

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

NGUYỄN XUÂN VINH

KHÓA LUẬN TỐT NGHIỆP
XÂY DỰNG ỨNG DỤNG NHẬN DIỆN VÀ GIẢI MÃ QR
ĐƯỢC NHÚNG TRONG VIDEO TRỰC TUYẾN

BUILDING APPLICATIONS TO DETECT AND DECODE QR
CODE EMBEDDED IN STREAMING VIDEOS

CỦ NHÂN NGÀNH KHOA HỌC DỮ LIỆU

TP. HỒ CHÍ MINH, 2022

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

NGUYỄN XUÂN VINH – 18521655

KHÓA LUẬN TỐT NGHIỆP
XÂY DỰNG ỨNG DỤNG NHẬN DIỆN VÀ GIẢI MÃ QR
ĐƯỢC NHÚNG TRONG VIDEO TRỰC TUYẾN
BUILDING APPLICATIONS TO DETECT AND DECODE QR
CODE EMBEDDED IN STREAMING VIDEOS

CỦ NHÂN NGÀNH KHOA HỌC DỮ LIỆU

GIẢNG VIÊN HƯỚNG DẪN
ThS. Phạm Thế Sơn

TP. HỒ CHÍ MINH, 2022

THÔNG TIN HỘI ĐỒNG CHẤM KHÓA LUẬN TỐT NGHIỆP

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số
ngày của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành nhất đến Ths. Phạm Thế Sơn, người đã đồng hành, quan tâm và giúp đỡ em trong thời gian qua, động viên em hoàn thành tốt luận văn này. Và thầy cũng là người truyền cảm hứng cho em, cho em đủ tự tin, dũng khí và để đạt được kết quả tốt.

Em cũng xin gửi lời cảm ơn đến tất cả các thầy cô giáo tại trường Đại học Công nghệ Thông tin, đặc biệt là Khoa Khoa học và Kỹ thuật Thông tin đã hướng dẫn và tận tình chỉ bảo cho chúng em. Đã truyền đạt cho chúng em nhiều kiến thức bổ ích để có được những kỹ năng cần thiết để đạt được thành công nào đó trong tương lai.

Cuối cùng, em xin gửi lời cảm ơn sâu sắc đến gia đình, bạn bè đã luôn bên cạnh em trong suốt thời gian qua, họ đã động viên, khích lệ, giúp đỡ, chỉ bảo những lúc khó khăn, tiếp thêm động lực để em hoàn thành tốt luận văn này.

Em xin chân thành cảm ơn!

Tác giả

Nguyễn Xuân Vinh

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG NHẬN DIỆN VÀ GIẢI MÃ QR ĐƯỢC NHÚNG TRONG VIDEO TRỰC TUYẾN.

TÊN ĐỀ TÀI (tiếng Anh): BUILDING APPLICATIONS TO DETECT AND DECODE QR CODE EMBEDDED IN STREAMING VIDEOS.

Cán bộ hướng dẫn: ThS. Phạm Thế Sơn

Thời gian thực hiện: Từ ngày 22/02/2022 đến ngày 10/07/2022

Sinh viên thực hiện:

Nguyễn Xuân Vinh - 18521655

Nội dung đề tài:

Mục tiêu nghiên cứu:

- Xây dựng bộ dữ liệu cho bài toán nhận diện mã QR
- Nghiên cứu mô hình, kỹ thuật liên quan đến bài toán nhận diện và giải mã QR được nhúng trong các video trực tuyến tăng cường và giải quyết bài toán được đặt ra.
- Chạy thử nghiệm mô hình và các kỹ thuật để kiểm chứng độ chính xác và đánh giá hiệu suất của mô hình đã thử nghiệm
- Cải thiện độ chính xác của mô hình
- Áp dụng mô hình vào thực tế, tạo một ứng dụng nhận diện và giải mã QR được nhúng trong video trực tuyến tăng cường.

Phạm vi nghiên cứu:

- Miền dữ liệu: Hình ảnh các mã QR được nhúng trong các video trực tuyến, mã QR được gán trên các biển báo, biển quảng cáo được chụp ngoài đời thực.
- Hình ảnh mã QR ở kích thước vừa và nhỏ, thường ở góc hình ảnh.

Đối tượng nghiêm cấm:

- Bài toán nhận diện và giải mã QR.
- Hình ảnh QR được nhúng trong các video trực tuyến.
- Hình ảnh QR được quay trực tiếp ngoài đời thực

Phương pháp thực hiện:

- **Xây dựng bộ dữ liệu.**
 - Dữ liệu được thu thập bằng cách sử dụng các video quay video trực tuyến có mã QR tăng cường, sau đó tách ra các khung hình có mã QR. Tiếp theo là chụp các ảnh mã QR xuất hiện trên các bảng quảng cáo trên đường, khu dân cư,...
 - Gán nhãn mã QR.
 - Kiểm tra lại việc gán nhãn dữ liệu và chỉnh sửa nếu phát sinh lỗi.
- **Bài toán:** Nhận diện và giải mã QR được nhúng trong video trực tuyến tăng cường.
 - Đầu vào: Hình ảnh có QR code (**INPUT**).
 - Đầu ra: Nhận diện QR code (**OUTPUT_1**) và giải mã QR code (**OUTPUT_2**).

Ví dụ minh họa:

INPUT



OUTPUT_1



Show the result scan QR code

<https://l.lead.me/springfield-12>

OUTPUT_2

Hình 1. Ví dụ minh họa về input và output của bài toán

- **Nghiên cứu phương pháp thực nghiệm:** Đối với bài toán nhận diện QR, có rất nhiều phương pháp để Object detection như các mô hình họ **YOLO** (You only look once) và các mô hình họ **R-CNN**. Tuy nhiên đối với bài toán này chúng tôi hướng đến việc nhận diện đối tượng QR theo giờ thực vì thế mô hình họ **YOLO** sẽ là một lựa chọn tối ưu đối với bài toán này. Các mô hình họ **YOLO** gồm:
 - **YOLOv1** (08/06/2015) được phát hành bởi tác giả Joseph Redmon. Tên bài báo là [You Only Look Once: Unified, Real-Time Object Detection](#)
 - **YOLOv2** (25/12/2016) được phát hành bởi tác giả Joseph Redmon và Ali Farhadi. Tên bài báo là [YOLO9000: Better, Faster, Stronger](#)
 - **YOLOv3** (08/04/2018) được cải thiện từ **YOLOv2** và tác giả Joseph Redmon và Ali Farhadi cùng với các tác giả ban đầu đã đóng góp. Tên bài báo là [YOLOv3: An Incremental Improvement](#)

- **YOLOv4** (23/04/2020) Khi tác giả gốc đang gặp bế tắc trong việc phát triển thì nhóm tác giả gồm Alexey Bochoknovskiy, Chien-Yao Wang và Hong-Yuan Mark Liao đã phát hành **YOLOv4**. Tên bài báo: [YOLOv4: Optimal Speed and Accuracy of Object Detection](#)
- **YOLOv5** (18/05/2020) Ngay sau khi phát hành YOLOv4, Glenn Jocher đã giới thiệu YOLOv5 bằng cách sử dụng khung Pytorch. Mã nguồn mở có sẵn trên [Github](#)

Chúng tôi sẽ chạy thực nghiệm trên các mô hình YOLOv3, YOLOv4 và YOLOv5 để tìm ra mô hình có kết quả tối tốt nhất cho việc nhận diện mã QR. Chúng tôi sử dụng các mô hình YOLOv3, YOLOv4 và YOLOv5 vì: Đầu tiên nói về cách thức hoạt động của mô hình YOLO so với mô hình R-CNN: mô hình YOLO là one stage detector còn mô hình R-CNN là two stage detector, tức mô hình R-CNN sẽ trải qua 2 giai đoạn: Detection và Pose Estimation, còn mô hình YOLO thì sẽ thực hiện cả 2 công việc tìm bounding box và classification vật thể. Vì vậy tốc độ nhận dạng của YOLO sẽ nhanh hơn nhiều so với R-CNN và đáp ứng được yêu cầu nhận dạng real-time của bài toán. Tiếp đến lý do chọn YOLOv3 vì đây là phiên bản cuối cùng của tác giả gốc. YOLOv4 là phiên bản được nhiều người tin dùng và có bài báo về mô hình. YOLOv5 phiên bản mới nhất đạt được kết quả cao tuy nhiên chưa có bài báo chính thức. Việc huấn luyện trên cả ba mô hình sẽ tìm được mô hình phù hợp với bộ dữ liệu. Và cuối cùng vì mô hình chỉ nhận dạng một class duy nhất là QR code nên mô hình YOLO sẽ cho kết quả rất cao và chính xác.

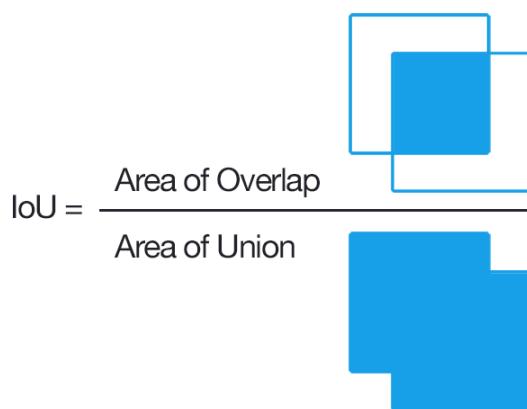
Tiếp đến việc giải mã QR chúng tôi sẽ sử dụng các thư viện đã được phát triển từ trước như: **pymzbar**, và **opencv-python**. Chúng tôi sẽ thực nghiệm với cả 2 thư viện giải mã QR và so sánh để tìm ra thư viện có thể hoạt động tốt nhất.

- Phương pháp đánh giá:

- 1. IoU:** Intersection over Union là chỉ số đánh giá được sử dụng để đo độ chính xác của detection task trên tập dữ liệu cụ thể. IOU thường được đánh giá hiệu năng của các mô hình detection.

Để tính IoU ta cần có hai bounding box: Thứ nhất là bounding box thực sự của đối tượng (chính là bounding box đã gán nhãn). Thứ hai là bounding box của đối tượng được dự đoán bởi mô hình.

Cách tính IoU sẽ là tỉ lệ diện tích giao nhau giữa bounding box thực sự và bounding box dự đoán với diện tích hợp của hai bounding box đó. Được minh họa ở hình dưới đây.



Hình 2. Minh họa cách tính độ đo IoU

2. True/false positive/negative:

Kết quả trả về của IoU sẽ nằm trong khoảng từ 0 đến 1, và mỗi detection sẽ có một IoU riêng. Để xác định được wrong detection hay correct detection thì chúng ta phải dựa vào một ngưỡng cho trước tùy vào bài toán (0.25, 0.5, 0.75,...). Nếu IoU của detection lớn hơn hoặc bằng so với ngưỡng cho trước thì đó là correct detection, ngược lại thì wrong detection.

Từ đó ta có định nghĩa về True/false positive/negative:

- True Positive (TP): IoU lớn hơn hoặc bằng ngưỡng, là một correct detection.
Trong trường hợp có nhiều box có IoU lớn hơn ngưỡng thì box có IoU lớn nhất sẽ được tính là TP, còn lại là FP.
- False Positive (FP): IoU bé hơn ngưỡng (là một wrong detection).
- False Negative (FN): trường hợp mà có bounding box nhưng không dự đoán.

3. Precision là thang đo độ chính xác của dự đoán.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

4. Recall là thang đo độ tốt của khả năng tìm thấy được các correct detection.

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

5. AP

Từ precision và recall được định nghĩa ở trên chúng ta có thể đánh giá mô hình bằng cách thay đổi ngưỡng và giá trị quan sát của precision và recall. Khái niệm Area Under the Curve (AUC) cũng được định nghĩa tương tự. Với Precision-Recall Curve, AUC còn có một tên khác là Average precision (AP). Giá trị AP là giá trị phía dưới đường biểu diễn mối quan hệ precision – recall. Tại mỗi recall level, ta thay giá trị precision bằng giá trị precision lớn nhất tại recall level đó. AP sẽ bằng phần diện tích phía dưới đường biểu diễn precision-recall bằng cách tính tổng các hình chữ nhật xấp xỉ.

Công thức:

$$P_{interp}(r_{n+1}) = \max_{r' >= r+1} p(r') \quad (3)$$

$$AP = \sum (r_{n+1} - r_n) P_{interp}(r_{n+1}) \quad (4)$$

6. mAP

Bài toán Object Detection của chúng ta có nhiều class, mỗi class ta sẽ tiến hành đo AP, sau đó lấy trung bình của tất cả các giá trị AP của các class ta được chỉ số mAP của mô hình.

7. Accuracy decode:

$$Accuracy = \frac{\text{Số QR decode đúng}}{\text{Tổng số QR}} \quad (5)$$

Trong đó:

- + Số QR decode đúng là số lượng QR code của bộ dữ liệu test sau khi qua hệ thống detect và decode đạt được kết quả đúng
- + Tổng số QR là số lượng của tất cả các mã QR mà decode bằng cách thủ công được trong bộ test.

Ví dụ: Có 100 ảnh trong bộ test, và có tổng 150 QR trong 100 ảnh đó. Và trong 150 mã QR này chúng ta có thể decode thủ công được 120 QR (30 QR còn lại không thể decode được bởi các trường hợp bất khả kháng như có thể bị nhòe, bị cắt 1 phần, bị quá chói hoặc quá tối hoàn toàn không thể decode được.). Thì Tổng số QR = 120. Sau khi 100 ảnh test qua hệ thống và đưa ra kết quả có được 90 QR được giải mã thì Accuracy sẽ bằng $90/120 = 0.75$ tức 75%.

Kết quả mong đợi:

- Bộ dữ liệu QR đạt kích thước 10000 ảnh với các ảnh được chụp và cắt ra từ các video trực tuyến. Bộ dữ liệu được đánh giá là tốt là bộ dữ liệu có kích thước lớn, với số lượng ảnh đa dạng, ảnh không bị nhòe, vỡ, chất lượng ảnh cao. - Kết quả đánh giá phương pháp YOLO trên bộ dữ liệu QR chúng tôi kỳ vọng trên 90% (mAP) với ngưỡng IoU >0.5 và trên 80% với ngưỡng IoU > 0.75 đối với tác vụ nhận diện đối tượng. Và kết quả accuracy decode đánh giá phương pháp giải mã từ các thư viện đạt trên 80% .
- Thách thức của bài toán: Video (ảnh) đầu vào được quay (chụp) có khoảng cách xa với mã QR. Các điều kiện về ánh sáng tác động khi quay (chụp) mã QR. Kích thước và độ tập trung của QR trên video (ảnh).

- Chúng tôi giải quyết được các mã QR với khoảng cách 10m với tỉ lệ của mã QR trên toàn bức ảnh đạt 10%.
- Tạo ra một ứng dụng áp dụng nhận diện và giải mã QR với việc áp dụng các mô hình tốt nhất và tốc độ xử lý của ứng dụng bằng với tốc độ thời gian thực.
- Thách thức đặt ra cho app mobile là thời gian xử lý sẽ phụ thuộc rất là lớn vào GPU của thiết bị. Chúng tôi sẽ cố gắng tạo một hệ thống nhỏ nhẹ mà kết quả vẫn tốt cho việc giải mã QR code cho nhiều thiết bị.

Tài liệu tham khảo:

1. Redmon, Joseph, and Ali Farhadi. "YOLOv3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).
2. Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).
3. Muniz, Ruben, Luis Junco, and Adolfo Otero. "A robust software barcode reader using the Hough transform." Proceedings 1999 International Conference on Information Intelligence and Systems (Cat. No. PR00446). IEEE, 1999.
4. Ohbuchi, Eisaku, Hiroshi Hanaizumi, and Lim Ah Hock. "Barcode readers using the camera device in mobile phones." 2004 International Conference on Cyberworlds. IEEE, 2004.
5. Wachenfeld, Steffen, Sebastian Terlunen, and Xiaoyi Jiang. "Robust recognition of 1-d barcodes using camera phones." 2008 19th International Conference on Pattern Recognition. IEEE, 2008.

Kế hoạch thực hiện:

- Chúng tôi thực hiện đề tài xây dựng ứng dụng nhận diện và giải mã QR được nhúng trong video trực tuyến trong vòng 5 tháng và kế hoạch thực hiện chi tiết trong bảng 1.

Bảng 1. Tóm tắt kế hoạch thực hiện đề tài trong 5 tháng

Công việc	Sinh viên	Tháng 2	Tháng 3	Tháng 4	Tháng 5	Tháng 6
Tìm hiểu tổng quan đề tài	Vinh					
Xây dựng bộ dữ liệu	Vinh					
Nghiên cứu phương pháp thực nghiệm	Vinh					
Xây dựng ứng dụng	Vinh					
Báo cáo	Vinh					

Xác nhận của CBHD

(Ký tên và ghi rõ họ tên)

TP. HCM, ngày 28 tháng 01 năm 2022

Sinh viên

(Ký tên và ghi rõ họ tên)

Nguyễn Xuân Vinh

MỤC LỤC

ĐỀ CƯƠNG CHI TIẾT	5
Chương 1. TỔNG QUAN	7
1.1. Giới thiệu đề tài	7
1.2. Tính ứng dụng của đề tài	8
1.3. Kết luận.....	9
Chương 2. CÁC CÔNG TRÌNH NGHIÊN CỨU LIÊN QUAN	10
2.1. Các công trình nghiên cứu liên quan	10
2.2. Kết luận.....	10
Chương 3. BỘ DỮ LIỆU	12
3.1. Xây dựng bộ dữ liệu	12
3.1.1. Thu thập dữ liệu.....	12
3.1.2. Gán nhãn dữ liệu	13
3.2. Chi tiết bộ dữ liệu	14
3.3. Kết luận.....	15
Chương 4. PHƯƠNG PHÁP TIẾP CẬN.....	16
4.1. Detect QR code.....	16
4.1.1. YOLOv1	16
4.1.2. YOLOv2	18
4.1.3. Tổng quan YOLOv3, YOLOv4 và YOLOv5.....	21
4.1.4. YOLOv3	22
4.1.5. YOLOv4.....	26

4.1.6.	YOLOv5	33
4.2.	DECODE QR CODE.....	35
4.2.1.	Pyzbar (Zbar).....	35
4.2.2.	QRCodeDetector (Opencv-python).....	36
4.3.	Kết luận.....	37
Chương 5.	CÀI ĐẶT THỬ NGHIỆM VÀ ĐÁNH GIÁ	38
5.1.	Cài đặt thử nghiệm	38
5.1.1.	YOLOv3	38
5.1.2.	YOLOv4	39
5.1.3.	YOLOv5	39
5.2.	Phương pháp đánh giá	40
5.2.1.	IoU:.....	40
5.2.2.	True/false positive/negative:	41
5.2.3.	Precision	41
5.2.4.	Recall	42
5.2.5.	AP	42
5.2.6.	mAP	42
5.2.7.	Accuracy decode:	43
5.3.	Kết quả thực nghiệm và đánh giá	43
5.3.1.	Kết quả thực nghiệm detection QR code.....	43
5.3.2.	Kết quả thực nghiệm decode QR code	44
5.4.	Phân tích lỗi	45
5.5.	Kết luận.....	46
Chương 6.	XÂY DỰNG ỨNG DỤNG	47

6.1.	Giới thiệu	47
6.2.	Sơ đồ Use Case	48
6.2.1.	Danh sách Actor	49
6.2.2.	Danh sách UseCase	49
6.2.3.	Đặc tả UseCase.....	49
6.2.3.1.	Đặc tả UseCase “Chọn ảnh từ thiết bị”	49
6.2.3.2.	Đặc tả UseCase “Chọn video từ thiết bị”.....	50
6.2.3.3.	Đặc tả UseCase “Sử dụng camera của thiết bị”	51
6.3.	Sơ đồ tuần tự.....	52
6.3.1.	Chọn hình ảnh từ thiết bị.....	52
6.3.2.	Chọn video từ thiết bị	53
6.3.3.	Sử dụng camera của thiết bị	54
6.4.	Mô hình hóa ứng xử	55
6.4.1.	Sơ đồ trạng thái	55
6.4.1.1.	Chọn ảnh từ thiết bị.....	55
6.4.1.2.	Chọn video từ thiết bị.....	56
6.4.1.3.	Sử dụng camera của thiết bị	57
6.4.2.	Luồng xử lý dữ liệu	58
6.5.	Qui trình xây dựng.....	59
6.5.1.	Server detection	59
6.5.2.	Server decode	61
6.5.3.	Server xử lý	61
6.5.4.	gRPC cho server detection	62
6.5.5.	Giao diện	63

6.5.5.1. Sơ đồ luồng màn hình	63
6.5.5.2. Mô tả chi tiết màn hình	63
6.6. Kết quả.....	68
6.7. Kết luận.....	68
Chương 7. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	69
7.1. Kết luận.....	69
7.2. Hạn chế	69
7.3. Hướng phát triển	69

DANH MỤC HÌNH

Hình 3.1: Quy trình xây dựng tổng quan bộ dữ liệu	12
Hình 3.2: Mô phỏng gán nhãn YOLO bằng tool LabelImg và output của LabelImg	13
Hình 3.3: Sơ đồ cây tập dữ liệu cho mô hình YOLOv3 và YOLOv4	14
Hình 3.4: Sơ đồ cây tập dữ liệu cho mô hình YOLOv5	15
Hình 4.1: Cấu trúc YOLOv1	17
Hình 4.2: Minh họa non-max suppression	18
Hình 4.3: Phát hiện nhiều đối tượng trong một ô lưới	19
Hình 4.4: Minh họa 1% số lượng Anchor box trong mạng RetinaNet	20
Hình 4.5: Cấu trúc Darknet19	21
Hình 4.6: Cấu trúc Darknet-53	23
Hình 4.7: Quá trình tìm kiếm vật thể	24
Hình 4.8: Kích thước của tensor	25
Hình 4.9: Quá trình đào tạo để tinh chỉnh kích thước của anchor box sao cho giống với vật thể nhất	26
Hình 4.10: Mô tả cấu trúc CSP	28
Hình 4.11: Cấu trúc DenseNet	28
Hình 4.12: Một số cấu trúc sử dụng cho phần cổ(Neck).	29
Hình 4.13: Drop block.....	30
Hình 4.14: Bag of Freebies.	31
Hình 4.15: Minh họa NMS.....	32
Hình 4.16: PANet.....	33
Hình 4.17: Kiến trúc YOLOv5.....	35
Hình 5.1: Minh họa cách tính độ đo IoU	41
Hình 5.2: Các trường hợp nhận dạng 1, 2, 3 theo thứ tự từ trái sang phải.....	45
Hình 5.3: Minh họa về hình ảnh đầu vào có chất lượng thấp	46
Hình 6.1: Qui trình nhận dạng QR code của hệ thống	47
Hình 6.2: Qui trình giải mã QR code của hệ thống.....	48

Hình 6.3: Sơ đồ Use Case của ứng dụng.....	48
Hình 6.4: Sơ đồ tuần tự chức năng “Chọn hình ảnh từ thiết bị”	52
Hình 6.5: Sơ đồ tuần tự chức năng “Chọn video từ thiết bị”	53
Hình 6.6: Sơ đồ tuần tự chức năng “Sử dụng camera của thiết bị”	54
Hình 6.7: Sơ đồ trạng thái chức năng “Chọn ảnh từ thiết bị”	55
Hình 6.8: Sơ đồ trạng thái chức năng “Chọn video từ thiết bị”	56
Hình 6.9: Sơ đồ trạng thái chức năng “Sử dụng camera của thiết bị”	57
Hình 6.10: Mô hình luồng dữ liệu của hệ thống trên website.....	58
Hình 6.11: Sơ đồ cây các thư mục và file xây dựng tensorflow serving	60
Hình 6.12: Sơ đồ luồng màn hình của ứng dụng Scan QR code.	63
Hình 6.13: Giao diện khi truy cập đến trang web	64
Hình 6.14: Giao diện với option Scan QR code with Image	65
Hình 6.15: Giao diện với option Scan QR code with Video.....	66
Hình 6.16: Giao diện với option Scan QR code with Camera.....	67

DANH MỤC BẢNG

Bảng 4.1: So sánh giữa các cấu trúc của YOLOv3, YOLOv4 và YOLOv5.....	22
Bảng 5.1: Kết quả thực nghiệm các mô hình detection	44
Bảng 5.2: Kết quả thực nghiệm framework decode pyzbar và opencv-python.....	44
Bảng 6.1: Bảng danh sách Actor.....	49
Bảng 6.2: Bảng danh sách UseCase.....	49
Bảng 6.3: Danh sách màn hình	63
Bảng 6.4: Thông số giao diện website với lựa chọn “See an illustration”	64
Bảng 6.5: Thông số giao diện website với lựa chọn “Scan QR code with Image” ..	65
Bảng 6.6: Thông số giao diện website với lựa chọn “Scan QR code with Video”...	66
Bảng 6.7: Thông số giao diện website với lựa chọn “Scan QR code with Video”...	67

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Ý nghĩa
1	BN	BatchNormalization
2	IoU	Intersection over Union
3	NMS	Non-max suppression
4	QR code	Quick Response code
5	R-CNN	Region-based Convolutional Neural Networks
6	YOLOv1	You Only Look Once, Version 1
7	YOLOv2	You Only Look Once, Version 2
8	YOLOv3	You Only Look Once, Version 3
9	YOLOv4	You Only Look Once, Version 4
10	YOLOv5	You Only Look Once, Version 5

TÓM TẮT KHÓA LUẬN

Với việc QR code đang được phổ biến như hiện tại việc giải mã nhanh chóng là một điều cần thiết để người dùng có thể tiếp nhận được thông tin một cách tức thời và hiệu quả. Ở bài khóa luận này chúng tôi đã xây dựng một hệ thống nhận diện và giải mã QR code một cách real-time áp dụng deep learning để tăng tốc độ nhận diện và có thể nhận diện được nhiều QR code cùng một lúc, mang đến tiện lợi cho người dùng. Trong khóa luận này chúng tôi đã xây dựng bộ dữ liệu ảnh QR code với 8734 ảnh được gán nhãn, bộ dữ liệu này sẽ được cung cấp miễn phí cho tất cả mọi người với mục đích nghiên cứu. Hệ thống gồm hai phần: nhận diện vùng có mã QR sử dụng YOLOv3, YOLOv4, YOLOv4-tiny, YOLOv5 và giải mã QR code bằng thư viện pyzbar (Zbar), QRCodeDetector (opencv-python). Kết quả tốt nhất cho phần nhận diện QR code là $mAP = 99.56\%$ của mô hình YOLOv3 với ngưỡng IoU = 0.5, và $mAP = 93.53\%$ của mô hình YOLOv4 với ngưỡng IoU = 0.75. Tiếp đến là kết quả tốt nhất của phần decode với kết quả accuracy = 77.36% của framework pyzbar.

