ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Đồ án cuối kì

Môn: Máy học

NHẬN DIỆN ĐEO KHẨU TRANG QUA CAMERA GIÁM SÁT

Giảng viên hướng dẫn PGS. TS Lê Đình Duy
ThS. Phạm Nguyễn Trường An

Thành viên

Trương Chí Diễn	19520464
Trần Hoàn Đức Duy	19521434
Trịnh Minh Hoàng	19521547

Mục lục

I. Giới thiệu bài toán

Đại dịch COVID-19 vẫn chưa có dấu hiệu hạ nhiệt và liên tục xuất hiện những biến thể mới ngày càng nguy hiểm và khó lường. Mặc dù đã có những phương pháp đã được đề ra để đối phó với đại dịch, điển hình là việc đeo khẩu trang khi ra ngoài và đến nơi đông người. Tuy nhiên vẫn còn những hạn chế về ý thức của cộng đồng về việc tự giác thực hiện. Lấy ý tưởng từ cuộc thi Datacomp do FPT tổ chức vào tháng 10/2021, chúng tôi giải quyết bài toán nhận diện đeo khẩu trang nơi đông người thông qua camera giám sát bằng mô hình YOLOv5.

Input: Một tấm ảnh hoặc một tập ảnh hoặc một video từ camera giám sát.

Output: Input đã được đánh bounding box vào gương mặt của những người có đeo hoặc không đeo khẩu trang.

II. Những vấn đề liên quan

1. Thiếu dữ liệu

Bộ dữ liệu từ cuộc thi chỉ bao gồm 1064 ảnh được trích xuất từ camera giám sát của một công ty. Số ảnh này quá ít để huấn luyện một mô hình học sâu với hàng chục triệu trọng số.

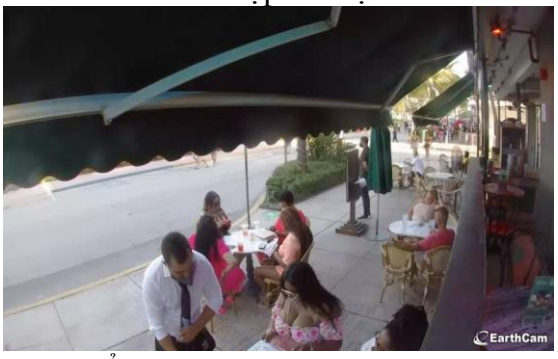
Do đó, chúng tôi quyết định thu thập thêm dữ liệu cho bài toán. Cụ thể, chúng tôi tham khảo các các bài toán có liên quan đến nhận diện đeo khẩu trang hiện có trên Google nhưng đa phần, các tập dữ liệu trên đều có ngữ cảnh không phù hợp với bài toán của chúng tôi.

Hình 1. Data từ một cuộc thi trên Kaggle về nhận diện đeo khẩu trang. Nhưng góc nhìn chính diện trong khi bài toán chúng tôi đang giải quyết là góc nhìn từ camera giám sát.

Sau đó chúng tôi tham khảo các bài báo trên Scholar thì tìm được một số nguồn dữ liệu liên quan đến camera giám sát như WILDTRACK, BrainWash, Oxford Town Center. Và trang web EarthCam chuyên chia sẻ dữ liệu các camera giám sát được công khai tại khắp nơi trên thế giới. Sau khi duyệt qua các một số camera trên EarthCam, chúng tôi chỉ chọn được 1 camera có góc nhìn tương tự như bài toán và có thể sử dụng được.

Hình 2a. Ảnh từ tập dữ liệu BrainWash

Hình 2b. Ảnh từ tập dữ liệu WILDTRACK



Hình 2c. Ảnh từ quán café Miami trên trang EarthCam

Hình 2d. Ảnh từ bộ dữ liệu Oxford Town Center

Sau khi quá trình tách khung hình từ video và chọn lọc thì chúng tôi thu được thêm 1152 ảnh (WILDTRACK: 572, BrainWash: 219, café Miami:265, Oxford Town

Center: 96), nâng tổng số ảnh trong tập dữ liệu lên thành 2,216 ảnh. Với số lượng này, chúng tôi tự tin rằng mô hình có thể học được trên nhiều ngữ cảnh hơn so với số lượng dữ liệu ít ỏi ban đầu.

2. Mất cân bằng dữ liệu

Trong tập dữ liệu ban đầu, số lượng lớp có đeo khẩu trang gấp đôi so với lớp không đeo khẩu trang. Tuy nhiên, sau quá trình thu thập thêm dữ liệu thì số lượng không đeo khẩu trang chỉ chưa bằng phân nửa lớp không đeo. Nguyên nhân là do dữ liệu thu thập đa số là dữ liệu có từ những năm trước đại dịch và mọi người thường không có thói quen đeo khẩu trang khi ra đường. Mặc dù vậy, dữ liệu từ quán café Miami được lấy trực tiếp trong năm 2020 – thời điểm dịch COVID – 19 đang bùng phát toàn cầu và mọi người bắt đầu mang khẩu trang khi ra đường nên chúng tôi lấy thêm được một lượng lớp có mang khẩu trang.

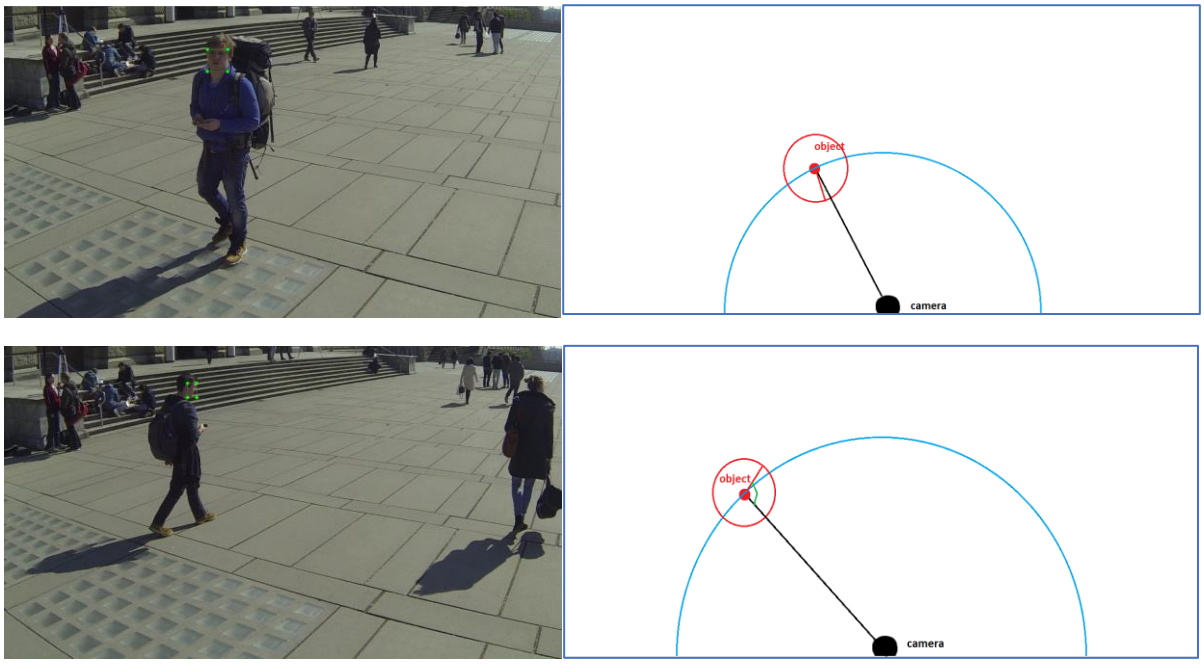
3. Quy cách gắn nhãn

Theo đúng tinh thần của cuộc thi Datacomp, chúng tôi nhận thức được sự ảnh hưởng xấu của dữ liệu không tốt lên mô hình. Cụ thể là việc gắn nhãn dữ liệu theo nhiều cách khác nhau sẽ khiến mô hình bị bối rối và học không tốt. Việc bất đồng trong cách gắn nhãn chủ yếu là do việc gắn nhãn được đảm nhận bởi cả nhóm nên nếu không có một quy cách chung thì dữ liệu sẽ được gắn nhãn theo nhiều cách bởi các thành viên khác nhau. Bên cạnh đó, chúng tôi sử dụng công cụ labelImg để gắn nhãn cho dữ liệu.

