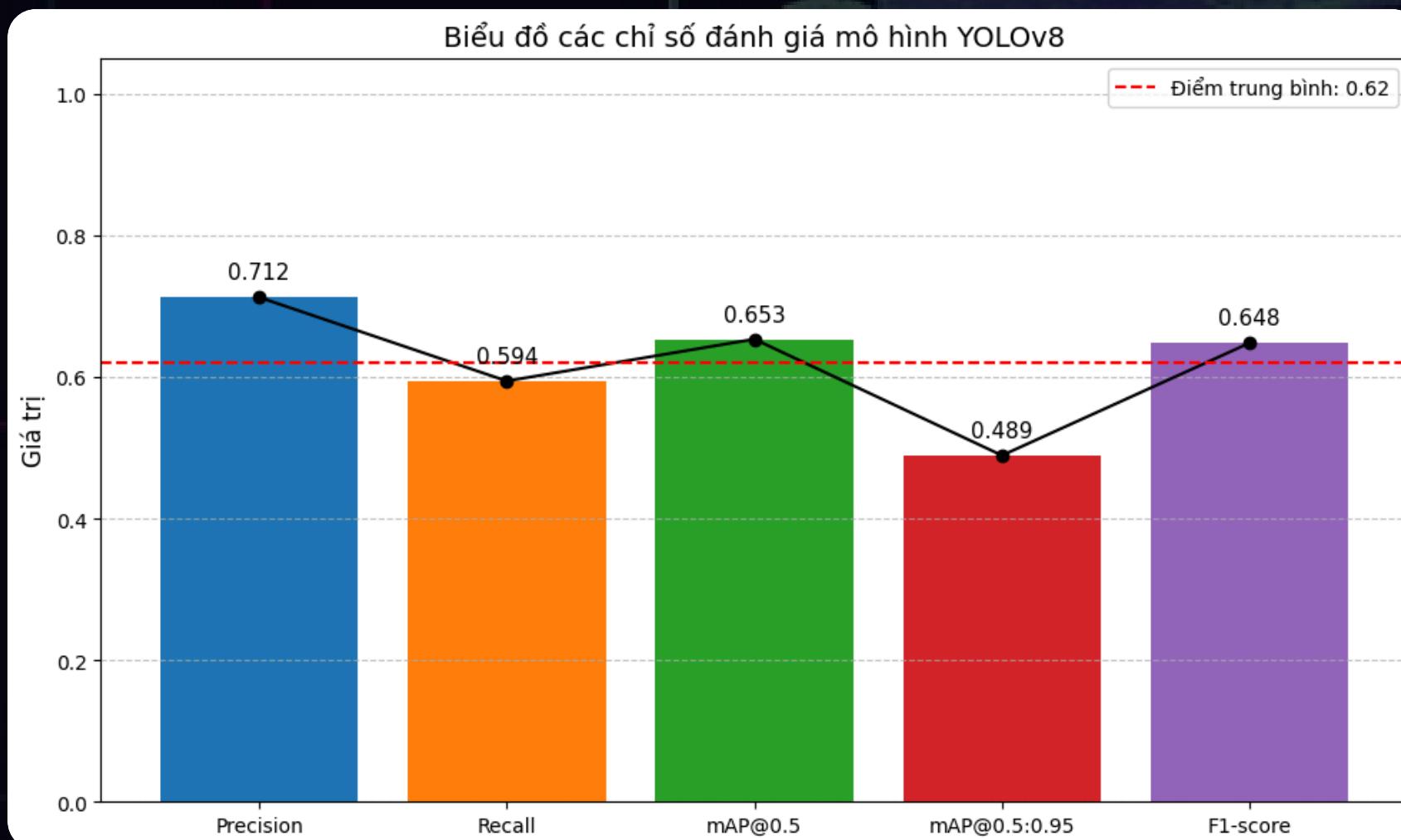
```
▶ # Lấy các chỉ số từ metrics
precision = metrics.results_dict['metrics/precision(B)']
recall = metrics.results_dict['metrics/recall(B)']
map50 = metrics.results_dict['metrics/mAP50(B)']
map5095 = metrics.results_dict['metrics/mAP50-95(B)']
f1_score = 2 * (precision * recall) / (precision + recall + 1e-6)
```

```
# In bảng kết quả
print("Bảng kết quả đánh giá mô hình:\n")
print(f"{'Chỉ số':<50}{{'Giá trị':>50}")
print("-" * 57)
print(f"{'Độ chính xác (Precision)':<51}{precision:.4f}")
print(f"{'Độ phủ (Recall)':<51}{recall:.4f}")
print(f"{'mAP@0.5':<51}{map50:.4f}")
print(f"{'mAP@0.5:0.95':<51}{map5095:.4f}")
print(f"{'Điểm F1':<51}{f1_score:.4f}")
```