MỞ ĐẦU

Đặt vấn đề

Trong xã hội hiện đại ngày nay, mã QR có mặt khắp nơi, từ các sản phẩm tiêu dùng, ứng dụng thanh toán, hội họp, hội nghị... vì nó khắc phục được những khuyết điểm của tem truyền thống. Mã QR lưu trữ nhiều dữ liệu khác nhau trong một không gian nhỏ, giảm thời gian thực hiện các thao tác trong khi vẫn mang lại hiệu quả mong muốn. Dưới đây là những ưu điểm của QR code:

- QR code có thể chứa nhiều thông tin khác nhau như: URL, thông tin liên hệ, thời gian diễn ra sự kiện, địa chỉ email, nội dung văn bản, tin nhắn ...
- So với mã vạch truyền thống, sử dụng QR code cho phép đọc thiết bị nhanh hơn và dễ dàng hơn, tiết kiệm thời gian và không gian.
- Tính bảo mật cao, an toàn cho người sử dụng.
- QR code có thể lưu trữ một lượng lớn thông tin lên đến hàng nghìn ký tự chữ và số, đồng thời chúng có thể chứa một lượng lớn thông tin, việc dễ sử dụng sẽ mang lại lợi ích rất nhiều cho các doanh nghiệp nhỏ..
- Khi sử dụng QR code, bạn có thể nhận được thông tin sản phẩm và sản phẩm đầy đủ hơn bằng cách tải xuống ứng dụng đọc QR code trên điện thoại thông minh của mình

Với những ưu điểm nổi bật của mình, QR code hiện nay đã có mặt ở khắp nơi và được xem như một cách nhận diện sản phẩm, dịch vụ trong cuộc sống. QR code hiện đang được sử dụng rộng rãi trong các lĩnh vực sau:

- Kiểm kê hàng hóa, thông tin sản phẩm.
- Thông tin cá nhân: trên card visit.
- Lưu trữ URL: điện thoại chỉ việc đọc QR Code để lấy URL, sau đó tự động mở trình duyệt.
- Sử dụng trong các bến xe buýt, xe lửa, tàu điện ngầm: Người dùng có thể quét QR code của bến xe để biết thông tin xe buýt.

- Sử dụng trong bảo tàng: Người dùng chỉ cần quét QR code bên cạnh một cuộc triển lãm để biết thông tin chi tiết và cập nhật về món đồ đó.
- Dùng để mua hàng ở bất cứ đâu: người dùng có thể đặt hàng ngay lập tức thông qua QR code và internet di động nếu họ thích hàng hóa được quảng cáo khi họ đang ở trên tàu điện ngầm, xe buýt....
- Sử dụng trong siêu thị: Nhận thông tin, hướng dẫn nấu ăn và hàm lượng dinh dưỡng của thực phẩm cần mua
- Sử dụng trong hội thảo, thuyết trình, sự kiện: Người tham gia có thể sử dụng QR code thay cho danh thiếp của mình.
- Làm việc với báo giấy và tạp chí: Người đọc có thể quét QR code in trên báo giấy hoặc tạp chí để truy cập phiên bản trực tuyến / di động của báo hoặc tạp chí.
- Đối với lễ hội âm nhạc, live show, quán bar, club: xác định bài hát, nghệ sĩ, ban nhạc, bài hát đang phát, tác giả của bài hát.
- Sử dụng tại nhà hàng, khách sạn, coffee shop: để biết được công thức và cách chế biến món ăn, thức uống, thông tin khách sạn.
- Sử dụng với đồ vật cá nhân (xe, áo thun...) : cung cấp thông tin chi tiết về món hàng, xuất xứ, giá cả.
- Sử dụng trong truyền thông quảng cáo: dần thay thế các quảng cáo dưới hình thức in ấn và phát tờ bướm. QR Code sẽ đính kèm thông tin thương hiệu trên các ấn phẩm, bảng hiệu cửa hàng, các bảng quảng cáo, billboard.
- Sử dụng QR code để tạo sự khác biệt trong chữ ký email thường dùng; trên những món quà, để tạo đặc biệt và bất ngờ cho người nhận, và làm cho người nhận thấy tò mò về nội dung của QR Code đính kèm...

Hiện nay đã tồn tại rất nhiều các app scan QR code với đặc điểm chung của các app là giải mã được một QR code với kích thước lớn, rõ ràng và focus được trọn khung hình trong app. Điều này dẫn đến một vấn đề được đặt ra: khi camera lướt qua những

QR code có kích thước vừa phải hay nhỏ, hoặc là khi có nhiều QR code trong một khung hình thì app không thể giải mã được. Ví dụ khi tham gia các hội nghị hội thảo các diễn giả thường sẽ có những thông tin cần chia sẻ, họ sẽ có nhúng QR code vào một góc nào đó để người tham dự có thể scan được, tuy nhiên với khoảng cách xa với màn hình thì điều này không hề dễ dàng,...và còn nhiều trường hợp khác nữa. Từ những vấn đề trên ta có bài toán được đặt ra:

- ❖ **Đầu vào:** Ảnh hoặc video có chứa QR code (có thể có 1 hoặc nhiều QR code).
- ❖ **Đầu ra:**
 - Các khung nhận diện QR code xuất hiện trong ảnh, video.
 - Nội dung các QR code (Nếu là một đường dẫn đến website thì sẽ đưa ra title để người dùng có thể dễ dàng hình dung hơn về nội dung trang web).

Mục tiêu khóa luận

Trong khóa luận này, chúng tôi nghiên cứu xây dựng một bộ dữ liệu hình ảnh QR code, áp dụng mô hình để đưa ra kết quả khả quan.

- Thứ nhất, chúng tôi xây dựng bộ dữ liệu cho bài toán nhận diện QR code, với bộ dữ liệu muốn xây dựng khoảng 10.000 ảnh QR trong nhiều trường hợp để tăng tính đa dạng giúp mô hình học tốt hơn.
- Thứ hai, chúng tôi tiến hành nghiên cứu mô hình, kỹ thuật liên quan đến bài toán nhận diện và giải mã QR code được nhúng trong các video trực tuyến tăng cường và giải quyết bài toán được đặt ra. Sau đó chạy thử nghiệm mô hình và các kỹ thuật để kiểm chứng độ chính xác và đánh giá hiệu suất của mô hình đã thử nghiệm và chọn ra mô hình tốt nhất cho bộ dữ liệu này.
- Thứ ba, chúng tôi áp dụng mô hình vào thực tế, tạo một ứng dụng nhận diện và giải mã QR được nhúng trong video trực tuyến tăng cường.

Đối tượng và phạm vi nghiên cứu

- **Đối tượng:** Xây dựng bộ dữ liệu hình ảnh QR code. Với các hình ảnh được chụp, được quay trong đời thực cho đến các QR code được trong các video trực tuyến.
- **Phạm vi:** Về phạm vi nghiên cứu, đề tài này tập trung vào việc xử lý các QR code có kích thước vừa và nhỏ, không được tập trung vào giữa khung hình mà có thể chỉ ở một góc nào đó. Và xử lý được nhiều QR code trên cùng một khung hình và xuất ra kết quả cho người dùng.

Kết quả nghiên cứu

Kết quả nghiên cứu chúng tôi thu được như sau:

- Xây dựng được bộ dữ liệu hình ảnh QR code cho việc nhận dạng đối tượng là QR code.
- Cài đặt mô hình học sâu để nhận dạng được các QR code xuất hiện trong một hình ảnh.
- Áp dụng mô hình đã nhận dạng được tạo thành một ứng dụng scan QR code cơ bản.

Cấu trúc khóa luận

Khóa luận gồm 7 chương với các nội dung chính như sau:

➤ **Chương 1: Tổng quan**

Giới thiệu về bài toán đặt ra đối với QR code, sự quan trọng của việc sử dụng QR code trong thời nay và việc scan QR code là một điều không thể thiếu. Các ứng dụng scan QR code còn khuyết thiếu và sẽ được làm lại trong đề tài này.

➤ **Chương 2: Các công trình nghiên cứu liên quan**

Chúng tôi giới thiệu một số công trình nghiên cứu trong và ngoài nước liên quan đến bài toán đặt ra.

➤ **Chương 3: Bộ dữ liệu**

Trong chương này, chúng tôi trình bày quy trình xây dựng bộ dữ liệu: thu thập, tiền xử lý và gán nhãn. Sau đó đề phương pháp cho bài toán và bộ dữ liệu.

➤ **Chương 4: Phương pháp tiếp cận**

Trình bày phương pháp học sâu và thư viện giải mã QR code mà chúng tôi nghiên cứu và áp dụng trên bộ dữ liệu.

➤ **Chương 5:** Cài đặt thử nghiệm và đánh giá

Trong chương này, chúng tôi trình bày các cách đánh giá, các bước cài đặt mô hình và phân tích các trường hợp giải thích cho kết quả đạt được.

➤ **Chương 6:** Xây dựng ứng dụng

Trình bày quy trình xây dựng một ứng dụng scan QR code được áp dụng mô hình đã đào tạo được ở chương 5.

➤ **Chương 7:** Kết luận và hướng phát triển

Tổng kết các kết quả đã đạt được và đề xuất các hướng phát triển trong tương lai để cải thiện được hiệu suất của mô hình.

Chương 1. TỔNG QUAN

1.1. Giới thiệu về tài

Ngày nay, với sự phát triển không ngừng của công nghệ, việc cập nhật thông tin là vô cùng cần thiết và kịp thời. Để trao đổi thông tin với nhau, QR code hiện đang được sử dụng rộng rãi. QR code mang lại sự tiện lợi cho mọi người. QR code sử dụng đơn giản và nhanh chóng, tiết kiệm thời gian và chi phí cho mọi người. QR code có thể được sử dụng để thanh toán, truy cập liên kết hoặc chỉ văn bản ẩn đằng sau chúng. Tất cả những gì chúng ta cần làm là quét và chúng ta sẽ nhận được thông tin ngay lập tức. Do đó, sự tiện lợi của QR code là điều không thể nghi ngờ.

QR code có thể chứa thông tin về địa chỉ web (URL), thời gian sự kiện, thông tin liên hệ, địa chỉ email, tin nhắn SMS, nội dung văn bản và thậm chí cả thông tin. Thông tin vị trí địa lý... tùy theo thiết bị đọc QR code khi quét sẽ dẫn đến trang web, quay số điện thoại, xem tin nhắn... khác nhau.

Việc giải mã một cách nhanh chóng và chính xác QR code cũng là một trong những điều cần thiết. Hàng loạt các ứng dụng giải QR code code ra đời, từ các app scan di động trong app store hay google play thì còn có app scan QR code mặc định trong các điện thoại: Xiaomi, Iphone, Oppo, Samsung, Ngoài ra ở Việt Nam không thể không kể đến Zalo với việc scan QR code một cách nhanh chóng.

Đa số các app đều yêu cầu ảnh scan đầu vào phải to và rõ để việc scan diễn ra nhanh chóng. Và nếu trong ảnh scan có nhiều hơn một QR code thì app chỉ xuất ra QR code được giải mã trước tiên và dừng lại. Vì vậy nếu muốn scan QR code còn lại thì phải di chuyển điện thoại sang QR code còn lại thì app mới giải mã được.

Nhận thấy sự hạn chế này nên chúng tôi đã xây dựng một hệ thống scan QR code với việc sử dụng deep learning để nhận dạng và scan được nhiều QR code cùng một lúc. Cùng với đó là có thể nhận dạng được QR code với các kích thước khác nhau và ở những môi trường khác nhau (nắng chói, tối mờ, nghiêng, ..). Việc áp dụng deep

learning vào hệ thống scan QR có tác dụng tăng cao sự chính xác khi nhận dạng QR code.

Ngoài ra việc tạo một hệ thống scan QR code một cách real-time có thể được tích hợp trong xe tự hành. Với việc xử lý nhiều luồng, hệ thống chạy với nhiều camera giúp xe có thể quét được nhiều QR code khi gặp ngoài đường. Có thể nhận biết những thông tin mà người ta đặt sau nó. Xe tự hành có thể phân tích thông tin và đưa ra các phán đoán chính xác. Giả sử một biển báo giao thông được chèn QR code để người ta biết được đoạn đường này từng xảy ra nhiều lần tai nạn thì xe có thể giảm tốc độ hoặc cảnh báo cho người có mặt trong xe.

1.2. Tính ứng dụng của đề tài

Việc tạo ứng dụng scan QR nhanh mang tính thiết thực đối với cuộc sống bấy giờ, áp dụng vào được nhiều công việc như:

- Scan QR trong các video streaming có nhúng QR code: Các video streaming hay chính xác hơn là các video được quay trong các buổi hội nghị, hội thảo, seminar,... các bài thuyết trình, trình chiếu có thể được người thuyết trình nhúng vào các trang trình chiếu để cung cấp thông tin cho người nghe một cách đầy đủ hơn. Tuy nhiên các QR code này thường được đặt ở các góc của màn hình trình chiếu gây khó khăn trong việc scan. Ứng dụng của chúng tôi có thể dựa vào mô hình học sâu để xác định tọa độ của QR code và tiến hành scan, sẽ tăng độ chính xác cho việc scan hơn.
- Chấm công nhân viên thông qua QR code: Mỗi nhân viên sẽ được cấp 1 QR code và khi nhân viên đứng ở cổng ra vào camera được tích hợp phần mềm scan QR có thể nhanh chóng nhận diện được QR code đó, thay thế cho việc chấm công thủ công bằng vân tay.
- Hỗ trợ cho xe tự hành: Với việc scan QR nhanh chóng và có thể scan cùng một lúc nhiều QR, các camera trên xe có thể đồng thời quét xung quanh và tiếp nhận được nhiều thông tin, tăng cường tính an toàn khi tham gia giao thông.

- Ngoài ra ứng dụng có thể dùng để scan như một app scan thông thường, có thể scan các mã thanh toán trong siêu thị, các shop.. Scan các thông tin cần thiết cho người sử dụng.

1.3. Kết luận

Ở chương này chúng tôi đã giới thiệu về QR code, các lợi ích thực tế và cùng với đó là các bất cập trong các trường hợp khó khăn trong việc giải mã QR code. Đặt ra vấn đề cần giải quyết. Và các ứng dụng thực tế khi giải quyết được các vấn đề đặt ra. Mang lại nhiều lợi ích đối với người dùng.

Chương 2. CÁC CÔNG TRÌNH NGHIÊN CỨU LIÊN QUAN

2.1. Các công trình nghiên cứu liên quan

Bài báo “Fast QR Code Detectionin Arbitrarily Acquired Images” [1] của tác giả Luiz F. F. Belussi và Nina S. T. Hirata công bố năm 2011. Việc phát hiện mã QR, một loại mã vạch 2D, như được mô tả trong tài liệu chỉ bao gồm việc xác định ranh giới của vùng ký hiệu trong hình ảnh thu được với mục đích cụ thể là làm nổi bật biểu tượng. Tuy nhiên, nhiều ứng dụng quan trọng như những ứng dụng liên quan đến công nghệ trợ năng hoặc robot, phụ thuộc vào việc phát hiện đầu tiên sự hiện diện của mã vạch trong môi trường. Nhóm tác giả sử dụng khung phát hiện đối tượng nhanh chóng của Viola-Jone để giải quyết vấn đề tìm mã QR trong các hình ảnh có được tùy ý. Khung này cung cấp một cách hiệu quả để tập trung quá trình phát hiện vào các vùng đầy hứa hẹn của hình ảnh và một cách tiếp cận tính toán tính năng rất nhanh để phân loại mẫu. Một nghiên cứu sâu rộng về các biến thể trong các tham số của khuôn khổ phát hiện các mẫu công cụ tìm kiếm, hiện diện ở ba góc của mỗi mã QR, đã được thực hiện. Độ chính xác phát hiện cao hơn 90%.

Bài báo “An Improvement on QR Code Limit Angle Detection using Convolution Neural Network” [2] của tác giả Wendy Cahya Kurniawan và cộng sự được công bố năm 2019. Nhóm tác giả đề xuất một thuật toán làm tăng khả năng đọc góc của mã QR và tối đa hóa khả năng phát hiện trên các máy ảnh có độ phân giải thấp. Thuật toán mạng thần kinh (CNN) được áp dụng cho việc nhận ra các phiên bản của mã QR khác nhau. Đề xuất cách tiếp cận mã QR ở góc rộng hơn. Các kết quả của nhóm tác giả chứng minh phương pháp tiếp cận đạt độ chính xác trên 90%. Những kết quả này chứng tỏ rằng phương pháp được đề xuất có tiềm năng lớn trong việc phát hiện mã QR từ góc có phôi cảnh rộng hơn.

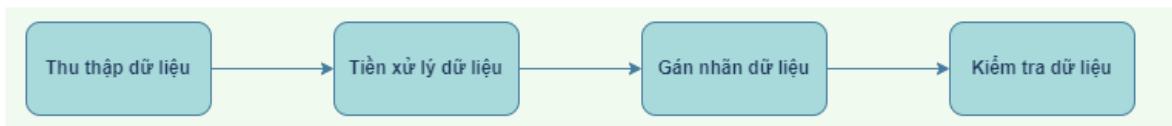
2.2. Kết luận

Nhờ có sự phát triển không ngừng của công nghệ hiện đại, mà các phương pháp học sâu đang ngày càng cải tiến và phát triển. Các khó khăn trong việc nghiên cứu và xử

lý các bài toán đang ngày được cải thiện. Ở bài báo “Fast QR Code Detectionin Arbitrarily Acquired Images” thì phát hiện ba góc của mã QR, còn bài báo “An Improvement on QR Code Limit Angle Detection using Convolution Neural Network” thì phát hiện khung giới hạn ở các khía cạnh khác nhau, hay nói cách khác là các góc độ khác nhau. Cá 2 phương pháp đều có độ chính xác cao đều hơn 90%. Ta có thể thấy được các phương pháp tiếp cận của các tác giả đều mang lại hiệu suất cao và tiềm năng phát triển

Chương 3. BỘ DỮ LIỆU

Đến với giai đoạn đầu tiên trong quá trình thực hiện đề tài, chúng tôi sẽ tạo bộ dữ liệu mới bằng cách thu thập hình ảnh QR code từ nhiều nguồn khác nhau. Quy trình thực hiện thu thập và gán nhãn dữ liệu sẽ được miêu tả ở hình ảnh dưới đây.



Hình 3.1: Quy trình xây dựng tổng quan bộ dữ liệu

3.1. Xây dựng bộ dữ liệu

3.1.1. Thu thập dữ liệu

Chúng tôi xây dựng bộ dữ liệu QR code bằng cách thu thập nhiều nguồn.

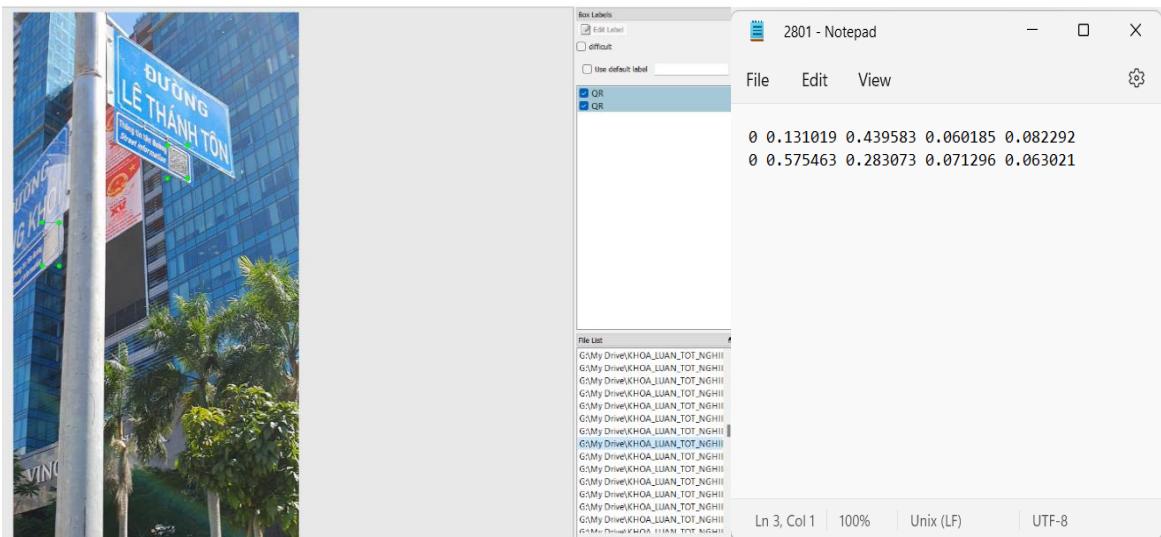
- Thu thập dữ liệu thủ công: quay các video chứa các QR code ở ngoài đời thực, như QR trên biển quảng cáo ở các trung tâm thương mại, băng thanh toán tại các quầy thu ngân, băng rôn trên các con đường,...
- Thu thập dữ liệu bán thủ công: Sử dụng tính năng tìm kiếm của google search để tìm kiếm các ảnh QR. Sau đó tải hết tất cả ảnh về bằng extensions: **Image downloader – Imageye**.

Các ảnh QR code được thu thập từ nhiều nơi, nhiều nguồn, nhiều hướng tăng sự đa dạng cho bộ dữ liệu giúp mô hình có thể học được hiệu quả hơn. Chúng tôi thu thập ảnh từ các trang tìm kiếm thông dụng, có sức chứa nguồn hình ảnh với số lượng lớn và có tính đa dạng cao như Bing, Google... Bên cạnh đó, chúng tôi còn sử dụng các thiết bị thông minh hoặc máy ảnh để quay trực tiếp các trường hợp QR ở ngoài thực tế như những QR code trên biển báo thông tin đường phố, các QR code trên các công trình, các biển cửa tiệm, các biển quảng cáo..(nói chung là các nơi có thể có QR) nhằm tạo ra bộ dữ liệu gần gũi, đa dạng phong phú về các bối cảnh mà QR có thể được đặt vào.

Sau khi thực hiện các bước thu thập xong, thì chúng tôi bắt đầu xử lý dữ liệu thô đã thu thập được. Đối với các video thì chúng tôi sẽ viết script python để cắt các frame ảnh ra từ video. Chúng tôi sẽ lấy ở khoảng 2 ảnh/giây. Bộ dữ liệu thô chúng tôi thu được hơn 10.000 ảnh tuy nhiên có rất nhiều ảnh bị trùng lặp. Vì vậy chúng tôi bắt đầu xóa đi các ảnh trùng lặp và kết quả sau cùng thu được 8.734 ảnh QR cho bộ dữ liệu hoàn thiện để chuẩn bị cho bước gán nhãn tiếp theo.

3.1.2. Gán nhãn dữ liệu

Chúng tôi sử dụng LabelImg – một công cụ gán nhãn cho task object detection của YOLO. Các file nhãn cho mô hình YOLO sẽ có định dạng .txt, trong đó có một file classes.txt gồm nhãn sẽ huấn luyện là QR code (nhãn 0). Các file còn lại sẽ là tên các ảnh và đuôi file nhãn sẽ là .txt. Ví dụ ảnh image.JPG sẽ có file nhãn là image.txt. Trong file nhãn này sẽ chứa thứ tự của nhãn được gán trong file classes.txt (ở đây nhãn của QR code là 0), và 4 con số thập phân liên quan đến thông tin của bounding box.



Hình 3.2: Mô phỏng gán nhãn YOLO bằng tool LabelImg và output của LabelImg

3.2. Chi tiết bộ dữ liệu

Bộ dữ liệu hình ảnh QR code của chúng tôi bao gồm 8.734 ảnh có định dạng “.JPG”.