Quy tắc gắn nhãn của chúng tôi tuân theo bộ dữ liệu ban đầu, cụ thể là:

- Đối với các khuôn mặt chính diện hoặc có góc nhìn nghiêng trong khoảng $[-90°; 90°]$, thấy rõ mặt thì sẽ được gắn nhãn từ trán xuống dưới phần cằm, bỏ phần tai.
- Các khuôn mặt có góc nghiêng lớn hơn 90° nhưng vẫn thấy rõ được có đeo khẩu trang hay không thì vẫn được gắn nhãn từ trán xuống dưới cằm và lấy luôn phần tai.
- Đối với các khuôn mặt bị khuất một phần nhưng vẫn nhận diện được bởi mắt người thì sẽ gắn nhãn phần không bị khuất.
- Các khuôn mặt bị khuất quá nhiều, góc quay quá nhiều không thể nhận diện được sẽ bị bỏ qua, không gắn nhãn.
- Những khuôn mặt nhỏ và quá mờ cũng sẽ bị bỏ qua.

Góc nhìn là góc được tạo bởi đoạn thẳng màu đen và đoạn thẳng màu đỏ. Trong đó, đoạn thẳng màu đen chỉ hướng từ người đến camera và đoạn thẳng màu đỏ chỉ hướng mặt người đang nhìn trên hình chiếu bằng.



Hình 3. Mô phỏng cách xác định góc nhìn của mặt

III. Hướng tiếp cận

Lấy cảm hứng từ cuộc thi Datacomp, chúng tôi sử dụng YOLOv5 để giải quyết bài toán này. YOLOv5 là một mô hình thuộc gia đình YOLO (You Look Only Once).

1. Tác giả và các phiên bản

Tác giả của YOLOv1-3 là Joseph Redmon và Ali Farhadi.

YOLOv1: You Only Look Once: Unified, Real-Time Object Detection, YOLOv1 là công cụ dò tìm vật thể thời gian thực, bằng cách dự đoán các Bouding Box xung quanh vật thể và cho biết rằng đối tượng đó là gì. Tốc độ phát hiện 22ms/img. Bài báo hoàn chỉnh công bố tháng 6 năm 2016 [1].

YOLOv2: Yolo9000: Better, faster, stronger. In Computer Vision and Pattern Recognition (CVPR). YOLOv2 là phiên bản cải tiến lên từ YOLOv1, để cải thiện về tốc độ và sự chính xác YOLOv2 sử dụng phương pháp training mới gồm Anchor Box và Multi-scale có thể train ở các kích cỡ hình ảnh khác nhau mục đích để mạng có thể học theo các phương thức khác nhau. Bài báo công bố năm 2017 tại IEEE Conference [2].

YOLOv3: An incremental improvement. YOLOv3 là 1 cải tiến tiếp theo được ra đời, nó hiệu quả hơn do phiên bản đã thay đổi nhỏ về cách thiết kế và đem lại kết quả chính xác hơn. Ở kích thước 320 × 320, YOLOv3 chạy trong 22 ms ở 28,2 mAP, chính xác như SSD nhưng nhanh hơn ba lần. Bài báo công bố năm 2018 [3].

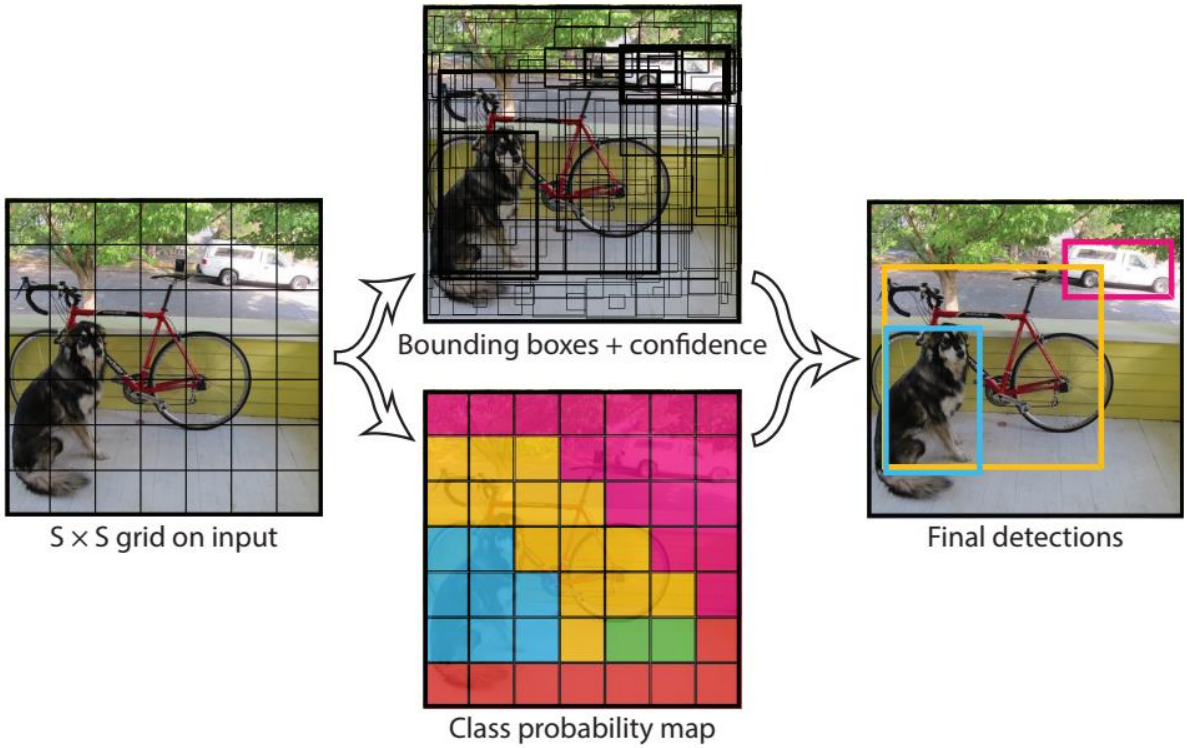
Joseph Redmon đã nói rằng sẽ ngừng nghiên cứu về Thị giác máy tính vì cách công nghệ này đang được sử dụng cho các ứng dụng quân sự và những lo ngại về quyền riêng tư có tác động đến xã hội.

YOLOv4: Optimal Speed and Accuracy of Object Detection của Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. Với mục tiêu tạo ra object detector hoạt động nhanh và đòi hỏi sự chính xác cao, được tối ưu hóa về tính toán khi thực hiện phát hiện vật thể. Bài báo được công bố vào ngày 23 tháng 4 năm 2020 tại Cornell University [4].

YOLOv5: Ra mắt một khoảng thời gian ngắn sau khi YOLOv4 công bố, sử dụng Pytorch framework. Tác giả là Glenn Jocher, tuy nhiên vẫn chưa có bài báo chính thức nào về nó của tác giả trên các hội nghị.

2. Cách hoạt động, Input, Output:

Đầu vào của mô hình là một ảnh, mô hình sẽ nhận dạng ảnh đó có đối tượng nào hay không, sau đó sẽ xác định tọa độ của đối tượng trong bức ảnh. Ảnh đầu vào được chia thành thành $S \times S$ ô thường thì sẽ là 3×3, 7×7, 9×9... việc chia ô này có ảnh hưởng tới việc mô hình phát hiện đối tượng.



Hình 4. Mô phỏng ý tưởng chung của YOLO. Nguồn: Viblo

Với Input là 1 ảnh, đầu ra mô hình là một ma trận 3 chiều có kích thước S×S×K(5+C) với số lượng tham số mỗi ô là K(5+C) với K và C lần lượt là số lượng Box và Class

mà mỗi ô cần dự đoán. Số 5 trong công thức trên bao gồm (x,y,w,h, object). Trong đó, x, y, w, h lần lượt là tọa độ tâm, chiều rộng và chiều dài của bounding box, còn object sẽ cho biết bounding box đó có chứa vật hay không (0 hoặc 1).

