

# Phát Hiện Đối Tượng Trên Ảnh Hồng Ngoại Tầm Cao

## Nhóm 15

Nguyễn Thùa Tuân - 22001652

Nguyễn Hoàng Việt - 22001658

Nguyễn Quang Việt - 22001659

**Giảng viên hướng dẫn:**  
TS. Cao Văn Chung

# Nội dung trình bày

- ① Đặt vấn đề và Mục tiêu nghiên cứu
- ② Bối cảnh
- ③ Dữ liệu thực nghiệm (HIT-UAV)
- ④ Phương pháp thực hiện
- ⑤ Kết quả Thực nghiệm
- ⑥ Bổ sung thêm so với bài giữa kì

# **PHẦN 1**

Đặt vấn đề và Mục tiêu nghiên cứu

# Đặt vấn đề

## Bối cảnh

- Các mô hình thị giác máy tính hiện đại (CNN, YOLO) đạt độ chính xác cao trên ảnh ánh sáng khả kiến (RGB).
- Ảnh hồng ngoại tầm cao (High-altitude infrared imagery) có vai trò quan trọng (giám sát, cứu nạn, quân sự) nhưng ít được nghiên cứu.

## Thách thức chính

- ❶ **Chất lượng ảnh:** Độ tương phản thấp, chi tiết mờ, nhiễu cao.
- ❷ **Đối tượng nhỏ (Small-object detection):** Góc nhìn từ trên cao khiến đối tượng chỉ chiếm vài pixel.
- ❸ **Thiếu hụt dữ liệu:** Dữ liệu thường mang tính bảo mật (quân sự), khó tiếp cận để huấn luyện mô hình.

# Mục tiêu nghiên cứu

**Mục tiêu tổng quát:** Xây dựng mô hình phát hiện đối tượng hoạt động hiệu quả trên ảnh hồng ngoại tầm cao.

## Mục tiêu cụ thể:

- **Mô hình:** Nghiên cứu và tối ưu hóa các mô hình hiện đại (YOLO) cho dữ liệu hồng ngoại.
- **Đánh giá:** Thực nghiệm và đánh giá dựa trên các chỉ số mAP, Precision, Recall.
- **Cải thiện:** Đề xuất giải pháp xử lý vấn đề phát hiện đối tượng nhỏ.

# **PHẦN 2**

Bối cảnh

# Bối cảnh: Từ truyền thống đến Deep Learning

## Phương pháp sơ khai:

- *Haar Cascade, HOG + SVM:* Dựa vào trích xuất đặc trưng thủ công.
- *Hạn chế:* Kém ổn định, nhạy cảm với ánh sáng.

→ *Nhu cầu cấp thiết về một mô hình cân bằng giữa Tốc độ và Độ chính xác.*

## Deep Learning (R-CNN Series):

- *R-CNN, Fast/Faster R-CNN:* Độ chính xác cao.
- *Hạn chế:* Xử lý đa giai đoạn → Tốc độ chậm, khó đáp ứng thời gian thực (Real-time).

# Lựa chọn mô hình: Hệ sinh thái Ultralytics

- **YOLOv5:**

- Xây dựng trên PyTorch, dễ triển khai và huấn luyện.
- Cân bằng tốt giữa độ chính xác và khả năng sử dụng thực tế.

- **YOLOv8 (Phiên bản chính được chọn):**

- Framework thông nhất cho nhiều tác vụ: Detection, Segmentation, Classification.
- Kiến trúc hiện đại (Anchor-free, New Backbone) giúp tăng hiệu suất trên các đối tượng khó.

# **PHẦN 3**

Giới thiệu bộ dữ liệu HIT-UAV

# Giới thiệu bộ dữ liệu HIT-UAV

## Tổng quan

- **HIT-UAV:** Bộ dữ liệu ảnh hồng ngoại nhiệt (Thermal Infrared) thu thập từ phương tiện bay không người lái (UAV) ở độ cao lớn.
- **Quy mô:** 2.898 ảnh được trích xuất chọn lọc từ 43.470 khung hình video gốc.
- **Môi trường:** Thực tế đa dạng (trường học, bãi đỗ xe, đường giao thông).

## Điểm khác biệt nổi bật

Khác với các bộ dữ liệu trước đây (VisDrone, UAV123), HIT-UAV cung cấp **thông tin chuyến bay chi tiết** cho từng ảnh:

- Độ cao bay (Altitude).
- Góc nghiêng camera (Camera pitch).
- Điều kiện thời tiết và ánh sáng (Ngày/Đêm).

