# **TRƯỜNG ĐẠI HỌC SƯ PHẠM ĐÀ NẴNG**

# **KHOA TOÁN-TIN**



# **BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Đề tài: Nghiên cứu và xây dựng hệ thống nhận diện**

**biển số xe sử dụng mạng nơ-ron tích chập YOLOv4-tiny**

**Giáo viên hướng dẫn : Phạm Anh Phương**

**Sinh viên thực hiện : Đinh Xuân Hiệp**

**Lớp : 23CNTT1**

*Ngày… tháng… năm 2025*

# LỜI CẢM ƠN

Trong thời gian thực hiện đồ án với sự giúp đỡ tạo điều kiện của trường Đại học Sư Phạm – Đại học Đà Nẵng, sự góp ý của các bạn và đặc biệt là sự quan tâm hướng dẫn trực tiếp, chỉ bảo tận tình của thầy giáo TS. Phạm Anh Phương em đã hoàn thành đề tài ………………………………..………………………………..………………………………..

Em xin chân thành cảm ơn!

**Nhận xét của giáo viên**

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

*Đà Nẵng, ngày … tháng … năm 2025*

**Giáo viên hướng dẫn**

**Phạm Anh Phương**

LỜI CAM ĐOAN

*Tôi xin cam đoan:*

1. *Những nội dung trong báo cáo này là do tôi thực hiện dưới sự hướng dẫn trực tiếp của thầy Phạm Anh Phương*
2. *Mọi tham khảo dùng trong đồ án thực tập đều được trích dẫn rõ ràng và trung thực tên tác giả, tên công trình, thời gian, địa điểm công bố.*
3. *Mọi sao chép không hợp lệ, vi phạm quy chế đào tạo, hay gian trá, tôi xin chịu hoàn toàn trách nhiệm.*

*Đà Nẵng, ngày tháng năm 2025*

**Sinh viên thực hiện**

**Đinh Xuân Hiệp**

**MỤC LỤC**

[MỞ ĐẦU 1](#_Toc215693991)

[**1. Lý do chọn đề tài 1**](#_Toc215693992)

[**a. Bối cảnh giao thông thông minh tại Việt Nam 1**](#_Toc215693993)

[**b. Lý do chọn đề tài 1**](#_Toc215693994)

[**2. Mục tiêu, đối tượng và phạm vi nghiên cứu 1**](#_Toc215693995)

[**a. Mục tiêu đề tài 1**](#_Toc215693996)

[**b. Đối tượng nghiên cứu 2**](#_Toc215693997)

[**c. Phạm vi đề tài 2**](#_Toc215693998)

[**3. Bố cục báo cáo 2**](#_Toc215693999)

[CHƯƠNG 1. TỔNG QUAN VỀ HỆ THỐNG NHẬN DIỆN BIỂN SỐ XE 3](#_Toc215694000)

[**1.1. Quy trình thực hiện tổng quát 3**](#_Toc215694001)

[**1.2. Tổng quan về Mạng nơ-ron tích chập (CNN) 4**](#_Toc215694002)

[**1.2.1. Cấu trúc cơ bản của mạng CNN 4**](#_Toc215694003)

[**1.2.2. Các lớp thành phần (Convolution, Pooling, FC) 4**](#_Toc215694004)

[**1.2.3. Các hàm kích hoạt (Activation Functions: ReLU, Leaky ReLU) 5**](#_Toc215694005)

[**1.3. Bài toán phát hiện đối tượng (Object Detection) 5**](#_Toc215694006)

[**1.3.1. One-stage vs Two-stage Detectors 5**](#_Toc215694007)

[**1.3.2. Các chỉ số đánh giá mô hình (IoU, Precision, Recall, F1-Score) 6**](#_Toc215694008)

[**1.3.3. Độ chính xác trung bình (mAP) và tốc độ (FPS) 7**](#_Toc215694009)

[**1.4. Sự phát triển của họ mô hình YOLO 7**](#_Toc215694010)

[**1.4.1. YOLOv1, YOLOv2 và YOLOv3 7**](#_Toc215694011)

[**1.4.2. Những cải tiến đột phá trong YOLOv4 8**](#_Toc215694012)

[**1.5. Kiến trúc mạng YOLOv4-tiny 8**](#_Toc215694013)

[**1.5.1. So sánh kiến trúc YOLOv4 và YOLOv4-tiny 8**](#_Toc215694014)

[**1.5.2. Khối CSP (Cross Stage Partial) trong Backbone 9**](#_Toc215694015)

[**1.5.3. Cấu trúc FPN (Feature Pyramid Network) rút gọn 9**](#_Toc215694016)

[**1.5.4. Hàm mất mát CIoU Loss (Complete IoU) 10**](#_Toc215694017)

[**1.5.5. Kỹ thuật tăng cường dữ liệu Mosaic 10**](#_Toc215694018)

[CHƯƠNG 2. XÂY DỰNG HỆ THỐNG VÀ HUẤN LUYỆN MÔ HÌNH 11](#_Toc215694019)

[**2.1. Thu thập và xử lý dữ liệu (Dataset) 11**](#_Toc215694020)

[**2.1.1. Nguồn dữ liệu và đặc điểm biển số xe Việt Nam 11**](#_Toc215694021)

[**2.1.2. Gán nhãn dữ liệu với công cụ LabelImg 11**](#_Toc215694022)

[**2.1.3. Phân chia tập dữ liệu (Train/Val/Test) 12**](#_Toc215694023)

[**2.2. Môi trường và công cụ cài đặt 12**](#_Toc215694024)

[**2.2.1. Google Colab và Framework Darknet 12**](#_Toc215694025)

[**2.3. Cấu hình tham số huấn luyện (Training Configuration) 12**](#_Toc215694026)

[**2.3.1. Tinh chỉnh file cấu hình (.cfg) cho YOLOv4-tiny 12**](#_Toc215694027)

[**2.3.2. Tính toán Anchor Boxes bằng thuật toán K-means 13**](#_Toc215694028)

[**2.3.3. (Hyperparameters: LR, Batch size, Epochs) 13**](#_Toc215694029)

[**2.4. Quá trình huấn luyện mô hình 13**](#_Toc215694030)

[CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ 14](#_Toc215694031)

[**3.1. Dữ liệu thực tế (Dataset) 14**](#_Toc215694032)

[**3.2. Môi trường Huấn luyện (Infrastructure) 14**](#_Toc215694033)

[**3.3. Quá trình huấn luyện 15**](#_Toc215694034)

[**3.4. Kết quả huấn luyện 15**](#_Toc215694035)

[**3.4.1. Về quá trình hội tụ (Hàm mất mát - Loss) 15**](#_Toc215694036)

[**3.4.2. Độ chính xác (mAP - Mean Average Precision) 16**](#_Toc215694037)

[**3.4.3. Các chỉ số đánh giá chi tiết 16**](#_Toc215694038)

[**3.4.4. Tốc độ xử lý (FPS) 16**](#_Toc215694039)

[**3.5. Xây dựng ứng dụng Demo nhận diện biển số 17**](#_Toc215694040)

[**3.5.1. Công nghệ sử dụng 17**](#_Toc215694041)

[**3.5.2. Chức năng và Giao diện 17**](#_Toc215694042)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 18](#_Toc215694043)

[**1. Kết quả đạt được 18**](#_Toc215694044)

[**2. Những hạn chế còn tồn tại 18**](#_Toc215694045)

[**3. Hướng phát triển trong tương lai 18**](#_Toc215694046)

[TÀI LIỆU THAM KHẢO 19](#_Toc215694047)

[PHỤ LỤC 20](#_Toc215694048)

[**PHỤ LỤC 1: MÃ NGUỒN HUẤN LUYỆN MÔ HÌNH (MODEL TRAINING) 20**](#_Toc215694049)

[**PHỤ LỤC 2: MÃ NGUỒN ỨNG DỤNG WEB (FLASK) 24**](#_Toc215694050)

[**PHỤ LỤC 3: HƯỚNG DẪN CÀI ĐẶT VÀ SỬ DỤNG 29**](#_Toc215694051)

**MỤC LỤC ẢNH**

[**Hình 1.1:Sơ đồ quy trình thực hiện xây dựng hệ thống nhận diện biển số. 4**](#_Toc215693406)

[**Hình 1.2:Kiến trúc tổng quát của một mạng nơ-ron tích chập (CNN). 5**](#_Toc215693407)

[**Hình 1.3:Minh họa cách tính chỉ số IoU (Intersection over Union). 7**](#_Toc215693408)

[**Hình 1.4:Cấu trúc khối CSP (Cross Stage Partial) giúp giảm chi phí tính toán. 10**](#_Toc215693409)

[**Hình 2.1:bộ ảnh biển số xe nguồn greenparking 12**](#_Toc215693410)

[**Hình 2.2:Giao diện phần mềm LabelImg đang thực hiện gán nhãn biển số xe. 13**](#_Toc215693411)

[**Hình 3.1:Biểu đồ sự thay đổi của hàm mất mát theo số lần lặp. 17**](#_Toc215693412)

[**Hình 3.2:Giao diện phần mềm nhận diện biển số xe đã xây dựng. 18**](#_Toc215693413)

# **MỞ ĐẦU**

## **1. Lý do chọn đề tài**

### **a. Bối cảnh giao thông thông minh tại Việt Nam**

 Trong những năm gần đây, tốc độ đô thị hóa tại Việt Nam diễn ra vô cùng mạnh mẽ, đặc biệt là tại các thành phố lớn như Hà Nội, TP. Hồ Chí Minh và Đà Nẵng. Kéo theo đó là sự gia tăng chóng mặt của các phương tiện giao thông cá nhân, bao gồm cả xe máy và ô tô. Theo thống kê của Ủy ban An toàn Giao thông Quốc gia, số lượng phương tiện đăng ký mới liên tục tăng qua các năm, gây áp lực khổng lồ lên cơ sở hạ tầng giao thông hiện hữu.

