

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CUỐI KỲ

NHẬN DIỆN KÍ HIỆU TAY

Khoa: Khoa học máy tính

Môn học: Thị giác máy tính nâng cao

Giảng viên: Mai Tiến Dũng

Lớp: CS331.P11

Thành phố Hồ Chí Minh, tháng 1 năm 2025

THÔNG TIN THÀNH VIÊN

STT	Thành viên	MSSV
1	Đoàn Công Tài	22521271
2	Lê Cường Thịnh	22521409
3	Nguyễn Duy Thịnh	22521414

BẢNG PHÂN CÔNG

STT	Nhiệm vụ	Mô tả	Thành viên	Tiến độ hoàn thành
1	Lí do chọn đề tài, tầm quan trọng của bài toán		Lê Cường Thịnh	100%
2	Phát biểu bài toán	Phân tích, chỉ rõ Problem, Input, Output	Lê Cường Thịnh	100%
3	Phương pháp sử dụng	SSD	Lê Cường Thịnh	100%
		Yolov5	Nguyễn Duy Thịnh	100%
		Yolov8	Đoàn Công Tài	100%
4	Kết quả thực nghiệm	Giới thiệu dataset, so sánh kết quả, độ đo	Nguyễn Duy Thịnh	100%
5	Đánh giá, nhận xét từng phương pháp	So sánh ưu điểm, hạn chế của từng phương pháp	Đoàn Công Tài	100%
6	Tài liệu tham khảo		Đoàn Công Tài	100%

MỤC LỤC

I. LÍ DO CHỌN ĐỀ TÀI.....	5
A. Lí do chọn đề tài.....	5
1. Tăng cường giao tiếp giữa người khiếm thính và cộng đồng.....	5
2. Ứng dụng trong giáo dục và hỗ trợ học tập ASL.....	5
3. Thách thức và cơ hội trong lĩnh vực thị giác máy tính và AI.....	5
B. Tầm quan trọng của bài toán phát hiện ngôn ngữ ký hiệu tay ASL.....	6
1. Hỗ trợ người khiếm thính và câm điếc.....	6
2. Ứng dụng trong dịch thuật ngôn ngữ ký hiệu.....	6
3. Tiềm năng ứng dụng rộng rãi trong công nghệ hiện đại.....	6
C. Thách thức trong bài toán phát hiện ASL.....	7
1. Sự phức tạp của ASL.....	7
2. Dữ liệu hạn chế và tính đa dạng.....	7
3. Yêu cầu độ chính xác cao.....	7
4. Kết luận.....	7
II. PHÁT BIỂU BÀI TOÁN.....	8
A. Problem.....	8
B. Input.....	8
1. Hình ảnh hoặc khung hình từ video chứa ký hiệu ASL.....	8
2. Định dạng dữ liệu.....	8
3. Annotations (Nhãn dữ liệu).....	8
C. Output.....	9
1. Bounding Box (Hộp giới hạn).....	9
2. Nhãn Dự Đoán Ký Hiệu Tay.....	9
3. Độ Tin Cậy (Confidence Score).....	10
III. CÁC PHƯƠNG PHÁP HỌC SÂU.....	10
A. SSDLite320.....	10
1. Khái niệm.....	10
2. Đặc điểm nổi bật.....	10
3. Áp dụng vào bài toán.....	12

B. YOLOv5 (YOU ONLY LOOK ONCE).....	16
1. Giới thiệu	16
2. Kiến trúc YOLOv5.....	17
3. Áp dụng vào bài toán	21
C. YOLOv8.....	23
1. Giới thiệu	23
2. Kiến trúc YOLOv8.....	24
3. Áp dụng vào bài toán	25
IV. THỰC NGHIỆM	27
A. Dataset	27
1. Giới thiệu	27
2. Một số thông tin cơ bản về bộ dữ liệu	28
B. Độ đo	28
1. Precision.....	28
2. Recall	28
3. Intersection Over Union (IOU).....	29
4. mean Average Precision (mAP)	30
C. Kết quả.....	30
V. ĐÁNH GIÁ.....	32
A. Ưu điểm.....	32
B. Hạn chế.....	32
VI. TÀI LIỆU THAM KHẢO.....	33

I. LÍ DO CHỌN ĐỀ TÀI

A. Lí do chọn đề tài

Ngôn ngữ ký hiệu Mỹ (ASL - American Sign Language) là một trong những ngôn ngữ ký hiệu phổ biến nhất trên thế giới, được sử dụng rộng rãi bởi cộng đồng người khiếm thính tại Mỹ và nhiều quốc gia khác. Tuy nhiên, không phải ai cũng hiểu và sử dụng được ASL, dẫn đến rào cản giao tiếp giữa người khiếm thính và cộng đồng rộng lớn. Chính vì vậy, nghiên cứu và phát triển một hệ thống phát hiện ngôn ngữ ký hiệu tay ASL là cần thiết, với những lợi ích chính như sau:

1. Tăng cường giao tiếp giữa người khiếm thính và cộng đồng

- ASL là một ngôn ngữ phi âm thanh, dựa vào cử động tay, nét mặt và ngôn ngữ cơ thể để truyền đạt thông tin.
- Việc phát triển một hệ thống nhận diện ASL giúp chuyển đổi cử chỉ tay sang văn bản, từ đó tạo cầu nối giữa người khiếm thính và những người không biết ASL.
- Điều này giúp người khiếm thính giao tiếp thuận tiện hơn trong đời sống hàng ngày, từ công việc, học tập cho đến các hoạt động xã hội.

2. Ứng dụng trong giáo dục và hỗ trợ học tập ASL

- Hệ thống nhận diện ASL có thể được tích hợp vào các nền tảng giáo dục để hỗ trợ người mới học ngôn ngữ này.
- Các ứng dụng sử dụng công nghệ nhận diện ký hiệu tay có thể cung cấp phản hồi theo thời gian thực, giúp người học biết họ đã thực hiện đúng cử chỉ hay chưa.
- Trẻ em khiếm thính có thể sử dụng công nghệ này để học tập dễ dàng hơn mà không cần người hướng dẫn trực tiếp.

3. Thách thức và cơ hội trong lĩnh vực thị giác máy tính và AI

- Bài toán nhận diện ASL không chỉ là một bài toán phân loại hình ảnh thông thường mà còn đòi hỏi mô hình có khả năng phân biệt các cử chỉ phức tạp với độ chính xác cao.
- Việc phát triển mô hình nhận diện ASL giúp nâng cao khả năng ứng dụng của AI trong thị giác máy tính, từ đó mở ra nhiều hướng nghiên cứu và phát triển trong lĩnh vực nhận diện cử chỉ, hành vi.
- Đây là một bài toán thú vị, kết hợp nhiều lĩnh vực như xử lý ảnh, học sâu (deep learning), và thị giác máy tính (computer vision).

B. Tầm quan trọng của bài toán phát hiện ngôn ngữ ký hiệu tay ASL

Việc phát triển hệ thống phát hiện ASL không chỉ có ý nghĩa về mặt công nghệ mà còn mang lại nhiều giá trị xã hội:

1. Hỗ trợ người khiếm thính và câm điếc

- Theo thống kê của Tổ chức Y tế Thế giới (WHO), có hơn **466 triệu người bị khiếm thính trên toàn cầu**, và con số này có thể tiếp tục tăng.
- Trong đó, nhiều người sử dụng ASL như ngôn ngữ giao tiếp chính, nhưng không phải ai cũng hiểu ASL.
- Một hệ thống phát hiện ASL có thể giúp những người này giao tiếp với cộng đồng dễ dàng hơn, giúp họ có nhiều cơ hội hơn trong công việc và cuộc sống.

2. Ứng dụng trong dịch thuật ngôn ngữ ký hiệu

- Hệ thống nhận diện ASL có thể được tích hợp vào các thiết bị thông minh hoặc phần mềm dịch thuật, giúp chuyển đổi cử chỉ tay thành văn bản hoặc giọng nói theo thời gian thực.
- Điều này giúp loại bỏ rào cản ngôn ngữ giữa người khiếm thính và người không biết ASL, tạo điều kiện cho họ tham gia vào nhiều lĩnh vực hơn trong xã hội.

