

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

-----o0o-----



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

HỆ THỐNG PHÁT HIỆN TRẠNG THÁI BÃI ĐỖ XE

GVHD: BÙI QUỐC BẢO

SVTH: HOÀNG DUY LỘC

MSSV: 1712037

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2021



Số: _____/BKĐT
Khoa: **Điện – Điện tử**
Bộ Môn: **Điện Tử**

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

1. HỌ VÀ TÊN: Hoàng Duy Lộc MSSV: 1712037

2. NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG**

3. Đề tài: Hệ thống phát hiện trạng thái bãi đỗ xe

4. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):

.....
.....
.....
.....
.....
.....

5. Ngày giao nhiệm vụ luận văn:

6. Ngày hoàn thành nhiệm vụ:

7. Họ và tên người hướng dẫn: **Th.S Bùi Quốc Bảo** Phần hướng dẫn
100%

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

Tp.HCM, ngày 15 tháng 1 năm 2022

CHỦ NHIỆM BỘ MÔN

NGƯỜI HƯỚNG DẪN CHÍNH

TS. Trần Hoàng Linh

Ths. Bùi Quốc Bảo

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):.....

Đơn vị:.....

Ngày bảo vệ :

Điểm tổng kết:

LỜI CẢM ƠN

Lời đầu tiên, em xin trân trọng cảm ơn thầy Bùi Quốc Bảo đã tận tình hướng dẫn em trong quá trình hoàn thành luận văn. Thầy là người rất tận tâm trong công việc giảng dạy, cũng như tận tình chỉ bảo sinh viên khi gặp khó khăn. Cùng đồng hành trong suốt hai năm đại học từ đồ án môn học, thực tập tốt nghiệp và cuối cùng là luận văn tốt nghiệp. Em sẽ không bao giờ quên được những công ơn từ thầy. Em xin chân thành cảm ơn các thầy, cô thuộc bộ môn Điện tử - Viễn Thông, Đại học Quốc gia thành phố Hồ Chí Minh – Trường Đại học Bách Khoa đã tạo điều kiện và hỗ trợ em trong suốt quá trình em thực hiện luận văn này.

Do bản thân em còn giới hạn về kiến thức và khả năng lý luận nên luận văn này còn nhiều thiếu sót và hạn chế, kính mong sự chỉ dẫn và đóng góp của các thầy, cô để luận văn của em hoàn thiện hơn. Em xin chân thành cảm ơn!

Tp. Hồ Chí Minh, ngày 15 tháng 1 năm 2022 .

Sinh viên

LỜI CAM ĐOAN

Tôi tên: Hoàng Duy Lộc, là sinh viên chuyên ngành Điện tử - Viễn Thông khóa 2017 tại Đại học Quốc gia thành phố Hồ Chí Minh – Trường Đại học Bách Khoa. Tôi xin cam kết luận văn này là công trình nghiên cứu của bản thân tôi, dưới sự hướng dẫn của Ths. Bùi Quốc Bảo, không sao chép của bất kỳ một cá nhân hoặc tổ chức đã được công bố nào khác. Các tài liệu được luận văn tham khảo, kế thừa và trích dẫn đều được liệt kê trong danh mục các tài liệu tham khảo.

Tôi xin chịu hoàn toàn trách nhiệm về lời cam đoan trên.

Tp. Hồ Chí Minh, ngày 15 tháng 1 năm 2022 .

Sinh viên

TÓM TẮT LUẬN VĂN

Trong luận văn này, nghiên cứu và xây dựng một hệ thống quản lý bãi đỗ xe thông qua các camera an ninh dựa trên mạng nơ-ron tích chập. Mục đích để tích hợp và giảm bớt chi phí khi vận hành cũng như quản lý chính xác kịp thời giúp bãi đỗ xe tối đa hóa được công suất. Luận văn cũng tìm hiểu về một số thuật toán học sâu cho bài toán xử lý ảnh và phân tích mức độ cần thiết của luận văn đối với toàn bộ hệ thống quản lý bãi đỗ xe.

Cùng với sự phát triển mạnh mẽ của máy học và học sâu, con người với sự giúp đỡ của máy tính đã có thể giải quyết được những vấn đề khó hơn và yêu cầu cao hơn. Luận văn này cũng sử dụng một trong các phương pháp học sâu để giải quyết yêu cầu bài toán đề ra. Kết quả của luận văn là đưa ra nhận định về tình trạng bãi đỗ xe trong thời gian thực. Cấu trúc đề tài này chia thành 3 bước chính. Đầu tiên, đánh dấu các vị trí đỗ xe trong bãi. Bước hai, các camera trích xuất từng khung ảnh và đưa vào mô hình học sâu để có thể phát hiện vị trí của đối tượng ở đây là xe ô tô. Bước cuối cùng, kiểm tra và đánh giá về trạng thái đỗ xe hiện tại của bãi đỗ. Từ đó, luận văn xây dựng một giao diện thân thiện với người dùng và cho phép áp dụng thiết kế trên với nhiều bãi đỗ khác nhau mà không cần phải đi sâu vào trong code hay phải sử dụng phần mềm bên ngoài.

Hệ thống đã chạy thử trên các luồng trực tiếp và cho kết quả độ chính xác cao và có tốc độ đáp ứng đủ nhanh. Do đó có thể ứng dụng hệ thống tích hợp vào camera an ninh trên phố cho việc quản lý các vị trí đỗ xe hay ứng dụng trong một hệ thống quản lý bãi đỗ xe thông minh.

MỤC LỤC

Chương 1 GIỚI THIỆU	1
1.1 Bối cảnh nghiên cứu	1
1.2 Mục tiêu nghiên cứu	1
1.3 Phương pháp nghiên cứu	2
1.4 Phạm vi nghiên cứu	3
1.5 Kết luận chương.....	3
Chương 2 SƠ ĐỒ KHỐI VÀ CƠ SỞ LÝ THUYẾT.....	4
2.1 Giới thiệu chung về hệ thống.....	4
2.1.1 Bài toán	4
2.1.2 Đầu vào/ra của hệ thống.....	5
2.1.3 Sơ đồ khối hệ thống	5
2.2 Một số phương pháp về phát hiện cạnh biên	7
2.2.1 Giới thiệu chung.....	7
2.2.2 Một số kiểu đường biên.....	7
2.2.3 Các phương pháp phát hiện biên.....	8
2.2.4 Phương pháp Gradient.....	8
2.2.4 Laplacian of Gaussian (LOG)	11
2.3 Giải thuật Mạng nơ-ron nhân tạo (Artificial Neural Network – ANN).....	13
2.4 Mạng nơ-ron tích chập.....	16
2.4.1 Phép tích chập	18
2.4.2 Padding và Stride	19
2.4.3 Convolution Layer.....	20
2.4.4 Pooling Layer	21
2.4.5 Fully Connected Layer	22
2.4.6 Mô hình convolutional neural network	23

2.4.7 Softmax Function	23
2.4.8 Loss Function	24
2.4.9 Gradient Descent	25
2.5 Thuật toán You Only Look Once (YOLO).....	26
2.5.1 Nhận xét, đánh giá và lý do lựa chọn.....	26
2.5.2 Mô hình mạng YOLO tổng quát	27
2.5.3 Độ chồng lấn (IOU).....	33
2.5.4 Loss Function	34
2.5.5 Anchor Box	36
2.5.6 Bounding Box	37
2.5.7 Non-max suppression (NMS).....	38
2.5.8 Đánh giá model	39
2.6 Kết luận chương.....	45
Chương 3 XÂY DỰNG MÔ HÌNH XÁC ĐỊNH TRẠNG THÁI BÃI ĐỖ XE.....	46
3.1 Huấn luyện mô hình nhận dạng xe ô tô	46
3.1.1 Chuẩn bị dữ liệu train model.....	46
3.1.2 Huấn luyện mô hình trên google colab	49
3.2 Xây dựng hệ thống xác định trạng thái của bãi đỗ xe.....	53
3.2.1 Xây dựng hệ thống ứng dụng mô hình học sâu Yolov3.....	53
3.2.2 Giải thuật phát hiện và định vị xe ô tô	54
3.2.3 Giải thuật đánh dấu vị trí đỗ xe	58
3.2.4 Giải thuật phân biệt trạng thái ô đỗ xe	60
3.2.5 Giải thuật phân biệt trạng thái ô đỗ xe dùng xử lý ảnh	61
3.2.6 Đánh giá việc dùng đỗ của xe ô tô	63
3.3 Kết luận.....	64
Chương 4 XÂY DỰNG ỨNG DỤNG QUẢN LÝ BÃI ĐỖ XE	65
4.1 Giới thiệu về tkinter	65

4.2 Xây dựng ứng dụng	66
4.3 Xây dựng nền tảng máy chủ Node.js	68
4.3.1 Giới thiệu Node.js	68
4.3.2 Giới thiệu RESTful Webservice.....	70
4.3.3 Thiết kế webserver	71
4.4 Hệ quản trị cơ sở dữ liệu.....	72
4.5 Kết luận.....	72
Chương 5 KẾT QUẢ THỰC HIỆN	73
5.1 Kết quả thực hiện	73
5.2 Đánh giá hệ thống	75
Chương 6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	77
6.1 Kết luận.....	77
6.2 Hướng phát triển của luận văn trong tương lai	77
TÀI LIỆU THAM KHẢO	78

DANH SÁCH HÌNH MINH HỌA

Hình 2.1 Sơ đồ hệ thống.....	5
Hình 2.2 Một số kiểu đường biên thông dụng.....	7
Hình 2.3 Toán tử Sobel.....	9
Hình 2.4 Toán tử Prewitt.....	9
Hình 2.5 Toán tử Robert.....	10
Hình 2.6 Toán tử Laplacian.....	12
Hình 2.7 Mạng nơ ron nhân tạo ba lớp.....	13
Hình 2.8 Đồ thị hàm sigmoid (a) và hàm tanh (b).....	15
Hình 2.9 Đồ thị hàm ReLU và Leaky ReLU.....	15
Hình 2.10 Cấu trúc của một mạng CNN hiện đại.....	18
Hình 2.11 Phép tính tích chập.....	19
Hình 2.12 Phép tính với padding = 1 và stride = 2.....	19
Hình 2.13 Tính tích chập với hình ảnh màu (RGB).....	21
Hình 2.14 Các dạng Pooling.....	22
Hình 2.15 Flattening.....	23
Hình 2.16 Mô hình CNN với ảnh đầu vào.....	23
Hình 2.17 Chia ảnh thành các ma trận ô vuông.....	29
Hình 2.18 Sơ đồ kiến trúc mạng YOLO.....	30
Hình 2.19 Kiến trúc mạng YOLOv3.....	31
Hình 2.20 Kiến trúc một output của model YOLO.....	32
Hình 2.21 Các bản đồ đặc trưng của mạng YOLO.....	33
Hình 2.22 Phép tính IoU.....	34
Hình 2.23 Chuẩn hóa đối tượng.....	35
Hình 2.24 Xác định anchor box cho một vật thể.....	37
Hình 2.25 Minh họa dự đoán khung giới hạn đối tượng.....	38

Hình 2.26 Non-max Suppression.....	39
Hình 2.27 Underfitting (a) Trade – Off (b) Overfitting (c)	40
Hình 2.28 Biểu đồ mô tả quá trình huấn luyện.....	41
Hình 2.29 Cách tính Precision và Recall	42
Hình 2.30 Bảng thống kê output.....	43
Hình 2.31 Đồ thị biểu thị mối quan hệ giữa precision và recall.....	44
Hình 2.32 Mối quan hệ giữa AP với precision-recall.....	45
Hình 3.1 Nhận dạng xe ô tô.....	46
Hình 3.2 Open Image Dataset V6.....	47
Hình 3.3 Ảnh minh họa và định dạng tệp train model YOLOv3	48
Hình 3.4 Cấu hình phần cứng Google Colab cung cấp	49
Hình 3.5 Quá trình huấn luyện mô hình	51
Hình 3.6 Sơ đồ giải thuật cho quá trình đào tạo và thử nghiệm mô hình Yolov3.....	52
Hình 3.7 Sơ đồ giải thuật hệ thống phát hiện trạng thái ô đỗ xe	53
Hình 3.8 Sơ đồ giải thuật phát hiện và định vị xe ô tô	55
Hình 3.9 Định dạng ngõ ra outs.....	56
Hình 3.10 Kết quả nhận được.....	57
Hình 3.11 Giải thuật đánh dấu vị trí đỗ xe	58
Hình 3.12 Kết quả thực hiện.....	59
Hình 3.13 Giải thuật phân biệt trạng thái ô đỗ xe	60
Hình 3.14 Giải thuật phân biệt trạng thái ô đỗ xe bằng xử lý ảnh.....	61
Hình 3.15 Chuyển đổi ảnh xám.....	62
Hình 3.16 Lọc nhiễu và làm mịn ảnh	62
Hình 3.17 Kết quả thực hiện.....	63
Hình 3.18 Lưu đồ sử dụng timer đánh giá trạng thái ô đỗ xe.....	64
Hình 4.1 Nguyên tắc hoạt động của Tkinter.....	65
Hình 4.2 Sơ đồ giải thuật của ứng dụng	66

Hình 4.3 Màn hình chức năng add camera	67
Hình 4.4 Màn hình chức năng monitor.....	67
Hình 4.5 Giải thuật của một luồng xử lý camera.....	68
Hình 4.6 Các kiểu MINE phổ biến thường dùng với REST service	71
Hình 4.7 Bảng chứa thông tin về quản lý user	72
Hình 4.8 Bảng chứa thông tin về tọa độ của vị trí đỗ xe	72
Hình 4.9 Bảng chứa thông tin trạng thái về vị trí đỗ xe	72
Hình 5.1 Một góc camera tại bãi đỗ xe 1.....	73
Hình 5.2 Tại một góc camera khác tại bãi đỗ xe 1	74
Hình 5.3 Tại bãi đỗ xe 2	75

Chương 1 GIỚI THIỆU

1.1 Bối cảnh nghiên cứu

Trong những năm gần đây cùng với sự phát triển của nền kinh tế, thu nhập đời sống của người dân được nâng lên cùng với đó là sự gia tăng chóng mặt của số lượng phương tiện đặc biệt ở các thành phố lớn. Trong khi đó, theo thống kê, hiện nay số lượng bãi đỗ xe có giấy phép ở các thành phố chỉ đáp ứng được 8-10% nhu cầu người dân, dẫn tới tình trạng thiếu bãi đỗ xe là vô cùng nghiêm trọng. Giao thông tĩnh ở các thành phố lớn như Hà Nội, TP. Hồ Chí Minh, Đà Nẵng hiện nay đang là một bài toán khó và cần giải quyết ngay. Hàng ngày, hàng giờ chúng ta vẫn thường xuyên được nghe những thông tin về tình trạng tắc đường tại các tuyến phố, đặc biệt trong những giờ cao điểm. Và một trong những nguyên nhân gây ra tình trạng đó là việc đỗ, dừng xe không đúng nơi quy định. Các điểm đỗ, dừng xe được tận dụng ở mọi chỗ, mọi nơi: trên vỉa hè, lòng đường, công viên các nơi không phép... điều đó ảnh hưởng không nhỏ đến giao thông nội đô và mỹ quan đô thị.

Cùng với đó là việc quản lý, vận hành trong các bãi đỗ xe chưa tối ưu và mang lại hiệu quả cao cho người quản lý cũng như người lái xe, ví dụ như: thiếu thông tin số lượng chỗ đỗ xe còn trống, người lái xe phải dành nhiều thời gian để tìm kiếm một chỗ đỗ trong một không gian đồ ô tô rộng lớn

Vì vậy, phát triển một hệ thống quản lý, nhận diện chỗ đỗ xe ô tô là thật sự cần thiết bởi các lợi ích: tiết kiệm thời gian, điều hành phương tiện lưu thông dễ dàng, thuận tiện cho người sử dụng, giải quyết tình trạng khan hiếm chỗ đỗ xe là vô cùng cần thiết đối với các đô thị lớn của nước ta.

1.2 Mục tiêu nghiên cứu

Đề tài: “ Hệ thống quản lí bãi giữ xe thông minh” đưa ra phương pháp để tối ưu hóa việc giám sát, quản lý bãi đỗ xe nhằm cải thiện những khó khăn còn gặp đối với người quản lý và người lái xe trong khu vực đỗ xe tại các thành phố lớn. Đối tượng nghiên cứu của đề tài đánh giá trạng thái của vị trí đỗ xe trong bãi là “trống” hay “không trống” theo thời gian thực để đưa ra những đánh giá tốt nhất về sức chứa của bãi đỗ hiện tại giúp nâng cao được tối đa công suất hoạt động của bãi đỗ xe

1.3 Phương pháp nghiên cứu

Trong các thành phố lớn, người lái xe luôn phải đối mặt với khó khăn trong việc lựa chọn bãi đỗ xe và tìm kiếm chỗ đỗ xe trong khu vực đỗ xe lớn. Một số phương pháp đã được đề xuất nhằm hỗ trợ việc quản lý bãi đỗ xe như sử dụng rào chắn kết hợp với cảm biến hồng ngoại, cảm biến mặt đất, camera nhiệt...

Phương pháp sử dụng rào chắn kết hợp với cảm biến hồng ngoại có ưu điểm là dễ triển khai, lắp đặt nhưng lại không linh động trong một số khu vực và khi có các yếu tố bên ngoài tác động, ví dụ như: xe ô tô đi vào nhưng không đỗ lại (xe giao hàng, xe đưa thư,...), xe đi vào là xe máy xe đạp,...

Phương pháp sử dụng cảm biến mặt đất có ưu điểm là đơn giản, dễ triển khai nhưng nhược điểm là tốn kém vì mỗi một vị trí cần đặt một cảm biến, không phù hợp với điều kiện thời tiết ngoài trời,...

Phương pháp sử dụng camera nhiệt có ưu điểm đơn giản, dễ triển khai, có thể thông báo được trạng thái vị trí đỗ xe. Tuy nhiên, nhược điểm chỉ có thể phát hiện khi các phương tiện duy trì nhiệt nếu không hệ thống sẽ phải hỗ trợ bởi các thiết bị khác, có thể nhận diện nhầm với các xe đang di chuyển qua bãi đỗ xe, hơn nữa chi phí của một chiếc camera nhiệt khá cao...

