TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

ỨNG DỤNG HỌC SÂU TRONG NHẬN DIỆN HÀNH VI KHÔNG ĐỘI MŨ BẢO HIỂM KHI THAM GIA GIAO THÔNG

Giảng viên hướng dẫn : TS. Nguyễn Quốc Tuấn
Sinh viên thực hiện : Nguyễn Văn Nghĩa
Lớp : CNTT3 – K62
Mã sinh viên : 211213161

Hà Nội – 2025

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

ỨNG DỤNG HỌC SÂU TRONG NHẬN DIỆN HÀNH VI KHÔNG ĐỘI MŨ BẢO HIỂM KHI THAM GIA GIAO THÔNG

Giảng viên hướng dẫn : TS. Nguyễn Quốc Tuấn
Sinh viên thực hiện : Nguyễn Văn Nghĩa
Lớp : CNTT3 – K62
Mã sinh viên : 211213161

Hà Nội – 2025

LỜI CẢM ƠN

Lời đầu tiên em xin gửi lời cảm ơn chân thành nhất đến toàn thể các thầy cô giáo trong khoa Công nghệ thông tin – Trường Đại học Giao thông vận tải. Các thầy cô không chỉ truyền đạt kiến thức mà còn chia sẻ những kinh nghiệm quý báu, giúp em có cơ hội được học tập, trau dồi, nâng cao năng lực. Đó là những hành trang quý báu giúp em vững bước, tự tin hơn trên con đường sự nghiệp sau này.

Bên cạnh đó, em xin gửi lời cảm ơn tới các bạn sinh viên trong lớp, trong khoa - những người đã luôn đồng hành, chia sẻ và hỗ trợ em trong suốt những năm tháng học tập. Nhờ có sự đồng hành ấy, em đã có thể vượt qua những khó khăn, thử thách để hoàn thành đề tài này.

Em cũng xin bày tỏ lòng biết ơn sâu sắc tới gia đình - chỗ dựa tinh thần vững chắc nhất trong cuộc sống. Sự yêu thương, động viên và ủng hộ vô điều kiện từ gia đình chính là nguồn động lực to lớn giúp em vững bước, kiên trì theo đuổi con đường học tập và trưởng thành.

Đặc biệt, em xin trân trọng cảm ơn thầy Nguyễn Quốc Tuấn – người đã tận tình hướng dẫn, hỗ trợ và đồng hành cùng em trong suốt quá trình thực hiện đề tài tốt nghiệp. Những góp ý, nhận xét và định hướng của thầy đã giúp em hoàn thiện đề tài một cách tốt nhất có thể.

Mặc dù đã rất nỗ lực, song trong quá trình thực hiện đề tài, em không thể tránh khỏi những thiếu sót và hạn chế. Vì vậy, em rất mong nhận được sự thông cảm và những góp ý chân thành từ quý thầy cô và các bạn để đề tài có thể được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hà Nội, ngày 30 tháng 05 năm 2025

Sinh viên thực hiện

Nguyễn Văn Nghĩa

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT 1
DANH MỤC BẢNG BIỂU 2
DANH MỤC HÌNH ẢNH 3
MỞ ĐẦU 4
CHƯƠNG 1: TỔNG QUAN VỀ CÔNG NGHỆ 5
1.1. Giới thiệu 5
1.2. Học sâu 5
1.2.1. Khái niệm 5
1.2.2. Mạng nơ-ron 6
1.2.3. Ứng dụng của học sâu 7
1.3. Mạng nơ-ron tích chập 7
1.3.1. Khái niệm 7
1.3.2. Cấu trúc 7
1.4. Thị giác máy tính 9
1.4.1. Thị giác máy tính và các bài toán liên quan 9
1.4.2. Bài toán Object Detection 10
1.4.3. Các mô hình của bài toán Object Detection 11
1.4.4. Các kĩ thuật xử lý dữ liệu cho bài toán Object Detection 12
1.5. Các công nghệ sử dụng 12
1.5.1. Python 13
1.5.2. Google Colab 13
1.5.3. Visual Studio Code 14
1.5.4. Numpy 14
1.5.5. OpenCV 15
1.5.6. PIL 15
1.5.7. Streamlit 16
CHƯƠNG 2: GIỚI THIỆU VỀ YOLO 17
2.1. Tổng quan về YOLO 17
2.1.1. Khái niệm 17
2.1.2. Lịch sử phát triển 17
2.1.3. So sánh các phiên bản YOLO 18
2.1.4. Kiến trúc của mô hình YOLO 19
2.2. Kiến trúc tổng thể của mô hình YOLOv11 25
2.2.1. Giới thiệu về YOLOv11 25
2.2.2. Kiến trúc tổng thể 25
2.2.3. Các tính năng chính của YOLOv11 33
2.2.4. So sánh các mô hình YOLOv11 33
2.2.5. Ứng dụng của YOLOv11 34
CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG 35

3.1. Giới thiệu bài toán 35
3.1.1. Đặt vấn đề 35
3.1.2. Mục tiêu 35
3.1.3. Phạm vi đề tài 36
3.2. Thu thập và xử lý dữ liệu 36
3.2.1. Nguồn dữ liệu 36
3.2.2. Gán nhãn dữ liệu 36
3.2.3. Tiền xử lý dữ liệu 37
3.2.4. Tăng cường dữ liệu 38
3.2.5. Chia tập dữ liệu 39
3.3. Huấn luyện mô hình 40
3.4. Đánh giá mô hình 42
3.4.1. Tiêu chí đánh giá 42
3.4.2. So sánh kết quả đánh giá từ mô hình 42
3.5. Triển khai mô hình 43
3.5.1. Thư viện và những hàm quan trọng 43
3.5.2. Triển khai ứng dụng thực tế 46
KẾT LUẬN 50
TÀI LIỆU THAM KHẢO 51

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Tên đầy đủ	Giải thích
C2F	Cross Stage Partial Focus	Khối xử lý đặc trưng tập trung.
C2PSA	Cross Stage Partial with Spatial Attention	Khối xử lý đặc trưng có thêm chú ý vị trí.
C3K	Cross Stage Partial with 3 Kernels	Khối xử lý đặc trưng dùng 3 loại kernel
C3K2	Cross Stage Partial with 2 C3K blocks	Hai khối C3K xếp chồng lên nhau.
CNN	Convolutional Neural Networks	Mạng nơ-ron tích chập
CSP	Cross Stage Partial	Kỹ thuật tối ưu cấu trúc mạng
FC	Fully Connected	Lớp kết nối đầy đủ
FPS	Frames Per Second	Tốc độ xử lý khung hình mỗi giây
IoU	Intersection over Union	Chỉ số đo sự trùng khớp của hộp dự đoán
mAP	mean Average Precision	Độ chính xác trung bình
PIL	Python Imaging Library	Thư viện xử lý ảnh cho Python.
PSA	Partial Spatial Attention	Cơ chế chú ý vào vùng quan trọng
R-CNN	Regions with Convolutional Neural Networks	Mạng nơ-ron tích chập theo vùng
SPPF	Spatial Pyramid Pooling Fast	Kỹ thuật gộp nhanh, giúp nhận diện vật thể ở nhiều kích thước.
YOLO	You Only Look Once	Mô hình phát hiện vật thể theo thời gian thực.

DANH MỤC BẢNG BIỂU

Bảng 2.1. So sánh các mô hình YOLOv11 huấn luyện trên tập COCO.....33
Bảng 3.1. Bảng so sánh hiệu suất các mô hình YOLO.....43

DANH MỤC HÌNH ẢNH

Hình 1.1. Mối quan hệ giữa Trí tuệ nhân tạo, Học máy và Học sâu 5
Hình 1.2. Cấu trúc mạng nơ-ron cơ bản 6
Hình 1.3. Cấu trúc tổng quan mạng nơ-ron tích chập 7
Hình 1.4. Cách thức hoạt động của Convolution Layer 8
Hình 1.5. Cách thức hoạt động của Pooling Layer 8
Hình 1.6. Cách thức hoạt động của Fully Connected Layer 9
Hình 1.7. Output của bài toán Object Detection 10
Hình 2.1. Các phiên bản YOLO 18
Hình 2.2. Biểu đồ so sánh hiệu năng các phiên bản của YOLO 18
Hình 2.3. Kiến trúc của mô hình YOLO 20
Hình 2.4. Quá trình hoạt động và trả về output của model YOLO 21
Hình 2.5. YOLO chia feature map trên ảnh có kích thước 416x416 21
Hình 2.6. Xác định anchor box cho một vật thể 22
Hình 2.7. Công thức ước lượng bounding box từ anchor box 24
Hình 2.8. Ảnh trước và sau khi thực hiện non-max suppression 24
Hình 2.9. Kiến trúc của mô hình YOLOv11 26
Hình 2.10. Cấu trúc Conv Block 27
Hình 2.11. Cấu trúc Bottle Neck Block 27
Hình 2.12. Cấu trúc khối C2F 28
Hình 2.13. Cấu trúc khối C3K2 29
Hình 2.14. Cấu trúc khối C3K 30
Hình 2.15. Cách thức hoạt động của SPPF 31
Hình 2.16. Cách thức hoạt động của C2PSA 32
Hình 2.17. Kiến trúc phần Head trong YOLOv11 33
Hình 3.1. Ảnh được gán nhãn 37
Hình 3.2. Hình ảnh được resize về kích thước 640x640 38
Hình 3.3. Ảnh sau khi sử dụng một số kỹ thuật tăng cường 39
Hình 3.4. Cấu trúc dữ liệu sau gán nhãn 39
Hình 3.5. Cấu trúc file data.yaml 40
Hình 3.6. Hàm Load model 43
Hình 3.7. Hàm vẽ bounding box 44
Hình 3.8. Hàm xử lý ảnh khi truyền vào 44

Hình 3.9. Hàm xử lý video.....45
Hình 3.10. Hàm thống kê khi người dùng tải ảnh lên.....46
Hình 3.11. Giao diện web của mô hình.....47
Hình 3.12. Giao diện sau khi người dùng tải ảnh lên.....47
Hình 3.13. Kết quả thống kê khi phát hiện ảnh.....48
Hình 3.14. Giao diện sau khi người dùng tải video lên.....48

MỞ ĐẦU

Tai nạn giao thông từ lâu vẫn luôn là vấn đề nhức nhối ở Việt Nam, gây thiệt hại lớn cả về người và tài sản. Theo Báo cáo tình hình kinh tế – xã hội của Tổng cục Thống kê năm 2024, cả nước đã xảy ra 23.484 vụ tai nạn giao thông, làm 10.944 người tử vong và 17.342 người bị thương [1]. Một trong những nguyên nhân chính gây ra hậu quả nghiêm trọng đó là việc nhiều người điều khiển xe máy không đội mũ bảo hiểm hoặc đội không đúng cách, nhất là ở vùng nông thôn, vào ban đêm hoặc tại các tuyến đường vắng.

Mũ bảo hiểm từ lâu đã được xem là vật dụng bắt buộc và không thể thiếu đối với người đi xe máy. Theo Tổ chức Y tế Thế giới (WHO), đội mũ bảo hiểm đúng cách có thể giảm 69% nguy cơ chấn thương sọ não và giảm 42% nguy cơ tử vong do tai nạn giao thông đường bộ [2]. Dù pháp luật Việt Nam đã quy định bắt buộc đội mũ bảo hiểm, nhưng việc giám sát hiện nay chủ yếu dựa vào lực lượng cảnh sát giao thông, vốn có hạn chế về nhân lực và khó đảm bảo kiểm tra liên tục, trên diện rộng. Do đó, việc xây dựng một hệ thống giám sát tự động, thông minh để phát hiện và xử lý vi phạm kịp thời là rất cần thiết.

Cùng với sự phát triển của trí tuệ nhân tạo (AI) và học sâu (Deep Learning), các mô hình phát hiện vật thể như YOLO đã trở nên hiệu quả trong giám sát giao thông nhờ khả năng nhận diện nhanh và chính xác. Vì vậy, em chọn đề tài "**Ứng dụng học sâu trong nhận diện hành vi không đội mũ bảo hiểm khi tham gia giao thông**" nhằm xây dựng hệ thống tự động phát hiện vi phạm từ hình ảnh hoặc video, hỗ trợ giám sát và xử lý kịp thời. Đề tài sử dụng mô hình YOLOv11, thực hiện thu thập dữ liệu, huấn luyện, đánh giá và triển khai hệ thống trên ứng dụng web đơn giản để minh họa khả năng ứng dụng thực tế.

Nội dung đồ án gồm các chương chính sau:

Chương 1 – Tổng quan đề công nghệ: Giới thiệu về Trí tuệ nhân tạo, Học máy, Học sâu, mạng nơ-ron, bài toán thị giác máy tính và các công nghệ sử dụng trong đề tài.

Chương 2 – Giới thiệu về YOLO: Giới thiệu về mô hình YOLO, lịch sử phát triển, cấu trúc của YOLOv11 và ứng dụng thực tế

Chương 3 – Xây dựng ứng dụng: Giới thiệu về bài toán, quá trình thu thập dữ liệu, huấn luyện mô hình và xây dựng ứng dụng trên nền tảng web

CHƯƠNG 1: TỔNG QUAN VỀ CÔNG NGHỆ

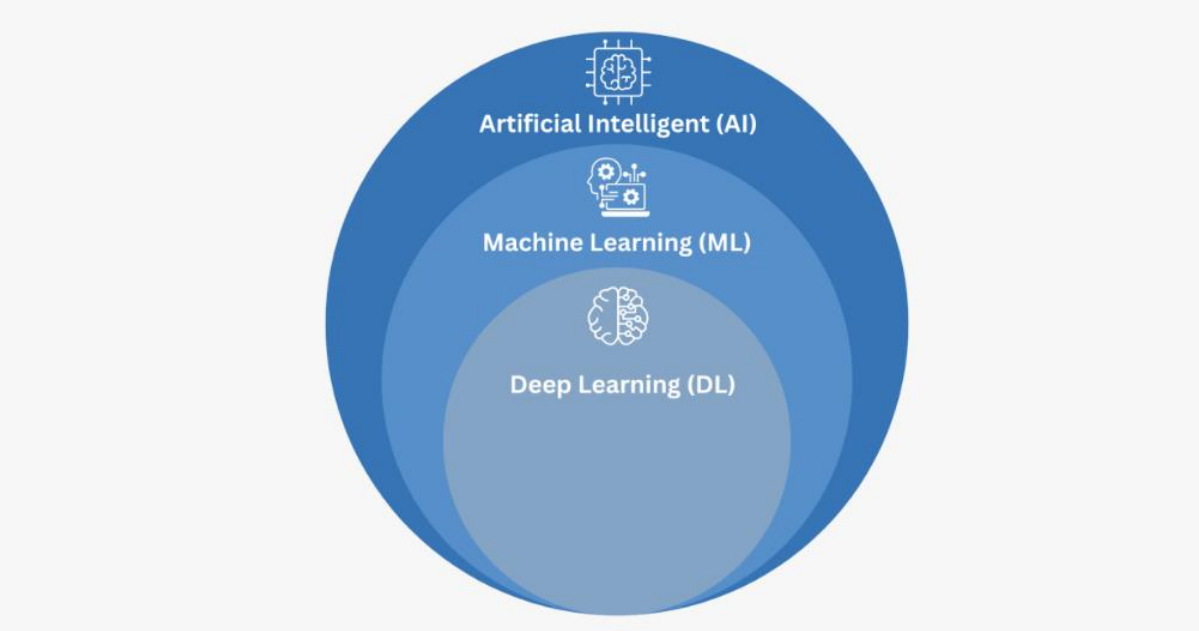
1.1. Giới thiệu

Trong kỷ nguyên mà công nghệ số đang bùng nổ như hiện nay, ba thuật ngữ nổi bật bao gồm: Trí tuệ nhân tạo (Artificial Intelligence - AI), Học máy (Machine Learning), và Học sâu (Deep Learning) đã trở thành trung tâm của nhiều cuộc thảo luận và nghiên cứu. Dù chúng có sự liên kết chặt chẽ và bổ sung cho nhau, song mỗi khái niệm đều mang đến những đặc điểm và ứng dụng đặc thù riêng.

Trí tuệ nhân tạo (AI) là một nhánh của khoa học máy tính, giúp máy móc có thể suy nghĩ và hành động thông minh như con người.

Học máy (Machine Learning) là một tập con của trí tuệ nhân tạo, giúp máy móc có thể tự học từ dữ liệu mà không cần lập trình cụ thể.

Học sâu (Deep Learning) là tập con của Học máy, mô phỏng cách hoạt động của não người, giúp máy móc học tốt hơn từ dữ liệu phức tạp.