3. Tiềm năng ứng dụng rộng rãi trong công nghệ hiện đại

- Hệ thống phát hiện ASL có thể được tích hợp vào các nền tảng hội thoại thông minh như trợ lý ảo, chatbot, hoặc các thiết bị IoT để hỗ trợ giao tiếp bằng cử chỉ.
- Các công nghệ nhận diện cử chỉ có thể được ứng dụng trong thực tế ảo (VR), điều khiển bằng cử chỉ trong các thiết bị không chạm (touchless control), hoặc robot hỗ trợ người khuyết tật.
- Trong tương lai, hệ thống nhận diện ASL có thể kết hợp với công nghệ deep learning và xử lý ngôn ngữ tự nhiên (NLP) để tạo ra các trợ lý ảo thông minh hỗ trợ ngôn ngữ ký hiệu.

C. Thách thức trong bài toán phát hiện ASL

Mặc dù có nhiều tiềm năng ứng dụng, bài toán phát hiện ASL vẫn gặp một số thách thức lớn:

1. Sự phức tạp của ASL

- ASL không chỉ dựa vào cử động tay mà còn có sự kết hợp với nét mặt và ngôn ngữ cơ thể.
- Một số ký hiệu rất giống nhau, chỉ khác biệt ở vị trí đặt tay hoặc hướng di chuyển, gây khó khăn trong nhận diện.

2. Dữ liệu hạn chế và tính đa dạng

- Thu thập dữ liệu ASL chất lượng cao là một thách thức, vì cần đảm bảo đủ độ phân giải, góc quay, ánh sáng, và đặc biệt là sự đa dạng về người ký hiệu.
- ASL có nhiều phong cách khác nhau tùy theo vùng miền và cá nhân, do đó mô hình phải có khả năng tổng quát hóa tốt.

3. Yêu cầu độ chính xác cao

- Nhận diện ASL đòi hỏi mô hình có độ chính xác cao, khả năng nhận diện theo thời gian thực để có thể ứng dụng thực tế.
- Việc xử lý hình ảnh theo thời gian thực, đặc biệt là khi phát hiện ký hiệu động, đòi hỏi phần cứng mạnh và thuật toán tối ưu.

4. Kết luận

Bài toán phát hiện ngôn ngữ ký hiệu ASL không chỉ có ý nghĩa quan trọng đối với cộng đồng người khiếm thính mà còn mở ra nhiều cơ hội phát triển công nghệ trong thị giác máy tính và trí tuệ nhân tạo.

- Hệ thống nhận diện ASL giúp thu hẹp khoảng cách giao tiếp giữa người khiếm thính và cộng đồng.
- Ứng dụng trong giáo dục, dịch thuật ngôn ngữ ký hiệu và các công nghệ điều khiển bằng cử chỉ có thể cải thiện chất lượng cuộc sống.
- Đây là một bài toán thú vị, đầy thách thức nhưng cũng có tiềm năng phát triển lớn, hứa hẹn những đóng góp quan trọng trong lĩnh vực AI và thị giác máy tính trong tương lai.

Việc nghiên cứu và phát triển một hệ thống nhận diện ASL hiệu quả không chỉ mang lại lợi ích cho cộng đồng người khiếm thính mà còn góp phần vào sự phát triển của công nghệ thông minh trong thời đại 4.0.

II. PHÁT BIỂU BÀI TOÁN

A. Problem

Bài toán phát hiện ngôn ngữ ký hiệu tay ASL (American Sign Language) là bài toán thuộc lĩnh vực thị giác máy tính (Computer Vision) và học sâu (Deep Learning). Mục tiêu của bài toán là xây dựng một mô hình có thể phát hiện và nhận diện chính xác các ký hiệu tay ASL từ hình ảnh hoặc video. Bài toán yêu cầu mô hình xác định vị trí bàn tay trong ảnh hoặc video, sau đó dự đoán ký hiệu ASL tương ứng. Hệ thống có thể được sử dụng trong nhiều ứng dụng như dịch ngôn ngữ ký hiệu sang văn bản, hỗ trợ người khiếm thính giao tiếp với cộng đồng.

Cụ thể hơn, bài toán nhận diện các ngôn ngữ ký hiệu tay của 26 chữ cái theo bảng chữ cái Alphabet:

Classes = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

B. Input

1. Hình ảnh hoặc khung hình từ video chứa ký hiệu ASL

- Ảnh có thể chứa một hoặc nhiều bàn tay thực hiện các ký hiệu ASL.
- Mỗi ảnh chứa một ký hiệu thuộc tập hợp các chữ cái từ **A-Z** (26 ký hiệu trong ASL).

2. Định dạng dữ liệu

- Dữ liệu có thể được lưu trữ dưới dạng **JPEG, PNG** hoặc các định dạng hình ảnh khác.
- Nếu xử lý video, đầu vào có thể là một chuỗi khung hình (**frames**) trích xuất từ video.

3. Annotations (Nhãn dữ liệu)

- Dữ liệu có thể được gán nhãn theo định dạng **COCO** hoặc **Pascal VOC**, trong đó mỗi hình ảnh có thông tin về:
 - **Bounding box** (tọa độ hộp giới hạn vị trí bàn tay).
 - **Label** (nhãn ký hiệu ASL tương ứng).

C. Output

Đầu ra của mô hình phát hiện ngôn ngữ ký hiệu tay ASL bao gồm **bounding box** (hộp giới hạn) xác định vị trí bàn tay trong ảnh và **nhãn dự đoán ký hiệu tay** tương ứng với chữ cái ASL.

Output có thể được biểu diễn dưới nhiều hình thức khác nhau, nhưng thường bao gồm các thông tin chính sau:

1. Bounding Box (Hộp giới hạn)

- Bounding box là một hộp chữ nhật mô tả khu vực chứa bàn tay đang thực hiện ký hiệu ASL.
- Được biểu diễn bằng **tọa độ và kích thước**, thường có định dạng:

(x, y, w, h)

Trong đó:

- **x, y**: Tọa độ của góc trên bên trái hộp giới hạn.
- **w, h**: Chiều rộng và chiều cao của hộp giới hạn.
- Nếu có nhiều bàn tay xuất hiện (mặc dù mô hình hiện tại chỉ làm việc với một bàn tay), bounding box sẽ chỉ tập trung vào một đối tượng duy nhất.
- Bounding box có thể thay đổi kích thước tùy thuộc vào khoảng cách của bàn tay so với camera.

2. Nhãn Dự Đoán Ký Hiệu Tay

- Nhãn dự đoán các chữ cái **A-Z** trong bảng chữ cái ASL mà mô hình nhận diện được từ ký hiệu tay trong ảnh.
- Mỗi bounding box được gán một nhãn dự đoán, ví dụ:
 - **"A"** nếu mô hình phát hiện bàn tay đang tạo ký hiệu ASL cho chữ "A".
 - **"B"** nếu mô hình nhận diện ký hiệu tay cho chữ "B".
- Dự đoán được thực hiện dựa trên mô hình deep learning đã được huấn luyện trên tập dữ liệu ASL.

3. Độ Tin Cậy (Confidence Score)

- Mô hình cũng có thể xuất ra một **giá trị độ tin cậy (confidence score)**, thường nằm trong khoảng $[0,1]$ hoặc được biểu diễn dưới dạng phần trăm (%).
- Giá trị này thể hiện mức độ chắc chắn của mô hình về dự đoán của nó.

III. CÁC PHƯƠNG PHÁP HỌC SÂU

A. SSDLite320

1. Khái niệm

SSDLite là một phiên bản tối ưu hóa của **Single Shot MultiBox Detector (SSD)**, được thiết kế để chạy nhanh hơn và nhẹ hơn, đặc biệt trên các thiết bị có tài nguyên tính toán hạn chế như điện thoại di động và nhúng (embedded systems).

- **SSDLite** được giới thiệu trong bài báo *MobileNetV2: Inverted Residuals and Linear Bottlenecks* của Google vào năm 2018.
- Phương pháp này sử dụng **MobileNetV2** làm backbone thay vì các kiến trúc CNN truyền thống như VGG16 (SSD gốc), giúp giảm đáng kể số lượng tham số và phép tính toán mà vẫn giữ được độ chính xác.
- **SSDLite320** là phiên bản của SSDLite sử dụng kích thước đầu vào cố định là **320x320 pixels**, giúp cân bằng giữa tốc độ và độ chính xác.

2. Đặc điểm nổi bật

a) Kiến trúc mô hình

SSDLite320 là một mô hình **one-stage detector**, tức là nó dự đoán các bounding boxes và nhãn trực tiếp từ một mạng duy nhất mà không cần giai đoạn đề xuất vùng (region proposal) như Faster R-CNN.