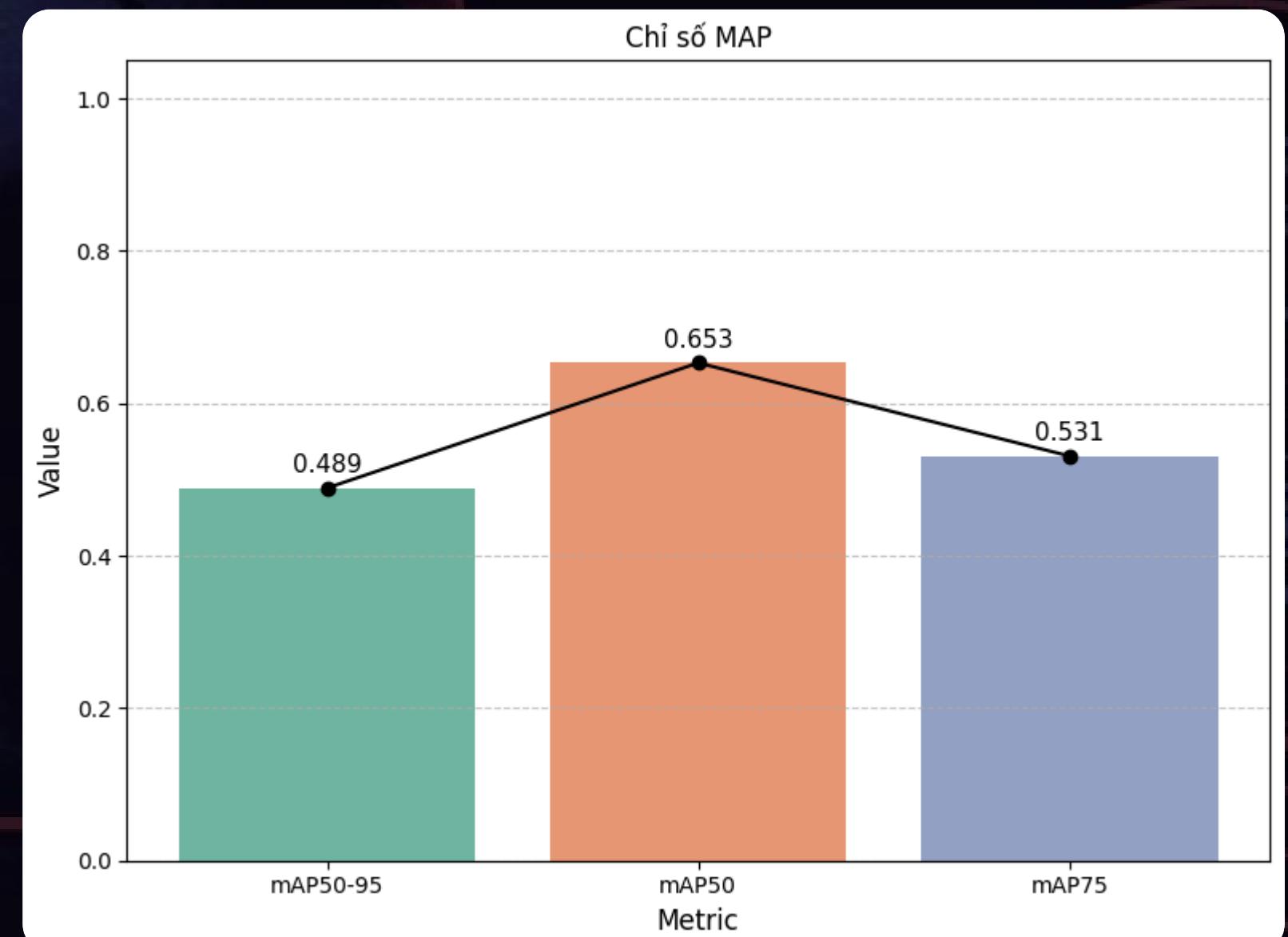
➡ Bảng kết quả đánh giá mô hình:

Chỉ số	Giá trị
Độ chính xác (Precision)	0.7122
Độ phủ (Recall)	0.5940
mAP@0.5	0.6531
mAP@0.5:0.95	0.4890
Điểm F1	0.6478