Hiện nay, các bãi đỗ xe thường được trang bị CCTV camera với mục đích giám sát đồng thời sử dụng cho các hệ thống phát hiện tự động. Với điều kiện trang thiết bị có sẵn và khả năng quan sát rộng nên hệ thống phân loại trạng thái chỗ đỗ xe bằng camera là giải pháp khả thi để giám sát bãi đỗ và giảm thời gian tìm kiếm vị trí đỗ xe trống cho người sử dụng, đặc biệt là với những bãi đỗ xe lớn. Ưu điểm của phương pháp này là ta có thể giám sát được vài chục vị trí đỗ chỉ với một camera, và việc thi công đơn giản không ảnh hưởng nhiều đến môi trường. Về hoạt động của hệ thống giám sát: Camera chụp hình ảnh toàn cảnh bãi đỗ xe, một bộ xử lý phải xử lý hình ảnh đó để đưa ra số vị trí còn trống và trống tại những vị trí nào. Các vị trí trống luôn được cập nhật liên tục và được hiển thị trên màn hình đặt tại các vị trí chiến lược trong khu vực đỗ xe nhằm hỗ trợ cho người dùng.

Nội dung nghiên cứu trong luận văn này tập trung vào giải quyết, phát triển phương pháp xác định trạng thái của các vị trí ô đỗ xe. Các nội dung được thực hiện trong quá trình nghiên cứu đề tài bao gồm:

- Xây dựng phần mềm cho phép người quản lý đánh dấu tọa độ vị trí các ô đỗ xe trong bãi
- Tìm hiểu các mô hình nhận diện và phân loại đối tượng theo thời gian thực
- Đánh giá, so sánh và lựa chọn mô hình tối ưu dựa trên kết quả
- Xây dựng giao diện quản lý và tương tác người dùng
- Đánh giá tổng quan hệ thống và đưa ra hướng phát triển cho tương lai

1.4 Phạm vi nghiên cứu

Phạm vi đề tài sẽ được trình bày thành các chương với những nội dung khái quát sau:

- **Chương 1:** Khảo sát các phương pháp sử dụng để quản lý bãi đỗ xe nhằm đưa ra được phương pháp tối ưu và mang hiệu quả cao cho người quản lý, người sử dụng hệ thống quản lý bãi đỗ xe. Từ đó, xác định được mục tiêu và nhiệm vụ phải thực hiện trong luận văn để đạt được kết quả với phương pháp giám sát, quản lý bãi đỗ xe sử dụng camera mà ta đã lựa chọn
- **Chương 2:** Trình bày sơ đồ khối hệ thống và lý thuyết về mạng nơ-ron tích chập, giải thuật yolov3 lí do lựa chọn mô hình yolov3 cho giai đoạn nhận diện xe ô tô
- **Chương 3:** Xây dựng mô hình xác định trạng thái của các chỗ đỗ xe áp dụng thuật toán yolov3
- **Chương 4:** Xây dựng ứng dụng quản lí bãi đỗ xe

Phần cuối, trình bày những kết quả đã đạt được sau khi hoàn thành công việc đề ra, tổng kết những hạn chế mà chương trình chưa đạt được. Cuối cùng, đưa ra một số giải pháp khắc phục những hạn chế và đề xuất hướng phát triển của hệ thống trong tương lai.

1.5 Kết luận chương

Trong phần này, chúng ta đã đi khảo sát các phương pháp quản lý bãi đỗ xe nhằm chọn ra một phương pháp quản lý tối ưu phù hợp với mục tiêu giúp người lái xe tiết kiệm được thời gian trong việc tìm kiếm vị trí đỗ xe trống. Từ đó, chúng ta xác định mục tiêu và nhiệm vụ phải thực hiện để xây dựng một hệ thống được trình bày chi tiết ở chương tiếp theo.

Chương 2 SƠ ĐỒ KHỐI VÀ CƠ SỞ LÝ THUYẾT

2.1 Giới thiệu chung về hệ thống

2.1.1 Bài toán

Bài toán đặt ra là xây dựng một hệ thống quản lý bãi đỗ xe ngoài trời tự động và ít có sự quản lý của con người. Hệ thống này có thể thông tin kịp thời cho người lái xe về vị trí bãi đỗ cũng như số lượng hay vị trí cụ thể của từng ô đỗ còn trống ở trong bãi. Nhiều phương pháp đã được triển khai sử dụng như lắp đặt cảm biến tại từng vị trí đỗ để kiểm soát trạng thái của từng vị trí, sau đó tổng hợp và đưa ra số lượng cũng như những vị trí đỗ còn trống. Phương pháp này có ưu điểm là dễ triển khai và đơn giản tuy nhiên lại gây tốn kém khi phải sử dụng nhiều cảm biến cho các bãi đỗ lớn. Hoặc một số nơi sử dụng bộ đếm ở cửa ra vào để kiểm soát số lượng xe trong bãi đỗ. Phương pháp này tuy đơn giản, chi phí thấp nhưng lại không bao quát được nhiều trường hợp như xe đi qua nhưng không đỗ lại hay không kiểm soát được vị trí đỗ của xe.

Ngoài những phương pháp trên, một trong những phương án phổ biến và có nhiều ưu điểm hiện nay là quản lý bãi đỗ xe sử dụng camera. Hệ thống quản lý bằng cách nhận diện trạng thái bãi đỗ xe ô tô sử dụng camera, có thể ứng dụng giám sát tại các bãi đỗ xe lòng đường, trung tâm thương mại, trường học... với các vị trí đỗ được bố trí theo những vị trí cố định và góc đặt camera cũng cố định. Phương pháp này có ưu điểm là lắp đặt dễ, thi công ít làm ảnh hưởng đến môi trường thực tế, giá thành rẻ và vận hành bảo dưỡng dễ dàng. Mỗi camera có thể kiểm soát được khoảng không gian khá lớn, có khả năng nhận ra vài chục vị trí đỗ ô tô và tình trạng chỗ đỗ, do đó tiết kiệm được kinh phí thiết bị cũng như kinh phí bảo dưỡng. Ở mỗi bãi đỗ ô tô, hệ thống cho phép phát hiện ra vị trí từng ô đỗ xe và đưa ra kết luận vị trí đó còn trống hay không, tổng hợp số chỗ đỗ trống, cập nhật về hệ thống quản lý trung tâm để cập nhật lên tiện ích của người dùng, người đang điều khiển phương tiện và có nhu cầu tìm kiếm vị trí đỗ.

Để phân loại vị trí đỗ dựa vào ảnh chụp bãi đỗ, hệ thống cần một thuật toán xử lý ảnh hiệu quả. Có nhiều thuật toán được sử dụng cho công việc này như Background Subtraction, Color-based SVM (Support Vector Machine), ... Tuy nhiên, phương án phổ biến hiện nay và cũng là phương án cho độ chính xác cao là sử dụng phương pháp học sâu (Deep Learning) hay mạng nơ-ron tích chập CNN. Phương pháp này có thể cho

độ chính xác gần ngang bằng với con người nhưng nhược điểm của nó là cần bộ dữ liệu đủ lớn cho quá trình huấn luyện để cho ra được kết quả chính xác.

Hệ thống sử dụng camera giám sát cần đạt được các yêu cầu sau:

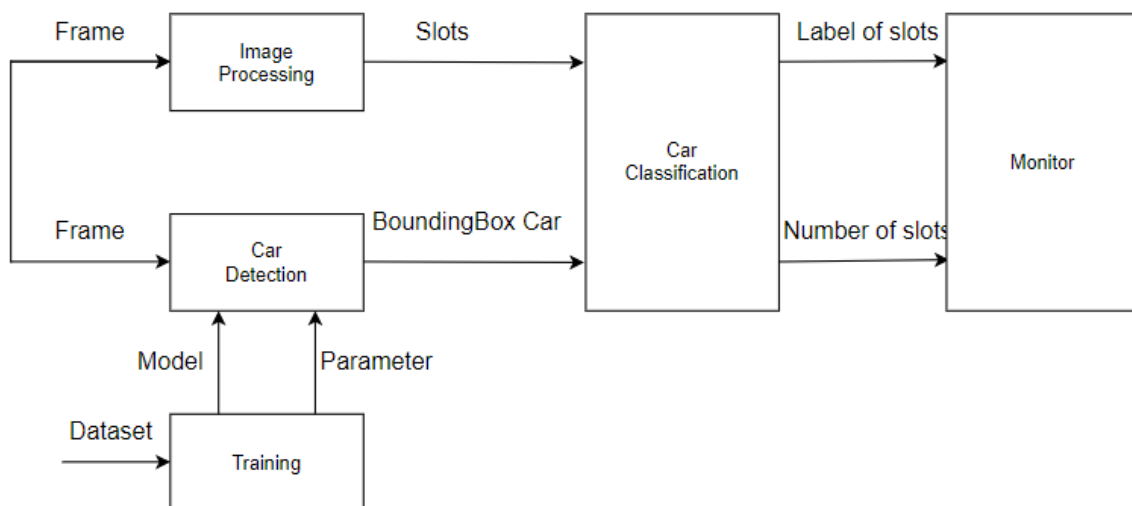
- Phát hiện được trạng thái có xe và không có xe đang đỗ tại vị trí lắp đặt với các loại xe ô tô cá nhân thông dụng tại Việt Nam
- Hoạt động trong điều kiện có ánh sáng ban ngày hoặc đèn chiếu sáng trên đường phố.
- Kết nối tới trạm xử lý tập trung qua các công nghệ truyền thông không dây
- Hệ thống stand – alone, có kích thước nhỏ gọn, tiện lắp đặt và triển khai

2.1.2 Đầu vào/ra của hệ thống

Hệ thống yêu cầu nhận dữ liệu đầu vào là các hình ảnh từ các luồng camera giám sát. Sau đó, kết quả hệ thống trả về là trạng thái của từng vị trí đỗ xe và tổng số vị trí còn trống.

2.1.3 Sơ đồ khối hệ thống

Sơ đồ khối của hệ thống được mô tả như hình sau:



Hình 2.1 Sơ đồ hệ thống

Mô tả hoạt động của hệ thống:

- Hệ thống xử lý sẽ nhận các hình ảnh (frame) từ camera và đưa vào khối Image Processing nhằm mục đích tiền xử lý hình ảnh trước khi phân loại vị trí đỗ. Tại khối này, cho phép người quản lý đánh dấu vị trí đỗ xe thông qua

thao tác click chuột. Thông thường khi camera quan sát một vùng bãi đỗ xe rộng sẽ có nhiều vị trí đỗ xe không thể dùng hình chữ nhật để bao phủ toàn bộ ô đỗ xe. Do đó, hình khối ta sử dụng ở đây là một tứ giác bất kỳ điều đó cho phép đánh dấu được toàn bộ ô đỗ xe. Sau đó tọa độ của các ô đỗ xe sẽ được lưu lại phục vụ cho quá trình xử lý tiếp theo.

- Khối Training có nhiệm vụ huấn luyện mô hình trên tập dữ liệu thu được (dataset) thông qua mô hình Yolov3 và đưa ra các trọng số tới khối Car Detection phục vụ cho quá trình phát hiện các xe hiện có trong bãi đỗ xe
- Từ model thu được trong quá trình Training khối Car Detection xác định có hay không xe ô tô trong bãi đỗ xe và vị trí cụ thể của ô tô dưới dạng bounding box bao quanh đối tượng.
- Từ tọa độ vị trí của các ô đỗ xe và bounding box quanh đối tượng xe ô tô, ta sử dụng IoU để kiểm tra xem vị trí đỗ xe đó có bị chiếm dụng (chồng lấn) bởi bất kỳ chiếc ô tô nào không, trả về trạng thái của vị trí đỗ xe.
- Khối monitor dùng để quản lý các camera và hiển thị thông tin về trạng thái bãi đỗ xe theo thời gian thực

Khi phát triển hệ thống trên, em đã thực hiện công việc như sau:

1. Phát triển phần mềm hệ thống, bao gồm:
 - a. Triển khai model sử dụng để nhận dạng xe ô tô.
 - b. Xây dựng phần mềm cho phép người quản lý đánh dấu những vị trí có thể đỗ xe trong bãi .
 - c. Xây dựng phần mềm hỗ trợ phân loại trạng thái vị trí đỗ xe.
 - d. Xây dựng phần mềm giao diện quản lý bãi đỗ xe.
2. Phát triển phần cứng hệ thống, bao gồm:
 - a. Nghiên cứu chọn lựa camera thích hợp cho hệ thống. Ảnh từ camera cần có độ phân giải cao để hỗ trợ việc nhận diện ô tô tại bãi đỗ xe. Camera cần hỗ trợ các chuẩn như USB, HDMI và cho phép truy cập theo địa chỉ IP. Hệ thống cho phép lựa chọn kết nối và điều khiển chụp tới một hoặc một vài camera.
 - b. Nghiên cứu kết nối truyền thông tin giữa camera và hệ thống.
3. Chạy thử nghiệm hệ thống với các video quay chụp từ các bãi đỗ xe được stream thông qua phần mềm VLC với giao thức truyền tin rtp.

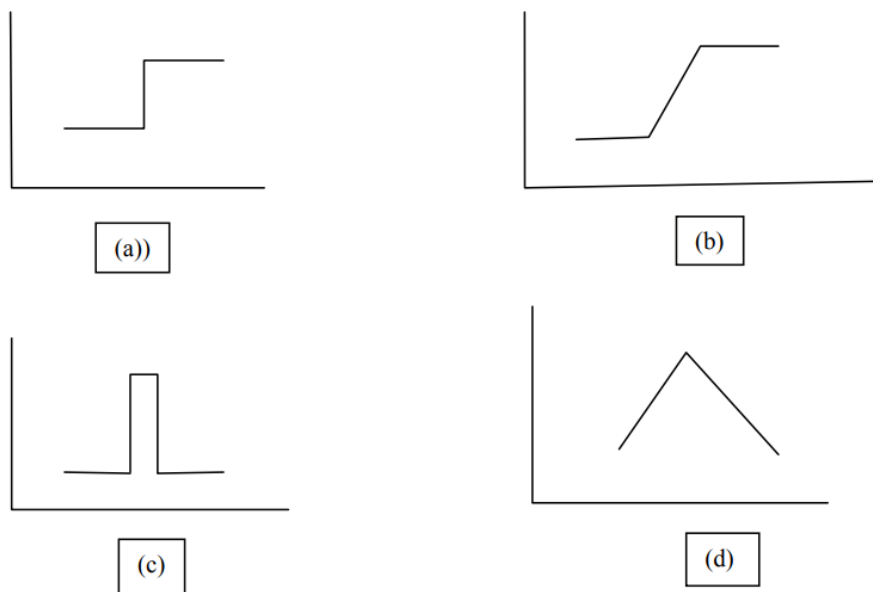
2.2 Một số phương pháp về phát hiện cạnh biên

2.2.1 Giới thiệu chung

Phát hiện biên của ảnh là một trong những nhiệm vụ quan trọng trong xử lý ảnh. Nhận dạng ảnh dùng máy tính liên quan tới việc nhận dạng và phân loại các đối tượng trong bức ảnh do đó phát hiện biên là một công cụ quan trọng. Phát hiện biên sẽ làm giảm một cách đáng kể khối lượng dữ liệu cần xử lý và loại bỏ các thông tin không cần thiết trong khi vẫn đảm bảo các thuộc tính quan trọng về cấu trúc của ảnh. Có rất nhiều kỹ thuật phát hiện biên hiện đang được sử dụng, mỗi kỹ thuật này thường làm việc một cách có hiệu quả cao đối với một loại đường biên cụ thể.

2.2.2 Một số kiểu đường biên

Đường biên là nơi mà các điểm ảnh lân cận nhau có cường độ thay đổi mạnh một cách đột ngột. Một số kiểu đường biên hay gặp trên thực tế được minh họa trên hình



Hình 2.2 Một số kiểu đường biên thông dụng

- (a) Biên dạng nhảy bậc
- (b) Biên dốc
- (c) Biên dạng xung vuông
- (d) Biên dạng hình nón

2.2.3 Các phương pháp phát hiện biên

Các phương pháp phát hiện biên truyền thống thường dựa trên kết quả của phép tích chập (convolution) giữa bức ảnh cần nghiên cứu $f(x,y)$ và một bộ lọc 2D (filter) $h(x,y)$ thường được gọi là mặt nạ (mask).

$$h(x,y) * f(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(k_1, k_2) f(x - k_1, y - k_2) dk_1 dk_2$$

Nếu $h(x,y)$ và $f(x,y)$ có dạng rời rạc thì công thức viết lại thành

$$h(n_1, n_2) * f(n_1, n_2) = \sum_{-\infty}^{+\infty} \sum_{-\infty}^{+\infty} h(k_1, k_2) f(x - k_1, y - k_2)$$

Trên thực tế người ta hay dùng $h(n_1, n_2)$ là ma trận $[3 \times 3]$ như sau :

$$h = \begin{pmatrix} h(-1,1) & h(0,1) & h(1,1) \\ h(-1,0) & h(0,0) & h(1,0) \\ h(-1,-1) & h(0,-1) & h(1,-1) \end{pmatrix}$$

Cấu trúc và giá trị của các toán tử phát hiện biên sẽ xác định hướng đặc trưng mà toán tử nhạy cảm với biên. Có một số toán tử thích hợp cho các đường biên có hướng nằm ngang, một số toán tử lại thích hợp cho việc tìm kiếm biên dạng thẳng đứng hay theo hướng đường chéo. Có nhiều phương pháp phát hiện biên đang được áp dụng, tuy nhiên ta có thể phân thành hai nhóm cơ bản là phát hiện biên dùng Gradient và phương pháp Laplacian. Phương pháp phát hiện biên dùng Gradient (sử dụng các toán tử Roberts, Prewitt, Sobel, Canny) dựa vào tính giá trị cực đại và cực tiểu của đạo hàm bậc nhất của ảnh. Phương pháp Laplacian sẽ tìm kiếm những điểm có giá trị 0 khi lấy đạo hàm bậc hai của ảnh (Mars-Hildreth).