- **Backbone:** MobileNetV2 (thay thế VGG16 trong SSD gốc).
- **Feature Pyramid:** Mô hình sử dụng nhiều tầng feature map để phát hiện vật thể với các kích thước khác nhau.
- **Lightweight Convolutions:**

- Sử dụng **Depthwise Separable Convolution** thay vì các convolution thông thường, giúp giảm số lượng tham số và tính toán mà vẫn giữ được hiệu suất cao.
- So sánh số lượng tham số giữa các mô hình SSD

Mô hình	Backbone	Số lượng tham số	FLOPs
SSD300	VGG16	26.3M	31.4B
SSD320Lite	MobileNetV2	~4.3M	~1.5B

Bảng 1: Thông số của mô hình SSD300 và SSD320Lite

→ **Nhận xét:** SSDLite320 có số lượng tham số và FLOPs thấp hơn rất nhiều so với SSD gốc, giúp tăng tốc độ xử lý mà vẫn đảm bảo chất lượng phát hiện.

b) Đặc tính hiệu suất

Tốc độ cao:

- Do sử dụng kiến trúc MobileNetV2 và depthwise separable convolution, SSDLite320 có thể chạy **nhANH GẤP 3-4 lần** so với SSD gốc mà không làm giảm đáng kể độ chính xác.
- Có thể đạt **30-50 FPS** trên GPU và **5-10 FPS** trên CPU của điện thoại.

Kích thước mô hình nhỏ:

- SSD gốc có thể có kích thước mô hình lên tới **100MB+**, trong khi SSDLite320 có thể chỉ khoảng **10-20MB**, rất phù hợp cho các thiết bị di động.

Khả năng tổng quát tốt:

- Mô hình có thể phát hiện đối tượng ở nhiều kích thước khác nhau nhờ sử dụng nhiều feature map (multi-scale feature maps).
- Hiệu suất gần với SSD300 nhưng nhẹ hơn rất nhiều.

c) Ứng dụng thực tế

- Nhận diện và theo dõi đối tượng trên thiết bị di động (Google Lens, AR applications).
- Nhận diện chữ viết tay hoặc ký hiệu tay trong ASL detection (phù hợp với bài toán của tôi).
- Ứng dụng trên xe tự hành, robot, camera giám sát nhờ khả năng chạy nhanh với tài nguyên hạn chế.

d) So sánh SSDLite320 với các phương pháp khác

Tiêu chí	SSD300	SSDLite320	YOLOv5s	YOLOv8n
Loại mô hình	One-stage	One-stage	One-stage	One-stage
Backbone	VGG16	MobileNetV2	CSPDarkNet	P5-PANet
Tốc độ FPS (GPU)	30-50	50-80	100+	120+
Số tham số (M)	26.3M	4.3M	~7M	~3.2M
Kích thước mô hình	100MB+	10-20MB	~14MB	~6MB
Ứng dụng trên di động	Không tối ưu	Rất phù hợp	Có thể dùng	Rất phù hợp

Bảng 2: Thông số của các mô hình SSD300, SSDLite320, YOLOv5s, YOLOv8n

3. Áp dụng vào bài toán

SSDLite320

Chuẩn bị và tăng cường dữ liệu.

```
train_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

model = ssdlite320_mobilenet_v3_large(pretrained=True)
model = model.to(device)

Cấu hình optimizer và scheduler
Optimizer: Adam với learning rate ban đầu là 0.001.
Scheduler: StepLR, giảm learning rate mỗi 10 epoch.

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.1)
```

```
for param in model.backbone.parameters():
    param.requires_grad = False
```

Khởi tạo mô hình SSD320-Lite với backbone MobileNetV3.

Đóng băng backbone và huấn luyện detection head trong 5 epoch và Fine-tune toàn bộ mô hình từ epoch 6 trở đi.

Hình 1: Các tham số sử dụng trong SSDLite320 cho bài toán

a) Chuẩn bị và tăng cường dữ liệu

- **transforms. ToTensor ()**: Chuyển đổi hình ảnh thành tensor có giá trị từ $[0,1]$.
- **transforms. ColorJitter(...)**:
 - **brightness=0.2**: Thay đổi độ sáng $\pm 20\%$.
 - **contrast=0.2**: Điều chỉnh độ tương phản $\pm 20\%$.
 - **saturation=0.2**: Điều chỉnh độ bão hòa $\pm 20\%$.
 - **hue=0.1**: Điều chỉnh sắc độ màu sắc $\pm 10\%$.
- **transforms. Normalize (mean, std)**: Chuẩn hóa dữ liệu với giá trị trung bình (**mean**) và độ lệch chuẩn (**std**) của tập ImageNet.

→ **Mục đích**: Giúp mô hình học tốt hơn bằng cách tạo ra các biến thể khác nhau của hình ảnh trong tập huấn luyện.

b) Khởi tạo mô hình SSDLite320 với backbone MobileNetV3

- Sử dụng mô hình SSDLite320 với MobileNetV3-Large làm backbone.
- Sử dụng trọng số đã được huấn luyện trên tập COCO hoặc ImageNet.
- Ở đây sử dụng backbone là MobileNetV3-Large thay vì MobileNetV2. Vì:

Đặc điểm	MobileNetV2	MobileNetV3-Large
Kiến trúc	Depthwise Separable Convolution + Linear Bottleneck + ReLU6	Depthwise Separable Convolution + Linear Bottleneck + h-swish
Kích thước mô hình	Nhỏ	Tương đương V2 nhưng tối ưu hơn
Tốc độ	Tốt	Nhanh hơn V2 (tối ưu trên CPU & GPU)
Độ chính xác	Khá tốt	Cao hơn khoảng 3-5%

Bảng 3: Thông số của các mô hình backbone MobileNetV2 và MobileNetV3-Large

→ **Kết quả**: MobileNetV3-Large có **tốc độ nhanh hơn, độ chính xác cao hơn** và tiêu tốn ít tài nguyên hơn so với MobileNetV2. MobileNetV3-Large có **nhiều convolution mở rộng hơn MobileNetV2**, giúp phát hiện vật thể nhỏ tốt hơn. Trong bài toán **phát hiện ngôn ngữ ký hiệu tay ASL**, mô hình cần nhận diện chính xác các ký hiệu ngón tay nhỏ → MobileNetV3-Large giúp làm điều này tốt hơn.

c) Đóng băng backbone và fine-tune mô hình

Giai đoạn 1: Huấn luyện detection head (5 epoch đầu)

- Với backbone đã bị đóng băng, chỉ detection head được huấn luyện.
- Điều này giúp mô hình học nhanh hơn, tránh làm thay đổi quá nhiều thông tin từ trọng số pre-trained.

Giai đoạn 2: Fine-tune toàn bộ mô hình (từ epoch 6 trở đi)

- Mở khóa backbone để toàn bộ mô hình được huấn luyện.
- Mô hình sẽ học lại cả đặc trưng của backbone lẫn detection head, giúp tối ưu hóa toàn diện.

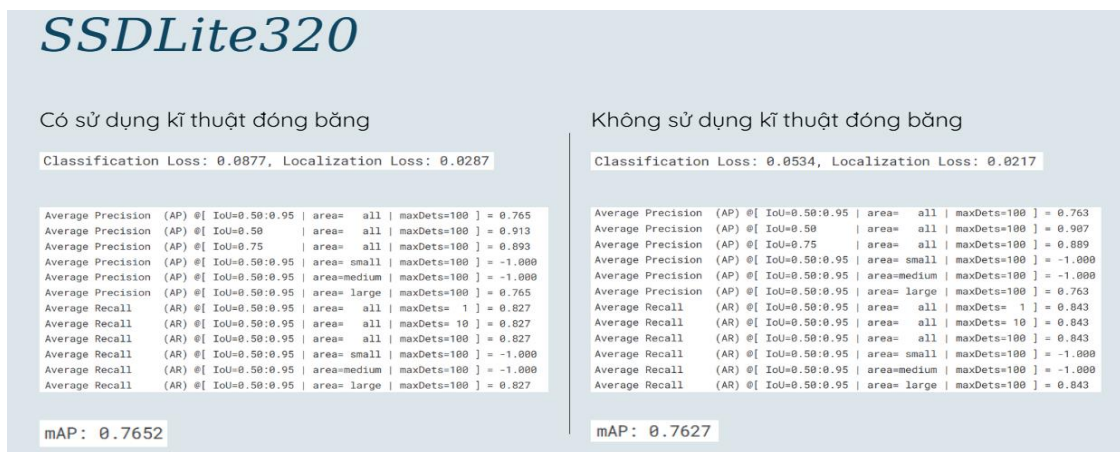
Tóm tắt quy trình huấn luyện SSDLite320

- Tiền xử lý và tăng cường dữ liệu (Data Augmentation).
- Khởi tạo mô hình với MobileNetV3 và chuyển lên GPU.
- Sử dụng Adam Optimizer và StepLR Scheduler để điều chỉnh learning rate.
- Đóng băng backbone và chỉ huấn luyện detection head trong 5 epoch đầu.
- Fine-tune toàn bộ mô hình từ epoch 6 trở đi.

