



Olivia Wilson

# PBL5: NHÀ XE THÔNG MINH

**GVHD: TS. Trịnh Công Duy**

**Sinh viên thực hiện:**

**Nguyễn Quốc Anh : 102220003**

**Bùi Chí Cường : 102220008**

**Nguyễn Quốc Tiến : 102220043**

...

# Tổng quan dự án



Hiện nay, tại nhiều gia đình, khu trọ, hay nhà ở dân sinh, việc quản lý khu vực để xe vẫn chủ yếu được thực hiện theo phương pháp thủ công, phụ thuộc vào sự giám sát của con người như khóa tay, ghi chú giấy hoặc camera thông thường mà không có tích hợp thông minh. Điều này dẫn đến nhiều bất cập như thiếu tính bảo mật, khó kiểm soát ra vào, và không có khả năng giám sát từ xa hoặc cảnh báo kịp thời khi có sự cố.

Trong khi đó, trên thế giới, các mô hình nhà thông minh (Smart Home) đang ngày càng phát triển và được ứng dụng rộng rãi, từ điều khiển thiết bị điện, an ninh, cho đến tự động hóa sinh hoạt hàng ngày. Tuy nhiên, việc triển khai hệ thống thông minh trong gara để xe một phần thiết yếu của nhà ở vẫn chưa phổ biến, đặc biệt là tại Việt Nam. Lý do chủ yếu đến từ chi phí đầu tư cao, hạ tầng kỹ thuật phức tạp, và thiếu sự đơn giản trong triển khai cho người dùng không chuyên.

Trong bối cảnh đó, Raspberry Pi nổi lên như một giải pháp lý tưởng để xây dựng các hệ thống thông minh quy mô nhỏ. Đây là một dòng máy tính nhúng mini, có chi phí thấp, tiêu thụ điện năng ít, dễ lập trình, và đặc biệt là có khả năng tích hợp mạnh mẽ với các mô hình trí tuệ nhân tạo (AI), bao gồm nhận diện khuôn mặt, xử lý hình ảnh, nhận dạng giọng nói, ...

## Vấn đề

## Giải pháp đề xuất

### Phần cứng

- Raspberry Pi 4
- Arduino Uno R3
- Camera Raspberry V1.3
- Driver ULN2003
- L298N

### Nhận diện con người

- Thử nghiệm với các mô hình YOLO, Fast - CNN, ....

### Nhận diện mặt người

- Thử nghiệm với các mô hình MobileNetv1, MobileNetv2, DeepFace, MobileFaceNet, , Face\_recognition, ...

### Chuyển lời nói thành văn bản

- Thử nghiệm với các mô hình: Open AI
- Whisper, VinAI PhoWhisper,..
- Thử nghiệm với Google Cloud Speech API

### Ứng dụng

- Xây dựng Website quản lý người dùng
- Kết nối với Arduino để gởi kết quả cũng như trạng thái xử lý

### Server

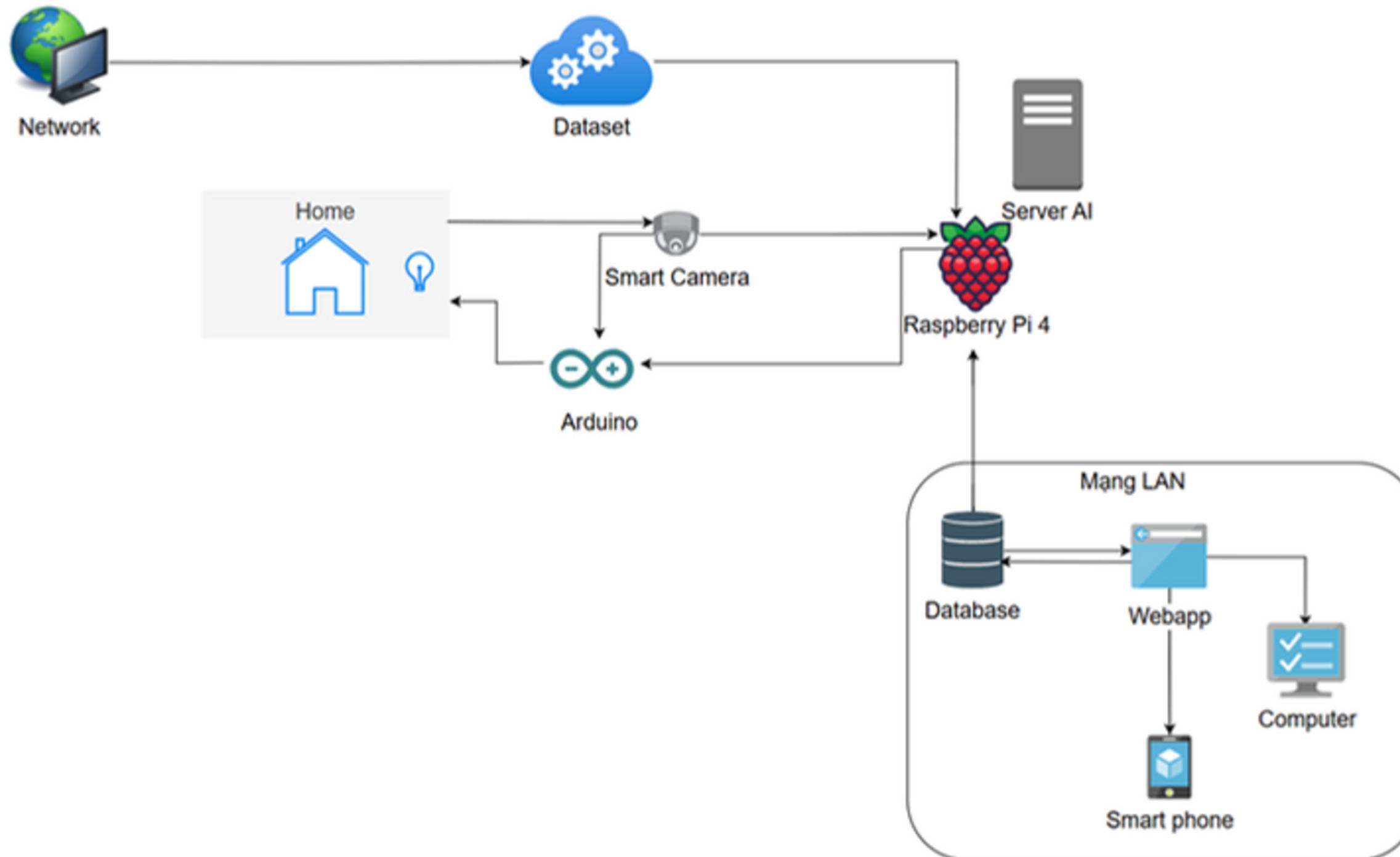
- Viết bằng FastAPI

## BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên thực hiện	Các nhiệm vụ	Đánh giá
Nguyễn Quốc Anh	Thiết kế mô hình IOT	Hoàn thành
	Nhận diện gương mặt bằng Face Recognition	Hoàn thành
	Xây dựng Website quản lý gara	Hoàn thành
	Xây dựng thuật toán điều khiển phần cứng	Hoàn thành
	Triển khai mô hình AI	Hoàn thành
	Lắp ráp phần cứng	Hoàn thành
Nguyễn Quốc Tiến	Thiết kế mô hình IOT	Hoàn thành
	Nhận diện người bằng YOLO	Hoàn thành
	Xây dựng Website quản lý gara	Hoàn thành
	Xây dựng thuật toán điều khiển phần cứng	Hoàn thành
	Triển khai mô hình AI	Hoàn thành
	Lắp ráp phần cứng	Hoàn thành
Bùi Chí Cường	Thiết kế mô hình IOT	Hoàn thành
	Nhận diện giọng nói bằng Whisper	Hoàn thành
	Xây dựng Website quản lý gara	Hoàn thành
	Xây dựng thuật toán điều khiển phần cứng	Hoàn thành
	Triển khai kết nối không dây	Hoàn thành
	Lắp ráp phần cứng	Hoàn thành