  Trước thực trạng đó, việc phát triển Hệ thống Giao thông Thông minh (Intelligent Transportation Systems - ITS) được xem là giải pháp tất yếu. Trong kiến trúc ITS, bài toán Nhận diện biển số xe tự động (License Plate Recognition - LPR) đóng vai trò là thành phần cốt lõi. Công nghệ này là tiền đề cho hàng loạt ứng dụng quản lý hiện đại như: hệ thống thu phí không dừng (ETC), giám sát và phạt nguội vi phạm giao thông, quản lý bãi đỗ xe thông minh tại các khu đô thị, cũng như kiểm soát an ninh tại các chốt chặn cửa ngõ.

### **b. Lý do chọn đề tài**

Từ bối cảnh thực tế trên, việc nghiên cứu và phát triển một hệ thống nhận diện biển số xe hoạt động hiệu quả trên thiết bị có tài nguyên hạn chế là một yêu cầu cấp thiết. Các phương pháp xử lý ảnh truyền thống (như Haar Cascade, SVM) tuy nhẹ nhưng độ chính xác thấp trong điều kiện môi trường phức tạp (thiếu sáng, mưa, biển số nghiêng). Trong khi đó, các mô hình học sâu (Deep Learning) như R-CNN tuy chính xác cao nhưng lại quá nặng nề để chạy thời gian thực trên thiết bị biên.

Để giải quyết bài toán cân bằng giữa **tốc độ xử lý** và **độ chính xác** trên phần cứng cấu hình thấp, đồ án lựa chọn nghiên cứu và ứng dụng mô hình **YOLOv4-tiny**. Đây là phiên bản rút gọn của YOLOv4, được tối ưu hóa đặc biệt cho các thiết bị biên, cho phép đạt tốc độ xử lý thời gian thực (Real-time) với độ chính xác tin cậy.

## **2. Mục tiêu, đối tượng và phạm vi nghiên cứu**

### ***a*. Mục tiêu đề tài**

Đồ án tập trung nghiên cứu và phát triển giải pháp nhận diện biển số xe tối ưu trên thiết bị có tài nguyên hạn chế, hướng tới các mục tiêu cụ thể sau:

* **Mục tiêu lý thuyết:** Hệ thống hóa cơ sở lý thuyết về Trí tuệ nhân tạo (AI), Học sâu (Deep Learning), đặc biệt là kiến trúc Mạng nơ-ron tích chập (CNN) và cơ chế hoạt động của thuật toán YOLOv4-tiny.
* **Mục tiêu thực nghiệm:**
  + Xây dựng thành công bộ dữ liệu (Dataset) biển số xe Việt Nam chất lượng cao. Quy trình bao gồm thu thập dữ liệu đa dạng, xử lý làm sạch và gán nhãn chuẩn theo định dạng YOLO để phục vụ huấn luyện.
  + Huấn luyện và tinh chỉnh (Fine-tuning) mô hình YOLOv4-tiny để nhận diện chính xác vị trí biển số của cả xe máy và ô tô trong các điều kiện môi trường khác nhau.

### **b. Đối tượng nghiên cứu**

Để thực hiện các mục tiêu trên, đồ án xác định các đối tượng nghiên cứu chính bao gồm:

* **Đối tượng nhận dạng:** Hình ảnh biển số xe cơ giới (xe máy và ô tô) lưu thông tại Việt Nam (bao gồm biển trắng, biển xanh, biển vàng; loại biển vuông và biển dài).
* **Đối tượng công nghệ:**
  + Các thuật toán Học sâu (Deep Learning) trong thị giác máy tính.
  + Mô hình mạng nơ-ron: YOLOv4-tiny (phiên bản tối ưu hóa tốc độ của YOLOv4).
  + Các công cụ và thư viện hỗ trợ: Ngôn ngữ lập trình Python, Framework (Darknet/TensorFlow/PyTorch), thư viện xử lý ảnh OpenCV.

### **c. Phạm vi đề tài**

Đồ án tập trung xây dựng một chương trình thử nghiệm (Prototype) thực hiện chức năng cốt lõi là **phát hiện và khoanh vùng (Detect)** vị trí biển số xe trong hình ảnh hoặc video.

Kết quả đầu ra của Demo là hình ảnh/video với các khung bao (Bounding Box) quanh biển số kèm theo độ tin cậy (Confidence score), chưa bao gồm chức năng nhận dạng ký tự (OCR) hay tích hợp hệ thống xử lý vi phạm hoàn chỉnh.

**3. Bố cục báo cáo**

  Báo cáo đồ án được cấu trúc thành 3 chương :

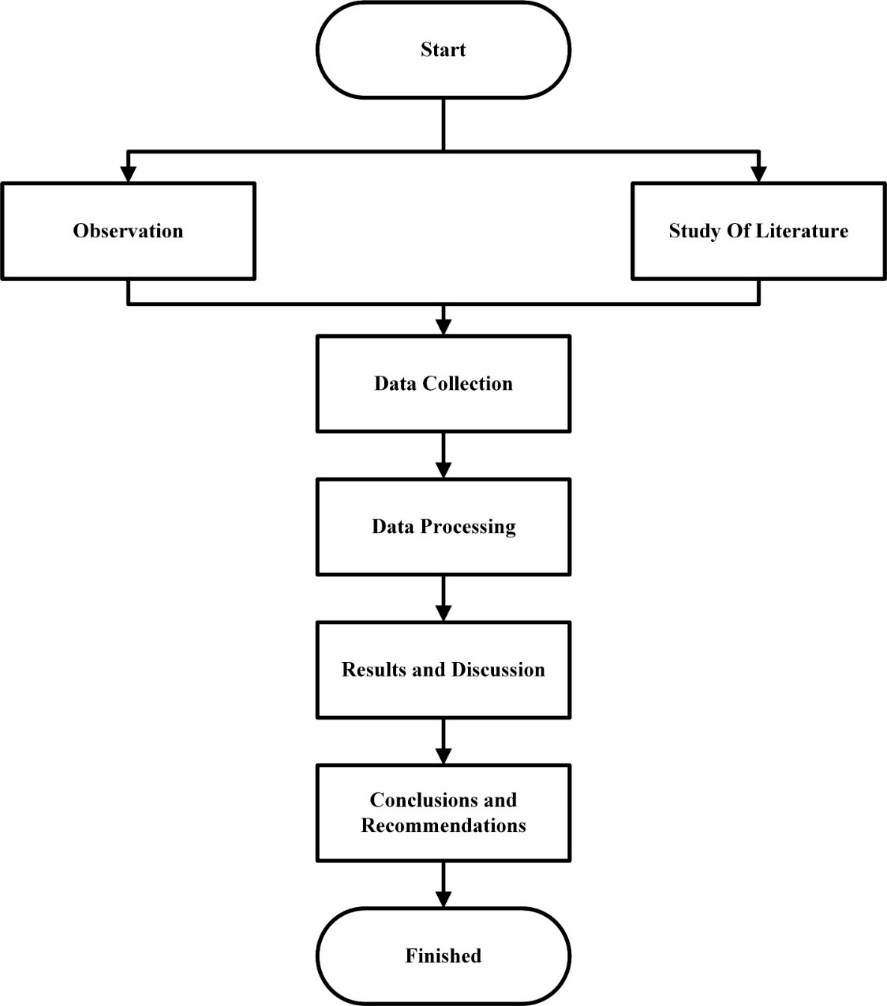
* **Chương 1:** Tổng quan về hệ thống nhận diện biển số xe
* **Chương 2:** Xây dựng hệ thống và huấn luyện mô hình
* **Chương 3:** Kết quả thực nghiệm và đánh giá

**CHƯƠNG 1. TỔNG QUAN VỀ HỆ THỐNG NHẬN DIỆN BIỂN SỐ XE**

## **1.1. Quy trình thực hiện tổng quát**

 Để xây dựng một mô hình Deep Learning hoàn chỉnh, việc tuân thủ một quy trình làm việc khoa học là vô cùng quan trọng. quy trình xây dựng hệ thống được thực hiện qua 5 giai đoạn nối tiếp chặt chẽ:

1. **Thu thập dữ liệu:** Tìm kiếm và tổng hợp nguồn ảnh thô từ thực tế và internet.
2. **Tiền xử lý và Gán nhãn:** Làm sạch dữ liệu và tạo nhãn chuẩn định dạng YOLO.
3. **Thiết lập môi trường:** Cài đặt Framework và tinh chỉnh các tham số mạng YOLOv4-tiny.
4. **Huấn luyện (Training):** Thực hiện quá trình học đặc trưng trên tập dữ liệu đã chuẩn bị.
5. **Kiểm thử và Đánh giá:** Sử dụng tập dữ liệu riêng biệt để đo lường các chỉ số hiệu năng.



Hình 1.1:Sơ đồ quy trình thực hiện xây dựng hệ thống nhận diện biển số.