Ưu điểm của phương pháp này:

- Tăng tốc huấn luyện ban đầu bằng cách tận dụng mô hình pre-trained.
- Giúp mô hình hội tụ tốt hơn mà không bị overfitting quá sớm.
- Tiết kiệm tài nguyên tính toán khi áp dụng trên GPU Kaggle hoặc thiết bị di động.

d) Khác biệt khi sử dụng kỹ thuật đóng băng và không sử dụng kỹ thuật đóng băng



Hình 2: Kết quả so sánh việc sử dụng kỹ thuật đóng băng so với ban đầu

(1) Giữ lại đặc trưng hữu ích từ MobileNetV3

- Khi đóng băng các tầng backbone, mô hình chỉ cập nhật trọng số của detection head.
- Điều này giúp giữ nguyên các đặc trưng tiền huấn luyện từ MobileNetV3, vốn đã được tối ưu cho nhận diện hình ảnh.

Kết quả:

- Mô hình không bị mất đi các đặc trưng quan trọng đã học từ tập dữ liệu ImageNet.
- Quá trình huấn luyện ổn định hơn, đặc biệt với tập dữ liệu nhỏ hoặc có nhiễu.

(2) Hạn chế overfitting và tăng tính tổng quát

- Nếu không đóng băng, toàn bộ trọng số của MobileNetV3 cũng được cập nhật.
- Điều này có thể dẫn đến quá khớp (overfitting) với tập dữ liệu huấn luyện, làm giảm khả năng tổng quát.
- Khi đóng băng backbone, mô hình học chậm hơn nhưng ổn định hơn, giúp phát hiện đối tượng tốt hơn.

Kết quả:

- mAP 0.7652 cao hơn so với 0.7627, tức là mô hình phát hiện chính xác hơn.
- Giá trị AP @ IoU=0.50 và AP @ IoU=0.75 đều cao hơn khi đóng băng backbone.

(3) Ảnh hưởng đến Classification Loss và Localization Loss

- Classification Loss khi có đóng băng cao hơn (0.0877 so với 0.0534) → Có thể do mô hình chưa được tối ưu hóa hoàn toàn cho bài toán.
- Localization Loss khi có đóng băng cao hơn (0.0287 so với 0.0217) → Có thể do bounding box chưa được fine-tune đủ tốt.
- Tuy nhiên, tổng thể mAP vẫn cao hơn → Mô hình vẫn phát hiện chính xác hơn.

Kết quả:

- Nếu muốn giảm classification loss, có thể fine-tune lại detection head trong nhiều epoch hơn.
- Nếu muốn cải thiện localization, có thể tăng dữ liệu hoặc sử dụng augmentation tốt hơn.

Tóm lại:

Phân loại (Classification):

- Hiệu suất phân loại của hai phương pháp tương đương, với classification loss và mAP gần như không chênh lệch.

Phát hiện đối tượng (Object Detection):

- Mô hình có đóng băng backbone nhỉnh hơn, đặc biệt ở chỉ số $mAP@[IoU=0.50]$ và $mAP@[IoU=0.75]$, nhờ duy trì ổn định các đặc trưng pretrain trong giai đoạn đầu.

Ý nghĩa:

- Đóng băng backbone giúp phát huy tối đa các đặc trưng tổng quát từ pretrain, hỗ trợ định vị bounding box chính xác hơn, phù hợp với dữ liệu quy mô nhỏ hoặc trung bình.
- ➔ **Kết luận:** Đóng băng backbone nhỉnh hơn trong việc phát hiện đối tượng, đặc biệt với các tập dữ liệu không quá lớn, nhờ vào việc bảo tồn các đặc trưng mạnh mẽ từ mô hình pretrain trong giai đoạn đầu huấn luyện.

B. YOLOv5 (YOU ONLY LOOK ONCE)

1. Giới thiệu

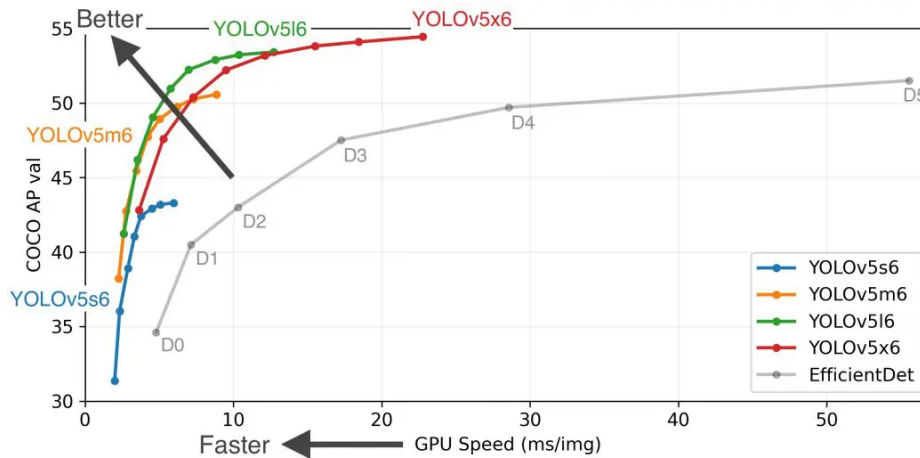
YOLOv5 (You Only Look Once version 5) là một trong những phiên bản nổi bật của dòng mô hình YOLO (You Only Look Once) được thiết kế để giải quyết bài toán phát hiện đối tượng (Object Detection). Mặc dù không phải là phiên bản chính thức từ tác giả gốc của YOLO, YOLOv5 vẫn được cộng đồng quan tâm nhờ tính dễ sử dụng và hiệu quả cao.

YOLOv5 được phát hành bởi một công ty có tên Ultralytics vào năm 2020. Nó được xuất bản trên GitHub (<https://github.com/ultralytics/yolov5>) bởi Glenn Jocher, Người sáng lập & Giám đốc điều hành tại Ultralytics, và nhanh chóng thu hút được sự chú ý ngay sau khi xuất bản.

YOLOv5 đã được phát hành với năm phiên bản khác nhau:

Model	Size (Pixels)	mAP @0.5:0.95	mAP @0.5	Time CPU b1 (ms)	Time V100 b1 (ms)	Time V100 b32 (ms)	Params (M)	FLOPS @640 (B)
YOLOv5n	640	28.0	45.7	45	6.6	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Hình 3: Chi tiết hiệu suất của từng phiên bản

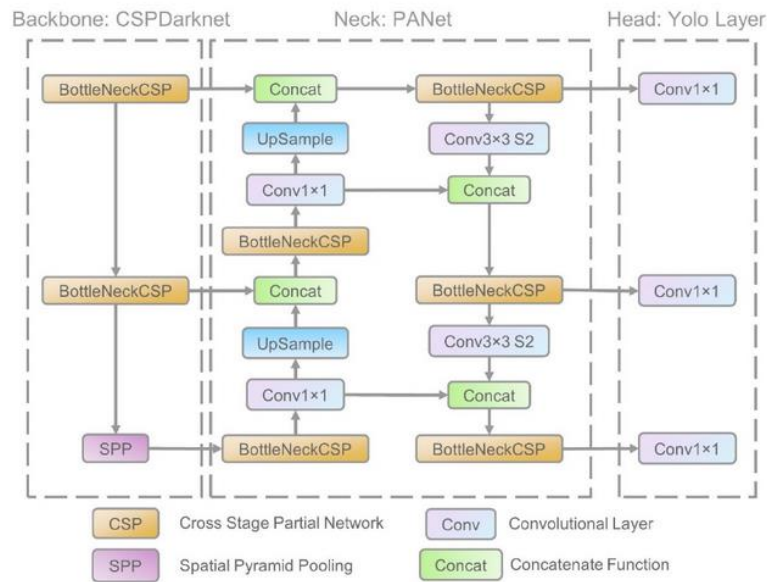


Hình 4: YOLOv5n6 đến YOLOv5x6 trên MS COCO

2. Kiến trúc YOLOv5

Kiến trúc của YOLOv5 bao gồm ba phần chính:

- **Backbone:** Đây là phần chính của mạng. Đối với YOLOv5, đường trục được thiết kế bằng cấu trúc CSP-Darknet53 mới, một bản sửa đổi của kiến trúc Darknet được sử dụng trong các phiên bản trước.
- **Neck:** Phần này kết nối **Backbone** và **Head**. Trong YOLOv5, cấu trúc SPPF và CSP-PAN mới được sử dụng.
- **Head:** Phần này chịu trách nhiệm tạo ra đầu ra cuối cùng. YOLOv5 sử dụng Đầu YOLOv3 cho mục đích này.