# Sơ đồ tổng quan hệ thống

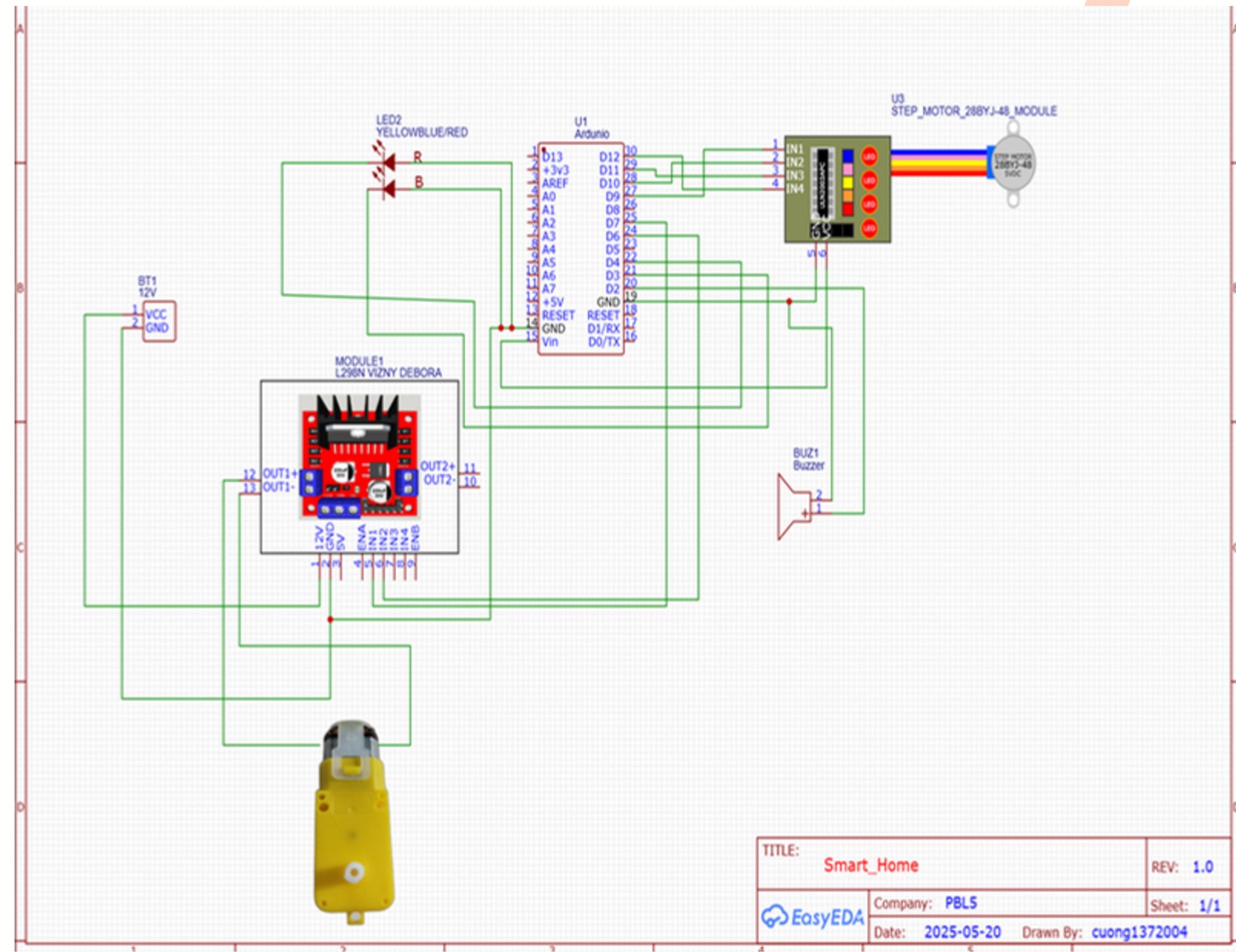
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9



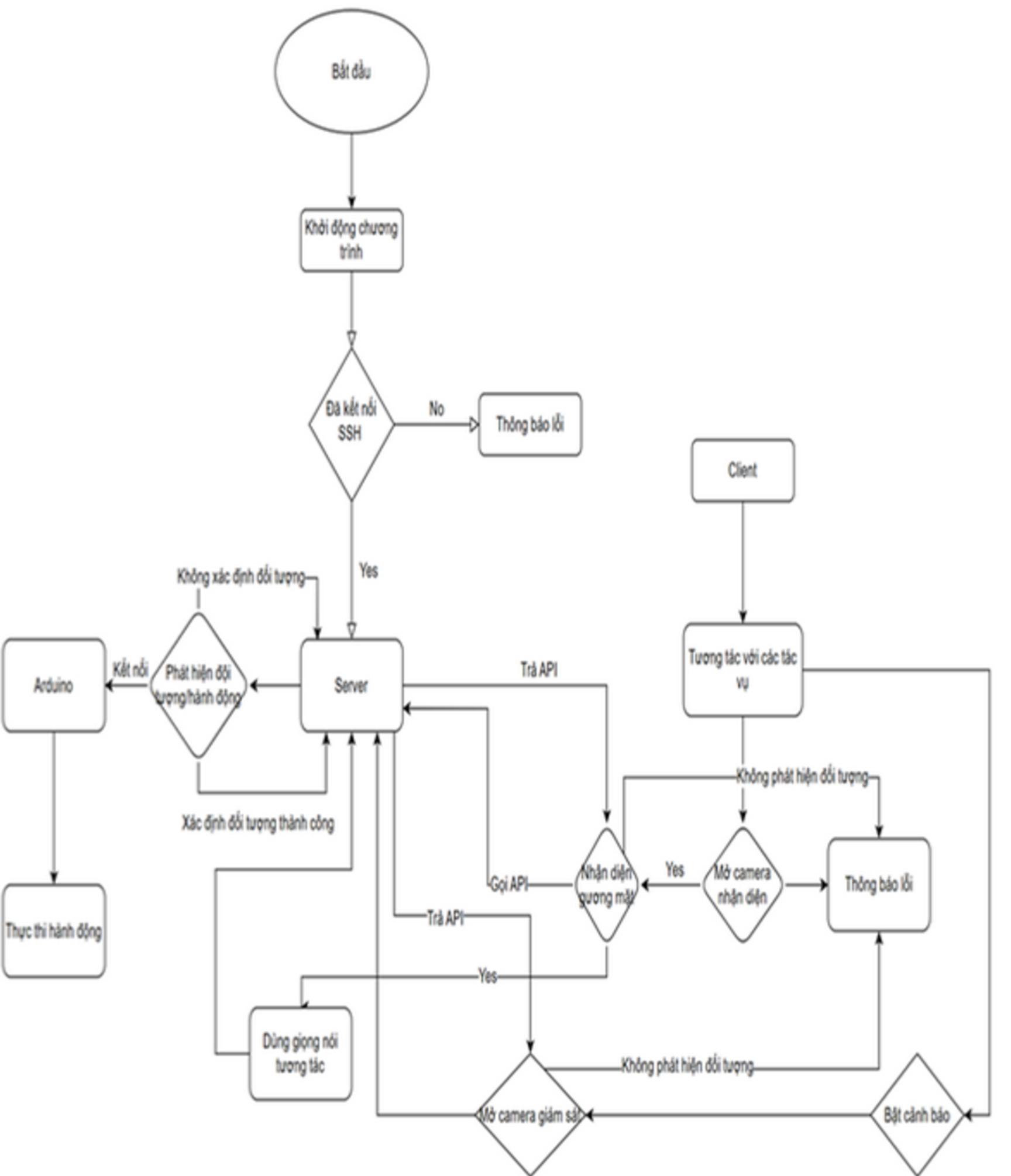
...