3. Điểm mạnh

Như tên gọi của nó, các mô hình YOLO nói chung đều có ưu thế về tốc độ nhận diện. Các phiên bản của YOLO đều đạt state-of-the-art về tốc độ vào thời điểm chúng ra mắt. Điều này giúp YOLO được tin dùng vào những tác vụ đòi hỏi thực hiện trên thời gian thực. Điển hình là phiên bản YOLOv5-nano được giới thiệu trên kho lưu trữ Github có thể xử lý 1 hình ảnh 640x640 chỉ trong 6.3 ms (tương đương 158 FPS) trên GPU NVIDIA V100 với batch size = 1. Thậm chí là 0.6 ms (tương đương 1666 FPS) khi tăng batch size = 32.

4. Điểm yếu

Mặc dù rất nhanh nhưng đó cũng là rào cản để YOLO đạt được độ chính xác cao. Chúng tôi không nói độ chính xác của YOLO là thấp. Tuy nhiên, nếu so sánh với các state-of-the-art khác thì độ chính xác của YOLO vẫn chưa sánh bằng. Cụ thể là trong top 10 mAP được đánh giá trên tập COCO test-dev và COCO minival không có sự xuất hiện của YOLO, ngoại trừ tập PASCAL VOC 2007 thì YOLOv3 vẫn đang đứng top 4 với mAP 83.68% [8].

5. Tại sao lại là YOLOv5?

Vì bài toán chúng tôi muốn giải quyết phải yêu cầu tốc độ có thể đáp ứng thời gian thực, tuy nhiên độ chính xác cũng không được quá thấp. Các mô hình hiện nay có thể đứng top với độ chính xác rất cao tuy nhiên khi nói về tốc độ thì YOLOv4 vẫn đang là mô hình nhận diện vật thể nhanh nhất [7]. Ngoài ra, với mức độ phủ sóng của mình, YOLO đang là một mô hình rất ổn định do được cập nhật và sửa lỗi thường xuyên. Theo như so sánh được đính kèm trong kho mã nguồn của YOLOv5, nó không những nhanh hơn YOLOv4 mà còn nhẹ hơn khoảng 90%, tuy nhiên điều này vẫn còn gây tranh cãi và chưa có bài báo chính thức nào để có thể xác thực [6]. Hơn nữa, mã nguồn của YOLOv5 cung cấp nhiều công cụ để có thể sử dụng cho các mục đích khác như: xuất các file model để xây dựng các ứng dụng demo.

6. Cách sử dụng

Chúng tôi sử dụng mã nguồn YOLOv5 được cung cấp công khai trên Github.

B1: Clone mã nguồn và cài đặt các thư viện cần thiết qua lệnh

```
git clone https://github.com/ultralytics/yolov5
cd yolov5
pip install -qr requirements.txt
```

B2: Chuẩn bị dữ liệu theo format của YOLO

B3: Tạo file data config để thiết lập các cài đặt về bộ dữ liệu. Chỉnh sửa file model config (nếu cần) để cấu hình các thông số của mô hình.

B4: Huấn luyện mô hình bằng lệnh

```
python train.py --img {size ảnh muốn đưa vào mô hình} \
--batch {batch size} --epochs {số epochs} \
--data {đường dẫn tới file data config } \
--weights {đường dẫn tới trọng số pretrained}  \
--cache –name {tên của phiên làm việc}
```

B5: Dự đoán bằng lệnh

```
python detect.py --weights {đường dẫn tới file trọng số} \
--img {size ảnh} --conf {ngưỡng confidence score} \
--source {đường dẫn tới ảnh muốn dự đoán}
```

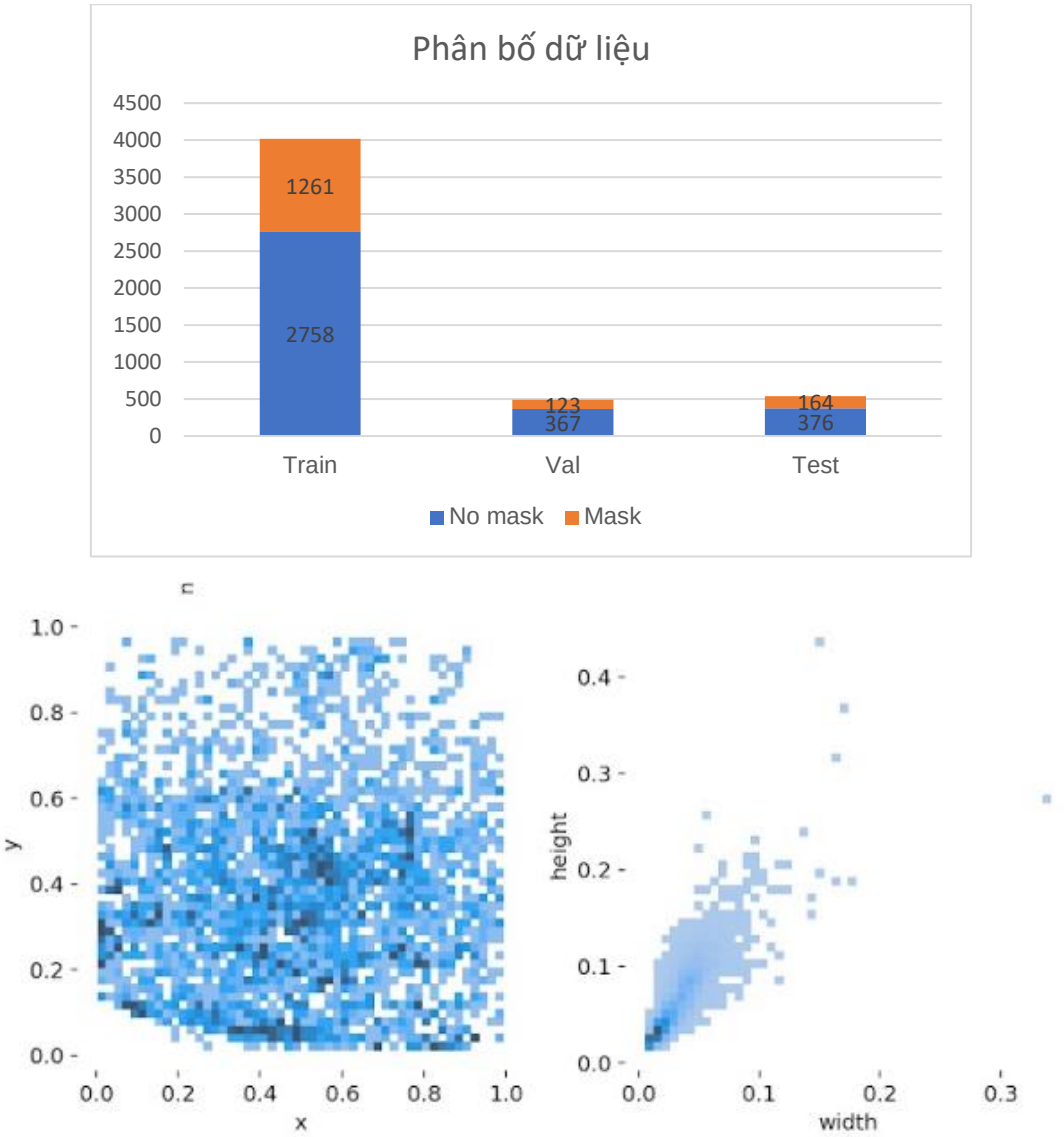
7. So sánh:

Chúng tôi huấn luyện 2 mô hình phát hiện vật thể khác trên cùng tập dữ liệu để có cái nhìn tổng quát và dễ so sánh. Mô hình chúng tôi chọn là RetinaNet và Faster R-CNN

IV. Thực nghiệm

1. Bộ dữ liệu:

Bộ dữ liệu bao gồm 2.216 ảnh, được chia theo tỉ lệ train:val:test là 8:1:1. Phân bố dữ liệu như biểu đồ bên dưới.