## **1.2. Tổng quan về Mạng nơ-ron tích chập (CNN)**

### **1.2.1. Cấu trúc cơ bản của mạng CNN**

 Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là mô hình chuyên biệt cho dữ liệu hình ảnh, sử dụng cơ chế kết nối cục bộ và chia sẻ trọng số để giảm tham số huấn luyện mà vẫn giữ được thông tin không gian. CNN sử dụng cơ chế kết nối cục bộ và chia sẻ trọng số. Điều này cho phép CNN giảm thiểu đáng kể số lượng tham số cần huấn luyện, đồng thời giữ lại được các thông tin không gian quan trọng của ảnh đầu vào.



Hình 1.2:Kiến trúc tổng quát của một mạng nơ-ron tích chập (CNN).

  Kiến trúc CNN gồm 3 phần tuần tự:

* **Lớp tích chập (Convolution):** Trích xuất đặc trưng (cạnh, góc, hình dạng).
* **Lớp gộp (Pooling):** Giảm chiều dữ liệu và tăng tính bất biến.
* **Lớp kết nối đầy đủ (Fully Connected):** Phân loại hoặc dự đoán dựa trên đặc trưng đã trích xuất.

### **1.2.2. Các lớp thành phần (Convolution, Pooling, FC)**

**Lớp tích chập (Convolution Layer):** Là thành phần cốt lõi thực hiện trích xuất đặc trưng. Quá trình này sử dụng các bộ lọc (Kernel) kích thước nhỏ (thường là 3 hoặc 5 5) trượt trên ảnh đầu vào để thực hiện phép nhân chập, tạo ra các bản đồ đặc trưng (Feature Map) . Nhờ cơ chế **chia sẻ trọng số** (weight sharing), lớp này giảm thiểu số lượng tham số cần huấn luyện và duy trì khả năng phát hiện đặc trưng bất biến với vị trí.

**Lớp gộp (Pooling Layer):** Thường nằm xen kẽ giữa các lớp tích chập nhằm giảm kích thước không gian dữ liệu (Downsampling) và khối lượng tính toán . Có hai phương pháp gộp phổ biến là Max Pooling và Average Pooling.

* Max Pooling: Lấy giá trị lớn nhất trong cửa sổ trượt, giúp giữ lại các đặc trưng biên cạnh nổi bật nhất. Đây là phương pháp được ưu tiên trong các mô hình YOLO.
* Average Pooling: Tính giá trị trung bình, có tác dụng làm mượt đặc trưng.

**Lớp kết nối đầy đủ (Fully Connected Layer - FC):** Đóng vai trò bộ phân loại cuối cùng. Các bản đồ đặc trưng 2D/3D từ các lớp trước được **làm phẳng** (flatten) thành vector một chiều. Tại đây, các nơ-ron kết nối toàn phần với nhau để tổng hợp các đặc trưng cấp cao, từ đó thực hiện nhiệm vụ phân loại đối tượng (Classification) hoặc hồi quy tọa độ khung bao (Regression).

### **1.2.3. Các hàm kích hoạt (Activation Functions: ReLU, Leaky ReLU)**

 Hàm kích hoạt đóng vai trò vô cùng quan trọng trong việc đưa tính phi tuyến (non-linearity) vào mô hình, cho phép mạng nơ-ron học được các mối quan hệ phức tạp trong dữ liệu. Trong lịch sử phát triển của CNN, hàm Sigmoid và Tanh từng được sử dụng phổ biến nhưng chúng gặp phải vấn đề biến mất đạo hàm (Vanishing Gradient) khi mạng trở nên sâu hơn.

**Hàm ReLU:**

**Hàm Leaky ReLu:**

(*Trong đó α là hệ số rò rỉ, thường chọn α = 0.1)*

 Để khắc phục hạn chế này, hàm ReLU (Rectified Linear Unit) đã ra đời và trở thành chuẩn mực cho các mô hì1nh Deep Learning. Hàm ReLU được định nghĩa đơn giản là, tức là giữ nguyên giá trị nếu nó dương và gán bằng 0 nếu nó âm. Ưu điểm của ReLU là tính toán cực nhanh và giúp quá trình huấn luyện hội tụ nhanh hơn. Tuy nhiên, ReLU gặp phải vấn đề "Dying ReLU", tức là khi đầu vào nhỏ hơn 0, đạo hàm bằng 0 khiến nơ-ron ngừng học hoàn toàn và không thể khôi phục lại.

 Trong kiến trúc YOLOv4-tiny, hàm Leaky ReLU được sử dụng thay thế để giải quyết triệt để vấn đề trên. Leaky ReLU cho phép một lượng nhỏ gradient đi qua khi đầu vào nhỏ hơn 0 bằng cách nhân với một hệ số α nhỏ (thường là 0.1).

## **1.3. Bài toán phát hiện đối tượng (Object Detection)**

### **1.3.1. One-stage vs Two-stage Detectors**

**Mô hình hai giai đoạn (Two-stage Detectors):** Điển hình là R-CNN, Fast R-CNN và Faster R-CNN. Quy trình gồm 2 bước tuần tự: (1) Tạo vùng đề xuất (Region Proposals) có khả năng chứa đối tượng, (2) Tinh chỉnh và phân loại các vùng này .

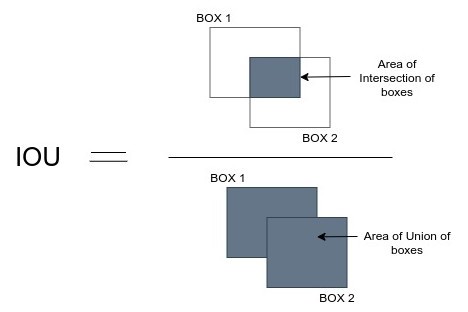
* Ưu điểm: Độ chính xác rất cao nhờ quá trình lọc kỹ.
* Nhược điểm: Tốc độ chậm (thường < 10 FPS), chi phí tính toán lớn, khó triển khai trên thiết bị nhúng.

**Mô hình một giai đoạn (One-stage Detectors):** Điển hình là YOLO (You Only Look Once) và SSD . Mô hình coi việc phát hiện là một bài toán hồi quy duy nhất. Mạng nơ-ron dự đoán trực tiếp tọa độ khung bao và xác suất lớp từ ảnh đầu vào trong một lần lan truyền (feed-forward).

* Ưu điểm: Tốc độ xử lý vượt trội (> 30 FPS), đáp ứng tốt yêu cầu thời gian thực.
* Nhược điểm: Độ chính xác có thể thấp hơn Two-stage đối với các vật thể nhỏ hoặc nằm chồng lấn, tuy nhiên các phiên bản YOLO mới đã khắc phục tốt vấn đề này.

### **1.3.2. Các chỉ số đánh giá mô hình (IoU, Precision, Recall, F1-Score)**

 Để đánh giá độ chính xác của một mô hình phát hiện đối tượng, chỉ số đầu tiên cần quan tâm là IoU (Intersection over Union). IoU đo lường mức độ trùng khớp giữa khung bao dự đoán (Predicted Box) và khung bao thực tế (Ground Truth Box).



Hình 1.3:Minh họa cách tính chỉ số IoU (Intersection over Union).

**Công Thức IoU:**

**Công thức Precision (độ chính xác) và Recall (độ phủ):**

*(Trong đó: TP là True Póitive, FP là False Positive, FN là False Negative)*

 Dựa trên IoU, ta xác định được các chỉ số cơ bản khác:

* **Precision (Độ chính xác):** Tỷ lệ số lần dự đoán đúng trên tổng số lần mô hình đưa ra dự đoán. Phản ánh độ tin cậy của mô hình .
* **Recall (Độ phủ):** Tỷ lệ số đối tượng tìm thấy trên tổng số đối tượng thực tế có trong ảnh. Phản ánh khả năng "không bỏ sót" của mô hình.
* **F1-Score:** Là trung bình điều hòa của Precision và Recall, dùng để đánh giá tổng quan khi cần sự cân bằng giữa độ chính xác và độ phủ.

### **1.3.3. Độ chính xác trung bình (mAP) và tốc độ (FPS)**

**mAP (mean Average Precision):** Là chỉ số quan trọng nhất để so sánh các mô hình. mAP được tính bằng cách lấy trung bình diện tích dưới đường cong Precision-Recall (AP) của tất cả các lớp đối tượng. Trong đồ án này, với 1 lớp "biển số", mAP chính là AP của lớp đó .

 Bên cạnh độ chính xác, tốc độ xử lý là yếu tố sống còn đối với các ứng dụng thực tế. Chỉ số **FPS (Frames Per Second**) đo lường số lượng khung hình mà mô hình có thể xử lý trong một giây. Đối với bài toán nhận diện biển số xe tại các trạm thu phí hay bãi xe, hệ thống cần đạt tốc độ tối thiểu 24-30 FPS để đảm bảo không bị trễ hình (latency) khi xe di chuyển.

## **1.4. Sự phát triển của họ mô hình YOLO**

### **1.4.1. YOLOv1, YOLOv2 và YOLOv3**

 Phiên bản đầu tiên YOLOv1 ra mắt năm 2016 đã gây tiếng vang lớn khi đề xuất ý tưởng chia ảnh thành một lưới times và dự đoán trực tiếp các bounding box tại mỗi ô lưới. Mặc dù tốc độ rất nhanh (45 FPS), YOLOv1 gặp khó khăn lớn trong việc định vị chính xác các vật thể nhỏ và các vật thể nằm sát nhau.

 Năm 2017, YOLOv2 (YOLO9000) được giới thiệu với nhiều cải tiến quan trọng, nổi bật nhất là việc áp dụng Anchor Boxes – các khung bao mẫu có kích thước định trước. Ngoài ra, YOLOv2 còn sử dụng lớp Batch Normalization giúp chuẩn hóa dữ liệu tại mỗi tầng.