# Sơ đồ lắp mạch

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9



# Sơ đồ khối hệ thống



# Các linh kiện phần cứng

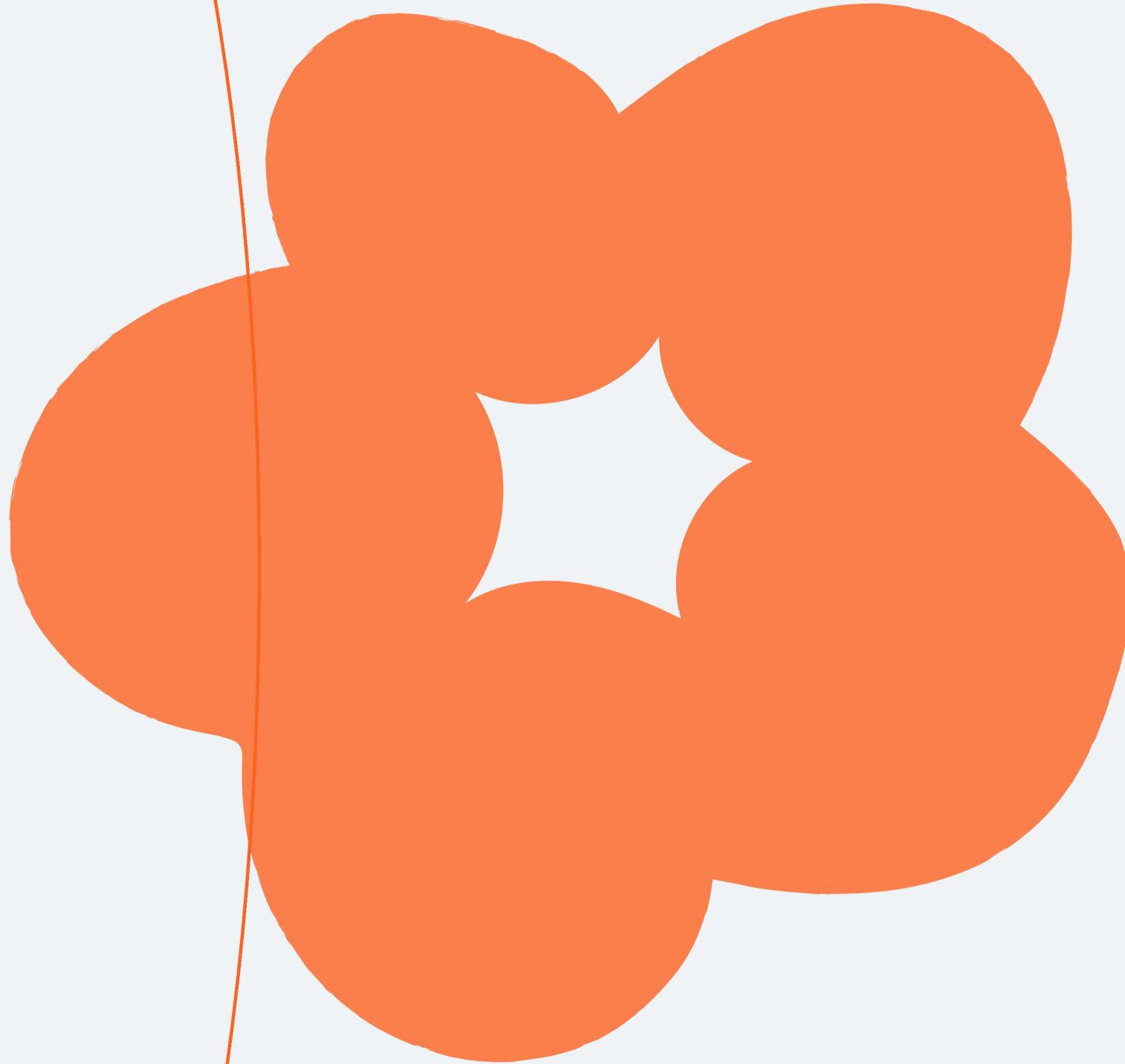
---

- Raspberry Pi 4 Model B 2019 – 4GB RAM
- Camera Raspberry Pi 5MP 1080P Version 1.3
- Mạch Arduino UNO R3
- Driver ULN2003
- Động cơ bước 28byj-48-5v
- L298N
- Động Cơ DC Giảm Tốc Vàng
- Nguồn 12v
- Thiết bị khác: đèn, dây, buzzer
- Sắt, nhựa Mica,....

# Tóm tắt thành phần và chức năng linh kiện

Thành phần	Thiết bị	Chức năng
Phát hiện người	Raspberry Pi 4 + Camera	Phát hiện người
Báo động	Buzzer + Đèn Led	Đưa tín hiệu cảnh báo
Xử lý ảnh	Raspberry Pi 4	Nhận diện đối tượng
Mở cửa cuốn	Driver ULN2003 Động cơ bước 28byj-48-5v	Mở cửa và đóng cửa cuốn khi nhận diện gương mặt
Mở cửa kéo	L298N Động Cơ DC Giảm Tốc Vàng	Mở cửa và đóng cửa cuốn khi nhận lệnh giọng nói

# Mô hình nhận diện khuôn mặt



## Mô hình Face\_recognition

---

Thuận lợi: Không cần huấn luyện lại mô hình từ đầu. Chỉ cần cung cấp dữ liệu theo thư mục: mỗi người một thư mục ảnh. Tương thích với nhiều mô hình nhẹ.

Mô hình Face Recognition sử dụng mạng học sâu đã được huấn luyện sẵn (pre-trained) để trích xuất đặc trưng khuôn mặt (face embeddings) từ ảnh đầu vào. Các đặc trưng này sau đó được so sánh để xác định danh tính người dùng.

# Kiến trúc mô hình Resnet-34

Block	Layer Configuration	Output Size
Conv1	$7 \times 7$ , 64 filters, stride 2	$112 \times 112 \times 64$
MaxPool	$3 \times 3$ , stride 2	$56 \times 56 \times 64$
Conv2_x	$3 \times 3 \times 3$ residual blocks	$56 \times 56 \times 64$
Conv3_x	$3 \times 3 \times 4$ residual blocks	$28 \times 28 \times 128$
Conv4_x	$3 \times 3 \times 6$ residual blocks	$14 \times 14 \times 256$
Conv5_x	$3 \times 3 \times 3$ residual blocks	$7 \times 7 \times 512$
AvgPool	Global average pooling	$1 \times 1 \times 512$
FC	Fully connected layer	128-d vector

Bảng 4. Mô hình Resnet-34

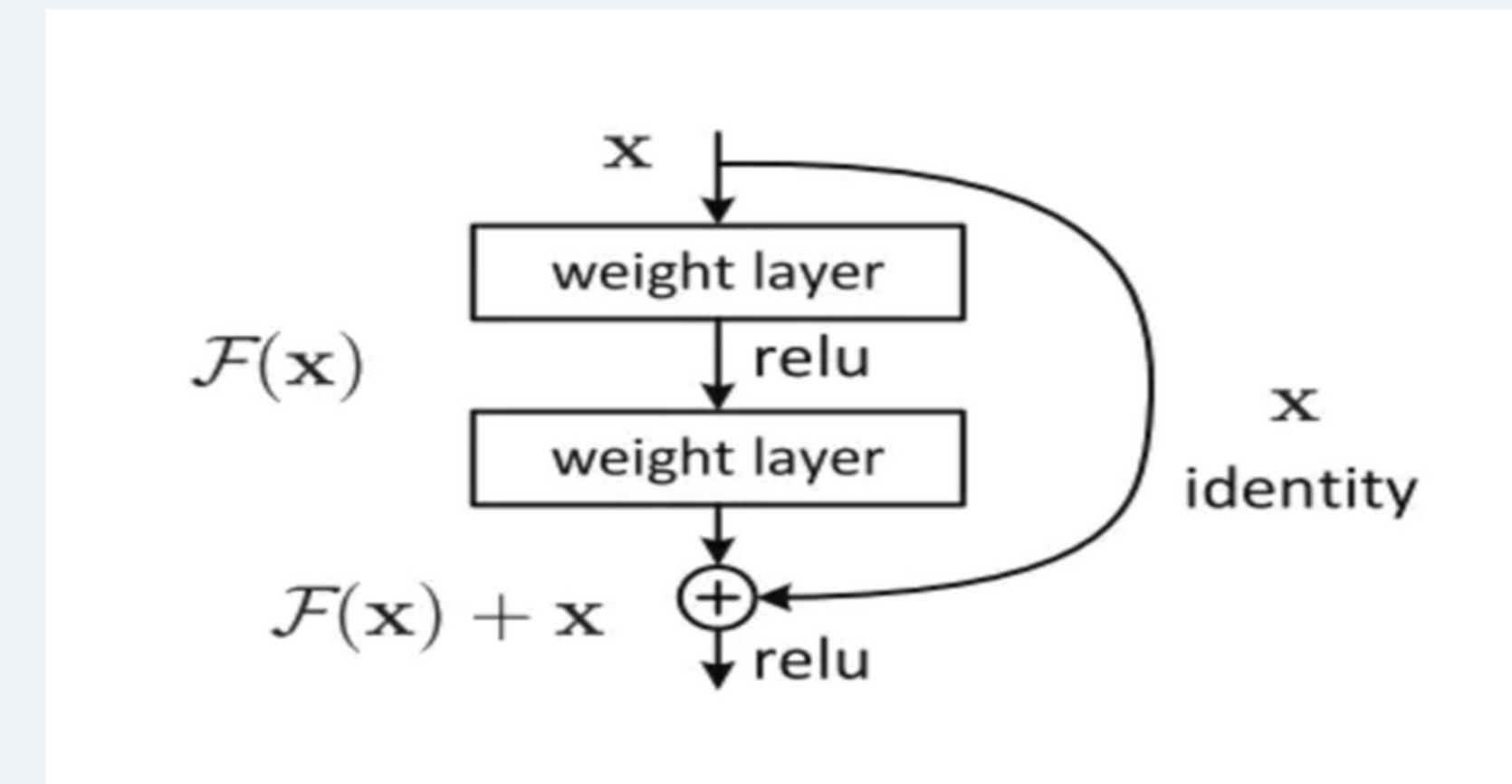
# Khối residual connections (skip connections)

