Vehicle Counting AI

Báo cáo đồ án Nhập môn Trí tuệ nhân tạo

Sinh viên thực hiện:
Nguyễn Minh Anh - 23001495
Nguyễn Trung Kiên - 23001530
Nguyễn Thế Quang - 23001549
Trần Đăng Tài - 23001558
Nguyễn Doãn Toàn - 23001564

Giảng viên: TS. Hoàng Anh Đức

Khoa Toán - Cơ - Tin học
Trường Đại học Khoa học Tự nhiên - ĐHQGHN

Hà Nội, ngày 8 tháng 12 năm 2025

Nội dung trình bày

1.1 Tổng quan đề tài

- **Mục tiêu:** Xây dựng hệ thống đếm xe tự động sử dụng Thị giác máy tính (Computer Vision) và Học sâu (Deep Learning).
- **Mô hình & Dữ liệu:**
  - Sử dụng **YOLOv8m** fine-tuning trên tập dữ liệu giao thông Việt Nam.
  - **Dữ liệu:** 1547 ảnh với 4 lớp: Car, Motor, Truck, Bus.
- **Kết quả đạt được:**
  - Độ chính xác: **mAP@0.5 đạt 92.49%**, Precision 85.62%.
  - Hiệu năng: **25-30 FPS** (Realtime) trên phần cứng phổ thông.
- **Hệ thống tích hợp:**
  - **Kỹ thuật:** Tracking ID + Vùng quan tâm (ROI) để chống đếm trùng.
  - **Ứng dụng Web:** Backend FastAPI + Frontend Next.js 14, hỗ trợ giám sát đa luồng Camera.

1.2 Bài toán đặt ra & Thách thức

- **Bối cảnh thực tế:**
  - Nhu cầu cấp thiết về dữ liệu lưu lượng để quy hoạch và cảnh báo ùn tắc tại các đô thị lớn (Hà Nội, TP.HCM).
  - **Hạn chế giải pháp cũ:** Đếm thủ công tốn kém nhân lực; Cảm biến vật lý chi phí lắp đặt/bảo trì cao.
  - **Giải pháp AI:** Tận dụng hạ tầng Camera giám sát có sẵn $\rightarrow$ Tiết kiệm, độ chính xác cao.

- **Các thách thức chính của đề tài:**
  - **Nhận diện đa dạng:** Phân loại chính xác 4 loại xe trong điều kiện thời tiết/ánh sáng/góc quay phức tạp tại VN.
  - **Chính xác dữ liệu:** Đếm đúng qua vùng ROI, xử lý bài toán phương tiện bị che khuất và xuất hiện nhiều lần (tránh đếm trùng).
  - **Hiệu năng Realtime:** Xử lý song song nhiều Camera đồng thời với độ trễ thấp.
  - **Triển khai:** Xây dựng hệ thống Full-stack hoàn chỉnh, ổn định.

2.1 Phương pháp
2.1.1 Lý thuyết về Mạng Nơ-ron Tích chập

Tổng quan YOLOv8m

YOLOv8m về bản chất là một mạng CNN sâu, được cấu thành từ các khối xây dựng cơ bản nhằm chuyển đổi dữ liệu hình ảnh thô thành các đặc trưng có ý nghĩa.

Các thành phần cốt lõi:

- **Lớp Tích chập:** Đóng vai trò là bộ trích xuất đặc trưng. Phát hiện từ các đặc trưng đơn giản (cạnh, góc, màu sắc) đến đặc trưng phức tạp (bánh xe, khuôn mặt).
- **Lớp Gộp:** Sử dụng kỹ thuật Max Pooling để giảm kích thước dữ liệu.
  - Giảm tham số tính toán, chống quá khớp (Overfitting).
  - Giữ đặc trưng quan trọng, đảm bảo tính bất biến với dịch chuyển nhỏ.
- **Hàm kích hoạt (SiLU):** Định nghĩa bởi công thức $f(x) = x \cdot \sigma(x)$. Đảm bảo tính phi tuyến tính, giúp mạng học được các dữ liệu phức tạp mà biến đổi tuyến tính không làm được.