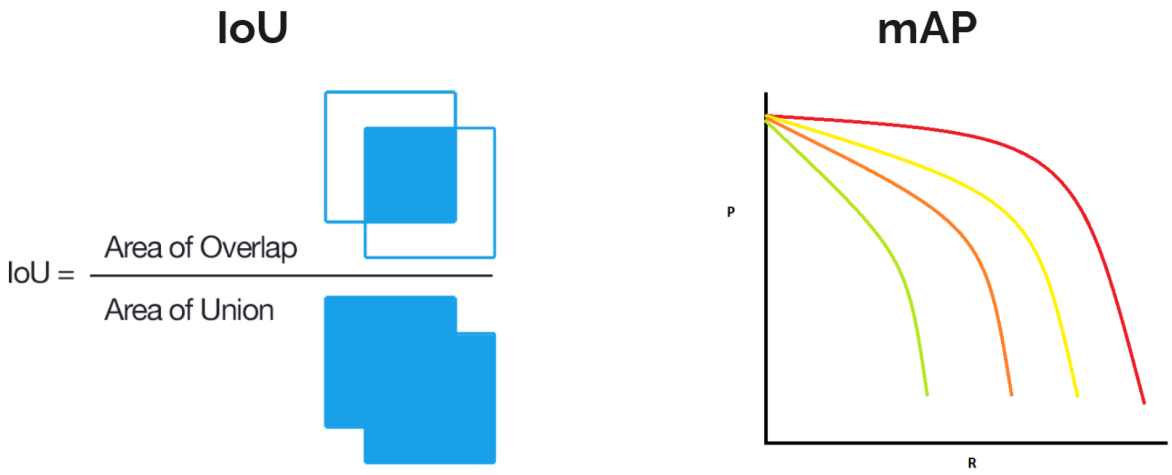
Hình 5. Trái: phân bố của đối tượng trong khung hình. Phải: Phân bố tỉ lệ bouding box

2. Phương pháp đánh giá:

IoU là độ đo được tính bằng diện tích phần giao giữa bounding box dự đoán và bounding box thực tế chia cho phần hợp giữa chúng (được mô tả như hình bên dưới).

Precision curve là đường cong thể hiện độ đánh đổi giữa Precision và Recall khi thay đổi ngưỡng Confidence score.

Chúng tôi sử dụng điểm mAP để đánh giá bài toán này. Điểm mAP được tính toán bằng trung bình phần diện tích bên dưới Precision Curve. Tuy nhiên, với các ngưỡng IoU khác nhau sẽ có 1 Precision curve khác nhau tương ứng với điểm mAP khác nhau. Chúng tôi sử dụng điểm mAP@0.5 và mAP@0.5:0.95 và cả tốc độ nhận diện mỗi ảnh (FPS) để đánh giá mô hình.

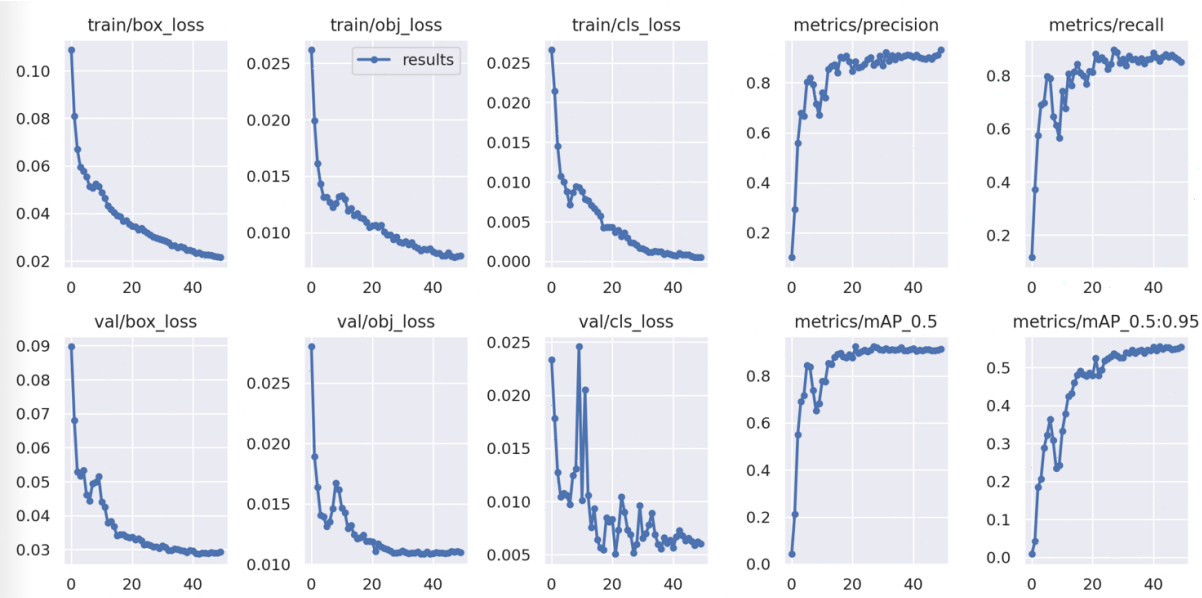


Hình 6. Mô tả độ đo IoU và mAP

3. Huấn luyện:

Trong thực nghiệm này, các mô hình đã được pretrained trên bộ dữ liệu COCO, sau đó được fine-tuning với bộ dữ liệu khẩu trang.

Train trên GPU: Tesla P100-PCIE-16GB.



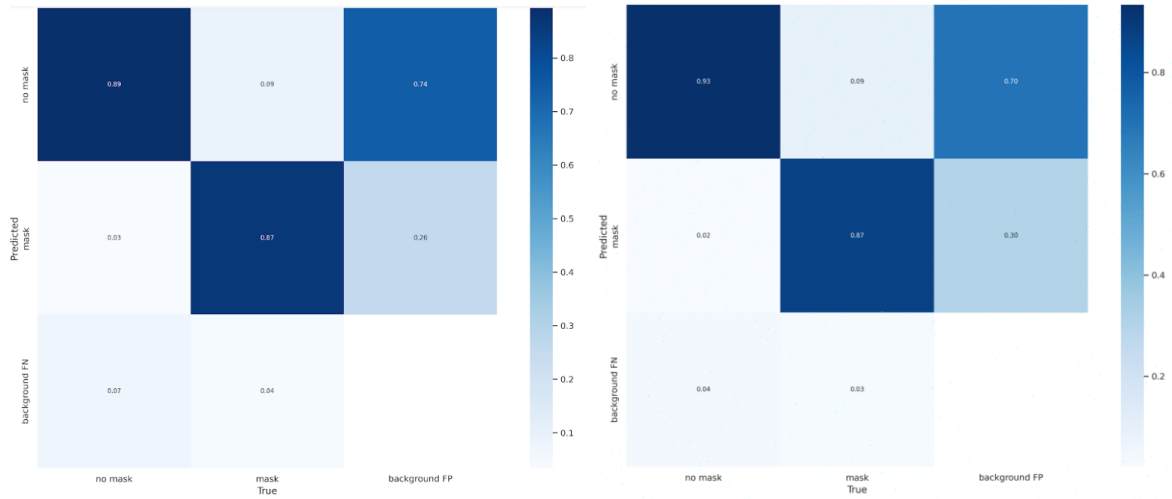
Hình 7: Các thông số giám sát quá trình huấn luyện.

Trong quá trình huấn luyện losses của tập train vẫn giảm nhưng losses và mAP trên bộ val không còn giảm nữa, nếu tiếp tục train sẽ bị overfit.

4. Kết quả:

Phương pháp	mAP@0.5	mAP@0.5:0.95	Inference time (FPS)
RetinaNet	0.865	0.496	14.7
Faster R-CNN	0.888	0.509	13.7
YOLOv5	0.899	0.535	**48.1**
YOLOv5 (bỏ Oxford Town Center)	**0.918**	**0.547**	