2.2.4 Phương pháp Gradient

Đạo hàm bậc nhất theo hướng ngang và dọc được tính theo

$$\Delta f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Biên độ của gradient vector hay độ lớn tổng cộng của giá trị đạo hàm nằm tại biên là kết hợp của cả hai giá trị này theo công thức

$$\Delta f = |\Delta f| = \sqrt{G_x^2 + G_y^2}$$

Hướng của gradient vector được xác định theo

$$angle\ of\ \nabla f = \tan^{-1}\left(\frac{G_y}{G_x}\right) = \sqrt{G_x^2 + G_y^2}$$

Hướng của biên sẽ vuông góc với hướng của gradient vector này.

Toán tử Sobel

Trên thực tế Sobel sử dụng hai mặt nạ có kích thước $[3 \times 3]$ trong đó một mặt nạ chỉ đơn giản là sự quay của mặt nạ kia đi một góc 90^0 . Các mặt nạ này được thiết kế để tìm ra các đường biên theo chiều đứng và chiều ngang một cách tốt nhất. Khi thực hiện phép convolution giữa ảnh và các mặt nạ này ta nhận được các gradient theo chiều đứng và chiều ngang G_x, G_y . Toán tử Sobel có dạng như hình

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Hình 2.3 Toán tử Sobel

Toán tử Prewitt

Phương pháp Prewitt gần giống với Sobel. Đây là phương pháp lâu đời nhất, cổ điển nhất. Toán tử Prewitt được mô tả trên hình

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Hình 2.4 Toán tử Prewitt

Toán tử Robert

Tương tự như Sobel, ta tính đường biên ngang và dọc một cách riêng rẽ dùng 2 mặt nạ như hình sau đó tổng hợp lại để cho đường biên thực của ảnh. Tuy nhiên do mặt nạ của Robert khá nhỏ nên kết quả là bị ảnh hưởng khá nhiều của nhiễu.

0	0	0
0	-1	0
0	0	1

0	0	0
0	0	-1
0	1	0

Hình 2.5 Toán tử Robert

Phương pháp Canny

Phương pháp này sử dụng hai mức ngưỡng cao và thấp. Ban đầu ta dùng mức ngưỡng cao để tìm điểm bắt đầu của biên, sau đó chúng ta xác định hướng phát triển của biên dựa vào các điểm ảnh liên tiếp có giá trị lớn hơn mức ngưỡng thấp. Ta chỉ loại bỏ các điểm có giá trị nhỏ hơn mức ngưỡng thấp. Các đường biên yếu sẽ được chọn nếu chúng được liên kết với các đường biên khỏe.

Phương pháp Canny bao gồm các bước sau:

Bước 1. Trước hết dùng bộ lọc Gaussian để làm mịn ảnh.

$$G'(x) = \left(-\frac{x}{\sigma^2}\right) e^{-\left(\frac{x^2}{2\sigma^2}\right)}$$

Bước 2. Sau đó tính toán gradient và của đường biên của ảnh đã được làm mịn

$$C_x[x, y] = -\left(\frac{j}{\sigma^2}\right) e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

$$C_y[x, y] = -\left(\frac{j}{\sigma^2}\right) e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

Bước 3. Tiếp theo là loại bỏ những điểm không phải là cực đại.

Bước 4. Bước cuối cùng là loại bỏ những giá trị nhỏ hơn mức ngưỡng.

Phương pháp này hơn hẳn các phương pháp khác do ít bị tác động của nhiễu và cho khả năng phát hiện các biên yếu. Nhược điểm của phương pháp này là nếu chọn ngưỡng

quá thấp sẽ tạo ra biên không đúng, ngược lại nếu chọn ngưỡng quá cao thì nhiều thông tin quan trọng của biên sẽ bị loại bỏ. Căn cứ vào mức ngưỡng đã xác định trước, ta sẽ quyết định những điểm thuộc biên thực hoặc không thuộc biên. Nếu mức ngưỡng càng thấp, số đường biên được phát hiện càng nhiều (nhưng kèm theo là nhiễu và số các đường biên giả cũng xuất hiện càng nhiều). Ngược lại nếu ta đặt mức ngưỡng càng cao, ta có thể bị mất những đường biên mờ hoặc các đường biên sẽ bị đứt đoạn.

Phương pháp Canny có các ưu điểm sau:

- Cực đại hóa tỷ số tín hiệu trên nhiễu làm cho việc phát hiện các biên thực càng chính xác.
- Đạt được độ chính xác cao của đường biên thực
- Làm giảm đến mức tối thiểu số các điểm nằm trên đường biên nhằm tạo ra các đường biên mỏng, rõ.

2.2.4 Laplacian of Gaussian (LOG)

Dùng phương pháp gradient sẽ cho kết quả là ảnh nhận được có cấu trúc không rõ nét do tạo nên những đường biên dày, không sắc nét. Để nhận được các đường biên mỏng và rõ nét, ta phải tiến hành các bước xử lý tiếp theo như loại bỏ những điểm không phải là cực trị (nonmaximum suppression) đồng thời áp dụng kỹ thuật liên kết biên (edge linking). Ngoài ra ta còn gặp phải vấn đề là làm thế nào để xác định được mức ngưỡng một cách chính xác. Việc chọn đúng giá trị ngưỡng phụ thuộc rất nhiều vào nội dung của từng bức ảnh. Nếu ta tăng gấp đôi kích thước của một bức ảnh mà không thay đổi giá trị cường độ của các điểm ảnh, ta sẽ nhận được gradients bị suy giảm đi một nửa. Mặt khác kích thước của mặt nạ (masks) cũng ảnh hưởng nhiều đến giá trị của gradients trong ảnh.

Phương pháp gradient chỉ thích hợp cho các vùng ảnh độ tương phản thay đổi có tính nhảy bậc, điều này gây khó khăn cho phát hiện các đường thẳng. Để khắc phục nhược điểm này ta thường dùng đạo hàm bậc hai. Phương pháp Laplacian cho phép xác định đường biên dựa vào giá trị 0 của đạo hàm bậc hai của ảnh. Laplacian của một ảnh tại điểm $I(x,y)$ được tính theo (10):

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Laplacian được kết hợp với bộ lọc làm mịn ảnh để tìm biên. Xét công thức sau:

$$h(r) = -e^{-\left(\frac{r^2}{2\sigma^2}\right)}$$

Ở đây $r^2 = x^2 + y^2$ và σ là độ lệch chuẩn (standard deviation). Nếu thực hiện phép tích chập của hàm này với ảnh cần tìm biên, kết quả là ảnh sẽ bị mờ đi, mức độ mờ phụ thuộc vào giá trị của σ . Laplacian của h tức đạo hàm bậc hai của h theo r là:

$$\nabla^2 h(r) = \left[\frac{r^2 - \sigma^2}{\sigma^4} \right] - e^{-\left(\frac{r^2}{2\sigma^2}\right)}$$

Hàm này thường được gọi là Laplacian of a Gaussian (LoG) do có dạng Gaussian.

Trong phương pháp này, bộ lọc Gaussian được kết hợp với Laplacian cho phép hiển thị những vùng ảnh có cường độ thay đổi nhanh do đó làm tăng hiệu quả phát hiện biên. Nó cho phép làm việc với một diện tích rộng hơn xung quanh điểm ảnh đang được nghiên cứu nhằm phát hiện chính xác hơn vị trí của đường biên. Nhược điểm của phương pháp này là không xác định được hướng của biên do sử dụng hai bộ lọc Laplacian quá khác nhau có dạng như trên hình

0	-1	0
-1	4	-1
0	-1	0

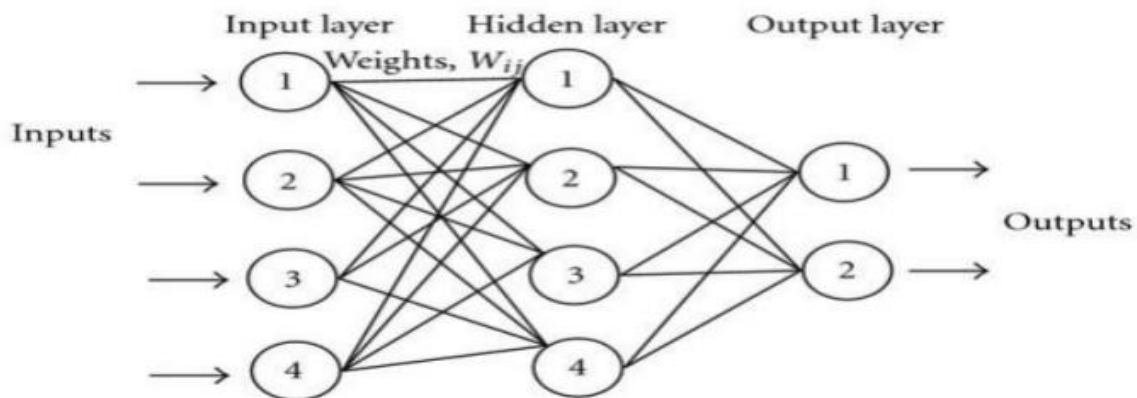
-1	-1	-1
-1	8	-1
-1	-1	-1

Hình 2.6 Toán tử Laplacian

2.3 Giải thuật Mạng nơ-ron nhân tạo (Artificial Neural Network – ANN)

Mạng nơ-ron nhân tạo là sự mô phỏng chức năng của hệ thần kinh con người với vô số các nơ-ron được liên kết và truyền thông với nhau. Giống như mạng nơ-ron của não người, ANN học và lưu những kinh nghiệm và sử dụng trong những tình huống phù hợp. Mạng nơ-ron đã áp dụng thành công trong nhiều vấn đề liên quan đến dự báo, phân loại và điều khiển thuộc nhiều lĩnh vực khác nhau, như tài chính, y tế, địa chất và vật lý. Ví dụ như khả năng nhận dạng mặt người, dự báo thời tiết và thiên tai, tự động điều khiển hệ thống lái tàu, hệ thống dự báo sự cố, v.v.

Kiến trúc chung của một mạng nơ-ron nhân tạo gồm 3 thành phần đó là: input layer, hidden layer và output layer. Trong đó, hidden layer gồm các thần kinh nhận dữ liệu input từ các neural ở lớp trước đó và chuyển đổi các input này cho các lớp xử lý tiếp theo. Trong một ANN có thể có nhiều lớp ẩn.



Hình 2.7 Mạng nơ-ron nhân tạo ba lớp

Quá trình xử lý thông tin của một mạng nơ-ron nhân tạo liên quan đến:

(1) Lớp Inputs: Dữ liệu nhập tương ứng các thuộc tính của dữ liệu.

(2) Lớp Outputs: Kết quả là một giải pháp cho một vấn đề.

(3) Lớp Weights: Trọng số liên kết là thành phần quan trọng thể hiện mức độ quan trọng (độ mạnh) của dữ liệu nhập đối với quá trình xử lý thông tin (quá trình chuyển đổi dữ liệu từ lớp này sang lớp khác). Quá trình học của ANN thực ra là quá trình điều chỉnh các trọng số của các dữ liệu nhập để có được kết quả mong muốn. Trong đó :

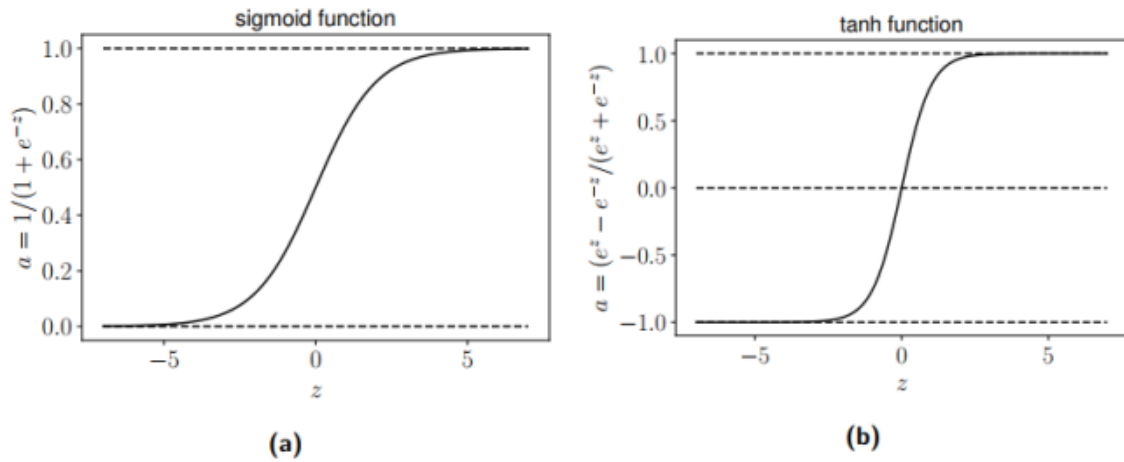
(1) Summation Function: hàm tổng tính tổng trọng số của tất cả các dữ liệu nhập đưa vào mỗi neural. Hàm tổng của một nơ-ron đối với n dữ liệu nhập được tính theo công thức sau:

$$Y = \sum_{i=1}^n X_i W_i$$

(2) Transfer Function: hàm tổng của một neural cho biết khả năng kích hoạt của neural đó, còn gọi là kích hoạt bên trong. Các nơ-ron này có thể sinh ra một kết quả hoặc không (nói cách khác rằng có thể kết quả của 1 neural có thể được chuyển đến lớp tiếp trong mạng nơ-ron hoặc không). Mối quan hệ giữa kích hoạt bên trong và kết quả được thể hiện bằng hàm chuyển đổi (Transfer Function). Việc lựa chọn hàm chuyển đổi có tác động lớn đến kết quả của một mạng nơ-ron nhân tạo. Một số hàm kích hoạt thường dùng là :

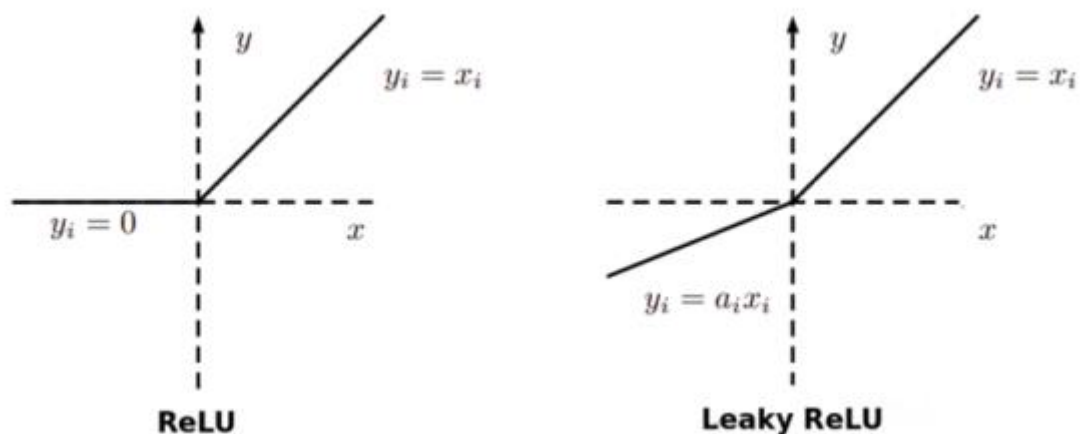
- Sigmoid: Hàm sigmoid có dạng $\text{sigmoid}(z) = \sigma(z) = \frac{1}{1+e^{-z}}$ với đồ thị tương ứng như trên hình. Nếu đầu vào lớn, hàm số sẽ cho giá trị gần bằng 1, còn với đầu vào nhỏ (rất âm) thì sẽ cho đầu ra gần với 0

- Tanh: Giá trị ngõ ra được chuyển về khoảng $[-1,1]$ khiến nó có tính chất tâm không thay vì chỉ dương như sigmoid. $\tanh(z) = \sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ với đồ thị như hình



Hình 2.8 Đồ thị hàm sigmoid (a) và hàm tanh (b)

- ReLU và Leaky ReLU : RELU lấy ngưỡng giá trị ở 0 (Thay thế các giá trị âm bằng 0) : $g(x) = \max(0, x)$.Leaky ReLU là biến thể của ReLU: $g(x) = \begin{cases} 0.01x, & x < 0 \\ x, & x \geq 0 \end{cases}$. Có thể suy ra chức năng của lớp ReLU là chuyển toàn bộ giá trị âm của ngõ vào (là kết quả lấy từ lớp Convolutional) thành giá trị 0, tạo ra tính phi tuyến cho mô hình để phù hợp với các mẫu trong thực tế không phải lúc nào cũng tuyến tính. Lớp ReLU làm biến đổi ngõ vào là xuất thẳng đến ngõ ra do đó không có trọng số được học trong lớp ReLU. Hình biểu thị dạng đồ thị của hai hàm kích hoạt này.



Hình 2.9 Đồ thị hàm ReLU và Leaky ReLU

(3) Kết quả xử lý tại các nơ-ron đôi khi rất lớn, vì vậy hàm chuyển đổi được sử dụng để xử lý kết quả này trước khi chuyển đến lớp tiếp theo. Đôi khi thay vì sử dụng hàm chuyển đổi người ta sử dụng giá trị ngưỡng để kiểm soát các kết quả của các nơ-ron tại một lớp nào đó trước khi chuyển các kết quả này đến các lớp tiếp theo. Nếu kết quả của một nơ-ron nào đó nhỏ hơn giá trị ngưỡng thì nó sẽ không được chuyển đến lớp tiếp theo.