# Quy trình xây dựng bộ dữ liệu

## 1. Ghi hình video (Data Recording):

- **Độ cao:** 60m – 130m (bước nhảy 10m).
- **Góc camera:** 30° – 90°.
- **Thời gian:** Cả ban ngày và ban đêm (ảnh đêm có độ tương phản nhiệt tốt hơn).

## 2. Trích xuất và Xử lý (Preprocessing):

- Tốc độ quay gốc: 7 FPS.
- Lấy mẫu: Trích xuất **1 ảnh mỗi 15 khung hình**.
- Mục đích: Loại bỏ trùng lặp, giữ lại sự thay đổi về vị trí và góc nhìn, tối ưu dung lượng.

# Gán nhãn đối tượng (Labeling)

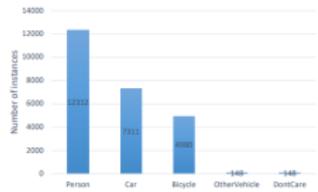
## Các lớp đối tượng (Classes):

- ① Person (Người).
- ② Car (Ô tô).
- ③ Bicycle (Xe đạp).
- ④ Other (Phương tiện khác).
- ⑤ DontCare (*Vùng nhiễu/quá nhỏ*).

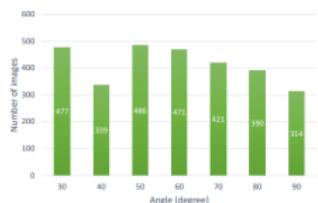
## Phân chia tập dữ liệu

Train (70%) - Test (20%) - Validation (10%)

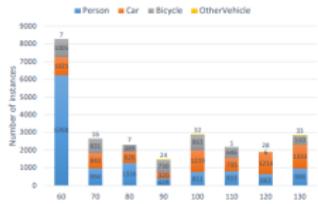
# Phân bố dữ liệu



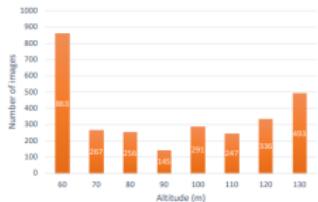
(a) The distribution of object instances across object categories



(c) The distribution of images across camera perspectives



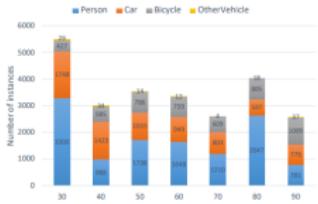
(e) The distribution of object instances with different categories across flight altitudes



(b) The distribution of images across flight altitudes



(d) The distribution of images across flight times



(f) The distribution of object instances with different categories across camera perspectives

Hình: Phân bố dữ liệu

# **PHẦN 4**

Phương pháp thực hiện

# Tổng quan hệ thống theo dõi đa đối tượng I

## Pipeline

Hệ thống theo dõi đa đối tượng đề xuất được thiết kế theo kiến trúc Tracking-by- Detection. Luồng xử lý dữ liệu của hệ thống được mô tả chi tiết qua các giai đoạn sau:

### Cụ thể về pipeline :

- **Tiền xử lý Phát hiện:** Ảnh nhiệt đầu vào  $I_t$  tại thời điểm  $t$  được đưa qua mô hình phát hiện (đã tối ưu hóa Backbone như trình bày ở Mục 3.2) để trích xuất tập hợp các hộp bao  $D_t = \{b_i, s_i\}$ , với  $b_i$  là tọa độ và  $s_i$  là độ tin cậy.
- **Ước lượng chuyển động nền (GMC):** Tính toán ma trận biến đổi giữa  $I_{t-1}$  và  $I_t$  để bù trừ sự dịch chuyển của camera UAV, đồng bộ hóa hệ tọa độ của các quỹ đạo cũ về thời điểm hiện tại.

## Tổng quan hệ thống theo dõi đa đối tượng II

- **Dự đoán trạng thái (Kalman Prediction):** Sử dụng bộ lọc Kalman để dự đoán vị trí mới của các quỹ đạo dựa trên mô hình vận tốc không đổi.
- **Liên kết dữ liệu (Data Association):** Sử dụng thuật toán ByteTrack để ghép nối các phát hiện  $D_t$  với các quỹ đạo dự đoán, ưu tiên các phát hiện có độ tin cậy cao nhưng không bỏ qua các phát hiện yếu.
- **Cập nhật Quản lý vòng đời:** Cập nhật trạng thái vector Kalman cho các quỹ đạo được ghép nối, khởi tạo quỹ đạo mới hoặc xóa bỏ các quỹ đạo đã mất dấu.