2.1.2 Kiến trúc chi tiết YOLOv8m

Hệ thống gồm 3 khối chức năng chính:

1. Backbone (Trích xuất đặc trưng)

- **CSPDarknet + C2f Module:** Chia luồng tín hiệu thành hai nhánh (tính toán sâu & đi tắt) giúp làm giàu gradient.
- **SPPF:** Mở rộng trường tiếp nhận nắm bắt được ngữ cảnh toàn cảnh của vật thể.

2. Neck (Hợp nhất đặc trưng)

- **PANet:** Trộn lẫn đặc trưng ngữ nghĩa (tầng sâu) và vị trí (tầng nông).
- Tăng khả năng nhận diện các vật thể ở nhiều kích thước khác nhau.

3. Head (Dự đoán)

- **Decoupled Head:** Tách riêng nhánh Phân loại và Hồi quy hộp.
- **Anchor-free Detection:** Dự đoán trực tiếp tâm vật thể, không dùng khung neo định sẵn.

2.1.3 Phương pháp Fine-tuning

Chiến lược Huấn luyện (Full Fine-tuning)

Khởi tạo mô hình từ trọng số **COCO** (`yolov8m.pt`), sau đó huấn luyện lại tất cả các layers (bao gồm cả Backbone và Head) trên tập dữ liệu phương tiện giao thông Việt Nam.

Kỹ thuật áp dụng

- **No-freeze:** Cập nhật trọng số toàn bộ mạng, không đóng băng layer nào.
- **Learning Rate:** Khởi tạo thấp ($lr = 0.01$) kết hợp *Cosine Annealing*.
- **Mục tiêu:** Tinh chỉnh nhẹ nhàng, tránh phá vỡ các đặc trưng cũ.

Hiệu quả đạt được

- ✓ **Hội tụ nhanh:** Tận dụng điểm khởi đầu tốt từ COCO.
- ✓ **Học sâu:** Tối ưu cả đặc trưng cấp thấp (Backbone) và cấp cao (Head).
- ✓ **Tổng quát hóa:** Kết hợp *Augmentation* & *Early Stopping* giúp giảm Overfitting trên 1547 ảnh.

2.1.4 Các chỉ số đánh giá hiệu quả

1. Intersection over Union (IoU)

Đo độ chồng lấn giữa dự đoán (B_p) và thực tế (B_{gt}):

$$\text{IoU} = \frac{\text{Diện tích phần giao}}{\text{Diện tích phần hợp}}$$

2. Precision (Độ chính xác)

Tỉ lệ dự đoán đúng trong số các box mô hình đã vẽ:

$$P = \frac{TP}{TP + FP}$$

3. Recall (Độ nhạy)

Tỉ lệ vật thể tìm được trong số vật thể thực tế có trong ảnh:

$$R = \frac{TP}{TP + FN}$$

4. Mean Average Precision (mAP)

Trung bình AP trên tất cả các lớp (Quan trọng nhất):

- **mAP@0.5:** Ngưỡng IoU = 0.5.
- **mAP@0.5:0.95:** Ngưỡng IoU chạy từ 0.5 đến 0.95.

2.1.5 Dữ liệu và Tiền xử lý

Tổng quan bộ dữ liệu

- **Nguồn:** Giao thông Việt Nam (1547 ảnh).
- **Đặc điểm:** Đa dạng ánh sáng và môi trường thực tế.
- **4 Lớp mục tiêu:** Car, Motor, Truck, Bus.

Tiền xử lý (Preprocessing)

- **Resize:** Chuẩn hóa về 640×640 pixels.
- **Lọc và gán lại nhãn:** Quy hoạch nhãn về 4 lớp đích.
- **Phân chia dữ liệu:** Chia thành 3 tập con tỷ lệ 80:10:10.

Mẫu dữ liệu thực tế sau khi gán nhãn:

Car

Motor

Truck

Bus

2.2 Triển khai
2.2.1 Môi trường và Công cụ phát triển

1. Lõi AI & Xử lý dữ liệu

- **Ngôn ngữ:** Python 3.10+.
- **Deep Learning:** PyTorch, Ultralytics YOLOv8.
- **Xử lý ảnh:** OpenCV (vẽ box, cắt frame).
- **Công cụ khác:** NumPy, Pandas, Matplotlib (thống kê biểu đồ).