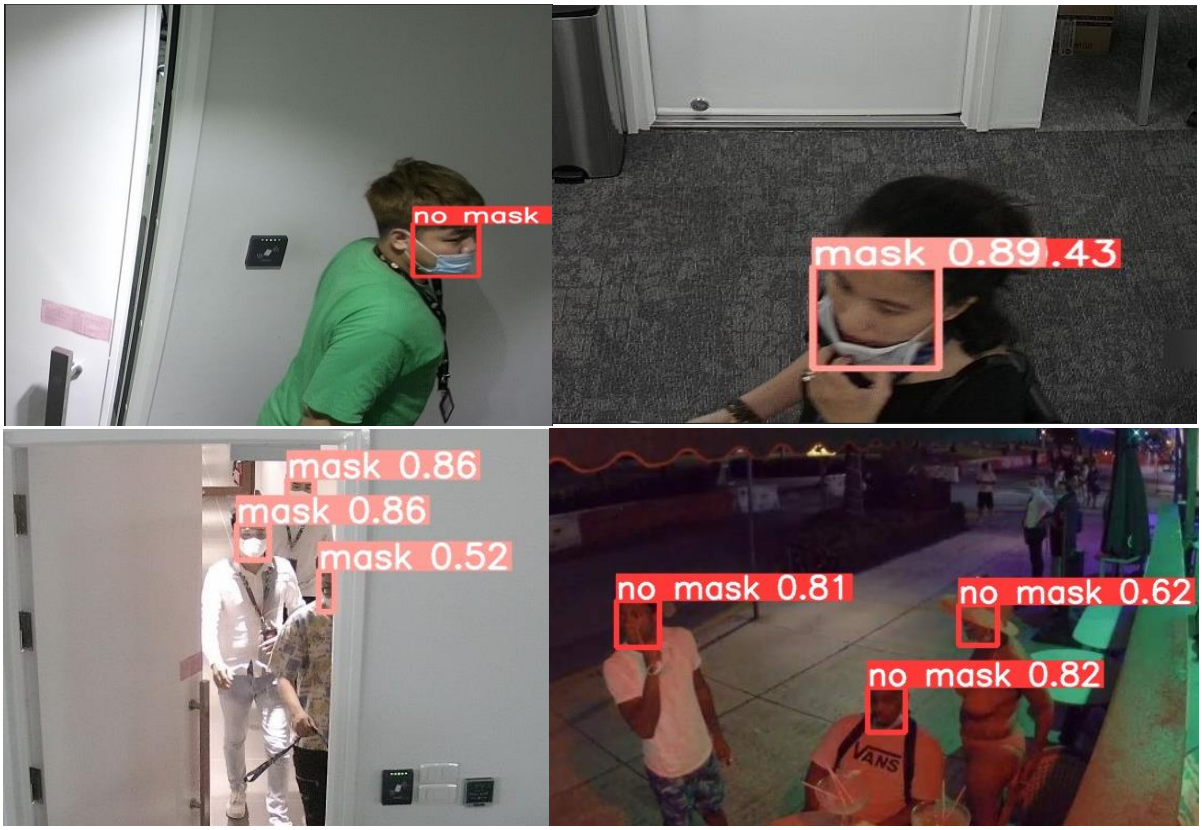
Bảng 1 Kết quả trên các phương pháp đã thực hiện



Hình 8: Confusion matrix của YOLO trên bộ test. Trái: Train trên toàn tập dữ liệu đã thu thập. Phải: Loại bỏ bộ Oxford Town Center.

Từ kết quả thực nghiệm, ta thấy YOLOv5 hoàn toàn tốt hơn các mô hình còn lại trên tất cả các độ đo. Với điểm mAP@0.5 ≈ 90%, đây là con số có thể chấp nhận được cho bài toán này nếu không quá khắt khe. Vì bộ dữ liệu Oxford Town Center có bối cảnh khác biệt khá lớn với các bộ dữ liệu còn lại, nên khi bỏ đi thì kết quả bài toán tăng nhẹ.

Confusion matrix trên cho thấy 74% các trường hợp nhầm lẫn background thành foreground thuộc về lớp no_mask. Điều này có thể lý giải do trong dữ liệu huấn luyện, tỉ lệ lớp no_mask là nhiều hơn. Tuy nhiên, nếu so sánh về tỉ lệ phân lớp đúng thì 2 lớp là ngang nhau. Đây có thể là do cơ chế giải quyết mất cân bằng dữ liệu được tích hợp sẵn trong mô hình. Cụ thể có thể phân tích thêm hàm Loss của YOLOv4 trong phần Phụ lục.

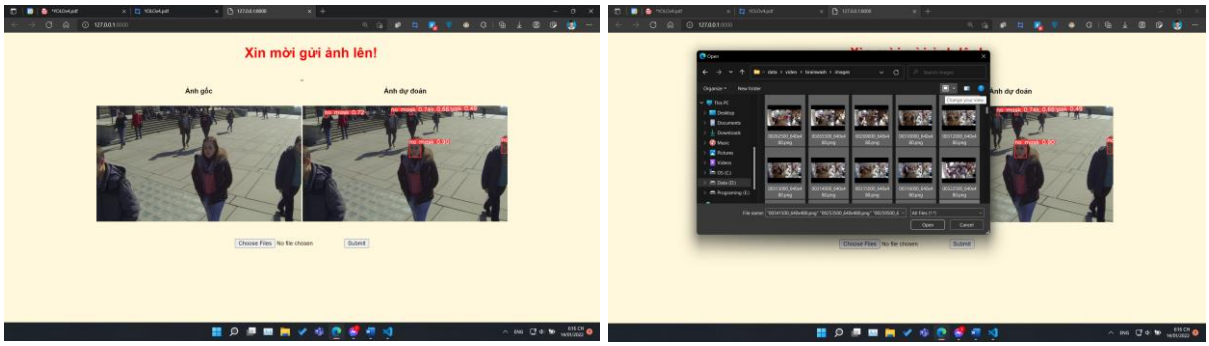


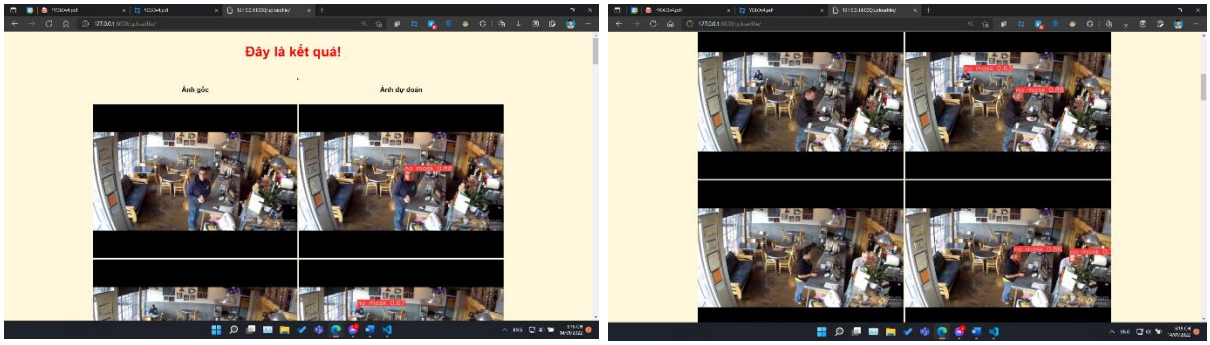
Hình 8. Một số hình ảnh nhận diện từ mô hình

Trong hầu hết trường hợp mô hình nhận diện tốt, thậm chí với các trường hợp đeo khẩu trang nhưng không che hết mũi (xem như là không đeo), các trường hợp bị khuất một phần và với điều kiện ánh sáng không tốt. Tuy vậy mô hình vẫn bị nhầm lẫn với các trường hợp đeo khẩu trang sai (được xem như là không đeo).

5. Demo trên web:

Từ mô hình đã được huấn luyện, chúng tôi xây dựng một ứng dụng demo đơn giản nhằm mục đích tạo một giao diện trực quan để dễ dàng quan sát và đánh giá mô hình trên các dữ liệu thực và chia sẻ cho mọi người có thể sử dụng tại link Github này.





Hình 9. Một số hình ảnh từ web demo

V. Kết luận

Từ kết quả đạt được qua thực nghiệm và quan sát qua ứng dụng demo, chúng tôi thấy rằng mô hình dự đoán gần như chính xác tuyệt đối trên một số góc nhìn nhất định như trong bộ dữ liệu WILDTRACK và bộ dữ liệu gốc.

Tuy nhiên các góc nhìn hơi khác ngữ cảnh như bộ Oxford Town Center thì hay có sự nhầm lẫn giữa các lớp. Khi bỏ bộ dữ liệu này đi thì kết quả có tăng lên.

Nhìn chung, dựa vào kết quả đạt được thì mô hình chưa đủ chính xác để có thể áp dụng cho bài toán thực tế. Chúng tôi đặt niềm tin rằng mô hình có thể đạt được độ chính xác cao hơn nếu như có nhiều dữ liệu chất lượng hơn.

Chúng tôi đánh giá mô hình YOLOv5 rất dễ huấn luyện và hiệu năng cũng rất ổn định. Công việc chúng tôi tập trung nhiều nhất cho đồ án này là thu thập và gắn nhãn cho dữ liệu. Công việc tương lai có thể làm để cải thiện hơn mô hình trên là thu thập thêm dữ liệu chất lượng cao và tùy chỉnh nhiều hơn file config của mô hình.