# Mô hình phát hiện đối tượng

## Ý tưởng cốt lõi

Kết hợp sức mạnh trích xuất đặc trưng của các mạng CNN chuyên biệt (Backbone) với khả năng phát hiện đối tượng nhanh nhạy của YOLOv8.

### Cấu trúc mô hình đề xuất:

- **1. Head & Neck (Giữ nguyên):** Sử dụng kiến trúc YOLOv8 (Ultralytics) để đảm bảo tính năng End-to-End và hiệu suất cao.
- **2. Backbone (Thay thế):** Loại bỏ CSPDarknet mặc định, thay thế bằng các mạng phân loại tối ưu hơn (Custom Backbone).
- **3. Cơ chế tích hợp:** Chỉnh sửa cấu hình (.yaml) để kết nối các tầng đặc trưng (P3, P4, P5) từ Backbone mới vào Neck (C2f, SPPF) của YOLOv8.

# Các mô hình Backbone được thử nghiệm

Nghiên cứu chia các Backbone thành 02 nhóm dựa trên mục tiêu tối ưu:

## Nhóm Lightweight/Mobile

(*Tối ưu tốc độ và chi phí tính toán*)

- **MobileNet(V2 và V3):** Mạng siêu nhẹ, sử dụng MBConv và SE Block.
- **EfficientNet (B0, B3):**
  - Sử dụng Compound Scaling (cân bằng chiều sâu, rộng, độ phân giải).
  - B0: Baseline siêu nhanh.
  - B3: Cân bằng tốt hơn.

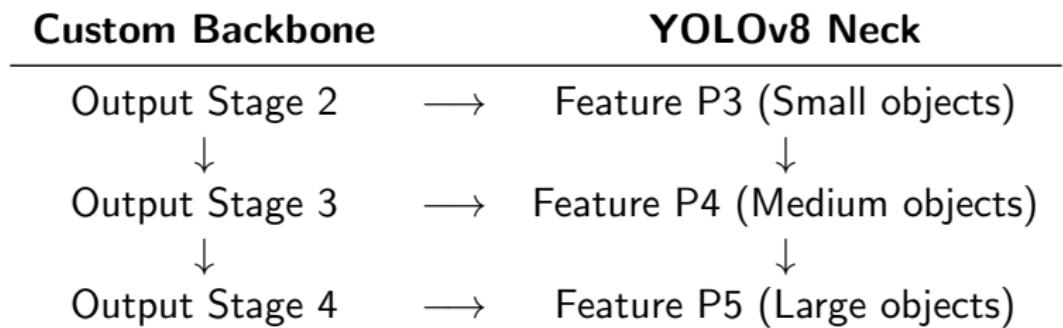
## Nhóm Modern CNN

(*Tối ưu độ chính xác*)

- **ConvNeXt-T (Tiny):** Thiết kế lại ResNet theo phong cách Vision Transformer (ViT).
- **ConvNeXt-S (Small):** Phiên bản lớn hơn, trích xuất đặc trưng mạnh mẽ nhờ Depthwise Convolution kích thước lớn ( $7 \times 7$ ).

# Cơ chế tích hợp đặc trưng (Feature Integration)

Việc thay thế Backbone được thực hiện bằng cách ánh xạ các tầng đầu ra tương ứng:

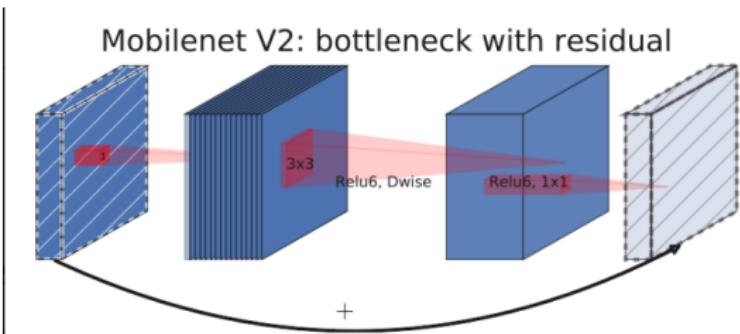


Việc kết nối này giúp YOLOv8 tận dụng được các đặc trưng tốt nhất mà các Backbone hiện đại (như ConvNeXt) trích xuất được.