2.4 Mạng nơ-ron tích chập

Mạng nơ-ron chuyển đổi hay mạng nơ-ron tích chập (CNN) là một trong những mô hình học sâu tiên tiến áp dụng nhiều trong các bài toán nhận dạng đối tượng trong hình ảnh hoặc phim ảnh, đồng thời ứng dụng để xây dựng những hệ thống nhận dạng thông minh với độ chính xác cao hiện nay. Những hệ thống xử lý ảnh của Facebook, Google hay Amazon đã áp dụng mô hình này để nhận diện khuôn mặt con người, phát triển xe ô tô tự hành hay thiết bị bay giao hàng tự động. Mạng nơ-ron tích chập đã giải quyết được hai bài toán là phân loại vật thể (Object Classification) và bài toán phát hiện vật thể (Object Recognition). Bài toán phân loại vật thể giúp ta phân loại được vật thể đầu vào thuộc loại nào, còn bài toán phát hiện vật thể giúp ta xác định vị trí của vật thể đầu vào. Hai bài toán trên thường được gộp chung lại làm bài toán nhận diện vật thể (Object Detection) vừa có thể phát hiện, phân loại và xác định vị trí của vật thể ấy. Đây là điều đặc biệt mà các bài toán xử lý ảnh cổ điển khó có thể đáp ứng được. Đối với những bài toán xử lý ảnh thông thường thì không nên sử dụng mạng tích chập vì độ phức tạp không cần thiết của nó.

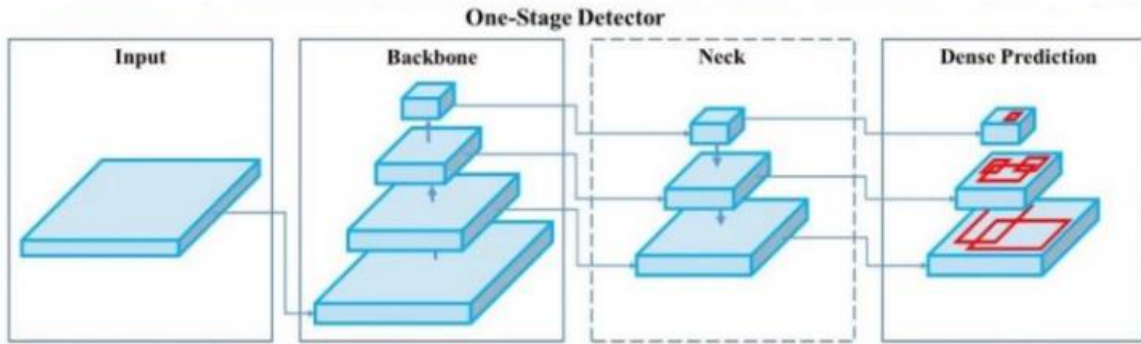
Mô hình ANN từ chương trước bao gồm 3 thành phần đó là: input layer, hidden layer và output layer. Mỗi hidden layer được gọi là lớp kết nối đầy đủ (fully connected layer), nếu mỗi nút trong hidden layer được kết nối với tất cả các nút trong layer trước. Cả mô hình được gọi là mạng nơ-ron kết nối đầy đủ (*fully connected neural network - FCNN*)

Khi sử dụng FCNN để xử lý ảnh thì ảnh màu 64×64 được biểu diễn dưới dạng 1 tensor $64 \times 64 \times 3$. Để biểu thị hết nội dung của bức ảnh thì cần truyền vào input layer tất cả các điểm ảnh ($64 \times 64 \times 3 = 12.288$), nghĩa là input layer bao gồm 12288 nút. Giả sử số lượng nút trong hidden layer đầu tiên là 1000. Số lượng trọng số w giữa input layer và hidden layer đầu tiên là $12.288 \times 1.000 = 12.288.000$, số lượng bias là 1.000 nên tổng số tham số là 12.289.000. Khi có thêm các hidden layer và kích thước hình ảnh tăng thì

số lượng tham số tăng rất nhanh, dẫn đến độ phức tạp tính toán rất cao. Chỉ với một lớp Hidden Layer và kích thước ảnh đầu vào khá nhỏ so với thực tế mà số lượng thông số đã lên đến hàng triệu, điều này khiến cho việc tính toán của máy tính cho mô hình toán rất nhiều công sức nhưng hiệu quả mang lại không cao. Do đó cần áp dụng bài toán với những đối tượng phức tạp tương ứng để thu lại hiệu quả mong muốn.

Một mô hình mạng CNN dùng trong bài toán nhận diện vật thể (Object Detection) sẽ gồm 2 phần chính và 1 phần tăng cường :

- Phần rút trích đặc trưng (backbone): trong phần này, mạng sẽ tiến hành tính toán rất nhiều phép tích chập (bao gồm lớp tích chập và hàm kích hoạt nếu cần thiết) hay phép hợp nhất (pooling) để phát hiện các đặc trưng chính của vật thể. Ở các mô hình Backbone hiện đại, phần này thường được xây dựng theo dạng kim tự tháp (nghĩa là độ phân giải giảm dần).
- Phần tăng cường (neck): Đối với các mạng hiện đại sẽ thường sử dụng phần này. Phần này sẽ khắc phục những hạn chế của phần Backbone – Khi xây dựng dạng kim tự tháp thì sẽ khiến cho các vật có kích thước bé sau khi tính tích chập lại bị mất ở các tầng có độ phân giải thấp hơn. Do đó phần tăng cường sẽ xây dựng kim tự tháp ngược và cũng thường được kết hợp (hoặc cộng) với tầng tương đương ở phần Backbone để giữ một số vật bị mất do quá trình tích chập, từ đó mô hình có thể học được nhiều đặc trưng hơn và cũng làm hạn chế việc làm mất đặc trưng của các đối tượng có kích thước nhỏ.
- Phần phân lớp và hồi quy boundingbox (head): Phần này thường nằm ở cuối của mô hình với nhiệm vụ phân lớp cho vật thể, tính xác suất vật thể, hoặc cũng có thể tính toán tọa độ hai điểm bên trái trên cùng và bên phải dưới cùng của vị trí vật thể đó để từ đó có thể vừa phân loại vật thể và vị trí của vật thể.



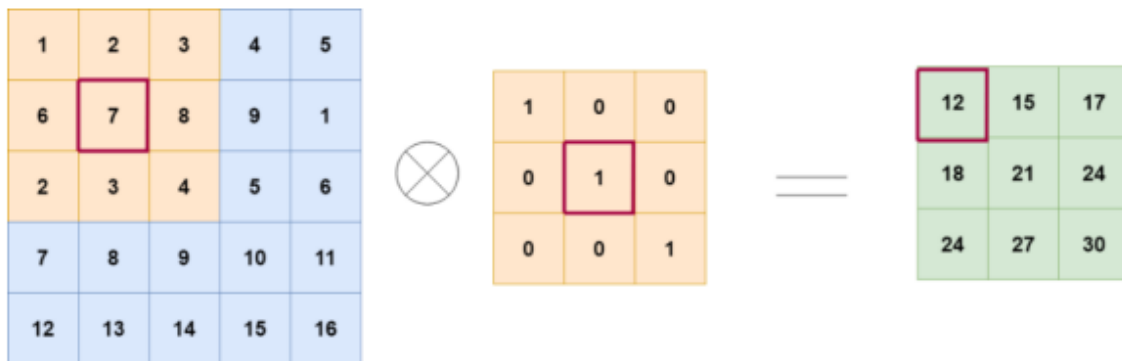
Hình 2.10 Cấu trúc của một mạng CNN hiện đại

Ngoài ra trong mạng CNN còn một số phép toán (tính tích chập, padding, stride) và các lớp (Lớp tích chập, lớp Pooling, lớp FullyConnected), hàm mất mát, cách mô hình hội tụ tại điểm mà hàm mất mát đạt giá trị thấp nhất, cũng như là cách mô hình cập nhật thông số như thế nào sẽ được đề cập dưới đây.

2.4.1 Phép tích chập

Phép tính tích chập thường được sử dụng chủ yếu để xử lý ảnh. Trong đó việc trích xuất đặc trưng là phép tính tích chập với dữ liệu hai chiều và rời rạc. Với ảnh đầu vào ta sẽ được qua một bộ lọc (kernel) với tâm bộ lọc sẽ là điểm trượt trên toàn bộ ảnh để trích xuất đặc trưng. Bộ lọc có kích thước là $F \times F$ trong đó F là các số lẻ. Sở dĩ F là số lẻ sẽ tiện cho việc phân biệt đặc trưng và dễ xác định tâm của bộ lọc đó. Kí hiệu của phép tính tích chập là \otimes , kí hiệu là $Y = X \otimes W$, trong đó Y là ngõ ra, X là ngõ vào và W là bộ lọc.

Hình là mô tả quá trình tích chập một ảnh xám đầu vào là X với kích thước là 5×5 với bộ lọc W kích thước 3×3 sẽ cho ra ảnh đầu ra Y với kích thước 3×3 . Bộ lọc sẽ được đặt như trên hình chồng lên ảnh đầu vào, sau đó lấy tích từng phần tử tương ứng giữa X với W với nhau rồi cộng lại để điền vào ảnh đầu ra, sau đó bộ lọc sẽ trượt và cứ lặp lại cho đến khi trượt hết ảnh, ta sẽ thu được ảnh ngõ ra Y .

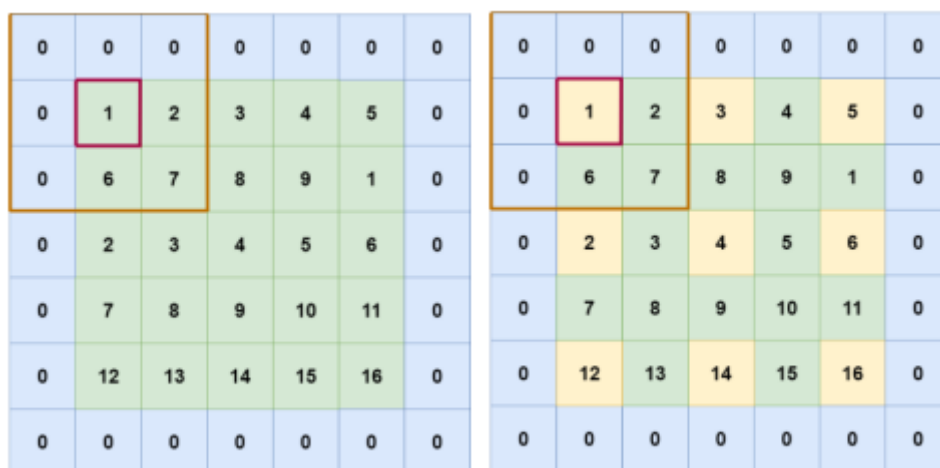


Hình 2.11 Phép tính tích chập

Ta thấy ảnh đầu ra Y có kích thước giảm đi so với ảnh đầu vào của X, điều này là một bất lợi và cần khắc phục, ở phần tiếp theo sẽ đề cập về toán tử Padding sẽ giải quyết bất lợi này.

2.4.2 Padding và Stride

Như đã thấy ở trên, sau khi thực hiện phép tính tích chập thì kích thước của ảnh đầu ra Y sẽ nhỏ hơn ảnh đầu vào X, điều này đôi khi là bất lợi. Do đó đã sinh ra phép Padding để khắc phục điều này. Với mong muốn thu được ảnh Y với kích thước như ảnh đầu vào X, ta sẽ thêm ngoài viền của X những giá trị 0 như trên hình 2.13 (trái). Cụ thể trên hình này tương ứng với padding = 1 là những ô vuông màu xanh ngoài viền.



Hình 2.12 Phép tính với padding = 1 và stride = 2

Hình (phải) là ảnh có padding = 1 và Stride = 2. Thì phép tính Stride ở đây hiểu đơn giản là phép trượt với k ô, ở ví dụ này là k = 2, các ô màu cam trên hình thể hiện là các tâm của bộ lọc 3x3 sau khi đã trượt hết ảnh để lại. Và phép tính Stride thường dùng

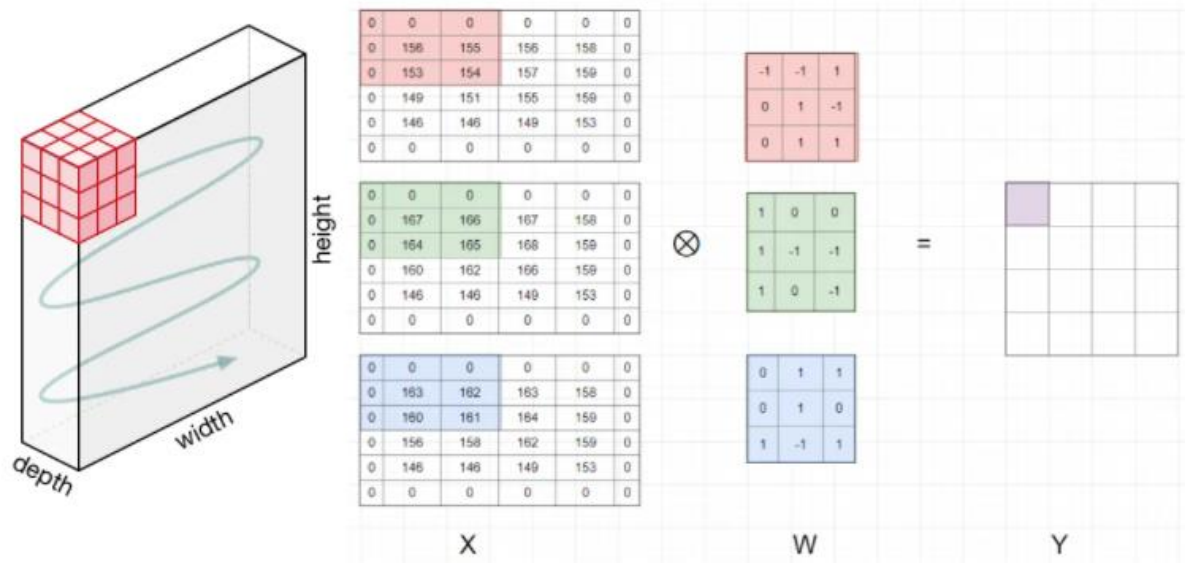
để giảm kích thước ảnh sau khi thực hiện phép tính tích chập. Công thức tổng quát cho phép tính tích chập với ảnh đầu vào X kích thước là $M \times N$ với bộ lọc W với kích thước $F \times F$, $\text{Stride} = S$ và $\text{Padding} = P$ sẽ cho ra ma trận Y với kích thước :

$$\left(\frac{M - F + 2P}{S} + 1\right) \left(\frac{N - F + 2P}{S} + 1\right)$$

2.4.3 Convolution Layer

Trong bài toán xử lý ảnh cổ điển thì các bộ lọc sẽ được chọn trước và cố định như bộ lọc gaussian và bộ lọc sobel (đã đề cập ở phần xử lý ảnh cổ điển). Do đó việc học thêm đặc trưng từ ảnh đầu vào là vô cùng khó nếu đặc trưng ấy là các đường cong. Do đó các bộ lọc thông thường sẽ khởi tạo trước theo một quy chuẩn nào đó để việc hội tụ diễn ra nhanh hơn và sau đó các bộ lọc này sẽ được cập nhật liên tục thông qua quá trình forward propagation và back propagation. Lớp tích chập (Convolutional Layer) là lớp chính thực hiện hầu hết các phép tính toán trích xuất đặc trưng của một mạng neural tích chập. Trong lớp này có chứa những nhiều tham số có khả năng học và cập nhật theo thời gian. Mỗi bộ lọc là một vùng không gian nhỏ, được đại diện bằng một ma trận số thực. Trong suốt quá trình huấn luyện, các bộ lọc này sẽ trượt dọc theo kích thước của ảnh đầu vào, lần lượt quét từ trái sang phải và từ trên xuống dưới và trích xuất ra được những đặc trưng trong những vùng ấy. Những đặc trưng đã học được sẽ được lưu lại vào một ma trận gọi là feature map.

Có nhiều bộ lọc khác nhau, do đó sẽ học được nhiều đặc trưng khác nhau của ảnh. Trong lớp tích chập ta sẽ dùng nhiều bộ lọc hơn để có thể học được nhiều thuộc tính của ảnh hơn. Vì mỗi bộ lọc sẽ cho đầu ra là một ma trận nên với số lượng bộ lọc bao nhiêu sẽ thu được lượng ngõ ra bấy nhiêu, thông số này được gọi là chiều sâu của đặc trưng. Kết quả đặc trưng sau khi thu được sẽ được đưa vào hàm kích hoạt để đưa ra kết quả cuối cùng và cũng chính là đầu vào kế tiếp của lớp tích chập tiếp theo. Quá trình tính toán liên tục và nối tiếp như vậy là một phần của quá trình forward propagation. Việc này lặp lại cho đến khi quá trình backbone và neck kết thúc.



Hình 2.13 Tính tích chập với hình ảnh màu (RGB)

Đầu ra của lớp tích chập đầu tiên sẽ là đầu vào của lớp tích chập kế tiếp. Một cách tổng quát, với ảnh đầu vào là ảnh có kích thước $H \times W \times D$, kích thước bộ lọc là $F \times F \times D$ (bộ lọc phải có chiều sâu giống như ảnh đầu vào), Stride: S và Padding: P . Lớp tích chập này được áp dụng cho K bộ lọc thì đầu ra của layer là tensor 3 chiều có kích thước:

$$\left(\frac{H - F + 2P}{S} + 1\right) \left(\frac{W - F + 2P}{S} + 1\right) \cdot K$$

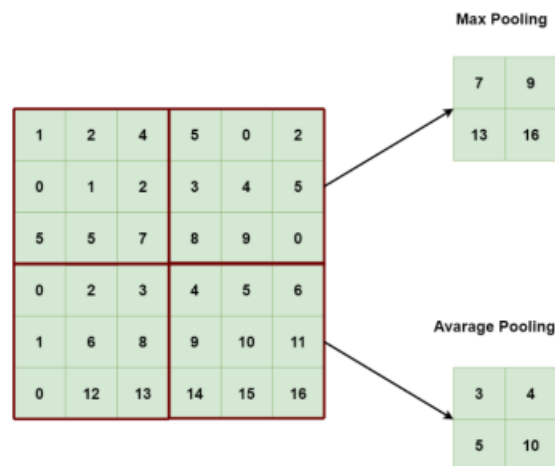
Output của convolutional layer sẽ qua hàm activation function trước khi trở thành input của convolutional layer tiếp theo.

Tổng số parameter của layer: Mỗi kernel có kích thước $F \times F \times D$ và có 1 hệ số bias, nên tổng parameter của 1 kernel là $F \times F \times D + 1$. Mà convolutional layer áp dụng K kernel \Rightarrow Tổng số parameter trong layer này là $K * (F \times F \times D + 1)$.