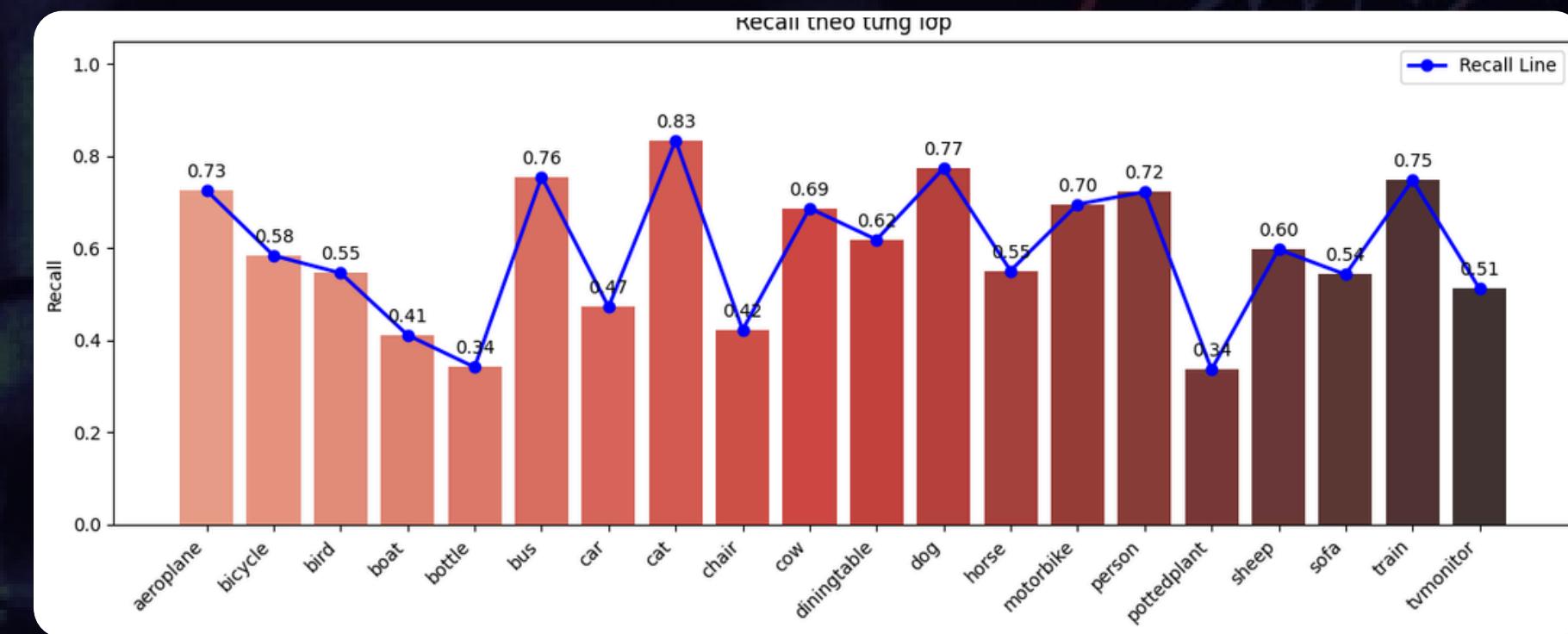
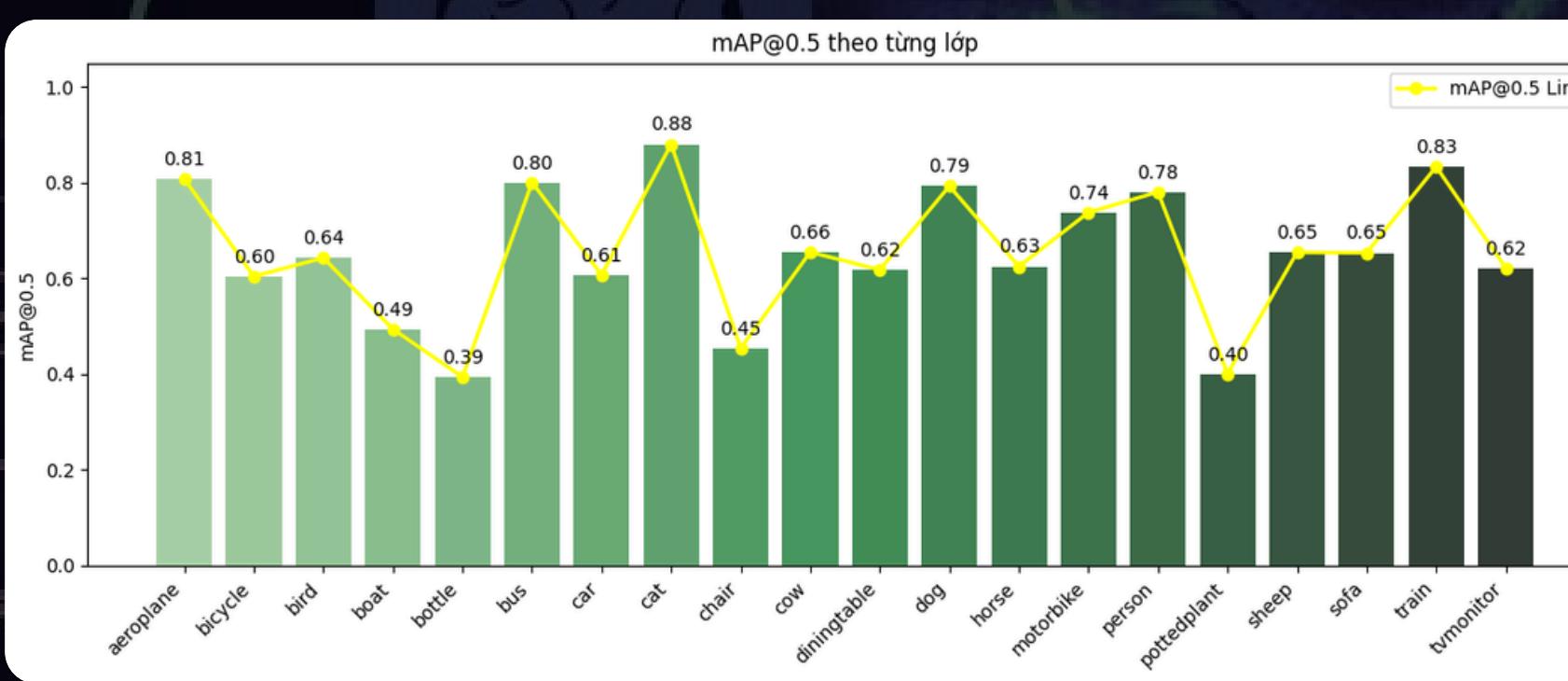
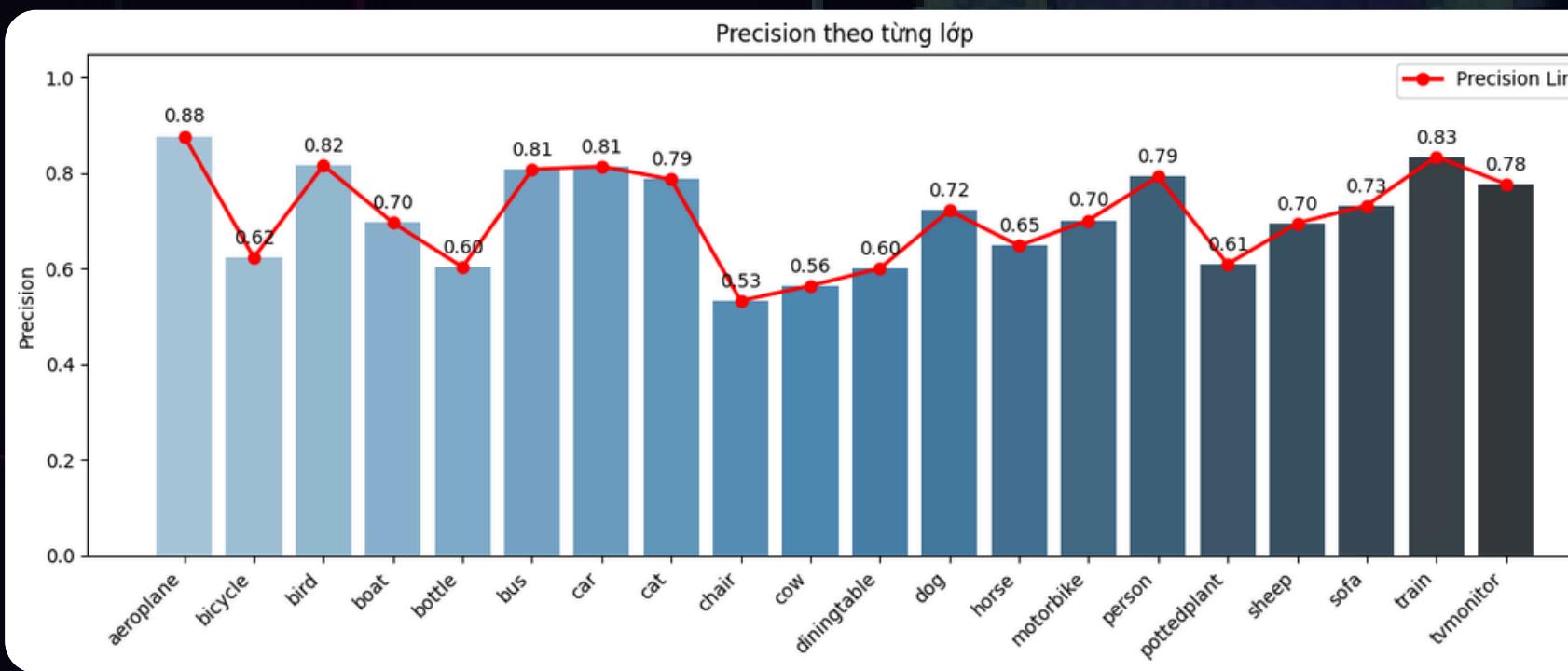
4. KẾT QUẢ ĐẠT ĐƯỢC