Chúng tôi cố gắng tạo bộ dữ liệu đa dạng nhất có thể bằng cách chụp nhiều ảnh, nhiều góc độ, nhiều nơi,...

Bộ dữ liệu được chia thành 3 tập train set, validation set và test set. Chúng tôi chia dữ liệu ngẫu nhiên với tỉ lệ train:val:test = 8:1:1. Với các tập train set: 7.151 ảnh, tập validation set: 787 ảnh và tập test set: 796 ảnh.

```
Data:.  
+---trainset  
|   images_QR_15-02-2022_01.jpg  
|   images_QR_15-02-2022_01.txt  
|   .....  
|   images_QR_13-03-2022_09.jpg  
|   images_QR_13-03-2022_09.txt  
+---testset  
|   images_QR_15-02-2022_07.jpg  
|   images_QR_15-02-2022_07.txt  
|   .....  
|   images_QR_22-02-2022_16.jpg  
|   images_QR_22-02-2022_16.txt  
+---validationset  
|   images_QR_16-01-2022_08.jpg  
|   images_QR_16-01-2022_08.txt  
|   .....  
|   images_QR_17-04-2022_05.jpg  
|   images_QR_17-04-2022_05.txt
```

Hình 3.3: Sơ đồ cây tập dữ liệu cho mô hình YOLOv3 và YOLOv4

Đối với mô hình YOLOv5, sự phân bổ dữ liệu sẽ có sự khác biệt, chi tiết được mô tả dưới sơ đồ cây sau:

```
Data:.  
+---images  
    +---trainset  
        |     images_QR_15-02-2022_01.jpg  
        |     images_QR_15-02-2022_02.jpg  
        |     .....  
    +---testset  
        |     images_QR_15-02-2022_07.jpg  
        |     images_QR_15-03-2022_02.jpg  
        |     .....  
    +---validationset  
        |     images_QR_17-04-2022_05.jpg  
        |     images_QR_17-04-2022_09.jpg  
        |     .....  
+---label  
    +---trainset  
        |     images_QR_15-02-2022_01.txt  
        |     images_QR_15-02-2022_02.txt  
        |     .....  
    +---testset  
        |     images_QR_15-02-2022_07.txt  
        |     images_QR_15-03-2022_02.txt  
        |     .....  
    +---validationset  
        |     images_QR_17-04-2022_05.txt  
        |     images_QR_17-04-2022_09.txt  
        |     .....
```

Hình 3.4: Sơ đồ cây tập dữ liệu cho mô hình YOLOv5

3.3. Kết luận

Như vậy chúng tôi đã hoàn thành việc xây dựng bộ dữ liệu hình ảnh QR code với 8.734 ảnh có chứa QR code và được gán nhãn hoàn tất. Đã được chia ra thành 3 tập train set, validation set và test set phục vụ cho quá trình huấn luyện mô hình ở bước tiếp theo.

Chương 4. PHƯƠNG PHÁP TIẾP CẬN

4.1. Detect QR code

Chúng tôi sẽ chạy thực nghiệm trên các mô hình YOLOv3, YOLOv4 và YOLOv5 để tìm ra mô hình có kết quả tối tốt nhất cho việc nhận diện QR code.

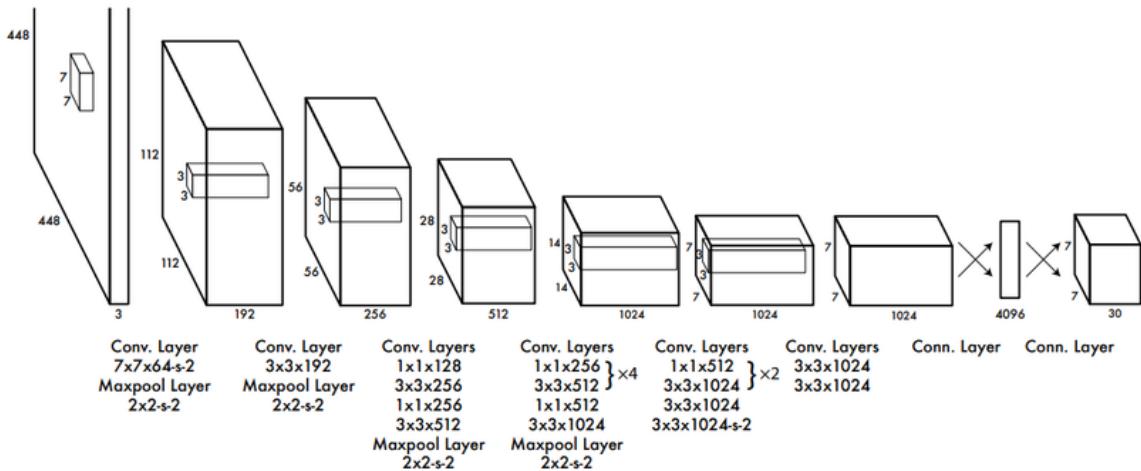
Chúng tôi sử dụng các mô hình YOLOv3, YOLOv4 và YOLOv5 bởi vì các lý do sau đây, đầu tiên nói về cách thức hoạt động của mô hình YOLO so với mô hình R-CNN: mô hình YOLO là one stage detector còn mô hình R-CNN là two stage detector, tức mô hình R-CNN sẽ trải qua 2 giai đoạn: Detection và Pose Estimation, còn mô hình YOLO thì sẽ thực hiện cả 2 công việc tìm bounding box và classification vật thể cùng một lúc. Vì vậy tốc độ nhận dạng của YOLO sẽ nhanh hơn nhiều so với R-CNN và đáp ứng được yêu cầu nhận dạng real-time của bài toán. Tiếp đến lý do chọn YOLOv3 vì đây là phiên bản cuối cùng của tác giả gốc. YOLOv4 là phiên bản được nhiều người tin dùng và có bài báo về mô hình. YOLOv5 phiên bản mới nhất đạt được kết quả cao tuy nhiên chưa có bài báo chính thức. Việc huấn luyện trên cả ba mô hình sẽ tìm được mô hình phù hợp với bộ dữ liệu. Và cuối cùng vì mô hình chỉ nhận dạng một class duy nhất là QR code nên mô hình YOLO sẽ cho kết quả rất cao và chính xác.

Để nói đến các version 3,4,5 của YOLO thì ta sẽ bắt đầu từ YOLO version 1 để hiểu rõ về cấu trúc của YOLO.

4.1.1. YOLOv1

YOLOv1 sử dụng 24 lớp chập: bao gồm lớp (1x1) Reduction layers (dùng để giảm kích thước ảnh), lớp chập (3x3) Convolution layers. Xen kẽ giữa 24 lớp chập là các lớp maxpool layers và kết thúc bằng 2 lớp fully connected. Kết quả là một ma trận 3 chiều có dạng $7 \times 7 \times 30$.

Với mỗi $1 \times 1 \times 30$ trong $7 \times 7 \times 30$ được gọi là một tensor, như vậy đầu ra sẽ có $7 \times 7 = 49$ tensors.

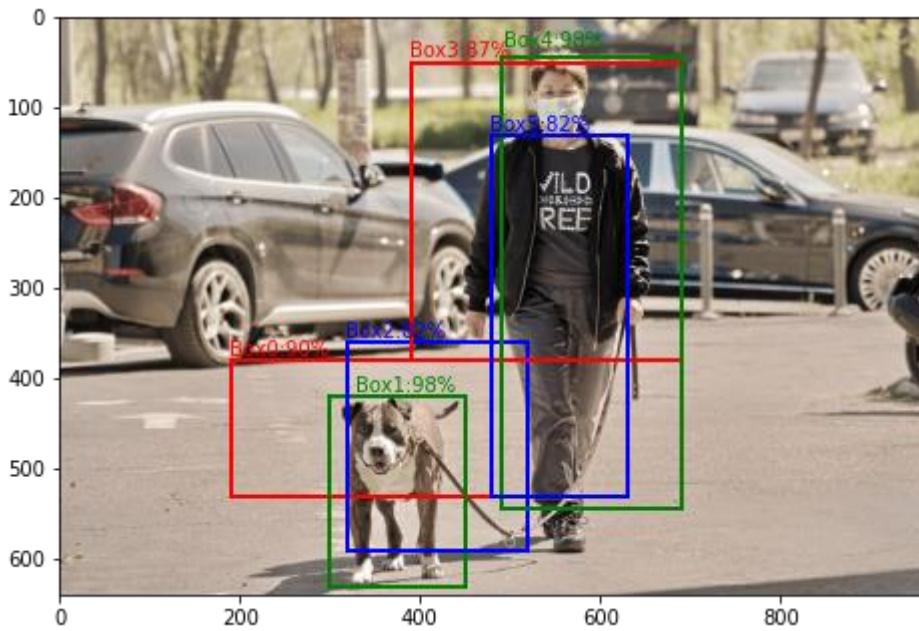


Hình 4.1: Cấu trúc YOLOv1

YOLOv1 chia hình ảnh thành các ô lưới $S \times S$ có kích thước bằng nhau ($S= 3, 5$ hoặc 7), thường là 7×7 . Mỗi ô lưới chịu trách nhiệm phát hiện đối tượng nếu tâm của các đối tượng rơi vào bên trong ô. Mỗi ô có thể dự đoán số lượng B cố định của các hộp giới hạn với điểm tin cậy. Mỗi hộp giới hạn bao gồm 5 giá trị x, y, w, h và điểm tin cậy. Ở đây, x, y, w và h lần lượt là trung tâm của hộp giới hạn, chiều rộng và chiều cao. Sau khi dự đoán hộp giới hạn, YOLO sử dụng IOU (Intersection Over Union) để chọn hộp giới hạn bên phải của một đối tượng cho ô lưới.

Để loại bỏ các hộp giới hạn dư thừa, YOLO sử dụng non-max suppression. Nếu $\text{IOU} \geq \text{IOU-threshold}$ với IOU-threshold là ngưỡng được mình khởi tạo, non-max suppression sẽ loại bỏ các hộp giới hạn dư thừa có điểm tin cậy thấp. Các bước thực hiện NMS như sau:

1. Loại bỏ những đối tượng có $\text{confidence } C < C\text{-threshold}$ (ngưỡng này chúng ta có thể tự khởi tạo).
2. Sắp xếp những đối tượng có $\text{confidence } C$ theo thứ tự giảm dần.
3. Chọn bounding box của những vật thể có $\text{confidence } C$ cao nhất và xuất ra kết quả.
4. Loại bỏ những bounding box có $\text{IOU} < \text{IOU-threshold}$ (ngưỡng này chúng ta có thể tự khởi tạo)
5. Quay lại bước 3 cho tới khi mọi kết quả được kiểm tra.



Hình 4.2: Minh họa non-max suppression

Để tính toán Loss function, YOLO sử dụng hàm “tổng bình phương lỗi” (sum-squared error. Viết tắt là SSE). “Lỗi” được nói đến ở đây chính là lấy giá trị của ground truth box (nhận gán bằng tay) trừ đi bounding box prediction (bounding box mà model dự đoán), sau đó bình phương giá trị này lên.

4.1.2. YOLOv2

Ở YOLOv1 đã xảy ra nhiều vấn đề trong việc xác định vị trí của vật thể, và giá trị recall luôn thấp nên tác giả đã phát triển YOLOv2 với mục đích giải quyết những vấn đề trên. Những cải tiến của tác giả như sau:

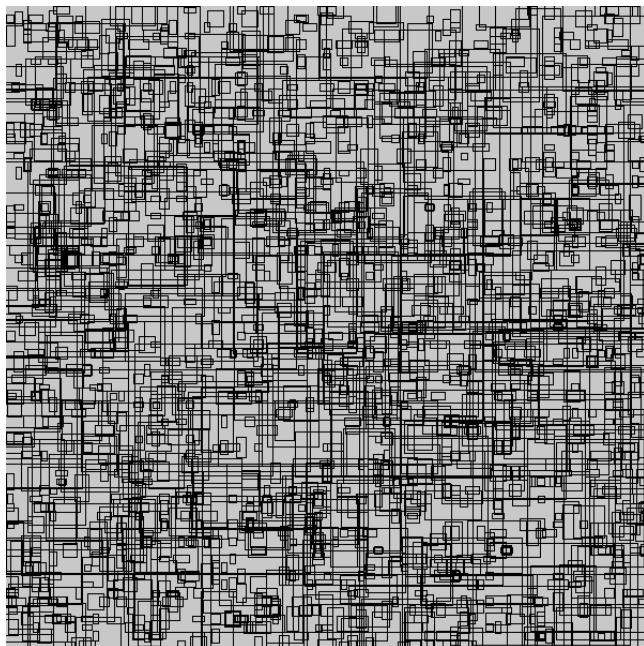
- **BatchNormalization(BN):** Thêm BN vào tất cả các lớp chập. Điều này có thể giúp chúng ta tránh được hiện tượng giá trị rơi vào khoảng bão hòa sau khi đi qua các hàm kích hoạt phi tuyến. Vì thế nó đảm bảo rằng không có sự kích hoạt nào bị quá cao hoặc quá thấp. Giúp cho các weights mà khi không dùng BN có thể sẽ không bao giờ được học thì nay lại được học bình thường. Nó còn giúp chúng ta làm giảm đi sự phụ thuộc vào giá trị khởi tạo của các tham số. Và sau khi áp dụng BN thì giá trị mAP của model đã tăng thêm 2%.
- **Bộ phân loại phân giải cao (High Resolution Classifier):**

- YOLOv1 sẽ đào tạo bộ phân loại với ảnh có kích thước 224×224 , sau đó sẽ tăng kích thước lên 448×448 để nhận dạng.
 - Còn đối với YOLOv2, tác giả vẫn đào tạo bộ phân loại với ảnh có kích thước 224×224 , sau đó đào tạo tiếp với ảnh có kích thước 448×448 ở 10 epochs cuối, cuối cùng mới detection ở ảnh có độ phân giải 448×448 . Việc sử dụng High Resolution Classifier này giúp tăng mAP lên tối đa 4%.
- Phát hiện nhiều đối tượng trong một ô lưới(gird cell):
- YOLOv1: đối với hình trên, mô hình sẽ chỉ có thể nhận dạng được hoặc là người hoặc là cavat trong ô lưới màu đỏ.
 - Nhưng tới YOLOv2, để giải quyết vấn đề đó, tác giả đã cho model dự đoán nhiều bounding box hơn. Và để dự đoán nhiều bounding box thì sử dụng **Anchor box** thay vì lớp fully connected như ở YOLOv1



Hình 4.3: Phát hiện nhiều đối tượng trong một ô lưới

- **Anchor box:** nó là các bounding box nhưng được tạo ra sẵn (còn bounding box sẽ là dự đoán của model). Với một ô lưới sẽ có trách nhiệm tạo ra một số K các Anchor box với các kích thước khác nhau. Các Anchor box này sẽ được dự đoán xem là nó có chứa vật thể hay không, dựa vào kết quả tính toán của IOU giữa nó và ground truth.(nếu $IOU > 50\%$ thì Anchor box đó được coi là chứa vật thể).



Hình 4.4: Minh họa 1% số lượng Anchor box trong mạng RetinaNet

- **Các vật thể nhỏ:** Thay vì YOLOv1 chia các bức ảnh 7×7 ô lưới (hoặc 3×3 và 5×5) thì ở YOLOv2 bức ảnh được chia thành 13×13 ô lưới nên tăng khả năng tìm kiếm được các vật thể có kích thước nhỏ cao hơn so với YOLOv1
- **Được đào tạo trên nhiều kích thước ảnh đầu vào:** đối với YOLOv1 thì chỉ được đào tạo trên một vài kích thước ảnh đầu vào cố định, nhưng sang YOLOv2 thì tác giả đã cho đào tạo mô hình trên các ảnh có kích thước thay đổi từ 320×320 cho đến 640×640 . Điều này làm cho model có thể học được nhiều đặc điểm của đối tượng hơn cũng như cho ra độ chính xác cao hơn.
- **Tensor:** Với việc bỏ đi các lớp fully connected và thay vào đó là sử dụng anchor box, kết quả cuối cùng của mô hình sẽ là : $13 \times 13 \times 125$. Với mỗi tensor có kích thước $1 \times 1 \times 125$ được tính như sau: $k \times (5+20)$, trong đó $k = 5$ và 20 là số lượng lớp vật thể được đào tạo sẵn.
- **Darknet19:** với YOLOv1 sử dụng backbone là 24 lớp convolution thì sang YOLOv2 lại được sử dụng Darknet19 với 19 lớp convolution cùng với 5 lớp maxpool layers (không có các lớp fully connected để dự đoán

mà thay vào đó thì anchor box được sử dụng). Darknet19 xử lý rất nhanh trong việc nhận dạng vật thể nên rất có ý nghĩa trong việc xử lý ở thời gian thực.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Hình 4.5: Cấu trúc Darknet19

4.1.3. Tổng quan YOLOv3, YOLOv4 và YOLOv5

Trong chuẩn hóa hàng loạt YOLOv2 đã được thêm vào cùng với các lớp tích chập để cải thiện độ chính xác và giảm vấn đề trang bị quá mức [3]. Trong YOLOv3, backbone feature extractor của Darknet19 [4], vốn gặp khó khăn trong việc phát hiện các vật thể nhỏ, đã được đổi thành Darknet-53 để giải quyết vấn đề này. Trong công việc đó, Residual block, skip connections và up-sampling đã được giới thiệu, giúp cải thiện đáng kể độ chính xác của thuật toán. Trong YOLOv4 một lần nữa, đường trực trích xuất tính năng đã được thay đổi thành CSPDarknet53, giúp cải thiện đáng kể tốc độ và độ chính xác của thuật toán. YOLOv5 là phiên bản mới nhất và nhẹ của các thuật toán YOLO trước đây và sử dụng khung PyTorch thay vì khung Darknet. Hình 2 cho thấy kiến trúc chung của thuật toán YOLO, và Bảng 2 tóm tắt sự so sánh giữa các kiến trúc thuật toán YOLOv3, YOLOv4 và YOLOv5. Loại mạng đầu và mạng

nơ-ron giống nhau cho tất cả các thuật toán, whereas backbone, neck, và loss function là khác nhau.

	YOLOv3	YOLOv4	YOLOv5
Neural Network Type	Fully convolution	Fully convolution	Fully convolution
Backbone	Darknet-53	CSPDarknet53	CSPDarknet53
Feature Extractor			
Loss Function	Binary cross entropy	Binary cross entropy and Logits loss function	Binary cross entropy and Logits loss function
Neck	FPN	SSP and PANet	PANet
Head	YOLO layer	YOLO layer	YOLO layer

Bảng 4.1: So sánh giữa các cấu trúc của YOLOv3, YOLOv4 và YOLOv5.

4.1.4. YOLOv3

Trong YOLOv3, Darknet-53 được sử dụng làm backbone feature extractor các tính năng từ hình ảnh đầu vào. Backbone của một mạng nơ-ron sâu bao gồm lớp tích chập có chức năng trích xuất các tính năng từ hình ảnh đầu vào. Nó sử dụng features pyramid network (FPN) [5] làm neck. Neck đóng một vai trò quan trọng để trích xuất các feature map từ các giai đoạn khác nhau bao gồm một số đường dẫn từ dưới lên và từ trên xuống và phần đầu bao gồm lớp YOLO. Vai trò của đầu trong máy dò một giai đoạn là thực hiện dự đoán cuối cùng bao gồm một vectơ chứa tọa độ hộp giới hạn: chiều rộng, chiều cao, nhãn của lớp và xác suất lớp. Đầu tiên, hình ảnh được đưa đến Darknet-53 để trích xuất tính năng và sau đó được đưa vào mạng kim tự tháp tính năng để hợp nhất tính năng. Cuối cùng, YOLO layer tạo ra kết quả.

Type	Filters	Size	Output
Convolutional	32	3×3	256×256
Convolutional	64	$3 \times 3 / 2$	128×128
1x	32	1×1	
	64	3×3	
	Residual		128×128
Convolutional	128	$3 \times 3 / 2$	64×64
2x	64	1×1	
	128	3×3	
	Residual		64×64
Convolutional	256	$3 \times 3 / 2$	32×32
8x	128	1×1	
	256	3×3	
	Residual		32×32
Convolutional	512	$3 \times 3 / 2$	16×16
8x	256	1×1	
	512	3×3	
	Residual		16×16
Convolutional	1024	$3 \times 3 / 2$	8×8
4x	512	1×1	
	1024	3×3	
	Residual		8×8
Avgpool		Global	
Connected		1000	
Softmax			

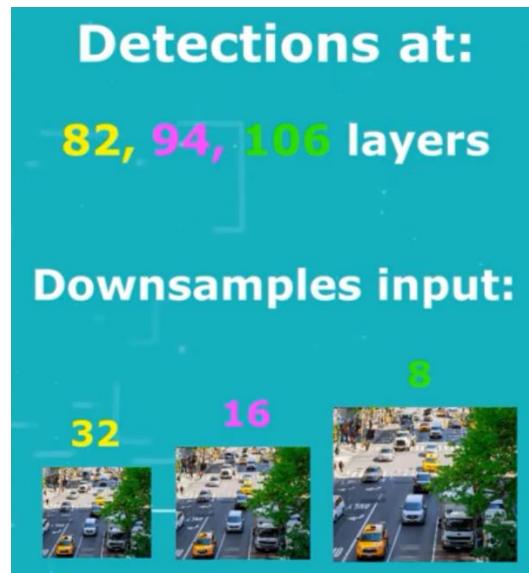
Hình 4.6: Cấu trúc Darknet-53

- **Nhận dạng ảnh đầu vào với các tỉ lệ khác nhau:** Đối với YOLOv1, chúng ta đã biết thuật toán này chia bức ảnh thành các ô lưới với kích thước SxS với S = 3, 5 hoặc 7. Còn đối với YOLOv2 thì kích thước này là 13. Nhưng đối với các kích thước đó, vẫn chưa đủ để có thể tìm kiếm những đối tượng có kích thước nhỏ trong bức hình. Vậy nên YOLOv3 đã xử lý bằng cách là nhận dạng 3 lần trên một bức ảnh với kích thước khác nhau.

Giả sử ta có bức ảnh đầu vào có kích thước : 416×416 :

- Tại lớp chập thứ 82: Bức ảnh được chia thành các ô lưới với kích thước 13×13 (bức ảnh đã được chia với 32). Tại đây, các ô lưới sẽ có trách nhiệm tìm các vật thể có kích thước lớn trong bức hình.

- Tại lớp chập thứ 94: Bức ảnh được chia thành các ô lưới với kích thước 26×26 (bức ảnh được chia với 16). Và có trách nhiệm tìm các vật thể có kích thước trung bình.
- Tương tự, tại lớp chập 106, bức ảnh được chia thành các ô lưới với kích thước 52×52 (ảnh được chia với 8) và có trách nhiệm tìm các vật thể có kích thước bé.
- Bức ảnh sau khi được chia thành các ô lưới, thì được gọi là Downsample images (ảnh giảm mẫu)



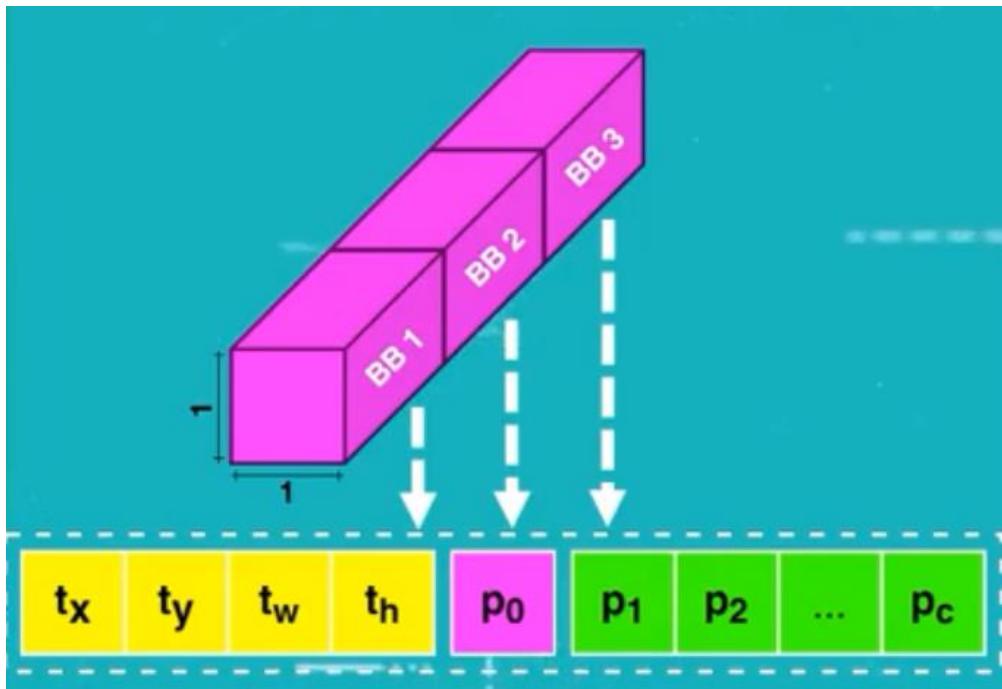
Hình 4.7: Quá trình tìm kiếm vật thể

- **Tensor:** Ở YOLOv1 ta đã biết, kết quả cuối cùng của model là $S \times S$ các tensor (sau 24 lớp chập và các lớp khác, kết thúc là 2 lớp fully connected), với mỗi tensor có kích thước $1 \times 1 \times 30$, còn ở YOLOv2 thì là $13 \times 13 \times 125$ với mỗi tensor có kích thước là $1 \times 1 \times 125$.