2.4.4 Pooling Layer

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model. . Có 2 loại lớp pooling thường gặp là : Max pooling và Average pooling.

Trong đó Max pooling là trích xuất đặc trưng cao nhất trong khu vực mà bộ lọc quét qua, còn Average Pooling thì lấy trung bình các giá trị trong khu vực mà bộ lọc quét qua. Các đầu ra của lớp này sẽ là đầu vào của lớp kế tiếp mà không cần phải đi qua hàm kích hoạt như ở lớp tích chập. Vì ở lớp này chỉ là giảm kích thước của thông số, chỉ giữ lại các thông số có ý nghĩa cao, nên không ảnh hưởng đến tốc độ tính toán của mô hình. Lớp Pooling vừa giảm thiểu số phép toán mô hình phải tính nên giúp tăng tốc độ xử lý của mô hình.

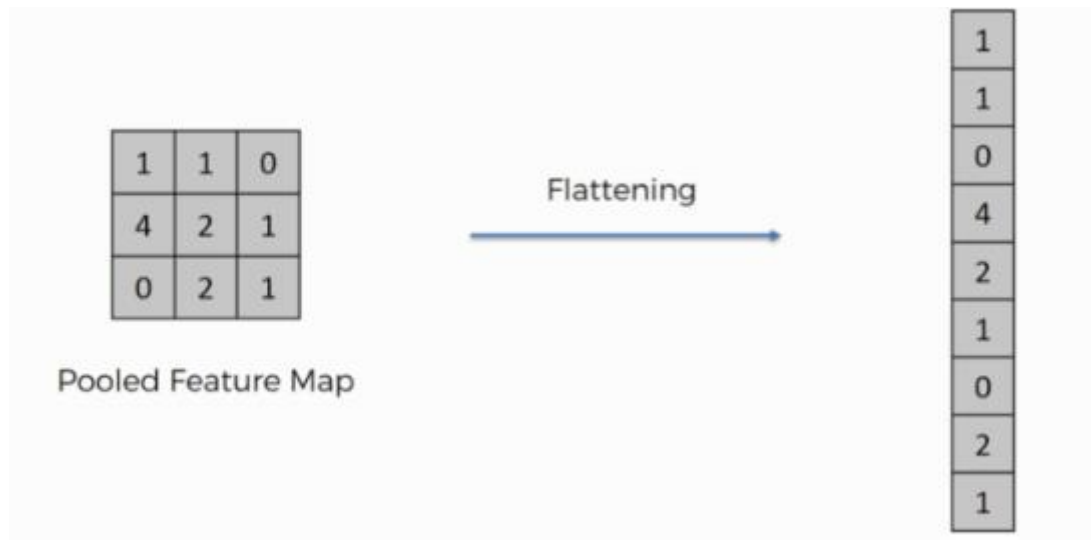


Hình 2.14 Các dạng Pooling

Trong một số model người ta dùng convolutional layer với stride > 1 để giảm kích thước dữ liệu thay cho pooling layer.

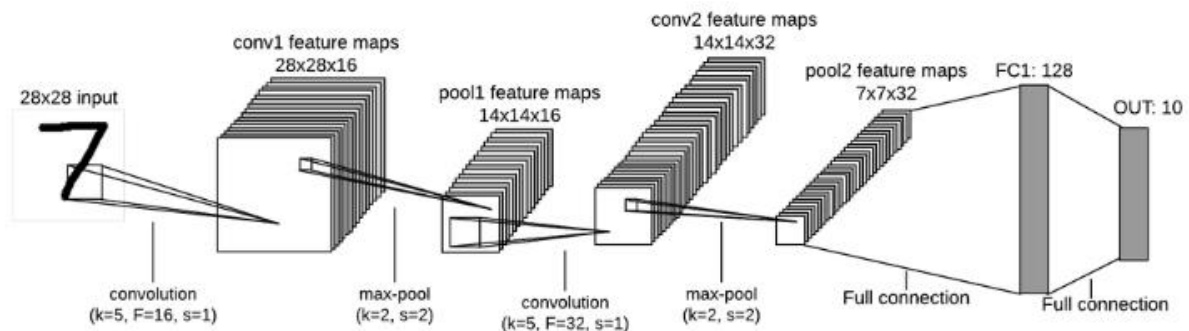
2.4.5 Fully Connected Layer

Lớp này thường được sử dụng cho các mạng con ở lớp head để phục vụ cho việc phân loại cũng như tìm vị trí boundingbox của vật thể. Sau khi trải qua rất nhiều lớp tích chập và lớp Pooling thì đặc trưng của vật thể sẽ được cô đọng trong những vector 3 chiều có kích thước là $H \times W \times D$, nhưng sau đó lại được chuyển về 1 vector chiều có kích thước là $(H \times W \times D)$, quá trình này được gọi là làm phẳng (Flatten). Sau đó, mỗi điểm của vector sẽ được liên kết với toàn bộ output của model giống như một lớp của mạng Neural Network.



Hình 2.15 Flattening

2.4.6 Mô hình convolutional neural network



Hình 2.16 Mô hình CNN với ảnh đầu vào

Input image -> Convolutional layer (Conv) + Pooling layer (Pool) -> Fully connected layer (FC) -> Output.

2.4.7 Softmax Function

Sau khi mạng CNN được học qua các lớp phía trước, ta thu được một vector đặc trưng như trên hình 2.3.6. Việc cần làm tiếp theo là dựa vào vector đặc trưng này để phân loại vào các lớp theo yêu cầu bài toán. Chúng ta cần một mô hình xác suất so cho với mỗi đầu vào x , sẽ tính được đầu ra thể hiện được xác suất để x rơi vào lớp thứ i . Sử dụng hàm softmax ta tính được điều này. Hàm softmax nhận đầu vào là một vector và đầu ra là 1 vector có cùng số chiều $a(z): \mathbb{R}^n \rightarrow \mathbb{R}^n$. Công thức cho hàm như sau:

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)}, \forall i = 1, 2, \dots, C$$

Nhận xét :

- Giá trị $z_i = w_i^T x$ càng lớn thì xác suất dữ liệu rơi vào lớp i càng cao.
- Các a_i lớn hơn 0 và có tổng bằng 1.

Softmax đặc biệt được sử dụng nhiều làm hàm kích hoạt ở lớp output trong bài toán phân loại qua mạng CNN. Những lớp phía trước có thể được coi như một bộ Feature Extractor (Trích xuất đặc trưng) , lớp cuối cùng của CNN cho bài toán phân loại thường là Softmax Regression .

2.4.8 Loss Function

Sau khi trả qua quá trình forward propagation (là quá trình tính toán nối tiếp giữa các lớp tích chập, lớp Pooling, lớp FullyConnected và Softmax) thì ta sẽ thu được một giá trị dự đoán \hat{y} thường cách biệt rất xa so với giá trị thực tế y . Như vậy, mong muốn của chúng ta là làm sao cho \hat{y} tiến đến y . Để có thể đạt được điều đó, người ta đề ra một hàm số thể hiện mối quan hệ giữa dự đoán \hat{y} và giá trị thực tế y để từ đó điều chỉnh lại các tham số. Hàm đó gọi là hàm mất mát (Loss function)

$$L(\hat{y}, y) = |\hat{y} - y|$$

Để tìm được hao phí ít nhất ta cần đạo hàm được hàm mất mát để tìm ra giá trị cực tiểu. Tuy nhiên đối với hàm có trị tuyệt đối thì không có đạo hàm tại 0. Do đó hàm sẽ được biến đổi thành hàm bình phương cực tiểu để cho ra đạo hàm đẹp hơn.

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

Đây là hàm cơ bản cho các bài toán hồi quy tuyến tính. Còn đối với các bài toán phân loại, hàm mất mát thường dùng là hàm Cross Entropy

$$L(\hat{y}, y) = -(\hat{y} \log(y) + (1 - \hat{y}) \log(1 - y))$$

Còn đối với bài toán phát hiện đối tượng (Object Detection) thì có hàm Yolo Loss Function sẽ được đề cập ở phần sau.

2.4.9 Gradient Descent

Sau khi đã thực hiện xong quá trình forward propagation và xác định được hàm mất mát, ta sẽ tiến hành đi cập nhật các thông số. Quá trình này gọi là Back Propagation. Với mong muốn hàm mất mát $L(x)$ đạt giá trị nhỏ nhất, đồng nghĩa với việc phải tìm vị trí x_0 mà $L'(x_0) = 0$ và giá trị nhỏ nhất của hàm mất mát là $L'(x_0) = 0$, vị trí này được là điểm cực tiểu toàn cục (global minimum), ngoài ra còn nhiều vị trí khác làm cho đạo hàm của hàm mất mát bằng 0 gọi là cực trị địa phương (local minimum).

Có hai hướng để tìm cực trị toàn cục. Thứ nhất, bằng một cách nào đó ta có thể tìm được toàn bộ (hữu hạn) các điểm cực tiểu, ta chỉ cần thay lần lượt từng điểm cực tiểu địa phương này vào và chọn ra vị trí mà hàm mất mát đạt giá trị nhỏ nhất. Tuy nhiên, trong hầu hết các trường hợp, việc giải phương trình đạo hàm bằng 0 là bất khả thi. Nguyên nhân có thể đến từ sự phức tạp của dạng hàm số sau khi đạo hàm. Hướng tiếp cận thứ 2, xuất phát từ một điểm mà chúng ta coi là gần với nghiệm của bài toán, sau đó dùng phép lặp để tiến dần đến điểm cần tìm, tức là khi đạo hàm gần với 0. Phương pháp thứ 2 được sử dụng rất phổ biến và được gọi là Gradient Descent.

Xét bài toán Gradient Descent cho hàm một biến với hàm mất mát là $L(x)$. Gọi x_t là điểm ta tìm được sau vòng lặp thứ t và x^* là vị trí cực trị toàn cục cần tìm. Ta cần tìm một thuật toán để đưa x_t về càng gần x^* càng tốt. Có hai quan sát nữa:

- Nếu đạo hàm của hàm số tại x_t : $L'(x_t) > 0$ thì x_t này bên phải so với x^* (và ngược lại). Để điểm tiếp theo là x_{t+1} gần với x^* , ta cần di chuyển x_t về phía bên trái, tức về phía âm.
- Khi x_t càng xa x^* về phía bên phải thì $L'(x_t)$ càng lớn hơn 0 (và ngược lại). Vậy, lượng di chuyển tỉ lệ thuận với $-L'(x_t)$

Do đó ta được công thức cập nhật là: $x_{t+1} = x_t - \eta L'(x_t)$, trong đó η là tốc độ học (learning rate) là một số dương thường từ 0.0001 đến 0.01. Nếu hệ số tốc độ học nhỏ thì quá trình cập nhật (học) sẽ lâu hội tụ và nếu lớn thì quá trình cập nhật sẽ khó hội tụ, do đó cần lựa chọn hệ số này để việc học được tối ưu về mặt thời gian và có khả năng hội tụ.

Trong thực tế, tốc độ học ban đầu được đặt lớn để hàm nhanh về điểm đích, nếu sau một vài lần lặp mà hàm mất mát không thay đổi nhiều, thì hệ số hội tụ sẽ được giảm xuống. Và trong các bài toán thực tế sẽ là một vector biến θ thay vì một biến x , lúc này

ta sẽ có công thức tương tự là: $\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} J(\theta_t)$. Với θ_{t+1} là vector tham số của mô hình được cập nhật sau lần lặp thứ $t+1$, θ_t là vector tham số của mô hình ở lần lặp thứ t , $\nabla_{\theta_t} J(\theta_t)$ là đạo hàm của hàm số mất mát J tại điểm θ_t . Nếu ta xem điểm cần cập nhật là một viên bi và đạo hàm tại vị trí đó là độ cao của dốc, thì ta nhận thấy rằng khi độ dốc càng sâu thì viên bi đi càng nhanh. Dựa vào đặc tính vật lý đó, người ta đưa ra một đại lượng Momentum để giúp các điểm cập nhật có thể đủ đà để vượt qua các điểm cực tiểu địa phương và tiến tới được điểm cực tiểu toàn cục (bởi vì khi dùng Gradient Descent rất dễ bị rơi vào các điểm cực trị địa phương mà không hội tụ được). Được cho bởi công thức: $\theta_{t+1} = \theta_t - v_t$. Và hàm v_t được cho bởi công thức: $v_t = \gamma v_{t-1} - \eta \nabla_{\theta_t} J(\theta_t)$. Trong đó γ được gọi là momentum và thường có giá trị là 0.9, và ban đầu $v_0 = 0$.

Ngoài ra, Gradient Descent thường có một số biến thể khác nhau như Batch Gradient Descent, Stochastic Gradient Descent, Mini-batch Gradient Descent. Quá trình cập nhật các thông số khi dùng Gradient Descent được gọi là BackPropagation (sẽ được nhắc trong phần huấn luyện mô hình)

2.5 Thuật toán You Only Look Once (YOLO)

2.5.1 Nhận xét, đánh giá và lý do lựa chọn

Việc áp dụng công nghệ mạng nơ-ron tích chập trong trí tuệ nhân tạo (AI) để nhận dạng và phân loại các đối tượng trong lĩnh vực thị giác máy tính chủ yếu tập trung 3 kỹ thuật chính đó là:

- Thuật toán áp dụng mạng nơ-ron tích chập CNN và các biến thể của mạng nơ-ron tích chập, bao gồm R-CNN, Fast R-CNN và Faster R – CNN.
- Thuật toán Single Shot Object Detectors (SSD)
- Thuật toán You Only Look Once (YOLO).

Đối với R-CNN là một trong những kỹ thuật dò tìm đối tượng dựa trên giải thuật học sâu đầu tiên và là một ví dụ về kỹ thuật dò tìm đối tượng theo phương pháp hai giai đoạn cụ thể đó là:

Giai đoạn 1: R-CNN bản đầu tiên chủ yếu hỗ trợ tính năng phát hiện đối tượng và phân đoạn ngữ nghĩa (2013 của Girshick et al). Bản nâng cấp sau đó xử lý nhằm phát hiện đối tượng theo yêu cầu thông qua thuật toán tìm kiếm chọn lọc (hoặc tương đương)

cụ thể để xác định và đề xuất các hộp giới hạn có thể chứa các đối tượng. Từ đó, thực hiện thao tác chuyển các vùng phát hiện vào CNN để phân loại thông qua kỹ thuật dò tìm đối tượng dựa trên giải thuật học sâu.

Vì vậy, Đối với bản đầu tiên của thuật toán R-CNN tiêu chuẩn là rất chậm và đối tượng dò tìm chưa chính xác cao.

Giai đoạn 2: Năm 2015, tác giả Girshick et al cho xuất bản một bài báo có tựa đề Fast R- CNN. Từ đó, thuật toán Fast R-CNN đã tạo ra những cải tiến đáng kể so với phiên bản R-CNN ban đầu. Cụ thể là tăng độ chính xác và giảm thời gian cần thiết để thực hiện chuyển tiếp. Tuy nhiên, thuật toán vẫn cần phải các thuật toán khác xử lý khu vực bên ngoài đối tượng.

Do đó, Girshick et al và cộng sự cũng đã đề xuất thuật toán Faster R – CNN đây là thuật toán tìm kiếm vị trí của vật thể trong ảnh và xử lý theo hướng phát hiện đối tượng theo thời gian thực. Từ đó, R-CNN đã trở thành một trình phát hiện đối tượng bằng giải thuật học sâu và loại bỏ yêu cầu tìm kiếm có chọn lọc và thay vào đó dựa vào mô hình mạng đề xuất khu vực đó là:

- Tích chập hoàn toàn.
- Có thể dự đoán các hộp giới hạn đối tượng và điểm số của đối tượng xác định (nghĩa là điểm số định lượng khả năng đó trong khu vực của một hình ảnh). Đầu ra của mô hình mạng đề xuất khu vực tích hợp, được chuyển sang thành phần R-CNN để phân loại và ghi nhãn cuối cùng.

Theo đánh giá, R-CNN là thuật toán xử lý có độ chính xác rất cao, nhưng ngược lại thì tốc độ xử lý chậm (chỉ đạt 5 FPS trên GPU). Trong khi SSD là giải thuật có thời gian thực hiện ngắn nhất, cho kết quả có độ chính xác thấp nhất.

Ngược lại Yolo vừa đảm bảo về tốc độ xử lý vừa đảm bảo về độ chính xác và đáp ứng nhu cầu ứng dụng thực tế. Từ đó nên em quyết định lựa chọn mô hình yolov3 để giải quyết bài toán đã đặt ra trong phần mục tiêu đề tài.

2.5.2 Mô hình mạng YOLO tổng quát

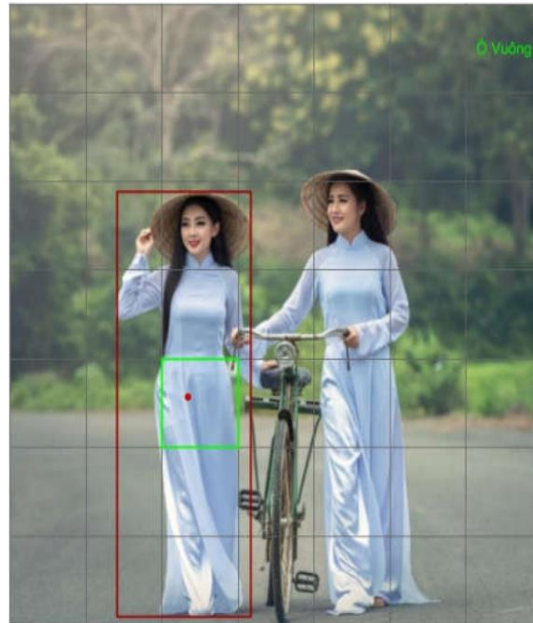
YOLO là một mô hình CNN để phát hiện đối tượng với ưu điểm nổi trội là nhanh hơn nhiều so với những mô hình trước, thậm chí, có thể vận hành tốt trên những thiết bị IoT. Có 3 phiên bản YOLO v1, YOLO v2 và YOLO v3, trong đó phiên bản cuối

cùng chạy nhanh hơn nhưng phức tạp và khó sử dụng hơn. Đầu vào mô hình là một bức ảnh, bài toán phát hiện đối tượng không chỉ phân loại được đối tượng trên bức ảnh mà còn định vị được vị trí của đối tượng đó. Phát hiện đối tượng trong thời gian thực có nhiều ứng dụng, ví dụ như hệ thống theo dõi đối tượng có thể giúp hoạt động giám sát, hoặc hệ thống xe ô tô tự lái phải xác định được người đi đường ở đâu để đưa ra quyết định di chuyển tiếp theo. Một trong những ưu điểm chính của YOLO là chỉ sử dụng thông tin toàn bộ bức ảnh một lần và dự đoán toàn bộ hình chữ nhật chứa các đối tượng. Mô hình được xây dựng theo kiểu end-to-end nên được huấn luyện hoàn toàn bằng gradient descent .