Hình 1.1. Mối quan hệ giữa Trí tuệ nhân tạo, Học máy và Học sâu

(Ảnh: Mark John)

1.2. Học sâu

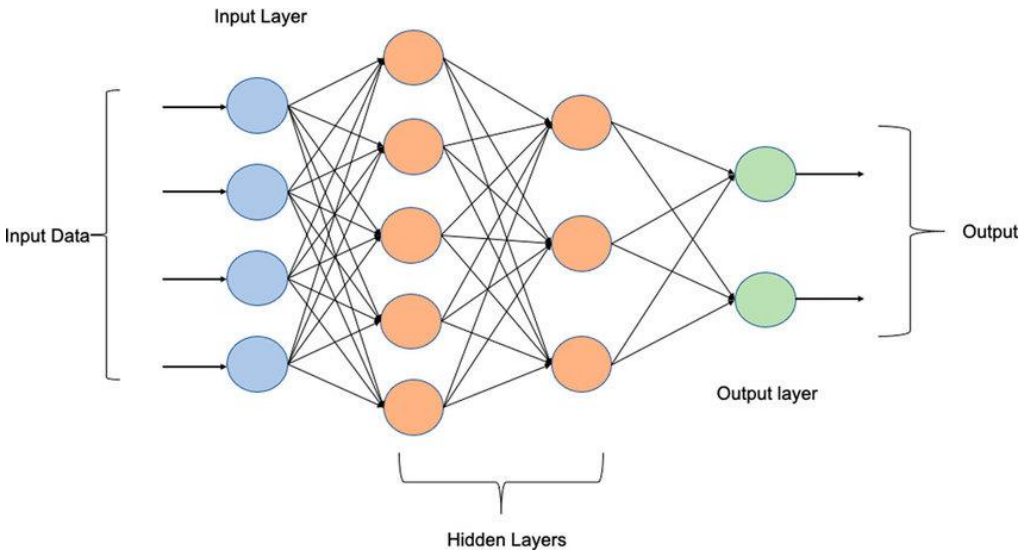
1.2.1. Khái niệm

Học sâu là một phân nhánh của Học máy, lấy cảm hứng từ cấu trúc và chức năng của não bộ con người, sử dụng các mạng nơ-ron (neural networks) nhiều lớp để học và trích xuất đặc trưng từ dữ liệu.

Học sâu hoạt động dựa trên nhiều lớp nơ-ron, trong đó mỗi lớp có nhiệm vụ xử lý một khía cạnh khác nhau của dữ liệu. Khi thông tin đi qua nhiều lớp này, hệ thống có thể hiểu sâu sắc hơn về dữ liệu đầu vào. Nhờ đó, học sâu đạt được những thành tựu ấn tượng trong các lĩnh vực như: nhận diện hình ảnh, xử lý giọng nói và trí tuệ nhân tạo trong y tế.

1.2.2. Mạng nơ-ron

Mạng nơ-ron (Neural Network) là một mô hình tính toán lấy cảm hứng từ mạng nơ-ron thần kinh trong não bộ của con người. Mạng nơ-ron được thiết kế để giải quyết các bài toán phức tạp cũng như: phân loại, nhận dạng hình ảnh và xử lý ngôn ngữ tự nhiên.



Hình 1.2. Cấu trúc mạng nơ-ron cơ bản

Mạng nơ-ron gồm các đơn vị nhỏ gọi là nơ-ron, mỗi nơ-ron như một hàm tính toán. Mạng thường có ba lớp chính:

- Lớp đầu vào (Input Layer): Là lớp nằm ngoài cùng bên trái, thực hiện nhiệm vụ tiếp nhận dữ liệu đầu vào. Mỗi nút trong lớp này đại diện cho một đặc trưng (feature) của dữ liệu.
- Lớp ẩn (Hidden Layer): Nằm giữa lớp đầu vào và lớp đầu ra. Đây là nơi diễn ra quá trình xử lý và trích xuất đặc trưng, giúp mạng học được các mối quan hệ phức tạp trong dữ liệu. Một mạng nơ-ron có thể có một hoặc nhiều lớp ẩn.
- Lớp đầu ra (Output Layer): Lớp cuối cùng bên phải, có nhiệm vụ đưa ra kết quả dự đoán hoặc phân loại dựa trên thông tin đã được xử lý từ các lớp ẩn.

1.2.3. Ứng dụng của học sâu

Học sâu được ứng dụng rộng rãi trong các lĩnh vực của đời sống giúp tự động hóa, nâng cao hiệu suất và hỗ trợ ra quyết định thông minh. Dưới đây là một số lĩnh vực tiêu biểu:

- Y tế: Chẩn đoán bệnh từ ảnh y khoa, phát hiện và dự đoán bệnh, …
- Giao thông: Xe tự lái, phân tích camera giao thông, …
- Tài chính ngân hàng: Phát hiện gian lận, dự đoán rủi ro tín dụng, …
- Thương mại điện tử: Gợi ý sản phẩm, Phân tích cảm xúc khách hàng, …
- Giải trí và truyền thông: Xử lý ngôn ngữ tự nhiên, tạo nội dung, …

1.3. Mạng nơ-ron tích chập

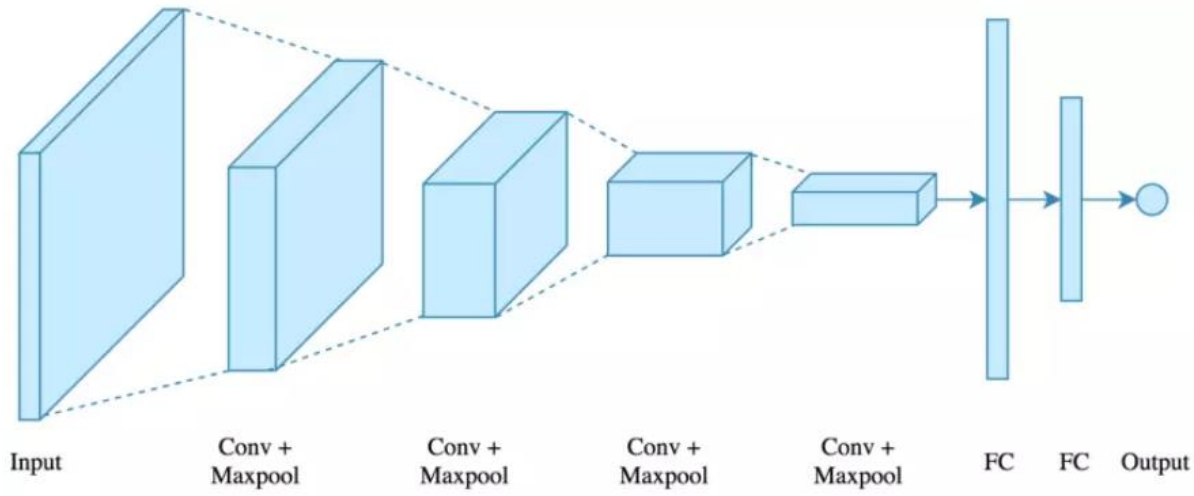
1.3.1. Khái niệm

Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một loại mạng nơ-ron nhân tạo chuyên biệt, được dùng để xử lý dữ liệu có cấu trúc lưới như: hình ảnh và video. CNN đặc biệt hiệu quả trong các bài toán liên quan đến thị giác máy tính như phân loại ảnh, phát hiện đối tượng và nhận dạng khuôn mặt.

1.3.2. Cấu trúc

CNN hoạt động theo trình tự:

Input image ➔ Convolution Layer (Conv) + Maxpool (Lặp lại nhiều lần) ➔ Fully Connected(FC) ➔ Output

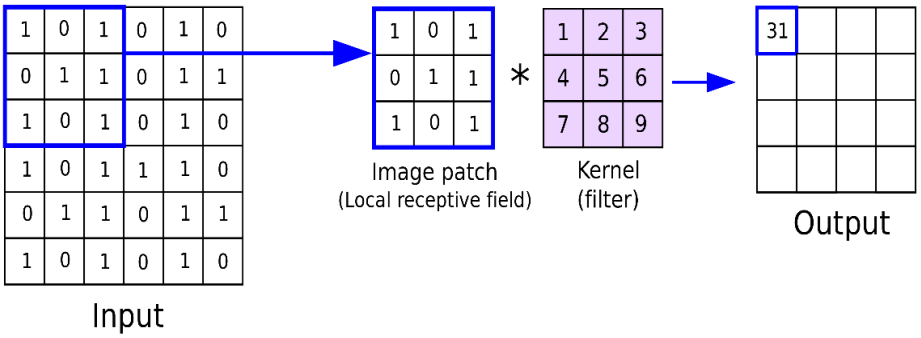


Hình 1.3. Cấu trúc tổng quan mạng nơ-ron tích chập

Lớp đầu vào (Input): Đây là nơi dữ liệu (ví dụ như một hình ảnh) được đưa vào mô hình. Hình ảnh thường được biểu diễn dưới dạng một ma trận pixel (chiều rộng x chiều cao x kênh màu).

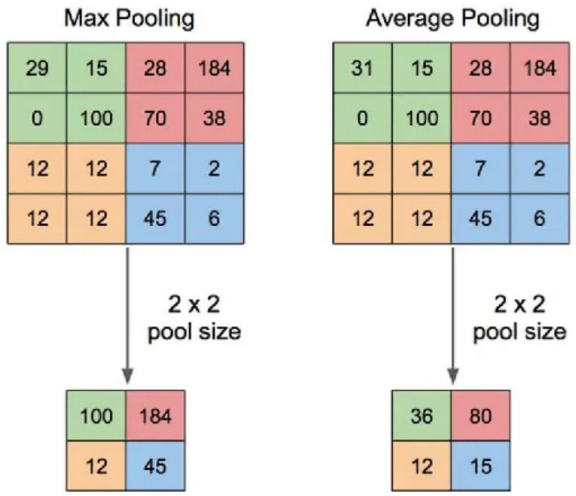
Các lớp tích chập và gộp (Conv + Maxpool):

- Lớp tích chập (Convolutional Layer - Conv): Lớp này áp dụng các bộ lọc (kernel) trên dữ liệu đầu vào. Mỗi bộ lọc sẽ quét qua toàn bộ hình ảnh và tạo ra một "bản đồ đặc trưng" (feature map) mới, làm nổi bật các đặc trưng như cạnh, đường viền, hoặc các mẫu hình cụ thể. Mô hình sẽ học cách điều chỉnh các giá trị trong bộ lọc này trong quá trình huấn luyện để nhận diện các đặc trưng có ý nghĩa.



Hình 1.4. Cách thức hoạt động của Convolution Layer

- Lớp Gộp (Pooling Layer - Maxpool): Lớp này thường được đặt sau lớp tích chập. Mục đích của lớp gộp là giảm kích thước của bản đồ đặc trưng, từ đó giảm số lượng tham số và tính toán, giúp mô hình bớt nhạy cảm với sự dịch chuyển nhỏ của các đặc trưng, và tránh overfitting. Max pooling là loại gộp phổ biến nhất, chọn giá trị lớn nhất trong một cửa sổ nhỏ.

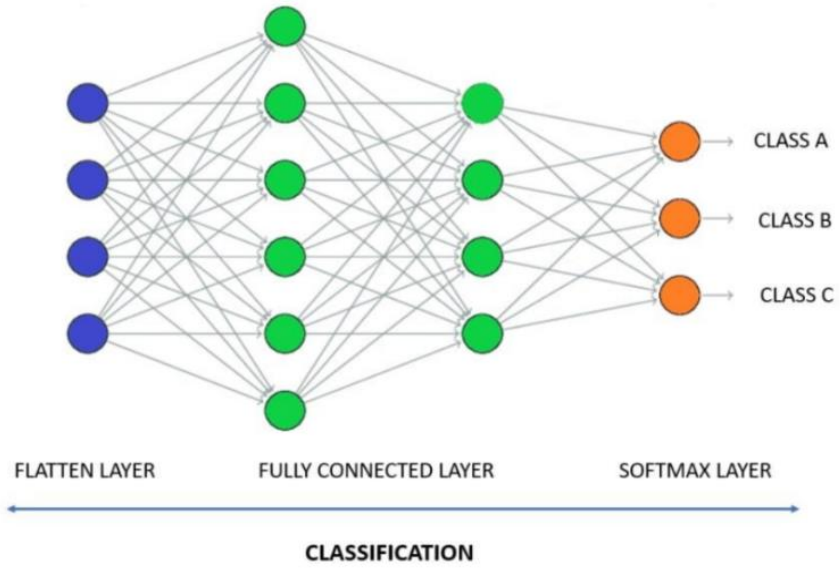


Hình 1.5. Cách thức hoạt động của Pooling Layer

Quá trình "Conv + Maxpool" này thường được lặp lại nhiều lần trong mạng, với các lớp sâu hơn học các đặc trưng phức tạp hơn từ các đặc trưng thô ban đầu.

Các lớp kết nối đầy đủ (Fully Connected - FC):

- Sau khi các lớp tích chập và gộp đã trích xuất các đặc trưng cấp cao, dữ liệu được "làm phẳng" (flatten) thành một vector duy nhất.



Hình 1.6. Cách thức hoạt động của Fully Connected Layer

- Vector này sau đó được đưa vào một hoặc nhiều lớp kết nối đầy đủ. Các lớp này hoạt động giống như một mạng nơ-ron truyền thống, nơi mỗi nơ-ron trong một lớp được kết nối với mọi nơ-ron trong lớp tiếp theo. Mục đích của các lớp FC là học các mối quan hệ phi tuyến giữa các đặc trưng được trích xuất và ánh xạ chúng đến đầu ra cuối cùng.

Lớp đầu ra (Output): Đây là lớp cuối cùng của mạng, tạo ra kết quả dự đoán.

- Đối với bài toán phân loại hình ảnh, lớp đầu ra thường sử dụng hàm kích hoạt softmax để đưa ra xác suất của hình ảnh thuộc về mỗi lớp.
- Đối với các bài toán khác, đầu ra có thể là một giá trị số

1.4. Thị giác máy tính

1.4.1. Thị giác máy tính và các bài toán liên quan

Thị giác máy tính (Computer vision) là một lĩnh vực của trí tuệ nhân tạo giúp máy tính “nhìn” và “hiểu” thế giới như con người. Nó xử lý và phân tích hình ảnh hoặc video để trích xuất thông tin, nhận dạng các mẫu và đưa ra quyết định dựa trên dữ liệu đó.

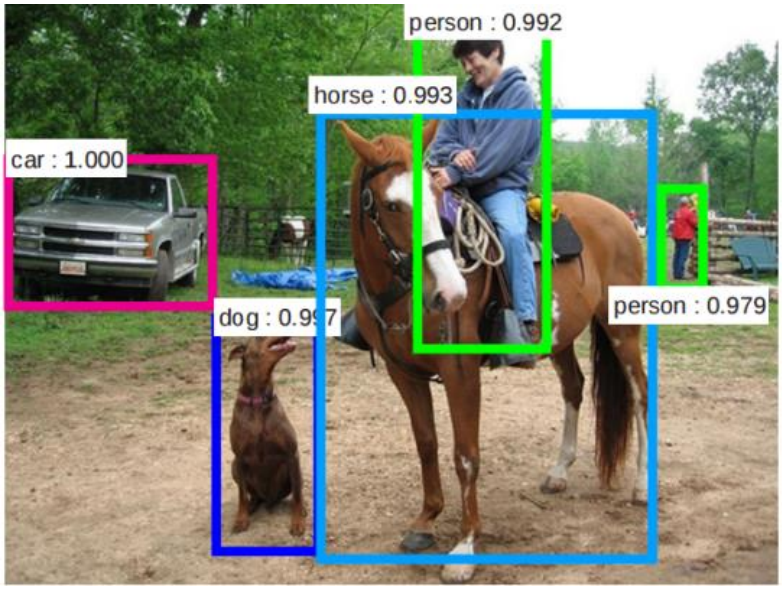
Thị giác máy tính đóng vai trò quan trọng trong nhiều ứng dụng thực tế, từ các hệ thống an ninh giám sát, y tế, sản xuất công nghiệp đến xe tự lái và thực tế ảo.

Các bài toán chính trong lĩnh vực thị giác máy tính bao gồm:

- Phân loại ảnh (Image Classification): Đây là bài toán xác định ảnh thuộc đối tượng nào trong danh sách các nhãn đã biết. Bài toán này hiệu quả khi ảnh chỉ chứa duy nhất một đối tượng.
- Phát hiện đối tượng (Object Detection): Bài toán này giúp mô hình phân loại và xác định vị trí của từng đối tượng trong ảnh thông qua các hộp giới hạn (bounding box).
- Phân đoạn ngữ nghĩa (Semantic Segmentation): Bài toán này yêu cầu mô hình gán nhãn cho từng pixel trong ảnh, sau đó phân loại chúng vào các lớp đối tượng cụ thể (ví dụ: người, xe, đường)
- Nhận dạng khuôn mặt (Face Recognition): Đây là bài toán dùng để xác định danh tính của một người dựa trên khuôn mặt, thường được sử dụng trong các hệ thống bảo mật, điểm danh.
- Theo dõi đối tượng (Object Tracking): Mục tiêu của bài toán này là theo dõi sự di chuyển của đối tượng trong một chuỗi video hoặc các khung hình liên tiếp, thường được kết hợp với Object Detection để vừa phát hiện, vừa theo dõi liên tục đối tượng đó qua thời gian.