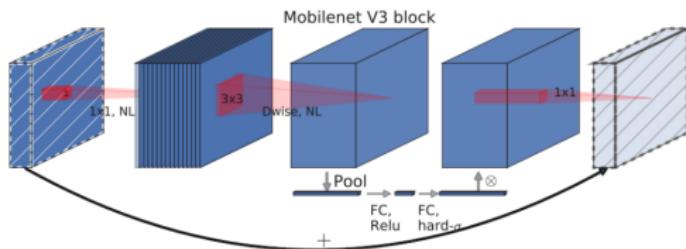
## Mobilenet backbone

Các phiên bản MobileNet được thiết kế cho các thiết bị di động bằng cách sử dụng MBConv (Mobile Inverted Bottleneck Convolution) . Đây là khôi cơ bản nhằm giảm thiểu chi phí tính toán. MobileNetV3 bổ sung thêm các tối ưu hóa như Squeeze-and-Excitation (SE) Block để tăng cường khả năng trích xuất đặc trưng mà vẫn giữ được tính hiệu quả cao.

# Mobilenet backbone



Hình: Cấu trúc MobileNet V2 Block



Hình: Cấu trúc MobileNet V3 Block

## Efficient backbone

EfficientNet sử dụng chiến lược **Compound Scaling**, đồng thời mở rộng cả chiều sâu (Depth), chiều rộng (Width), và độ phân giải (Resolution) của mạng một cách thống nhất, nhằm tối đa hóa hiệu suất. . Việc thử nghiệm EfficientNetB0 (phiên bản baseline nhẹ nhất) và EfficientNetB3 (phiên bản cân bằng hơn) giúp đánh giá sự đánh đổi giữa kích thước và độ chính xác.

# Efficient backbone

Bảng: Tóm tắt kiến trúc mạng

<b>Stage</b>	<b>Operator</b>	<b>Resolution</b>	<b>#Channels</b>	<b>#Layers</b>
1	Conv3x3	$224 \times 224$	32	1
2	MBConv1, k3x3	$112 \times 112$	16	1
3	MBConv6, k3x3	$112 \times 112$	24	2
4	MBConv6, k5x5	$56 \times 56$	40	2
5	MBConv6, k3x3	$28 \times 28$	80	3
6	MBConv6, k5x5	$14 \times 14$	112	3
7	MBConv6, k5x5	$14 \times 14$	192	4
8	MBConv6, k3x3	$7 \times 7$	320	1
9	Conv1x1 & Pooling & FC	$7 \times 7$	1280	1

## Convnext backbone

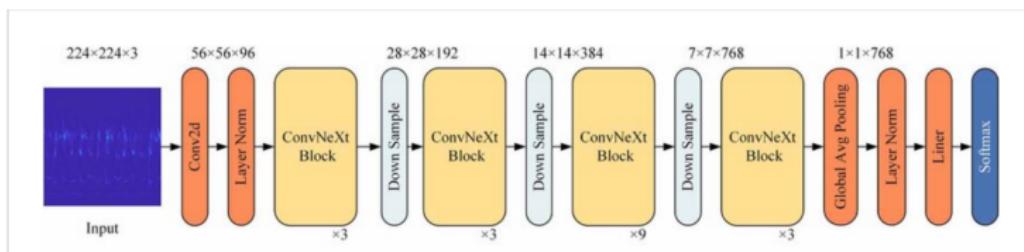
ConvNeXt được xem là mô hình CNN thế hệ mới, được "hiện đại hóa" bằng cách áp dụng các ý tưởng thiết kế chính từ \*\*Vision Transformer (ViT)\*\* . Nó thay thế các khối ResNet truyền thống bằng các khối đơn giản hơn, sử dụng Depthwise Convolution kích thước lớn ( $7 \times 7$ ), giúp cải thiện khả năng học ngữ cảnh mà không cần cơ chế Attention phức tạp.

# Convnext backbone

Bảng: Kiến trúc ConvNeXt-Tiny với đầu vào  $224 \times 224 \times 3$

Giai đoạn	Thành phần	Output size (H×W)	Channels	YOLOv8
Input	Ảnh đầu vào	$224 \times 224$	3	-
Stem	Conv $4 \times 4$ , stride 4	$56 \times 56$	96	-
Stage 1	$3 \times$ ConvNeXt Blocks	$56 \times 56$	96	Không dùng
Stage 2	Downsampling + $3 \times$ ConvNeXt Blocks	$28 \times 28$	192	P3
Stage 3	Downsampling + $9 \times$ ConvNeXt Blocks	$14 \times 14$	384	P4
Stage 4	Downsampling + $3 \times$ ConvNeXt Blocks	$7 \times 7$	768	P5