Hệ thống lưới (grid system): ảnh được chia thành ma trận ô vuông 7×7 , mỗi ô vuông bao gồm một tập các thông tin mà mô hình phải dự đoán. Hình . là ví dụ.

Đối tượng duy nhất mà ô vuông đó chứa. Tâm của đối tượng cần xác định nằm trong ô vuông nào thì ô vuông đó chứa đối tượng đó. Ví dụ tâm của cô gái nằm trong ô vuông màu xanh, do đó mô hình phải dự đoán được nhãn của ô vuông đó là cô gái. Lưu ý cho dù phần ảnh cô gái có nằm ở ô vuông khác mà tâm không thuộc ô vuông đó thì vẫn không tính là chứa cô gái, ngoài ra, nếu có nhiều tâm nằm trong một ô vuông thì chúng ta vẫn chỉ gán một nhãn cho ô vuông đó thôi. Chính ràng buộc mỗi ô vuông chỉ chứa một đối tượng làm cho mô hình không thể phát hiện những đối tượng có tâm nằm cùng một ô vuông. Tuy nhiên, việc tăng kích thước ô vuông từ 7×7 lên kích thước lớn hơn có thể giúp phát hiện được nhiều đối tượng hơn. Ngoài ra, kích thước của ảnh đầu vào phải là bội số của kích thước ô vuông.

Mỗi ô vuông chịu trách nhiệm dự đoán 2 boundary box của đối tượng. Mỗi boundary box dự đoán có chứa đối tượng hay không và thông tin vị trí của boundary box gồm trung tâm boundary box của đối tượng và chiều dài, rộng của boundary box đó. Ví dụ ô vuông màu xanh cần dự đoán 2 boundary box chứa cô gái như hình 31. Một điều cần lưu ý, việc cài đặt không dự đoán giá trị điểm ảnh mà cần phải chuẩn hóa kích thước ảnh về đoạn từ $[0,1]$ và dự đoán độ lệch của tâm đối tượng đến hình chữ nhật chứa đối tượng đó. Ví dụ, thay vì dự đoán vị trí điểm ảnh của điểm màu đỏ, thì cần dự đoán độ lệch a, b trong ô vuông chứa tâm đối tượng.



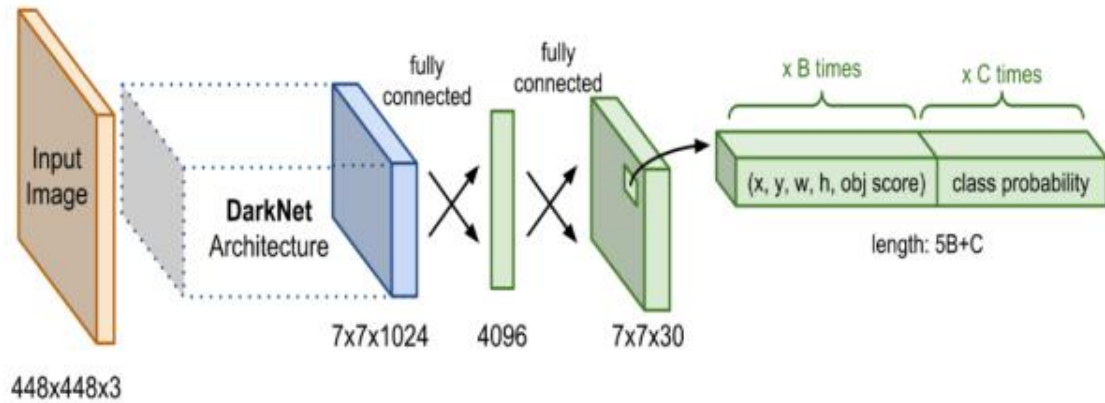
Hình 2.17 Chia ảnh thành các ma trận ô vuông

Như vậy, mỗi ô vuông có thể dự đoán các thông tin sau: ô vuông có chứa đối tượng nào hay không chứa đối tượng, độ lệch 2 hình chữ nhật chứa đối tượng so với ô vuông hiện tại và lớp của đối tượng. Mỗi ô vuông cần một vector có số chiều đủ lớn để dự đoán các thông tin.

Sử dụng CNN kết hợp YOLO: thông tin dự đoán đã được xác định đối với mỗi ô vuông, vấn đề còn lại là xây dựng mô hình CNN cho ra output với kích thước phù hợp, nghĩa là kích thước ô vuông \times kích thước ô vuông \times số chiều, ví dụ với kích thước ô vuông là 7×7 , mỗi ô vuông dự đoán 2 hình chữ nhật, và có 3 loại đối tượng thì output là $7 \times 7 \times 13$ từ mô hình CNN ($2 + 2 \times 4 + 3 = 13$).

YOLO sử dụng hồi quy tuyến tính (linear regression) để dự đoán các thông tin ở mỗi ô vuông. Do đó, layer cuối cùng không sử dụng bất kỳ hàm kích hoạt. Ví dụ, với ảnh input có kích thước 448×448 , mô hình CNN có 6 lớp max pooling với kích thước 2×2 sẽ giảm 64 lần kích thước ảnh xuống còn 7×7 ở output đầu ra. Đồng thời, thay vì sử dụng lớp kết nối đầy đủ ở các layer cuối cùng, có thể thay thế bằng lớp 1×1 convolution với 13 feature map để output cho ra $7 \times 7 \times 13$.

Sau khi tìm hiểu về nguyên lý hoạt động của mạng, ta sẽ đi sâu vào kiến trúc mạng yolo. Kiến trúc mạng yolo bao gồm: Nhiều lớp mạng Convolution chịu trách nhiệm trích xuất đặc trưng. Phần phía sau là những lớp mở rộng (Extra Layer) dùng để xác định được vị trí và phân lớp của vật thể.



Hình 2.18 Sơ đồ kiến trúc mạng YOLO

Thành phần Darknet Architecture được gọi là phần Backbone có tác dụng trích xuất đặc trưng. Output của phần Backbone sẽ là một feature map có kích thước 7x7x1024 sẽ được sử dụng làm đầu vào cho các Extra Layer có tác dụng dự đoán nhãn và tọa độ boundingbox của vật thể.

Trong YOLO version 3 tác giả áp dụng một mạng feature extractor là darknet-53. Mạng này gồm 53 convolutional layers kết nối liên tiếp, mỗi layer được theo sau bởi một batch normalization và một activation Leaky Relu. Để giảm kích thước của output sau mỗi convolution layer, tác giả down sample bằng các filter với kích thước là 2. Mạng này có tác dụng giảm thiểu số lượng tham số cho mô hình.

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

Hình 2.19 Kiến trúc mạng YOLOv3

Các bức ảnh khi được đưa vào mô hình sẽ được scale để về chung một kích thước phù hợp với input shape của mô hình và sau đó được gom lại thành batch đưa vào huấn luyện.

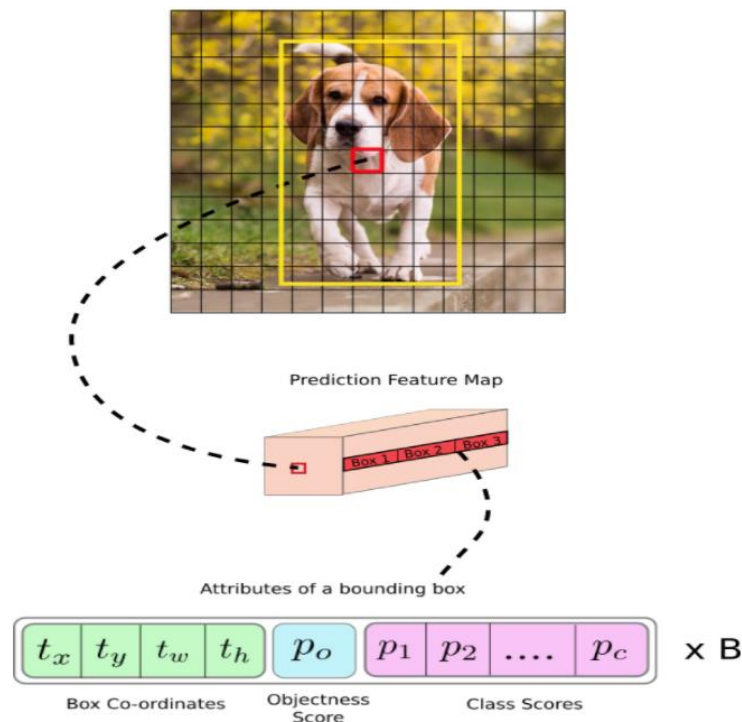
Các bức ảnh khi được đưa vào mô hình sẽ được tinh chỉnh kích thước để chung một kích thước phù hợp với đầu vào của mô hình đó là 416x416. Mỗi một đầu vào sẽ có một thiết kế các layers riêng phù hợp với kích thước đầu vào. Sau khi đi qua các lớp Convolutional thì kích thước giảm dần theo cấp số nhân là 2. Cuối cùng ta thu được một feature map có kích thước tương đối nhỏ để dự báo vật thể trên từng ô của feature map. Kích thước của feature map đối với mô hình Yolov3-tiny là 13x13 và 26x26 và có thêm 52x52 ở mô hình yolov3. Và đầu ra của mô hình Yolo sẽ là một vector như sau:

$$y^T = [p_0, \underbrace{\langle t_x, t_y, t_w, t_h \rangle}_{\text{bounding box}}, \underbrace{\langle p_1, p_2, \dots, p_C \rangle}_{\text{scores of } c \text{ classes}}]$$

Trong đó:

- p_0 là xác suất dự báo vật thể xuất hiện trong bounding box.
- $\langle t_x, t_y, t_w, t_h \rangle$ giúp xác định bounding box. Trong đó t_x, t_y là tọa độ tâm và t_w, t_h là kích thước rộng, dài của bounding box.
- $\langle p_1, p_2, \dots, p_C \rangle$ là véc tơ phân phối xác suất dự báo của các classes

Việc hiểu được output khá là quan trọng để chúng ta cấu hình được tham số chuẩn xác khi huấn luyện mô hình qua các mã nguồn mở như darknet. Như vậy output cho một ô của feature map sẽ được xác định theo công thức là: $(n_{class} + 5) * 3$, trong đó n là số lượng phân lớp của mô hình và 5 thông số gồm: $x, y, h, w, p_{objectness_score}$ lần lượt là tọa độ tâm và kích thước của đối tượng và thông số cho biết có hay không đối tượng trong ô đó và mỗi ô được định nghĩa bởi 3 anchor box.

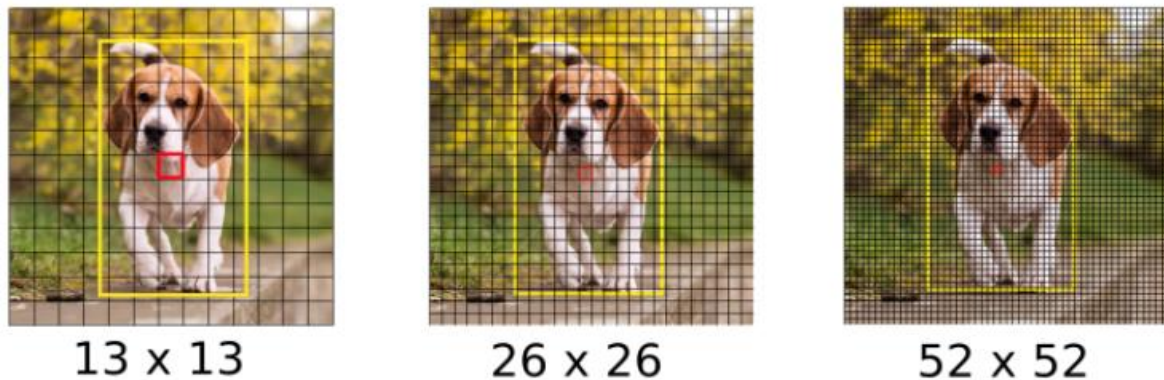


Hình 2.20 Kiến trúc một output của model YOLO

Hình ảnh gốc là một feature map kích thước 13x13. Trên mỗi một cell của feature map chúng ta lựa chọn ra 3 anchor boxes với kích thước khác nhau lần lượt là Box 1, Box 2, Box 3 sao cho tâm của các anchor boxes trùng với cell. Khi đó output của YOLO là một véc tơ concatenate của 3 bounding boxes. Các attributes của một bounding box được mô tả như dòng cuối cùng trong hình.

Cũng tương tự như SSD, YOLOv3 dự báo trên nhiều feature map. Những feature map ban đầu có kích thước nhỏ giúp dự báo được các object kích thước lớn. Những feature map sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo các vật thể kích thước nhỏ. Đối với mô hình yolov3 thì hỗ trợ được kích thước (13x13, 26x26, 52x52). Còn với mô hình yolov3-tiny thì chỉ hỗ trợ được kích thước (13x13 và 26x26) nên đây là một hạn chế nhận diện các vật thể các

kích thước nhỏ hơn, nhưng bù lại tốc độ của mô hình yolov3-tiny có thể gấp 3 hoặc 4 lần so với mô hình yolov3.



Hình 2.21 Các bản đồ đặc trưng của mạng YOLO

Trên mỗi một cell của các feature map chúng ta sẽ áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 feature map x 3 anchor box). Đồng thời trên một feature map hình vuông $S \times S$, mô hình YOLOv3 sinh ra một số lượng anchor box là: $S \times S \times 3$. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là: $(13 \times 13 + 26 \times 26 + 52 \times 52) = 10647$ (anchor box). Đây là một số lượng rất lớn và là nguyên nhân khiến quá trình huấn luyện mô hình YOLO vô cùng chậm bởi chúng ta cần dự báo đồng thời nhãn và bounding box trên đồng thời 10647 bounding boxes.

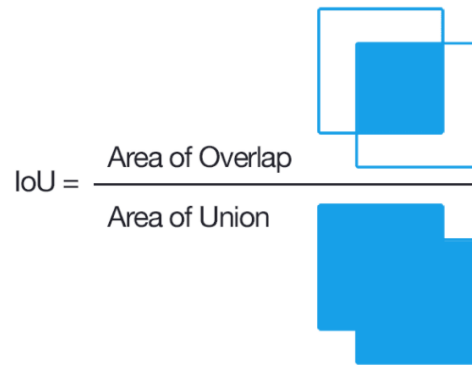
2.5.3 Độ chồng lấn (IOU)

Intersection over Union (IOU) là chỉ số đánh giá được sử dụng để đo độ chính xác của Object detector trên tập dữ liệu cụ thể.

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

Trong đó :

- Area of overlap là diện tích phần giao nhau giữa khung dự đoán và khung được gán nhãn của đối tượng (Khung đúng của đối tượng).
- Area of Union là diện tích phần hợp giữa khung dự đoán và khung được gán nhãn của đối tượng.



Hình 2.22 Phép tính IoU

Nếu một dự đoán có hệ số IOU > 0.5 thì được đánh giá là tốt.

2.5.4 Loss Function

YOLO sử dụng hàm sai số bình phương giữa dự đoán và kết quả thực để tính sai số cho mô hình. Cụ thể, sai số tổng là tổng của 3 sai số cục bộ sau:

- Sai số của dự đoán loại đối tượng, hay sai số phân loại (classification loss).
- Sai số của dự đoán tọa độ và chiều dài, rộng của boundary box, hay sai số vị trí (localization loss).
- Sai số của ô vuông có chứa đối tượng hay không có đối tượng, sai số lựa chọn (confidence loss).

Trong quá trình huấn luyện, mô hình nhìn vào những ô vuông có chứa đối tượng, tăng điểm phân loại lớp đúng của đối tượng đó lên. Tìm boundary box tốt nhất trong 2 hình chữ nhật được dự đoán, tăng điểm vị trí của boundary box đó lên, thay đổi thông tin boundary box để gần đúng với kết quả thực. Đối với những ô vuông không chứa đối tượng, giảm điểm lựa chọn và không quan tâm đến điểm phân loại và điểm vị trí của những ô vuông này.