1.4.2. Bài toán Object Detection

Object Detection (Phát hiện đối tượng) là bài toán dùng để **xác định vị trí và phân loại nhiều đối tượng trong cùng một ảnh** bằng cách vẽ các hộp giới hạn (bounding boxes). So với bài toán phân loại ảnh, bài toán này xử lý tốt các ảnh có nhiều đối tượng khác nhau**.**



Hình 1.7. Output của bài toán Object Detection

Quy trình của bài toán Object Detection bao gồm 3 bước chính:

- Classification (Phân loại đối tượng): Từ dữ liệu ảnh đưa vào, mô hình sẽ xác định những loại đối tượng nào có trong ảnh.
- Localization (Định vị đối tượng): Tại đây, mô hình sẽ xác định vị trí của từng đối tượng bằng các hộp giới hạn.
- Detection (Phát hiện đối tượng): Cuối cùng kết hợp cả phân loại và định vị để xác định rõ từng đối tượng trong ảnh cùng với vị trí và nhãn tương ứng.

Bài toán Object Detection được ứng dụng rộng rãi trong nhiều lĩnh vực thực tế, tiêu biểu như:

- Giám sát an ninh: Phát hiện người lạ, theo dõi chuyển động,…
- Giao thông thông minh: Nhận diện biển số xe, phát hiện vi phạm giao thông, đếm phương tiện, …
- Y tế: Phát hiện khối u, tế bào bất thường trong ảnh chụp X-quang, …
- Nông nghiệp: Theo dõi sâu bệnh, đếm trái cây, phân loại cây trồng qua ảnh,...
- Bán lẻ: Theo dõi hành vi khách hàng, kiểm kê hàng hóa, chống trộm cắp,…
- Công nghiệp: Giám sát dây chuyền sản xuất, phát hiện lỗi sản phẩm,…

1.4.3. Các mô hình của bài toán Object Detection

Các mô hình của bài toán Object Detection được chia thành 2 nhóm chính:

- **Phương pháp hai giai đoạn (Two-stage Detectors):** Đây là nhóm mô hình có độ chính xác cao nhưng tốc độ xử lý chậm hơn. Quá trình phát hiện gồm hai bước: đầu tiên đề xuất các vùng có khả năng chứa đối tượng, sau đó phân loại và điều chỉnh hộp giới hạn cho từng vùng. Các mô hình tiêu biểu thuộc nhóm này là: R-CNN, Fast R-CNN và Faster R-CNN.
- **Phương pháp một giai đoạn (One-stage Detectors):** các mô hình của nhóm này ưu tiên tốc độ xử lý nhanh, phù hợp với các bài toán yêu cầu thời gian thực. Mô hình sẽ trực tiếp xác định vị trí và phân loại đối tượng chỉ trong một lần truyền qua mạng. Đại diện tiêu biểu là SSD và các phiên bản của YOLO (You Only Look Once).

Trong đồ án này, em lựa chọn sử dụng YOLO - một mô hình phát hiện đối tượng một giai đoạn, nổi bật với khả năng xử lý nhanh, độ chính xác cao và tối ưu cho môi trường giao thông thực tế. So với các mô hình phát hiện hai giai đoạn như Faster R-CNN, YOLO cho phép phát hiện đối tượng gần như thời gian thực, rất phù hợp với yêu cầu phát hiện hành vi không đội mũ bảo hiểm. Chi tiết về YOLO được trình bày ở chương 2.

1.4.4. Các kĩ thuật xử lý dữ liệu cho bài toán Object Detection

Để huấn luyện mô hình phát hiện đối tượng hiệu quả, dữ liệu đầu vào cần được xử lý kỹ lưỡng. Một số kỹ thuật xử lý dữ liệu phổ biến gồm:

- Chuẩn hóa ảnh (Scaling): Ảnh đầu vào được thay đổi kích thước về một chuẩn cố định để đảm bảo tính đồng nhất và giúp mô hình học tập hiệu quả hơn.
- Tăng cường dữ liệu (Data Augmentation): Áp dụng các phép biến đổi như xoay ảnh, lật ngang, phóng to/thu nhỏ, thay đổi độ sáng hoặc độ tương phản. Điều này giúp tăng đa dạng dữ liệu huấn luyện, giảm hiện tượng overfitting và cải thiện khả năng tổng quát của mô hình.
- Xử lý Bounding Boxes**:** Tọa độ các hộp giới hạn (bounding boxes) được chuyển đổi sang dạng tương đối so với kích thước ảnh để mô hình dễ dàng học. Ngoài ra, thuật toán Non-Maximum Suppression được sử dụng để loại bỏ các dự đoán trùng lặp, giữ lại những bounding box có độ tin cậy cao nhất.
- Chia tập dữ liệu**:** Dữ liệu được phân chia thành các tập huấn luyện, kiểm tra và validation để đánh giá mô hình một cách khách quan và tránh tình trạng mô hình chỉ học thuộc dữ liệu huấn luyện.
- Xử lý nhãn phân loại (Categorical Labels)**:** Các nhãn đối tượng được chuyển đổi từ dạng chữ sang dạng số (one-hot encoding hoặc integer encoding) để phù hợp với đầu vào của mô hình học sâu.

1.5. Các công nghệ sử dụng

Trong đề tài "Ứng dụng học máy trong nhận diện hành vi không đội mũ bảo hiểm khi tham gia giao thông", em có sử dụng ngôn ngữ lập trình Python kết hợp với một số công cụ hiện đại và thư viện mã nguồn mở mạnh mẽ như: Google Colab, Visual Studio Code, NumPy, OpenCV, PIL và Streamlit.

Tất cả những công cụ và thư viện này đóng vai trò then chốt trong việc xây dựng, huấn luyện, kiểm thử và triển khai mô hình học sâu.

1.5.1. Python

Python là ngôn ngữ lập trình chính xuyên suốt toàn bộ đề tài. Python được lựa chọn nhờ cú pháp đơn giản, dễ tiếp cận, tính linh hoạt cao và đặc biệt là có hệ sinh thái phong phú trong lĩnh vực trí tuệ nhân tạo.

Với số lượng thư viện lớn và cộng đồng sử dụng đông đảo, Python giúp người phát triển dễ dàng thử nghiệm và hiện thực hóa các ý tưởng học máy một cách nhanh chóng và hiệu quả. Trong đề tài này, Python được dùng để viết toàn bộ mã nguồn xử lý dữ liệu, huấn luyện mô hình, kiểm thử kết quả cũng như xây dựng giao diện demo trên web.

1.5.2. Google Colab

Google Colab là nền tảng điện toán đám mây được sử dụng để thực hiện các bước xử lý tốn tài nguyên như huấn luyện mô hình YOLO. Với lợi thế hỗ trợ GPU miễn phí, Google Colab giúp đẩy nhanh tốc độ huấn luyện mà không yêu cầu máy tính cấu hình cao. Ngoài ra, Colab còn tích hợp sẵn nhiều thư viện học máy như PyTorch, NumPy, OpenCV, giúp việc lập trình trở nên đơn giản hơn. Trong đề tài, em sử dụng Google Colab để:

- Tải tập dữ liệu đã gán nhãn lên môi trường điện toán đám mây.
- Cài đặt và cấu hình thư viện YOLO từ Ultralytics.
- Huấn luyện mô hình nhận diện hành vi không đội mũ bảo hiểm.
- Đánh giá hiệu suất mô hình thông qua các chỉ số như độ chính xác (precision), độ nhạy (recall) và mAP (mean Average Precision).

1.5.3. Visual Studio Code

Visual Studio Code (VS Code) là môi trường phát triển mã nguồn được sử dụng để lập trình và triển khai hệ thống cục bộ. Với khả năng hỗ trợ tốt cho Python, cùng hệ thống tiện ích mở rộng phong phú như Pylance, Jupyter, Git và Streamlit plugin, VS Code giúp quá trình lập trình, kiểm thử và tổ chức dự án trở nên dễ dàng và trực quan hơn.

1.5.4. Numpy

NumPy là thư viện nền tảng dùng để thao tác và tính toán trên dữ liệu dạng mảng (array) và ma trận (matrix). Trong đề tài, NumPy hỗ trợ các chức năng xử lý ảnh đầu vào như biến đổi kích thước, chuẩn hóa giá trị pixel, và tính toán các chỉ số liên quan đến bounding box. Thư viện này giúp giảm thiểu độ trễ trong pipeline xử lý ảnh và tăng tốc quá trình truyền dữ liệu vào mô hình.

1.5.5. OpenCV

OpenCV (Open Source Computer Vision Library) là thư viện xử lý ảnh và video rất phổ biến trong các ứng dụng thị giác máy tính. Trong đề tài này, OpenCV đóng vai trò quan trọng trong toàn bộ quy trình xử lý ảnh đầu vào và hiển thị kết quả đầu ra từ mô hình học máy.

Cụ thể, OpenCV được sử dụng để đọc ảnh và video từ nhiều nguồn khác nhau như tệp ảnh, video có sẵn hoặc luồng trực tiếp từ webcam; chuyển đổi không gian màu từ BGR (chuẩn của OpenCV) sang RGB (chuẩn của các thư viện học sâu như PyTorch) để đảm bảo tương thích với mô hình huấn luyện; đồng thời, sau khi mô hình dự đoán xong, OpenCV được dùng để vẽ các hộp giới hạn (bounding box) bao quanh các đối tượng được phát hiện và gắn nhãn kết quả tương ứng như "helmet" hoặc "no_helmet" lên ảnh gốc, giúp người dùng dễ dàng quan sát kết quả nhận diện một cách trực quan và sinh động.

1.5.6. PIL

PIL (Python Imaging Library), hay cụ thể là Pillow – phiên bản nâng cấp và được duy trì hiện nay của PIL, là thư viện hỗ trợ thao tác ảnh tĩnh rất phổ biến trong Python. Trong dự án này, Pillow được sử dụng để mở ảnh từ nhiều định dạng khác nhau như JPEG, PNG,… một cách linh hoạt và tiện lợi; đồng thời thực hiện các thao tác xử lý như thay đổi kích thước (resize), chuyển đổi định dạng ảnh để phù hợp với yêu cầu đầu vào của mô hình hoặc các bước xử lý tiếp theo. Bên cạnh đó, Pillow còn hỗ trợ hiển thị và lưu ảnh đầu ra sau khi mô hình dự đoán, đặc biệt khi tích hợp với giao diện người dùng

Streamlit, cho phép hiển thị kết quả trực quan ngay trong trình duyệt, từ đó nâng cao trải nghiệm tương tác và đánh giá hiệu quả mô hình.

1.5.7. Streamlit

Streamlit là một thư viện Python rất tiện lợi, giúp cho người dùng dễ dàng tạo ra giao diện cho các ứng dụng học máy mà không cần phải dùng đến các framework phức tạp như Flask hay Django.

Trong đề tài này, em dùng Streamlit để xây dựng một trang web nhỏ, nơi người dùng có thể tải ảnh hoặc video lên để kiểm tra xem người trong ảnh có đội mũ bảo hiểm hay không. Sau khi người dùng tải ảnh hoặc video lên, ứng dụng sẽ gọi mô hình đã huấn luyện để dự đoán và hiển thị kết quả ngay trên giao diện. Kết quả gồm ảnh gốc kèm theo các hộp phát hiện (bounding box), nhãn lớp ("helmet" hoặc "no_helmet") và xác suất. Ngoài ra, giao diện cũng thêm vào các nút bấm như chọn ảnh, xem kết quả, hoặc tải ảnh kết quả về máy để người dùng sử dụng dễ dàng và tiện lợi hơn.

CHƯƠNG 2: GIỚI THIỆU VỀ YOLO

2.1. Tổng quan về YOLO

2.1.1. Khái niệm

YOLO (You Only Look Once) là một thuật toán phát hiện đối tượng theo kiến trúc one-stage, tức là thực hiện phát hiện và phân loại trong một bước duy nhất bằng cách áp dụng một mạng nơ-ron tích chập (CNN) trực tiếp lên toàn bộ ảnh đầu vào. Được giới thiệu lần đầu vào năm 2015 bởi Redmon et al. [3], YOLO nổi bật bởi khả năng phát hiện nhanh chóng, phù hợp với các ứng dụng thời gian thực (real-time)

Tên gọi "You Only Look Once" thể hiện đúng bản chất của mô hình: chỉ cần nhìn ảnh một lần để đưa ra toàn bộ dự đoán.

2.1.2. Lịch sử phát triển

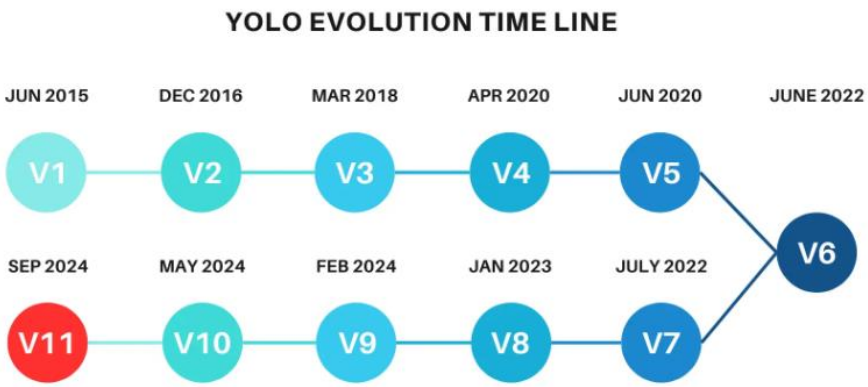
YOLOv1 (2015): Là mô hình YOLO đầu tiên, được phát triển bởi Joseph Redmon và cộng sự, tập trung vào việc đạt được hiệu suất thời gian thực thông qua một kiến trúc mạng duy nhất. Mặc dù có hạn chế trong việc phát hiện các đối tượng nhỏ do hệ thống lưới thô, YOLOv1 đã đặt nền móng vững chắc cho hướng tiếp cận sau này

YOLOv2 (2016): Còn gọi là YOLO9000, phiên bản này cải thiện đáng kể độ chính xác và khả năng định vị đối tượng thông qua việc áp dụng các kỹ thuật như Batch Normalization, Anchor Boxes và khả năng xử lý đầu vào với độ phân giải cao hơn.

YOLOv3 (2018): Giới thiệu kiến trúc phát hiện đa tỉ lệ (multi-scale prediction) thông qua việc sử dụng các tháp đặc trưng (feature pyramids), cho phép mô hình phát hiện hiệu quả hơn các đối tượng ở nhiều kích thước khác nhau.

YOLOv4 (4/2020): Tập trung vào việc tối ưu hóa các kỹ thuật tăng cường dữ liệu (data augmentation) như Mosaic Augmentation và Self-Adversarial Training. Đồng thời, kiến trúc backbone cũng được tối ưu hóa để tăng tốc độ suy luận.

YOLOv5 (6/2020): Được triển khai trên framework PyTorch, YOLOv5 nhanh chóng trở nên phổ biến nhờ tính dễ sử dụng, khả năng tối ưu cho triển khai thực tế và hiệu suất cao.



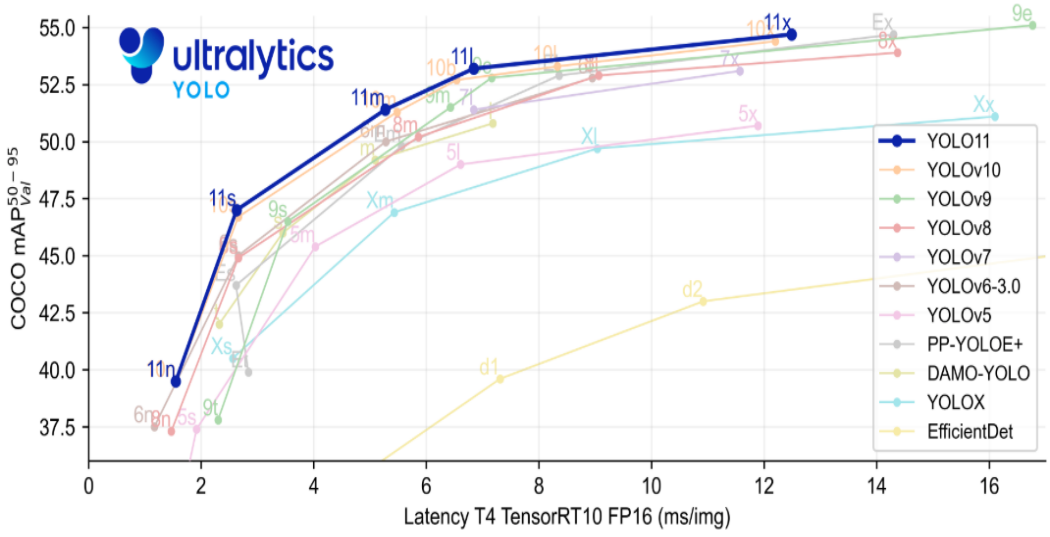
Hình 2.1. Các phiên bản YOLO (Ảnh: Analytics Vidhya)

YOLOv6 và YOLOv7 (2022): Các phiên bản này tiếp tục cải thiện về độ chính xác và khả năng mở rộng của mô hình, đồng thời giới thiệu các biến thể hiệu quả hơn (ví dụ: YOLOv7 Tiny) phù hợp cho các thiết bị biên.