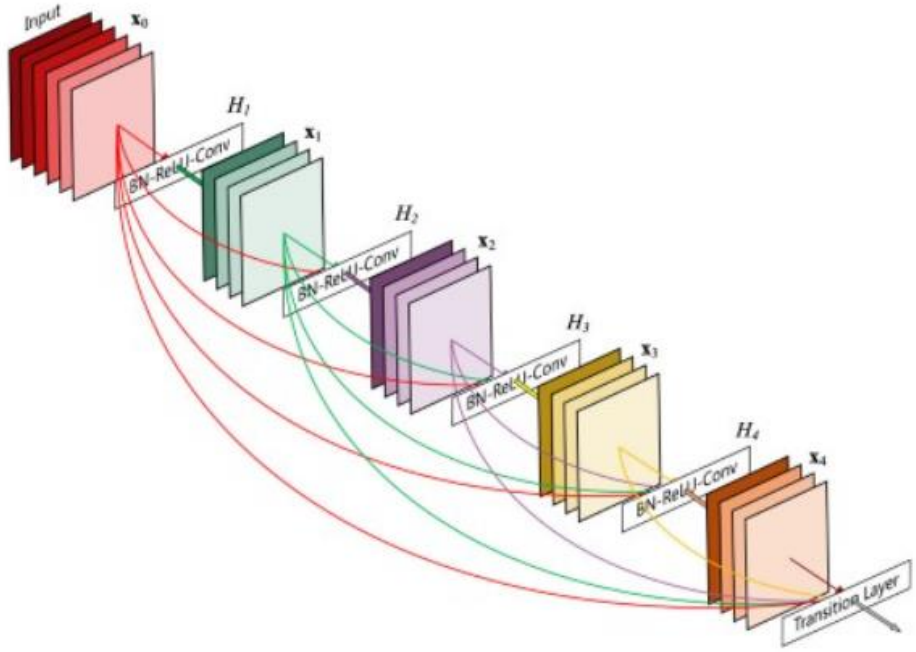
VI. Phụ lục: YOLOv4

1. Kiến trúc chi tiết

Vì YOLOv5 vẫn chưa có bài báo chính thức nào nên chúng tôi tập trung phân tích YOLOv4 thay vào đó.

Backbone

DenseBlock chứa nhiều lớp tích chập, với mỗi lớp H thay vì sử dụng output của lớp liền trước thì H sử dụng output của tất cả các lớp trước đó làm input.



DenseNet được hình thành bởi nhiều khối DenseBlock, tạo nên 1 mạng học sâu với mục đích tăng độ phức tạp của mô hình.

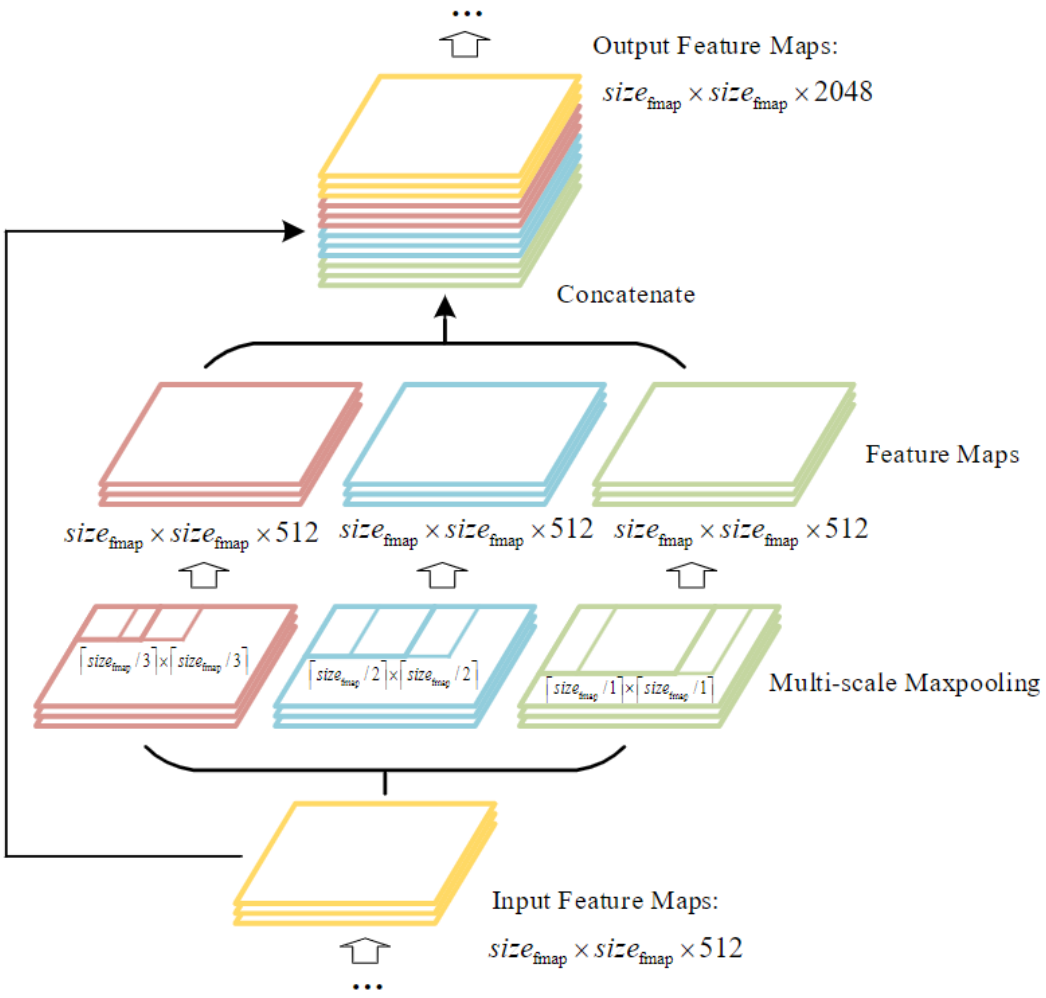
CSP tách feature maps đầu vào thành 2 phần. Phần đầu tiên sẽ làm input cho lớp DenseBlock tiếp theo, phần còn lại sẽ được giữ nguyên và gộp với output của DenseBlock. Mục đích CSPNet sẽ làm giảm độ phức tạp tính toán bằng cách tách input thành 2 phần và chỉ 1 phần đi qua DenseBlock.

CSPDarknet53 mà một mạng được cấu thành từ các khối DenseBlock và CSP. YOLOv4 sử dụng một số lớp trong CSPDarknet53 làm backbone để tăng tốc độ tính toán mà vẫn giữ được độ chính xác.

Neck

Để làm giàu dữ liệu trước khi đưa vào Object Detectors, các feature maps từ Backbone được đi qua phần Neck gồm Top-down stream, Bottom-up stream và các skip connection để tăng lượng thông tin và phát hiện trên nhiều scale khác nhau.

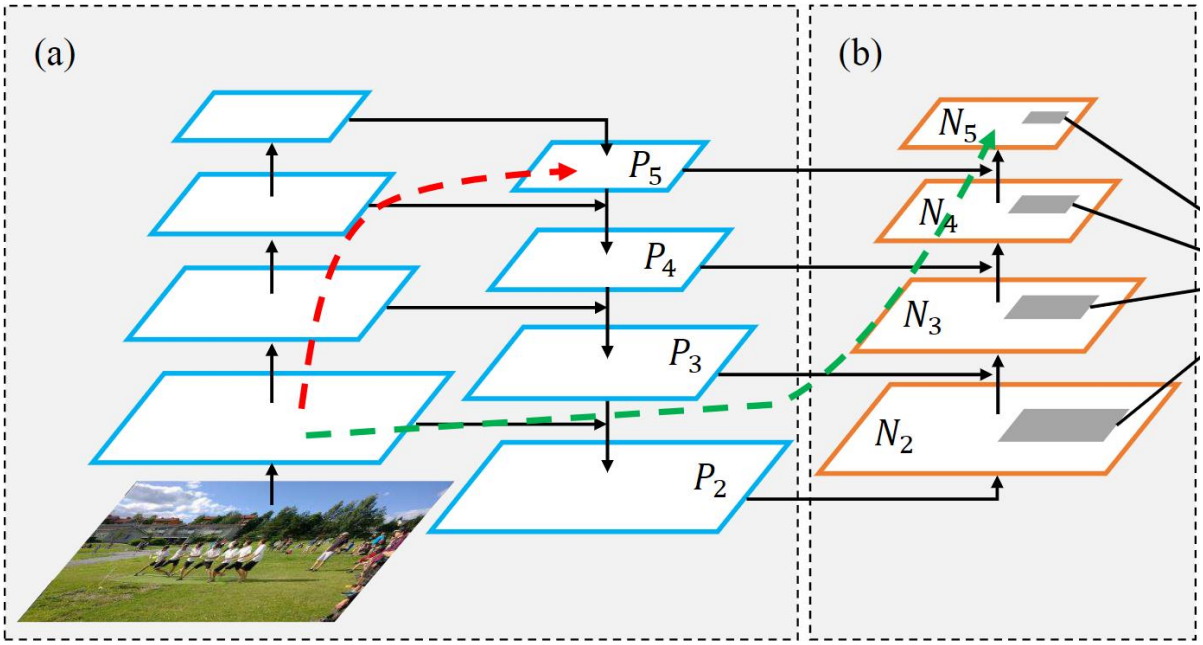
SPP (spatial pyramid pooling layer)