 Năm 2018, YOLOv3 ra đời đánh dấu bước nhảy vọt về kiến trúc với mạng nền Darknet-53 và kỹ thuật **FPN (Feature Pyramid Network)**, cho phép dự đoán ở 3 tỷ lệ (lớn, trung bình, nhỏ). Điều này khắc phục đáng kể nhược điểm phát hiện vật thể nhỏ của các đời trước

**1.4.2. Những cải tiến đột phá trong YOLOv4**

 YOLOv4, ra mắt vào tháng 4 năm 2020 bởi Alexey Bochkovskiy và các cộng sự, đã tối ưu hóa toàn diện kiến trúc YOLO để đạt được "tốc độ và độ chính xác tối ưu" trên một GPU đơn lẻ. Tác giả đã phân loại các kỹ thuật cải tiến thành hai nhóm: "Bag of Freebies" và "Bag of Specials".

 **Bag of Freebies (BoF):** Các kỹ thuật tăng cường dữ liệu và huấn luyện (như Mosaic, CutMix, Class label smoothing) giúp cải thiện độ chính xác mà **không** làm tăng chi phí tính toán khi suy luận.

 **Bag of Specials (BoS):** Các module kiến trúc mạng (như Mish activation, SPP block, PANet) làm tăng nhẹ chi phí tính toán nhưng mang lại độ chính xác vượt trội.

**1.5. Kiến trúc mạng YOLOv4-tiny**

### **1.5.1. So sánh kiến trúc YOLOv4 và YOLOv4-tiny**

  YOLOv4-tiny là phiên bản nén của YOLOv4, được thiết kế chuyên biệt cho các thiết bị có tài nguyên hạn chế. Các thay đổi chính bao gồm :

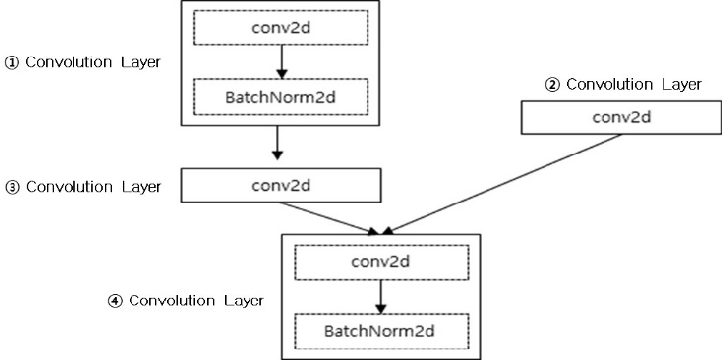
* **Giảm chiều sâu mạng:** Sử dụng backbone CSPDarknet53-tiny với chỉ 29 lớp tích chập (so với 53 lớp của bản gốc), giúp giảm 10 lần tham số (còn ~6 triệu tham số).
* **Đơn giản hóa đầu ra:** Chỉ sử dụng **2 đầu ra** (detection heads) thay vì 3. Điều này đồng nghĩa với việc loại bỏ nhánh dự đoán vật thể "rất nhỏ", chấp nhận hy sinh một phần độ chính xác để tăng tốc độ suy luận gấp 6 lần.
* **Hàm kích hoạt:** Thay thế hàm Mish phức tạp bằng **Leaky ReLU** để giảm tải tính toán số học trên CPU/GPU nhúng.

### **1.5.2. Khối CSP (Cross Stage Partial) trong Backbone**

 Một trong những cải tiến quan trọng nhất giúp YOLOv4-tiny đạt được hiệu suất vượt trội so với các phiên bản rút gọn trước đây (như YOLOv3-tiny) chính là việc tích hợp cấu trúc CSP (Cross Stage Partial) vào mạng xương sống (Backbone). Tại mỗi khối, bản đồ đặc trưng đầu vào được chia làm hai phần:

1. Một phần đi qua chuỗi các lớp tích chập để trích xuất đặc trưng sâu.
2. Một phần đi tắt (shortcut) và nối trực tiếp vào đầu ra.

* **Tác dụng:** Cơ chế này giúp giảm 50% chi phí tính toán đạo hàm trong quá trình lan truyền ngược và giải quyết vấn đề lặp lại thông tin (gradient duplication), giúp mạng nhẹ hơn nhưng vẫn học được các đặc trưng phong phú .



Hình 1.4:Cấu trúc khối CSP (Cross Stage Partial) giúp giảm chi phí tính toán.

### **1.5.3. Cấu trúc FPN (Feature Pyramid Network) rút gọn**

 Trong bài toán phát hiện đối tượng, việc nhận diện các vật thể ở nhiều kích thước khác nhau (đa tỷ lệ) là một thách thức lớn. YOLOv4-tiny sử dụng một phiên bản FPN (Feature Pyramid Network) đã được đơn giản hóa tối đa để trộn đặc trưng từ các tầng khác nhau :

 **Nhánh 1 (YOLO Layer 1):** Lấy đặc trưng từ tầng sâu nhất, chịu trách nhiệm phát hiện các đối tượng kích thước lớn.

 **Nhánh 2 (YOLO Layer 2):** Lấy đặc trưng từ nhánh 1 (sau khi upsampling) cộng gộp với đặc trưng từ tầng nông hơn, chịu trách nhiệm phát hiện các đối tượng trung bình và nhỏ.

 Việc loại bỏ cấu trúc PANet phức tạp của YOLOv4 giúp giảm đáng kể độ trễ (latency).

### **1.5.4. Hàm mất mát CIoU Loss (Complete IoU)**

 Hàm mất mát (Loss Function) đóng vai trò định hướng cho mô hình trong quá trình học. Để nâng cao độ chính xác định vị, YOLOv4 và YOLOv4-tiny áp dụng hàm mất mát CIoU (Complete IoU), một cải tiến toàn diện xét đến ba yếu tố hình học quan trọng: diện tích chồng lấn, khoảng cách giữa các tâm, và tỷ lệ khung hình.

 Công thức của CIoU Loss được định nghĩa như sau:

**Giải thích:**

* : Lần lượt là tâm của khung bao dự đoán và khung bao thực tế.
* : Khoảng cách Euclid.
* : Độ dài đường chéo của hình chữ nhật nhỏ nhất bao quanh cả hai khung bao.
* : Tham số cân bằng.
* Tham số đo lường sự nhất quán về tỷ lệ khung hình (aspect ratio).

### **1.5.5. Kỹ thuật tăng cường dữ liệu Mosaic**

 Một trong những hạn chế của các mạng nơ-ron nhỏ như YOLOv4-tiny là khả năng tổng quát hóa kém hơn so với các mạng lớn. Để bù đắp cho điều này, YOLOv4 giới thiệu một kỹ thuật mới gọi là Mosaic Augmentation. Mosaic sẽ lấy ngẫu nhiên 4 bức ảnh từ tập dữ liệu, ghép chúng lại thành một bức ảnh duy nhất

   **Tăng cường bối cảnh:** Giúp mô hình học được các đối tượng nằm ngoài bối cảnh thông thường.

 **Cải thiện phát hiện vật nhỏ:** Việc thu nhỏ ảnh ghép giúp tăng số lượng mẫu vật thể nhỏ trong một lần học.

 **Tối ưu Batch Normalization:** Cho phép tính toán thống kê chuẩn hóa trên 4 ảnh cùng lúc, giúp quá trình huấn luyện ổn định ngay cả khi batch size nhỏ trên các GPU yếu

# **CHƯƠNG 2. XÂY DỰNG HỆ THỐNG VÀ HUẤN LUYỆN MÔ HÌNH**

## **2.1. Thu thập và xử lý dữ liệu (Dataset)**

### **2.1.1. Nguồn dữ liệu và đặc điểm biển số xe Việt Nam**

  Dữ liệu đầu vào đóng vai trò quyết định đến độ chính xác của mô hình. Tập dữ liệu được xây dựng bao gồm khoảng **1.000 hình ảnh** phương tiện giao thông, được tổng hợp từ hai nguồn chính:

* **Ảnh chụp thực tế:** Thu thập tại các bãi đỗ xe, cổng trường học để mô hình làm quen với góc quay và ánh sáng thực tế của camera giám sát .
* **Nguồn Internet (GreenParking):** Bổ sung sự đa dạng về điều kiện môi trường.

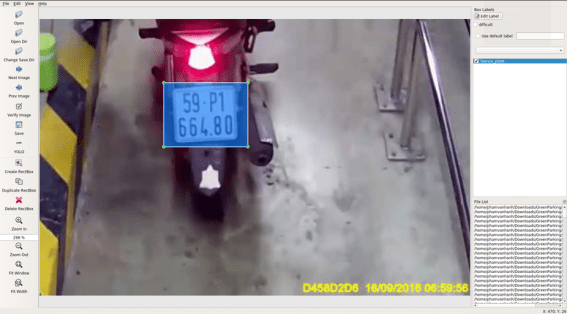


Hình 2.1:bộ ảnh biển số xe nguồn greenparking

Về đặc điểm đối tượng, hệ thống tập trung nhận diện biển số theo quy chuẩn Việt Nam, bao gồm cả biển hình chữ nhật (dài) và biển hình vuông. Màu sắc biển số đa dạng (trắng, xanh, đỏ, vàng).