2. Nền tảng Ứng dụng Web

- **Backend:** FastAPI (Async, REST API, WebSocket).
- **Frontend:** Next.js 14, TypeScript, Tailwind CSS.
- **Database:** PostgreSQL + SQLAlchemy (Async).

3. Tính năng mở rộng: AI Chatbot (Tư vấn luật)

Sử dụng kiến trúc **RAG** (Retrieval-Augmented Generation): Tích hợp LangChain, Google Gemini và ChromaDB cung cấp tư vấn về luật giao thông.

2.2.2 Cấu trúc và Tổ chức mã nguồn

1. **Phân hệ Cấu hình (Configs):**
   - Quản lý tham số qua tệp **.yaml**, tách biệt hoàn toàn mã nguồn khỏi siêu tham số.
   - **`model_config` & `training_config`:** Thiết lập ngưỡng tin cậy (0.25), classes, Epochs (100), Batch (16)...
   - **`app.yaml`:** Cấu hình Runtime, đường dẫn video và vùng quan tâm (ROI).
2. **Phân hệ Mô hình (Model):**
   - **`model_trainer.py`:** Quản lý Fine-tuning YOLOv8 và đánh giá (Precision, Recall).
3. **Phân hệ Backend (FastAPI):**
   - Kiến trúc **MVC**; **`api/`** (Endpoints); **`road_services`** (Xử lý đa tiến trình).
   - **`rag_services`:** Tích hợp Chatbot AI (LangChain, ChromaDB).
4. **Phân hệ Phân tích (Analysis):**
   - **`analyze.py`:** Pipeline xử lý, Peak detection; **`load_data`** (Binary-safe tail reading).
   - **`visualize.py`:** Trực quan hóa Realtime (In-place updates).
5. **Phân hệ Frontend (Next.js 14):**
   - Sử dụng App Router; **`components/`** (VideoPlayer, Charts); **`lib/`** (TypeScript Utils).

2.2.3 Quy trình Xây dựng Mô hình

1. **Chuẩn bị và Tiền xử lý dữ liệu:**
   - **Dữ liệu:** 1547 ảnh giao thông VN (4 lớp: Car, Motor, Truck, Bus).
   - **Xử lý:** Chuẩn hóa nhãn [0, 1]; phân chia tỷ lệ **80:10:10** (Train/Val/Test).
   - Tạo file `data.yaml` định nghĩa đường dẫn và danh sách lớp.
2. **Khởi tạo mô hình Pre-trained:**
   - Áp dụng kỹ thuật **Fine-tuning** trên nền **YOLOv8m** (COCO, $\sim$25.9M tham số).
   - Tải weights `yolov8m.pt`, huấn luyện lại toàn bộ layers (không đóng băng).
3. **Cấu hình tham số huấn luyện (Hyperparameters):**
   - **Thiết lập:** 100 Epochs (Patience: 30), Batch 16, Size 640x640.
   - **Chiến lược:** Learning rate 0.01 (Cosine annealing); Augmentation (Mosaic, HSV).
4. **Quá trình Fine-tuning:**
   - Thực thi bởi `model_trainer.py`: Forward/Backward propagation cập nhật trọng số.
   - Tự động lưu checkpoint tối ưu (`best.pt`) dựa trên chỉ số **mAP@0.5:0.95**.
5. **Đánh giá và Tối ưu hóa:**
   - **Đánh giá:** Tính Precision, Recall, mAP trên tập Test độc lập.
   - **Tối ưu:** Đo kiểm FPS thực tế; tinh chỉnh ngưỡng Confidence/IoU để cân bằng hiệu năng.