Cách hoạt động chi tiết

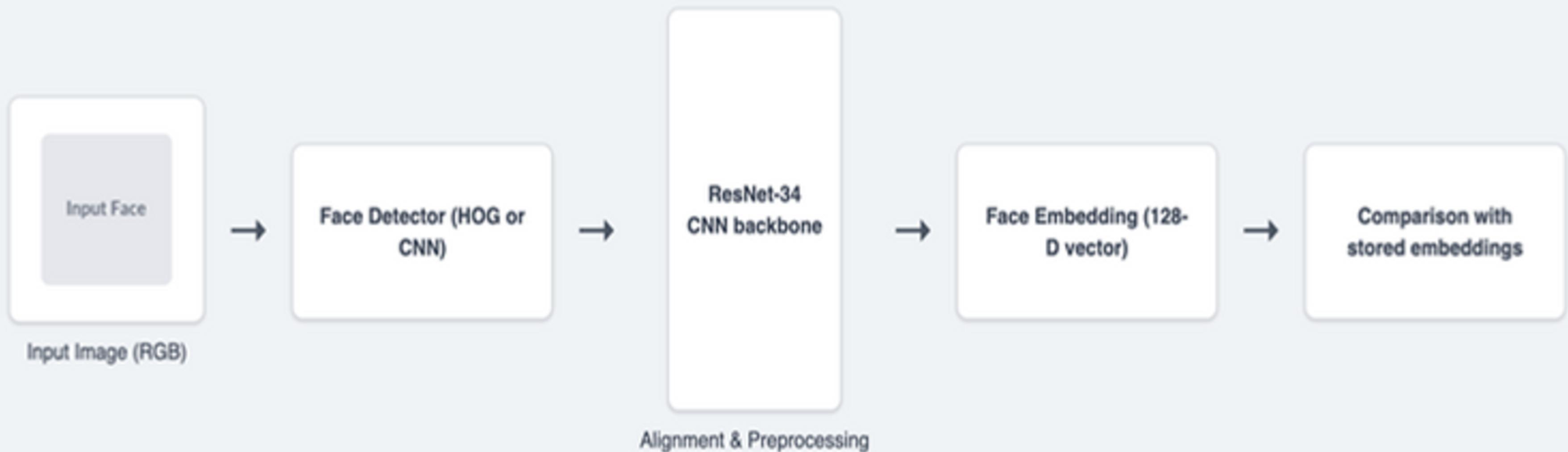
1. Đầu vào  $x$  đi vào 2 lớp convolution (cùng với Batch Normalization và ReLU).
2. Kết quả đầu ra của 2 lớp conv gọi là  $F(x)$  (hàm ánh xạ mà block muốn học).
3. Đầu vào ban đầu  $x$  được cộng trực tiếp vào  $F(x)$ , tức là  $y = F(x) + x$ .
4. Kết quả cộng  $y$  được đưa qua hàm kích hoạt ReLU (hoặc một số trường hợp khác).

Ý nghĩa của phép cộng  $F(x) + x$

- Cho phép mạng học được phần sai số (residual) so với đầu vào ban đầu.
- Nếu block không học gì (ví dụ  $F(x) \approx 0$ ), thì output  $y \approx x$  thông tin không bị mất đi.
- Việc này giúp gradient trong quá trình huấn luyện có thể truyền trực tiếp qua các block, tránh hiện tượng biến mất gradient.
- Khi đi qua nhiều lớp, nếu không có phép cộng này, thông tin gốc có thể bị biến dạng hoặc mất dần.
- Với phép cộng, thông tin đầu vào luôn được bảo toàn trong quá trình lan truyền, tránh mất mát quan trọng.



# Quy trình nhận diện



# Mô hình phát hiện người

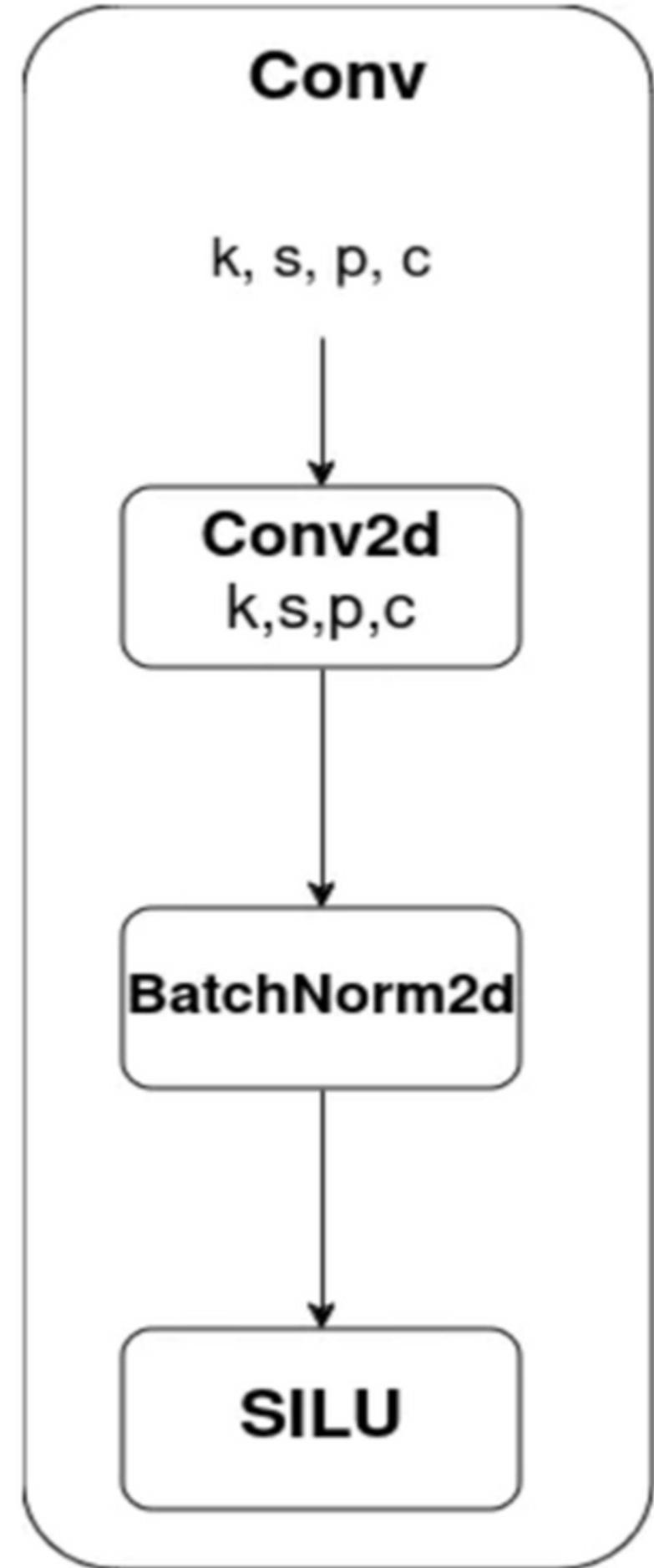
# Mô hình phát hiện người Yolov8n

---

YOLOv8n (YOLOv8 Nano) là một biến thể nhẹ nhất trong dòng mô hình YOLOv8 do Ultralytics phát triển, được thiết kế tối ưu cho thiết bị có tài nguyên hạn chế như điện thoại, Raspberry Pi, hoặc khi cần tốc độ suy luận cao. YOLOv8n kế thừa các cải tiến của dòng YOLO hiện đại như:

- Anchor-free (không dùng anchor box)
- Kiến trúc C2f + Bottleneck mới giúp nhẹ nhưng vẫn giữ hiệu quả
- SPPF (Spatial Pyramid Pooling - Fast) giúp tăng khả năng phát hiện vật thể ở nhiều tỷ lệ

Mặc dù nhỏ gọn (chỉ ~3 triệu tham số), YOLOv8n vẫn đạt độ chính xác tốt và là lựa chọn lý tưởng cho bài toán thời gian thực, nơi tốc độ ưu tiên hơn độ chính xác tuyệt đối.