Hình 5: Kiến trúc mạng cho YOLOv5

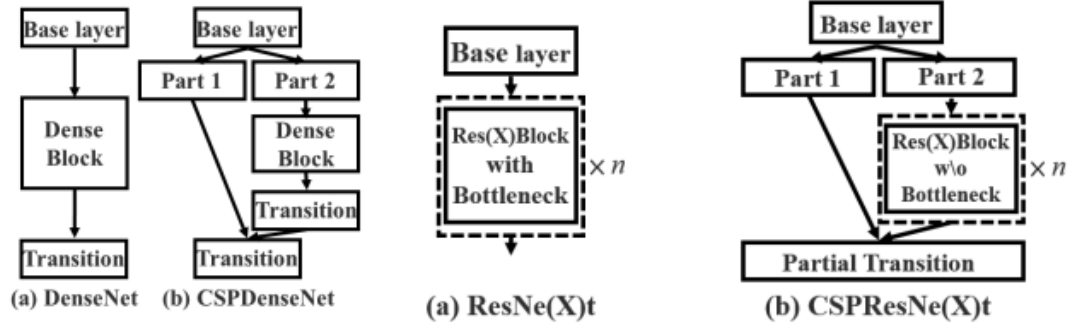
a) Backbone: CSP-Darknet53

YOLOv5 sử dụng CSP-Darknet53 làm **Backbone**. CSP-Darknet53 chỉ là mạng tích chập Darknet53 được sử dụng làm xương sống cho YOLOv3 mà các tác giả đã áp dụng chiến lược mạng **Cross Stage Partial** (CSP).

YOLO là một mạng sâu, nó sử dụng các khối dư và dày đặc để cho phép luồng thông tin đến các lớp sâu nhất và khắc phục vấn đề độ dốc biến mất. Tuy nhiên, một trong những lợi ích của việc sử dụng các khối dày đặc và dư thừa là vấn đề về độ dốc dư thừa.

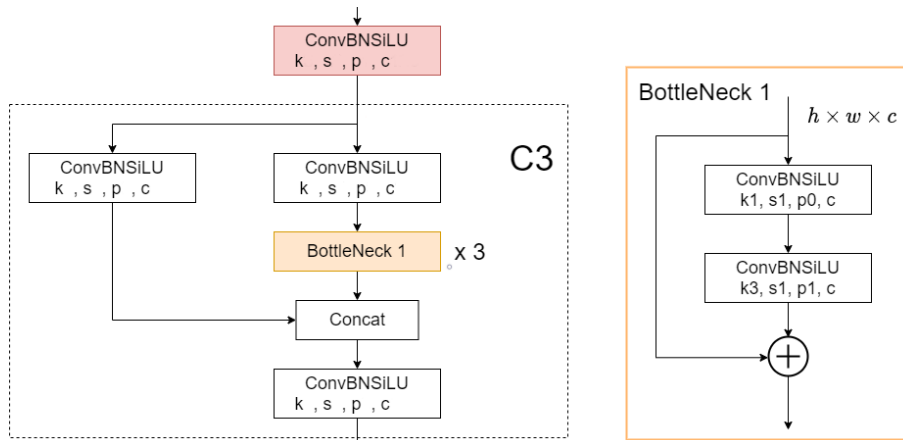
CSPNet giúp giải quyết vấn đề này bằng cách cắt bớt luồng gradient. Theo tác giả:

“Mạng CSP duy trì lợi thế về các đặc tính tái sử dụng tính năng của DenseNet và giúp giảm lượng thông tin độ dốc dư thừa bằng cách cắt bớt *gradient flow*.”



Hình 6: Áp dụng CSPNet cho ResNet và DenseNet

YOLOv5 sử dụng chiến lược CSPNet để phân chia feature map của base layer thành hai phần và sau đó hợp nhất chúng thông qua hệ thống phân cấp nhiều giai đoạn như trong hình bên dưới:



Hình 7: Kiến trúc khối C3 với Bottleneck

Áp dụng chiến lược này mang lại lợi ích lớn cho YOLOv5, vì nó giúp giảm số lượng tham số và giúp giảm lượng tính toán quan trọng (ít FLOPS hơn) dẫn đến tăng tốc độ suy luận, tham số quan trọng trong các mô hình phát hiện đối tượng thời gian thực.

b) Neck

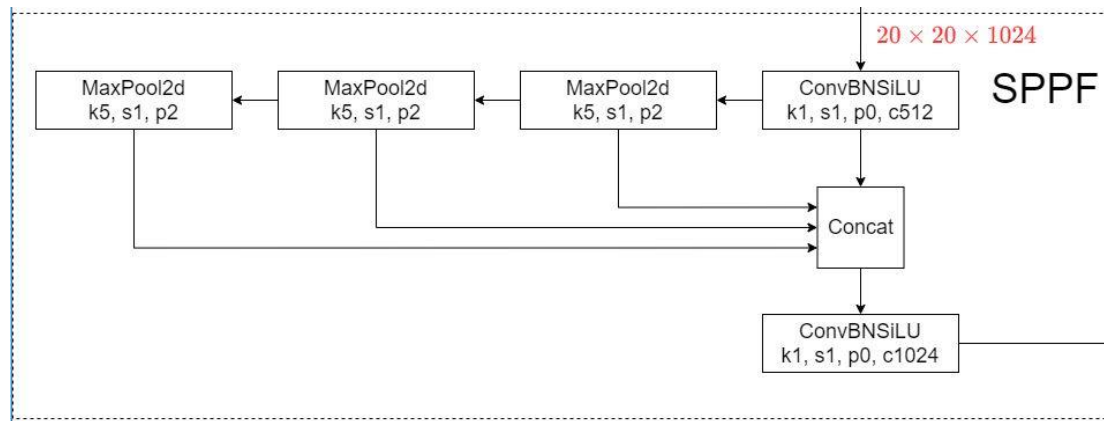
YOLOv5 mang lại hai thay đổi lớn cho the model neck. Đầu tiên, một biến thể của **Spatial Pyramid Pooling (SPP)** đã được sử dụng và the **Path Aggregation Network (PANet)** đã được sửa đổi bằng cách kết hợp BottleneckCSP trong kiến trúc của nó.

Path Aggregation Network (PANet)

PANet là một mạng feature pyramid, nó đã được sử dụng trong phiên bản trước của YOLO (YOLOv4) để cải thiện luồng thông tin và giúp định vị chính xác các pixel trong nhiệm vụ mask prediction. Trong YOLOv5, mạng này đã được sửa đổi bằng cách áp dụng chiến lược CSPNet cho mạng như thể hiện trong hình kiến trúc của mạng.

Spatial Pyramid Pooling (SPP)

Khối SPP thực hiện tổng hợp thông tin nhận được từ đầu vào và trả về đầu ra có độ dài cố định. Do đó, nó có ưu điểm là tăng đáng kể trường tiếp nhận và tách biệt các tính năng ngữ cảnh phù hợp nhất mà không làm giảm tốc độ của mạng.



Hình 8: Kiến trúc khối SPPF

Khối này đã được sử dụng trong các phiên bản trước của YOLO (yolov3 và yolov4) để tách các tính năng quan trọng nhất khỏi backbone, tuy nhiên trong YOLOv5(6.0/6.1) SPPF đã được sử dụng, đây chỉ là một biến thể khác của khối SPP, để cải thiện tốc độ của mạng.

c) Head

YOLOv5 sử dụng cùng head với YOLOv3 và YOLOv4. Nó bao gồm ba lớp tích chập dự đoán vị trí của các hộp giới hạn (x, y, chiều cao, chiều rộng), điểm số và các lớp đối tượng. Phương trình tính tọa độ đích cho các hộp giới hạn đã thay đổi so với các phiên bản trước, sự khác biệt được thể hiện trong hình dưới đây.

$$\begin{array}{ll} b_x = \sigma(t_x) + c_x & b_x = (2 \cdot \sigma(t_x) - 0.5) + c_x \\ b_y = \sigma(t_y) + c_y & b_y = (2 \cdot \sigma(t_y) - 0.5) + c_y \\ b_w = p_w \cdot e^{t_w} & b_w = p_w \cdot (2 \cdot \sigma(t_w))^2 \\ b_h = p_h \cdot e^{t_h} & b_h = p_h \cdot (2 \cdot \sigma(t_h))^2 \end{array}$$

(a) (b)

Hình 9: (a) Các phương trình được sử dụng trong các phiên bản trước (YOLOv2, YOLOv3).