Trong YOLOv4 thì SPP nhận đầu vào là feature maps từ bước trước đó. Sau đó, áp dụng Max Pooling với các kernel size và stride khác nhau. Output được nối lại kèm với feature maps ban đầu qua Residual connection. Công dụng của SPP là giảm chiều dữ liệu như Max Pooling để giảm chi phí tính toán tuy nhiên trên nhiều scale khác nhau để có thể phát hiện được các đối tượng có kích thước khác nhau và vẫn giữ được các tính chất về không gian.

Path Aggregation Network (PAN)

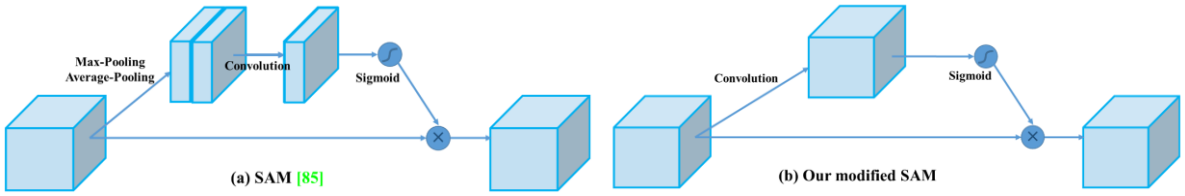
Đối với FPN (trong hình (a)), khi một feature map low-level muốn truyền đến một layer nằm phía trên đỉnh cần phải trải qua hàng trăm lớp trung gian. Trong khi đó PAN thêm vào một đường đi lên phía sau FPN (trong hình (b)) tạo một shortcut làm đường đi của các feature maps low-level tới các layer ở đỉnh ngắn hơn đáng kể (khoảng 10 layers).



PAN trong YOLOv4 đã được sửa đổi so với bản gốc. Tác giả thay vì cộng các features maps qua các Skip connection thì sẽ là nhân từng phần tử với nhau.

Spatial Attention Module (SAM)

SAM là một kiến trúc tách đầu vào thành một nhánh mới và áp dụng cả Max Pooling và Average Pooling rồi đưa qua vài lớp tích chập trước khi đưa qua một hàm Sigmoid để lấy đầu ra như một bộ trọng số cho các không gian trong feature maps đầu vào.



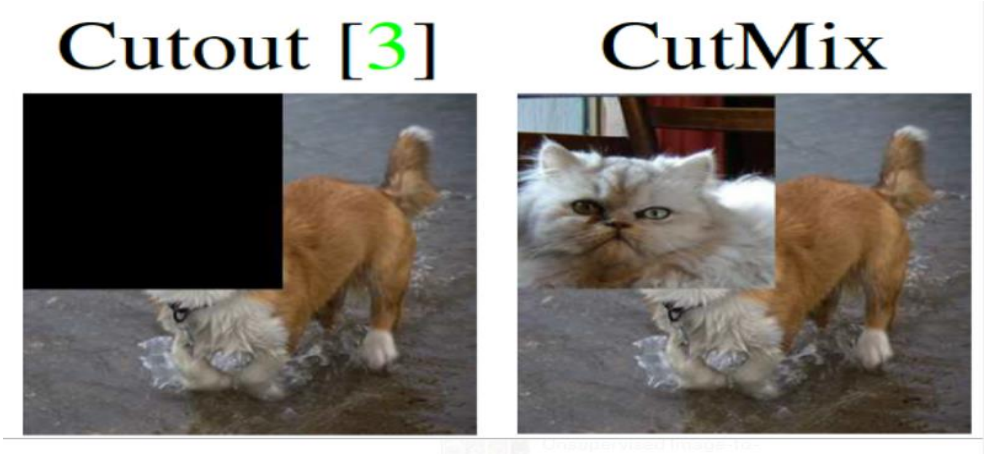
SAM trong YOLOv4 được thiết kế lại, thay không dùng Pooling mà chỉ dùng một mạng CNN trước khi đưa qua hàm Sigmoid.

CutMix data augmentation

Cutout data augmentation sẽ loại bỏ một vùng của hình ảnh, một phần của hình ảnh chứa những thông tin không quan trọng sẽ gây nên sự lãng phí. Trong CutMix, một phần hình ảnh sẽ được cắt và dán trên một hình ảnh khác.

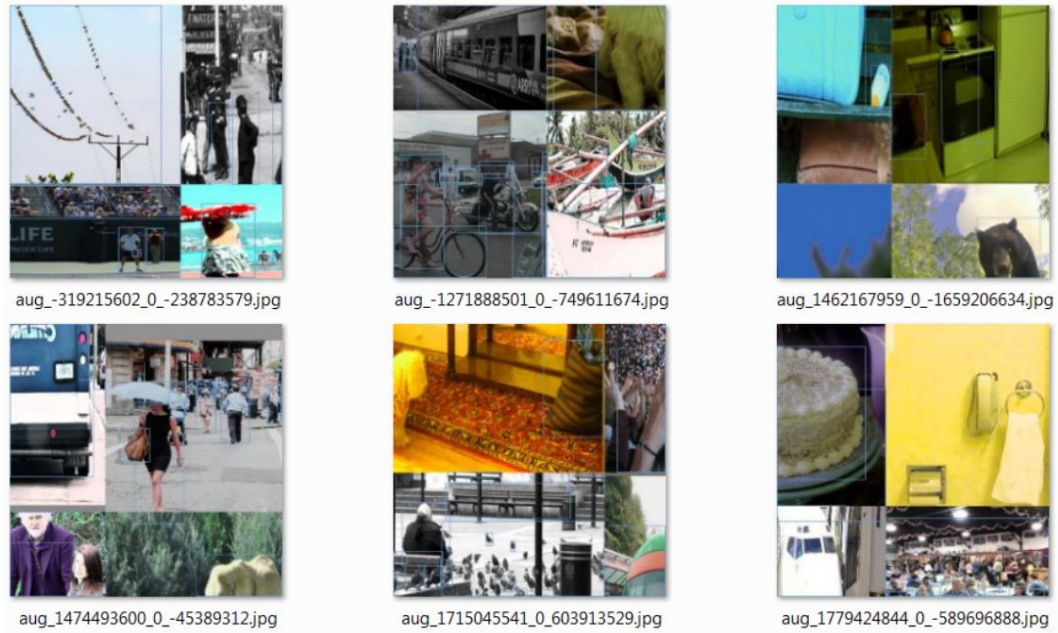
DropBlock regularization (CutOut)

Trong các mạng sử dụng lớp Fully-connected, người ta thường dùng dropoff để bỏ đi một số điểm ảnh, khiến cho mô hình học được từ đa dạng các đặc trưng. Tuy nhiên, đối với lớp Convolutional, mối tương quan giữa các pixel gần nhau là cần thiết. Vì vậy tác giả sử dụng DropBlock, bỏ đi các khối ngẫu nhiên trong hình thay vì chỉ bỏ đi từng pixel ngẫu nhiên.