# Khối Conv

Khối cơ bản trong kiến trúc gồm lớp Conv2d, BatchNorm2d và hàm kích hoạt SiLU:

- Conv2d: Lớp tích chập 2 chiều trượt bộ lọc (kernel) trên ảnh đầu vào để trích xuất đặc trưng.
  - k: số bộ lọc, xác định độ sâu đầu ra.
  - s: bước trượt, ảnh hưởng kích thước đầu ra.
  - p: padding, thêm viền để giữ kích thước không gian.
  - c: số kênh đầu vào (ví dụ ảnh RGB có c=3).
- BatchNorm2d: Chuẩn hóa đầu ra của lớp Conv2d theo từng batch, giúp ổn định và tăng tốc độ huấn luyện bằng cách duy trì giá trị đầu ra trong phạm vi phù hợp.
- Hàm kích hoạt SiLU (Swish):  $\text{SiLU}(x) = x * \text{sigmoid}(x)$  ( $\text{sigmoid}(x) = 1 / (1 + e^{-x})$ ). SiLU tạo gradient mượt mà, giúp cải thiện hiệu quả huấn luyện và giảm vấn đề gradient biến mất.

# Khối nút thắt cổ chai (Bottleneck Block)

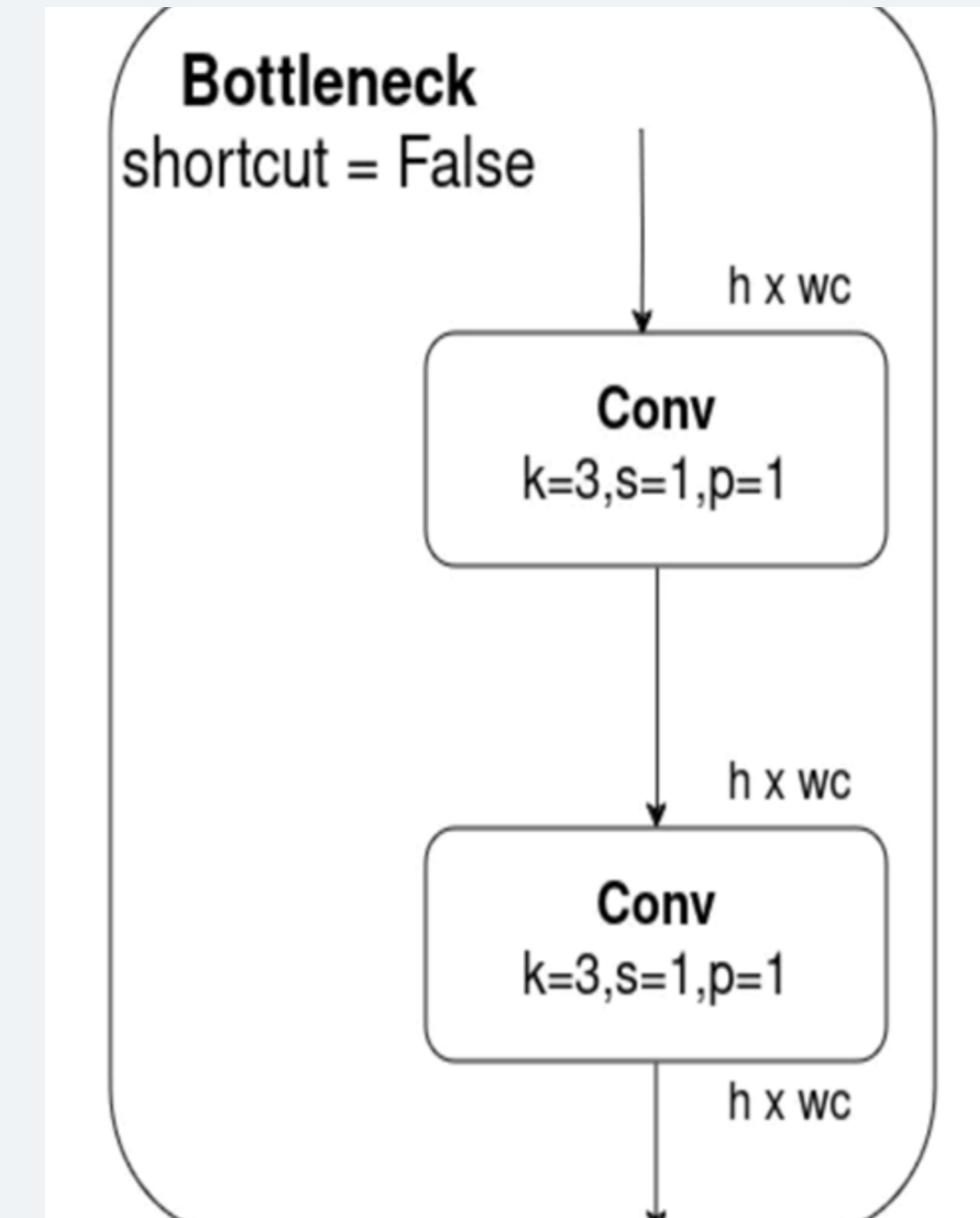
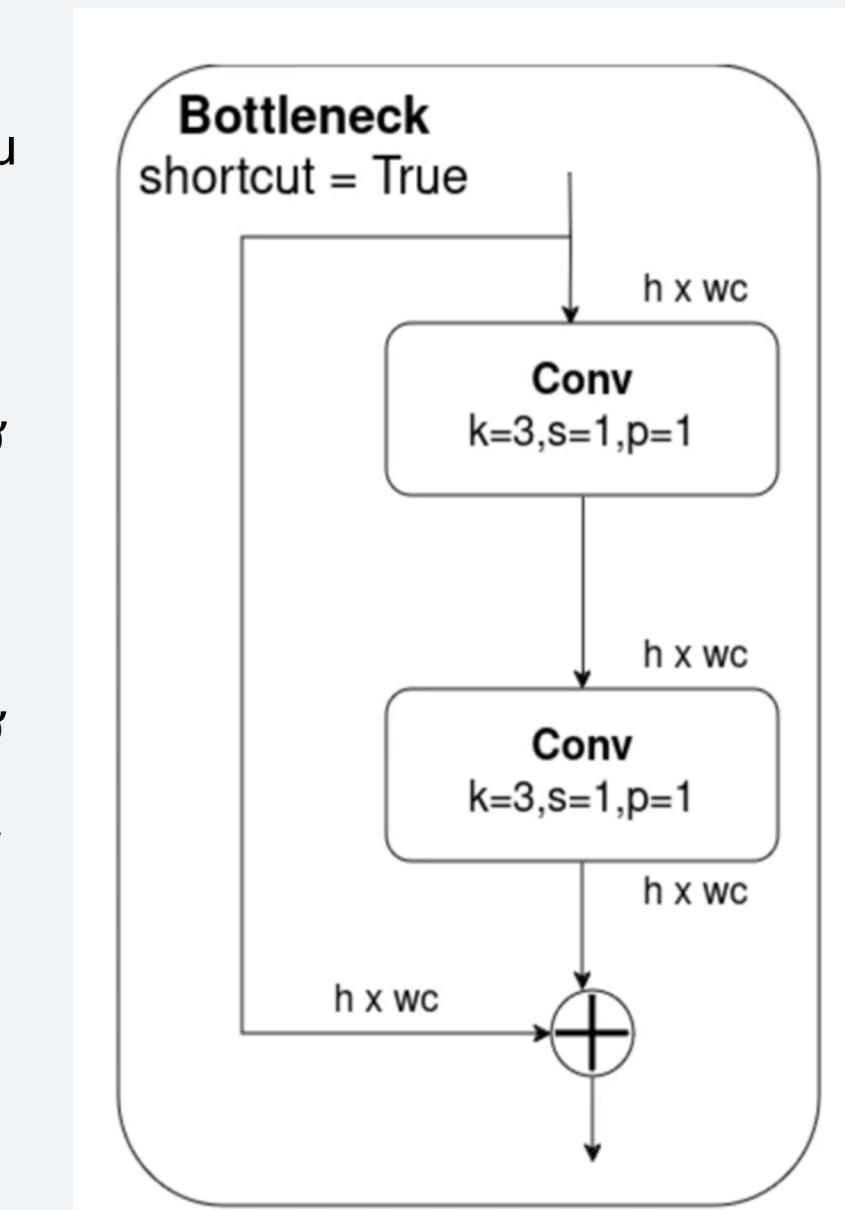
Khối nút thắt cổ chai (Bottleneck Block) là một khối gồm hai lớp Conv nối tiếp và tùy chọn kết nối phím tắt (shortcut).