(b) Các phương trình được sử dụng trong YOLOv5

3. Áp dụng vào bài toán

a) Chuẩn bị dữ liệu

Với dữ liệu đầu vào, YOLO yêu cầu định dạng dữ liệu **COCO-style** hoặc **YOLO-style**, trong đó mỗi ảnh sẽ có tệp nhãn .txt đi kèm.

Tệp .txt chứa thông tin theo định dạng:

class_id x_center y_center width height

Trong đó:

- **class_id**: ID của lớp ký hiệu tay (0, 1, 2...).
- **x_center, y_center**: Tọa độ trung tâm của bounding box (được chuẩn hóa giữa 0 và 1).
- **width, height**: Chiều rộng và chiều cao của bounding box (được chuẩn hóa).

b) Lựa chọn mô hình

Để thực hiện bài toán này, chúng tôi lựa chọn phiên bản yolov5n.

YOLOv5n (YOLOv5 nano) là phiên bản nhỏ gọn nhất trong dòng YOLOv5, được thiết kế để tối ưu hóa tốc độ và giảm yêu cầu tài nguyên hệ thống.

Mặc dù nhỏ gọn, mô hình này vẫn cung cấp độ chính xác đủ để giải quyết các bài toán nhận diện với số lượng lớp ít và yêu cầu độ phức tạp vừa phải, như bài toán nhận diện ký hiệu tay.

c) Fine-tuning

Dưới đây là các tham huấn luyện YOLOv5 tốt cho bài toán sau khi đã thử nghiệm ở nhiều giá trị khác nhau:

<i>img</i>	<i>batch</i>	<i>epochs</i>	<i>optimizer</i>	<i>weights</i>
640	16	50	SGD	yolov5s.pt

Bảng 4: Các tham số của YOLOv5 sử dụng cho bài toán

- Kích thước ảnh đầu vào được chọn là **640x640** pixel. Đây là kích thước tiêu chuẩn được tối ưu hóa cho YOLOv5.
- Kích thước batch được chọn là **16**, nghĩa là mỗi lần cập nhật trọng số, mô hình sẽ xử lý 16 ảnh.
- Số vòng lặp huấn luyện được đặt là **50 epochs**.
- Thuật toán tối ưu hóa được chọn là **SGD** (Stochastic Gradient Descent). So với Adam, SGD có xu hướng tránh bị overfitting trên tập dữ liệu nhỏ, phù hợp với bài toán nhận diện ký hiệu tay.
- Sử dụng trọng số khởi tạo từ mô hình **YOLOv5s**

C. YOLOv8

1. Giới thiệu

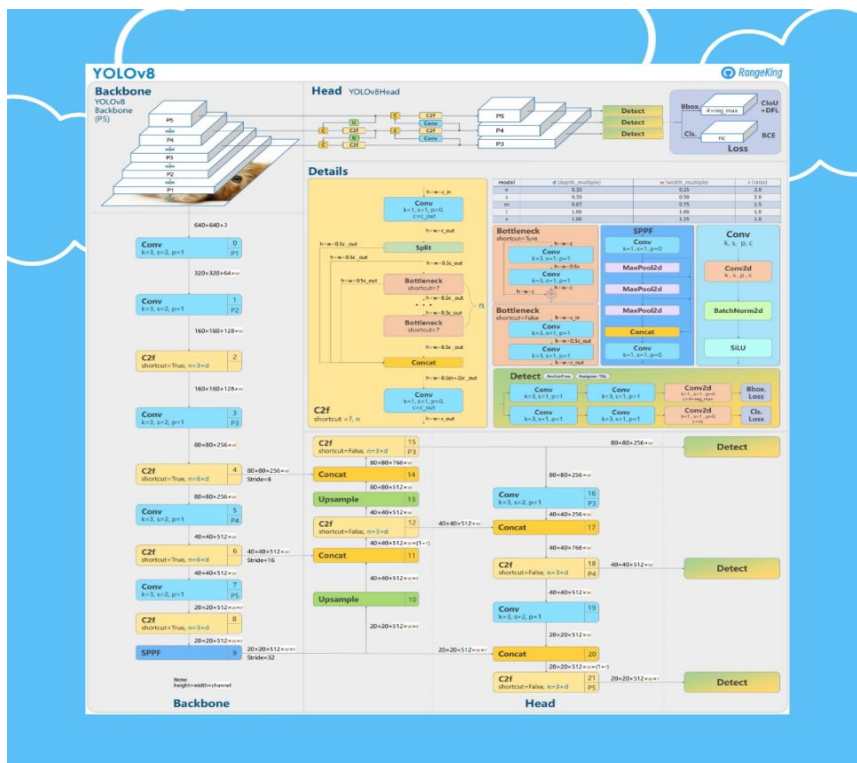
YOLOv8 là một thuật toán thị giác máy tính được sử dụng để phát hiện đối tượng. Mô hình này được phát triển dựa trên các phiên bản trước của YOLO (You Only Look Once – Nhận diện vật thể bằng một lần nhìn).

Ngoài ra, YOLOv8 còn được thiết kế để phát hiện đối tượng trong thời gian thực và có thể xử lý hình ảnh trong vài mili giây.

YOLOv8 đã được phát hành với năm phiên bản khác nhau:

Người mẫu	kích cỡ (điểm ảnh)	giá trị mAP 50-95	Tốc độ CPU ONNX (bệnh đa xơ cứng)	Tốc độ A100 TensorRT (bệnh đa xơ cứng)	tham số (M)	Thất bại (B)
YOLOv8n	640	18.4	142.4	1.21	3.5	10.5
YOLOv8s	640	27.7	183.1	1.40	11.4	29.7
YOLOv8m	640	33.6	408.5	2.26	26.2	80.6
YOLOv8l	640	34.9	596.9	2.43	44.1	167.4
YOLOv8x	640	36.3	860.6	3.56	68.7	260.6

Hình 10: Chi tiết hiệu suất từng phiên bản



Hình 11: Mô hình tổng quan kiến trúc Yolov8

2. Kiến trúc YOLOv8

a) Kiến trúc của YOLOv8 bao gồm ba phần chính:

- **Backbone:** YOLOv8 sử dụng xương sống CSPDarknet53 tùy chỉnh, sử dụng các kết nối một phần xuyên giai đoạn để cải thiện luồng thông tin giữa các lớp và nâng cao độ chính xác của phát hiện.
- **Neck:** Phần này kết nối **Backbone** và **Head**. Thay vì Mạng kim tự tháp tính năng (FPN) truyền thống, YOLOv8 sử dụng mô-đun C2f mới.
- **Head:** Là phần cuối cùng của mạng và có trách nhiệm tạo ra các đầu ra, chẳng hạn như hộp giới hạn và điểm tin cậy để phát hiện đối tượng.

b) Cách hoạt động:

- **Tạo hộp giới hạn:** Tạo các hộp giới hạn liên quan đến các đối tượng tiềm năng trong hình ảnh.
- **Chấm điểm độ tin cậy:** Gán điểm độ tin cậy cho mỗi hộp giới hạn để chỉ ra khả năng hiện diện của đối tượng.
- **Sắp xếp theo danh mục:** Sắp xếp các đối tượng được phát hiện trong các hộp giới hạn theo danh mục tương ứng của chúng.

Chi tiết: YOLOv8 sử dụng nhiều mô-đun phát hiện trong phần đầu, giúp dự đoán các hộp giới hạn, điểm đối tượng và xác suất lớp cho mỗi ô lưới trong bản đồ đặc điểm. Những dự đoán này sau đó được tổng hợp lại để có được những phát hiện cuối cùng.

c) Các cải tiến của YoloV8 so với YoloV5:

- **Backbone và Neck:**
 - **YOLOv5:**
 - Sử dụng kiến trúc **CSPDarknet53** làm backbone, kết hợp với các module CSP (Cross Stage Partial) để giảm số lượng tham số và cải thiện khả năng học tập của mạng.
 - Neck sử dụng **PANet** (Path Aggregation Network) để kết hợp thông tin từ các tầng khác nhau.