Sang tới YOLOv3 thì kết quả đầu ra lại được thay đổi thành $S \times S \times 255$, trong đó:

- S lần lượt được gán với các giá trị: 13, 26, 52.
- Giá trị 255 = $B \times (5+C)$ với $B = 3$ là số lượng bounding box, và $C = 80$ là số lớp vật thể.
- Mỗi bounding box chứa 5 giá trị bao gồm: x, y, w, h và p (xác suất ô lưới có chứa vật thể).

- Để tính ra kết quả cuối cùng thì mỗi bounding box sẽ nhận giá trị P lần lượt với P_1, P_2, \dots, P_c (là xác suất ô lưới đó chứa các vật thể cụ thể) và lựa chọn ra kết quả cao nhất.
- Với mỗi bức ảnh được nhận dạng, thì YOLOv3 sẽ sinh ra $(13 \times 13 + 26 \times 26 + 52 \times 52) \times 3 = 10.647$ bounding box. Từ đó sẽ tính xác suất và xuất ra kết quả là những vật thể có xác suất cao nhất.



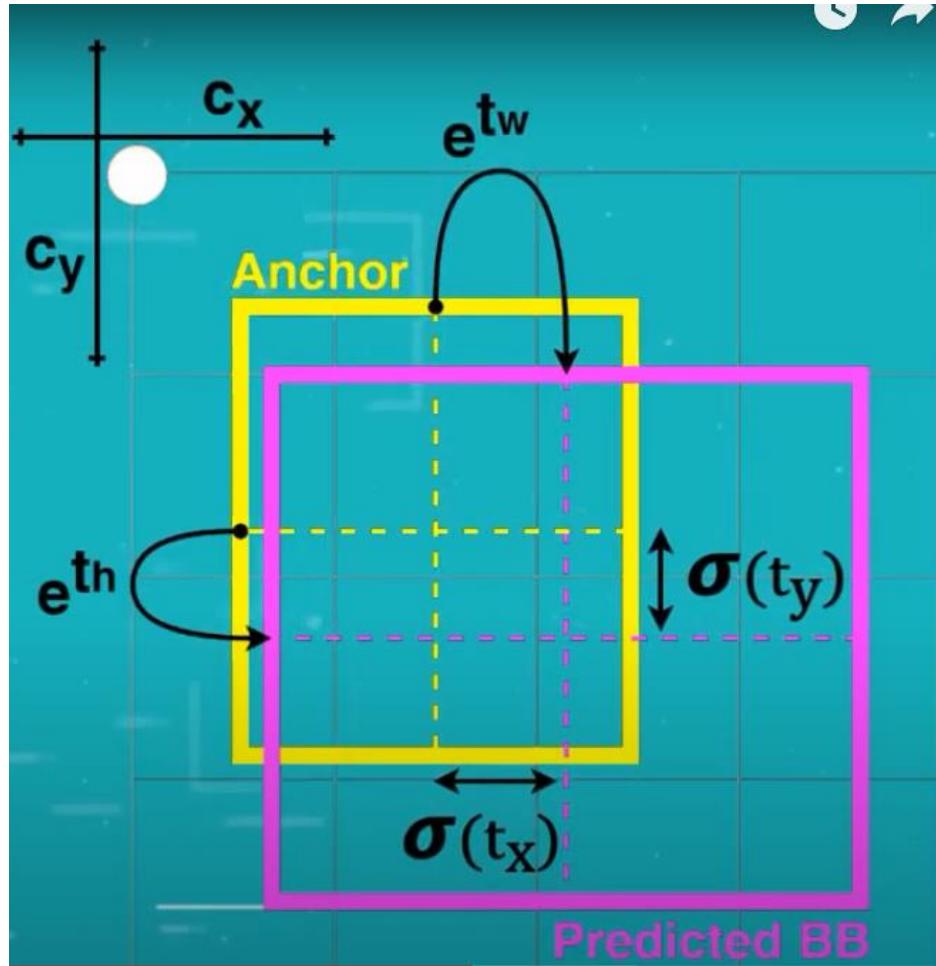
Hình 4.8: Kích thước của tensor

- **Anchor box:**

Với mỗi ô lưới sẽ có 9 anchor box khác nhau với kích thước:

- Ô lưới 13×13 : $(116 \times 90), (156 \times 198), (373 \times 326)$.
- Ô lưới 26×26 : $(30 \times 61), (62 \times 45), (59 \times 119)$.
- Ô lưới 52×52 : $(10 \times 13), (16 \times 30), (33 \times 23)$.

Trong quá trình đào tạo sẽ kết hợp với thuật toán Kmean cluster (thuật toán phân cụm). Cùng với ground truth để tính ra sai sót giữa ground truth và anchor box bằng cách điều chỉnh các giá trị x, y, w, h , từ đó học được các đặc điểm của vật thể.



Hình 4.9: Quá trình đào tạo để tinh chỉnh kích thước của anchor box sao cho giống với vật thể nhất

4.1.5. YOLOv4

Là phiên bản sửa đổi của YOLOv3, YOLO4 sử dụng Cross Stage Partial Network (CSPNet) trong Darknet, tạo ra một feature extractor backbone mới được gọi là CSPDarknet53. Kiến trúc convolution dựa trên DenseNet đã được sửa đổi [6]. Nó chuyển một bản sao của feature map từ base layer sang lớp tiếp theo thông qua dense block. Những lợi thế của việc sử dụng DenseNet bao gồm các ván đê biến mất gradient giảm dần, thúc đẩy backpropagation, loại bỏ nút thắt cổ chai tính toán và cải thiện việc học tập. Neck bao gồm lớp spatial pyramid pooling (SPP) và path aggregation network (PANet). Lớp SPP và path aggregation network (PANet) được sử dụng để tổng hợp tính năng để cải thiện trường tiếp nhận và rút ngắn các features quan trọng từ backbone. Ngoài ra, phần head được cấu tạo từ lớp YOLO. Đầu tiên,

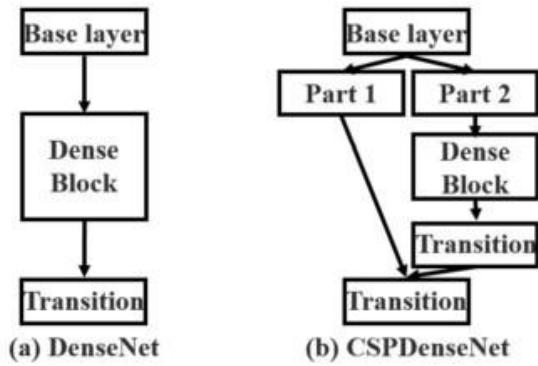
hình ảnh được đưa đến CSPDarknet53 để trích xuất tính năng và sau đó được đưa vào mạng tổng hợp đường dẫn PANet để hợp nhất. Cuối cùng, lớp YOLO tạo ra kết quả, tương tự như YOLOv3, YOLOv4 sử dụng bag of freebies [7] và bag of specials [8] để cải thiện hiệu suất thuật toán. Bag of freebies bao gồm Complete IOU loss (CIOU), drop block regularization và các kỹ thuật tăng cường khác nhau. Bags of specials bao gồm mish activation, Diou-NMS [9] và sửa đổi path aggregation network (PANet).

Tiếp đến ta sẽ đi sâu vào từng phần:

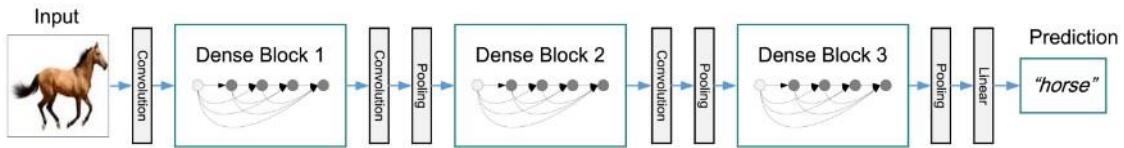
- **CSPDarknet53:** được cấu tạo từ CSP, Darknet53.

CSP (Cross-Stage-Partial connections) có nguồn gốc từ kiến trúc DenseNet sử dụng đầu vào trước đó và nối nó với đầu vào hiện tại trước khi chuyển vào Dense layer. Nó có nhiệm vụ chia đầu vào của khối thành 2 phần, một phần sẽ qua các khối chập, và phần còn lại thì không (đi thẳng tới cuối khối). sau đó hai phần sẽ được cộng lại và đưa vào khối tiếp theo. Ý tưởng ở đây là loại bỏ các nút thắt tính toán trong DenseNet và cải thiện việc học bằng cách chuyển phiên bản chưa chỉnh sửa của feature maps (bản đồ đặc điểm)

DenseNet (Dense connected convolutional network) là một trong những network mới nhất cho visual object recognition. Nó cũng gần giống Resnet nhưng có một vài điểm khác biệt. Densenet có cấu trúc gồm các dense block và các transition layers. Được stack dense block- transition layers-dense block- transition layers như hình vẽ. Với CNN truyền thông nếu chúng ta có L layer thì sẽ có L connection, còn trong densenet sẽ có $L(L+1)/2$ connection (tức là các lớp phía trước sẽ được liên kết với tất cả các lớp phía sau nó).



Hình 4.10: Mô tả cấu trúc CSP



Hình 4.11: Cấu trúc DenseNet

Yolov4 sử dụng CSPDarknet53 để làm backbone vì theo tác giả, CSPDarknet53 có độ chính xác trong task object detection cao hơn so với ResNet; và mặc dù ResNet có độ chính xác trong task classification cao hơn, hạn chế này có thể được cải thiện nhờ hàm activation Mish và một vài kỹ thuật khác.

- **Neck:**

Neck có nhiệm vụ trộn và kết hợp các bản đồ đặc trưng (features map) đã học được thông qua quá trình trích xuất đặc trưng (backbone) và quá trình nhận dạng (YOLOv4 gọi là Dense prediction).

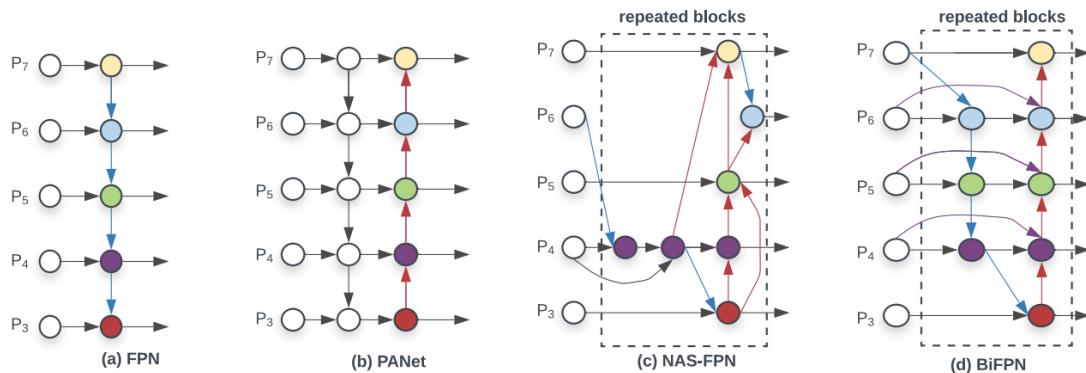
Với mỗi lần thực hiện detect với các kích thước ảnh rescale khác nhau tác giả đã thêm các luồng đi từ dưới lên và các luồng đi từ trên xuống vào cùng nhau theo từng hoặc được nối với nhau trước khi đưa vào head (phần đầu), từ đó lớp nhận dạng sẽ chứa thông tin phong phú hơn.

Tác giả của YOLOv4 đã cho phép tùy biến sử dụng các cấu trúc cho phần Neck như là :

- FPN
- PAN
- NAS-FPN

- BiFPN
- ASFF
- SFAM
- SSP

Một số cấu trúc điển hình:



Hình 4.12: Một số cấu trúc sử dụng cho phần cổ(Neck).

- **Head:**

YOLOv4 sử dụng phần head giống như YOLOv3 với các anchor box và nhận dạng với ảnh có kích thước khác nhau.

- **Bag of freebies:**

- **CIoU-loss:** Để cập nhật trọng số cho mô hình, các phương pháp object detection thường dùng hàm loss là IoU.

Tuy nhiên, khi ta sử dụng IoU, xét 2 trường hợp prediction bounding box và groundtruth bounding box không overlapping, ta không thể nói trường hợp nào tốt hơn trường hợp nào, do đó gây khó khăn cho việc cập nhật trọng số theo hướng khiến prediction bounding box tiến gần tới groundtruth bounding box. GIoU giải quyết điều này bằng cách thêm vào IoU một thành phần C: bounding box nhỏ nhất mà chứa cả prediction bounding box và groundtruth bounding box (xem như khoảng cách giữa 2 bounding box này).

Tuy nhiên, khi sử dụng GIoU lại có một vấn đề là mô hình có khuynh hướng mở rộng prediction bounding box trước cho tới khi nó overlapping với groundtruth, sau đó mới co lại để giảm IoU.

DIoU giải quyết được vấn đề này bằng cách không chỉ đưa bounding box C vào ràng buộc mà còn đưa khoảng cách giữa tâm của prediction bounding box và tâm của groundtruth bounding box. Bây giờ, thành phần C ở mẫu chỉ đóng vai trò chuẩn hóa khoảng cách giữa 2 tâm của prediction bounding box và groundtruth bounding box.

Cuối cùng, CIoU đưa thêm vào tham số giúp duy trì tỷ lệ của các bounding box.

- **Drop block regularization:**

Trong Dropout, giả thiết là các điểm gần nhau thường có đặc điểm giống nhau, do đó ta có thể loại bỏ các điểm này bằng cách set weight = 0 tại một số vị trí trên feature map.

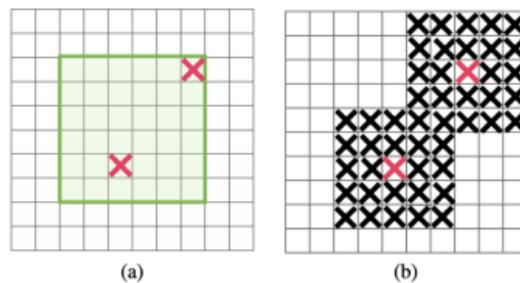
Trong DropBlock, các vị trí được chọn không phân bố ngẫu nhiên nữa mà tập trung thành các block.

Algorithm 1 DropBlock

```

1: Input: output activations of a layer ( $A$ ),  $block\_size$ ,  $\gamma$ ,  $mode$ 
2: if  $mode == Inference$  then
3:   return  $A$ 
4: end if
5: Randomly sample mask  $M$ :  $M_{i,j} \sim Bernoulli(\gamma)$ 
6: For each zero position  $M_{i,j}$ , create a spatial square mask with the center being  $M_{i,j}$ , the width,
   height being  $block\_size$  and set all the values of  $M$  in the square to be zero (see Figure 2).
7: Apply the mask:  $\hat{A} = A \times M$ 
8: Normalize the features:  $A = \hat{A} \times \text{count}(M) / \text{count\_ones}(M)$ 

```



Hình 4.13: Drop block

- **Phương pháp khác:** Những phương pháp giúp cải thiện kết quả inference mà không làm ảnh hưởng tới tốc độ inference. Những phương pháp này

thường là data augmentation, class imbalance, cost function, soft labeling,

...

Data augmentation: CutMix data augmentation, Mosaic data augmentation.

Tác giả đã có nghiên cứu về việc ứng dụng từng gói riêng biệt và kết hợp và đưa ra sự ảnh hưởng của nó đối với kết quả của mô hình:

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
✓							77.9%	94.0%
	✓						77.2%	94.0%
		✓					78.0%	94.3%
			✓				78.1%	94.5%
				✓			77.5%	93.8%
					✓		78.1%	94.4%
						✓	64.5%	86.0%
✓	✓					✓	78.9%	94.5%
✓	✓			✓			78.5%	94.8%
✓	✓			✓		✓	79.8%	95.2%

Hình 4.14: Bag of Freebies.

o **Bags of specials:**

- **Mish activation:** Theo tác giả, sử dụng Mish thu được kết quả tốt hơn so với ReLu, SoftPlus, Swish cũng như một số activation function khác (Adam, Ranger, RangerLars, Novograd, ...)

$$f(x) = x \tanh(\ln(1 + e^x)) \quad (1)$$

Tối thiểu của $f(x)$ là ≈ 0.30884 tại $x \approx -1.1924$

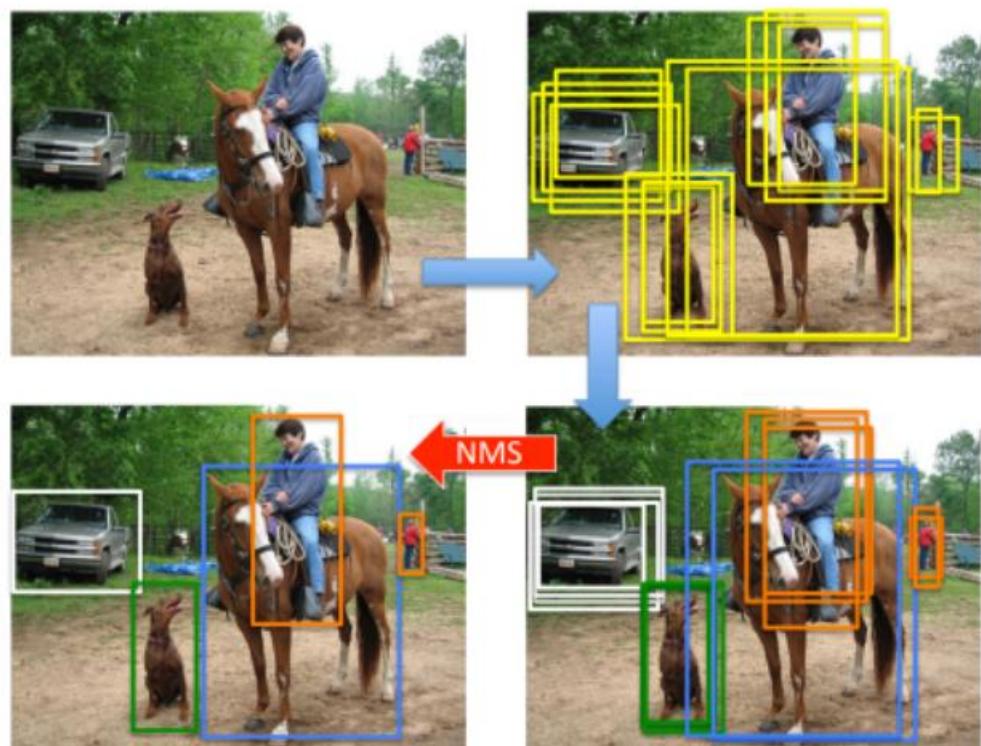
Một số thuộc tính quan trọng của hàm Mish: Không có cận trên; Có cận dưới; Không đơn điệu, giữ lại một phần nhỏ negative gradient cho phép mô hình học được tốt hơn; Liên tục, Mish có đạo hàm bậc 1 tại mọi điểm thuộc miền giá trị.

Tính smooth của Mish được thể hiện bằng cách tăng số lượng hidden layer của network, đo độ chính xác của mô hình khi sử dụng ReLu/Swish/Mish.

Ta có thể thấy performance của Mish không bị giảm quá nhiều khi tăng số lượng layer như đối với ReLu và Swish.

- **Diou-NMS:**

NMS lọc ra các prediction bounding box cho cùng một object và chỉ lấy bounding box có confident cao nhất. Confident ở đây được đo bằng IoU.



Hình 4.15: Minh họa NMS

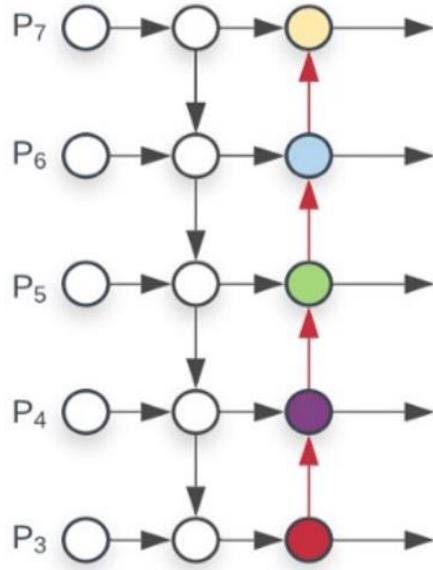
DIoU-NMS có thay đổi một chút là confident được đo bằng DIoU.

- **Path aggregation network (PANet):**

Mạng DeepLearning càng sâu thì càng làm mất thông tin, do đó, để detect được các đối tượng có kích thước nhỏ, researchers đã đề xuất ra nhiều phương pháp như DenseBlock, FPN. PANet là một cải tiến của FPN nhằm cải thiện localized information trên các top layers.

PANet thêm 1 bottom-up pathway, trong đó, mỗi layer lấy input là feature maps của stage trước đó, đi qua một conv 3×3 . Output được add với feature

map của top-down pathway với stage tương ứng. Thiết kế của Neck có thể được mô tả như hình dưới:



Hình 4.16: PANet

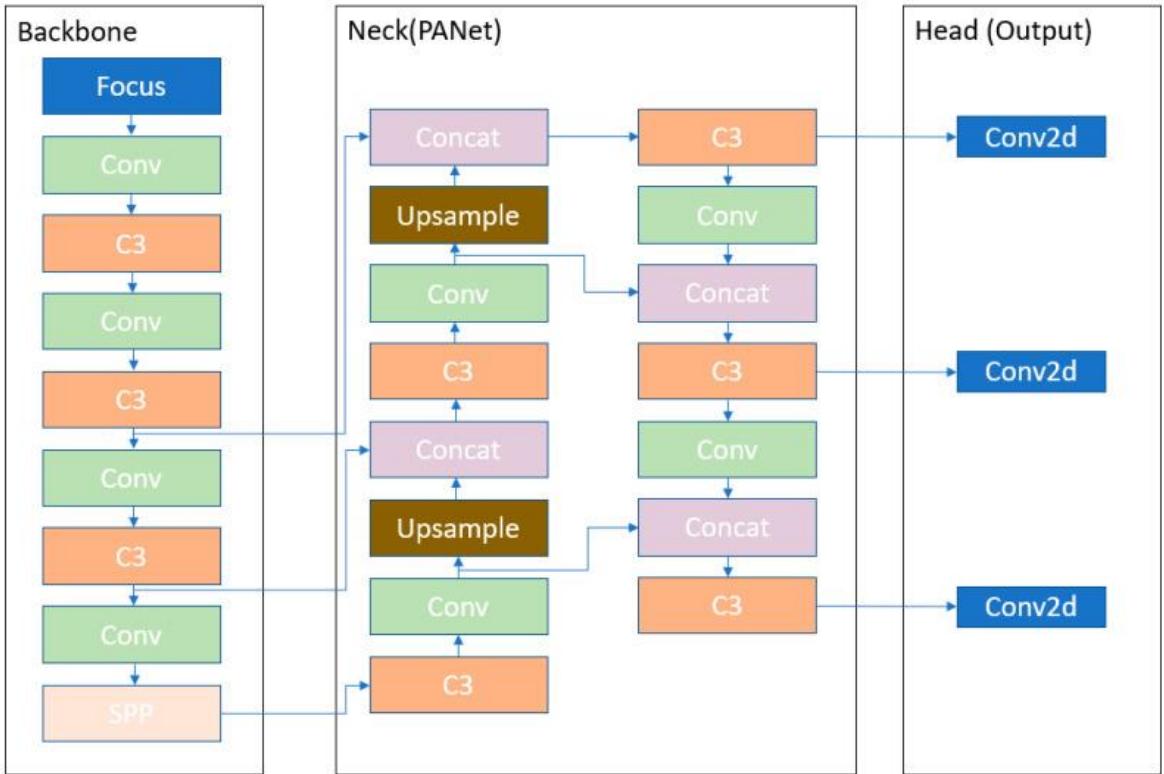
Trong Yolov4, các tác giả chỉnh sửa hàm add thành hàm concat.

Tại các stage của augmented bottom-up pathway, object được detect một cách độc lập với các kích thước khác nhau. Điều này có thể dẫn tới sự dư thừa về dữ liệu hoặc tại mỗi stage sẽ không sử dụng thông tin từ các stage khác. Do đó, tại mỗi stage, các feature maps sẽ được đẩy qua một mạng SPP (=ROIAlign), rồi sau đó đưa qua lớp fully connected layer, kết quả thu được sau các lớp fully connected layer này sẽ được element-wise max operation để thu được prediction.