YOLOv8 (2023): Đánh dấu sự thay đổi kiến trúc đáng kể với backbone CSPDarkNet và cơ chế tổng hợp đường dẫn (path aggregation), mang lại những cải tiến đáng kể về cả tốc độ và độ chính xác so với các phiên bản trước.

YOLOv9 (2/2024), YOLOv10 (5/2024), YOLOv11 (9/2024): Các phiên bản gần đây tiếp tục được phát triển, tập trung nâng cao hiệu quả kiến trúc (C3K2 blocks, SPPF) và tích hợp cơ chế chú ý (C2PSA). Mục tiêu là cải thiện khả năng phát hiện đối tượng nhỏ và độ chính xác tổng thể, trong khi vẫn duy trì tốc độ suy luận thời gian thực của YOLO.

2.1.3. So sánh các phiên bản YOLO



Hình 2.2. Biểu đồ so sánh hiệu năng các phiên bản của YOLO (Ảnh: Ultralytics)

Từ biểu đồ trên, có thể thấy rằng YOLOv11 là mô hình có hiệu năng vượt trội nhất. Các phiên bản YOLOv11 (11n, 11s, 11m, 11l, 11x) đều nằm trên đường biên hiệu quả (Pareto front), nghĩa là không có mô hình nào khác vừa nhanh hơn vừa chính xác hơn. Cụ thể:

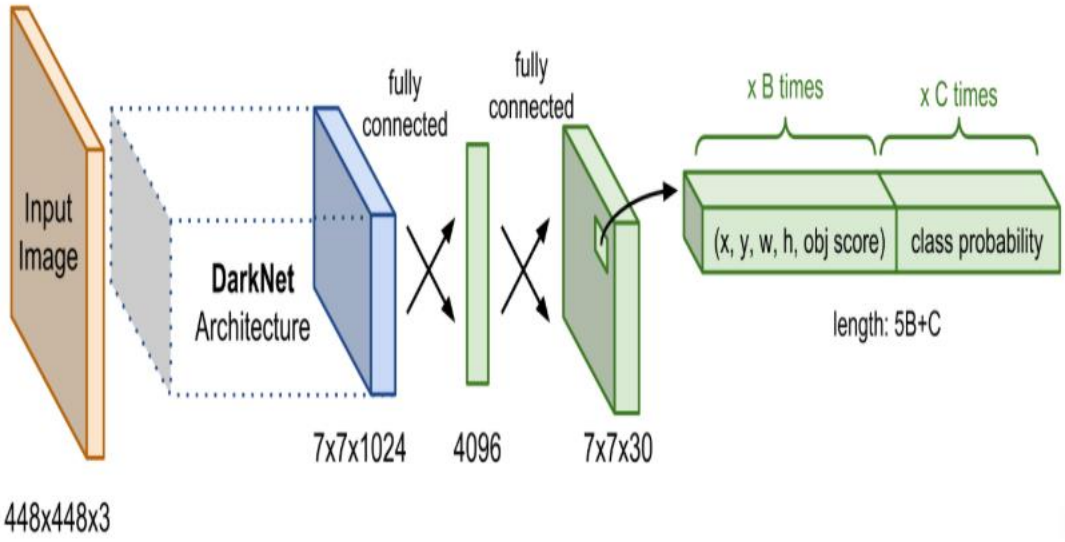
- YOLOv11x đạt độ chính xác cao nhất (~54.8 mAP) trong toàn bộ các mô hình.
- YOLOv11s chỉ mất khoảng 2.5ms để xử lý mỗi ảnh, trong khi vẫn đạt mAP khoảng 51.5 – cực kỳ tối ưu cho các ứng dụng thời gian thực.
- So với YOLOv8 và YOLOv10, YOLOv11 thể hiện sự cải thiện rõ rệt về cả tốc độ lẫn độ chính xác.
- Các mô hình như EfficientDet và YOLOX có độ chính xác thấp hơn đáng kể, mặc dù độ trễ cao hơn.

Đề tài sử dụng YOLOv11 bởi vì:

- Hiệu năng vượt trội: YOLOv11 có mAP cao nhất và độ trễ thấp – đáp ứng cả độ chính xác và tốc độ cho các hệ thống thời gian thực.
- Phiên bản mới nhất: YOLOv11 là phiên bản mới nhất được Ultralytics hỗ trợ mạnh mẽ, thường xuyên cập nhật, tài liệu đầy đủ.
- Dễ triển khai: YOLOv11 hỗ trợ nhiều định dạng đầu ra như YOLO, ONNX TorchScript… phù hợp để tích hợp với hệ thống camera, ứng dụng
- Tối ưu linh hoạt: Có thể lựa chọn phiên bản phù hợp theo tài nguyên phần cứng: 11s/11m cho hệ thống nhẹ, 11x cho độ chính xác cao nhất.

2.1.4. Kiến trúc của mô hình YOLO [4]

Kiến trúc YOLO bao gồm: base network là các mạng convolution làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những Extra Layers được áp dụng để phát hiện vật thể trên feature map của base network.



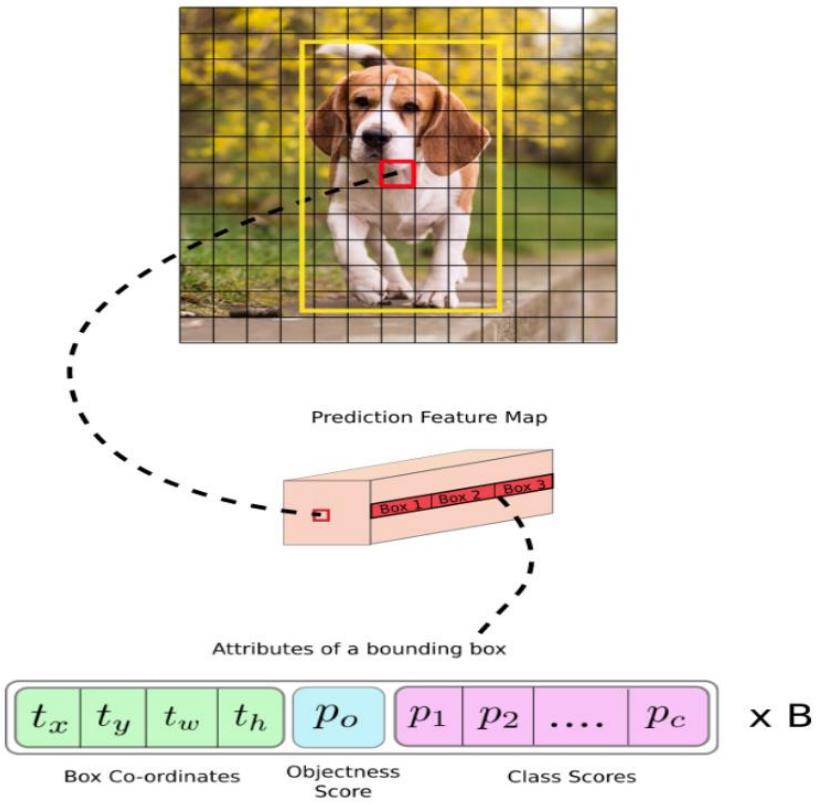
Hình 2.3. Kiến trúc của mô hình YOLO

Thành phần Darknet Architechture được gọi là base network có tác dụng trích suất đặc trưng. Output của base network là một feature map có kích thước 7x7x1024 sẽ được sử dụng làm input cho các Extra layers có tác dụng dự đoán nhãn và tọa độ bounding box của vật thể.

Hiện tại YOLO đang hỗ trợ 2 đầu vào chính là 416x416 và 608x608. Mỗi một đầu vào sẽ có một thiết kế các layers riêng phù hợp với shape của input. Sau khi đi qua các layer convolutional thì shape giảm dần theo cấp số nhân là 2. Cuối cùng ta thu được một feature map có kích thước tương đối nhỏ để dự báo vật thể trên từng ô của feature map.

Kích thước của feature map sẽ phụ thuộc vào đầu vào. Đối với input 416x416 thì feature map có các kích thước là 13x13, 26x26 và 52x52. Và khi input là 608x608 sẽ tạo ra feature map 19x19, 38x38, 72x72.

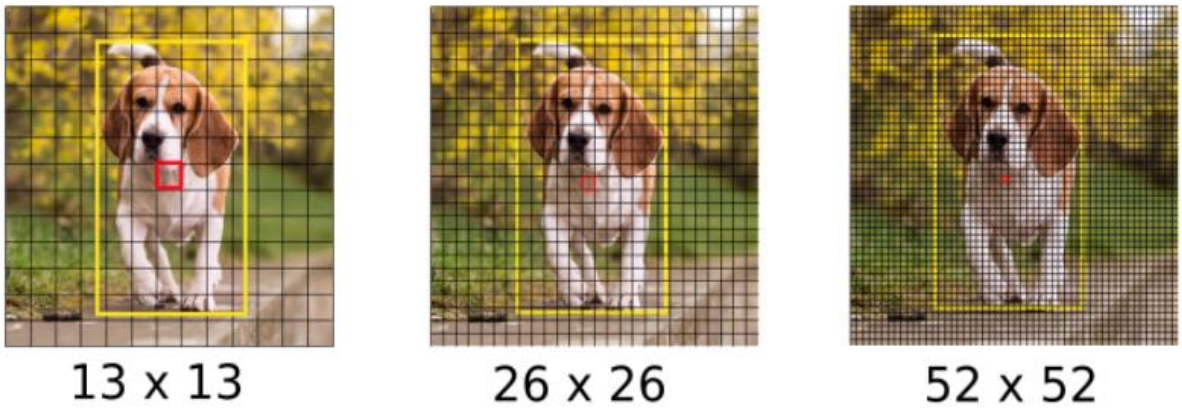
Output của YOLO bao gồm: Danh sách xác suất dự báo vật thể xuất hiện trong bounding box, vị trí tọa độ của các bounding box và xác suất dự báo của các classes.



Hình 2.4. Quá trình hoạt động và trả về output của model YOLO

Dự báo trên nhiều feature map

Trên mỗi một cell của các feature map chúng ta sẽ áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 feature map x 3 anchor box).



Hình 2.5. YOLO chia feature map trên ảnh có kích thước 416x416

Đồng thời trên một feature map hình vuông S x S, mô hình YOLO sinh ra một số lượng anchor box là: S x S x 3. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là:

(13 x 13 + 26 x 26 + 52 x 52) x 3 = 10647(anchor boxes)

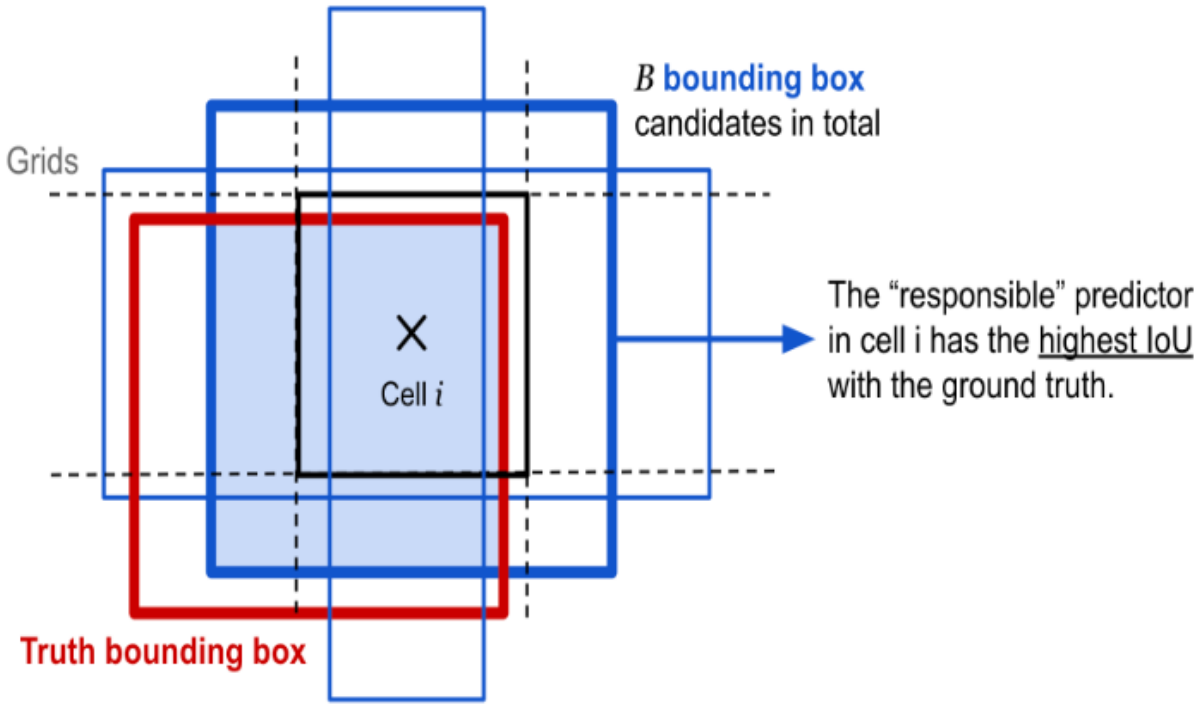
Đây là một số lượng rất lớn và là nguyên nhân khiến quá trình huấn luyện mô hình YOLO vô cùng chậm bởi chúng ta cần dự báo đồng thời nhãn và bounding box trên đồng thời 10647 bounding boxes.

Một số lưu ý khi huấn luyện YOLO:

- Khi huấn luyện YOLO sẽ cần phải có RAM dung lượng lớn hơn để lưu được 10647 bounding boxes như trong kiến trúc này.
- Không thể thiết lập các batch_size quá lớn như trong các mô hình classification vì rất dễ hết bộ nhớ.
- Thời gian xử lý của một step trên YOLO lâu hơn rất rất nhiều lần so với các mô hình classification. Do đó nên thiết lập steps giới hạn huấn luyện cho YOLO nhỏ.

Anchor Box

Để tìm được bounding box (hộp giới hạn) cho vật thể, YOLO sẽ cần các anchor box làm cơ sở ước lượng. Những anchor box này sẽ được xác định trước và sẽ bao quanh vật thể một cách tương đối chính xác. Sau này thuật toán regression bounding box sẽ tinh chỉnh lại anchor box để tạo ra bounding box dự đoán cho vật thể.



Hình 2.6. Xác định anchor box cho một vật thể

Trong một mô hình YOLO:

- Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một anchor box. Trong trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchor box mà có IoU với ground truth bounding box là cao nhất.
- Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một cell trên feature map mà chứa điểm midpoint của vật thể.

Loss Function (Hàm mất mát)

Hàm loss function của YOLO chia thành 2 phần:

- Localization loss (L_{loc}): đo lường sai số của bounding box dự báo so với thực tế
- Confidence loss (L_{cls}): đo lường sai số của phân phối xác suất các classes.

Mô hình YOLO sẽ tối ưu hóa 2 loss function trên sao cho giá trị của chúng gần về 0 nhất.

Dự báo bounding box

Dự báo các bounding box dựa trên phép biến đổi từ anchor box và cell.

YOLO sẽ dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều. Nếu bounding box dự đoán có thể đặt vào bất kỳ phần nào của hình ảnh, như trong region proposal network (mạng đề xuất khu vực), việc huấn luyện mô hình có thể trở nên không ổn định.