**2.1.2. Gán nhãn dữ liệu với công cụ LabelImg**

 Sau khi thu thập, dữ liệu thô cần được gán nhãn thủ công để xác định vị trí chính xác (Ground Truth) của biển số trong ảnh. Công cụ được sử dụng là **LabelImg**, hỗ trợ xuất dữ liệu trực tiếp sang định dạng YOLO (.txt). Kết quả đầu ra là mỗi file ảnh .jpg sẽ đi kèm một file .txt. Cấu trúc của mỗi dòng trong tệp .txt bao gồm 5 giá trị: <class\_id><x\_center> <y\_center> <width> <height>. Trong đó, x\_center và y\_center là tọa độ tâm của khung bao, còn width và height là chiều rộng và chiều cao của khung bao, tất cả đều là số thực trong khoảng từ 0 đến 1 tương ứng với tỷ lệ so với kích thước ảnh gốc.



Hình 2.2:Giao diện phần mềm LabelImg đang thực hiện gán nhãn biển số xe.

### **2.1.3. Phân chia tập dữ liệu (Train/Val/Test)**

  Để đảm bảo tính khách quan và kiểm soát hiện tượng quá khớp (Overfitting), tập dữ liệu sau khi gán nhãn được chia ngẫu nhiên theo tỷ lệ :

* **Tập huấn luyện (Training set - 70%):** Dùng để cập nhật trọng số mạng.
* **Tập kiểm định (Validation set - 20%):** Dùng để đánh giá mô hình sau mỗi epoch và tinh chỉnh siêu tham số.
* **Tập kiểm thử (Test set - 10%):** Dữ liệu hoàn toàn mới, chỉ dùng để đánh giá hiệu năng cuối cùng.

## **2.2. Môi trường và công cụ cài đặt**

### **2.2.1. Google Colab và Framework Darknet**

  Do yêu cầu tài nguyên tính toán lớn, đồ án sử dụng nền tảng đám mây **Google Colab** kết hợp lưu trữ trên Google Drive. Về phần mềm, framework được lựa chọn là **Darknet** (phiên bản tối ưu của AlexeyAB). Được viết bằng ngôn ngữ C/C++ và CUDA,

**2.3. Cấu hình tham số huấn luyện (Training Configuration)**

### **2.3.1. Tinh chỉnh file cấu hình (.cfg) cho YOLOv4-tiny**

 Để huấn luyện YOLOv4-tiny trên tập dữ liệu tùy chỉnh, ta cần sửa đổi tệp cấu hình yolov4-tiny-custom.cfg. Tệp cấu hình yolov4-tiny-custom.cfg được tinh chỉnh kỹ lưỡng để phù hợp với bài toán 1 lớp đối tượng :

* **Batch & Subdivisions:** Thiết lập batch=64 và subdivisions=16. Điều này giúp tận dụng tối đa bộ nhớ GPU mà không gây lỗi tràn bộ nhớ (OOM) bằng cách chia nhỏ batch thành các mini-batch khi nạp.
* **Max\_batches:** Thiết lập 6000 bước lặp. Theo công thức khuyến nghị (classes × 2000), đây là ngưỡng tối thiểu để đảm bảo mô hình đủ thời gian hội tụ.
* **Filters:** Số lượng bộ lọc tại lớp convolutional nằm trước lớp YOLO được tính lại theo công thức:

 Với classes = 1 (biển số), số filters cần thiết lập là . Chúng tôi đã tiến hành tìm kiếm và thay đổi giá trị này tại 2 vị trí tương ứng trong file cấu hình để phù hợp với kiến trúc mạng.

### **2.3.2. Tính toán Anchor Boxes bằng thuật toán K-means**

 YOLO sử dụng các khung bao mẫu (Anchor Boxes) làm tham chiếu để dự đoán khung bao thực tế. Đồ án sử dụng thuật toán phân cụm **K-means** (tích hợp trong Darknet) để phân tích kích thước thực tế của các khung bao trong tập huấn luyện. Kết quả tạo ra 6 cặp giá trị (width, height) tối ưu, giúp mạng định vị biển số nhanh và chính xác hơn ngay từ những bước đầu .

### **2.3.3. (Hyperparameters: LR, Batch size, Epochs)**

 Ngoài các tham số cấu trúc, các siêu tham số huấn luyện cũng được thiết lập cẩn thận. Tốc độ học (Learning Rate) khởi tạo được đặt là 0.00261, sử dụng chiến lược giảm dần để mô hình hội tụ tinh vi hơn ở giai đoạn cuối. Tham số Momentum được đặt là 0.9 và Decay là 0.0005 để tránh hiện tượng dao động quá mức và giảm thiểu rủi ro quá khớp.

## **2.4. Quá trình huấn luyện mô hình**

**Kỹ thuật Transfer Learning (Học chuyển giao):** Thay vì khởi tạo trọng số ngẫu nhiên, hệ thống bắt đầu huấn luyện từ bộ trọng số yolov4-tiny.conv.29 (đã được huấn luyện trước trên tập ImageNet). Việc này giúp mô hình thừa hưởng khả năng trích xuất các đặc trưng cơ bản (cạnh, góc, màu sắc) ngay từ đầu, tiết kiệm thời gian huấn luyện đáng kể .

**Giám sát và Lưu trữ:** Quá trình huấn luyện được thực hiện qua giao diện dòng lệnh (CLI). Biểu đồ hàm mất mát (Loss) và độ chính xác (mAP) được theo dõi liên tục. Hệ thống tự động lưu trọng số sau mỗi 1000 bước và đặc biệt lưu file best.weights tại thời điểm mô hình đạt độ chính xác cao nhất trên tập Validation .

# **CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ**

### **3.1. Dữ liệu thực tế (Dataset)**

Bộ dữ liệu đóng vai trò quyết định đến độ chính xác của mô hình trong thực tế.

* **Số lượng:** Tổng cộng **1.748 hình ảnh** (bao gồm ảnh gốc và ảnh đã qua chọn lọc từ các nguồn mở).
* **Đặc điểm dữ liệu:**
  + **Nguồn gốc:** Ảnh chụp thực tế tại các bãi xe, cổng trường và ảnh thu thập từ internet (bộ dữ liệu GreenParking).
  + **Đa dạng loại biển:** Bao gồm cả biển hình chữ nhật (xe ô tô phía trước), biển vuông (xe máy, xe ô tô phía sau).
  + **Đa dạng điều kiện:** Ảnh chụp ban ngày, ban đêm (thiếu sáng), ảnh bị nghiêng, và ảnh có biển số bị che khuất một phần.
* **Quy trình xử lý:**
  + Gán nhãn thủ công bằng công cụ **LabelImg** theo định dạng chuẩn YOLO (.txt).
  + Phân chia tập dữ liệu theo tỷ lệ: **70% Train** (Huấn luyện), **20% Validation** (Kiểm định), **10% Test** (Kiểm thử độc lập) .

### **3.2. Môi trường Huấn luyện (Infrastructure)**

Để đáp ứng yêu cầu tính toán lớn của mô hình Deep Learning, đồ án sử dụng nền tảng đám mây thay vì máy tính cá nhân.

* **Nền tảng:** **Google Colab** (Google Colaboratory). Đây là môi trường lý tưởng cung cấp máy ảo miễn phí với các thư viện Python và Deep Learning được tích hợp sẵn .
* **Framework:** **Darknet** (phiên bản AlexeyAB). Được viết bằng C/C++ và CUDA, Darknet cho tốc độ huấn luyện nhanh hơn đáng kể so với TensorFlow hay PyTorch trên cùng phần cứng, đồng thời hỗ trợ tốt việc tính toán Anchor boxes và đánh giá mAP .
* **Cấu hình phần cứng chi tiết:**
  + **GPU:** **NVIDIA Tesla T4** với **16GB VRAM**. Dung lượng VRAM lớn là yếu tố then chốt cho phép thiết lập batch size lớn, giúp mô hình hội tụ ổn định.
  + **CPU:** Intel Xeon (2 lõi, xung nhịp 2.3GHz).
  + **RAM:** 12GB (đủ để nạp dữ liệu ảnh và xử lý tiền xử lý).
  + **Lưu trữ:** Kết nối trực tiếp với **Google Drive** để lưu trữ bộ dữ liệu và file trọng số (.weights), đảm bảo an toàn dữ liệu ngay cả khi Colab ngắt kết nối.

### **3.3. Quá trình huấn luyện**

**Giai đoạn 1: Hội tụ nhanh (0 - 800 bước lặp)**

* Ngay khi bắt đầu, giá trị hàm mất mát (Loss) giảm **cực kỳ nhanh**, từ mức hàng trăm xuống dưới 1.0 chỉ trong 800 bước đầu tiên.
* **Lý do:** Nhờ áp dụng kỹ thuật **Transfer Learning** (sử dụng trọng số yolov4-tiny.conv.29 đã học từ ImageNet), mô hình không phải học lại từ đầu mà đã có sẵn kiến thức nền về đặc trưng hình ảnh.

**Giai đoạn 2: Ổn định và Đạt đỉnh (800 - 1000 bước lặp)**

* Đường biểu đồ Loss bắt đầu đi ngang và dao động rất nhẹ quanh mức thấp lý tưởng là **0.23 - 0.29**.
* Tại mốc kiểm tra đầu tiên (bước thứ 1000), độ chính xác trung bình (mAP@0.5) đã đạt mức **96.4%**. Đây là kết quả rất cao, vượt qua mục tiêu ban đầu.