4.1.6. YOLOv5

YOLOv5 khác với các version trước đó. Nó sử dụng PyTorch thay vì Darknet. Nó sử dụng CSPDarknet53 làm backbone. Đường trực này giải quyết thông tin gradient lặp đi lặp lại trong các đường trực lớn và tích hợp thay đổi gradient vào bản đồ tính năng giúp giảm tốc độ suy luận, tăng độ chính xác và giảm kích thước mô hình bằng cách giảm các tham số. Nó sử dụng path aggregation network (PANet) làm cỗ để thúc đẩy luồng thông tin. PANet áp dụng một feature pyramid network (FPN) bao gồm một số lớp

từ dưới lên và từ trên xuống. Điều này cải thiện sự lan truyền của các tính năng cấp thấp trong mô hình. PANet cải thiện localization ở các lớp thấp hơn, giúp tăng cường độ chính xác localization của đối tượng. Ngoài ra, head trong YOLOv5 giống như YOLOv4 và YOLOv3 tạo ra ba đầu ra khác nhau của feature maps để đạt được dự đoán đa tỷ lệ. Nó cũng giúp tăng cường dự đoán các đối tượng từ nhỏ đến lớn một cách hiệu quả trong mô hình. Hình ảnh được đưa đến CSPDarknet53 để trích xuất tính năng và một lần nữa được đưa đến PANet để hợp nhất tính năng. Cuối cùng, layer YOLO tạo ra kết quả. Trong kiến trúc của thuật toán YOLOv5l được trình bày. Lớp Focus [10] được phát triển từ cấu trúc YOLOv3. Nó thay thế ba lớp đầu tiên của YOLOv3 và tạo một lớp duy nhất trong YOLOv5. Ngoài ra, ở đây Conv biểu thị một lớp tích chập. C3 bao gồm ba lớp tích chập và một mô-đun được xếp tầng bởi các nút thắt cỗ chai khác nhau. Spatial pyramid pooling (SPP) là một lớp gộp được sử dụng để loại bỏ ràng buộc kích thước cố định của mạng. Upsample được sử dụng trong việc lấy mẫu tổng hợp lớp trước đó trong node gần nhất. Concat là một lớp cắt lát và được sử dụng để cắt lớp trước đó. Cuối cùng 3 Conv2d là các mô-đun phát hiện được sử dụng trong head của mạng.



Hình 4.17: Kiến trúc YOLOv5

Sự khác biệt chính giữa kiến trúc YOLOv3, YOLOv4 và YOLOv5 là YOLOv3 sử dụng backbone Darknet53. Kiến trúc YOLOv4 sử dụng CSPdarknet53 làm backbone và YOLOv5 sử dụng cấu trúc Focus với CSPdarknet53 làm backbone. Focus layer lần đầu tiên được giới thiệu trong YOLOv5. Focus layer thay thế ba layer đầu tiên trong thuật toán YOLOv3. Ưu điểm của việc sử dụng lớp Focus là giảm bộ CUDA memory, reduced layer, increased forward propagation, và backpropagation [10].

4.2. DECODE QR CODE

Chúng tôi sử dụng 2 framework để decode QR code là Pyzbar (Zbar) và QRCodeDetector (opencv-python).

4.2.1. Pyzbar (Zbar)

Pyzbar là một thư viện của Zbar bar code reader dành cho python. ZBar là một bộ phần mềm mã nguồn mở để đọc mã vạch từ nhiều nguồn khác nhau, chẳng hạn như luồng video, tệp hình ảnh,...

Việc triển khai linh hoạt, phân lớp tạo điều kiện thuận lợi cho việc quét và giải mã mã vạch cho bất kỳ ứng dụng nào: sử dụng nó độc lập với các chương trình GUI và dòng lệnh đi kèm, dễ dàng tích hợp tiện ích quét mã vạch vào ứng dụng, tận dụng một trong các tập lệnh hoặc giao diện lập trình (Python, Perl, C++) ... tất cả các cách xuống một thư viện C được sắp xếp hợp lý phù hợp để sử dụng nhúng.

Các tính năng được giới thiệu của Zbar:

- Đa nền tảng - Linux / Unix, Windows, iPhone®, nhúng ...
- Tốc độ cao - quét *thời gian thực* từ các luồng video
- Vùng phủ bộ nhớ nhỏ
- Kích thước mã nhỏ - máy quét lõi và bộ giải mã EAN đại diện dưới 1K dòng mã C
- Không giới hạn ở hình ảnh
- Không có hoạt động dấu phẩy động
- Thích hợp cho các ứng dụng nhúng sử dụng bộ xử lý / phần cứng rẻ tiền
- Các thành phần mô-đun có thể được sử dụng cùng nhau hoặc riêng biệt

4.2.2. QRCodeDetector (Opencv-python)

OpenCV là tên viết tắt của open source computer vision library – có thể được hiểu là một thư viện nguồn mở cho máy tính. Cụ thể hơn OpenCV là kho lưu trữ các mã nguồn mở được dùng để xử lý hình ảnh, phát triển các ứng dụng đồ họa trong thời gian thực.

OpenCV cho phép cải thiện tốc độ của CPU khi thực hiện các hoạt động real time. Nó còn cung cấp một số lượng lớn các mã xử lý phục vụ cho quy trình của thị giác máy tính hay các learning machine khác.

Thư viện OpenCV được phát hành với giấy phép BDS. Do đó các dịch vụ nó cung cấp là hoàn toàn miễn phí và được hạn chế tối đa các rào cản thông thường. Cụ thể, bạn được phép sử dụng phần mềm này cho cả hoạt động thương mại lẫn phi thương mại. OpenCV sở hữu giao diện thiên thiện với mọi loại ngôn ngữ lập trình, ví dụ như

C++, C, Python hay Java... Ngoài ra, nó cũng dễ dàng tương thích với các hệ điều hành khác nhau, bao gồm từ Windows, Linux, Mac OS, iOS cho đến cả Android.

Kể từ lần đầu xuất hiện từ năm 1999, giờ đây OpenCV đã sở hữu đội ngũ người dùng hùng hậu, con số ước tính có thể lên tới 47.000 người. Tất cả là nhờ những ưu điểm vượt trội của OpenCV.

OpenCV đang được sử dụng rộng rãi trong các ứng dụng bao gồm: Hình ảnh street view; Kiểm tra và giám sát tự động Robot và xe hơi tự lái; Phân tích hình ảnh y tế; Tìm kiếm và phục hồi hình ảnh/video phim - cấu trúc 3D từ chuyển động Nghệ thuật sắp đặt tương tác

OpenCV có các chức năng: Image/video I/O, xử lý, hiển thị (core, imgproc, highgui) Phát hiện các vật thể (objdetect, features2d, nonfree) Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab) Computational photography (photo, video, superres) Machine learning & clustering (ml, flann) CUDA acceleration (gpu).

Trong khóa luận này chúng tôi có sử dụng nhiều kiến thức liên quan đến OpenCV để xử lý ảnh và scan QR code với lớp **QRCodeDetector** của OpenCV.

4.3. Kết luận

Như vậy chúng tôi sẽ áp dụng các mô hình YOLO được nói ở trên cho quá trình nhận dạng để có thể so sánh được khả năng chính xác của từng mô hình. Cùng với đó là 2 thư viện open source là Pyzbar của Zbar và QRCodeDetector của OpenCV. Với việc nhận dạng được các hình ảnh QR từ trước sẽ nâng cao được độ chính xác cho các thư viện thực hiện giải mã QR code.

Chương 5. CÀI ĐẶT THỬ NGHIỆM VÀ ĐÁNH GIÁ

Trong chương này, chúng tôi sẽ tiến hành cài đặt theo mô hình đã giới thiệu trong phần phương pháp tiếp cận. Sau khi có kết quả thử nghiệm chúng tôi tiến hành phân tích đánh giá kết quả đạt được.

5.1. Cài đặt thử nghiệm

Trong khóa luận này, chúng tôi cài đặt các mô hình học sâu được đánh giá là tốt nhất trong việc nhận dạng đối tượng ở thời điểm hiện tại. Đó là các mô hình họ YOLO (You only look once). Chúng tôi thực nghiệm 3 mô hình: YOLOv3, YOLOv4 và YOLOv5. Mô hình YOLOv3 và YOLOv4 sử dụng Darknet còn YOLOv5 thì sử dụng PyTorch.

5.1.1. YOLOv3

Tiến hành huấn luyện mô hình YOLOv3 trên tập dữ liệu train set. Các bước huấn luyện mô hình YOLOv3 như sau:

- Đầu tiên clone source Darknet có chứa các mô hình YOLO (v1 đến v4).
- Sau khi clone hoàn tất chúng tôi chỉnh sửa makefile để huấn luyện bằng GPU (Chúng tôi sử dụng google colab để huấn luyện mô hình). Cụ thể là: GPU=1, CUDNN=1, CUDNN_HALF=1, LIBSO=1.
- Tạo file YOLO.names chứa nhãn huấn luyện, ở đây chúng tôi để nhãn là QR code.
- Tạo file YOLO.data để áp dụng khi thực thi huấn luyện, gồm có: classes (số lượng lớp được huấn luyện), train (các đường dẫn đến ảnh được huấn luyện), valid (các đường dẫn đến ảnh validation), names (file YOLO.names được tạo phía trên để nhận biết nhãn đó là lớp nào) và cuối cùng là backup (nơi sẽ lưu lại file weight chúng ta đã huấn luyện được).
- Chỉnh sửa file YOLOv3-custom.cfg trong thư mục cfg để cài đặt các lớp cho mạng YOLOv3. Cụ thể các tham số sẽ được điều chỉnh như sau: width=416, height=416 (Giảm cách thước ảnh để tránh trường hợp bị out

memory); batch_size=64; subdivision=16; max_batches=6000; classes=1 (QR code); filter=(số class + 5)*3=18.

- Sau khi chỉnh sửa hoàn tất tất cả các file trên, chúng tôi chạy file Makefile bằng lệnh !make. Sau đó chúng tôi bắt đầu huấn luyện mô hình với file weight ban đầu là file “darknet53.conv.74”. Trong quá trình huấn luyện vì google colab giới hạn thời gian sử dụng GPU nên việc huấn luyện bị gián đoạn. Để tiếp tục huấn luyện chúng tôi thay đổi file weight ban đầu thành “YOLOv3-custom_last.weights” để tiếp tục huấn luyện.

5.1.2. YOLOv4

YOLOv4 được huấn luyện các bước hoàn toàn giống YOLOv3, thay vì thay đổi các thông số ở file YOLOv3-custom.cfg thì chỉnh sửa ở file YOLOv4-custom.cfg. Và khi huấn luyện file weights ban đầu thay vì là “darknet53.conv.74” thì chúng tôi sử dụng “YOLOv4.conv.137”.

Chúng tôi tiếp tục tiến hành huấn luyện mô hình YOLOv4-tiny vì mô hình yolov4 khá nặng, chạy real-time không được tối ưu so với YOLOv4-tiny. Tương tự với việc huấn luyện mô hình YOLOv3 và YOLOv4, mô hình YOLOv4-tiny cũng phải sửa các tham số ở file cấu hình YOLOv4-tiny-custom.cfg và file weight ban đầu sẽ là “yolov4-tiny.conv.29”.

5.1.3. YOLOv5

Đối với YOLOv5 chúng tôi sử dụng framework PyTorch để huấn luyện mô hình. Các bước huấn luyện mô hình YOLOv5 được thực hiện như sau:

- Đầu tiên chúng tôi clone thư mục YOLOv5 bằng lệnh !git clone <https://github.com/ultralytics/YOLOv5>. Sau đó chúng tôi cài đặt môi trường bằng lệnh !pip install -r requirements.txt.
- Tiếp đến để chuẩn bị huấn luyện chúng tôi sẽ tạo 2 file custom_dataset.yaml và custom_model.yaml.

- Đổi với file custom_dataset.yaml chúng tôi tạo 2 đường dẫn đến thư mục của hình ảnh. Chính sửa số nc=1 (num class) và names = “QR code” (class names).
- Đổi với file custom_model.yaml chúng tôi sử dụng model YOLOv5s, và chỉnh sửa số class từ 80 thành 1 cho phù hợp với dataset.
- Và cuối cùng chúng tôi huấn luyện mô hình với các tham số cho việc huấn luyện như --img 416 (ảnh được đưa về kích thước 416x416) --batch 16, --epochs 500, --data custom_data.yaml –cfg custom_model.yaml.

Như vậy chúng tôi đã hoàn thành việc huấn luyện mô hình và kết quả huấn luyện sẽ được trình bày ở phần sau

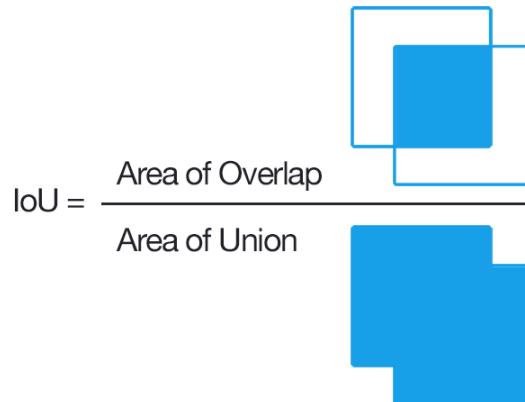
5.2. Phương pháp đánh giá

5.2.1. IoU:

Intersection over Union là chỉ số đánh giá được sử dụng để đo độ chính xác của detection task trên tập dữ liệu cụ thể. IOU thường được đánh giá hiệu năng của các mô hình detection.

Để tính IoU ta cần có hai bounding box: Thứ nhất là bounding box thực sự của đối tượng (chính là bounding box đã gán nhãn). Thứ hai là bounding box của đối tượng được dự đoán bởi mô hình.

Cách tính IoU sẽ là tỉ lệ diện tích giao nhau giữa bounding box thực sự và bounding box dự đoán với diện tích hợp của hai bounding box đó. Được minh họa ở hình dưới đây.



Hình 5.1: Minh họa cách tính độ đo IoU

5.2.2. True/false positive/negative:

Kết quả trả về của IoU sẽ nằm trong khoảng từ 0 đến 1, và mỗi detection sẽ có một IoU riêng. Để xác định được wrong detection hay correct detection thì chúng ta phải dựa vào một ngưỡng cho trước tùy vào bài toán (0.25, 0.5, 0.75,...). Nếu IoU của detection lớn hơn hoặc bằng so với ngưỡng cho trước thì đó là correct detection, ngược lại thì wrong detection.

Từ đó ta có định nghĩa về True/false positive/negative:

- True Positive (TP): IoU lớn hơn hoặc bằng ngưỡng, là một correct detection. Trong trường hợp có nhiều box có IoU lớn hơn ngưỡng thì box có IoU lớn nhất sẽ được tính là TP, còn lại là FP.
- False Positive (FP): IoU bé hơn ngưỡng (là một wrong detection).
- False Negative (FN): trường hợp mà có bouding box nhưng không dự đoán.

5.2.3. Precision

Là thang đo độ chính xác của dự đoán.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

5.2.4. Recall

Là thang đo độ tốt của khả năng tìm thấy được các correct detection.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

5.2.5. AP

Từ precision và recall được định nghĩa ở trên chúng ta có thể đánh giá mô hình bằng cách thay đổi ngưỡng và giá trị quan sát của precision và recall. Khái niệm Area Under the Curve (AUC) cũng được định nghĩa tương tự. Với Precision-Recall Curve, AUC còn có một tên khác là Average precision (AP). Giá trị AP là giá trị phía dưới đường biểu diễn mối quan hệ precision – recall. Tại mỗi recall level, ta thay giá trị precision bằng giá trị precision lớn nhất tại recall level đó. AP sẽ bằng phần diện tích phía dưới đường biểu diễn precision-recall bằng cách tính tổng các hình chữ nhật xấp xỉ.

Công thức:

$$P_{interp}(rn+1) = max_{r'} r' >= rn (r') \quad (4)$$

$$AP = \sum (rn+1 - rn) P_{interp}(rn+1) \quad (5)$$

5.2.6. mAP

Bài toán Object Detection của chúng ta có nhiều class, mỗi class ta sẽ tiến hành đo AP, sau đó lấy trung bình của tất cả các giá trị AP của các class ta được chỉ số mAP của mô hình.

5.2.7. Accuracy decode:

$$Accuracy = \frac{\text{Số QR decode đúng}}{\text{Tổng số QR}} \quad (6)$$

Trong đó:

- + Số QR decode đúng là số lượng QR code của bộ dữ liệu test sau khi qua hệ thống detect và decode đạt được kết quả đúng
- + Tổng số QR là số lượng của tất cả các QR code mà decode bằng cách thủ công được trong bộ test.

Ví dụ: Có 100 ảnh trong bộ test, và có tổng 150 QR trong 100 ảnh đó. Và trong 150 QR code này chúng ta có thể decode thủ công được 120 QR (30 QR còn lại không thể decode được bởi các trường hợp bất khả kháng như có thể bị nhòe, bị cắt 1 phần, bị quá chói hoặc quá tối hoàn toàn không thể decode được.). Thì Tổng số QR = 120. Sau khi 100 ảnh test qua hệ thống và đưa ra kết quả có được 90 QR được giải mã thì Accuracy sẽ bằng $90/120 = 0.75$ tức 75%.

5.3. Kết quả thực nghiệm và đánh giá

5.3.1. Kết quả thực nghiệm detection QR code

Bảng dưới đây mô tả kết quả thực nghiệm với các mô hình detection cho QR code.

Model	IoU	Precision	Recall	mAP (%)	FPS
YOLOv3	0.5	0.98	0.99	99.56	5
	0.75	0.86	0.86	81.04	
YOLOv4	0.5	0.95	1.00	99.50	4

	0.75	0.90	0.95	93.53	
YOLOv4-tiny	0.5	0.90	0.99	97.75	27
	0.75	0.82	0.90	81.54	
YOLOv5s	0.5	0.98	0.97	98.70	13
	0.75	0.93	0.75	81.70	

Bảng 5.1: Kết quả thực nghiệm các mô hình detection

Như vậy mô hình đạt độ chính xác cao nhất ở ngưỡng IoU = 0.5 là mô hình YOLOv3 và ở ngưỡng IoU = 0.75 là mô hình YOLOv4. Về tốc độ khi sử dụng thì mô hình YOLOv4-tiny cho kết quả FPS cao nhất với hơn 27 (khung hình/giây). Mô hình YOLOv4 cho kết quả FPS thấp nhất với chỉ 4 (khung hình/giây).

5.3.2. Kết quả thực nghiệm decode QR code

Đối với việc thực nghiệm decode QR code, chúng tôi tạo bộ dữ liệu test riêng dành cho việc này với 100 ảnh test được gán nhãn số lượng QR code có thể decode được của ảnh đó. Sau khi thực hiện thực nghiệm chúng tôi có bảng kết quả như dưới đây.

	YOLOv3	YOLOv4	YOLOv4-tiny	YOLOv5s
Pyzbar	71.70	77.36	73.58	50.94
QRCodeDetector	30.19	33.96	39.62	20.75

Bảng 5.2: Kết quả thực nghiệm framework decode pyzbar và opencv-python

Từ bảng trên ta có thể nhận thấy độ chính xác của framework decode pyzbar vượt trội hơn nhiều so với framework decode của QRCodeDetector. Và kết quả đạt được cao nhất với accuray là 77.36% với mô hình detection YOLOv4 và decode là framework pyzbar.

5.4. Phân tích lỗi

Việc nhận dạng QR code với mô hình YOLOv4 không gặp quá nhiều khó khăn, vì số lớp chỉ là một nên việc nhận dạng nhầm lẫn với các lớp khác là không thể xảy ra. Việc gây khó khăn cho decode QR code chính là bounding box sau khi nhận dạng của QR code. Bounding box sẽ được chia làm 3 trường hợp.

- Trường hợp 1: Bounding box bao ngoài QR code.
- Trường hợp 2: Bounding box vừa khít QR code.
- Trường hợp 3: Bounding box nằm trong QR code.

Đối với trường hợp 1: Framework pyzbar có khả năng cao giải mã được QR code.

Đối với trường hợp 2: Cả 2 framework đều giải mã tốt trong trường hợp này.

Đối với trường hợp 3: Gần như cả 2 framework đều không giải mã được QR code vì thiếu thông tin của QR code để tiến hành giải mã.

Như vậy lý do chính dẫn đến việc chênh lệch giữa độ chính xác của framework pyzbar và QRCodeDetector là trường hợp 1.

Và lý do tại sao mà có những QR code đã nhận dạng được nhưng không thể giải mã được là nằm ở trường hợp 3.



Hình 5.2: Các trường hợp nhận dạng 1, 2, 3 theo thứ tự từ trái sang phải
Ngoài ra việc giải mã QR code cũng phụ thuộc vào chất lượng ảnh đầu vào, nếu chất lượng ảnh quá kém thì sẽ dẫn đến việc dù nhận diện được QR code một cách hoàn hảo nhưng vẫn không thể giải mã được chúng. Chất lượng ảnh càng cao thì khả năng

giải mã được QR code càng lớn. Vì khi crop ảnh QR code thì sẽ có thể thấy rõ được các thông tin trong QR code đó, việc decode sẽ tốt hơn.



Hình 5.3: Minh họa về hình ảnh đầu vào có chất lượng thấp

5.5. Kết luận

Như vậy chúng tôi đã hoàn thành việc nghiên cứu và thực nghiệm các phương pháp đã đặt ra. Chúng tôi đã có những thành quả nhất định trong việc thực hiện các phương pháp này. Tiếp đến chúng tôi sẽ xây dựng ứng dụng scan QR code bằng các phương pháp đã thực nghiệm ở trên.

Chương 6. XÂY DỰNG ỨNG DỤNG

6.1. Giới thiệu

Chúng tôi xây dựng một ứng dụng scan QR code áp dụng phương pháp 2 bước:

- Bước 1: Nhận dạng vùng có QR code và vẽ khung giới hạn
- Bước 2: Giải mã QR code trong từng khung giới hạn

Đối với bước 1: Chúng tôi sử dụng mô hình object detection YOLO các phiên bản version 3, version 4 và version 5. Các mô hình được convert và đưa lên tensorflow serving. Tạo thành server detection.

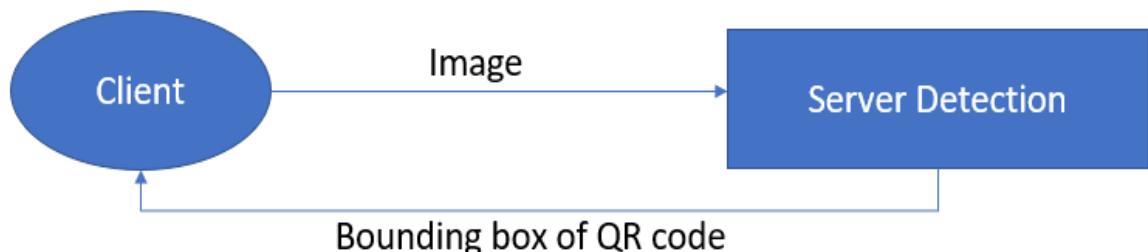
Đối với bước 2: Vì độ chính xác trong việc giải mã của thư viện Pyzbar có độ chính xác vượt trội hơn nên chúng tôi sẽ tạo server decode với thư viện Pyzbar.

Và cuối cùng chúng tôi tạo một server xử lý để có thể xử lý hình ảnh, gửi các yêu cầu đến server detection và server decode để giải mã QR code.

Chúng tôi cũng có xây dựng một ứng dụng scan QR code nhỏ với app android bằng cách convert các mô hình sang Tensorflow lite. Tuy nhiên vì chúng tôi không có kiến thức với ngôn ngữ cho lập trình android nên app được chưa được hoàn thiện. Nên chúng tôi sẽ chỉ đề cập về ứng dụng được xây dựng trên web ở các phần tiếp theo.

Dưới đây là hệ thống scan QR code của chúng tôi:

Bước 1: Detection QR code



Hình 6.1: Qui trình nhận dạng QR code của hệ thống

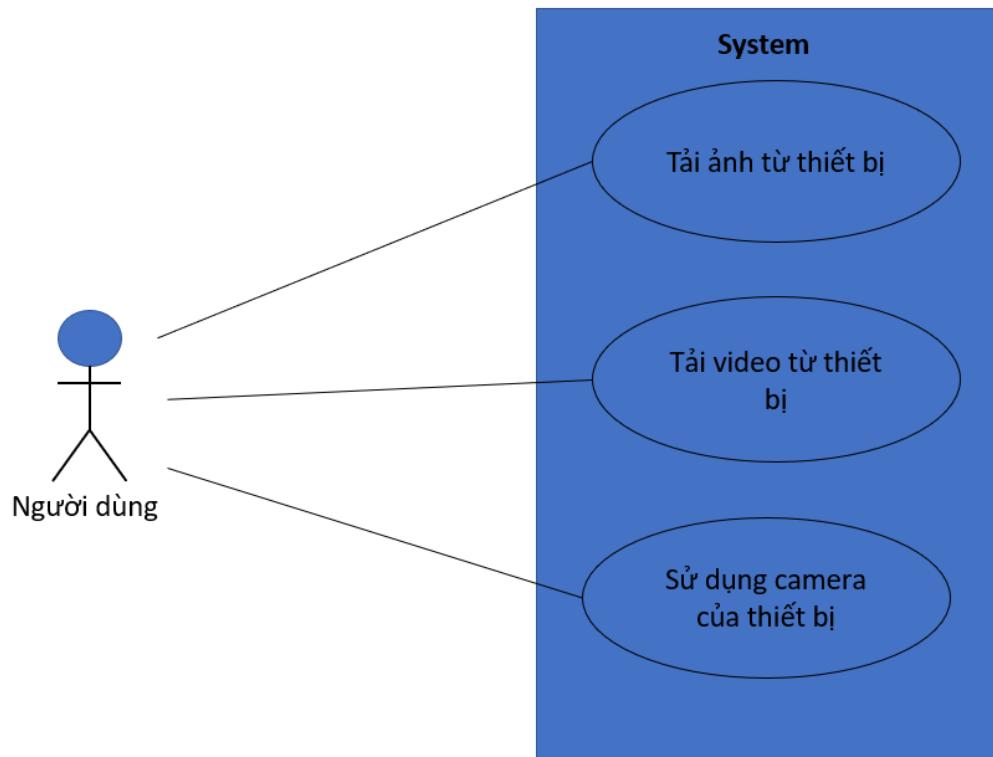
Bước 2: Decode QR code



Hình 6.2: Qui trình giải mã QR code của hệ thống

Tùy 2 quy trình trên client sẽ được trả về kết quả của bounding box và nội dung của mỗi QR code trong hình ảnh được gửi đi.