Cho một anchor box có kích thước (p_w, p_h) tại cell nằm trên feature map với góc trên cùng bên trái của nó là (c_x, c_y), mô hình dự đoán 4 tham số (t_x, t_y, t_w, t_h) trong đó 2 tham số đầu là độ lệch (offset) so với góc trên cùng bên trái của cell và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định bounding box dự đoán b có tâm (b_x, b_y) và kích thước (b_w, b_h) thông qua hàm sigmoid và hàm exponential như các công thức bên dưới:

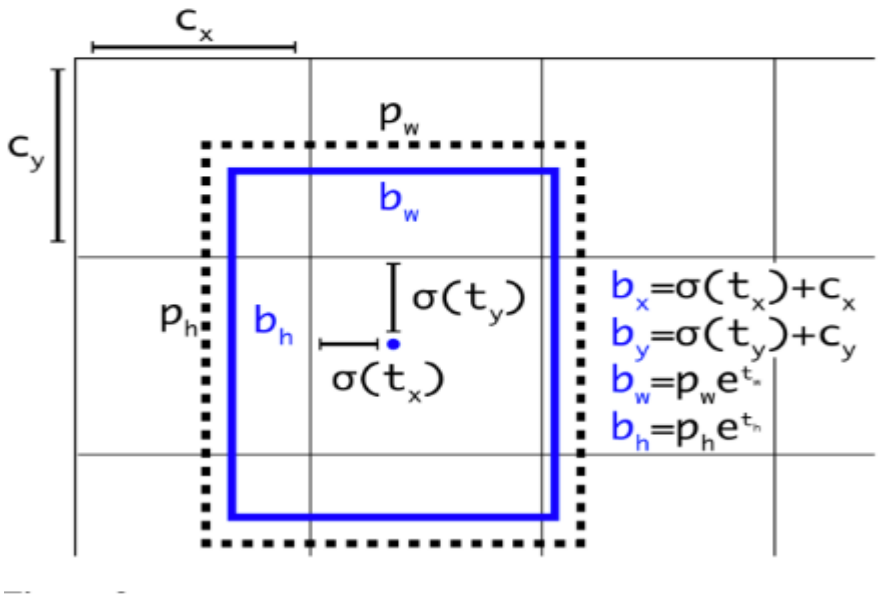
$$\mathrm{b_x} = \sigma(\mathrm{t_x}) + \mathrm{c_x}$$

$$\mathrm{b_y} = \sigma(\mathrm{t_y}) + \mathrm{c_y}$$

$$\mathrm{b_w} = \mathrm{p_w} e^{\mathrm{t_w}}$$

$$\mathrm{b_h} = \mathrm{p_h} e^{\mathrm{t_h}}$$

Ngoài ra do các tọa độ đã được hiệu chỉnh theo chiều rộng (width) và chiều dài (height) của bức ảnh nên luôn có giá trị nằm trong ngưỡng [0, 1]. Do đó khi áp dụng hàm sigmoid giúp ta giới hạn được tọa độ không vượt quá xa các ngưỡng này.



Hình 2.7. Công thức ước lượng bounding box từ anchor box

Giải thích:

Hình chữ nhật nét đứt bên ngoài là anchor box có kích thước là (p_w, p_h)). Tọa độ của một bounding box sẽ được xác định dựa trên đồng thời cả anchor box và cell mà nó thuộc về. Điều này giúp kiểm soát vị trí của bounding box dự đoán đâu đó quanh vị trí của cell và bounding box mà không vượt quá xa ra bên ngoài giới hạn này

Non-max suppression (Loại bỏ hộp giới hạn trùng lặp)

Do thuật toán YOLO dự báo ra rất nhiều bounding box trên một bức ảnh nên đối với những cell có vị trí gần nhau, khả năng các khung hình bị overlap là rất cao. Trong trường hợp đó YOLO sẽ cần đến Non-Max Suppression để giảm bớt số lượng các khung hình được sinh ra một cách đáng kể

Hình 2.8. Ảnh trước và sau khi thực hiện Non-Max Suppression

Các bước của Non-Max Suppression:

- Step 1: Đầu tiên chúng ta sẽ tìm cách giảm bớt số lượng các bounding box bằng cách lọc bỏ toàn bộ những bounding box có xác suất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó, thường là 0.5.
- Step 2: Đối với các bouding box giao nhau, Non-Max Suppression sẽ lựa chọn ra một bounding box có xác xuất chứa vật thể là lớn nhất. Sau đó tính toán chỉ số giao thoa IoU với các bounding box còn lại.

Nếu chỉ số này lớn hơn ngưỡng threshold thì điều đó chứng tỏ 2 bounding boxes đang overlap nhau rất cao. Ta sẽ xóa các bounding có có xác xuất thấp hơn và giữ lại bouding box có xác xuất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

2.2. Kiến trúc tổng thể của mô hình YOLOv11

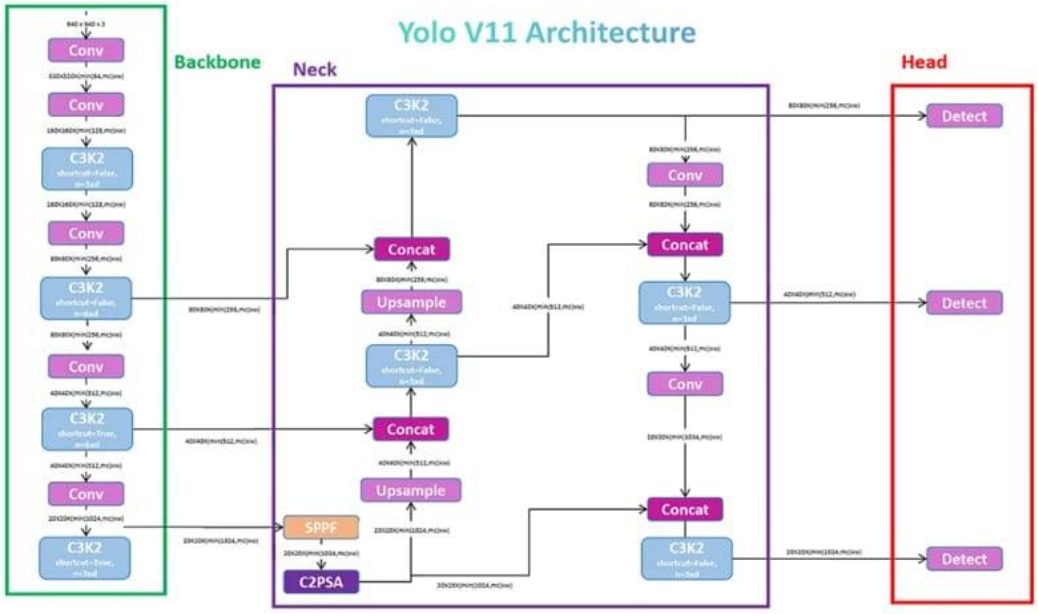
2.2.1. Giới thiệu về YOLOv11 [5]

YOLOv11 là phiên bản mới nhất trong dòng mô hình phát hiện đối tượng thời gian thực YOLO của Ultralytics, tái định nghĩa những gì có thể đạt được với độ chính xác, tốc độ và hiệu suất tiên tiến. Dựa trên những cải tiến ấn tượng từ các phiên bản YOLO trước đó, YOLOv11 giới thiệu những cải tiến đáng kể về kiến trúc và phương pháp huấn luyện, giúp nó trở thành một lựa chọn linh hoạt cho nhiều tác vụ thị giác máy tính khác nhau

2.2.2. Kiến trúc tổng thể

YOLOv11 là phiên bản mới nhất trong họ mô hình YOLO, với kiến trúc được chia thành ba phần chính: **Backbone, Neck** và **Head**, mỗi phần đảm nhận một vai trò riêng biệt trong quá trình xử lý ảnh và phát hiện vật thể.

YOLOv11 được thiết kế để tối ưu hóa cả tốc độ và độ chính xác, dựa trên những tiến bộ được giới thiệu trong các phiên bản YOLO trước đó như YOLOv8, YOLOv9 và YOLOv10. Những cải tiến kiến trúc chính trong YOLOv11 xoay quanh khối C3K2, mô-đun SPPF và khối C2PSA, tất cả đều nâng cao khả năng xử lý thông tin không gian trong khi vẫn duy trì suy luận tốc độ cao.



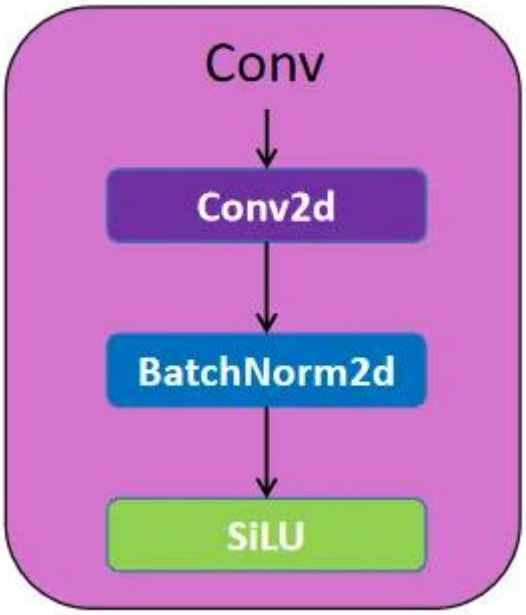
Hình 2.9. Kiến trúc của mô hình YOLOv11 (Ảnh: Analytics Vidhya)

Backbone

Backbone là thành phần cốt lõi trong kiến trúc của YOLOv11, chịu trách nhiệm trích xuất các đặc trưng quan trọng từ ảnh đầu vào. Thông qua việc sử dụng các khối tích chập tiên tiến và khối Bottle Neck, Backbone có khả năng nhận diện hiệu quả các mẫu và chi tiết quan trọng, tạo nền tảng vững chắc cho quá trình phát hiện đối tượng chính xác.

Conv Block

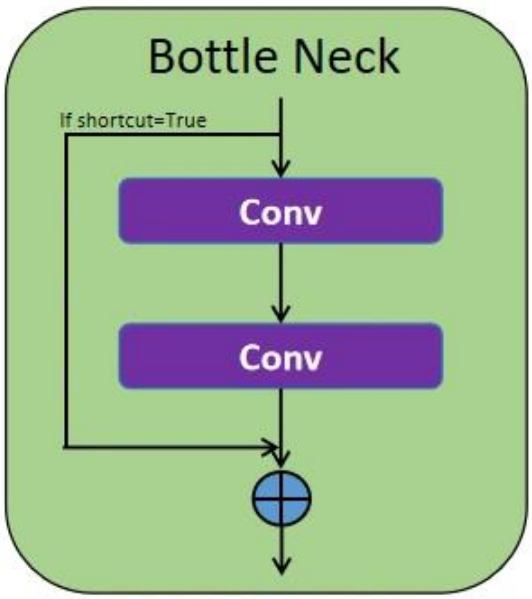
Một trong những thành phần cơ bản của Backbone là khối tích chập (Convolutional Block), hay còn gọi là **Conv Block**, được thiết kế để xử lý đầu vào với kích thước (c, h, w) – tương ứng với số kênh, chiều cao và chiều rộng. Cấu trúc khối này bao gồm ba thành phần chính được sắp xếp theo thứ tự: lớp tích chập 2D (2D Convolutional Layer), lớp chuẩn hóa batch 2D (2D Batch Normalization Layer) và hàm kích hoạt SiLU (SiLU Activation Function). Cấu trúc này giúp mô hình học được các đặc trưng phức tạp một cách hiệu quả, đồng thời tăng cường độ ổn định trong quá trình huấn luyện.



Hình 2.10. Cấu trúc Conv Block (Ảnh: Analytics Vidhya)

Bottle Neck Block

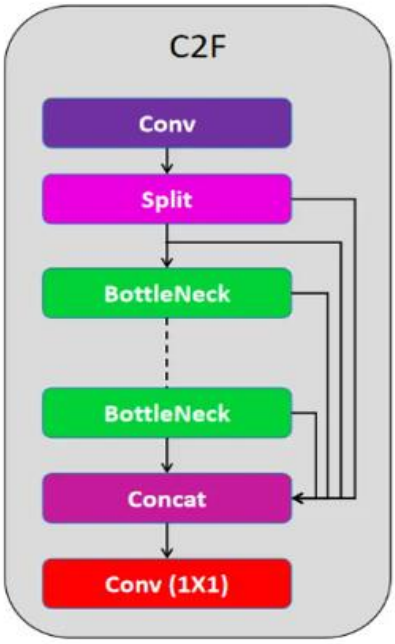
Bottle Neck Block là một chuỗi các khối tích chập được thiết kế kèm theo tham số shortcut, quyết định việc có sử dụng kết nối tắt (residual connection) hay không. Cơ chế residual giúp giảm hiện tượng mất mát thông tin trong quá trình truyền dẫn qua các lớp sâu, đồng thời cải thiện khả năng hội tụ của mô hình.



Hình 2.11. Cấu trúc Bottle Neck Block (Ảnh: Analytics Vidhya)

Khối C2F

Khối C2F (Cross Stage Partial Focus) được kế thừa từ mạng CSP, với trọng tâm là tối ưu hiệu suất và bảo toàn thông tin bản đồ đặc trưng. Cấu trúc của khối này bắt đầu bằng một Conv Block, sau đó đầu ra được chia đôi theo số kênh. Hai nửa này được xử lý qua một chuỗi gồm 'n' lớp Bottle Neck, rồi các đầu ra được ghép nối (concatenate) và đưa qua một Conv Block cuối cùng. Cơ chế này cho phép tăng cường sự kết nối giữa các bản đồ đặc trưng mà không gây dư thừa thông tin, đồng thời tiết kiệm tài nguyên tính toán.



Hình 2.12. Cấu trúc khối C2F (Ảnh: Analytics Vidhya)

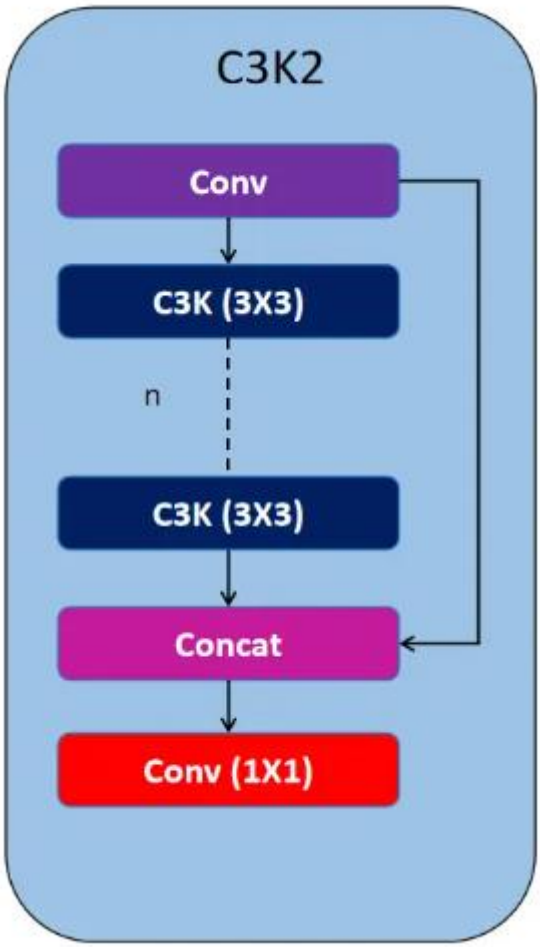
Khối C3K2 và C3K

Khối C3K2

Một điểm nổi bật trong kiến trúc của YOLOv11 chính là việc sử dụng khối C3K2- thành phần cốt lõi của Backbone có nhiệm vụ trích xuất đặc trưng ở các giai đoạn khác nhau. C3K2 được xem là phiên bản cải tiến từ cấu trúc CSP (Cross Stage Partial) bottleneck trong các phiên bản YOLO trước đó, mang lại hiệu quả vượt trội cả về tốc độ và khả năng biểu diễn đặc trưng.

Thay vì sử dụng các kernel lớn, C3K2 áp dụng các phép tích chập với **kernel nhỏ** 3×3, giúp giảm chi phí tính toán trong khi vẫn duy trì được khả năng nhận diện các đặc trưng quan trọng. Cấu trúc này cho phép xử lý các bản đồ đặc trưng nhỏ một cách riêng

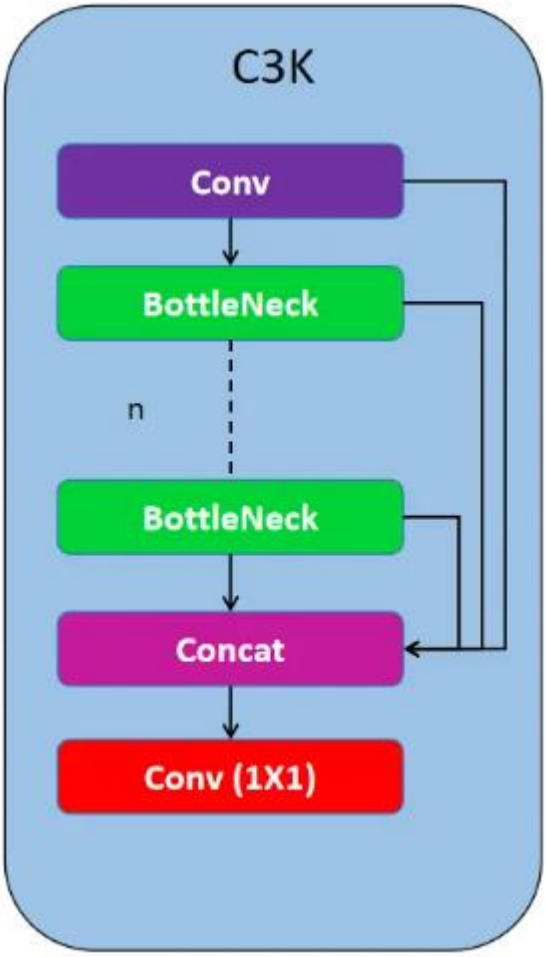
biệt, sau đó ghép nối chúng lại sau nhiều lớp tích chập, giúp mô hình học được nhiều thông tin hơn với số lượng tham số ít hơn so với khối **C2F** trong YOLOv8.