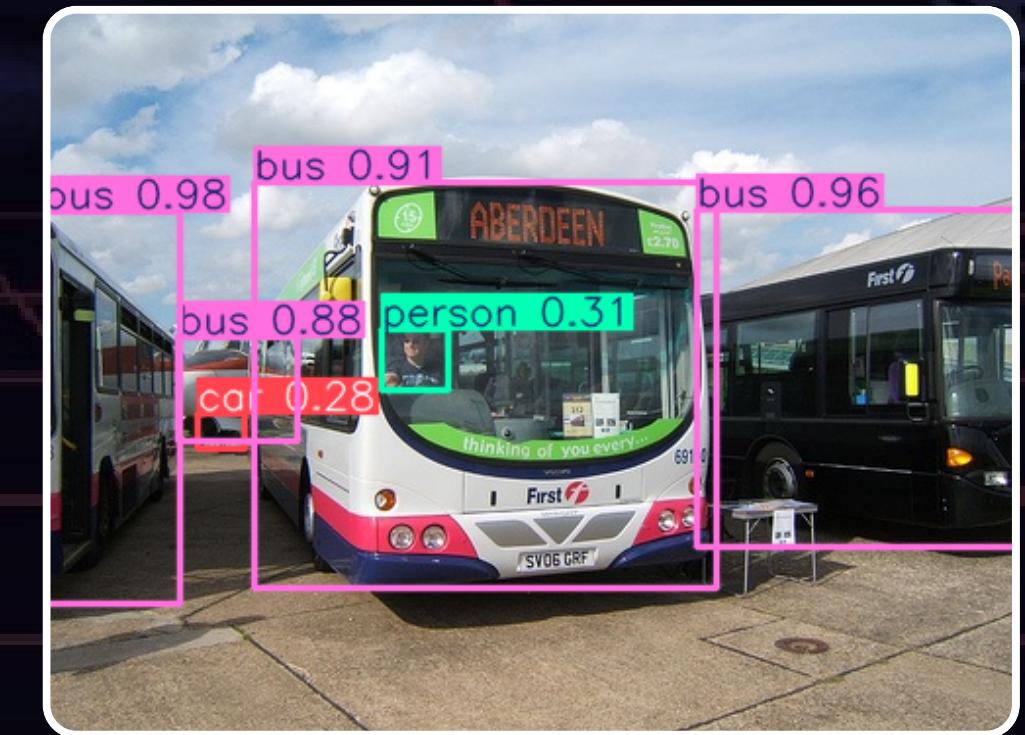
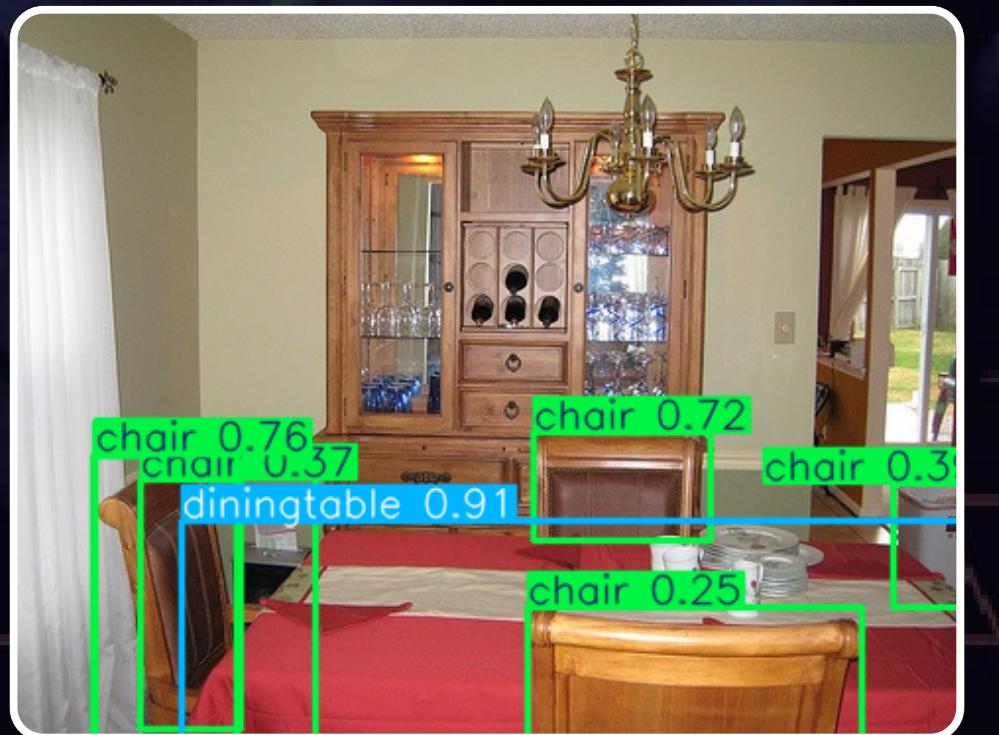
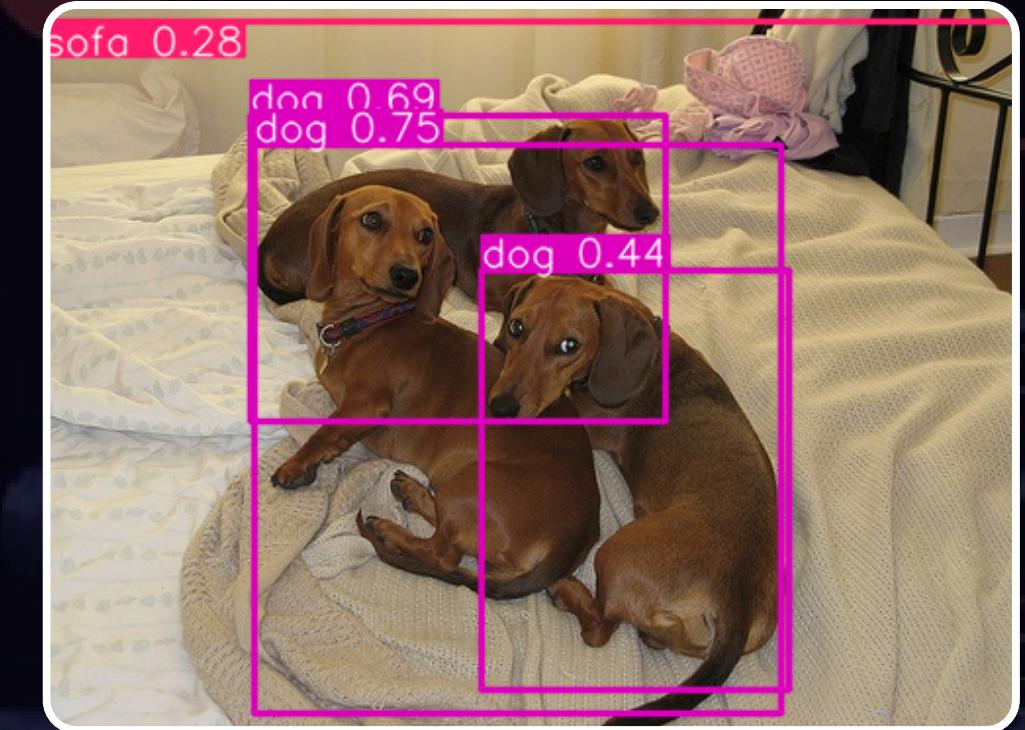
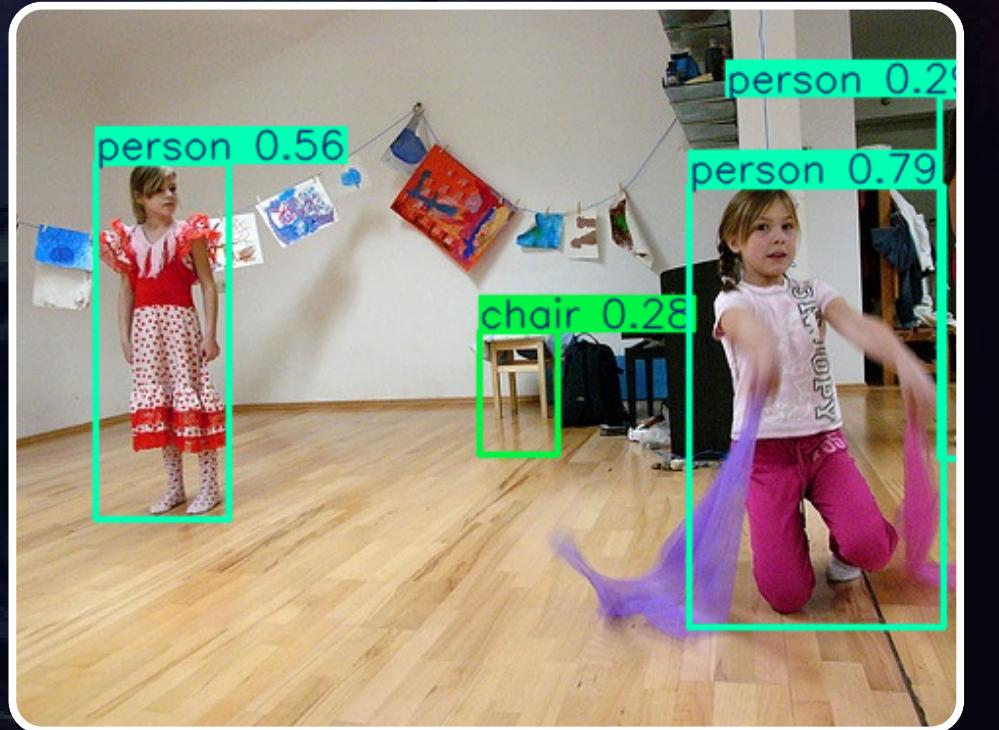
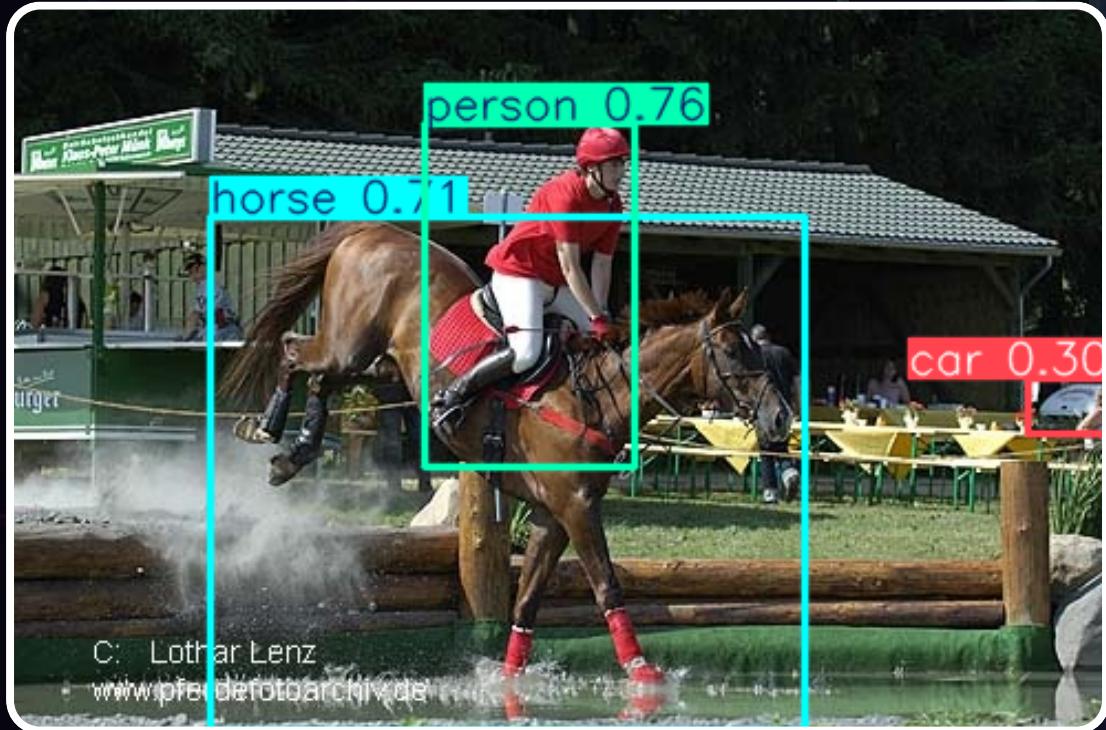
- Precision đạt 0.712, cao hơn Recall (0.594) → mô hình ít phát hiện nhầm.
- mAP@0.5 cao nhất (0.653), cho thấy mô hình hoạt động tốt ở ngưỡng IoU thấp.
- mAP@0.5:0.95 thấp (0.489) → mô hình chưa tối ưu ở các ngưỡng IoU cao.
- F1-score đạt 0.648, gần điểm trung bình → mô hình khá ổn định.