- **YOLOv8:**

- Thay backbone CSPDarknet bằng **CSPNext**, một biến thể của **ConvNeXt**, hiện đại hơn với thiết kế tập trung vào hiệu suất và tính tổng quát.
- Sử dụng các module **ELAN (Efficient Layer Aggregation Networks)** để tối ưu hóa khả năng học đặc trưng.

- **Cải tiến:**

CSPNext trong YOLOv8 giúp cải thiện khả năng tổng quát hóa, tăng tốc độ huấn luyện, và hiệu quả hơn khi xử lý các đặc trưng phức tạp.

- **Head:**

- **YOLOv5:**

- Sử dụng cơ chế phân tách lớp (class prediction), hộp (box regression), và objectness (xác suất đối tượng) từ các đặc trưng trích xuất trong backbone và neck.

- **YOLOv8:**

- Áp dụng **Decoupled Head**, tách riêng biệt quá trình dự đoán bounding box và class scores, giúp cải thiện hiệu suất.

- **Cải tiến:**

- Decoupled Head giúp YOLOv8 xử lý tốt hơn các bài toán phân loại đối tượng và dự đoán hộp giới hạn, đặc biệt là trong các tình huống phức tạp.

3. **Áp dụng vào bài toán**

a) **Chuẩn bị dữ liệu**

Với dữ liệu đầu vào, YOLO yêu cầu định dạng dữ liệu **COCO-style** hoặc **YOLO-style**, trong đó mỗi ảnh sẽ có:

- Tập nhãn .txt đi kèm.
- Tập .txt chứa thông tin theo định dạng:
class_id x_center y_center width height

Trong đó:

- **class_id**: ID của lớp ký hiệu tay (0, 1, 2...).
- **x_center, y_center**: Tọa độ trung tâm của bounding box (được chuẩn hóa giữa 0 và 1).
- **width, height**: Chiều rộng và chiều cao của bounding box (được chuẩn hóa).

b) Lựa chọn mô hình

Để thực hiện bài toán này, chúng tôi lựa chọn phiên bản yolov8n.

YOLOv8n (YOLOv8 nano) là phiên bản nhỏ gọn nhất trong dòng YOLOv8, được thiết kế để tối ưu hóa tốc độ và giảm yêu cầu tài nguyên hệ thống.

Mặc dù nhỏ gọn, mô hình này vẫn cung cấp độ chính xác đủ để giải quyết các bài toán nhận diện với số lượng lớp ít và yêu cầu độ phức tạp vừa phải, như bài toán nhận diện ký hiệu tay.

c) Fine-tuning

Dưới đây là các tham huấn luyện YOLOv8 được sử dụng trong bài toán:

<i>img</i>	<i>batch</i>	<i>epochs</i>	<i>optimizer</i>	<i>patience</i>	<i>weights</i>
640	16	100	Auto	20	Yolov8n.pt

Bảng 5: Các tham số của Yolov8 sử dụng cho bài toán

- Kích thước ảnh đầu vào được chọn là **640x640** pixel. Đây là kích thước tiêu chuẩn được tối ưu hóa cho YOLOv8.
- Kích thước batch được chọn là **16**, nghĩa là mỗi lần cập nhật trọng số, mô hình sẽ xử lý 16 ảnh.
- Số vòng lặp huấn luyện được đặt là **100 epochs**.
- Thuật toán tối ưu hóa được chọn là **Auto** (Thư viện Ultralytics sẽ tự chọn tham số tối ưu dựa trên thuật toán huấn luyện và các siêu tham số mặc định.).

- Patience: chọn tham số này bằng 20 thì cứ sau 20 epochs nếu kết quả mô hình không được cải thiện thì sẽ dừng quá trình training lại.

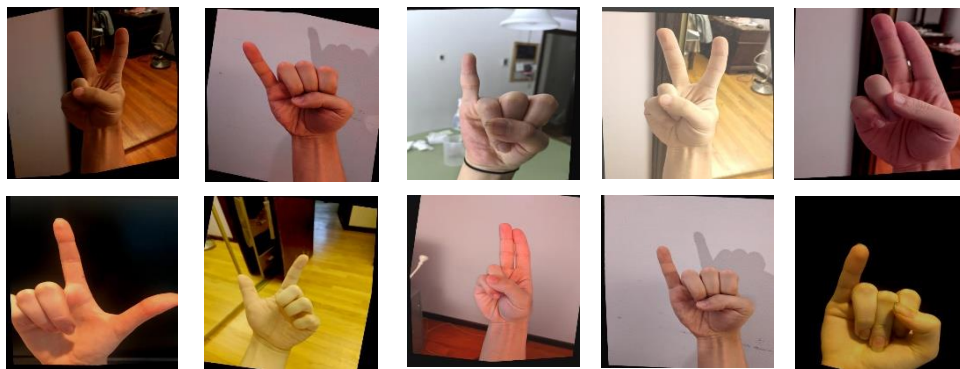
IV. THỰC NGHIỆM

A. Dataset

1. Giới thiệu

Việc chọn dataset là một phần quan trọng và căn bản của quá trình huấn luyện mô hình máy học hay học sâu. Dữ liệu đóng vai trò quan trọng trong việc định hình và đánh giá hiệu suất của mô hình dành cho bài toán. Chính vì thế, chúng tôi đã lựa chọn *American Sign Language Letters Computer Vision Project* để làm dataset cho bài toán của chúng tôi.

Bộ dữ liệu *American Sign Language Letters Computer Vision Project* trên **Roboflow** là một trong những bộ dữ liệu phổ biến được sử dụng trong lĩnh vực nhận dạng nhận diện ngôn ngữ ký hiệu (**Sign Language**). Được công bố trên **Roboflow** vào năm 2021, đây là một bộ dữ liệu lớn chứa các hình ảnh ký hiệu tay đã được gán.



Hình 12: Một số ảnh trong bộ dữ liệu

2. Một số thông tin cơ bản về bộ dữ liệu

- **Kích thước:** Dữ liệu bao gồm các ảnh màu 640x640 pixel chứa các kí hiệu tay. Bộ dữ liệu chứa khoảng 5256 hình ảnh kí hiệu tay với 5040 ảnh cho tập train, 144 ảnh cho tập validation và 72 ảnh cho tập test.
- **Loại kí hiệu:** Mỗi hình ảnh được gắn nhãn với một trong 26 chữ cái cơ bản:

["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z"]

- **Tăng cường dữ liệu:** khác với các bộ dữ liệu thông thường, bộ dữ liệu chúng tôi sử dụng đã được tăng cường với các thuộc tính sau:
 - Đầu ra trên mỗi training example: 10
 - Cắt: zoom tối thiểu 0%, tối đa 50%
 - Xoay: từ -10° đến +10°
 - Cắt: ±10° ngang, ±10° dọc
 - Thang độ xám: áp dụng cho 10% hình ảnh
 - Màu sắc: giữa - 25° và +25°
 - Độ bão hòa: từ -25% đến +25%
 - Độ sáng: từ -25% đến +25%
 - Độ phơi sáng: từ -15% đến +15%
 - Độ mờ: lên đến 1,25px

B. Độ đo

1. Precision

Precision đại diện cho độ tin cậy của mô hình. Tức là trong N lần dự đoán thì mô hình sẽ dự đoán chính xác bao nhiêu phần trăm.

2. Recall

Recall đại diện cho độ nhạy của mô hình. Tức là sẽ cho biết mô hình có thể dự đoán chính xác được bao nhiêu mẫu trong dữ liệu được cho.

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

Hình 12: Công thức tính Precision

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

Hình 13: Công thức tính Recall

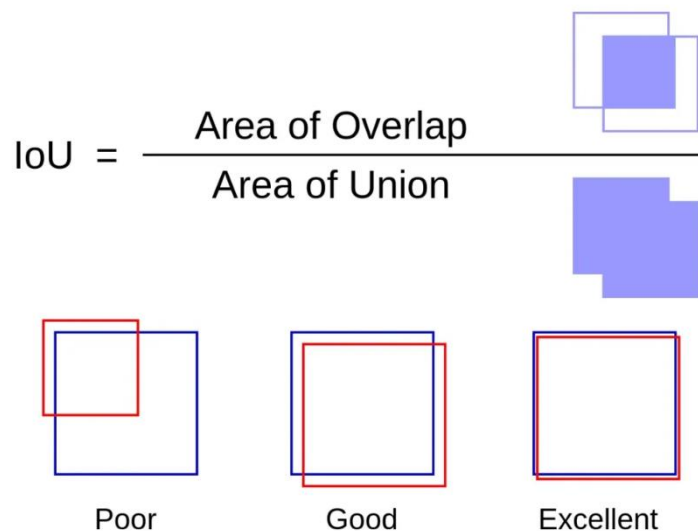
Trong đó:

- TP (True Positive): Một bounding box dự đoán được coi là True Positive nếu dự đoán cùng lớp với ground truth bbox và có $IoU \geq$ ngưỡng.
- FP (False Positive): Một bounding box dự đoán không khớp với bất kỳ bbox ground truth nào hoặc dự đoán cùng lớp với ground truth nhưng $IoU <$ ngưỡng.
- FN (False Negative): Một ground truth bbox được coi là FN nếu mô hình không phát

3. Intersection Over Union (IOU)

Ở hai độ đo trên, chúng tôi có đề cập tới một khái niệm IoU. Khi mô hình dự đoán ra được một bounding box, ta cần xác định xem bounding box đó khớp với ground truth bbox hay không.