2.2.4 Phân tích Dữ liệu

1. **Kiến trúc Pipeline xử lý (Real-time):**
   - **Tối ưu hiệu năng:** Sử dụng kỹ thuật *Binary-safe tail reading* (chỉ đọc phần cuối file), đạt độ phức tạp **O(1)**.
   - **Xử lý:** Chuẩn hóa DataFrame và **Resampling** dữ liệu theo chuỗi thời gian (mặc định 1 phút).
2. **Tính toán thống kê toàn diện:**
   - **Phân tích cơ cấu:** Tính toán tỷ trọng lưu lượng từng loại xe (Car, Motor, Truck, Bus).
   - **Phân tích xu hướng:** Gom nhóm dữ liệu theo khung giờ (0-23h) để xác định giờ cao điểm/thấp điểm.
3. **Phát hiện điểm nóng (Hotspot Detection):**
   - **Thuật toán:** Sử dụng Rolling Mean và Standard Deviation trên cửa sổ trượt (window=5).
   - **Cảnh báo:** Áp dụng nguyên lý **3-sigma** (ngưỡng $> mean + 3\sigma$) để tự động phát hiện mật độ tăng đột biến.
4. **Xuất dữ liệu và Trực quan hóa:**
   - **Đa định dạng:** Xuất CSV (cho Excel/BI) và JSON (chuẩn hóa cho API Backend/Frontend).
   - **Visualization:** Module `visualize.py` cập nhật biểu đồ Realtime (In-place updates) mượt mà.

2.2.5 Hệ thống đếm xe

1. **Kiến trúc Đa tiến trình (Multiprocessing):**
   - Sử dụng **Python Multiprocessing** để tránh GIL: Mỗi camera chạy trên một Process độc lập, ngăn chặn lỗi dây chuyền.
   - Dùng `Manager.dict()` chia sẻ dữ liệu (Count, FPS) an toàn giữa các tiến trình.
2. **Quy trình Phát hiện và Tracking (YOLOv8):**
   - **Tối ưu đầu vào:** Resize frame xuống 480x270 và Skip frame (3) để tăng tốc độ xử lý.
   - **Tracking:** Gọi `model.track(persist=True)` để gán và duy trì ID duy nhất cho từng phương tiện qua các frame.
3. **Thuật toán Đếm với vùng quan tâm (ROI):**
   - Cấu hình vùng đếm linh hoạt (Polygon/Rect) trong `app.yaml`.
   - **Logic:** Chỉ đếm khi tâm (center point) của box di chuyển từ *ngoài* vào *trong* ROI (đảm bảo không đếm trùng).
4. **Lưu trữ và Cập nhật dữ liệu:**
   - **Realtime:** Cập nhật liên tục để API truy xuất ngay lập tức.
   - **Bền vững:** Tự động lưu file JSON định kỳ (60s), phân chia theo 24 khung giờ trong ngày.
5. **Tối ưu hóa Hiệu năng:**
   - Lọc kết quả với ngưỡng tin cậy **Confidence** $\geq$ **0.4** (giảm False Positives).
   - Tự động giải phóng bộ nhớ các Track ID cũ không còn xuất hiện.

2.2.6 Xây dựng Web Application

1. **Backend API (FastAPI - Python):**
   - **Hybrid Architecture:** Kết hợp RESTful API (lấy thống kê/lịch sử) và WebSockets (stream Video/Data thời gian thực).
   - **Data Management:** Sử dụng SQLAlchemy (Async/Await) hỗ trợ PostgreSQL/SQLite; Cấu hình CORS an toàn.
2. **Frontend (Next.js 14 & TypeScript):**
   - **Công nghệ:** App Router, Tailwind CSS (Responsive), React Hooks quản lý state.
   - **Components:** Tái sử dụng cao (`VideoPlayer` stream, `RealtimeCharts` hiển thị biểu đồ động).
3. **Triển khai và Cấu hình:**
   - **Config:** Quản lý tập trung qua file `.env` (API keys, DB URL) cho từng môi trường (Dev/Prod).
   - **Production:** Hỗ trợ **Docker** containerization; Tối ưu hóa build (Code splitting, Minification).
4. **Tính năng mở rộng - AI Chatbot (RAG):**
   - **Mục tiêu:** Tư vấn luật giao thông qua ngữ nghĩa (Semantic search).
   - **Stack:** LangChain (Pipeline) + Google Gemini (LLM) + ChromaDB (Vector Database).