4. KẾT QUẢ ĐẠT ĐƯỢC



4. KẾT QUẢ ĐẠT ĐƯỢC



4. KẾT QUẢ ĐẠT ĐƯỢC

Ma trận nhầm lẫn

Confusion Matrix (Top 20 Confusing Classes)	
True Label	Predicted Label
aeroplane	304 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
bicycle	0 373 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
bird	0 0 320 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
boat	0 0 0 333 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
bottle	0 0 0 0 367 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
bus	0 0 0 0 0 235 0 0 0 0 0 0 0 0 0 0 0 0 0 0
car	0 0 0 0 0 0 673 0 0 0 0 0 0 0 0 0 0 0 0 0
cat	0 0 0 0 0 0 0 523 0 0 0 0 0 0 0 0 0 0 0 0
chair	0 0 0 0 0 0 0 0 1203 0 0 0 0 0 0 0 0 0 0 0
cow	0 0 0 0 0 0 0 0 0 386 0 0 0 0 0 0 0 0 0 0
diningtable	0 0 0 0 0 0 0 0 0 0 376 0 0 0 0 0 0 0 0 0
dog	0 0 0 0 0 0 0 0 0 0 0 682 0 0 0 0 0 0 0 0
horse	0 0 0 0 0 0 0 0 0 0 0 0 302 0 0 0 0 0 0 0
motorbike	0 0 0 0 0 0 0 0 0 0 0 0 0 304 0 0 0 0 0 0
person	0 0 0 0 0 0 0 0 0 0 0 0 0 0 6486 0 0 0 0 0
pottedplant	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 298 0 0 0 0
sheep	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 405 0 0 0
sofa	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 280 0 0
train	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 256 0
tvmonitor	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 259

1. Chính xác tuyệt đối:

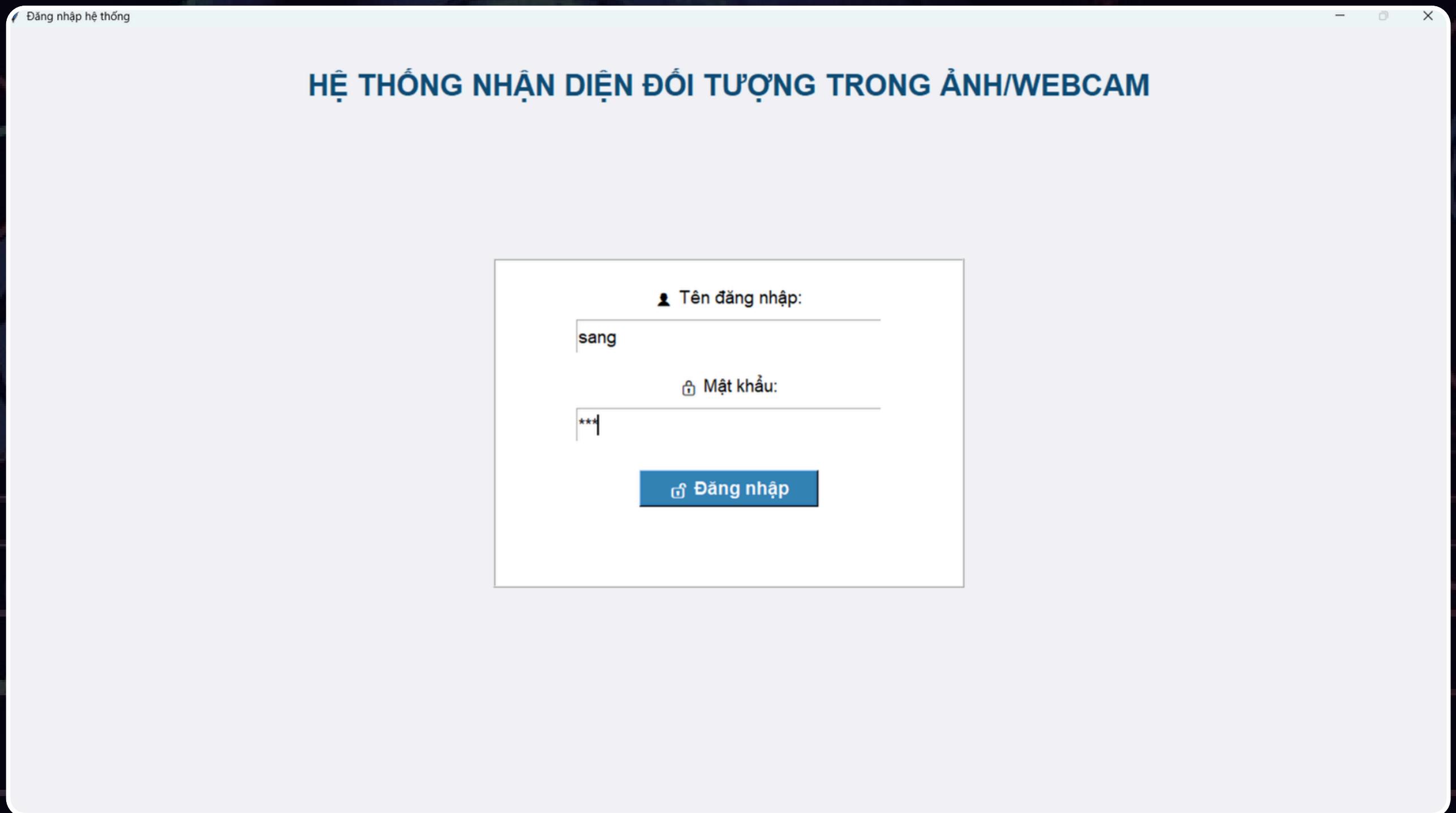
- Mô hình dự đoán đúng 100% cho tất cả 20 lớp phổ biến nhất.
- Không có nhầm lẫn giữa các lớp.