Hình: Minh họa kiến trúc ConvNeXt



# Thiết lập Huấn luyện và Đánh giá

## Cấu hình Huấn luyện:

- **Framework:** Python, thư viện Ultralytics.
- **Phần cứng:** Sử dụng GPU P100.
- **Khởi tạo trọng số:** Transfer Learning (Huấn luyện trước trên ImageNet).
- **Tham số chính:**
  - Image Size:  $640 \times 640$
  - Batch size: 16 và 8
  - Epochs: 50 và 40

## Chỉ số đánh giá:

- Precision (P)
- Recall (R)
- mAP@.50
- mAP@.5:0.95
- F1-score

# **PHẦN 5**

Kết quả Thực nghiệm

# Tổng hợp kết quả thực nghiệm

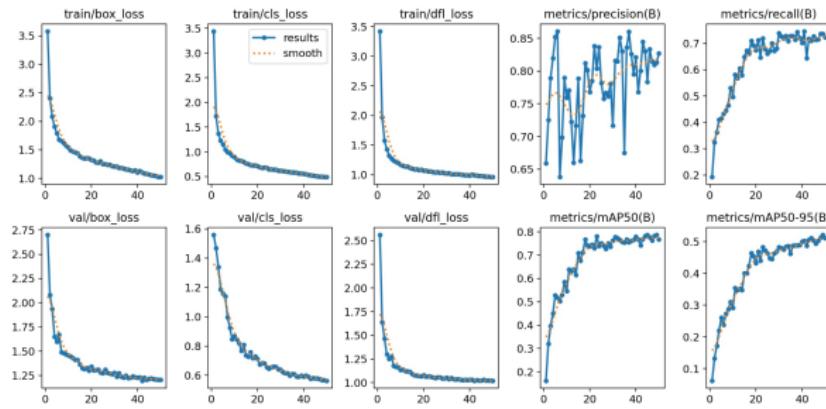
Bảng: So sánh hiệu suất các mô hình

Mô hình	Backbone	Precision (P)	Recall (R)	F1-score	mAP@.50	mAP@.5:.95
<b>YOLOv8</b>	CSPDarknet (YOLOv8-n)	0.89	0.74	0.808	0.808	0.521
MobileNetV3 + YOLOv8	MobileNetV3-L	0.812	0.737	0.773	0.787	0.521
MobileNetV2 + YOLOv8	MobileNetV2	0.789	0.737	0.762	0.772	0.503
EfficientNetB3 + YOLOv8	EfficientNetB3	0.82	0.737	0.776	0.787	0.53
EfficientNetB0 + YOLOv8	EfficientNetB0	0.914	0.765	0.833	0.811	0.521
ConvNeXt-T + YOLOv8	ConvNeXt-T	0.845	0.762	0.802	0.814	0.537
ConvNeXt-S + YOLOv8	ConvNeXt-S	0.87	0.785	0.825	0.861	0.572
MobileNetV3 + YOLOv10	MobileNetV3-L	0.7	0.565	0.625	0.601	0.338
MobileNetV2 + YOLOv10	MobileNetV2	0.75	0.636	0.688	0.699	0.425
<b>YOLOv11</b>	Custom/MBV3	0.793	0.561	0.657	0.655	0.404

Bảng: Kết quả tracking với EfficientNetB0 + YOLOv8

Frames	MOTA	MOTP	IDF1	MT	ML	ID Sw
216	63.3%	0.308	80.9%	11	0	1

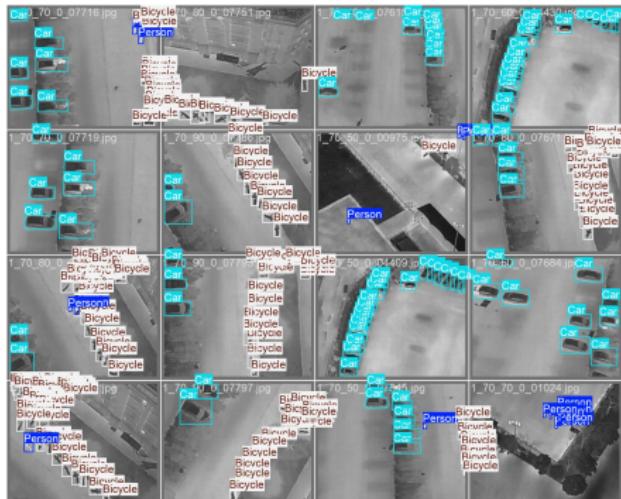
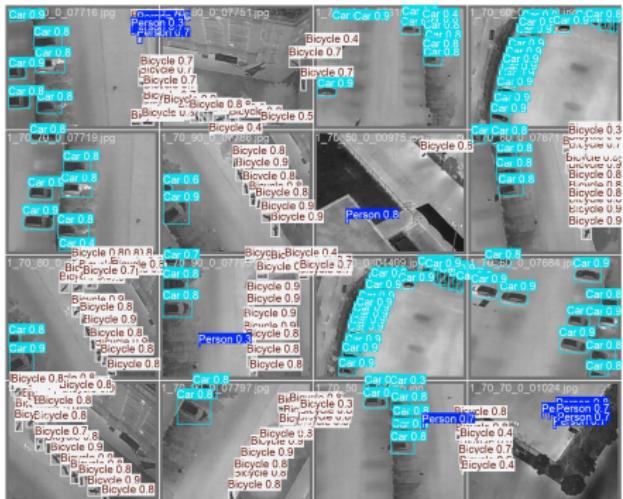
# Kết quả: MobileNetV2 + YOLOv8