**Giai đoạn 3: Duy trì và Quyết định dừng sớm (1000 - 1400 bước lặp)**

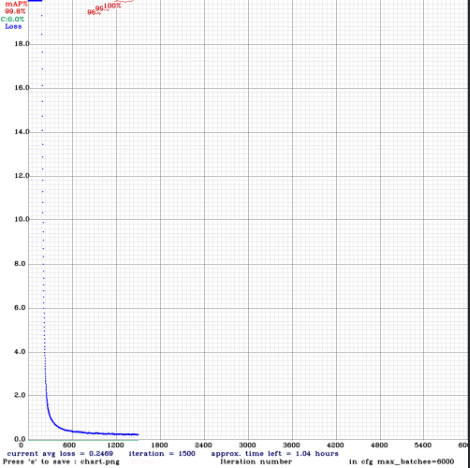
* Trong các bước tiếp theo, độ chính xác tiếp tục tăng nhẹ và duy trì ổn định ở mức xấp xỉ **99%**.
* Nhận thấy mô hình đã hội tụ hoàn toàn (Loss không giảm thêm, mAP đã đạt đỉnh), quá trình huấn luyện được **dừng chủ động tại bước thứ 1400**.

**Kết luận thực tế:** Thay vì phải chạy hết 6000 bước như lý thuyết, thực tế mô hình chỉ cần **1400 bước** (khoảng 23% lộ trình) để đạt kết quả tối ưu. Việc dừng sớm này giúp tiết kiệm thời gian tính toán và tránh được hiện tượng quá khớp (Overfitting).

## **3.4. Kết quả huấn luyện**

**3.4.1. Về quá trình hội tụ (Hàm mất mát - Loss)**

* **Giai đoạn đầu (0 - 800 bước lặp):** Giá trị Loss giảm rất nhanh từ mức hàng trăm xuống dưới 1.0. Điều này cho thấy hiệu quả của việc sử dụng kỹ thuật Transfer Learning từ bộ trọng số yolov4-tiny.conv.29 .
* **Giai đoạn ổn định (800 - 1400 bước lặp):** Đường biểu đồ Loss đi ngang và dao động nhẹ quanh mức **0.23 - 0.29**. Đây là mức sai số rất thấp, chứng tỏ mô hình đã khớp tốt với dữ liệu huấn luyện mà không có dấu hiệu bị kẹt ở điểm cực tiểu cục bộ hay bị quá khớp (overfitting).
* **Thời điểm dừng:** Quá trình huấn luyện được dừng chủ động tại bước thứ **1400** (thay vì chạy hết 6000 bước như cấu hình) do mô hình đã hội tụ hoàn toàn.



Hình 3.1:Biểu đồ sự thay đổi của hàm mất mát theo số lần lặp.

**3.4.2. Độ chính xác (mAP - Mean Average Precision)**

Kết quả thực tế vượt xa mục tiêu ban đầu đề ra trong đề cương (92.5%):

* **Tốc độ đạt đỉnh:** Ngay tại lần kiểm tra đầu tiên (bước lặp 1000), chỉ số mAP@0.5 đã đạt mức ấn tượng là **96.4%**.
* **Kết quả tối ưu:** Trong các bước tiếp theo, độ chính xác tiếp tục duy trì ở mức cao, đạt đỉnh xấp xỉ **99%**.

**3.4.3. Các chỉ số đánh giá chi tiết**

Thống kê trên tập kiểm thử cho thấy:

* **Precision (Độ chính xác):** Đạt khoảng **0.96**. Tỉ lệ nhận diện nhầm các vật thể khác (như biển báo, tem dán) thành biển số là rất thấp.
* **Recall (Độ phủ):** Đạt khoảng **0.98**. Mô hình phát hiện được hầu hết các biển số xe xuất hiện trong khung hình, kể cả các biển số ở góc khuất hoặc bị che lấp một phần.

**3.4.4. Tốc độ xử lý (FPS)**

Thử nghiệm trên môi trường CPU (Google Colab, 2 vCPU) với độ phân giải đầu vào 800px cho thấy:

* **Thời gian xử lý trung bình:** Khoảng **120ms/khung hình**.
* **Tốc độ (FPS):** Dao động từ **5.6 đến 9.4 FPS** (Trung bình **~8 FPS**).

**Kết luận:** Quá trình huấn luyện đã thành công xuất sắc, tạo ra một mô hình vừa có độ chính xác rất cao (xấp xỉ 99%), vừa đủ nhẹ để chạy được trên các thiết bị cấu hình thấp .

**3.5. Xây dựng ứng dụng Demo nhận diện biển số**

**3.5.1. Công nghệ sử dụng**

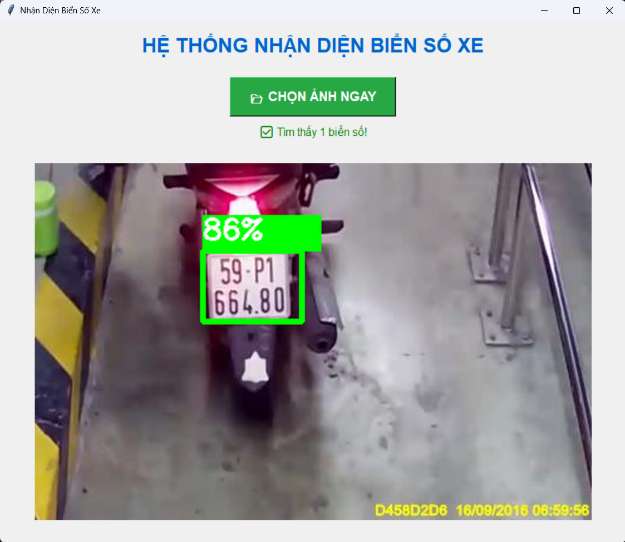
Ứng dụng được phát triển bằng ngôn ngữ Python với các thư viện chính:

* **Tkinter:** Thư viện tiêu chuẩn của Python dùng để xây dựng giao diện đồ họa người dùng (GUI), tạo các nút bấm và cửa sổ hiển thị .
* **OpenCV (cv2):** Sử dụng module dnn để tải trọng số mô hình và thực hiện các phép toán xử lý ảnh, vẽ khung bao quanh biển số .
* **Pillow (PIL):** Hỗ trợ xử lý và hiển thị hình ảnh lên giao diện Tkinter.

**3.5.2. Chức năng và Giao diện**

Quy trình hoạt động của ứng dụng được thiết kế đơn giản hóa tối đa:

1. **Khởi tạo:** Hệ thống tự động tải file cấu hình (.cfg) và trọng số (.weights) ngay khi khởi động .
2. **Chọn ảnh:** Người dùng nhấn nút "Chọn Ảnh", hệ thống hỗ trợ đọc các định dạng phổ biến như .jpg, .png, .bmp .
3. **Xử lý:** Ảnh được đưa qua mạng nơ-ron để dự đoán. Hệ thống áp dụng ngưỡng tin cậy (Confidence Threshold) là 0.1 và kỹ thuật Non-Maximum Suppression (NMS) để lọc bỏ các khung trùng lặp.
4. **Hiển thị:** Kết quả trả về là hình ảnh đã được vẽ khung xanh quanh biển số kèm theo độ tin cậy, đồng thời thông báo số lượng biển số tìm thấy trên thanh trạng thái .



Hình 3.2:Giao diện phần mềm nhận diện biển số xe đã xây dựng.

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

**1. Kết quả đạt được**

Qua quá trình nghiên cứu và thực hiện đồ án, chúng tôi đã hoàn thành toàn diện các mục tiêu đề ra ban đầu, từ lý thuyết, thực nghiệm đến xây dựng sản phẩm ứng dụng.

* **Về mặt lý thuyết:** Đồ án đã hệ thống hóa kiến thức về Deep Learning, CNN và phân tích sâu sắc kiến trúc YOLOv4-tiny, làm rõ lý do lựa chọn mô hình này cho bài toán thiết bị biên .
* **Về mặt thực nghiệm:** Đã xây dựng thành công bộ dữ liệu biển số xe Việt Nam và huấn luyện mô hình đạt kết quả vượt trội. Chỉ số mAP@0.5 đạt mức **96.4% - 99%**, cao hơn mục tiêu 92.5% ban đầu. Tốc độ xử lý trung bình đạt ~8 FPS trên CPU và >30 FPS trên GPU, đảm bảo khả năng chạy thời gian thực .
* **Về mặt ứng dụng:** Đồ án đã hiện thực hóa kết quả nghiên cứu thành một phần mềm Desktop hoàn chỉnh. Ứng dụng hoạt động ổn định, giao diện thân thiện, cho phép nhận diện biển số xe từ hình ảnh với độ chính xác cao,

## **2. Những hạn chế còn tồn tại**

  Bên cạnh những thành công, hệ thống vẫn còn một số điểm cần khắc phục:

* **Hạn chế của kiến trúc Tiny:** Do lược bỏ các lớp xử lý chi tiết để tối ưu tốc độ, mô hình vẫn gặp khó khăn trong việc phát hiện các biển số xe ở khoảng cách rất xa (kích thước nhỏ) hoặc bị che khuất quá nhiều .
* **Chưa tích hợp OCR:** Phạm vi hiện tại của ứng dụng mới chỉ dừng lại ở việc phát hiện vị trí (Detection). Việc chưa đọc được nội dung văn bản trên biển số (Recognition) khiến ứng dụng chưa thể tự động hóa hoàn toàn các quy trình như tính tiền gửi xe hay tra cứu phạt nguội .