- Nếu shortcut = True, đầu vào sẽ được cộng trực tiếp vào đầu ra (residual connection).
- Nếu shortcut = False, đầu vào đi qua đủ hai lớp Conv mà không cộng thêm.

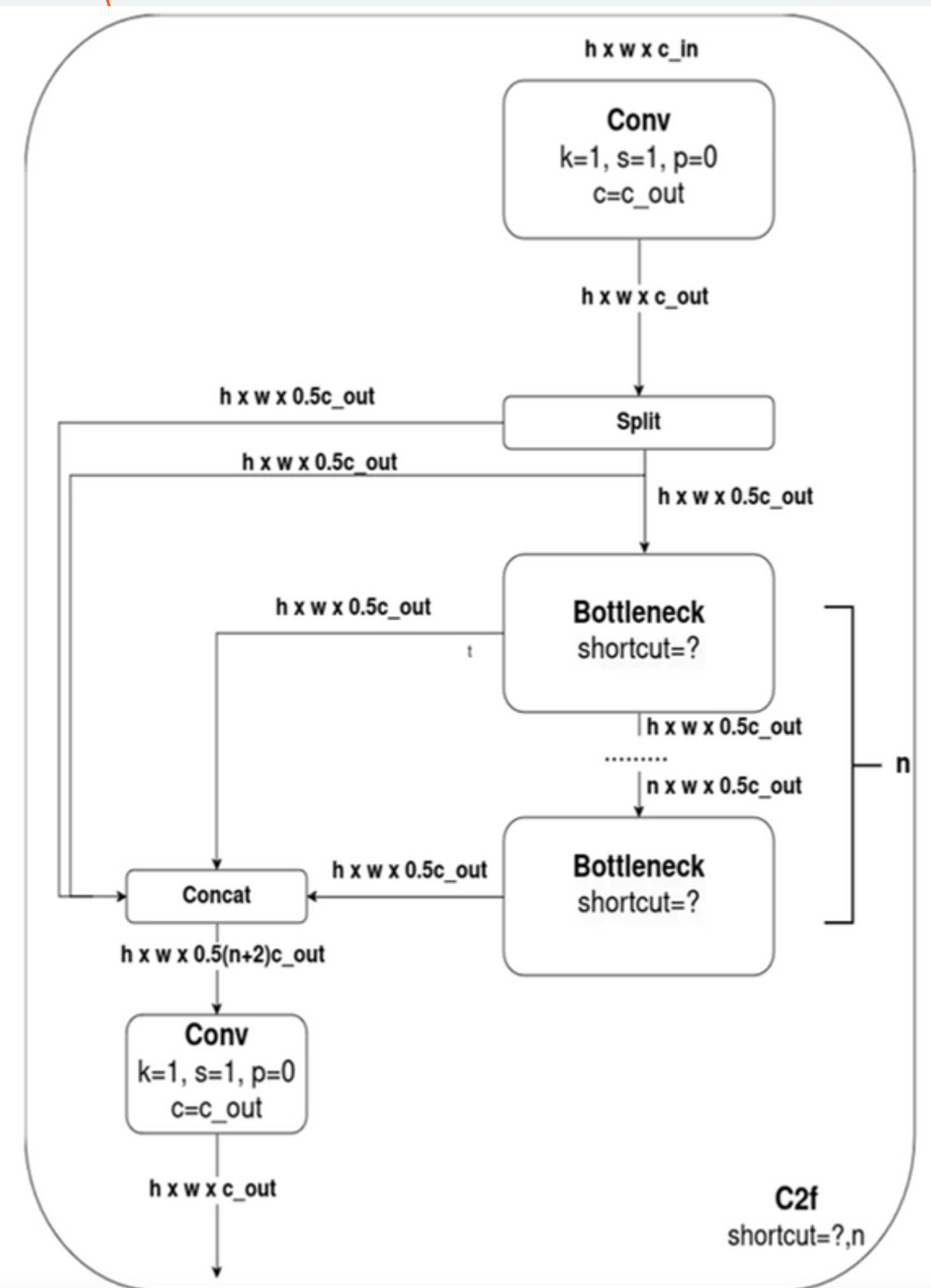
Kết nối phím tắt (Shortcut Connection) là một loại kết nối bỏ qua một hoặc nhiều lớp, giúp gradient truyền ngược dễ dàng hơn. Cơ chế này giúp:

- Giảm hiện tượng gradient biến mất,
- Cho phép học ánh xạ danh tính dễ dàng hơn,
- Tăng hiệu quả huấn luyện trong các mạng sâu.

Gradient biến mất (Vanishing Gradient) là hiện tượng gradient trở nên rất nhỏ khi lan truyền ngược qua nhiều lớp, làm cho các lớp đầu khó học. Shortcut giúp khắc phục vấn đề này bằng cách duy trì dòng gradient mạnh hơn trong quá trình huấn luyện.



# Khối C2f (Cross-Stage Partial with Bottlenecks)



Khối C2f (Cross-Stage Partial with Bottlenecks) là một kiến trúc bao gồm:

- Một lớp Conv2d đầu vào tạo ra bản đồ đặc trưng.
- Bản đồ này được chia thành hai phần:
  - o Một phần được đưa qua chuỗi các khối Bottleneck
    - Bottleneck Block là một khối gồm nhiều lớp Conv2D được sắp xếp theo một cách đặc biệt để:
      - Nén (giảm) số lượng kênh trung gian → giảm chi phí tính toán.
      - Sau đó mở rộng lại → giữ lại thông tin quan trọng.
      - Có thể kết hợp với kết nối tắt (shortcut/residual connection).
    - o Phần còn lại đi thẳng đến khối Concat.
  - Số lượng khối Bottleneck được điều khiển bởi tham số `depth_multiple` của mô hình.
  - Cuối cùng, đầu ra từ các khối Bottleneck và phần đặc trưng còn lại được nối lại (concatenated) rồi đưa qua một lớp Conv để tạo đầu ra khối C2f.
- Cấu trúc này giúp giảm chi phí tính toán, tận dụng kết nối tắt, đồng thời duy trì khả năng học các đặc trưng sâu.