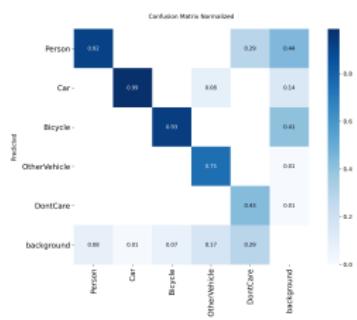
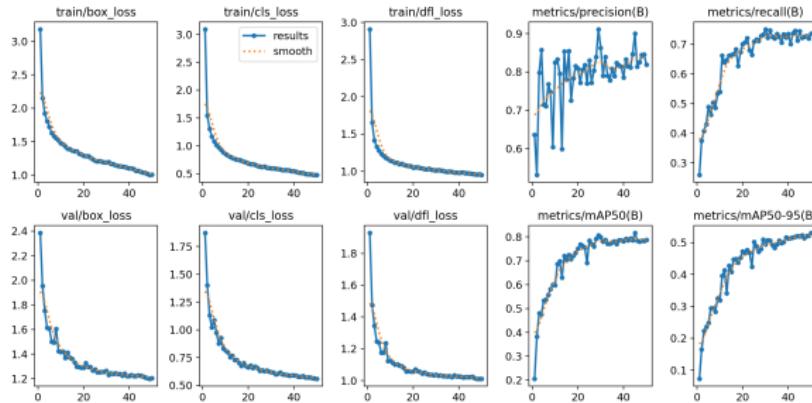
Hình: Confusion Matrix

Hình: Kết quả phát hiện (Detection Result)

Kết quả : MobileNetV2 + Yolov8



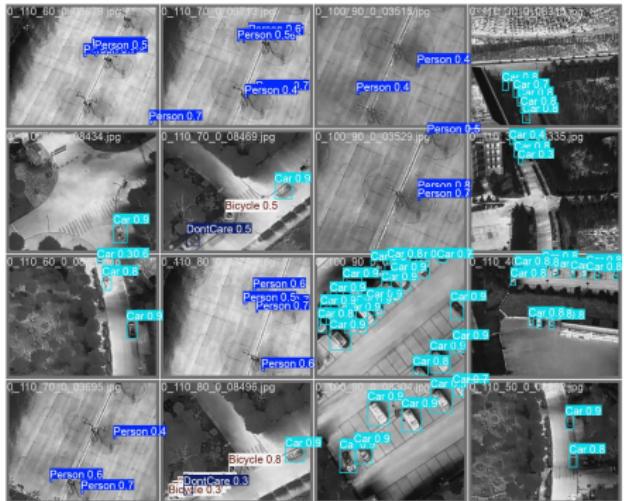
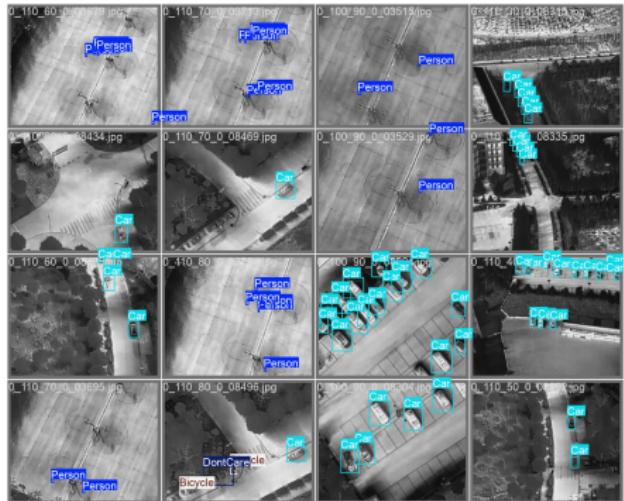
# Kết quả: EfficientNetB0 + YOLOv8