Để đo lường độ "khớp" này, người ta đề xuất một thang đo gọi là IoU. IoU được tính bằng tỉ lệ giữa diện tích giao trên diện tích hợp của hai bounding box.

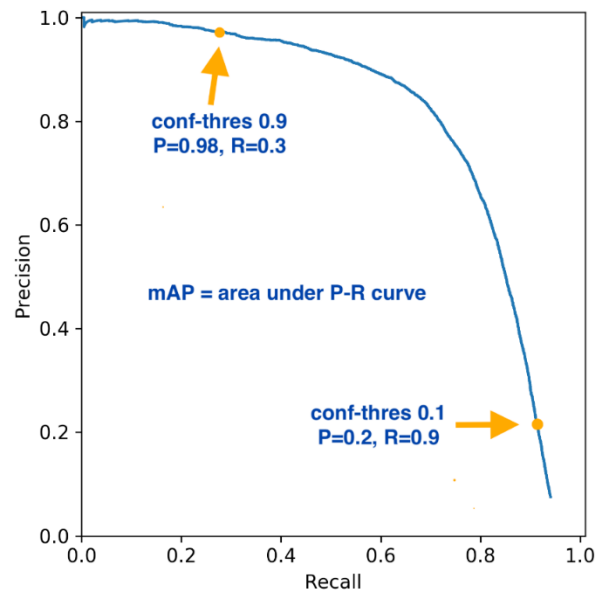
$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Poor Good Excellent

Hình 14: Công thức tính IoU

4. mean Average Precision (mAP)

Chúng tôi sử dụng chỉ số mAP (mean Average Precision) để đánh giá hiệu suất của mô hình. Mean Average Precision (mAP) là một thước đo phổ biến trong lĩnh vực thị giác máy tính, dùng để đánh giá độ chính xác của các mô hình phân loại và nhận diện đối tượng. Thước đo này kết hợp giữa Precision (độ chính xác) và Recall (tỉ lệ nhận diện) nhằm đánh giá khả năng của mô hình trong việc xác định và định vị các đối tượng trên hình ảnh



Hình 15: Ảnh minh họa cho mAP

C. Kết quả

Model	Precision	Recall	mAP (IoU = 0.5)	mAP (IoU = 0.5 – 0.95)
SSDLite320	0.890	0.812	0.907	0.763
SSDLite320 with freezing layers	0.897	0.820	0.913	0.765
Yolov5n	0.908	0.884	0.949	0.634
Yolov8n	0.938	0.892	0.969	0.778

Bảng 6: kết quả thực nghiệm trên tập dữ liệu *American Sign Language Letters Computer Vision Project*

Kết quả thực nghiệm cho thấy **Yolov8n** là mô hình hiệu quả nhất trong bài toán nhận diện ký hiệu tay, vượt trội ở tất cả các chỉ số đánh giá, bao gồm Precision, Recall, và mAP. So với các mô hình khác, **Yolov8n** không chỉ đạt độ chính xác cao (**Precision = 0.938**), mà còn có khả năng bao quát tốt (**Recall = 0.892**) và định vị chính xác các đối tượng trên ảnh, thể hiện qua mAP (**IoU = 0.5–0.95**) là **0.778**.

Trong khi đó, **SSDLite320** với kỹ thuật đóng băng lớp có sự cải thiện nhẹ về hiệu suất so với phiên bản gốc, chứng minh rằng việc áp dụng đóng băng lớp có thể giúp tận dụng tốt hơn các đặc trưng đã học. Tuy nhiên, hiệu quả tổng thể của **SSDLite320** vẫn thấp hơn đáng kể so với **Yolov8n**.

Yolov5n, mặc dù có kết quả tương đối tốt, nhưng hiệu suất thấp hơn cả **SSDLite320** (với đóng băng lớp) ở mAP (**IoU = 0.5–0.95**), cho thấy khả năng định vị đối tượng không mạnh mẽ bằng các mô hình khác.

Nhìn chung, **Yolov8n** đạt được hiệu quả cao trong ba mô hình, là lựa chọn tối ưu nhất cho bài toán.

V. ĐÁNH GIÁ

A. Ưu điểm

Các mô hình như YOLOv5, YOLOv8 và SSD đã chứng tỏ hiệu quả trong việc nhận diện hầu hết các ký hiệu tay có trong tập dữ liệu huấn luyện. Điều này cho thấy khả năng tổng quát hóa tốt của các mô hình đối với các dạng ký hiệu đa dạng, giúp đảm bảo hiệu suất nhận diện cao.

Một trong những điểm mạnh đáng chú ý là khả năng làm việc ổn định trong các môi trường có độ sáng khác nhau. Điều này rất quan trọng trong thực tế, nơi điều kiện ánh sáng có thể thay đổi liên tục. Dù là ánh sáng mạnh, yếu hay không đồng đều, các mô hình vẫn có thể nhận diện chính xác các ký hiệu tay.

Các mô hình không bị giới hạn bởi tay của một cá nhân cụ thể, mà có thể nhận diện chính xác các ký hiệu từ tay của nhiều người khác nhau. Điều này chứng minh rằng mô hình đã học được các đặc trưng chung, thay vì phụ thuộc vào các yếu tố cá nhân như kích thước hay màu da của tay.

Tốc độ xử lý của mô hình như YOLOv8 gần như đạt mức real-time, giúp ứng dụng dễ dàng được triển khai trong các hệ thống yêu cầu tốc độ cao.

B. Hạn chế

Mặc dù hiệu suất tổng thể tốt, các mô hình đôi khi vẫn phát hiện sai các ký hiệu có hình dáng tương tự nhau. Ví dụ, các ký hiệu có sự khác biệt nhỏ về góc độ hoặc vị trí ngón tay dễ gây nhầm lẫn.

Một vấn đề khác là khi người dùng thực hiện ký hiệu không đúng tư thế như trong dữ liệu huấn luyện, mô hình có thể không nhận diện được. Điều này thường xảy ra khi người dùng thực hiện ký hiệu nhanh, thiếu chính xác hoặc ở góc nhìn khác so với dữ liệu gốc. Đây là hạn chế quan trọng vì trong thực tế, khó đảm bảo rằng tất cả các ký hiệu đều được thực hiện đúng theo chuẩn mẫu. Hạn chế này đặt ra thách thức trong việc mở rộng khả năng của mô hình để xử lý các trường hợp không chuẩn mực.

Hiệu quả của các mô hình này phụ thuộc lớn vào độ phong phú và chất lượng của dữ liệu huấn luyện. Nếu tập dữ liệu chưa bao quát hết các trường hợp đặc biệt hoặc chưa đa dạng hóa đủ về góc nhìn, độ sáng, hoặc người thực hiện, hiệu suất của mô hình sẽ bị hạn chế.

VI. TÀI LIỆU THAM KHẢO

- 1) *AI, M. (2019). Xây dựng hệ thống nhận diện ngôn ngữ ký hiệu tay – Mì AI. Retrieved from Youtube*
- 2) *David Lee. (2022, 10 12). sign language detection Computer Vision Project. Retrieved from universe.roboflow.com*
- 3) *Huy, T. V. (n.d.). mAP - mean Average Precision for Object Detection. Retrieved from huytranvan2010.github.io*
- 4) *Lee, D. (n.d.). Retrieved from universe.roboflow.com*
- 5) *Rahima Khanam* and Muhammad Hussain. (2024, 6 30). What is YOLOv5: A deep look into the internal features of the popular object detector.*
- 6) *Ultralytics. (n.d.). Tài liệu YoloV8. Retrieved from docs.ultralytics.com*
- 7) *Ultralytics. (n.d.). Ultralytics YOLOv5 Ngành kiến trúc. Retrieved from docs.ultralytics.com*
- 8) *viblo. (n.d.). viblo.asia.*