2. Lớp mất cân bằng:

- Lớp person chiếm số lượng lớn (6486 mẫu), các lớp khác ít hơn → mất cân bằng dữ liệu.

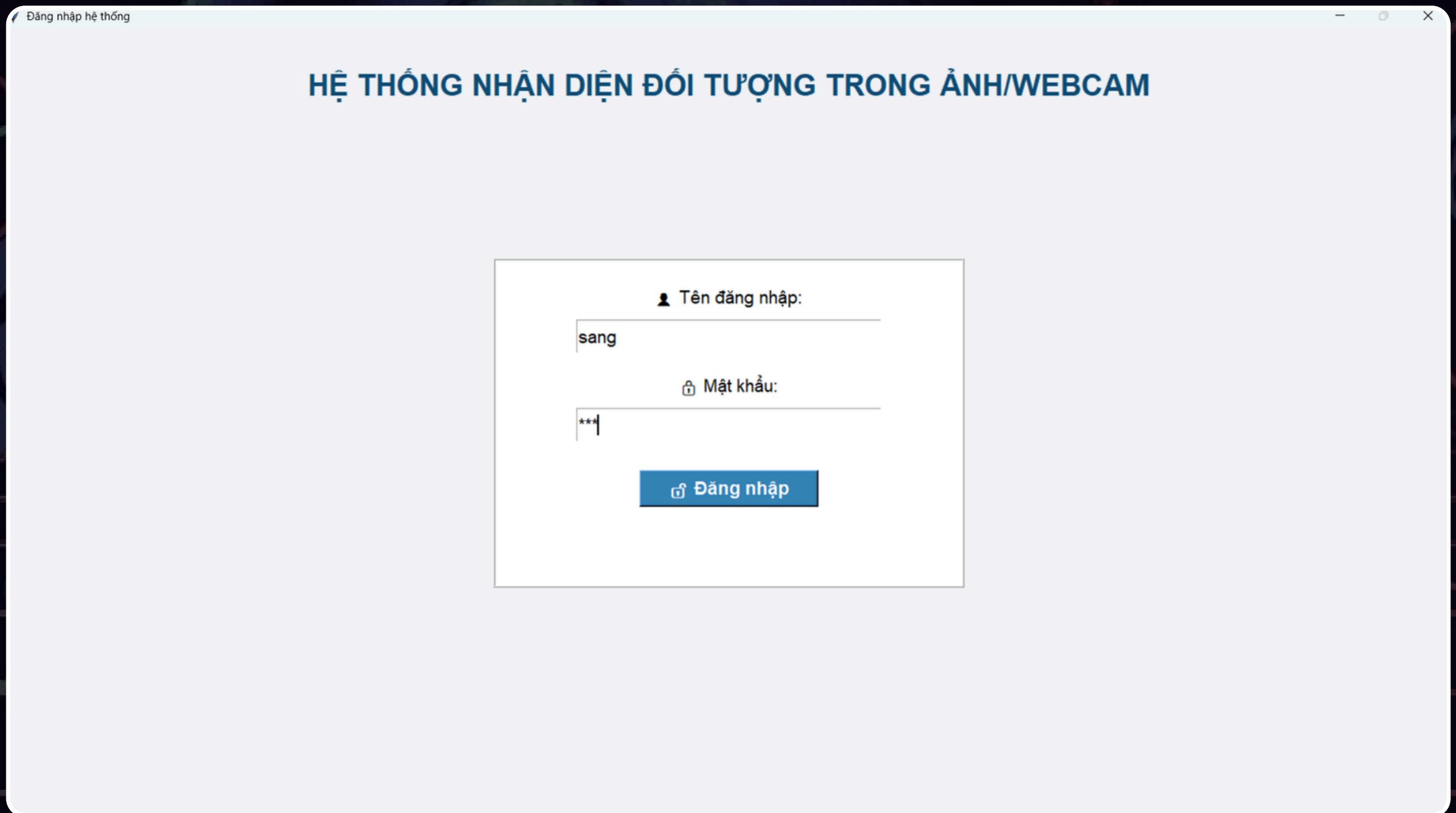
5. XÂY DỰNG HỆ THỐNG

Màn hình đăng nhập



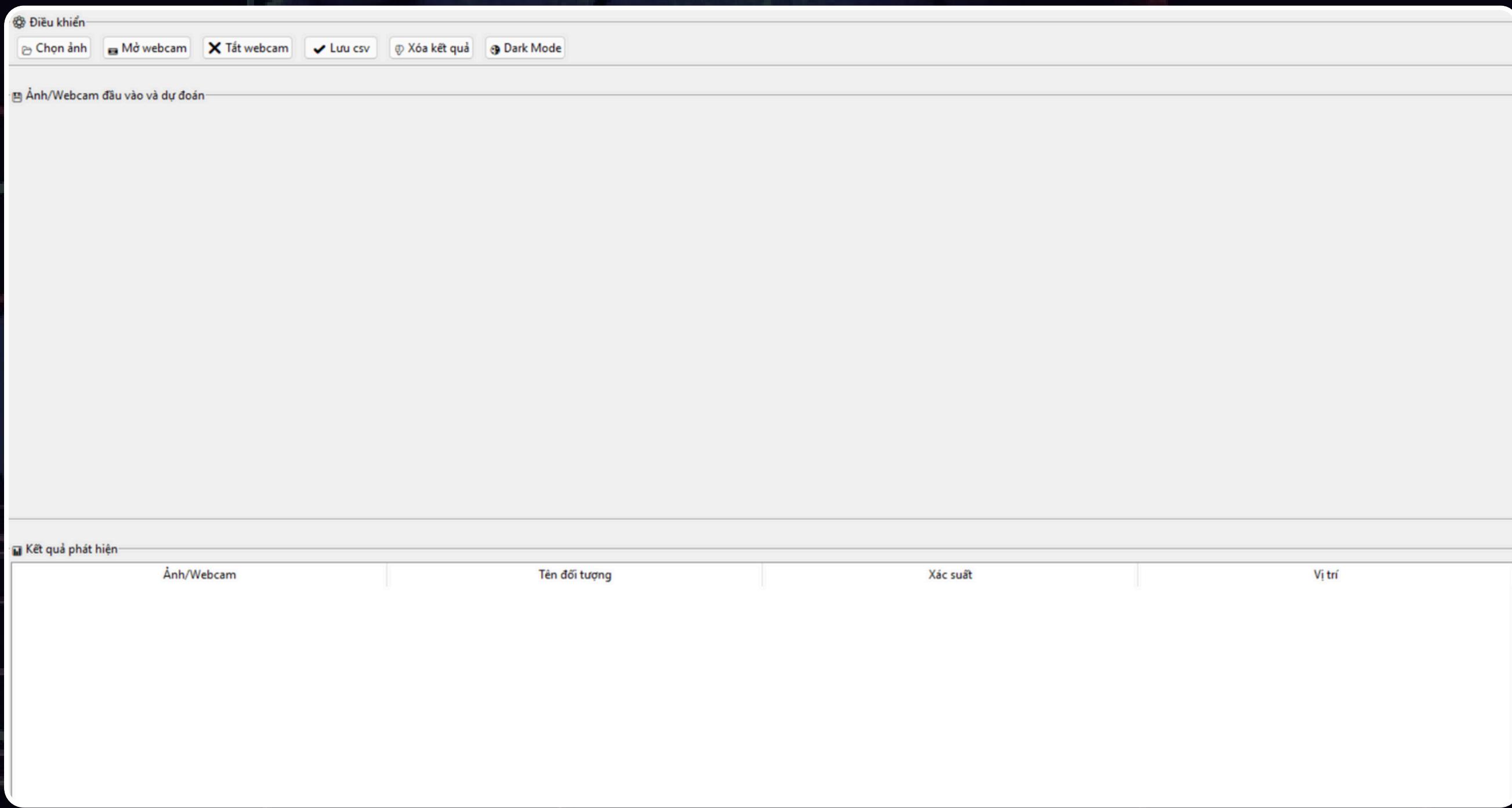
5. XÂY DỰNG HỆ THỐNG

Màn hình đăng nhập



5. XÂY DỰNG HỆ THỐNG

Màn hình đăng nhập



6. KẾT LUẬN

Trong khuôn khổ đề tài, nhóm đã nghiên cứu và triển khai thành công hệ thống phát hiện đối tượng trong ảnh sử dụng mô hình YOLO (You Only Look Once) trên tập dữ liệu Pascal VOC 2012. Quá trình thực nghiệm đã được tiến hành đầy đủ qua các bước:

Tiền xử lý dữ liệu: Chuẩn hóa, phân chia tập train/val/test, trực quan hóa dữ liệu gốc và bounding box.

Huấn luyện mô hình YOLOv8 kết hợp FPN: Sử dụng mô hình YOLO hiện đại có độ chính xác cao và khả năng nhận diện thời gian thực.

Đánh giá mô hình: Sử dụng các chỉ số đánh giá phổ biến như mAP@0.5, Precision, Recall, và Confusion Matrix. Mô hình đạt mAP@0.5 trung bình 0.66, trong đó các lớp như cat, car, aeroplane đạt kết quả rất tốt.

Phân tích kết quả: Rút ra được những ưu nhược điểm của mô hình trong từng tình huống cụ thể, qua đó đề xuất các hướng cải tiến mô hình.

Kết quả cho thấy mô hình có khả năng nhận diện tốt các đối tượng trong ảnh với tốc độ nhanh, độ chính xác cao, phù hợp để triển khai trong các ứng dụng thực tế như giám sát giao thông, kiểm tra sản phẩm công nghiệp, và các hệ thống an ninh tự động.

Việc sử dụng mô hình YOLO mang lại lợi thế lớn nhờ kiến trúc một bước (one-stage detector), giúp cân bằng tốt giữa tốc độ và độ chính xác, đặc biệt hữu ích trong các bài toán yêu cầu xử lý thời gian thực.