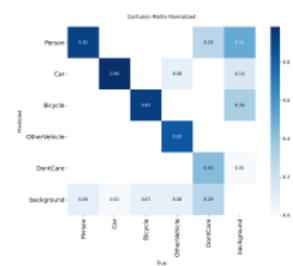
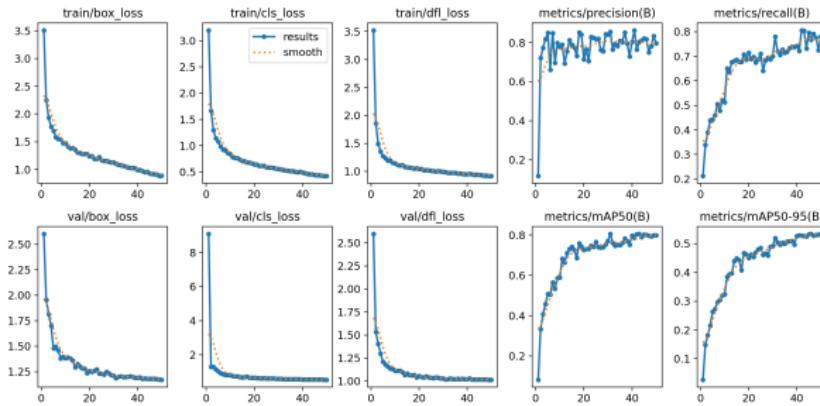
Hình: Confusion Matrix

Hình: Kết quả phát hiện (Detection Result)

# Kết quả : EfficientNetB0 + Yolov8



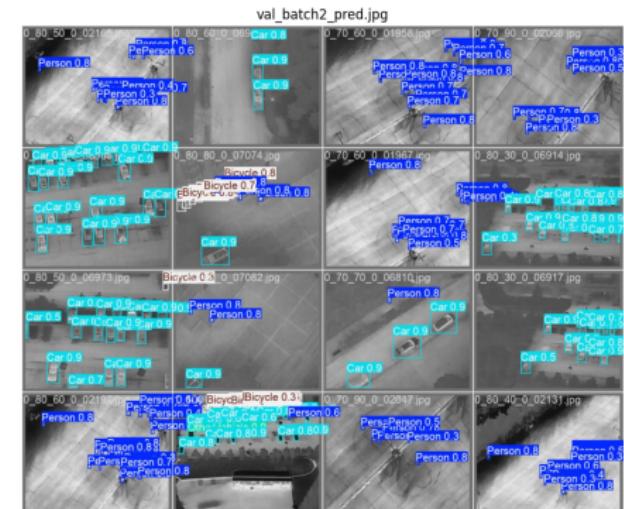
# Kết quả: ConvNeXt-T + YOLOv8



Hình: Confusion Matrix

Hình: Kết quả phát hiện (Detection Result)

# Kết quả : ConvNeXt-T + Yolov8



# **PHẦN 6**

Bổ sung thêm so với bài giữa kì

# Vấn đề đặt ra từ Báo cáo giữa kỳ

Từ Phát hiện đơn lẻ đến Theo dõi liên tục

## Hạn chế của Báo cáo giữa kỳ

- Các mô hình chỉ thực hiện **Phát hiện vật thể (Detection)** trên từng khung hình độc lập.
- **Thiếu thông tin thời gian:** Chưa định danh (ID) được đối tượng và chưa vẽ được quỹ đạo di chuyển.

## Thách thức thực tế cần giải quyết (Motivation)

- ① **Chuyển động của UAV (Ego-motion):** Camera rung lắc và di chuyển liên tục khiến các thuật toán nhầm lẫn chuyển động của nền là chuyển động của vật thể.
- ② **Chất lượng ảnh nhiệt Che khuất:** Ảnh nhiệt có độ nhiễu cao; khi đối tượng bị che khuất (cây, tòa nhà), độ tin cậy (confidence) giảm mạnh dẫn đến *mất dấu (Lost Track)*.

# Bù trừ chuyển động toàn cục (GMC)

Global Motion Compensation

## Vấn đề: Chuyển động của nền (Ego-motion)

Do UAV di chuyển và rung lắc, vị trí pixel của vật thể đứng yên cũng thay đổi → Gây nhiễu cho bộ lọc Kalman.