6.2. Sơ đồ Use Case



Hình 6.3: Sơ đồ Use Case của ứng dụng.

6.2.1. Danh sách Actor

STT	Tên Actor	Ý nghĩa/Ghi chú
1	Người dùng	Là người tham gia quá trình sử dụng hệ thống

Bảng 6.1: Bảng danh sách Actor

6.2.2. Danh sách UseCase

STT	Tên UseCase	Ý nghĩa/Ghi chú
1	Chọn ảnh từ thiết bị	Người dùng chọn hình ảnh từ file của thiết bị lên, hệ thống xử lý hình ảnh và trả về kết quả cho người dùng.
2	Chọn video từ thiết bị	Người dùng chọn file video từ thiết bị lên, hệ thống xử lý hình ảnh và trả về kết quả cho người dùng.
3	Sử dụng camera của thiết bị	Người dùng sẽ cho phép ứng dụng truy cập camera và hệ thống sẽ xử lý hình ảnh liên tục từ camera và xuất ra kết quả cho người dùng theo thời gian thực

Bảng 6.2: Bảng danh sách UseCase

6.2.3. Đặc tả UseCase

6.2.3.1. Đặc tả UseCase “Chọn ảnh từ thiết bị”

- **Tóm tắt:** Người dùng sẽ chọn và tải ảnh từ thư mục lên, hệ thống sẽ nhận dạng và giải mã QR code và hiển thị nội dung của QR code lên cho người dùng.
- **Dòng sự kiện:**
 - Dòng sự kiện chính:
 - * Người dùng chọn ra hình ảnh từ thư mục dưới thiết bị
 - * Hình ảnh sẽ được tự động tải lên sever.

* Hệ thống nhận dạng và giải mã QR code và hiển thị lên cho người dùng

- Dòng sự kiện khác:

* Hình ảnh được tải lên bị lỗi, hệ thống báo lỗi và yêu cầu chọn hình ảnh khác.

- **Trạng thái hệ thống khi bắt đầu thực hiện Use-case:**

- Hiển thị giao diện website.

- **Trạng thái hệ thống sau khi thực hiện Use-case:**

- Giao diện hiển thị hình ảnh được tải lên và được vẽ bounding box trên QR code kèm theo là nội dung sau khi giải mã của QR code. Nếu QR code là website, sẽ hiển thị thêm title của website.

6.2.3.2. Đặc tả UseCase “Chọn video từ thiết bị”

- **Tóm tắt:** Người dùng sẽ chọn và tải video từ thư mục lên, hệ thống sẽ nhận dạng và giải mã QR code và hiển thị nội dung của QR code lên cho người dùng.

- **Dòng sự kiện:**

- Dòng sự kiện chính:

* Người dùng chọn ra video từ thư mục dưới thiết bị

* Video sẽ được tự động tải lên sever.

* Hệ thống nhận dạng và giải mã QR code và hiển thị lên cho người dùng

- Dòng sự kiện khác:

* Video được tải lên bị lỗi, hệ thống báo lỗi và yêu cầu chọn video khác.

- **Trạng thái hệ thống khi bắt đầu thực hiện Use-case:**

- Hiển thị giao diện website.

- **Trạng thái hệ thống sau khi thực hiện Use-case:**

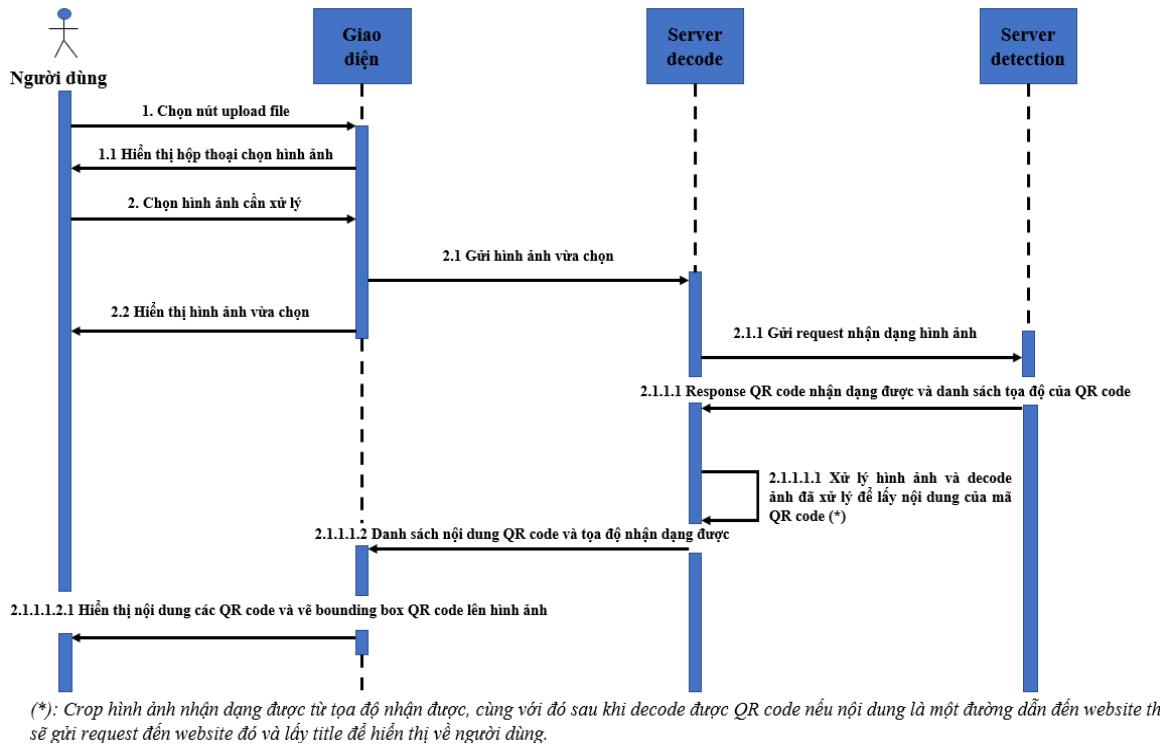
- Giao diện hiển thị video được tải lên và được vẽ bouding box trên QR code kèm theo là nội dung sau khi giải mã của QR code. Nếu QR code là website, sẽ hiển thị thêm title của website.

6.2.3.3. Đặc tả UseCase “Sử dụng camera của thiết bị”

- **Tóm tắt:** Người dùng sẽ sử dụng camera của thiết bị đang sử dụng. Hình ảnh từ camera sẽ được tải lên và hệ thống sẽ nhận dạng và giải mã QR code xuất hiện trong khung hình và hiển thị nội dung của QR code lên cho người dùng.
- **Dòng sự kiện:**
 - Dòng sự kiện chính:
 - * Người dùng chọn sử dụng camera của thiết bị
 - * Hình ảnh từ camera sẽ được tự động tải lên sever.
 - * Hệ thống nhận dạng và giải mã QR code và hiển thị lên cho người dùng
 - Dòng sự kiện khác:
 - * Ứng dụng không truy cập được camera, yêu cầu người dùng chọn nguồn camera khác.
- **Trạng thái hệ thống khi bắt đầu thực hiện Use-case:**
 - Hiển thị giao diện website.
- **Trạng thái hệ thống sau khi thực hiện Use-case:**
 - Giao diện hiển thị hình ảnh từ camera và được vẽ bouding box trên QR code kèm theo là nội dung sau khi giải mã của QR code theo thời gian thực. Nếu QR code là website, sẽ hiển thị thêm title của website.

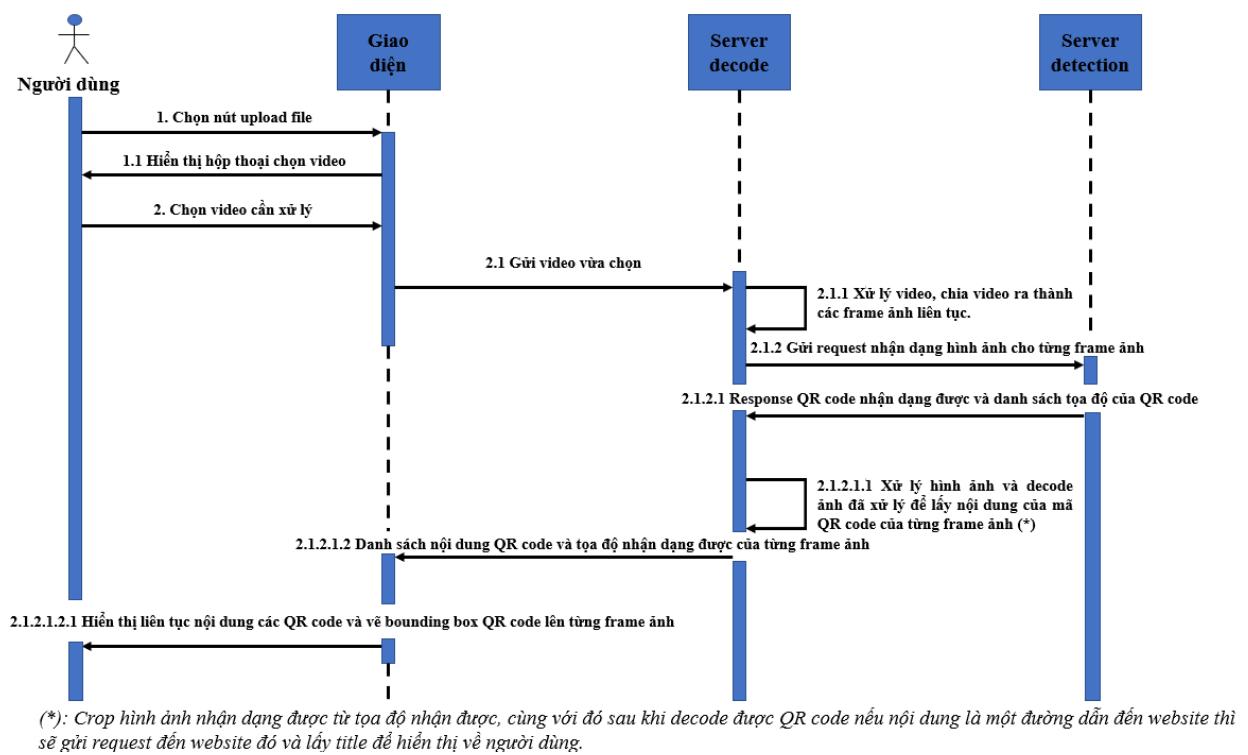
6.3. Sơ đồ tuần tự

6.3.1. Chọn hình ảnh từ thiết bị



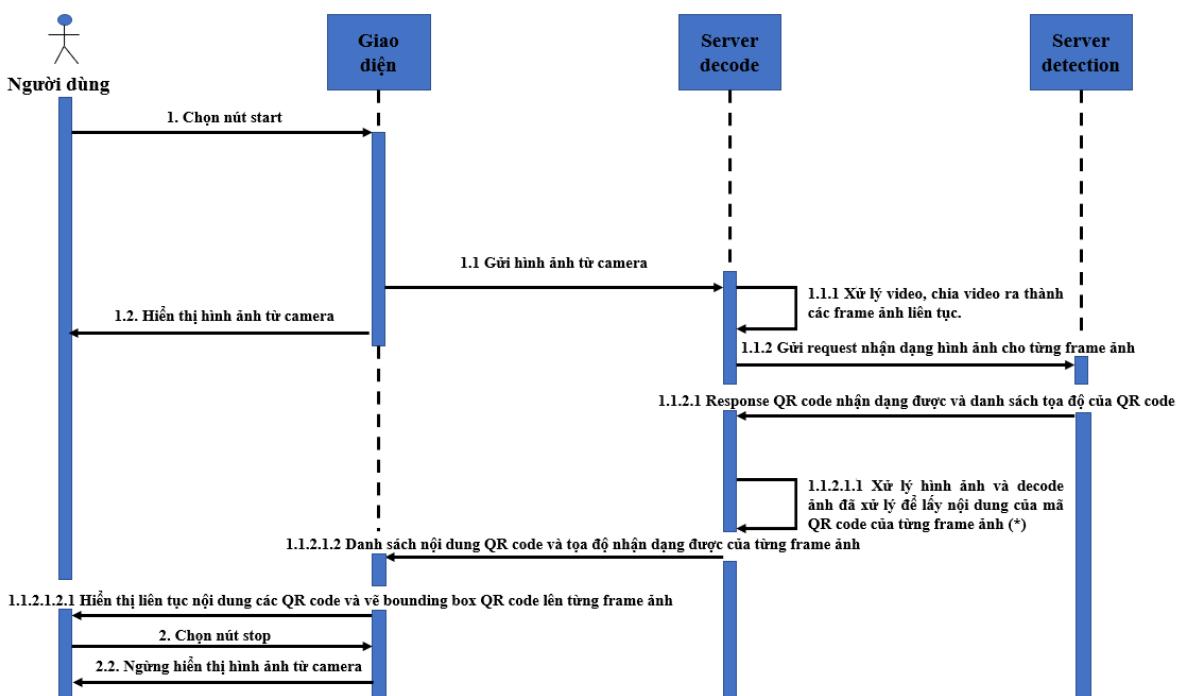
Hình 6.4: Sơ đồ tuần tự chức năng “Chọn hình ảnh từ thiết bị”

6.3.2. Chọn video từ thiết bị



Hình 6.5: Sơ đồ tuần tự chức năng “Chọn video từ thiết bị”

6.3.3. Sử dụng camera của thiết bị

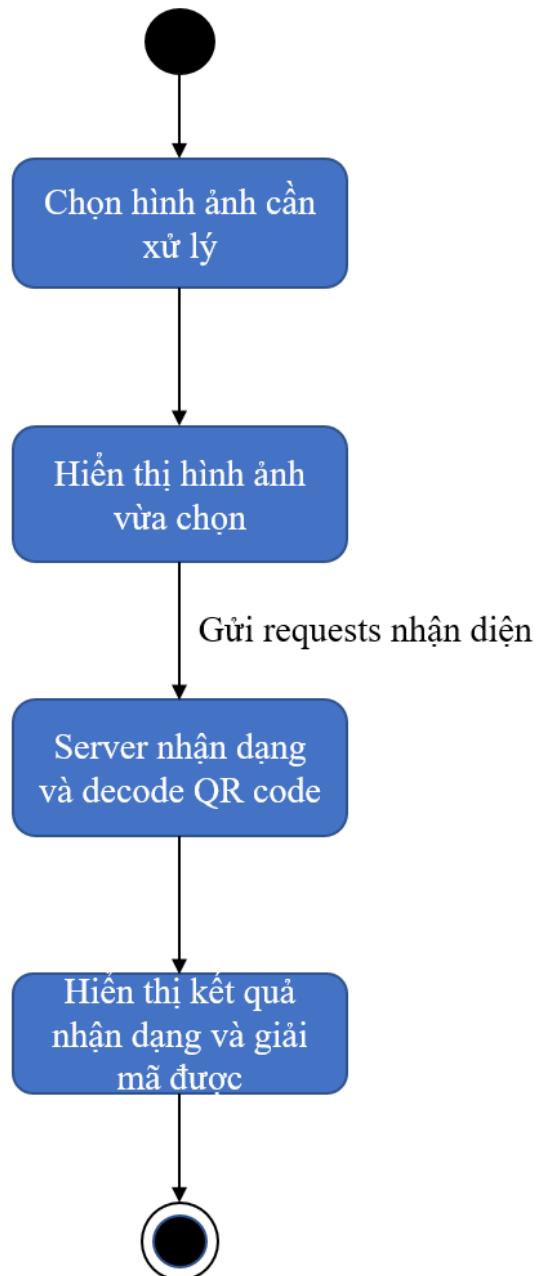


Hình 6.6: Sơ đồ tuần tự chức năng “Sử dụng camera của thiết bị”

6.4. Mô hình hóa ứng xử

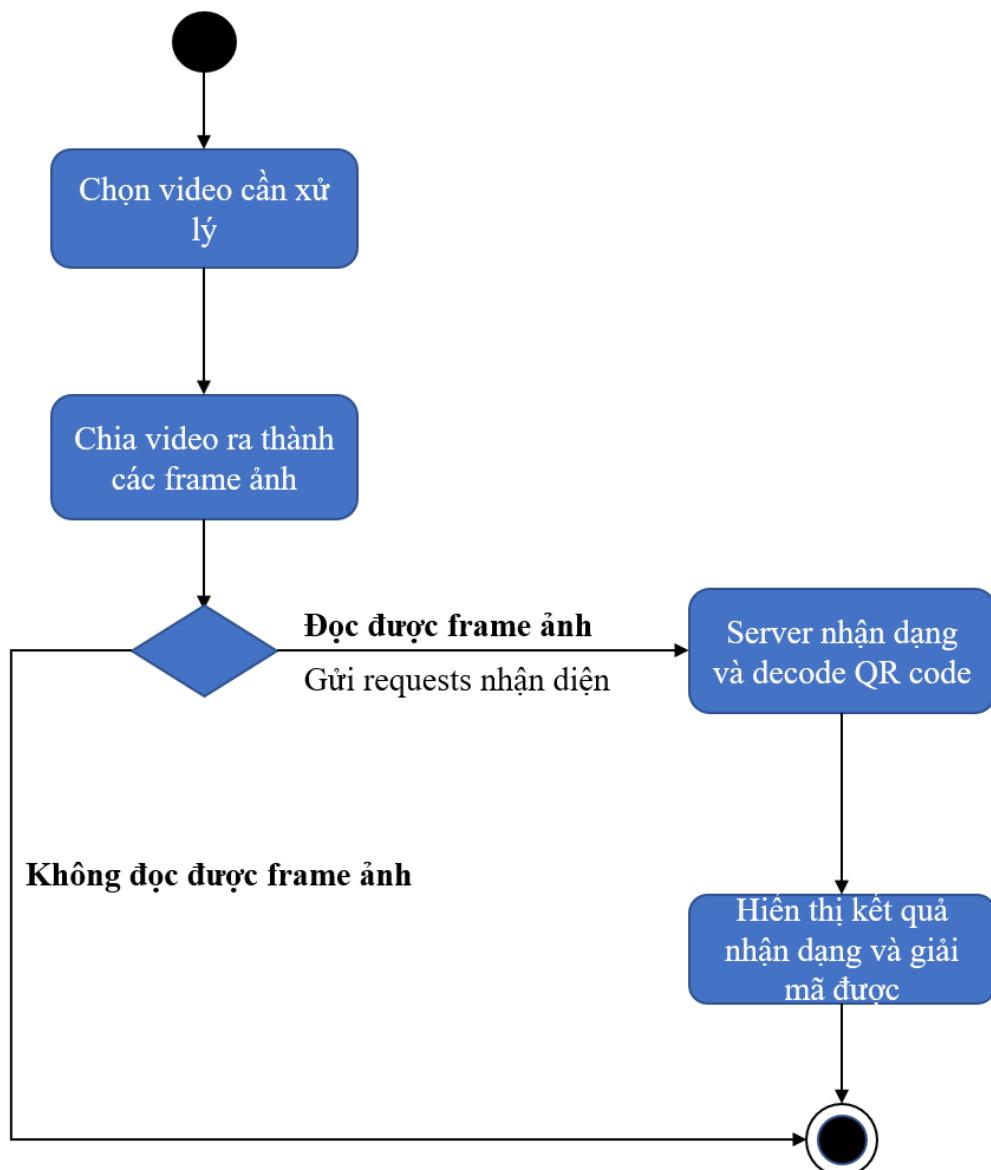
6.4.1. Sơ đồ trạng thái

6.4.1.1. Chọn ảnh từ thiết bị



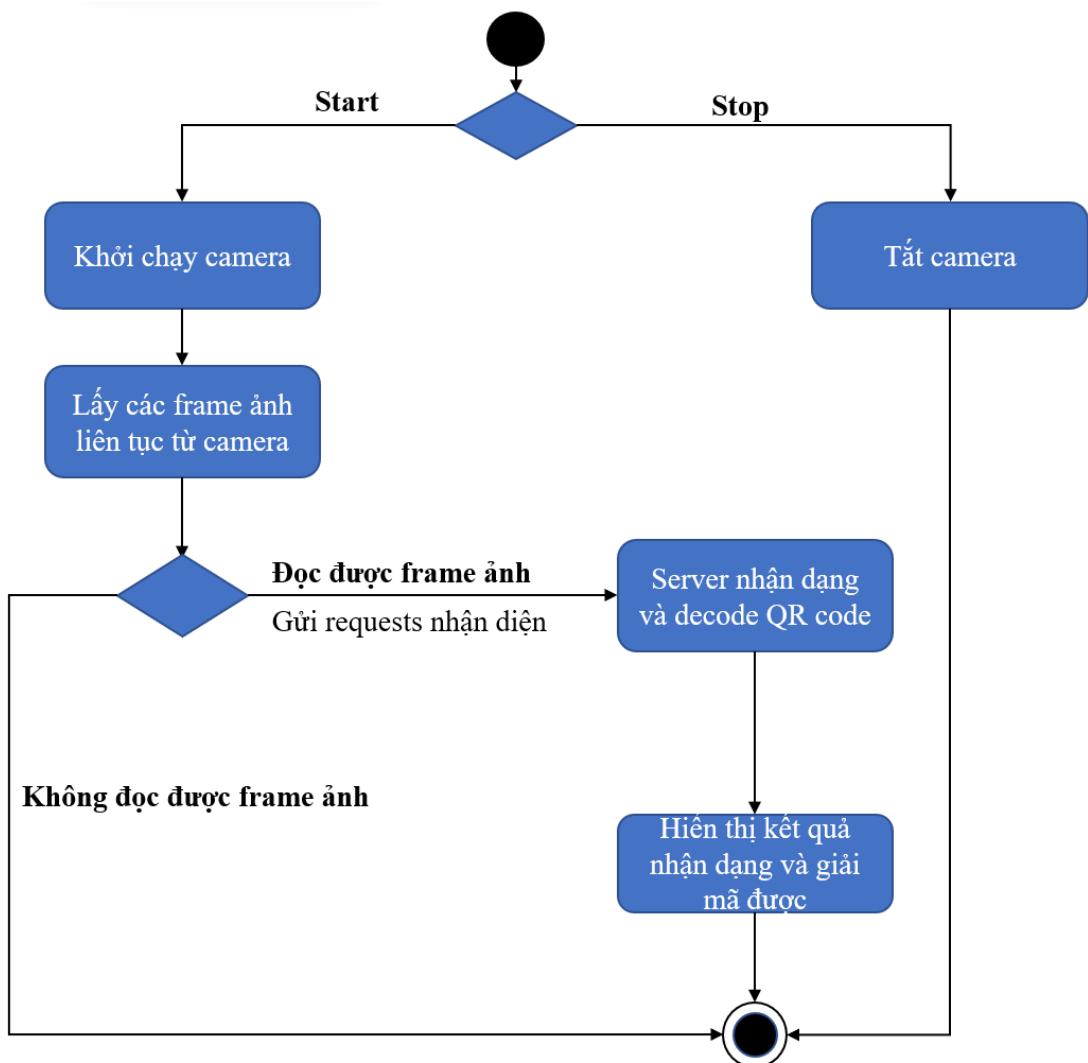
Hình 6.7: Sơ đồ trạng thái chức năng “Chọn ảnh từ thiết bị”

6.4.1.2. Chọn video từ thiết bị



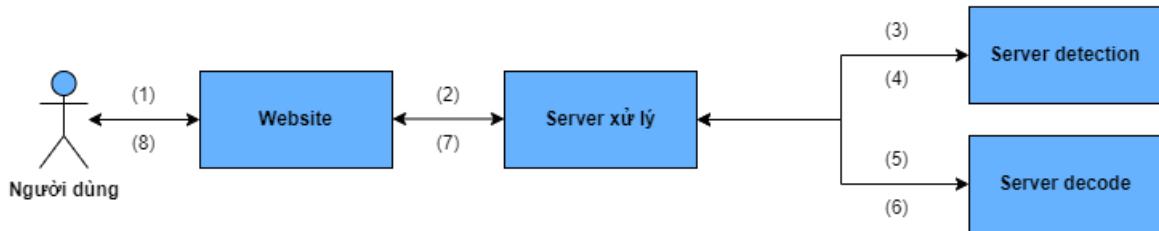
Hình 6.8: Sơ đồ trạng thái chức năng “Chọn video từ thiết bị”

6.4.1.3. Sử dụng camera của thiết bị



Hình 6.9: Sơ đồ trạng thái chức năng “Sử dụng camera của thiết bị”

6.4.2. Luồng xử lý dữ liệu



Hình 6.10: Mô hình luồng dữ liệu của hệ thống trên website

Với ứng dụng của đề tài, luồng dữ liệu sẽ được xử lý như sau:

- (1): Đầu tiên người dùng có thể chọn upload một hình ảnh (hoặc một video và có thể mở trực tiếp camera của thiết bị) để tải lên server.
- (2): Website gửi thông tin người dùng vừa tải lên cho server xử lý đồng thời nếu là hình ảnh được tải lên thì sẽ hiển thị hình ảnh vừa được tải lên trong trạng thái chờ.
- (3): Tại server xử lý: hình ảnh, video được tải lên hoặc camera đều sẽ được tiền xử lý và đưa về thành các frame ảnh. Sau đó sẽ gửi request yêu cầu server detection nhận dạng các QR code xuất hiện trong frame ảnh.
- (4): Tại server detection: sẽ nhận dạng QR code từ hình ảnh được yêu cầu và trả về lại tọa độ các điểm bounding box của QR code cho server xử lý.
- (5): Tại server xử lý: khi nhận được tọa độ các điểm của bounding box của QR code sẽ tiến hành xử lý, crop hình ảnh theo tọa độ nhận được. Sau đó sẽ gửi request yêu cầu server decode giải mã QR code của hình được crop.
- (6): Tại server decode: sẽ tiến hành giải mã QR code từ hình ảnh được yêu cầu và sẽ trả về nội dung của QR code đó cho server xử lý.
- (7): Tại server xử lý: Khi nhận được nội dung giải mã của QR code, nếu nội dung là một đường dẫn đến website thì sẽ request đường dẫn và lấy title của nội dung trang web đó. Và server xử lý sẽ gửi các kết quả: tọa độ bounding box, nội dung QR code (title đường dẫn website nếu có) cho website.