Hình 2.13. Cấu trúc khối C3K2 (Ảnh: Analytics Vidhya)

Khối C3K

Trung tâm của C3K2 là khối C3K, một biến thể đơn giản hóa từ C2F. Khác với C2F chia đầu vào thành hai nhánh, C3K không thực hiện chia tách, mà đưa toàn bộ đầu vào qua một **Conv Block**, tiếp theo là chuỗi các lớp **Bottle Neck** có kết nối ghép nối (concatenation), và kết thúc bằng một **Conv Block** cuối.



Hình 2.14. Cấu trúc khối C3K (Ảnh: Analytics Vidhya)

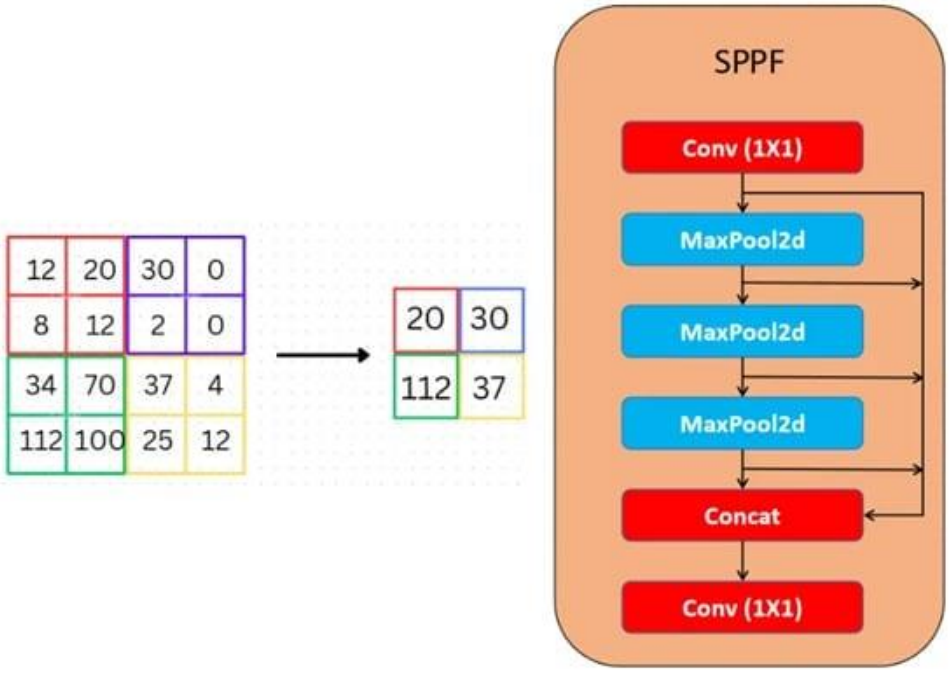
Khối C3K2 được cấu trúc theo hướng tận dụng sức mạnh của C3K: bắt đầu bằng một Conv Block đầu, tiếp nối bởi một chuỗi khối C3K, sau đó ghép nối đầu ra từ Conv Block ban đầu với đầu ra từ khối C3K cuối cùng, và cuối cùng đưa qua một Conv Block kết thúc. Thiết kế này giúp tăng cường luồng thông tin trong mạng, cải thiện khả năng học đặc trưng và cân bằng hiệu quả giữa **tốc độ xử lý** và **độ chính xác** - yếu tố then chốt trong các hệ thống phát hiện thời gian thực như YOLO.

Neck

Trong YOLOv11, phần Neck tiếp tục đóng vai trò quan trọng trong việc kết nối giữa backbone và head, nơi tổng hợp và khuếch đại các đặc trưng quan trọng để cải thiện khả năng phát hiện đối tượng. Hai cải tiến chính trong phần này là việc giữ lại khối **SPPF** (Spatial Pyramid Pooling Fast) và tích hợp cơ chế chú ý không gian thông qua **C2PSA** (Cross Stage Partial with Spatial Attention).

Khối SPPF

Khối SPPF là một phiên bản tối ưu của SPP truyền thống, được thiết kế nhằm trích xuất thông tin từ nhiều vùng ảnh ở các tỉ lệ không gian khác nhau. Bằng cách áp dụng các phép **max pooling với kích thước kernel khác nhau**, SPPF cho phép mô hình **tổng hợp thông tin đa tỉ lệ** hiệu quả mà không làm tăng đáng kể độ trễ. Đặc biệt, điều này giúp cải thiện khả năng phát hiện các đối tượng nhỏ, một thách thức thường gặp ở các phiên bản YOLO trước. Nhờ đó, YOLOv11 vừa duy trì được tốc độ xử lý thời gian thực, vừa nâng cao độ chính xác ở những vùng có độ phân giải thấp.

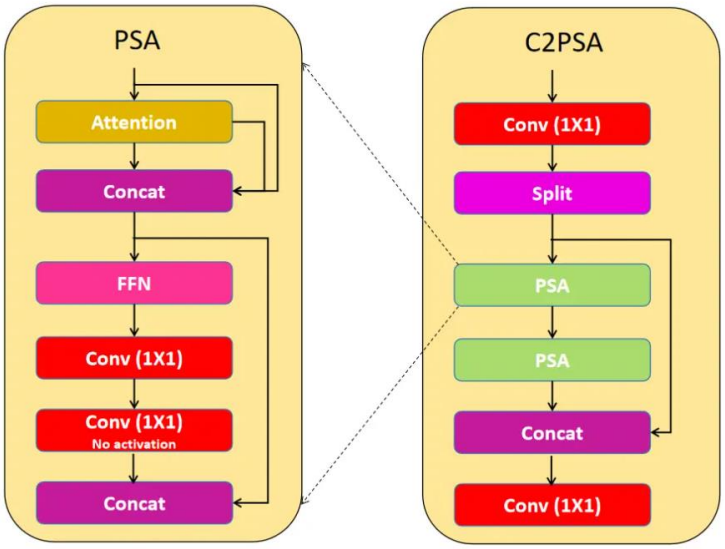


Hình 2.15. Cách thức hoạt động của SPPF (Ảnh: Analytics Vidhya)

Upsampling và C2PSA

Bên cạnh SPPF, YOLOv11 còn tích hợp cơ chế **Upsampling** nhằm đưa các đặc trưng từ tầng sâu lên tầng nông, cho phép mô hình kết hợp thông tin ở nhiều mức độ trừu tượng. Tuy nhiên, điểm đột phá nằm ở sự xuất hiện của khối **C2PSA**, một cấu trúc mới giúp mô hình tập trung mạnh mẽ vào các vùng không gian quan trọng trong ảnh.

Khối **C2PSA** là sự kết hợp giữa thiết kế Cross Stage Partial (CSP) và Position-Sensitive Attention (PSA)**.** Cấu trúc này sử dụng hai nhánh PSA hoạt động song song trên các phần khác nhau của bản đồ đặc trưng, sau đó được nối lại tương tự như cấu trúc C2F. Mỗi nhánh PSA áp dụng cơ chế chú ý theo vị trí, cho phép mô hình làm nổi bật các khu vực quan trọng như các đối tượng nhỏ, bị che khuất hoặc ở xa. Sau khi tổng hợp, kết quả sẽ đi qua một mạng feed-forward, tiếp theo là các khối Conv, rồi được ghép nối lại với đầu vào ban đầu nhằm tăng cường luồng thông tin**.**



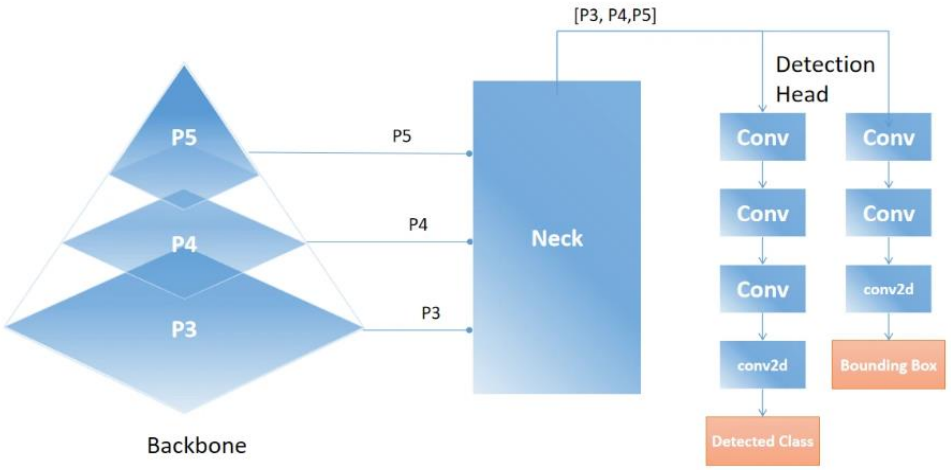
Hình 2.16. Cách thức hoạt động của C2PSA (Ảnh: Analytics Vidhya)

Head

Tương tự các phiên bản trước, YOLOv11 sử dụng kiến trúc **multi-scale prediction** (dự đoán đa tỉ lệ) để phát hiện vật thể ở nhiều kích thước khác nhau.

Phần head của mô hình dựa trên ba tầng đặc trưng P3, P4 và P5, được trích xuất từ backbone và xử lý qua neck, tương ứng với các mức độ chi tiết từ thấp đến cao. Ba tầng này được đưa vào các nhánh dự đoán riêng biệt nhằm tạo ra các hộp giới hạn (bounding box) và phân loại đối tượng.

Nhờ đó, YOLOv11 có thể phát hiện chính xác vật thể nhỏ ở P3 và vật thể lớn ở P5, đảm bảo hiệu quả trong cả độ chính xác lẫn tốc độ xử lý, phù hợp với các ứng dụng thời gian thực.



Hình 2.17. Kiến trúc phần Head trong YOLOv11 (Ảnh: Analytics Vidhya)

2.2.3. Các tính năng chính của YOLOv11

- Trích xuất tính năng nâng cao: Kiến trúc backbone và neck cải tiến giúp phát hiện đối tượng chính xác và xử lý tác vụ phức tạp.
- Tối ưu hiệu quả và tốc độ: Thiết kế kiến trúc và quy trình đào tạo tối ưu, tăng tốc xử lý, cân bằng độ chính xác và hiệu suất.
- Độ chính xác cao với ít tham số: YOLOv11m đạt mAP cao hơn và giảm 22% tham số so với YOLOv8m, tiết kiệm tài nguyên.
- Khả năng thích ứng đa môi trường: Triển khai linh hoạt trên edge, đám mây và hệ thống GPU NVIDIA.
- Hỗ trợ đa tác vụ: Phát hiện, phân đoạn, phân loại, ước tính tư thế và phát hiện đối tượng theo hướng.

2.2.4. So sánh các mô hình YOLOv11

Bảng 2.1. So sánh các mô hình YOLOv11 huấn luyện trên tập COCO

(Nguồn: Ultralytics YOLO Documentation)

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Bảng so sánh cho thấy các mô hình YOLOv11 có sự khác biệt đáng kể về hiệu suất và yêu cầu tài nguyên.

- Mô hình YOLOv11n có kích thước nhỏ nhất, tốc độ suy luận nhanh nhất trên cả CPU và GPU, nhưng độ chính xác (mAP@50–95) thấp nhất, do đó phù hợp với các ứng dụng yêu cầu tốc độ cao trên thiết bị tính toán hạn chế.
- Ngược lại, YOLOv11x đạt độ chính xác cao nhất nhưng yêu cầu dung lượng bộ nhớ lớn và thời gian suy luận dài, chỉ thích hợp với hệ thống có GPU mạnh và không yêu cầu xử lý thời gian thực.
- Trong khi đó, các mô hình trung gian như YOLOv11s và YOLOv11m đạt được sự cân bằng hợp lý giữa độ chính xác và tốc độ xử lý, khiến chúng trở thành lựa chọn phù hợp cho các ứng dụng thời gian thực trên thiết bị có cấu hình trung bình đến cao.

2.2.5. Ứng dụng của YOLOv11

Dù mới ra mắt vào tháng 9/2024, nhưng YOLOv11 đã được ứng dụng rộng rãi trong nhiều lĩnh vực của đời sống nhờ khả năng phát hiện đối tượng nhanh và chính xác. Cụ thể:

- An ninh giám sát: theo dõi hành vi bất thường, nhận diện vũ khí…
- Giao thông: nhận diện biển báo, phát hiện phương tiện vi phạm giao thông…
- Nông nghiệp: giám sát tình trạng cây trồng, phát hiện sâu bệnh, đếm vật nuôi…
- Y tế: phát hiện bất thường trong ảnh X-quang, hỗ trợ chẩn đoán tự động.
- Thương mại và bán lẻ: nhận diện sản phẩm, theo dõi hành vi khách hàng, hỗ trợ kiểm kê và tự động hóa quy trình bán hàng.

CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG

3.1. Giới thiệu bài toán

3.1.1. Đặt vấn đề

Đội mũ bảo hiểm khi đi xe máy là quy định bắt buộc và cực kỳ quan trọng để bảo vệ an toàn, giảm thiểu chấn thương nghiêm trọng khi tai nạn. Đáng tiếc là nhiều người vẫn chưa chấp hành nghiêm chỉnh, dẫn đến nguy cơ tai nạn nghiêm trọng và thiệt hại nặng nề về người và của.

Việc phát hiện hành vi không đội mũ bảo hiểm tự động qua camera giám sát là thách thức lớn do môi trường giao thông phức tạp, điều kiện ánh sáng thay đổi liên tục và mật độ phương tiện đông đúc. Tất cả những yếu tố này gây khó khăn cho việc nhận diện chính xác.

May mắn thay, sự phát triển của công nghệ Deep Learning, đặc biệt là các mô hình phát hiện đối tượng hiện đại như YOLO, đã mang lại một giải pháp vô cùng hiệu quả. Với khả năng nhận diện nhanh chóng và chính xác theo thời gian thực, Deep Learning mở ra cánh cửa tự động hóa quá trình giám sát này. Do đó, xây dựng một hệ thống phát hiện hành vi không đội mũ bảo hiểm dựa trên Deep Learning là rất cần thiết và mang ý nghĩa thực tiễn lớn, giúp hỗ trợ giám sát giao thông và nâng cao ý thức của người dân.

3.1.2. Mục tiêu

Mục tiêu chính của đề tài là xây dựng một hệ thống tự động sử dụng Deep Learning để phát hiện hành vi không đội mũ bảo hiểm khi tham gia giao thông. Hệ thống này không chỉ dừng lại ở việc nhận diện mà còn hướng tới góp phần vào an toàn giao thông, bằng cách:

- Tạo ra một công cụ giám sát hiệu quả: Giúp các cơ quan chức năng dễ dàng hơn trong việc phát hiện và xử lý các trường hợp vi phạm.
- Nâng cao ý thức người dân: Thông qua việc tự động hóa quá trình giám sát, hệ thống sẽ gián tiếp nhắc nhở và khuyến khích người tham gia giao thông tuân thủ luật pháp.
- Ứng dụng công nghệ tiên tiến: Vận dụng Deep Learning để giải quyết một vấn đề thực tiễn, chứng minh khả năng của công nghệ trong việc cải thiện đời sống xã hội.

Các bước cụ thể bao gồm nghiên cứu mô hình, xây dựng dữ liệu, huấn luyện và đánh giá mô hình, sau đó triển khai thành một giải pháp hoàn chỉnh.