Mosaic data augmentation

Mosaic kết hợp 4 hình trong bộ dữ liệu lại với nhau thành 1 mẫu dữ liệu. Nó giúp mô hình học được trong cả những ngữ cảnh khác biệt. Ngoài ra, khi áp dụng Mosaic thì số lượng ảnh trong 1 mini-batch cũng sẽ được giảm xuống, giúp tăng tốc độ xử lý.



Class label smoothing

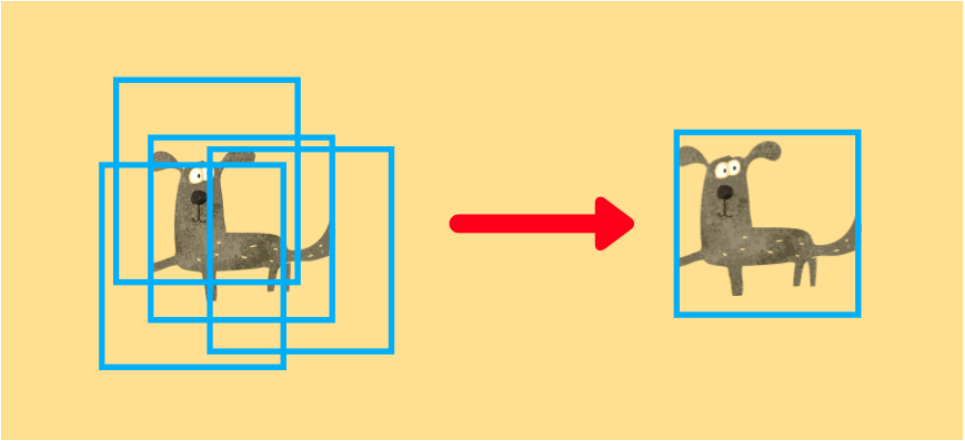
Nếu sự tin cậy bằng 100% khi dự đoán có thể thấy mô hình đang ghi nhớ dữ liệu thay vì học hỏi, thay vì gắn nhãn là 1 thì sẽ sử dụng 0.9, điều này cũng sẽ cải thiện hiện tượng overfitting.

Self-Adversarial Training (SAT)

Thông thường, khi thực hiện lan truyền ngược mô hình sẽ cập nhật các trọng số. Đối với SAT, nó sẽ thay đổi ảnh đầu vào sao cho mô hình sẽ dự đoán kém nhất. Đây là một cách tăng cường dữ liệu, trong lượt huấn luyện tiếp theo, mô hình sẽ được huấn luyện trên ảnh đã được biến đổi. Nó giúp mô hình được khái quát hóa và giảm overfitting.

DIoU-NMS

NMS lọc ra những B-Box khác dự đoán và giữ lại những B-Box với độ tin cậy cao nhất



DIoU sử dụng như 1 yếu tố trong việc non-maximum suppresion (NMS). Phương pháp lấy IoU và khoảng cách giữa các điểm trung tâm của 2 B-Box khi loại bỏ các B-Box dư thừa. Sẽ giúp cho B-Box chính xác hơn đối với object.

Anchor boxes

Kể từ YOLOv2, Anchor Box được sử dụng trong việc hồi quy bounding box. Thay vì trực tiếp hồi quy về các giá trị x,y,w,h của bounding box một cách tự do, mô hình sẽ hồi quy về các giá trị offset cho những "khung bounding box" được định nghĩa sẵn gọi là Anchor box.

Trong YOLO, Anchor Box được xác định dựa trên Kmean, bằng cách xem xét các bounding boxes trong tập dữ liệu huấn luyện.

Việc sử dụng Anchor boxes giúp cho mô hình đạt được IoU cao hơn.

Loss

Hàm loss của YOLOv4 có phần tương tự như các thế hệ YOLO trước đó. Tuy nhiên có một số thay đổi, cụ thể là:

Loss của Object và Classification dùng Binary Cross Entropy Loss thay vì Mean Square Error.

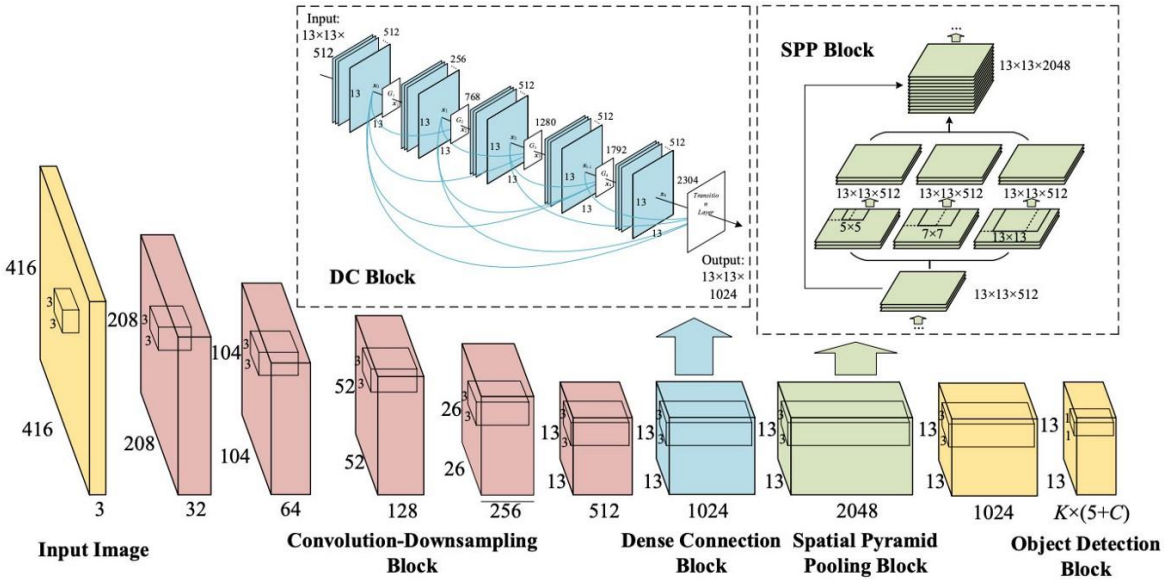
Sử dụng Complete IoU Loss (CIoU) cho việc hồi quy bounding box thay vì Mean Square Error. Điều này giúp tăng độ trùng lắp giữa bounding box dự đoán và bounding box nhãn, giảm khoảng cách giữa tâm của chúng, duy trì được tỉ lệ cạnh của bounding box.

Nhìn chung, hàm Loss của YOLO nói chung và YOLOv4 nói riêng sẽ bao gồm các thành phần chính như bên dưới: Loss bounding box (CIoU), Loss object (dòng 2 và 3) và Loss classification (dòng 4).

$$
\begin{aligned}
\mathcal{L}(\hat{z}, z) = & \mathcal{L}_{CIoU} \\
& - \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\hat{C}_i \log\left(C_i\right) + \left(1 - \hat{C}_i\right) \log\left(1 - C_i\right) \right] \\
& - \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{\text{noobj}} \left[\hat{C}_i \log\left(C_i\right) + \left(1 - \hat{C}_i\right) \log\left(1 - C_i\right) \right] \\
& - \sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c \in \text{ classes}} \left[\hat{p}_i(c) \log\left(p_i(c)\right) + \left(1 - \hat{p}_i(c)\right) \log\left(1 - p_i(c)\right) \right]
\end{aligned}
$$

2. Kiến trúc tổng quan

Về tổng quan, mô hình YOLOv4 sẽ lần lượt bao gồm các lớp Convolutional và DenseBlock kết hợp với kiến trúc CSP ở phần Backbone. Sau đó Feature maps được đưa qua phần Neck bao gồm các khối SPP, PAN và SAM để tăng cường thông tin. Cuối cùng sẽ đi qua một mạng Dense Detection ở phần Head để dự đoán đầu ra.



VII. Tài liệu tham khảo

1. REDMON, Joseph, et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. p. 779-788.

2. REDMON, Joseph; FARHADI, Ali. YOLO9000: better, faster, stronger. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017. p. 7263-7271.

3. REDMON, Joseph; FARHADI, Ali. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

4. BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

5. YOLOv4. While object detection matures in the… | by Jonathan Hui | Medium

6. YOLO v4 or YOLO v5 or PP-YOLO? Which should I use? | Towards Data Science

7. Object Detection in 2022: The Definitive Guide - viso.ai

8. PASCAL VOC 2007 Benchmark (Object Detection) | Papers With Code