- (8): Khi website nhận được kết quả từ server xử lý, website sẽ hiển thị kết quả nhận được cho người dùng theo từng khung bounding box của các QR code nhận diện được và nội dung QR code ở phía dưới (title website nếu nó là đường dẫn).

6.5. Qui trình xây dựng

6.5.1. Server detection

Đầu tiên, chúng tôi xây dựng hệ thống nhận dạng QR code bằng tensorflow serving.

Bắt đầu, chúng tôi convert tất cả các file weight của các mô hình YOLO sang format của tensorflow model. Đối với mô hình YOLOv3, YOLOv4 và YOLOv4-tiny chúng tôi sử dụng source <https://github.com/hunglc007/tensorflow-yolov4-tflite> của tác giả Việt Hùng. Đối với mô hình YOLOv5 vì chúng tôi huấn luyện trên pytorch nên đã có sẵn source export.py để convert sang saved_model của tensorflow.

Tiếp theo, sau khi có model chúng tôi tiến hành tạo docker image cho server. Cấu trúc các thư mục sẽ được sắp xếp như hình dưới đây:

```
Server:.
+---models
    +---model1/1
        |   save_model.pb
        +---assets
            |       save_model.json
        +---variables
            |       checkpoint
            |       variable.data-00000-of00002
            |       variable.data-00001-of00002
            |       variable.index
    +---model2/1
        |   save_model.pb
        +---assets
            |       save_model.json
        +---variables
            |       checkpoint
            |       variable.data-00000-of00002
            |       variable.data-00001-of00002
            |       variable.index
    +---model3/1
        |   save_model.pb
        +---assets
            |       save_model.json
        +---variables
            |       checkpoint
            |       variable.data-00000-of00002
            |       variable.data-00001-of00002
            |       variable.index
    +---model4/1
        |   save_model.pb
        +---assets
            |       save_model.json
        +---variables
            |       checkpoint
            |       variable.data-00000-of00002
            |       variable.data-00001-of00002
            |       variable.index
models.config
docker-compose.yaml
dockerfile
```

Hình 6.11: Sơ đồ cây các thư mục và file xây dựng tensorflow serving

Trong đó các model1 là mô hình YOLOv4-tiny, model2 là YOLOv5, model3 là YOLOv3 và model4 là YOLOv4.

Ở file models.config là nơi cấu hình tên và vị trí các mô hình trên server, file docker-compose dùng để run docker và mở port truy cập đến server.

Tiếp đến chúng tôi bắt đầu build docker bằng docker file.

Cuối cùng chạy docker-compose để khởi động server.

6.5.2. Server decode

Đối với server decode chúng tôi sử dụng thư viện là pyzbar (Zbar) vì có độ chính xác thực nghiệm cao hơn so với thư viện QRCodeDetector của opencv-python. Vì vậy nên chúng tôi sẽ tạo một function ở server này để gọi thư viện với đối số là hình ảnh đầu vào được yêu cầu. Và đầu ra sẽ là nội dung giải mã của QR code.

6.5.3. Server xử lý

Server xử lý chúng tôi sử dụng thư viện streamlit, ở server chúng tôi sẽ có nhiều hàm gồm các mục đích xử lý ảnh và gửi các yêu cầu đến server detection và server decode.

Đầu tiên là các hàm xử lý ảnh:

- Hàm **preprocessing_image_input**: Xử lý hình ảnh đầu vào phù hợp với mô hình nhận dạng.
- Hàm **convert_bbox_yolov3_and_yolov4**: Xử lý đầu ra của mô YOLOv3 và YOLOv4 hình thành các điểm tọa độ bounding box phù hợp.
- Hàm **convert_bbox_yolov5**: Xử lý đầu ra của mô hình YOLOv5 thành các điểm tọa độ bounding box phù hợp.

Tiếp đến là các hàm yêu cầu:

- Hàm **scan_image**: Tương ứng với chọn hình ảnh từ thiết bị. Với đối số đầu vào là hình ảnh được tải lên, hàm sẽ trả lại kết quả gồm 2 ảnh và nội dung: 2 ảnh gồm 1 ảnh gốc và 1 ảnh được đánh bounding box QR code. Nội dung là nội dung QR code giải mã được (title website nếu nội dung là một đường dẫn), nếu không có QR code hoặc không giải mã sẽ trả lại kết quả “Unable to decode QR code”

- Hàm **scan_video**: Tương ứng với chọn video từ thiết bị. Với đối số đầu vào là một video được tải lên, hàm sẽ cắt video ra thành các frame ảnh và xử lý. Và đầu ra sẽ gồm video với các frame ảnh được vẽ liên tục các bounding box QR code và nội dung QR code (title website nếu nội dung là một đường dẫn) xuất hiện trên khung hình theo thời gian thực. Khi kết thúc video sẽ thống kê lại tất cả QR code đã giải mã được.
- Hàm **scan_camera**: Tương ứng với sử dụng camera của thiết bị. Hàm này sẽ yêu cầu bật camera từ thiết bị và cắt các frame ảnh từ camera. Các frame ảnh sẽ được xử lý và được xuất ra liên tục theo thời gian thực. Đầu ra tương ứng sẽ là các frame ảnh được vẽ các bounding box lên QR code và nội dung của QR code ngay phía dưới ô nội dung.

6.5.4. gRPC cho server detection

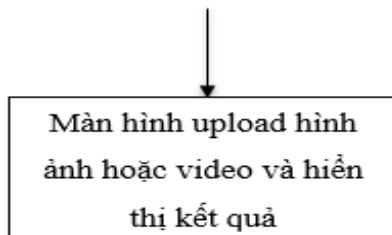
gRPC là sự kết hợp của Protocol Buffers và http2, Protocol Buffers được phát triển bởi google nó nhẹ hơn, nhanh hơn và cung cấp hiệu năng tốt hơn so với sử dụng XML hoặc Json gRPC cũng cho phép định nghĩa cấu trúc của data dưới dạng file protoc và nó tự động generate ra file sử dụng để giao tiếp với ngôn ngữ python.

Việc sử dụng gRPC để thực hiện giao tiếp giữa client và server trong microservices sẽ mang lại tốc độ xử lý nhanh chóng hơn nhiều so với việc sử dụng API thông thường.

Kết quả sẽ được qua server xử lý để chuẩn hóa về chuẩn bounding box phù hợp và tiến hành crop image theo bounding box và gửi yêu cầu decode hình ảnh.

6.5.5. Giao diện

6.5.5.1. Sơ đồ luồng màn hình



Hình 6.12: Sơ đồ luồng màn hình của ứng dụng Scan QR code.

Danh sách màn hình:

STT	Tên màn hình	Ý nghĩa/Ghi chú
1	Màn hình upload hình ảnh hoặc video và hiển thị kết quả	Cho phép người dùng upload hình ảnh hoặc video từ thiết bị để nhận diện và giải mã nội dung QR code.

Bảng 6.3: Danh sách màn hình

6.5.5.2. Mô tả chi tiết màn hình

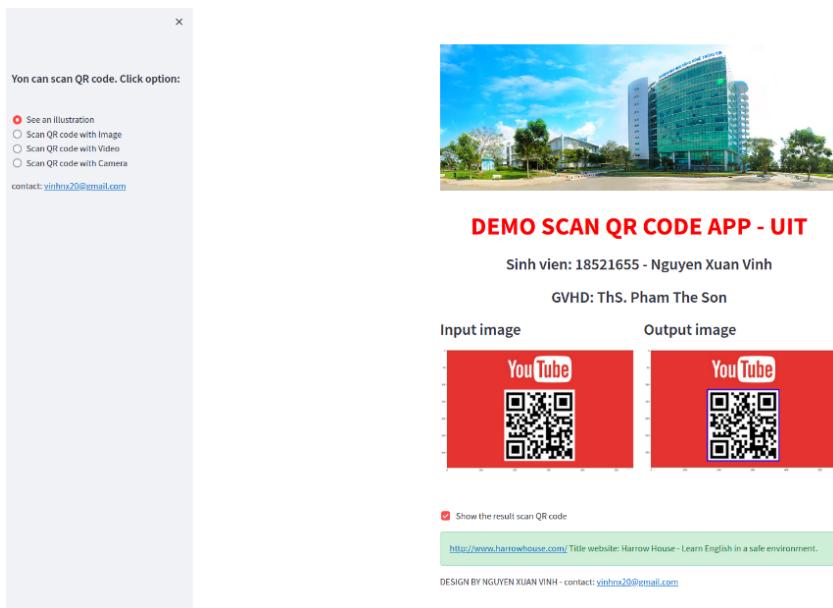
Chúng tôi sẽ có 3 option cho người dùng lựa chọn:

- Scan QR code with image (Chọn hình ảnh từ thiết bị)
- Scan QR code with video (Chọn video từ thiết bị)
- Scan QR code with camera (Sử dụng camera của thiết bị)

Ngoài ra khi người dùng vừa truy cập đến trang web sẽ xuất hiện với lựa chọn: See an illustration để người dùng có thể hình dung được kết quả khi mình sử dụng ứng dụng này.

Dưới đây là hình ảnh và thông số của từng website tương ứng với các lựa chọn:

- See an illustration:

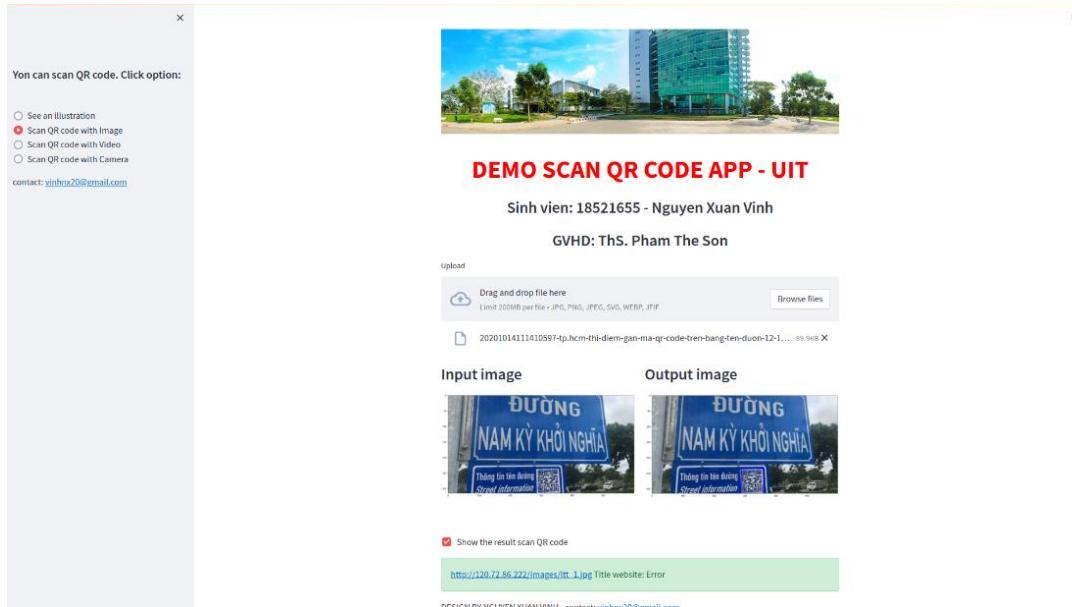


Hình 6.13: Giao diện khi truy cập đến trang web

Tên	Kiểu	Mô tả
Click option	Choice	Các lựa chọn dành cho người dùng.
Input image	Image	Vùng hiển thị hình ảnh đầu vào
Output image	Image	Vùng hiển thị hình ảnh đầu ra. Được vẽ bounding box lên các QR code
Show the result scan QR code	Check box	Hiển thị kết quả giải mã của QR code.

Bảng 6.4: Thông số giao diện website với lựa chọn “See an illustration”

- Scan QR code with Image:

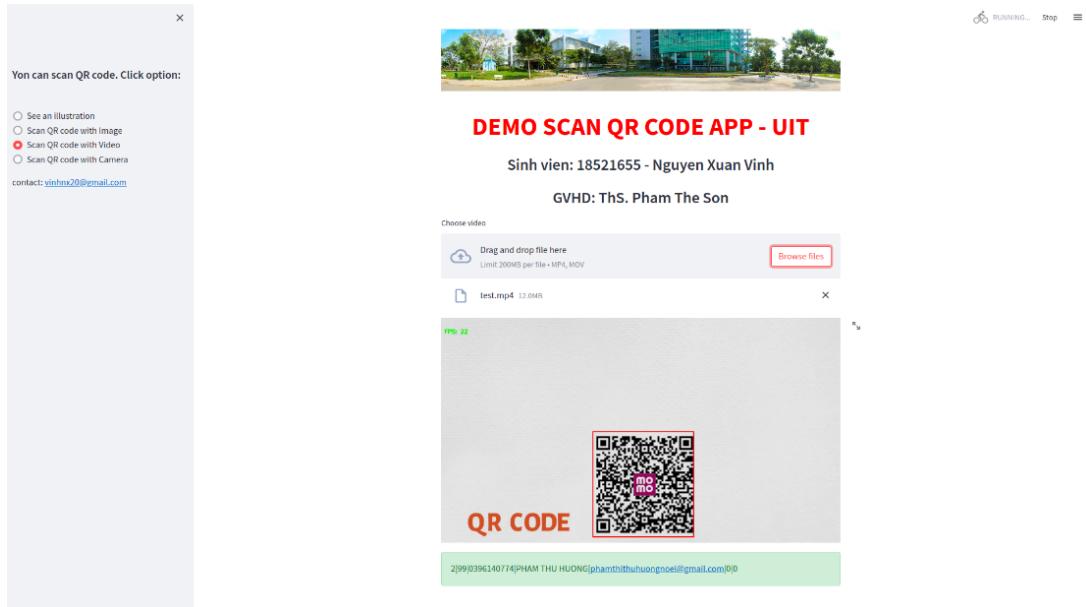


Hình 6.14: Giao diện với option Scan QR code with Image

Tên	Kiểu	Mô tả
Click option	Choice	Các lựa chọn dành cho người dùng.
Upload	InputFile	Vùng chọn hình ảnh từ thư mục thiết bị.
Input image	Image	Vùng hiển thị hình ảnh đầu vào
Output image	Image	Vùng hiển thị hình ảnh đầu ra. Được vẽ bounding box lên các QR code
Show the result scan QR code	Check box	Hiển thị kết quả giải mã của QR code.

Bảng 6.5: Thông số giao diện website với lựa chọn “Scan QR code with Image”

- Scan QR code with Video:

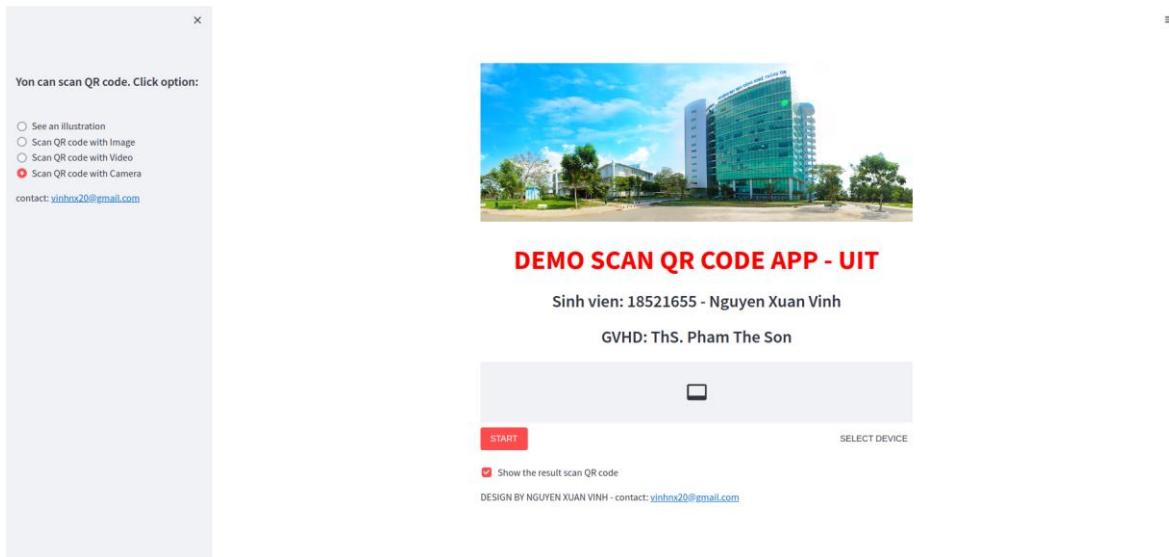


Hình 6.15: Giao diện với option Scan QR code with Video

Tên	Kiểu	Mô tả
Click option	Choice	Các lựa chọn dành cho người dùng.
Choose video	InputFile	Vùng chọn video từ thư mục thiết bị.
Video	Video	Vùng hiển thị video đầu ra
Content	Text	Hiển thị kết quả giải mã của QR code.

Bảng 6.6: Thông số giao diện website với lựa chọn “Scan QR code with Video”

- Scan QR code with Camera:



Hình 6.16: Giao diện với option Scan QR code with Camera

Tên	Kiểu	Mô tả
Click option	Choice	Các lựa chọn dành cho người dùng.
Start/Stop	Button	Khởi chạy/Tạm dừng camera
SELECT DEVICE	Choice	Lựa chọn thiết bị camera
Video	Video	Vùng hiển thị video của camera đầu ra
Show the result scan QR code	Check box	Hiển thị kết quả giải mã của QR code.

Bảng 6.7: Thông số giao diện website với lựa chọn “Scan QR code with Video”

6.6. Kết quả

Chúng tôi đã xây dựng được hệ thống ứng dụng scan QR code cho người dùng với các lựa chọn trong các trường hợp khác nhau, ứng dụng đáp ứng được nhu cầu cơ bản của người dùng. Áp dụng được các phương pháp đã nghiên cứu và chạy thực nghiệm ở các bước trên.

Scan QR code with image chúng tôi sử dụng mô hình tốt nhất là YOLOv4 kết hợp với Pyzbar.

Scan QR code with video và scan QR code with camera chúng tôi sử dụng mô hình có tốc độ FPS gần với thời gian thực nhất và có kết quả cao nhất trong 2 mô hình có FPS real-time là YOLOv4-tiny kết hợp với Pyzbar.

6.7. Kết luận

Như vậy chúng tôi đã sơ bộ hoàn thành sản phẩm ứng dụng scan QR code áp dụng mô hình học sâu, giải mã các trường hợp QR code được nêu ra từ ban đầu.

Chương 7. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1. Kết luận

Trong khóa luận tốt nghiệp này chúng tôi tiến hành xây dựng bộ dữ liệu QR code, tiến hành nghiên cứu và cài đặt các mô hình object detection cho bài toán nhận dạng QR code, sử dụng các mô hình đã đào tạo được để xây dựng một ứng dụng scan QR code. Chúng tôi đã đạt được một số kết quả nhất định như sau. Hoàn thành thu thập và gán nhãn bộ dữ liệu gồm 8.734 ảnh QR code. Tiến hành cài đặt các mô hình object detection gồm: YOLOv3, YOLOv4, YOLOv4-tiny và YOLOv5s. Và thu được kết quả tốt nhất cho IoU = 0.5 là **YOLOv3** với mAP = **99.56%** và tốt nhất cho IoU = 0.75 là **YOLOv4** với mAP = **93.53%**. Về chỉ số FPS (tốc độ khung hình/giây) thì mô hình YOLOv4-tiny và YOLOv5s đưa ra tốc độ cao hơn hẳn mô hình YOLOv3 và YOLOv4 tuy nhiên độ chính xác bị giảm đi đáng kể. Chỉ số FPS cao nhất là chạy trên mô hình **YOLOv4-tiny** với **27 FPS**. Cùng với đó là kết quả giải mã QR code đạt kết quả tốt nhất khi sử dụng mô hình detection YOLOVv4 và framework giải mã **pypybar** với accuracy = **77.36%**. Và chúng tôi đã xây dựng được một ứng dụng scan QR code bằng cách sử dụng các mô hình và framework đã sử dụng ở trên, ứng dụng có thể chạy real-time với mô hình YOLOV4-tiny và YOLOv5s với FPS trung bình là 27 FPS và 13 FPS.

7.2. Hạn chế

Bên cạnh những kết quả đạt được ở trên của chúng tôi thì cũng tồn tại những hạn chế trong khóa luận này. Ứng dụng chưa được hoàn thiện đầy đủ, còn ít sự lựa chọn đối với người dùng. Giao diện người dùng chưa thật sự tốt còn cần nhiều cải thiện. Cùng với đó là tìm kiếm thêm các phương pháp để cải thiện độ chính xác trong việc decode QR code.

7.3. Hướng phát triển

Về hướng phát triển trong tương lai, chúng tôi đưa ra một số ý kiến như sau:

- Đổi với giao diện người dùng của ứng dụng cần tham khảo ý kiến người dùng và xây dựng lại một cách hợp lý.
- Chức năng cần tăng thêm, đa dạng nhiều lựa chọn cho người dùng.
- Tối ưu hóa tốc độ xử lý của hệ thống bằng các thuật toán tối ưu hóa.
- Xây dựng ứng dụng trên thiết bị di động, tăng cường sự thuận tiện cho người dùng. Thay vì chỉ được sử dụng ứng dụng trên web.

DANH MỤC CÔNG TRÌNH TÁC GIẢ

Với những nghiên cứu đến bài toán giải mã QR code cho những trường hợp được nhắc đến trong khóa luận, chúng tôi đã có bài báo gửi tới hội nghị “2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)”

Bài báo “Intelligent Augmented Video Streaming Services Using Lightweight QR Code Scanner” đưa ra giải pháp giải mã QR code cho video trực tuyến tăng cường với mã QR được nhúng vào các video trực tuyến phát liên tục gồm hai giai đoạn. Trong giai đoạn đầu sẽ nhận dạng đối tượng bằng cách sử dụng một thuật toán phát hiện đối tượng theo thời gian thực. Trong giai đoạn thứ hai, vùng phát hiện mã QR ở giai đoạn thứ nhất sẽ được đưa vào bộ đọc mã QR để trích xuất thông tin dữ liệu được nhúng. Kết quả thử nghiệm cho kết quả đạt hiệu suất cao về thời gian phản hồi và hứa hẹn sẽ được triển khai trên điện thoại thông minh.

Xuan-Vinh Nguyen, Gia-Huy Lam, Quang-Nhat Le, Quoc-Loc Duong, The-Manh Nguyen, Bao-Long Le, Quang Dieu Tran, Trong-Hop Do, Nhu-Ngoc Dao. (2021).

Intelligent Augmented Video Streaming Services Using Lightweight QR Code Scanner. Proceedings of 2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT), 103-107.