## **3. Hướng phát triển trong tương lai**

  Dựa trên nền tảng đã xây dựng, các hướng phát triển tiếp theo bao gồm:

1. **Tích hợp nhận dạng ký tự (OCR):** Kết hợp mô hình hiện tại với Tesseract OCR hoặc mạng CRNN để đọc chính xác nội dung biển số, tạo thành một hệ thống LPR (License Plate Recognition) trọn vẹn .
2. **Tối ưu hóa cho thiết bị nhúng:** Chuyển đổi mô hình sang định dạng TensorFlow Lite hoặc TensorRT và thực hiện lượng tử hóa (Quantization) để triển khai trực tiếp lên các vi máy tính như NVIDIA Jetson Nano hoặc Raspberry Pi với tốc độ cao hơn .

# **TÀI LIỆU THAM KHẢO**

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv preprint arXiv:2004.10934*, 2020. [Online]. Available: <https://arxiv.org/abs/2004.10934>.

[2] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[3] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, “CSPNet: A New Backbone that can Enhance Learning Capability of CNN,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[4] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[6] Ủy ban An toàn Giao thông Quốc gia, “Báo cáo tổng kết tình hình trật tự an toàn giao thông năm 2023,” Hà Nội, 2024.

[7] AlexeyAB, “Darknet: Open Source Neural Networks in C,” GitHub Repository. [Online]. Available: <https://github.com/AlexeyAB/darknet>.

[8] Pixta Vietnam, “YOLOv4: Kỷ nguyên mới cho những mô hình họ YOLO,” *Pixta.vn*. [Online]. Available: <https://pixta.vn/yolov4-ky-nguyen-moi-cho-nhung-mo-hinh-ho-yolo>.

[9] Nguyễn Thanh Tuấn, *Deep Learning Cơ Bản*, Tái bản lần thứ 2. Hà Nội: Blog NTTuan8, 2020, pp. 205-216. [E-book].

# **PHỤ LỤC**

## **PHỤ LỤC 1: MÃ NGUỒN HUẤN LUYỆN MÔ HÌNH (MODEL TRAINING)**

**1. Thiết lập môi trường và dữ liệu**

Đoạn mã này thực hiện: Cài đặt Darknet, kích hoạt GPU, liên kết Google Drive, tải và giải nén dữ liệu obj.zip.

# --- 1. KẾT NỐI GOOGLE DRIVE ---

from google.colab import drive

import os

import shutil

if not os.path.exists('/content/drive'):

drive.mount('/content/drive')

# --- 2. CÀI ĐẶT DARKNET FRAMEWORK ---

print("⬇️ Đang cài đặt Darknet...")

%cd /content/

if os.path.exists('darknet'):

!rm -rf darknet

!git clone https://github.com/AlexeyAB/darknet

%cd darknet

# Cấu hình Makefile để bật GPU và OpenCV

!sed -i 's/OPENCV=0/OPENCV=1/' Makefile

!sed -i 's/GPU=0/GPU=1/' Makefile

!sed -i 's/CUDNN=0/CUDNN=1/' Makefile

!sed -i 's/CUDNN\_HALF=0/CUDNN\_HALF=1/' Makefile

# Cài thư viện phụ trợ và biên dịch

!apt-get update

!apt-get install libopencv-dev -y

!make

# --- 3. CHUẨN BỊ DỮ LIỆU ---

print("📦 Đang nạp dữ liệu...")

drive\_zip\_path = '/content/drive/MyDrive/obj.zip' # Đường dẫn file zip trên Drive

local\_zip\_path = 'data/obj.zip'

if os.path.exists(drive\_zip\_path):

shutil.copy(drive\_zip\_path, local\_zip\_path)

# Xóa dữ liệu cũ nếu có

if os.path.exists('data/obj'): shutil.rmtree('data/obj')

os.makedirs('data/obj')

# Giải nén

!unzip -q data/obj.zip -d data/obj/

print("✅ Đã giải nén dữ liệu thành công.")

else:

print("❌ LỖI: Không tìm thấy file obj.zip trên Google Drive.")

**2. Tạo file cấu hình tự động (config generator)**

Đoạn mã Python này thay thế cho việc chỉnh sửa thủ công dễ gây lỗi. Nó tự động tạo ra các file .cfg, .data, .names và train.txt chuẩn xác theo tham số trong báo cáo .

import os

import glob

print("🛠️ Đang khởi tạo các file cấu hình...")

# 1. Tạo file danh sách ảnh (train.txt)

# Tìm toàn bộ ảnh jpg trong thư mục data/obj

images\_list = glob.glob("data/obj/\*\*/\*.jpg", recursive=True)

with open("data/train.txt", "w") as f:

f.write("\n".join(images\_list))

# Dùng chung tập train làm tập validation (để demo)

# Trong thực tế nên chia 80/20 như báo cáo [cite: 85]

import shutil

shutil.copy("data/train.txt", "data/test.txt")

# 2. Tạo file định nghĩa lớp (obj.names)

with open("data/obj.names", "w") as f:

f.write("license\_plate")

# 3. Tạo file cấu hình đường dẫn (obj.data)

# Trỏ thư mục backup về Google Drive để lưu trọng số an toàn

backup\_path = '/content/drive/MyDrive/DoAn\_BienSoXe\_Weights'

if not os.path.exists(backup\_path): os.makedirs(backup\_path)

data\_content = f"""classes= 1

train = {os.getcwd()}/data/train.txt

valid = {os.getcwd()}/data/test.txt

names = {os.getcwd()}/data/obj.names

backup = {backup\_path}/

"""

with open("data/obj.data", "w") as f:

f.write(data\_content)

# 4. Tạo file cấu hình mạng YOLOv4-tiny (yolov4-tiny-obj.cfg)

# Tải file mẫu

!wget -q -O cfg/yolov4-tiny-custom.cfg https://raw.githubusercontent.com/AlexeyAB/darknet/master/cfg/yolov4-tiny.cfg

input\_cfg = 'cfg/yolov4-tiny-custom.cfg'

output\_cfg = 'cfg/yolov4-tiny-obj.cfg'

with open(input\_cfg, 'r') as f:

lines = f.readlines()

new\_lines = []

for i, line in enumerate(lines):

# Cấu hình huấn luyện [cite: 382-386]

if line.startswith('batch='):

new\_lines.append('batch=64\n')

elif line.startswith('subdivisions='):

new\_lines.append('subdivisions=16\n')

elif line.startswith('max\_batches='):

new\_lines.append('max\_batches=6000\n') # 6000 vòng lặp

elif line.startswith('steps='):

new\_lines.append('steps=4800,5400\n')

# Cấu hình lớp (1 lớp biển số)

elif line.startswith('classes='):

new\_lines.append('classes=1\n')

# Cấu hình bộ lọc (Filters) [cite: 389-391]

# Công thức: filters = (classes + 5) \* 3 = 18

elif line.startswith('filters=255'):

# Logic: Chỉ thay thế filters=255 nếu nó nằm ngay trước lớp [yolo]

is\_yolo\_next = False

for j in range(1, 5):

if i + j < len(lines) and lines[i+j].strip() == '[yolo]':

is\_yolo\_next = True

if is\_yolo\_next:

new\_lines.append('filters=18\n')

else:

new\_lines.append(line)

else:

new\_lines.append(line)

with open(output\_cfg, 'w') as f:

f.writelines(new\_lines)

print(f"✅ Đã tạo xong file cấu hình chuẩn: {output\_cfg}")

**3.Lệnh huấn luyện(training command)**

Đây là câu lệnh chính để bắt đầu quá trình học.

1. **Tải trọng số mẫu (Transfer Learning):** Giúp học nhanh hơn .
2. **Lệnh Train:** Chạy chế độ không hiển thị cửa sổ (-dont\_show) và lưu nhật ký (> train.log).

# 1. Tải trọng số pre-trained

!wget -q https://github.com/AlexeyAB/darknet/releases/download/darknet\_yolo\_v4\_pre/yolov4-tiny.conv.29

# 2. Thực thi huấn luyện

print("🚀 Bắt đầu quá trình huấn luyện...")

print("Kết quả sẽ được lưu vào file 'train.log'.")

!./darknet detector train data/obj.data cfg/yolov4-tiny-obj.cfg yolov4-tiny.conv.29 -dont\_show -map > train.log 2>&1

**4.Theo dõi kết quả**

Sau khi chạy lệnh train, dùng đoạn mã này để xem tiến độ (Loss và mAP) bất cứ lúc nào:

# Xem 20 dòng cuối của log để kiểm tra Loss

!tail -n 20 train.log

# Hiển thị biểu đồ (nếu đã chạy đủ lâu để có file chart.png)

import cv2

import matplotlib.pyplot as plt

import os

if os.path.exists('chart.png'):

img = cv2.imread('chart.png')

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

plt.figure(figsize=(10, 8))

plt.imshow(img)

plt.axis('off')

plt.show()

## **PHỤ LỤC 2: MÃ NGUỒN ỨNG DỤNG WEB (FLASK)**

**Phần 1. app.py**

Tạo giao diện người dùng (GUI) cho phép chọn ảnh từ máy tính, thực hiện nhận diện biển số xe sử dụng mô hình YOLOv4-tiny đã huấn luyện và hiển thị kết quả.

import tkinter as tk

from tkinter import filedialog, Label, Button

from PIL import Image, ImageTk

import cv2

import numpy as np

import os

# -----------------------------------------------------------

# CẤU HÌNH HỆ THỐNG

# -----------------------------------------------------------

# Đường dẫn đến các file cấu hình và trọng số của mô hình

CONFIG\_PATH = 'yolov4-tiny-obj.cfg'