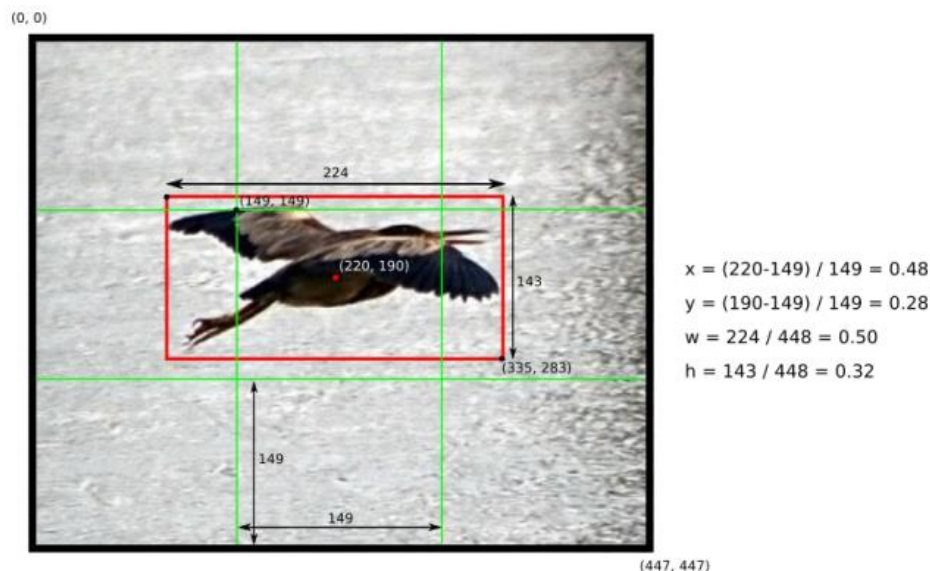
Classification loss: Độ mất mát của việc dự đoán loại nhãn của đối tượng, sai số phân loại chỉ được tính cho những ô vuông được gán giá trị là có đối tượng. Sai số phân loại tại những ô vuông đó được tính bằng sai số bình phương giữa giá trị được dự đoán và giá trị đúng của nó.

$$L_{classification} = \sum_{i=0}^{S^2} \mathbb{I}_{ij}^{obj} \sum_{c \in class} (p_i(c) - \hat{p}(c))^2$$

Trong đó:

- \mathbb{I}_i^{obj} : bằng 1 nếu ô vuông đang xét có đối tượng và ngược lại bằng 0.
- $\hat{p}(c)$: là xác suất có điều kiện của lớp c tại ô vuông tương ứng mà mô hình dự đoán.

Localization loss: Sai số vị trí dùng để tính giá trị sai số cho boundary box được dự đoán bao gồm offset x, y và width, height so với giá trị đúng. Giá trị sai số này không thể tính toán trực tiếp trên kích thước của ảnh mà cần chuẩn dưới kính thước ảnh về đoạn $[0,1]$ đối với tọa độ điểm tâm, và không dự đoán trực tiếp điểm tâm mà phải dự đoán giá trị lệch offset x, y so với ô vuông tương ứng. Việc chuẩn hóa kích thước ảnh và dự đoán offset làm cho mô hình nhanh hội tụ hơn so với việc dự đoán giá trị mặc định. Hãy cùng xét một ví dụ trên hình6, ta thấy kích thước ảnh đầu vào là 448x448 và $S = 3$, sau đó ta tiến hành chuẩn hóa kích thước của đối tượng theo kích thước ảnh, và chuẩn hóa kích thước của tâm đối tượng theo kích thước của ô vuông cơ sở ($\frac{480}{3} \approx 149; \frac{448}{3} \approx 149$). Tại mỗi ô có chứa đối tượng ta chọn 1 boundingbox có IOU (IOU giữa đối tượng với ô vuông cơ sở) tốt nhất, rồi sau đó tính độ mất mát.



Hình 2.23 Chuẩn hóa đối tượng

Giá trị hàm mất mát dự đoán tọa độ tâm (x,y) của boundingbox dự đoán và (\hat{x}, \hat{y}) là tọa độ tâm của boundingbox được gán nhãn, cho bởi công thức sau:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \parallel_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

Giá trị hàm mất mát dự đoán kích thước (w,h) của boudingbox dự đoán và (\hat{w}, \hat{h}) là kích thước của boudingbox được gắn nhãn, cho bởi công thức sau:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \parallel_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

Trong đó thông số λ_{coord} là một thông số thực nghiệm và tác giả của tờ báo đã chọn là 5.0, S^2 là số ô vuông sau khi chia ảnh và B là số lượng boudingbox cho đối tượng đang xét.

Confidence loss: Sai số lựa chọn thể hiện sai số giữa dự đoán boundary box chứa đối tượng so với giá trị đúng tại ô vuông tương ứng. Sai số này tính trên cả những ô vuông chứa đối tượng và không chứa đối tượng. Nhưng, sai số tại ô vuông chứa đối tượng quan trọng hơn là sai số tại ô vuông không chứa đối tượng, do vậy hệ số lambda được sử dụng để cân bằng.

$$L_{confidence} = \sum_{i=0}^{S^2} \sum_{j=0}^B \parallel_{ij}^{noobj} (C_i - \hat{C}_i)^2 + \lambda_{noobject} \sum_{i=0}^{S^2} \sum_{j=0}^B \parallel_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

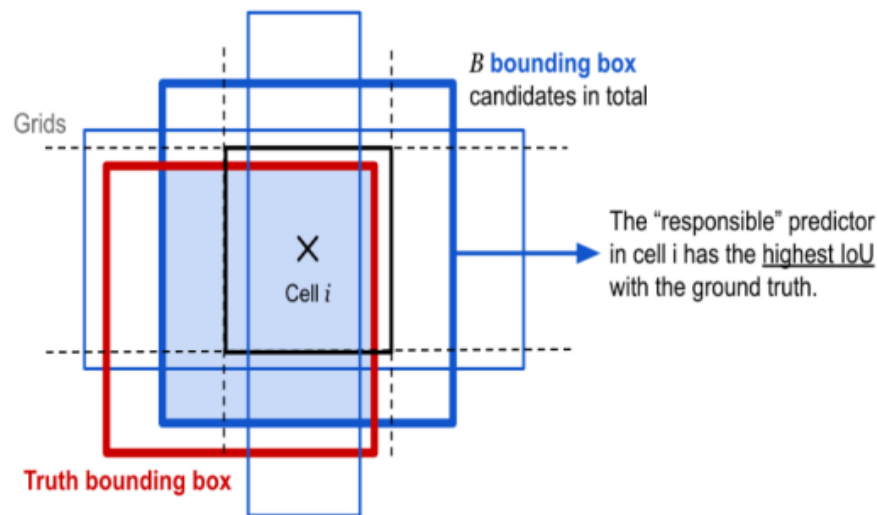
Tổng sai số của mô hình là tổng của 3 loại sai số:

$$L_{total} = L_{confidence} + L_{localization} + L_{confidence}$$

2.5.5 Anchor Box

Để tìm được boudingbox cho vật thể, mô hình sẽ cần các anchor box làm cơ sở ước lượng. Những anchor box này sẽ được xác định trước và sẽ bao quanh vật thể theo một cách tương đối chính xác. Thuật toán regression boudingbox (sẽ đề cập trong phần dự đoán boudingbox) sẽ tinh chỉnh lại anchor box để tạo ra boudingbox dự đoán cho vật thể. Trong một mô hình Yolo, mỗi vật thể trong hình ảnh huấn luyện được phân bố về một anchor box. Trong trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchorbox mà có IoU với boudingbox gốc của ảnh cao nhất.

Để hiểu rõ hơn ta lấy ví dụ trên hình. Tại vị trí Ô thứ i có chứa vật thể ta xác định được 3 anchor boxes viền xanh. Cả 3 anchor box này đều giao nhau với bounding box của vật thể. Tuy nhiên ta chỉ có anchor box có đường viền dày nhất màu xanh được lựa chọn làm anchor box cho vật thể bởi nó có IoU so với đối tượng trong ảnh gốc là cao nhất.

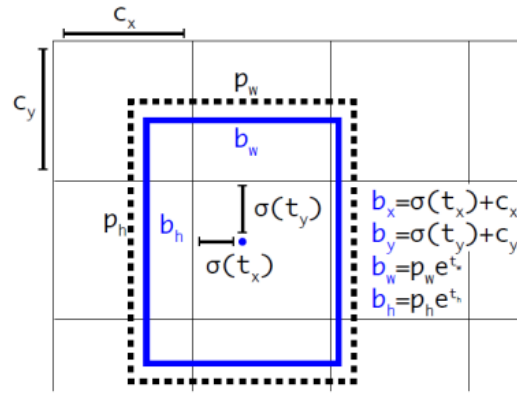


Hình 2.24 Xác định anchor box cho một vật thể

Như vậy khi xác định một vật thể ta cần xác định hai thành phần gắn liền với nó là (ô và anchor box). Tuy nhiên trong một số trường hợp hi hữu thì 2 vật thể sẽ bị tâm ở cùng một ô, điều này sẽ khiến cho thuật toán gặp khó khăn trong việc xác định phân lớp của chúng

2.5.6 Bounding Box

Để dự đoán boundingbox cho một vật thể chúng ta dựa trên phép biến đổi từ anchorbox và ô. Mô hình đang cố gắng làm sao cho boundingbox không lệch khỏi vị trí trung trung quá nhiều.



Hình 2.25 Minh họa dự đoán khung giới hạn đối tượng

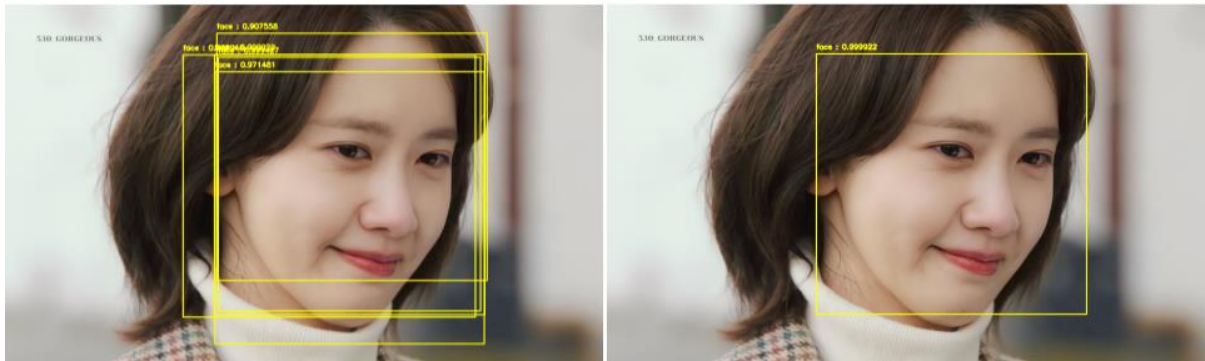
Trên hình ta thấy các thông số cần ước lượng là (b_x, b_y, b_w, b_h) tương ứng là tâm và kích thước boundingbox được dự đoán cho vật thể. Thông số (p_w, p_h) là kích thước của anchor box được cài đặt trước. (c_x, c_y) là vị trí của góc trên cùng bên trái của ô chứa đối tượng. Và 4 thông số (t_x, t_y, t_w, t_h) lần lượt là tâm và kích thước của anchor và 4 thông số này đều đã được chuẩn hóa và được học trong quá trình back propagation và đạo hàm (gradient descent). Tọa độ của một boundingbox sẽ được xác định dựa trên đồng thời cả anchor box và ô mà nó thuộc về. Điều này giúp kiểm soát vị trí của boundingbox dự đoán đâu đó quanh vị trí của ô và bounding box mà không vượt quá xa ra bên ngoài kích thước của anchorbox cho trước đó.

Đối với mô hình yolov3 với đầu vào là hình ảnh có kích thước 416×416 thì sẽ có ba ngõ ra với số lượng ô tương ứng là 13×13 , 26×26 và 52×52 . Mà tương với mỗi ngõ ra, các đối tượng trên đó sẽ được cung cấp 3 anchor box làm khuôn. Điều này khiến cho mỗi đối tượng trên ngõ ra có thể tới 9 anchor box, điều này rất bất tiện trong việc tìm chính xác vị trí dự đoán của vật thể, do đó cần một phương pháp để tìm ra một boundingbox tốt nhất trong số các anchorbox đó. Phương pháp đó là Non-max suppression.

2.5.7 Non-max suppression (NMS)

Do thuật dự báo ra nhiều boundingbox cho cùng một vật thể, cho nên đối với những ô vuông cơ sở có vị trí gần nhau, khả năng các khung hình bị trùng lặp là rất cao. Do đó mô hình cần một giải pháp để giảm bớt số lượng các khung hình được sinh ra một cách đáng kể.

Dưới đây là một ví dụ output của một mô hình face detection khi chưa được xử lý bằng NMS và đã được xử lý qua NMS



Hình 2.26 Non-max Supression

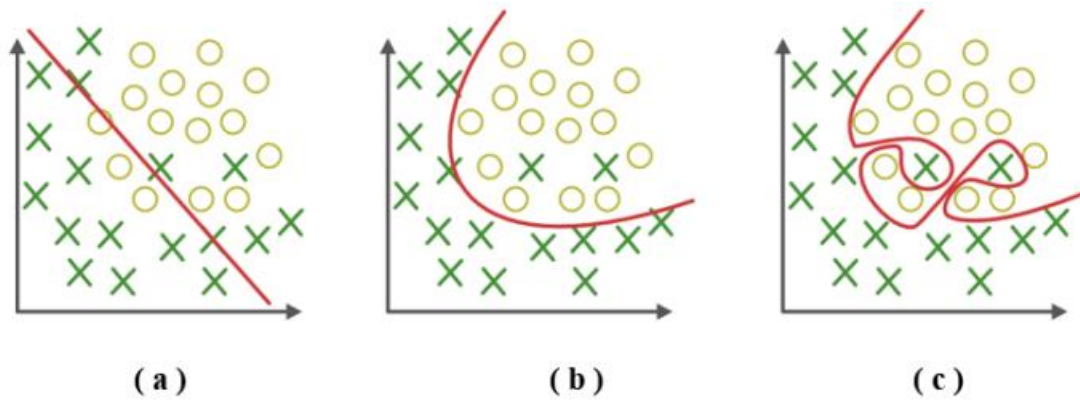
Ta thấy khi không dùng kỹ thuật NMS thì một đối tượng có thể có nhiều boudingbox, điều này là một điểm sai trong bài toán nhận diện. Còn khi áp dụng NMS thì chỉ còn một boudingbox cho một đối tượng. Giải thuật Non-max suppression:

Bước 1: Đầu tiên chúng ta sẽ tìm cách giảm bớt số lượng các boudingbox bằng cách lọc bỏ toàn bộ những boudingbox có xác suất chứa vật thể nhỏ hơn một ngưỡng nào đó, thường chọn là 0.5

Bước 2: Đối với các boudingbox giao nhau, non-max supression sẽ lựa chọn ra một boudingbox có xác suất chứa vật thể cao nhất. Sau đó tính toán chỉ số giao thoa IoU với các boudingbox còn lại. Nếu các boudingbox còn lại có IoU với boudingbox làm chuẩn này nhỏ hơn 0.5 thì ta xem như là hai boudingbox này không chồng lấn, hay boudingbox đó sẽ thuộc về một vật thể khác. Còn nếu lớn hơn 0.5 thì ra xem là chồng chéo nhau và ta tiến hành xóa các boudingbox này. Sau cùng ta được một boudingbox duy nhất có xác suất cho nhất cho một vật thể.

2.5.8 Đánh giá model

Khi huấn luyện một mô hình, ta luôn cố gắng tìm mô hình dự đoán chính xác nhất có thể dựa trên những dữ liệu thu thập được, để từ đó có thể dự đoán chính xác ở dữ liệu mới. Nhưng việc huấn luyện không đơn giản như vậy, sẽ có 4 vấn đề cần quan tâm : Generalization (Học tổng quát), Overfitting (Học quá sâu), Underfitting (Học chưa đủ) và Trade – Off (Điểm vàng trong quá trình học)



Hình 2.27 Underfitting (a) Trade – Off (b) Overfitting (c)

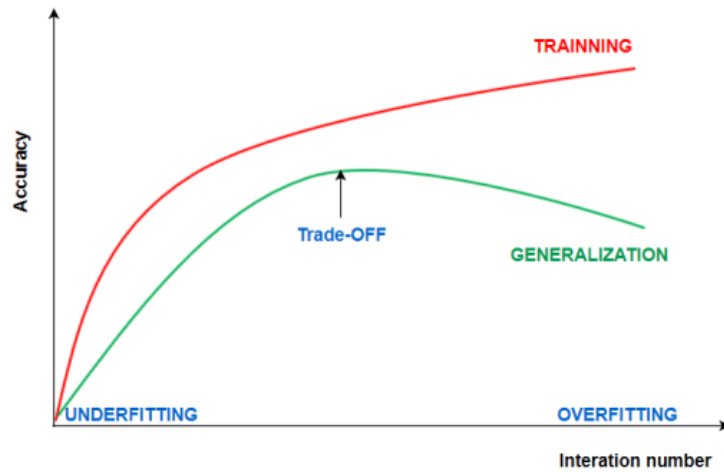
Hình trên là 3 trường hợp sẽ xuất hiện khi huấn luyện mô hình. Với đường màu đỏ là mô hình đã được huấn luyện và các điểm data để huấn luyện. Với hình (a) thì mô hình quá đơn giản chưa có thể phân biệt được hai loại data đó, còn hình (c) thì mô hình học quá sâu và nó đang cố gắng phân biệt rõ ràng hai loại dữ liệu, do đó nó mất tính tổng quát. Chỉ có hình (b) là mô hình được huấn luyện bình thường và có được tính tổng quát.

Generalization: là trường hợp mà mô hình dự đoán được chính xác phần lớn các trường hợp dữ liệu mới thì ta gọi đó mà một mô hình đạt được độ khái quát hóa.

Underfitting: là trường hợp mô hình được học quá đơn giản, chưa huấn luyện tốt ở tập dữ liệu huấn luyện, nên cũng không đạt kết quả tốt ở tập dữ liệu thực tế mới.

Overfitting: là trường hợp mô hình được học quá phức tạp và gắn chặt với dữ liệu huấn luyện. Mang lại kết quả rất tốt trên dữ liệu huấn luyện, nhưng lại mang kết quả không tốt trên tập dữ liệu thực tế.

Trade – Off: Ta nhận thấy mô hình sẽ học tốt trong khoảng từ khi mô hình bị underfitting cho tới khi mô hình chuẩn bị underfitting. Sẽ luôn tồn tại một điểm vàng trong khoảng đó mà đạt mức độ generalization là cao nhất.