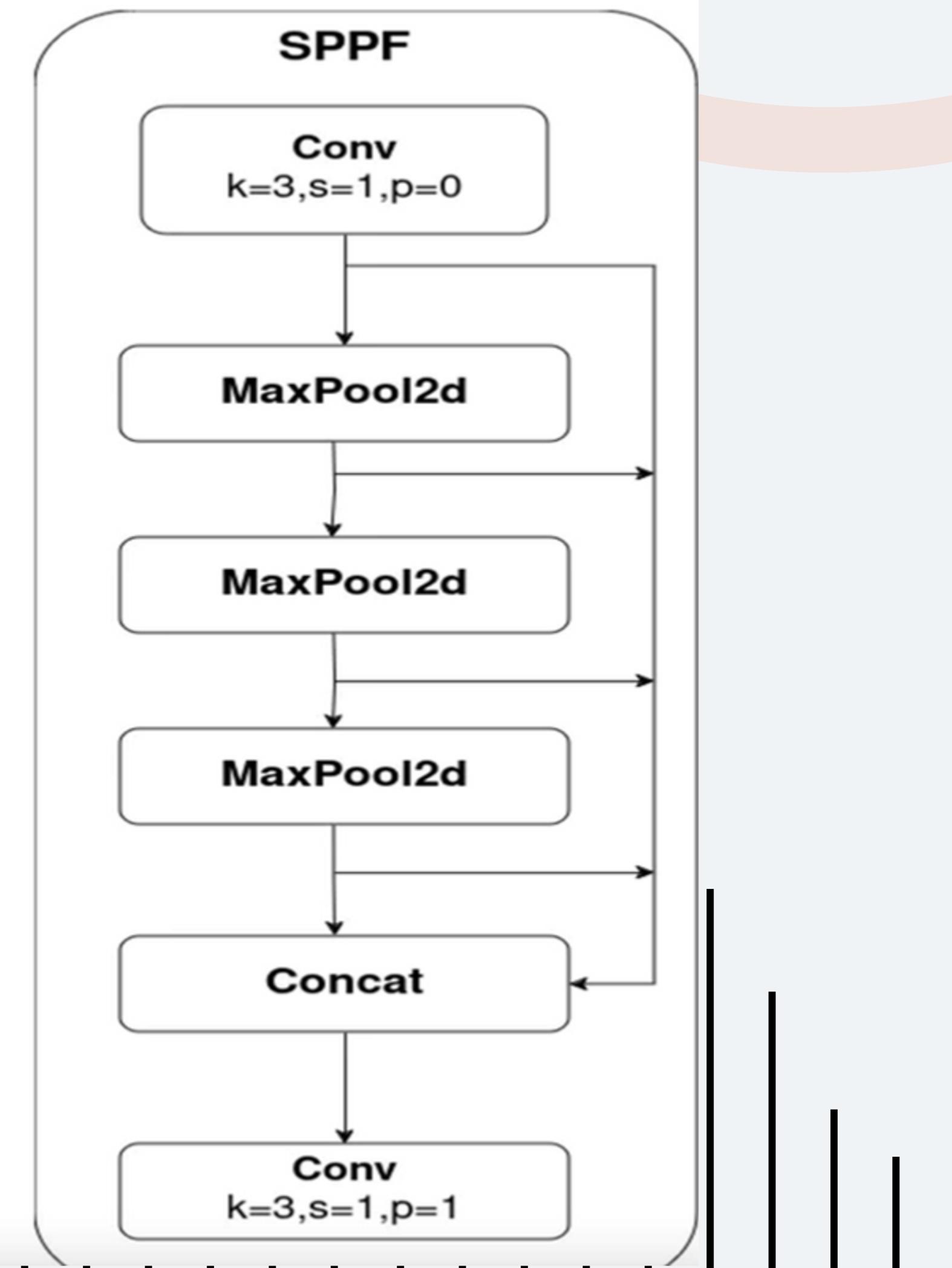
# Khối SPPF (Spatial Pyramid Pooling - Fast)

Khối SPPF (Spatial Pyramid Pooling - Fast) gồm:

- Một lớp Conv2d đầu vào,
- Theo sau là ba lớp MaxPool2d liên tiếp (thường với kernel cố định, ví dụ 5x5),
- Các đầu ra từ mỗi lớp MaxPool2d được nối (concatenate) với đầu ra ban đầu,
- Sau đó được đưa qua một lớp Conv cuối để sinh ra đặc trưng đầu ra.

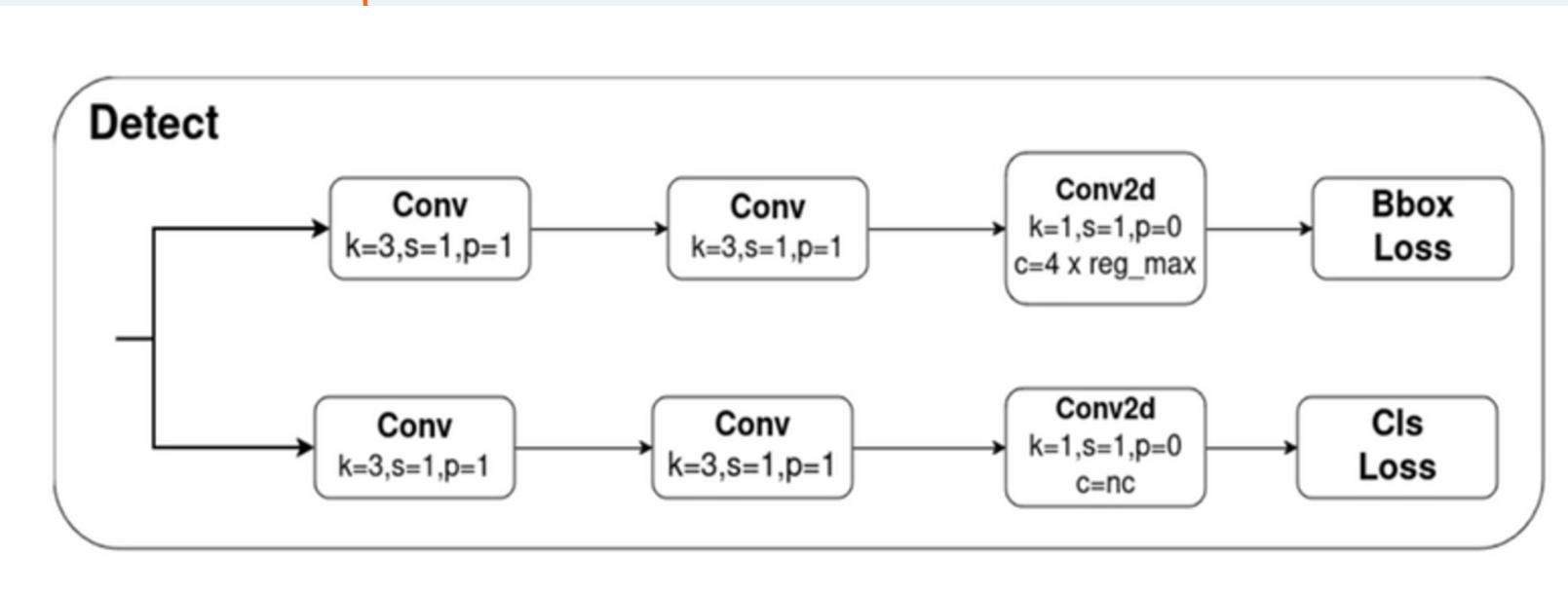
Ý tưởng chính của SPP là trích xuất đặc trưng đa tỷ lệ bằng cách gộp thông tin từ nhiều vùng không gian, giúp mạng xử lý hiệu quả hình ảnh với kích thước và tỷ lệ đối tượng khác nhau. Tuy nhiên, SPPF đơn giản hóa SPP bằng cách chỉ dùng một kích thước kernel cố định, giúp giảm chi phí tính toán mà vẫn giữ được hiệu quả nhận dạng.

MaxPool2d là lớp gộp tối đa 2D, dùng để giảm kích thước không gian và giữ lại đặc trưng nổi bật nhất trong mỗi vùng. Việc giảm mẫu này giúp giảm độ phức tạp và tăng tính khái quát của mô hình.