3.1.3. Phạm vi đề tài

Đồ án này tập trung vào việc xây dựng một hệ thống phát hiện hành vi không đội mũ bảo hiểm sử dụng Deep Learning, cụ thể là mô hình YOLO, để xử lý hình ảnh và luồng video. Phạm vi thực hiện được giới hạn như sau:

- Mô hình công nghệ: Em sẽ tập trung triển khai và tinh chỉnh một phiên bản cụ thể của YOLO cho bài toán phát hiện hành vi không đội mũ bảo hiểm
- Dữ liệu đầu vào: Hệ thống được thiết kế để xử lý dữ liệu từ camera cố định hoặc video đã quay sẵn trong điều kiện ánh sáng và góc quay tương đối ổn định
- Kết quả đầu ra: Hệ thống sẽ trả về vị trí (bounding box) và nhãn (có mũ/không mũ) của từng người được phát hiện.
- Giới hạn về môi trường: Hiệu suất của hệ thống có thể bị ảnh hưởng bởi các yếu tố ngoại cảnh khắc nghiệt như ánh sáng quá yếu, mưa lớn, hoặc các tình huống che khuất phức tạp.
- Không bao gồm: Đề tài chưa đi sâu vào các vấn đề như nhận diện biển số xe, theo dõi đối tượng liên tục trên nhiều khung hình (tracking), hoặc tích hợp trực tiếp vào hệ thống xử phạt tự động quy mô lớn.

3.2. Thu thập và xử lý dữ liệu

3.2.1. Nguồn dữ liệu

Với các bài toán Deep Learning, dữ liệu đóng vai trò rất quan trọng, ảnh hưởng trực tiếp đến quá trình huấn luyện và đánh giá hiệu quả mô hình. Hiểu được điều đó, trong đề tài "Ứng dụng học sâu trong nhận diện hành vi không đội mũ bảo hiểm khi tham gia giao thông", em đã thu thập dữ liệu ảnh từ nhiều nguồn khác nhau như: thu thập thủ công bằng camera, thu thập từ Internet, và sử dụng các bộ dữ liệu công khai (public datasets) trên các nền tảng trực tuyến. Việc này nhằm đảm bảo dữ liệu đa dạng về góc chụp, điều kiện ánh sáng, độ phân giải và các tình huống thực tế.

Việc kết hợp nhiều nguồn dữ liệu giúp tập dữ liệu trở nên phong phú và đa dạng hơn, từ đó hỗ trợ mô hình nhận diện chính xác trong nhiều điều kiện khác nhau.

Sau khi thu thập, dữ liệu sẽ được xử lý sơ bộ để loại bỏ các ảnh không hợp lệ hoặc không rõ đối tượng và chuyển sang bước gán nhãn và tiền xử lý.

3.2.2. Gán nhãn dữ liệu

Quá trình gán nhãn được thực hiện trên nền tảng Roboflow, đảm bảo tính nhất quán và trực quan trong việc đánh dấu các đối tượng "helmet" và "no_helmet".

Hình 3.1. Ảnh được gán nhãn

Tập dữ liệu của em bao gồm hai nhãn chính: **"helmet"** (đội mũ bảo hiểm) và **"no_helmet"** (không đội mũ bảo hiểm). Trong đó, nhãn "helmet" dùng để chỉ những trường hợp người đi xe máy đội mũ đúng quy định, thường được gán bằng màu xanh. Ngược lại, nhãn "no_helmet" được gán màu đỏ, bao gồm cả những trường hợp không đội mũ, đội mũ cối hay các loại mũ không đạt chuẩn an toàn.

Việc gán nhãn là bước rất quan trọng trong quá trình xử lý dữ liệu vì nó giúp mô hình học được sự khác biệt giữa hai hành vi: an toàn và không an toàn. Nếu dữ liệu được gán nhãn chính xác, mô hình sẽ học tốt hơn và cho kết quả đáng tin cậy hơn. Đây cũng là yếu tố ảnh hưởng trực tiếp đến các chỉ số đánh giá mô hình như accuracy, precision hay recall. Vì vậy, em đã cố gắng gán nhãn thật kỹ và thống nhất để đảm bảo chất lượng huấn luyện mô hình.

3.2.3. Tiền xử lý dữ liệu

Sau khi hoàn tất bước gán nhãn, em tiến hành tiền xử lý dữ liệu với hai thao tác chính. Đầu tiên là tự động xoay ảnh (auto-orient) để đảm bảo tất cả hình ảnh hiển thị đúng hướng, tránh trường hợp ảnh bị xoay lệch do thông tin EXIF. Tiếp theo, toàn bộ ảnh được resize về kích thước 640x640 pixel để đảm bảo tính đồng nhất đầu vào và phù hợp với kiến trúc của mô hình YOLOv11 trong quá trình huấn luyện.

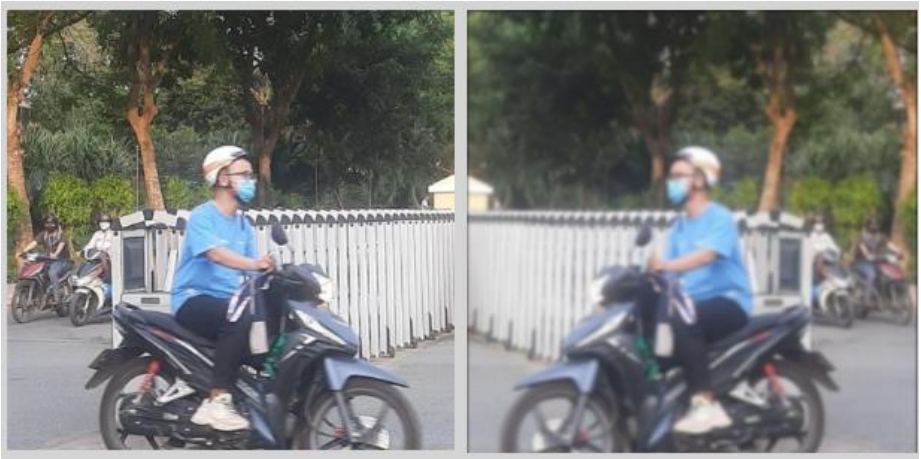
Hình 3.2. Hình ảnh được resize về kích thước 640x640

3.2.4. Tăng cường dữ liệu

Tăng cường dữ liệu (Data Augmentation) là quá trình tạo ra các biến thể mới từ ảnh gốc bằng cách áp dụng nhiều kỹ thuật biến đổi khác nhau.

Mục tiêu chính của bước này là làm phong phú thêm tập dữ liệu huấn luyện và giúp mô hình tránh bị học quá khớp (overfitting). Trong quá trình thực hiện đề tài, em đã áp dụng một số kỹ thuật tăng cường dữ liệu phổ biến như:

- **Lật ảnh (Flip)**: Lật ngang ảnh để mô phỏng các góc nhìn khác nhau, giúp mô hình nhận diện tốt hơn trong các tình huống thực tế.
- **Điều chỉnh độ sáng (Brightness)**: Thay đổi độ sáng của ảnh để mô hình có thể học được trong nhiều điều kiện ánh sáng khác nhau.
- **Làm mờ ảnh (Blur)**: Áp dụng các hiệu ứng làm mờ Gaussian Blur để mô hình làm quen với ảnh bị mất nét, mờ do chuyển động hoặc chất lượng camera kém.
- **Thêm nhiễu (Noise)**: Chèn thêm nhiễu để tăng tính kháng nhiễu và giúp mô hình ổn định hơn khi gặp ảnh có chất lượng thấp.

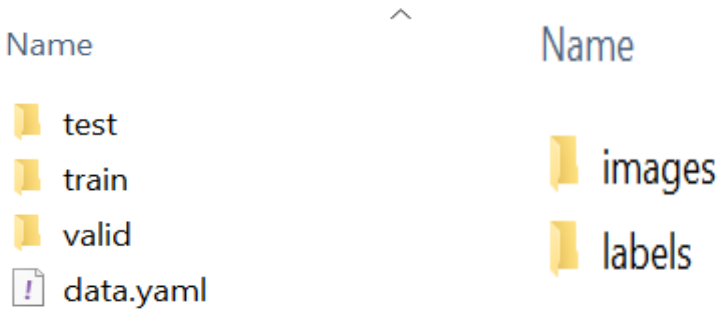


Hình 3.3. Ảnh sau khi sử dụng một số kỹ thuật tăng cường

Việc tăng cường dữ liệu không chỉ giúp cải thiện chất lượng đầu vào mà còn đóng vai trò quan trọng trong việc nâng cao độ chính xác và khả năng tổng quát của mô hình khi triển khai thực tế.

3.2.5. Chia tập dữ liệu

Sau khi toàn bộ hình ảnh được gán nhãn, em thu được một tập dữ liệu gồm 9351 ảnh, được chia thành ba phần: tập huấn luyện (train) chiếm 82% tương đương 7712 ảnh, tập xác thực (validation) chiếm 10% tương đương 899 ảnh, và tập kiểm tra (test) chiếm 8% tương đương 740 ảnh.



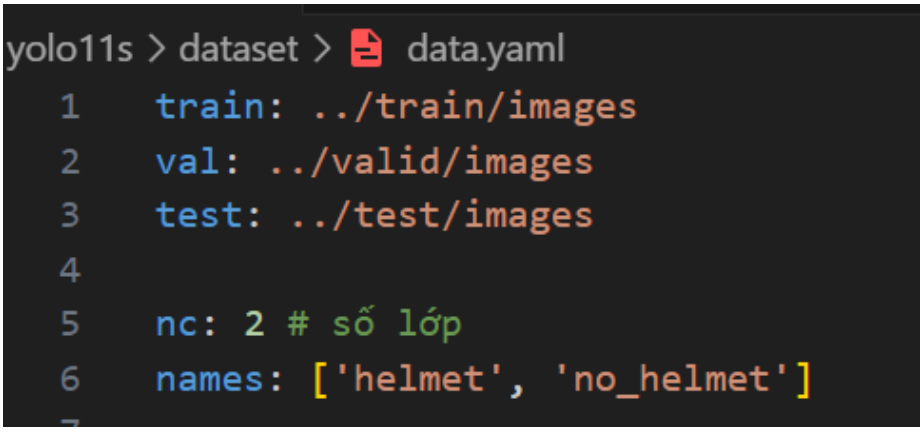
Hình 3.4. Cấu trúc dữ liệu sau gán nhãn

Thư mục images chứa các ảnh đầu vào dùng để huấn luyện mô hình, còn thư mục labels chứa các file nhãn (.txt) tương ứng với từng ảnh. Các file nhãn này được định dạng theo chuẩn YOLO, mỗi dòng trong file có cấu trúc:

```
class_id x_center y_center width height
```

Trong đó class_id là chỉ số đại diện cho lớp (0 : “helmet”, 1 : “no_helmet”), các giá trị còn lại là tọa độ tâm và kích thước của khung giới hạn (bounding box), đã được chuẩn hóa về khoảng [0, 1] theo kích thước gốc của ảnh.

Toàn bộ cấu trúc dữ liệu này được khai báo trong file data.yaml, bao gồm đường dẫn đến các tập train, val, test, số lượng lớp (nc) và danh sách tên các lớp (names).



Hình 3.5. Cấu trúc file data.yaml

3.3. Huấn luyện mô hình

Việc huấn luyện mô hình được thực hiện trên nền tảng **Google Colab** nhằm tận dụng sức mạnh xử lý của GPU miễn phí, giúp rút ngắn đáng kể thời gian đào tạo mô hình. Mô hình được lựa chọn là YOLOv11 – phiên bản cải tiến tiếp theo trong họ mô hình YOLO, được cung cấp bởi thư viện Ultralytics, một framework mạnh mẽ và thân thiện với người dùng trong các bài toán phát hiện đối tượng.

Quá trình huấn luyện được thực hiện theo phương pháp fine-tuning, tức là tinh chỉnh lại một mô hình đã được huấn luyện trước trên một tập dữ liệu lớn (pre-trained model), giúp mô hình thích nghi tốt hơn với tập dữ liệu cụ thể của đề tài.

Các bước của quá trình huấn luyện mô hình:

- Kết nối với Google Colab

```
from google.colab import drive
drive.mount('/content/drive')
```

- Giải nén file đã tải lên từ Google Drive

```
!unzip /content/drive/MyDrive/DATN/yolov11.zip
```

- Cài đặt thư viện **ultralytics**

```
!pip install ultralytics
```

- Huấn luyện mô hình YOLO (ở đây em chọn mô hình YOLO11s)

```
from ultralytics import YOLO

model = YOLO('/content/yolo11s.pt')  # Tải mô hình yolo11s
model.train(data='/content/data.yaml', epochs=50, imgsz=640, batch= 16, save = True)
```

Giải thích

model.train(...): Bắt đầu quá trình huấn luyện mô hình.

data='/content/data.yaml': Dùng bộ dữ liệu được cấu hình trong tệp data.yaml.

epochs=50: Mô hình sẽ được huấn luyện lặp lại 50 lần

imgsz=640: Thay đổi kích thước ảnh đầu vào thành 640x640 pixel.

batch =16: Xử lý 16 ảnh trong mỗi lô (batch).

save=True: Lưu lại các kết quả huấn luyện và trọng số mô hình

- Bắt đầu quá trình huấn luyện

```
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/50      4.37G      1.547      1.709      1.335         44        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all        899       1782      0.733      0.584      0.658      0.371

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/50      4.35G      1.554       1.25       1.34         49        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/50      4.33G       1.55      1.213      1.345         61        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
```

- Kết thúc quá trình huấn luyện

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      48/50      4.32G      0.904     0.4395      1.004         47        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      49/50      4.32G     0.8962     0.4348      1.003         28        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50      4.33G      0.884     0.4273      0.994         33        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%


50 epochs completed in 2.423 hours.
```

Sau khi huấn luyện, mô hình được lưu dưới dạng “best.pt” trong thư mục train, với đường dẫn: “/content/runs/detect/train/weights/best.pt”

- Dự đoán trên tập test

```
from ultralytics import YOLO
model = YOLO('/content/runs/detect/train/weights/best.pt')
model.predict("/content/test/images", show=True)
```

3.4. Đánh giá mô hình

3.4.1. Tiêu chí đánh giá

- Accuracy (Độ chính xác)là tỷ lệ dự đoán đúng trên tổng số dữ liệu, chỉ số này dễ hiểu và phổ biến nhất, nhưng không phù hợp nếu dữ liệu mất cân bằng
- Precision (Độ chính xác của dự đoán): là tỷ lệ giữa số dự đoán đúng thuộc lớp dương (positive) trên tổng số dự đoán thuộc lớp dương.
- Recall (Độ bao phủ): là tỷ lệ giữa số dự đoán đúng thuộc lớp dương (positive) trên tổng số thực tế thuộc lớp dương**.**
- F1-score: là trung bình điều hòa giữa Precision và Recall, giúp cân bằng khi hai chỉ số này không đều nhau
- IoU (Intersection over Union): dùng trong phát hiện đối tượng. Là tỷ lệ diện tích vùng giao giữa bounding box dự đoán và bounding box thực tế. Nếu IoU lớn hơn 0.5 (hoặc ngưỡng cho trước) thì được tính là phát hiện đúng.
- mAP (mean Average Precision): là chỉ số tổng hợp trong object detection, đo độ chính xác trung bình của mô hình trên tất cả các lớp và các ngưỡng IoU khác nhau (thường là mAP@0.5 và mAP@0.5:0.95).
- Inference Time (Thời gian suy luận): là thời gian để mô hình xử lý một mẫu đầu vào (ví dụ: một ảnh). Với các mô hình ứng dụng thời gian thực, thời gian suy luận càng thấp càng tốt
- Throughput (Thông lượng): là số lượng ảnh (hoặc mẫu) mà mô hình có thể xử lý trong 1 giây**,** được đo bằng FPS. Nếu xử lý thời gian thực, thông lượng càng cao càng tốt.

3.4.2. So sánh kết quả đánh giá từ mô hình

Sau quá trình huấn luyện và đánh giá trên bộ dữ liệu mũ bảo hiểm, em đã thu được những thông số hiệu suất quan trọng của các mô hình YOLO và tổng hợp trong bảng sau:

Bảng 3.1. Bảng so sánh hiệu suất các mô hình YOLO

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95	Inference Time (ms)	FPS (ước tính)
YOLOv11s	**0.9**	0.87	**0.93**	**0.66**	9.44	106
YOLOv11n	0.87	**0.88**	0.92	0.64	**4.57**	**219**
YOLOv8n	0.88	0.78	0.85	0.45	5.41	185
YOLOv8s	0.89	0.81	0.86	0.48	7.5	133
YOLOv12n	0.82	0.77	0.83	0.44	5.27	190

Từ bảng kết quả, có thể thấy rõ sự vượt trội của các phiên bản YOLOv11 so với YOLOv8 và YOLOv12n trong bài toán này. Cụ thể:

- YOLOv11s đạt độ chính xác phát hiện cao nhất (Precision = 0.9, mAP@0.5 = 0.93, mAP@0.5:0.95 = 0.66), rất phù hợp nếu yêu cầu về độ chính xác là ưu tiên hàng đầu.
- YOLOv11n thể hiện sự cân bằng tối ưu giữa độ chính xác và tốc độ, với Recall cao nhất (0.88) và đặc biệt là thời gian suy luận nhanh nhất (4.57 ms), cho phép xử lý 219 khung hình mỗi giây (FPS), lý tưởng cho các ứng dụng thời gian thực.
- Các phiên bản YOLOv8 và YOLOv12n dù có tốc độ khá tốt nhưng hiệu suất về các chỉ số chính xác (mAP) thấp hơn đáng kể so với YOLOv11 trên bộ dữ liệu được sử dụng.

3.5. Triển khai mô hình

3.5.1. Thư viện và những hàm quan trọng

Để triển khai mô hình, em có sử dụng python và các thư viện như:

- Open CV, PIL: để xử lý ảnh và video đầu vào
- Numpy: để xử lý mảng và các phép toán số học
- Ultralytics: để tải mô hình YOLO
- Và thư viện Streamlit để tạo giao diện người dùng đơn giản

Một vài hàm quan trọng