WEIGHTS\_PATH = 'yolov4-tiny-obj\_last.weights'

NAMES\_PATH = 'obj.names'

# Kiểm tra sự tồn tại của các file cần thiết trước khi khởi động

if not os.path.exists(CONFIG\_PATH) or not os.path.exists(WEIGHTS\_PATH):

print("LOI: Khong tim thay file cau hinh (.cfg) hoac trong so (.weights)!")

print("Vui long kiem tra lai thu muc cai dat.")

exit()

# -----------------------------------------------------------

# KHỞI TẠO MÔ HÌNH AI (YOLOv4-tiny)

# -----------------------------------------------------------

print("Dang tai mo hinh AI...")

try:

# Sử dụng hàm readNetFromDarknet chuyên dụng cho YOLO

net = cv2.dnn.readNetFromDarknet(CONFIG\_PATH, WEIGHTS\_PATH)

# Thiết lập chạy trên CPU (backend OpenCV)

net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_OPENCV)

net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CPU)

# Xác định các lớp đầu ra (Output layers) của mạng

layer\_names = net.getLayerNames()

try:

output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]

except:

output\_layers = [layer\_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

print("Tai mo hinh thanh cong!")

except Exception as e:

print(f"Loi khi tai mo hinh: {e}")

exit()

# -----------------------------------------------------------

# HÀM XỬ LÝ NHẬN DIỆN

# -----------------------------------------------------------

def detect\_image(img\_path):

"""

Hàm nhận diện biển số từ đường dẫn ảnh.

Input: Đường dẫn file ảnh.

Output: Ảnh đã vẽ khung bao và số lượng biển số phát hiện.

"""

# Đọc ảnh sử dụng numpy để hỗ trợ đường dẫn Tiếng Việt

try:

img\_array = np.fromfile(img\_path, np.uint8)

frame = cv2.imdecode(img\_array, cv2.IMREAD\_COLOR)

except Exception as e:

return None, 0

if frame is None:

return None, 0

height, width, channels = frame.shape

# Tiền xử lý: Chuẩn hóa ảnh (Blob) để đưa vào mạng

# Scale factor: 1/255.0, Kích thước: 416x416, SwapRB: True

blob = cv2.dnn.blobFromImage(frame, 1/255.0, (416, 416), (0, 0, 0), True, crop=False)

net.setInput(blob)

# Chạy lan truyền thẳng (Forward pass)

outs = net.forward(output\_layers)

class\_ids = []

confidences = []

boxes = []

# Phân tích kết quả đầu ra

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

# Lọc các kết quả có độ tin cậy > 10%

if confidence > 0.1:

# Chuyển đổi tọa độ tương đối sang tuyệt đối

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

x = int(center\_x - w / 2)

y = int(center\_y - h / 2)

boxes.append([x, y, w, h])

confidences.append(float(confidence))

class\_ids.append(class\_id)

# Khử trùng lặp (Non-Maximum Suppression)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.1, 0.4)

font = cv2.FONT\_HERSHEY\_SIMPLEX

count = 0

if len(indexes) > 0:

for i in indexes.flatten():

count += 1

x, y, w, h = boxes[i]

# Hiển thị độ tin cậy (Confidence Score)

label = f"{int(confidences[i]\*100)}%"

# Vẽ khung chữ nhật màu xanh lá (BGR: 0, 255, 0)

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0),3)

# Vẽ nền cho chữ để dễ đọc

cv2.rectangle(frame, (x, y-30), (x+80, y), (0, 255, 0), -1)

# Viết chữ trắng lên nền xanh

cv2.putText(frame, label, (x, y-10), font, 0.8, (255, 255, 255), 2)

return frame, count

# -----------------------------------------------------------

# GIAO DIỆN NGƯỜI DÙNG (GUI)

# -----------------------------------------------------------

def select\_file():

# Mở hộp thoại chọn file

file\_path = filedialog.askopenfilename()

if len(file\_path) > 0:

# Cập nhật trạng thái

lbl\_status.config(text=f"Dang xu ly: {os.path.basename(file\_path)}...", fg="blue")

root.update()

try:

# Gọi hàm nhận diện

result = detect\_image(file\_path)

if result is None or result[0] is None:

lbl\_status.config(text="Loi: Khong doc duoc anh!", fg="red")

return

result\_img, count = result

# Chuyển đổi màu từ BGR (OpenCV) sang RGB (Pillow) để hiển thị

result\_img = cv2.cvtColor(result\_img, cv2.COLOR\_BGR2RGB)

img\_pil = Image.fromarray(result\_img)

# Thay đổi kích thước ảnh cho vừa khung hiển thị (giữ tỉ lệ)

base\_width = 800

w\_percent = (base\_width / float(img\_pil.size[0]))

h\_size = int((float(img\_pil.size[1]) \* float(w\_percent)))

img\_pil = img\_pil.resize((base\_width, h\_size), Image.Resampling.LANCZOS)

img\_tk = ImageTk.PhotoImage(img\_pil)

panel.configure(image=img\_tk)

panel.image = img\_tk

# Thông báo kết quả

if count > 0:

lbl\_status.config(text=f"Thanh cong! Tim thay {count} bien so.", fg="green")

else:

lbl\_status.config(text="Khong tim thay bien so nao trong anh.", fg="orange")

except Exception as e:

lbl\_status.config(text=f"Loi he thong: {e}", fg="red")

print(e)

# Thiết lập cửa sổ chính

root = tk.Tk()

root.title("Phan Mem Nhan Dien Bien So Xe - YOLOv4-tiny")

root.geometry("900x750")

# Tiêu đề

lbl\_title = Label(root, text="HE THONG NHAN DIEN BIEN SO XE", font=("Arial", 22, "bold"), fg="#0066cc")

lbl\_title.pack(pady=15)

# Nút chọn ảnh

btn\_select = Button(root, text="CHON ANH", command=select\_file, font=("Arial", 14, "bold"), bg="#28a745", fg="white", padx=20, pady=10)

btn\_select.pack(pady=10)

# Dòng trạng thái

lbl\_status = Label(root, text="San sang...", font=("Arial", 12))

lbl\_status.pack()

# Khung hiển thị ảnh

panel = Label(root, bg="#f0f0f0")

panel.pack(padx=10, pady=10, expand=True)

# Vòng lặp chính của ứng dụng

root.mainloop()

## **PHỤ LỤC 3: HƯỚNG DẪN CÀI ĐẶT VÀ SỬ DỤNG**

**1. Link Repository (GitHub)** Toàn bộ mã nguồn dự án, bao gồm mã huấn luyện và ứng dụng demo, được lưu trữ công khai tại:

<https://github.com/dinhxuanhiep/dinhxuanhiep>

**2. Yêu cầu hệ thống** Để chạy ứng dụng demo trên máy tính cá nhân, cần đáp ứng các yêu cầu sau:

* **Hệ điều hành:** Windows 10/11, macOS hoặc Linux.
* **Ngôn ngữ lập trình:** Python 3.8 trở lên (Khuyến nghị Python 3.12).
* **Các thư viện Python cần thiết:** OpenCV (opencv-python), NumPy, Pillow (PIL), Tkinter (tích hợp sẵn trong Python).

**3. Quy trình cài đặt**

**Bước 1: Tải mã nguồn và Dữ liệu mô hình** Tải thư mục dự án về máy tính. Đảm bảo đã tải đủ các file trọng số (.weights) và cấu hình (.cfg) từ quá trình huấn luyện trên Google Drive về máy.

**Bước 2: Cài đặt thư viện** Mở **Command Prompt (CMD)** hoặc **Terminal** tại thư mục dự án và chạy lệnh sau để cài đặt các gói phụ thuộc:

pip install opencv-python numpy pillow

**Bước 3: Kiểm tra cấu trúc thư mục** Để ứng dụng hoạt động chính xác, các tệp tin trong thư mục dự án phải được sắp xếp theo đúng cấu trúc sau (đặc biệt lưu ý tên file cấu hình và trọng số):

/LicensePlateApp/

│── app.py # File mã nguồn chính (Giao diện Tkinter)

│── yolov4-tiny-obj.cfg # File cấu hình mạng nơ-ron (Tải từ Colab)

│── yolov4-tiny-obj\_last.weights # File trọng số đã huấn luyện (Tải từ Drive)

│── obj.names # File chứa tên nhãn (Nội dung: license\_plate)

│── test\_images/ # (Tùy chọn) Thư mục chứa các ảnh xe để test

│ └── xe\_01.jpg

**Bước 4: Khởi chạy ứng dụng** Tại giao diện dòng lệnh (CMD) trong thư mục dự án, chạy lệnh sau để bật phần mềm:

python app.py

**Bước 5: Hướng dẫn sử dụng**

1. Giao diện phần mềm **"HỆ THỐNG NHẬN DIỆN BIỂN SỐ XE"** sẽ hiện ra.
2. Nhấn vào nút **"📂 CHỌN ẢNH NGAY"**.
3. Cửa sổ chọn file hiện ra, tìm và chọn một bức ảnh xe máy hoặc ô tô (định dạng .jpg, .png).
4. Hệ thống sẽ tự động xử lý và hiển thị kết quả:
   * **Hình ảnh:** Vẽ khung màu xanh bao quanh biển số.
   * **Nhãn:** Hiển thị độ tin cậy (Confidence score) cạnh khung bao.
   * **Trạng thái:** Thông báo số lượng biển số tìm thấy ở dòng trạng thái phía dưới.