6. HẠN CHẾ

Tuy đạt được những kết quả khả quan, hệ thống vẫn tồn tại một số hạn chế:

Hiệu suất chưa đồng đều giữa các lớp: Một số lớp có mAP thấp do kích thước nhỏ, bị che khuất, hoặc có đặc điểm hình dạng không rõ ràng (ví dụ: bottle, pottedplant, chair).

Chưa xử lý mất cân bằng lớp triệt để: Các lớp có số lượng mẫu ít ảnh hưởng đến quá trình huấn luyện và hiệu suất tổng thể.

Chưa áp dụng nhiều kỹ thuật tối ưu: Như Focal Loss, Mosaic augmentation, hoặc các kiến trúc nâng cao như YOLOv8-seg, YOLOv8x.

6. HƯỚNG PHÁT TRIỂN

Đề xuất cải thiện mô hình

Trong tương lai, nhóm đề xuất một số hướng phát triển nhằm cải thiện hiệu suất và khả năng ứng dụng của hệ thống:

1. Tối ưu hóa mô hình
 - Thử nghiệm các backbone mạnh hơn như ResNet-101, EfficientNet, CSPDarknet.
 - Áp dụng tối ưu siêu tham số tự động bằng các kỹ thuật như Grid Search hoặc Bayesian Optimization.
2. Tăng cường dữ liệu và xử lý mất cân bằng
 - Áp dụng kỹ thuật Mosaic Augmentation, CutMix, Random Crop để tạo sự đa dạng.
 - Sử dụng Focal Loss để giải quyết bài toán mất cân bằng lớp.
 - Thêm dữ liệu huấn luyện hoặc khai thác từ các tập mở rộng như COCO, Open Images Dataset.
3. Giảm chi phí tính toán
 - Tối ưu hóa mô hình để hoạt động hiệu quả trên các thiết bị phần cứng có tài nguyên hạn chế: như thiết bị di động hoặc các máy tính tại các khu vực y tế thiếu thốn.
4. Mở rộng bài toán
 - Nâng cấp mô hình sang YOLOv8-Segmentation để phát hiện đồng thời vị trí và phân vùng đối tượng.

**THANKS FOR
WATCHING THE
PRESENTATION**