2.2.7 Quy trình triển khai tổng thể

1. **Chuẩn bị Môi trường và Dữ liệu:**
   - **Env:** Python 3.10+, Node.js 18+; Cài đặt dependencies (`requirements.txt`).
   - **Data:** Kiểm tra nhãn YOLO (4 lớp), chia tập ngẫu nhiên **80:10:10**.
2. **Huấn luyện Mô hình (Training):**
   - Cấu hình `training_config.yaml`; Fine-tune **YOLOv8m** (Pre-trained).
   - Validate trên tập Test độc lập; Xuất trọng số tối ưu (`best.pt`).
3. **Thiết lập Backend:**
   - Cấu hình Video Source/ROI; Khởi tạo DB (PostgreSQL/SQLite) và ChromaDB (Chatbot).
   - Khởi chạy **FastAPI Server**, kiểm tra các Endpoints.
4. **Khởi chạy Hệ thống Đếm xe:**
   - Server tự động kích hoạt **Multiprocessing** (Analyzer) cho từng Camera.
   - Kiểm tra ghi Log file và dữ liệu API Realtime (`/info`, `/frames`).
5. **Xây dựng và triển khai Frontend:**
   - Build tối ưu (`npm run build`); Deploy lên Hosting (Vercel/Netlify) hoặc Server.
   - Kiểm thử tích hợp kết nối WebSocket và hiển thị Video stream.
6. **Vận hành và bảo trì:**
   - **Monitor:** Giám sát chỉ số FPS, Resource (CPU/RAM); Backup định kỳ.
   - **Scale:** Retrain model khi có dữ liệu mới; Mở rộng ngang khi tăng tải.

3.1 Kết quả huấn luyện mô hình

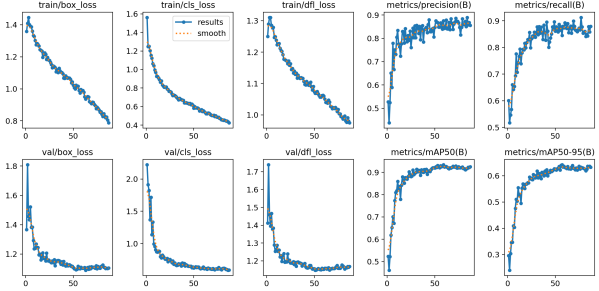
- **Cấu hình Huấn luyện:**
  - **Mô hình:** Fine-tuning **YOLOv8m** (từ trọng số COCO).
  - **Tham số:** 87 Epochs, Batch size 16, Input 640x640.
  - **Chiến lược:** Learning rate 0.01 (Cosine annealing).
- **Kết quả đánh giá mô hình trên tập Test:**

Chỉ số (Metric)	Giá trị	Ý nghĩa
Precision	**85.62%**	Tỷ lệ dự đoán đúng cao
Recall	**87.95%**	Khả năng phát hiện phương tiện tốt
mAP@0.5	**92.49%**	Độ chính xác xuất sắc ở ngưỡng tiêu chuẩn
mAP@0.5:0.95	**63.21%**	Định vị bounding box tốt ở nhiều ngưỡng

- **Phân tích quá trình học:**
  - **Hội tụ (Convergence):** Các đường Loss giảm và ổn định sau **30-40 epochs**.
  - **Ma trận nhầm lẫn:** Phân loại tốt 4 lớp; tỷ lệ nhầm lẫn thấp (chủ yếu giữa Car/Truck).

3.2 Phân tích hiệu năng theo từng Lớp

Lớp (Class)	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Bus	90.91%	88.24%	**93.31%**	72.50%
Car	90.84%	87.88%	92.06%	**75.60%**
Truck	83.31%	86.93%	88.26%	66.31%
Motor	83.35%	80.64%	85.48%	51.08%



Biểu đồ quá trình huấn luyện

Ma trận nhầm lẫn

3.3 Hiệu năng xử lý Realtime

- **Chiến lược Tối ưu hóa:**
  - **Resize Input:** Giảm kích thước frame xuống **480x270 pixels**.
  - **Giảm tải:** Áp dụng Frame skipping (`skip_frames=3`) và ngưỡng tin cậy **0.4**.
- **Kết quả Đo lường Hiệu năng:**

Kịch bản triển khai	Phần cứng/Cơ chế	Tốc độ (FPS)
1 Camera	GPU NVIDIA / CPU Đa lõi	**25 – 30 FPS**
Nhiều Camera (2 Cam)	Python Multiprocessing	**20 – 25 FPS**

- **Đánh giá Kiến trúc Hệ thống:**
  - **Multiprocessing:** Phân tán tải xử lý lên các core CPU riêng biệt, duy trì độ trễ thấp.
  - **Shared Memory (`Manager.dict`):** Đảm bảo API truy xuất dữ liệu đếm xe tức thì (non-blocking), không làm gián đoạn luồng xử lý video.