# Khối Detect

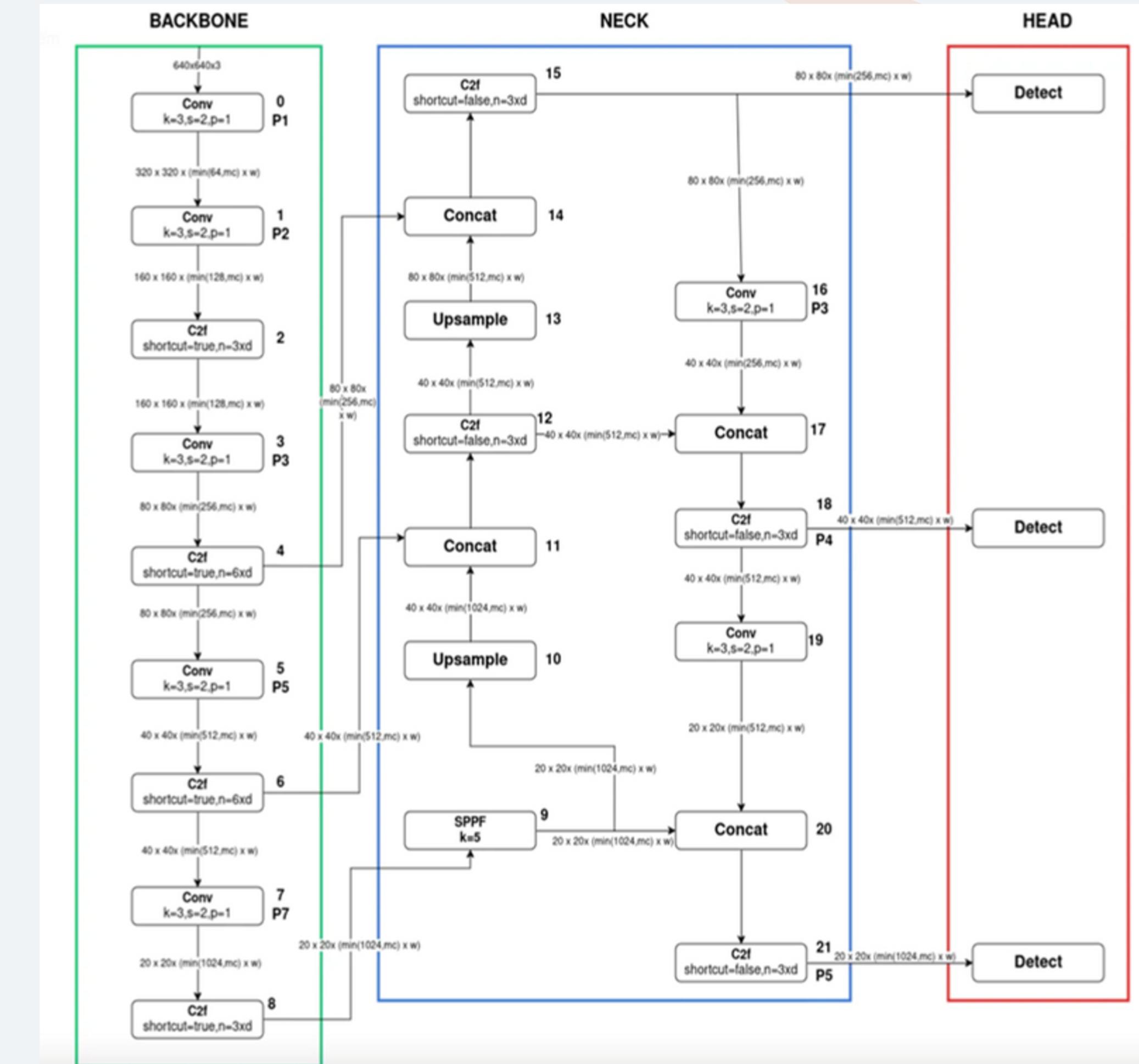
---



Khối Detect (Phát hiện) chịu trách nhiệm dự đoán vị trí và phân loại đối tượng. Không giống các phiên bản trước, YOLOv8 sử dụng kiến trúc không neo (anchor-free), trong đó mô hình dự đoán trực tiếp tâm đối tượng và kích thước hộp giới hạn thay vì lệch so với các anchor box cố định. Cách tiếp cận này giúp giảm số lượng dự đoán cần thiết, tăng tốc độ xử lý hậu kỳ và cải thiện độ chính xác. Khối Detect gồm hai nhánh (track):

- Nhánh 1: Dự đoán hộp giới hạn (Bounding Box),
  - Nhánh 2: Dự đoán phân lớp (Class Prediction).
- Mỗi nhánh gồm hai lớp tích chập (Conv) và kết thúc bằng một lớp Conv2d duy nhất để tạo ra đầu ra tương ứng:
- Bounding Box Loss ở nhánh 1,
  - Class Loss ở nhánh 2.

# Sơ đồ tổng thể

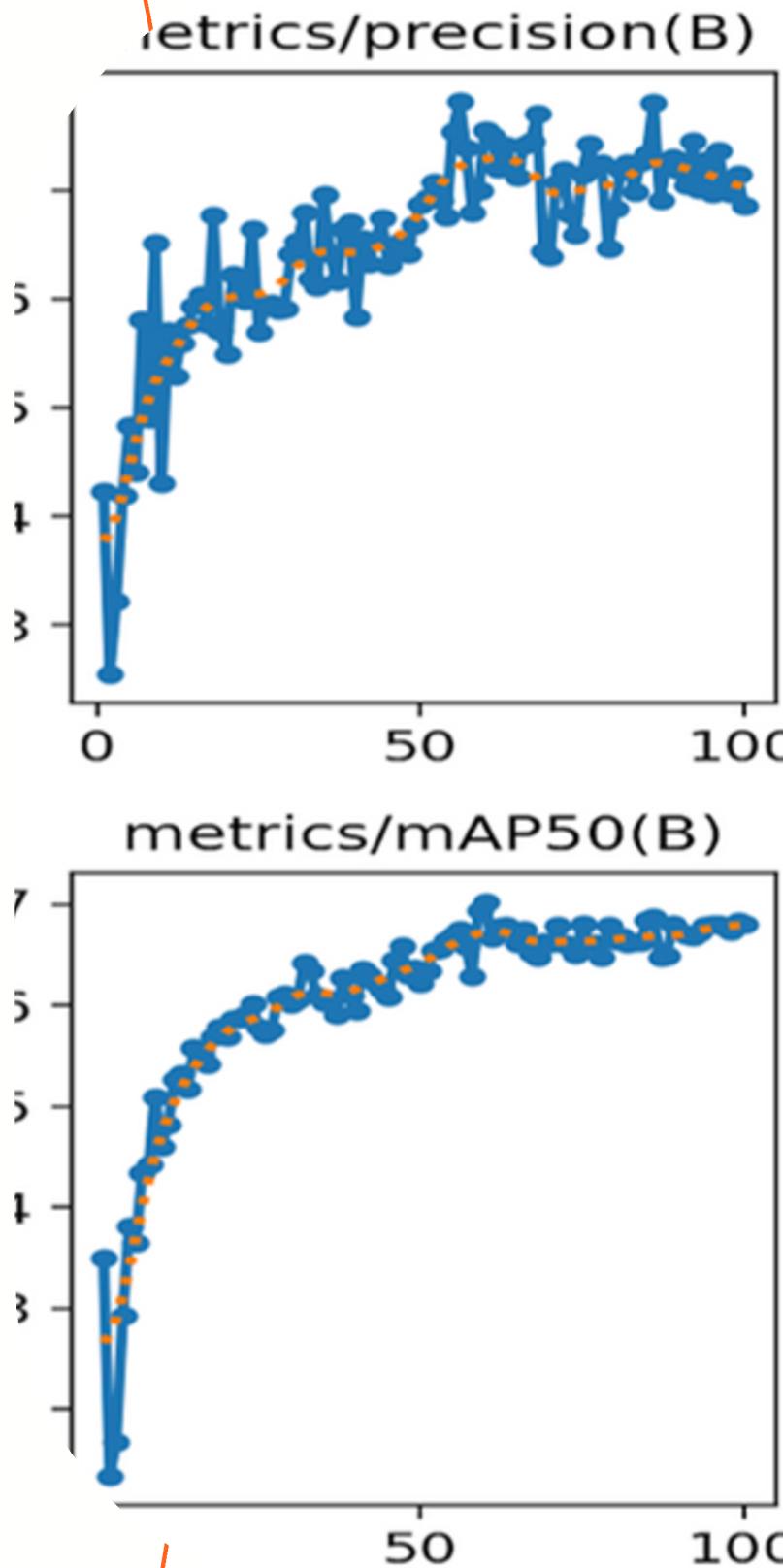


# Mô hình YOLOv8n

Mô hình YOLOv8n cho thấy hiệu quả tốt trong bài toán phát hiện đối tượng nhờ vào kiến trúc hiện đại, nhẹ và tối ưu. Với thiết kế anchor-free, kết hợp cùng các khối C2f, Bottleneck và SPPF, mô hình có khả năng học đặc trưng hiệu quả và xử lý nhanh chóng ngay cả trên thiết bị có cấu hình thấp.

Trong quá trình huấn luyện và đánh giá, mô hình thể hiện khả năng nhận diện đối tượng ổn định, phản hồi nhanh và phù hợp với các ứng dụng yêu cầu thời gian thực như giám sát, cảnh báo thông minh hoặc hệ thống nhúng.

Tổng thể, YOLOv8n là một lựa chọn phù hợp khi cần cân bằng giữa tốc độ, độ chính xác và tài nguyên phần cứng.



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9



***Thank  
you***