```
# Load model
@st.cache_resource
def load_model():
    with st.spinner("🚀 Đang tải mô hình YOLO..."):
        return YOLO(r"D:\Helmet_Detection\models\bestyolo.onnx")

model = load_model()
```

Hình 3.6. Hàm Load model

```python
# Vẽ bounding box
def draw_boxes(image, results, actual_fps=None, font_scale_base=0.5):
    class_names = results.names
    boxes = results.boxes
    stats = {'total': 0, 'helmet': 0, 'no_helmet': 0, 'confidences': []}

    frame_height, frame_width, _ = image.shape
    font_scale = font_scale_base * (frame_width / 640)
    thickness = max(1, int(frame_width / 640 * 2))

    for box in boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        conf = float(box.conf[0])
        cls_id = int(box.cls[0])
        label = class_names[cls_id]

        color = (0, 255, 0) if label == 'helmet' else (0, 0, 255)
        cv2.rectangle(image, (x1, y1), (x2, y2), color, thickness)
```

Hình 3.7. Hàm vẽ bounding box

```python
# Xử lý hình ảnh
def process_image(image, confidence_threshold, iou_threshold):
    with st.spinner("🔍 Đang phân tích hình ảnh..."):
        # Truyền ngưỡng tin cậy và IoU cho mô hình
        results = model(image, conf=confidence_threshold, iou=iou_threshold, verbose=False)[0]
        image_bgr = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        annotated_image, stats = draw_boxes(image_bgr, results)
        return cv2.cvtColor(annotated_image, cv2.COLOR_BGR2RGB), stats
```

Hình 3.8. Hàm xử lý ảnh khi truyền vào

```python
# Xử lý Video
def process_video(video_path, confidence_threshold, iou_threshold, skip_frames=5):
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        st.error("Không mở được video. Vui lòng kiểm tra file.")
        return None

    stframe = st.empty()
    progress_bar = st.progress(0)
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    frame_count = 0

    status_text = st.empty()
    status_text.info(f"Đang xử lý video ({total_frames} frames)...")

    stats = {
        'total_frames': total_frames,
        'processed_frames': 0,
        'helmet_counts': [],
        'no_helmet_counts': [],
        'fps_list': [],
        'start_time': datetime.now()
    }

    last_display_time = time.time()

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame_count += 1
```

Hình 3.9. Hàm xử lý video

```
st.subheader("📊 Thống kê hình ảnh")
cols = st.columns(4)
with cols[0]:
    st.metric("Tổng đối tượng", stats['total'])
    st.markdown('</div>', unsafe_allow_html=True)
with cols[1]:
    st.metric("Có mũ", stats['helmet'], delta_color="off")
    st.markdown('</div>', unsafe_allow_html=True)
with cols[2]:
    st.metric("Không mũ", stats['no_helmet'], delta_color="off")
    st.markdown('</div>', unsafe_allow_html=True)
with cols[3]:
    st.metric(
        "Độ an toàn",
        f"{(stats['helmet'] / stats['total']) * 100:.1f}%"
        if 'helmet' in stats and 'total' in stats and stats['total'] > 0
        else "N/A"
    )
    st.markdown('</div>', unsafe_allow_html=True)

st.session_state.report_data.append({
    'Thời gian': datetime.now(),
    'Loại': 'Ảnh',
    'Tổng đối tượng': stats['total'],
    'Có mũ': stats['helmet'],
    'Không mũ': stats['no_helmet'],
    'Độ an toàn': f"{(stats['helmet'] / stats['total']) * 100:.1f}%"
        if 'helmet' in stats and 'total' in stats and stats['total'] > 0
        else "N/A"
})
```

Hình 3.10. Hàm thống kê khi người dùng tải ảnh lên

3.5.2. Triển khai ứng dụng thực tế

Dựa trên kết quả đánh giá chi tiết từ Bảng 3.1, mô hình YOLOv11n đã được lựa chọn để triển khai cho ứng dụng thực tế. Quyết định này dựa trên sự cân bằng lý tưởng giữa độ chính xác cao và tốc độ suy luận vượt trội, đáp ứng yêu cầu về hiệu năng thời gian thực của bài toán phát hiện hành vi không đội mũ bảo hiểm.

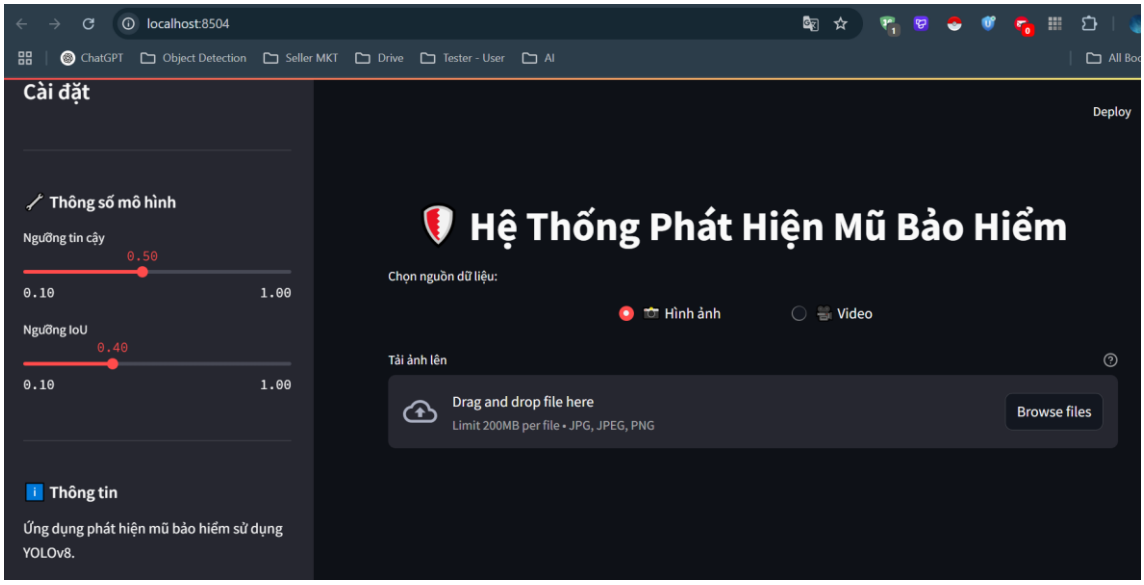
Mô tả hoạt động và chức năng

Mô tả cách thức hoạt động trực quan của mô hình thông qua việc triển khai lên môi trường web bằng Streamlit. Người dùng có thể cung cấp ảnh tĩnh hoặc video từ tệp làm đầu vào. Mô hình sẽ tự động xử lý, phát hiện và định vị người điều khiển phương tiện.

Kết quả được hiển thị bằng cách vẽ các hộp giới hạn quanh đầu, kèm theo nhãn phân loại (“helmet” hoặc “no_helmet”) và độ tin cậy. Để tăng tính trực quan, các hộp giới hạn sẽ được tô màu khác nhau (ví dụ: xanh cho “helmet”, đỏ cho “no_helmet”)

Triển khai ứng dụng

Giao diện web



Hình 3.11. Giao diện web của mô hình

Giao diện của mô hình có các chức năng chính: cho phép người dùng tải ảnh (ở định dạng jpg, jpeg, png) hoặc video muốn nhận diện về giao thông

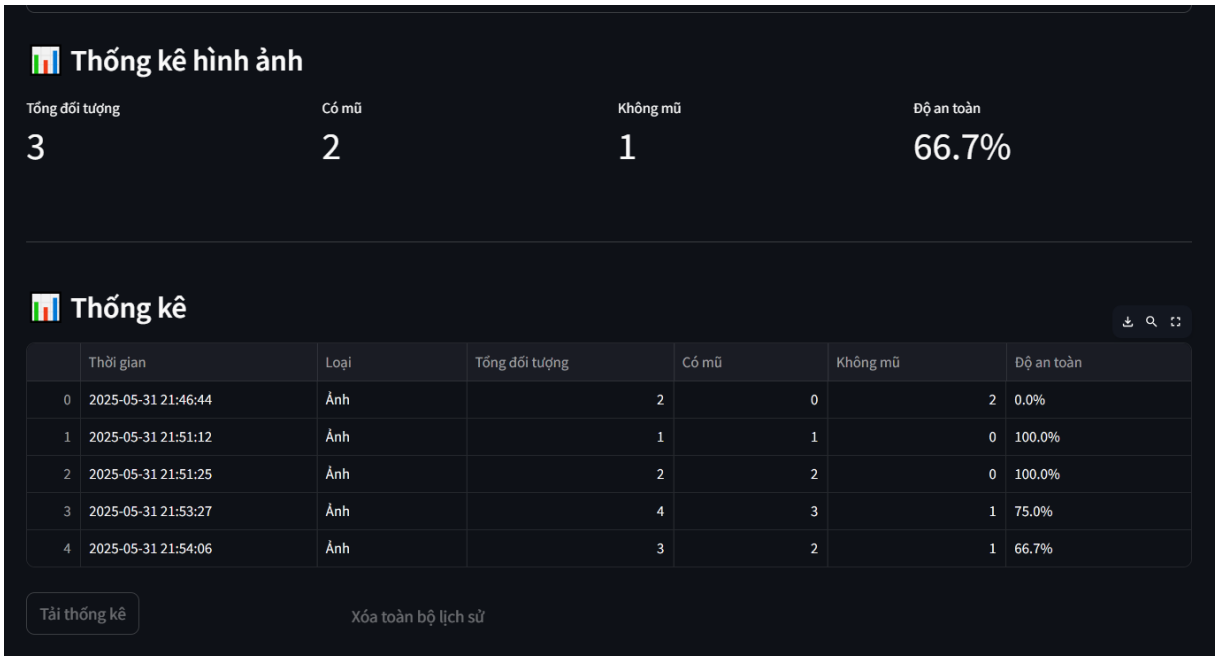
Ngoài ra, còn có chức năng cài đặt, cho phép người dùng có thể điều chỉnh các ngưỡng tin cậy và ngưỡng IoU tùy thích

Khi người dùng tải ảnh

Hình 3.12. Giao diện sau khi người dùng tải ảnh lên

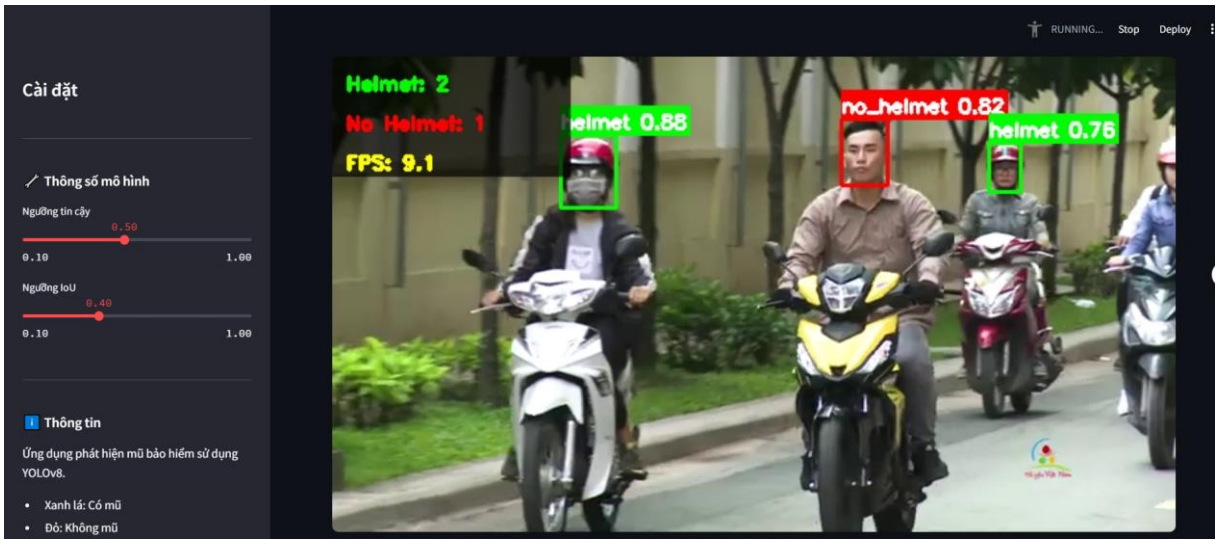
Khi người dùng tải ảnh lên, mô hình sẽ tiến hành phát hiện số người đội mũ và không đội mũ trong khung hình bằng cách tạo các bounding box trên đầu của đối tượng kèm tỉ lệ dự đoán.

Bên dưới là bảng thống kê kết quả mà mô hình đã phát hiện được từ ảnh của người dùng, cho phép người dùng xem và tải kết quả dạng file excel



Hình 3.13. Kết quả thống kê khi phát hiện ảnh

Khi người dùng tải video



Hình 3.14. Giao diện sau khi người dùng tải video lên

Khi người dùng tải video lên, mô hình sẽ tự động phát hiện số người đội mũ và không đội mũ trong khung hình bằng cách tạo các bounding box trên đầu của đối tượng kèm tỉ lệ dự đoán:

- “helmet” - màu xanh (đối tượng có đội mũ bảo hiểm)
- “no_helmet” – màu đỏ (đối tượng không đội mũ bảo hiểm)

Ngoài ra, bên góc trái là thống kê về số: helmet, no_helmet và số khung hình xử lý trong một giây của mô hình (FPS)

KẾT LUẬN

Quãng thời gian thực hiện đề tài đã trang bị, củng cố cho em thêm nhiều kiến thức về công nghệ đặc biệt là về AI, Deep Learning, giúp em có cơ hội được nghiên cứu, tìm hiểu một số thuật toán liên quan đến bài toán mà mình đang thực hiện. Ngoài ra nhờ sự hướng dẫn, góp ý của thầy và sự giúp đỡ của các bạn trong nhóm, em đã hoàn thành đề tài "Ứng dụng học sâu trong nhận diện hành vi không đội mũ bảo hiểm khi tham gia giao thông" với tất cả sự cố gắng của mình.

Bên cạnh những thành quả đạt được, ứng dụng của em còn tồn tại những hạn chế nhất định như: tốc độ tải mô hình còn chậm, FPS chưa cao (8-10 FPS), tỉ lệ phát hiện chưa đáp ứng được kỳ vọng dẫn tới chưa thể kết nối với cơ sở dữ liệu. Ngoài ra, mô hình mới chỉ ở dạng phân loại: "có" hoặc "không" đội mũ bảo hiểm, chưa có cơ chế nhận diện biển số xe, hay phát ra tín hiệu nếu phát hiện vi phạm.

Dù vậy, đây vẫn là một đề tài hay, có ích cho xã hội và cộng đồng. Trong tương lai, nếu có cơ hội em sẽ phát triển thêm các tính năng: nhận diện biển số xe, phát tín hiệu khi nhận thấy dấu hiệu vi phạm để tạo ra một ứng dụng có ích, góp phần nâng cao nhận thức và tuyên truyền ý nghĩa của việc đội mũ bảo hiểm khi tham gia giao thông đến tất cả mọi người.

TÀI LIỆU THAM KHẢO

[1] Số vụ tai nạn tại Việt Nam năm 2024:

https://baotintuc.vn/xa-hoi/10944-nguoi-chet-vi-tai-nan-giao-thong-trong-nam-2024-20250106103121152.htm

[2] Báo cáo "Helmets – A road safety manual for decision-makers and practitioners" của WHO, xuất bản năm 2006, trang 40:

https://iris.who.int/bitstream/handle/10665/43261/9241562994_eng.pdf

[3] Redmon et al., “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016

https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

[4] Giới thiệu về YOLO

https://phamdinhkhanh.github.io/2020/03/09/DarknetAlgorithm.html

[5] Kiến trúc YOLOv11

https://www.analyticsvidhya.com/blog/2024/10/yolov11-object-detection/