→ Hệ thống đáp ứng tốt yêu cầu giám sát giao thông thời gian thực.

3.4 Kết quả Hệ thống Đếm Xe

- **Hiệu quả Thực nghiệm:**
  - Độ chính xác đạt $>$ **90%** so với phương pháp đếm thủ công trên video thực tế.
  - Cơ chế **Tracking ID** kết hợp **ROI** hoạt động hiệu quả, đảm bảo mỗi phương tiện chỉ được đếm duy nhất một lần.
- **Thách thức và Hạn chế:**
  - **Tốc độ cao:** Phương tiện di chuyển quá nhanh có thể gây mất dấu (lost track) $\rightarrow$ Hệ thống gán ID mới gây đếm trùng.
  - **Che khuất (Occlusion):** Xe bị che khuất một phần làm gián đoạn quá trình tracking liên tục.
  - **Môi trường:** Điều kiện ánh sáng thay đổi (Mưa, Ban đêm) làm giảm độ tin cậy.
  - **Dữ liệu (Domain Shift):** Góc quay hoặc độ sáng khác biệt nhiều so với tập huấn luyện dẫn đến bỏ sót phương tiện (False Negatives).
- **Kết luận:** Tuy tồn tại các yếu tố nhiễu ngoại cảnh, các kỹ thuật tối ưu hóa đã giúp hệ thống duy trì độ ổn định và chính xác cao trong đa số các kịch bản kiểm thử.

3.5 Kết quả Phân tích Dữ liệu

- **Hiệu năng Xử lý Log (Log Processing):**
  - **Thử nghiệm:** Hệ thống hoạt động ổn định với các file log lớn (hàng trăm MB).
  - **Giải pháp:** Ứng dụng kỹ thuật *Binary-safe tail reading*.
  - **Kết quả:** Thời gian đọc đạt độ phức tạp **O(1)** (hằng số) $\rightarrow$ Đảm bảo xử lý Realtime ngay cả khi kích thước file tăng dần theo thời gian.
- **Phát hiện Điểm nóng (Hotspot Detection):**
  - **Thuật toán:** Sử dụng phương pháp thống kê **3-sigma** (Mean + 3 Std Dev).
  - **Hiệu quả:**
    - Xác định chính xác các thời điểm lưu lượng tăng đột biến.
    - Tự động gửi cảnh báo khi vượt ngưỡng bình thường, hỗ trợ điều phối giao thông.

3.6 Kết quả Web Application

- **Hiệu năng Backend (FastAPI):**
  - **Tốc độ:** REST API đạt thời gian phản hồi trung bình $<$ **100ms**.
  - **Realtime:** WebSocket duy trì độ trễ thấp, đảm bảo stream video và cập nhật số liệu liên tục, không gián đoạn.
- **Trải nghiệm Frontend (Next.js):**
  - **Tối ưu hóa:** Code splitting & Tree shaking giúp tốc độ tải trang ban đầu nhanh.
  - **Trực quan hóa:** Các biểu đồ (*LineChart, CounterChart*) hiển thị dữ liệu rõ ràng, hỗ trợ phân tích xu hướng hiệu quả.
- **Tính năng mở rộng (AI Chatbot):**
  - Hoạt động ổn định trên kiến trúc **RAG** (Retrieval-Augmented Generation).
  - Cung cấp khả năng tra cứu và tư vấn Luật giao thông chính xác dựa trên tài liệu tham khảo.