TÀI LIỆU THAM KHẢO

- [1] Belussi, Luiz, and Nina Hirata. "Fast QR code detection in arbitrarily acquired images." 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images. IEEE, 2011.
- [2] Kurniawan, Wendy Cahya, Hiroshi Okumura, and Anik Nur Handayani. "An improvement on qr code limit angle detection using convolution neural network." 2019 International Conference on Electrical, Electronics and Information Engineering (ICEEIE). Vol. 6. IEEE, 2019.
- [3] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
- [4] Redmon J. Darknet: Open Source Neural Networks in C; 2013–2016. [(accessed on 20 October 2021)]. Available online: <https://pjreddie.com/darknet/>
- [5] Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
- [6] Iandola, Forrest, et al. "Densenet: Implementing efficient convnet descriptor pyramids." arXiv preprint arXiv:1404.1869 (2014).
- [7] Zhang, Zhi, et al. "Bag of freebies for training object detection neural networks." arXiv preprint arXiv:1902.04103 (2019).
- [8] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).
- [9] Zheng, Zhaozhi, et al. "Distance-IoU loss: Faster and better learning for bounding box regression." Proceedings of the AAAI conference on artificial intelligence. Vol. 34. No. 07. 2020.

- [10] Glenn Jocher. Ultralytics: Github. 2021. [(accessed on 20 October 2021)].
YOLOv5 Focus() Layer #3181. Available online:
<https://github.com/ultralytics/yolov5/discussions/3181m1>

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/354519090>

Intelligent Augmented Video Streaming Services Using Lightweight QR Code Scanner

Conference Paper · July 2021

DOI: 10.1109/COMNETSAT53002.2021.9530819

CITATIONS

0

READS

93

9 authors, including:



Xuan-Vinh Nguyen

Ho Chi Minh City University of Information Technology

2 PUBLICATIONS 0 CITATIONS

[SEE PROFILE](#)



Gia-Huy Lam

Ho Chi Minh City University of Information Technology

4 PUBLICATIONS 1 CITATION

[SEE PROFILE](#)



Quang-Nhat Le

Ho Chi Minh City University of Information Technology

2 PUBLICATIONS 1 CITATION

[SEE PROFILE](#)



Quoc-Loc Duong

Ho Chi Minh City University of Information Technology

2 PUBLICATIONS 0 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Vehicle communication using Visible Light Communication [View project](#)



StoryQ: from Quantizing to Understanding Stories [View project](#)

Intelligent Augmented Video Streaming Services Using Lightweight QR Code Scanner

Xuan-Vinh Nguyen

*University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
18521655@gm.uit.edu.vn*

Gia-Huy Lam

*University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
18520832@gm.uit.edu.vn*

Quang-Nhat Le

*University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
18521190@gm.uit.edu.vn*

Quoc-Loc Duong

*University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
18521006@gm.uit.edu.vn*

The-Manh Nguyen

*University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
18521084@gm.uit.edu.vn*

Bao-Long Le

*University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
19521782@gm.uit.edu.vn*

Quang Dieu Tran

*Ho Chi Minh National Academy of
Politics
Hanoi, Vietnam
dieutq@hcma.vn*

Trong-Hop Do

*University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
hopdt@uit.edu.vn*

Nhu-Ngoc Dao

*Dept. of Computer Science and Engineering
Sejong University
Seoul, South Korea
nndao@sejong.ac.kr*

Abstract—Video streaming is a multimedia service that continuously transmits the data over the Internet and presents the content on user screens without predownloading the entire video. Augmented video streaming is an advanced version of video streaming, where the video is enriched with additional embedded information in video frames. These additional data aim to provide a better user experience. Using QR code is one of the efficient approaches to incorporate information into video streams in this context. However, receiving the data in the embedded QR code is considered a challenging task owing to video quality and view angles. This paper proposes a lightweight two-stage QR code decoder for augmented video streaming using deep learning technologies. In the first stage, the position of the embedded QR code is detected using an online object detection algorithm. In the second stage, the detected region of the QR code is fed into a QR code reader to extract the embedded data. The experimental results show that the proposed decoder achieves high performances in terms of response time and decoding accuracy while being very lightweight, which is promising to be implemented in smartphones.

Index Terms—Augmented video streaming, QR code, augmented reality, deep learning

I. INTRODUCTION

Due to the ever-increasing need for video content, videos are required to be played back or played directly at the time of the recording without downloading. To fulfill this need, media streaming has appeared and gradually dominate the entertainment industry [1]–[3]. Augmented video streaming a type of media streaming where streaming videos are embedded with additional data. These embedded data can be used to enhance the user experience and provide more ways for the

service provider to deliver content to end users. For example, links to the website of new events or products can be embedded in the videos which introduce these events and products. These additional data can be embedded through many means including text, icon, logo, gif, audio, and QR code. Among these options, the QR code, which was proposed by Mr. DensoWave in 1994 is especially suitable for augmented video streaming as it can be used to embed large amounts of information without significantly interfering with viewer experiences. Embedded QR codes can appear in videos in two forms: active and passive.

As shown in Fig. 1a, the QR code is actively embedded in the video to provide a link to the website of a music tour. In Fig. 1b, the printed QR code which provides the location appears video scene. Active embedded QR codes can intentionally provide information to viewers in marketing, education, entertainment, healthcare, and many other fields. In the case of Fig. 1a, the viewer can scan the shown QR code to go to the music event website or buy the ticket directly. Passive embedded QR code is not used intentionally but still supplying useful information to viewers. For example, in the case of Fig. 1b, the viewer can scan the show QR code to find the location of the place in the video. To these ends, a type of streaming QR code scanner, which is an application installed in a camera enable smart device must be used. Smartphones, which are widely available nowadays, are the perfect device for installing these applications thanks to high-resolution cameras and powerful processors equipped with them.



(a) Active QR code

(b) Passive QR code

Fig. 1: QR code embedded in videos.

In augmented video streaming, QR codes are continuously embedded in frames of the video to provide a stream of additional information. The constant changes of the QR code appearing in the video will introduce a lot of difficulties for current QR code scanners, which are developed for decoding static QR codes. This study focuses on proposing a new lightweight streaming QR code scanner to obtain embedded information in Augmented Video Streaming. The proposed algorithm is developed to be not only accurate in detecting the QR but also lightweight to guarantee the possibility of implementation on smartphones.

In the proposed algorithm, the process of extracting information from the QR code embedded in the video is achieved in two steps. In the first step, the position of the QR code in the video is determined by applying object detection algorithms such as Yolov4. In the second step, the areas containing the QR code located in the previous step will be cut out from each frame and fed to a QR code reader to decode to retrieve the embedded information.

To develop the above QR code detect program, a collection of augmented videos has been created. QR code is embedded in these videos in two forms: active and passive. In the first form, QR codes are actively embedded in each video frame. In the second form, the QR code appears as a printed image in the captured videos. The augmented videos were then played in various types of displays including monitor and projector screen. A smartphone was used to record the video of the display to create the dataset for the proposed streaming QR code scanner. The performance of the proposed algorithm is evaluated through many aspects including decoding accuracy, detectable distance, and detection rate. The experimental results show that the proposed algorithm has high performance in these aspects. In addition, this algorithm is also very compact, so it can be installed on smartphones.

II. RELATED WORKS

Various methods have been proposed for the barcode recognition problem over the years. One of the first barcode identification papers was that of Muniz *et al.* [4] for a barcode scanning application on prescriptions. This early approach for QR code recognition soon became outdated by later methods.

The appearance of the camera on mobile phones has inspired many articles on barcode identification by mobile phone cameras. In [5], Ohbuchi *et al.* developed a mobile application that can detect both QR and EAN codes. In [6], Wachenfeld *et al.* proposed a method for identifying 1D barcodes, in which decoding is used as a tool to find bar codes. Both Ohbuchi

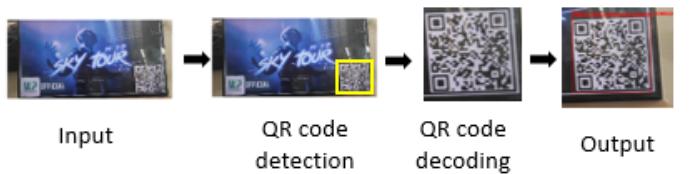


Fig. 2: Procedure of the proposed algorithm.

and Wachenfeld are heavily dependent on the user pointing the camera at the barcode.

In recent papers, the focus has been on offering algorithms for barcode recognition with little reliance on human camera alignment. Some methods that depend on simple geometrical operations such as the paper Katona and Nyul [7] improved from their work in [8]. The improved version adds Euclidean distance to eliminate objects in the distance. This paper is one of the problems related to the recognition of both 1D and 2D barcodes. The data used for testing in this paper are 17,280 composite images and a set of 1000 real-life images with a 1D barcode. This dataset has not been published yet and the author has not tested their algorithm on any of the benchmark datasets yet. However, Sörös *et al.* [9] evaluated Katona's performance plus their own algorithms per 1000 1D images from WWU Muenster Barcode Database [10], [11]. This test shows the Katona algorithm's low score and indicates that though Katona reports high accuracy on their own datasets.

In [12], Creusot *et al.* provided a modern model for 1D barcode detection. The article uses Muenster BarcodeDB and Arte-Lab database to test model performance. Creusot's test results outperformed both Arte-Lab and Muenster BarcodeDB. In [13], the author continued for a follow-up article that improves the results of the previous article by using a method called Parallel Segment Detector (PSD) based on Line Segment Detector (LSD).

III. PROPOSED TWO-STAGE ALGORITHM FOR STREAMING QR CODE DECODER

The proposed two-stage algorithm for streaming QR code decoder is shown in Fig. 2. As suggested by the name, the process of extracting the information embedded in the QR code is performed through two stages. In the first stage, the position of the QR code in the frame is located using an object detection algorithm. In the second stage, the QR code is decoded to extract the hidden information, which is the link to the music event in the example shown in the figure.

A. QR Code Detection

YOLO was originally introduced as the first object detection model that combined bounding box prediction and objects classification into an end-to-end distinguishable network. It is written and maintained in a framework called Darknet.

In yolov4 [14], the main new features introduced by this new version are:

- 1) **Bag of Freebies:** BoF is the improvement that only increases training costs to get better performance such as

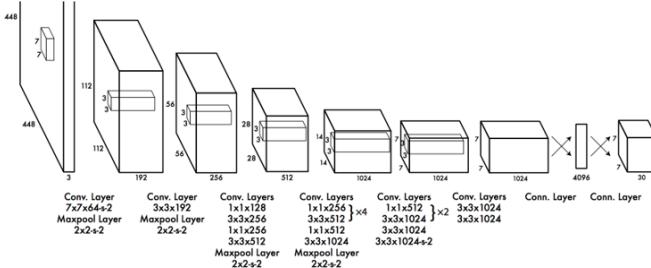


Fig. 3: YOLO architecture.

increased images. The authors discuss several methods and choose specific ones like CutMix or the Regular DropBlock, among others.

- 2) **Bag of Specials:** BoS are plugin modules and post-processing methods only increase inference cost to achieve better performance, such as the SPP module. The authors, and a few others, chose Mish activation and partially cross-stage connections.
- 3) **Architecture neck:** The authors introduce a block in the middle of the backbone, CSPDarknet53, and the first part, YOLOv3. She is SPP (Spatial Pyramid Pooling) and PAN (Path Aggregation Network).
- 4) **Mosaic augmentation:** The authors also introduced a new augmentation method called Mosaic. It combines four images of the training data set into one image.

In this paper, a transfer learning with YOLO, which is illustrated in Fig. 3, is used for QR code detection. The final layer of the network is modified to output the bounding box of the QR code. All the pre-trained parameters of earlier layers are used as the initialized parameters.

B. QR Code Decoding

After applying QR code detection, the position of the QR code in the frame has been located. The area of the QR code is cropped and fed into a QR code decoder. In this method, Pyzbar, which is a library available at <https://pypi.org/project/pyzbar/>, is used for this task.

The structure features of QR code symbol must be adequately used in its decoder procedure. The flow chart of QR code recognition is shown as Fig. 4. The main steps in the procedure of the decoding algorithm include:

- Binarization;
- Obtain the approximate region of QR code, and implement coarse positioning for QR image according to the finder patterns;
- Implement accurate positioning according to the alignment patterns;
- Calculate the angle of inclination to rotate QR image, and implement rectification processing;
- Obtain version number and implement self-adaptive sampling;
- Decode based on corrected image and input a standard 2D matrix.

IV. EXPERIMENTAL RESULTS

A. Dataset Preparation

To prepare the dataset for the proposed streaming QR code decoder, a collection of augmented videos need to be created. As mentioned earlier, this paper deals with two types of augmented video: active and passive. The active augmented video generation is shown in Fig. 5. Firstly, videos of various topics including entertainment, advertising, education, technology are collected. QR codes are then inserted into frames of these videos to embed information such as links, text, locations. Note that the embedded QR code change through frames of each video to continuously convey different information. The size of the QR code also varied to evaluate the effect of QR code size on the performance of the proposed algorithm. The procedure of making passive augmented videos are shown in Fig. 6. Scenes with printed QR codes were found and recorded. The distance from the camera to the printed QR code varies so that the effect of distance on the performance of the proposed algorithm can be evaluated.

Upon obtaining augmented videos, the dataset for training and testing the proposed algorithm was created. Firstly, the original augmented video was played on displays including computer monitors and projector scenes. Then smartphone cameras were used to record the video of these displays. These videos are cut into frames. Those frames where QR codes appear are labeled to create the dataset for training.

A total of 68 videos has been recorded using the rear camera on iPhone. We then separate the frames to proceed with labeling the dataset. The total number of photos we extracted from the videos for labeling was 3,388. We also divided the total number of tagged photos into two pieces of training and validations with the number of photos for each episode of 2779 and 609 respectively. Finally, we created a test set with 574 images to evaluate the model.

B. Training

Model parameters to train for QR code recognition: class = 1, max batches = 6000, steps = 4800, 5400 and width = 416, height = 416. The transfer learning model based on yolov4 was trained on google colab pro. Initially, the yolov4.conv.137 weight file, which was created by the authors of the darknet is trained with the dataset created in this paper. At the iteration of 1600, the loss of the model was shown to be stable and thus the training was thus decided to stop. The weight file at this iteration is used for the second training. In the second training, the model was trained with 6000 iterations in 16 hours.

C. Experimental Results

Figure 7a show the result of the detection step. It can be seen that the QR code in the streaming video can be detected at high accuracy. A test set with 574 images is used to evaluate the proposed streaming QR code decoding algorithm. The accuracy of the detection stage is evaluated through IOU (Intersection over union) and Average precision (AP). Usually, with object detection problems, mAP is the main index used for evaluation. However, since there is only one type of object,

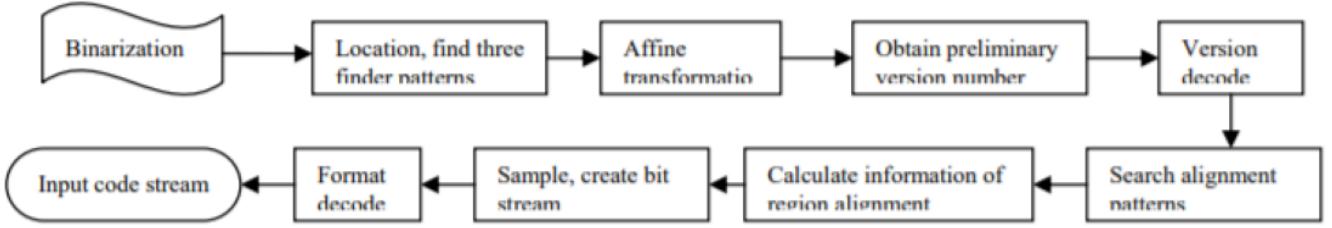


Fig. 4: The decoder flow chart of QR code.

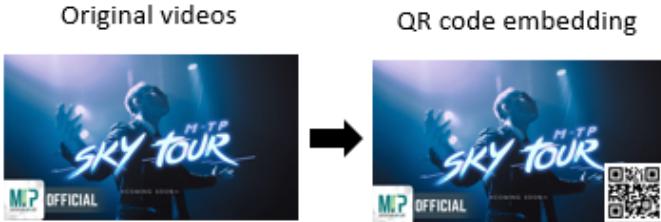


Fig. 5: Active augmented video generation.

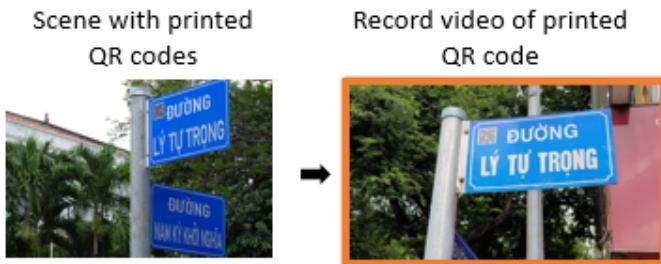


Fig. 6: Passive augmented video generation.

that is QR code, in this problem, the detection accuracy is evaluated through AP. The AP of the QR code detection step is displayed in Fig. 8. It can be seen that the detection part works well at 99.93 % AP with threshold IoU > 0.5. This is because since QR codes all have a reasonable size to have a decent possibility of being successfully decoded.

An example of the result of the decoding stage is shown in Fig. 7b. As mentioned earlier, the performance of the proposed algorithm is affected by the size of the QR code and the distance from the camera to the QR code. When the QR code is too small or the distance is too far, the QR code cannot be decoded even though it still can be detected. Figure 9 show the limitation of the proposed algorithm. More specifically, the figure shows the detectable distance corresponding to the size of the QR code to achieve a decoding rate of at least 90%.

In general, distance and size are the deciding factors for QR code decoding. As shown in Fig. 10 (a) and (b), even when the augmented video is viewed at a small angle, the QR code still can be detected and decoded successfully. Besides distance and size, the quality of the QR code images that appear on the smartphone camera also greatly affects the possibility of decoding. The QR code in the video shown in Fig. 10 (c) is large and viewed at a straight angle. However, because of the

light reflecting on the monitor, the QR code does not appear clearly in the smartphone camera. Therefore, even though being detectable, it cannot be decoded as shown in Fig. 10(d).

V. CONCLUSION

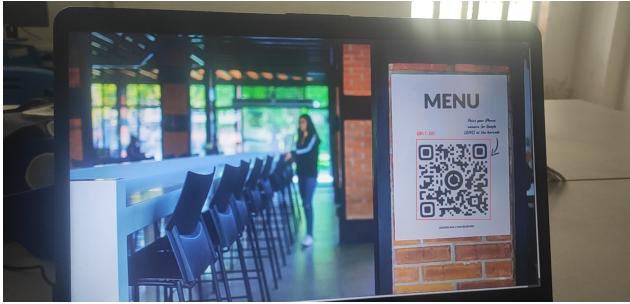
In augmented video streaming, QR codes are embedded on frames of videos streamed over the Internet. Thanks to the use of QR codes, various types of information can be intentionally embedded into videos for advertisement, education, and other applications. Since QR codes are inserted continuously into frames in the video, existing QR code decoders, which are developed for decoding static QR code, cannot be used. This paper deal with the problem of detecting and extracting information of QR code in augmented video streaming. More specifically, this paper proposed a two-stage algorithm for streaming QR code decoding. In the first stage, transfer learning with Yolov4 is adopted to detect the QR code in the frame. In the second stage, the Pyzbar library is used for decoding the QR code in the area detected in the previous stage. Experimental results on active and passive augmented videos show a good performance of the proposed algorithm. More specifically, QR codes in the tested videos were detected with an average precision (AP) of up to 99.93 %. QR code is shown to be detected and decoded successfully even when the displays which present the augmented videos are viewed at small angles. QR codes in approximately 90 % of tested videos were successfully detected. The QR codes in the remaining 10 % cannot be decoded due to the blurriness or small size of the QR code in the augmented videos. Overall, experimental results show that the proposed algorithm works well on live augmented video. At the same time, the proposed algorithm is also very lightweight, which ensures the implementation on smartphones.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1G1A1008105).

REFERENCES

- [1] A.-T. Tran, N.-N. Dao, and S. Cho, "Bitrate adaptation for video streaming services in edge caching systems," *IEEE Access*, vol. 8, pp. 135 844–135 852, 2020.
- [2] N.-N. Dao, D. T. Ngo, N.-T. Dinh, T. V. Phan, N. D. Vo, S. Cho, and T. Braun, "Hit ratio and content quality tradeoff for adaptive bitrate streaming in edge caching systems," *IEEE Systems Journal*, 2020.



(a) Detect QR code



(b) Decode QR code

Fig. 7: An experimental example of QR detection and decoding.

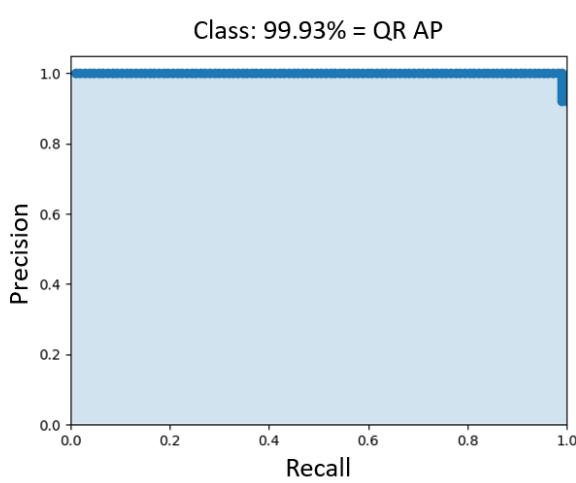


Fig. 8: Average precision.

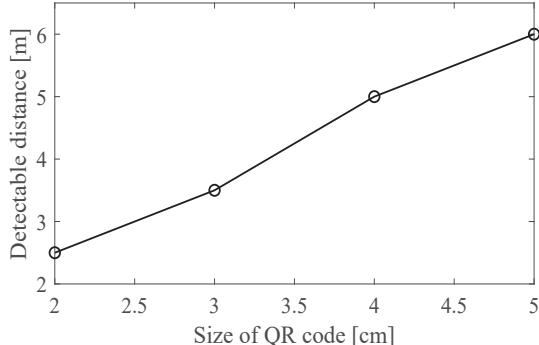


Fig. 9: Decoding efficiency.

- [3] A.-T. Tran, T.-V. Nguyen, V.-D. Tuong, N.-N. Dao, and S. Cho, “On stalling minimization of adaptive bitrate video services in edge caching systems,” in *IEEE International Conference on Information Networking (ICOIN)*. IEEE, 2020, pp. 115–116.
- [4] R. Muniz, L. Junco, and A. Otero, “A robust software barcode reader using the Hough transform,” in *IEEE International Conference on Information Intelligence and Systems (Cat. No. PR00446)*, 1999, pp. 313–319.
- [5] E. Ohbuchi, H. Hanaizumi, and L. A. Hock, “Barcode readers using the camera device in mobile phones,” in *IEEE International conference on cyberworlds*, 2004, pp. 260–265.
- [6] S. Wachenfeld, S. Terlunen, and X. Jiang, “Robust recognition of 1-D

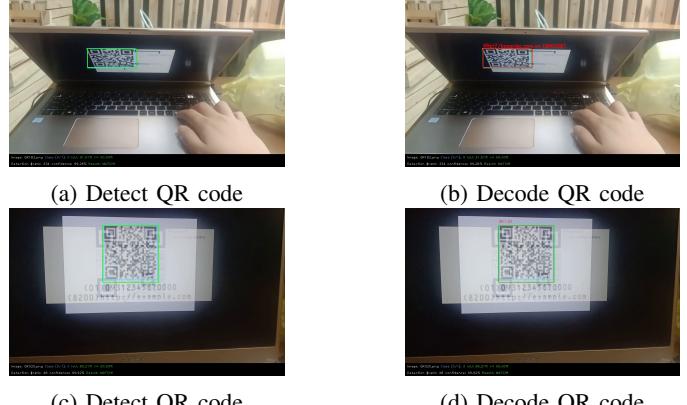


Fig. 10: Difficult cases with successful results.

barcodes using camera phones,” in *IEEE International Conference on Pattern Recognition*, 2008, pp. 1–4.

- [7] M. Katona and L. G. Nyúl, “Efficient 1D and 2D barcode detection using mathematical morphology,” in *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, 2013, pp. 464–475.
- [8] ——, “A novel method for accurate and efficient barcode detection with morphological operations,” in *IEEE International Conference on Signal Image Technology and Internet Based Systems*, 2012, pp. 307–314.
- [9] G. Sörös and C. Flöckemeier, “Blur-resistant joint 1D and 2D barcode localization for smartphones,” in *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, 2013, pp. 1–8.
- [10] O. Gallo and R. Manduchi, “Reading 1D barcodes with mobile phones using deformable templates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1834–1843, 2010.
- [11] E. Tekin and J. Coughlan, “BLaDE: Barcode localization and decoding engine,” *Tech. Rep. 2012-RERC. 01*, 2012.
- [12] C. Creusot and A. Munawar, “Real-time barcode detection in the wild,” in *IEEE winter conference on applications of computer vision*, 2015, pp. 239–245.
- [13] ——, “Low-computation egocentric barcode detector for the blind,” in *IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 2856–2860.
- [14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.