### Quy trình xử lý Image-based GMC:

- ① **Trích xuất:** Dùng thuật toán **FAST** tìm điểm đặc trưng trên nền  $I_{t-1}$ .
- ② **Theo dõi:** Dùng **Optical Flow (Lucas-Kanade)** để tìm vị trí trên  $I_t$ .
- ③ **Ước lượng:** Dùng **RANSAC** tính ma trận biến đổi Affine  $A_t$ :

$$A_t = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \quad (1)$$

- ④ **Đồng bộ:** Cập nhật trạng thái quỹ đạo về hệ tọa độ mới trước khi dự đoán:  $x_{corrected} = A_t \times x_{t-1}$

# Bộ lọc Kalman: Mô hình chuyển động

## Không gian trạng thái và Dự đoán

Để theo dõi đối tượng trong không gian 2D, hệ thống sử dụng giả định **Vận tốc không đổi (Constant Velocity Model)**.

**Vector trạng thái (8 chiều):**

$$\mathbf{x} = [x_c, y_c, a, h, \dot{x}_c, \dot{y}_c, \dot{a}, \dot{h}]^T \quad (2)$$

( $x_c, y_c$ ): Tâm,  $a$ : Tỷ lệ,  $h$ : Chiều cao.

**Quy trình lọc:**

- **Dự đoán:**

$$\mathbf{x}_{k|k-1} = \mathbf{F}_k \mathbf{x}_{k-1|k-1} + \mathbf{Q}_k$$

- **Cập nhật đo lường:**

$$\mathbf{y}_k = \mathbf{z}_k - \mathbf{H}_k \mathbf{x}_{k|k-1}$$

Trong đó  $\mathbf{z}_k$  là hộp bao (bounding box) nhận được từ mô hình YOLO.

# Bộ lọc Kalman: Cơ chế thích ứng nhiễu (NSA)

Noise Scale Adaptive Kalman Filter

## Thách thức từ ảnh nhiệt

Hộp bao phát hiện thường có độ tin cậy ( $c_k$ ) thấp và vị trí không ổn định khi đối tượng bị che khuất hoặc nhiễu nhiệt.

**Giải pháp NSA:** Điều chỉnh ma trận hiệp phương sai nhiễu đo lường  $R_k$  dựa trên độ tin cậy:

$$R_k = (1 - c_k) \cdot R_{std} \quad (3)$$

Ý nghĩa:

- Khi  $c_k$  cao (Rõ nét):  $R_k$  nhỏ → Bộ lọc tin tưởng vào YOLO.
- Khi  $c_k$  thấp (Mờ/Che khuất):  $R_k$  lớn → Bộ lọc tin tưởng vào Dự đoán (Motion Model).

⇒ Giúp quỹ đạo mượt mà (smoothness) hơn.

# Chiến lược liên kết dữ liệu (ByteTrack)

Tận dụng các phát hiện có độ tin cậy thấp

Khác với DeepSORT (bỏ qua box điểm thấp), ByteTrack thực hiện liên kết 2 giai đoạn để "cứu" các đối tượng bị che khuất:

## ① Phân loại phát hiện:

- $D_{high}$ : Độ tin cậy cao ( $> \tau_{high}$ ).
- $D_{low}$ : Độ tin cậy thấp (thường là đối tượng bị che khuất).

② **Giai đoạn 1 (High Match):** Ghép quỹ đạo (Tracks) với  $D_{high}$  bằng thuật toán Hungarian + IoU.

③ **Giai đoạn 2 (Low Match):** Các quỹ đạo chưa được ghép ( $T_{remain}$ ) sẽ được tìm kiếm trong  $D_{low}$ .

**Kết quả:** Giảm thiểu đáng kể việc đứt quãng quỹ đạo (ID Switch) trong môi trường phức tạp.

## Quản lý vòng đời quỹ đạo (Lifecycle)

Hệ thống sử dụng máy trạng thái (State Machine) để lọc nhiễu dương tính giả (False Positives):

- **New (Mới):** Khi phát hiện đối tượng mới. Phải được ghép nối liên tiếp trong **3 frames** thì mới được xác nhận là *Tracked*.
- **Tracked (Đang theo dõi):** Trạng thái ổn định. Cập nhật vị trí liên tục.
- **Lost (Mất dấu):** Khi không tìm thấy hộp bao tương ứng.
  - Hệ thống vẫn duy trì dự đoán trong bộ nhớ thêm *Max\_Age (30 frames)*.
  - Nếu đối tượng xuất hiện lại trong khoảng này → **Re-activate** (Giữ nguyên ID).
  - Nếu quá hạn → **Delete** (Xóa vĩnh viễn).