3.7 Thảo luận

- **Hiệu quả đạt được:**
  - **Mô hình:** Kỹ thuật Fine-tuning phát huy hiệu quả trên dữ liệu nhỏ (1547 ảnh), đạt độ chính xác cao (**mAP@0.5 = 92.49%**).
  - **Hệ thống:** Độ chính xác đếm xe $>$ **90%**; Duy trì hiệu năng Realtime trên nhiều camera nhờ tối ưu hóa phần cứng.
- **Hạn chế & Thách thức:**
  - **Nhiễu:** Khó nhận diện khi phương tiện bị che khuất hoặc ánh sáng yếu.
  - **Tracking:** Có thể mất dấu (Lost track) đối với phương tiện di chuyển tốc độ quá cao.
  - **Thử nghiệm:** Cần kiểm thử thêm trên nhiều điều kiện thời tiết đa dạng để tăng tính tổng quát.
- **Tổng kết:** Hệ thống đã chứng minh được tính thực tiễn, hiệu suất ổn định và kiến trúc dễ dàng mở rộng, tạo nền tảng vững chắc cho các phát triển tương lai.

4.1 Kết luận

- **Xây dựng thành công mô hình nhận diện:**
  - Ứng dụng phương pháp **Fine-tuning** trên YOLOv8m với dữ liệu giao thông Việt Nam (1547 ảnh).
  - Kết quả ấn tượng: mAP@0.5 đạt 92.49%, Precision 85.62%, Recall 87.95%.
- **Triển khai hệ thống giám sát toàn diện:**
  - **Kiến trúc đa tầng:** Tích hợp AI Core, Pipeline phân tích (Hotspot detection), Backend (FastAPI) và Frontend (Next.js).
  - **Hiệu năng:** Xử lý Realtime đa camera (25-30 FPS); Độ chính xác đếm xe $>$ 90% nhờ kỹ thuật Tracking ROI.
- **Đóng góp công nghệ:**
  - Chứng minh tính hiệu quả của Fine-tuning đối với bài toán dữ liệu hạn chế.
  - Các kỹ thuật tối ưu hóa (Multiprocessing, Frame skipping, Binary-safe I/O) đảm bảo tính ổn định và khả năng mở rộng trong thực tế.

4.2 Hạn chế

- **Chênh lệch dữ liệu:**
  - Sự khác biệt lớn về góc quay, ánh sáng giữa tập Training và môi trường thực tế dẫn đến việc bỏ sót phương tiện.
- **Hạn chế với phương tiện tốc độ cao:**
  - Thuật toán Tracking có thể mất dấu khi xe di chuyển quá nhanh $\rightarrow$ Hệ thống gán ID mới, gây ra lỗi đếm trùng.
- **Vấn đề bị che khuất:**
  - Độ chính xác giảm khi phương tiện bị che khuất một phần bởi vật thể hoặc các xe khác trong dòng giao thông đông đúc.
- **Điều kiện môi trường đa dạng:**
  - Hiệu năng chưa được tối ưu hoàn toàn trong các điều kiện khắc nghiệt (mưa lớn, sương mù, ban đêm).

4.3 Hướng phát triển

- **Nâng cao dữ liệu và khả năng thích ứng:**
  - **Đa dạng hóa dữ liệu:** Bổ sung mẫu ở nhiều góc quay, điều kiện thời tiết (mưa, sương mù) để tăng tính tổng quát.
  - **Kỹ thuật:** Áp dụng Domain Adaptation và xử lý ảnh thiếu sáng để cải thiện nhận diện ban đêm.
- **Tối ưu hóa Thuật toán và Hiệu năng:**
  - **Tracking:** Nâng cấp lên DeepSORT hoặc các mô hình Deep Learning để bắt phương tiện tốc độ cao chính xác hơn.
  - **Edge Computing:** Sử dụng Quantization và Model Pruning để giảm nhẹ mô hình, triển khai trên thiết bị cấu hình thấp.
- **Mở rộng Tính năng và Quy trình vận hành:**
  - **Phân tích nâng cao:** Thêm tính năng dự báo lưu lượng, phát hiện tai nạn và vi phạm giao thông.
  - **MLOps:** Xây dựng pipeline tự động hóa (Data collection $\rightarrow$ Retrain $\rightarrow$ Deploy) để cập nhật mô hình định kỳ.

Cảm ơn Thầy và các bạn đã lắng nghe!