Hình 2.28 Biểu đồ mô tả quá trình huấn luyện

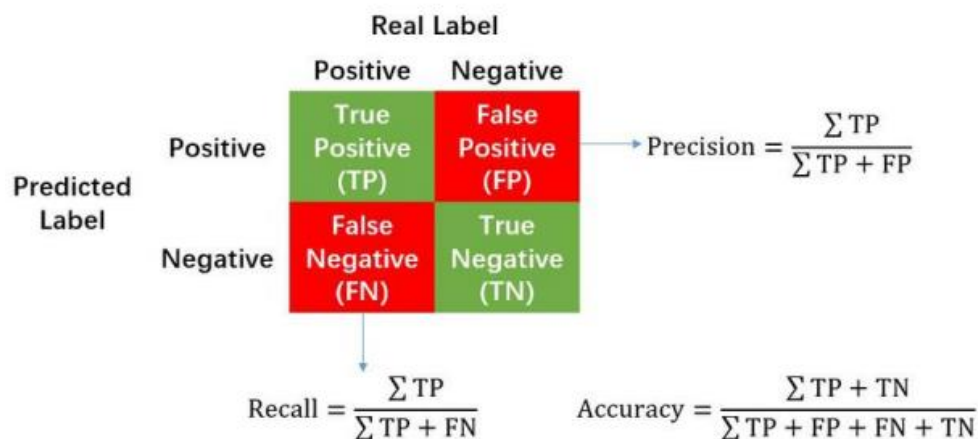
Ngoài ra, đối với mỗi model Object detection sau khi đào tạo, cần có những thang điểm để đánh giá sự chính xác của nó. Hiện tại chúng ta đánh giá một model dựa trên: loss function, IOU avg, mAP, đánh giá trực quan, và nhiều bài báo cũng như các trang web lớn thường sử dụng mAP như là thước đo chính.

Để tìm hiểu về mAP, trước tiên ta review lại khái niệm về Precision, Recall và IoU

Precision: Đánh giá độ tin cậy của kết luận đưa ra (bao nhiêu % lời kết luận của model là chính xác). Dùng cho các trường hợp cần sự chính xác tuyệt đối. Giả sử trong việc xác định ca bệnh thì precision trong trường hợp này là tỷ lệ giữa những người có bệnh so với tất cả các ca được dự đoán là có bệnh. Thường được sử dụng trong các lĩnh vực về y tế.

Recall: Đánh giá khả năng tìm kiếm toàn bộ các ground truth của mô hình (bao nhiêu % positive samples mà model nhận diện được). Giả sử trong việc xác định ca bệnh thì recall trong trường hợp này là trong số những người thực sự có bệnh thì có bao nhiêu người được dự đoán đúng là có bệnh. Recall phục vụ cho các trường hợp ta có bỏ sót những người nào đang mắc bệnh không

IoU (Intersection over Unit): Đo độ overlap giữa ground truth bounding box với bounding box mà mô hình dự đoán



Hình 2.29 Cách tính Precision và Recall

Trong đó:

- True/False ý chỉ những gì chúng ta dự đoán đã đúng hay chưa (true or false).
- Positive/Negative ý chỉ những gì chúng ta dự đoán (có hoặc không).
- TP (True Positive): Tổng số trường hợp dự báo khớp Positive.
- TN (True Negative): Tổng số trường hợp dự báo khớp Negative.
- FP (False Positive): Tổng số trường hợp dự báo các quan sát thuộc nhãn Negative thành Positive.
- FN (False Negative): Tổng số trường hợp dự báo các quan sát thuộc nhãn Positive thành Negative.

Một ví dụ khác. Bạn muốn xây dựng một hệ thống gợi ý sản phẩm trực tuyến. Dự đoán “Positive” trong trường hợp này chính là “Những sản phẩm thật sự thu hút khách hàng”. Mô hình của bạn sẽ hiển thị những sản phẩm liên quan đến sản phẩm mà họ đang xem để họ có thể mua thêm nhiều sản phẩm khác trên trang web bán hàng của bạn (Amazon, Tiki, Lazada,... chẳng hạn).

Nếu precision quá cao trong khi recall lại thấp, những gợi ý của bạn đúng là thu hút được khách hàng nhưng bạn lại bỏ qua quá nhiều sản phẩm tiềm năng khác cũng có khả năng thu hút họ không kém.

Ngược lại, nếu precision thấp trong khi recall cao thì bạn sẽ chắc chắn tất cả các sản phẩm tiềm năng sẽ được giới thiệu đến khách hàng. Tuy nhiên, những sản phẩm

thừa mứa và vô vị khác cũng sẽ chen chân vào đây và khiến cho khách hàng của bạn không mấy mặn mà, họ có thể đổi sang trang khác để mua!

Từ đó Người ta mong muốn là Precision và Recall đều cao, nên sinh ra thông số $F1_score$.

$$\frac{2}{F_1} = \frac{1}{Precision} + \frac{1}{Recall}$$

AP (Average Precision): Đây là thông số được lựa chọn nhiều nhất trong việc đánh giá mô hình nhận diện đối tượng. Bởi vì bản thân các thông số Precision và Recall chỉ làm tốt việc đánh giá việc phân loại đối tượng, nhưng không đánh giá được việc định vị đối tượng. Do đó mới sinh ra thêm các đại lượng $AP^{IoU=0.5}$ (đánh giá với những đối tượng có $IoU \geq 0.5$) và $AP^{IoU=0.75}$ (đánh giá với những đối tượng có $IoU \geq 0.75$).

Xét bài toán đơn giản: Detect “xe” trong tập dữ liệu. Giả sử, trong tập dữ liệu chỉ có 5 bounding box có nhãn ‘xe’. Ta thu thập tất cả các prediction mà output ra là ‘xe’ trên toàn bộ tập dữ liệu. Sau đó sắp xếp chúng theo thứ tự giảm dần độ confidence. Ta được bảng dưới:

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

Hình 2.30 Bảng thống kê output

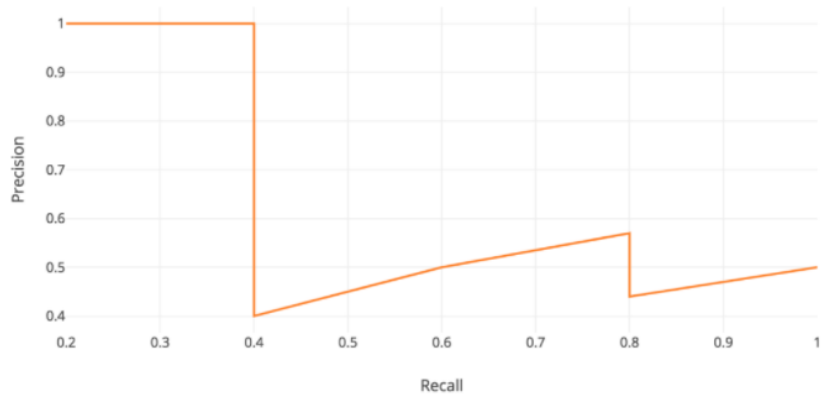
Cột thứ 2 chỉ ra nếu prediction là đúng với ground truth hay không. True nếu $IoU \geq 0.5$ và False nếu $IoU < 0.5$

Xét rank #3:

- $\text{precision} = \text{TP} / (\text{all predictions}) = 2/3$;
- $\text{recall} = \text{TP} / (\text{all positive ground truth}) = 2/5$

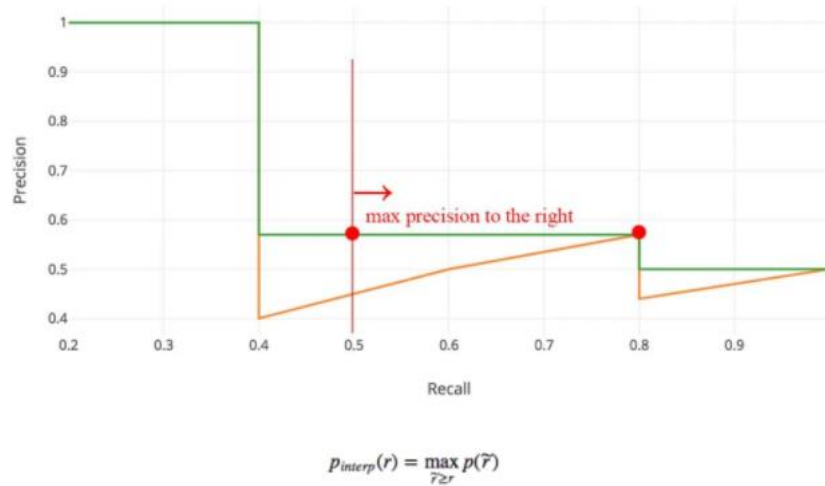
Khi càng xuống rank thấp, recall luôn tăng (từ 0 tới 1) còn giá trị precision thay đổi zig-zag, tăng khi prediction đúng, giảm khi prediction sai.

Đồ thị mối quan hệ của precision, recall:



Hình 2.31 Đồ thị biểu thị mối quan hệ giữa precision và recall

Giá trị AP là giá trị phía dưới đường biểu diễn mối quan hệ precision – recall. Để đơn giản việc tính đường zig-zag này, ta xấp xỉ chúng như sau: Tại mỗi recall level, ta thay giá trị precision bằng giá trị precision lớn nhất tại recall level đó.



Hình 2.32 Mối quan hệ giữa AP với precision-recall

Chia đường biểu diễn precision – recall (sau khi đã xấp xỉ) thành 11 đoạn bằng nhau theo recall (0, 0.1, 0.2, ..., 1). Khi đó, AP được tính bằng công thức:

$$AP = \frac{1}{11} \times [AP_r(0) + AP_r(0.1) + \dots + AP_r(1.0)]$$

Tuy nhiên công thức này chỉ đánh giá một lớp, nên đã sinh một đại lượng mới là: mAP (Mean average precision), cho bởi công thức: $mAP = \frac{\sum_{i=1}^K AP_i}{K}$, với K là số lớp ($K > 1$)

2.6 Kết luận chương

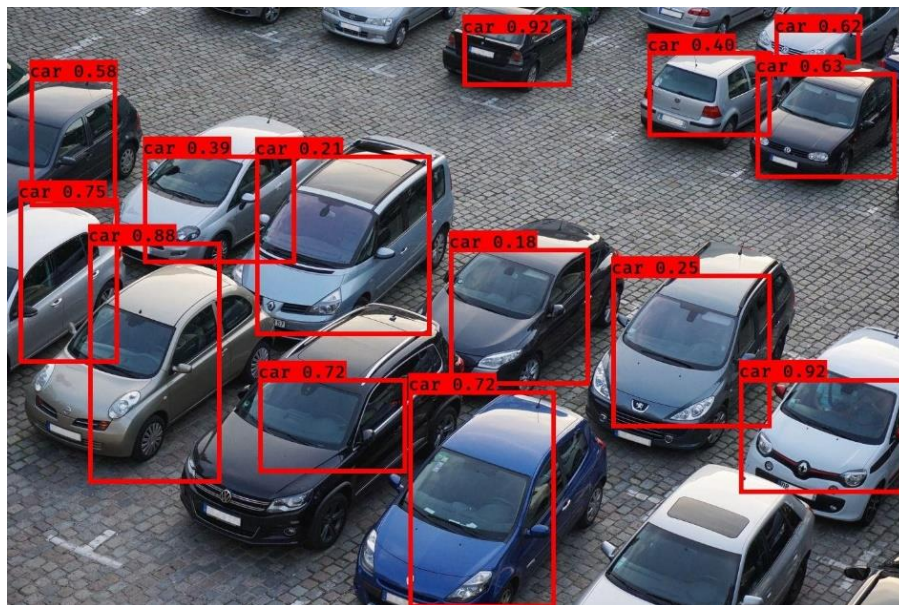
Trong chương này, chúng ta đã nắm bắt được sơ đồ khối và cách hoạt động của toàn bộ hệ thống nhận dạng chỗ đỗ xe ô tô bằng camera cũng như là khối lượng công việc mà em đã đề cập, một số lý thuyết về mô hình mạng yolo và lý do tại sao lựa chọn mô hình yolo cho đề tài lần này cũng như một số lý thuyết cơ bản về mạng nơ ron nhân tạo, mạng nơ ron tích chập. Từ đó chúng ta thực hiện các công việc cụ thể để giải quyết bài toán đặt ra, điều này sẽ được trình bày chi tiết ở chương tiếp theo

Chương 3 XÂY DỰNG MÔ HÌNH XÁC ĐỊNH TRẠNG THÁI BÃI ĐỖ XE

3.1 Huấn luyện mô hình nhận dạng xe ô tô

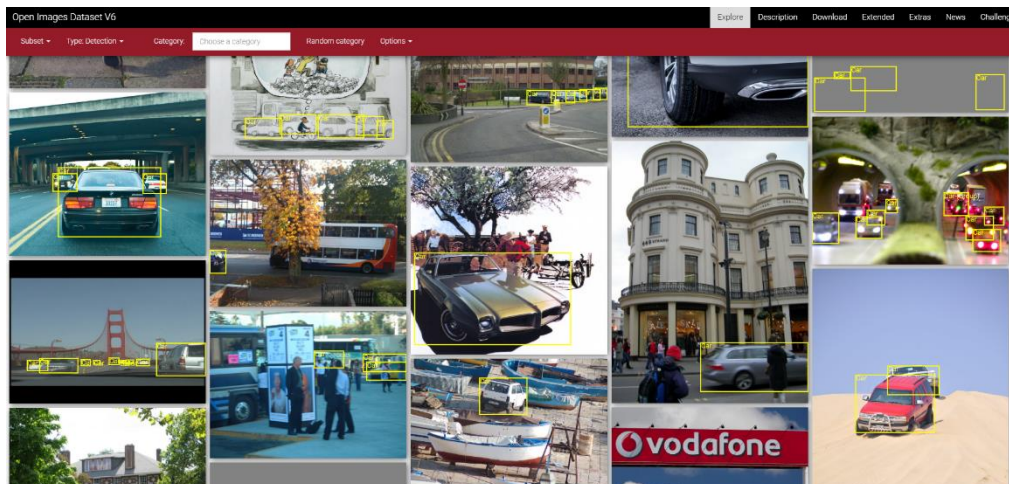
3.1.1 Chuẩn bị dữ liệu train model

Như đã đề cập trong sơ đồ khối của hệ thống ở chương 2, ta sẽ cần phải nhận diện được các đối tượng xe ô tô xuất hiện trong khung hình. Từ đó ta bắt đầu đi thu thập dữ liệu cần thiết cho quá trình huấn luyện mô hình nhận dạng xe ô tô.



Hình 3.1 Nhận dạng xe ô tô

Tập dữ liệu về xe ô tô sẽ được thu thập Open Images V6. Đây là tập dữ liệu gồm 9 triệu hình ảnh, chứa tổng cộng 16 triệu hộp giới hạn cho 600 lớp đối tượng trên 1,9 triệu hình ảnh, làm cho nó trở thành tập dữ liệu lớn nhất hiện có với chú thích vị trí đối tượng. Các hộp phần lớn được vẽ thủ công bởi các nhà chú giải chuyên nghiệp để đảm bảo độ chính xác và nhất quán. Các hình ảnh rất đa dạng và thường chứa các cảnh phức tạp với một số đối tượng (trung bình 8,3 trên một hình ảnh).



Hình 3.2 Open Image Dataset V6

Tập dữ liệu bao gồm 5000 ảnh về xe ô tô và 4000 ảnh về biển số xe ô tô. Mỗi hình sẽ có tương ứng một tệp tin văn bản với đuôi *.txt. Dữ liệu được thu thập thông qua một open source OVIDV4 ToolKit và chuyển đổi định dạng về cấu trúc dữ liệu YOLOv3. Để gia tăng tính đa dạng của dữ liệu cũng như phục vụ cho quá trình phát hiện vào ban đêm thì sẽ tạo thêm những biến thể ảnh xám để giả định cho ảnh của các camera hồng ngoại vào ban đêm. Bên trong tệp là các nhãn được đánh dấu bằng định dạng của YOLO với các thông tin:

- Class id : là các con số đại diện cho thứ tự của nhãn trong danh sách các nhãn. Ví dụ ở đây có 2 đối tượng là “Car” và “Plate” thì tương ứng ta có 2 id = 0 và id = 1
- Các thông số tiếp theo là x, y, w, h là các thông số cần thiết về bounding box tương ứng là tâm (x,y), chiều dài w và chiều cao h của đối tượng



Hình 3.3 Ảnh minh họa và định dạng tệp train model YOLOv3

Như vậy với tổng số lượng ảnh là 9000 ảnh ta tiến hành chia thành 3 tập cho quá trình huấn luyện, bao gồm :

- Tập training : Đây thường là một tập dữ liệu có kích thước lớn, được dùng để training trong quá trình huấn luyện máy học. là tập dữ liệu máy dùng để học và rút trích được những đặc điểm quan trọng để ghi nhớ lại. Training set là tập các cặp input và output dùng để huấn luyện trong quá trình máy học. Do đó nó sẽ chiếm 80% tổng số lượng ảnh
- Tập validation : Do trong quá trình huấn luyện máy học sẽ có một vấn đề là Overfitting. Validation test cũng giống như tập training set, nó cũng bao gồm các cặp giá trị input và output tương ứng. Nhưng nó lại khác training set ở chỗ, nó được sử dụng để kiểm thử độ chính xác của mô hình máy học trong quá trình huấn luyện để tránh hiện tượng máy học thuộc lòng dữ liệu training dẫn tới việc đạt độ chính xác rất cao ở tập training nhưng khi sang tập test thì độ chính xác lại rất thấp. Do đó nó sẽ chiếm 10% tổng số lượng ảnh
- Tập test : testing set là tập dữ liệu dùng để test sau khi máy đã học xong. Một mô hình máy học sau khi được huấn luyện, sẽ cần phải được kiểm chứng xem nó có đạt hiệu quả không. Khác với Training set, Testing set chỉ gồm các giá trị input mà không có các giá trị output. Máy tính sẽ nhận những

giá trị input này, và xử lý các giá trị, sau đó đưa ra output tương ứng cho giá trị input. Do đó nó sẽ chiếm 10% tổng số lượng ảnh

Như vậy là ta đã chuẩn bị xong cho quá trình huấn luyện. Tiếp theo ta sẽ tới cách huấn luyện mô hình trên google colab

3.1.2 Huấn luyện mô hình trên google colab

Colaboratory hay còn gọi là Google Colab, là một sản phẩm từ Google Research, nó cho phép chạy các dòng code python thông qua trình duyệt, đặc biệt phù hợp với Data analysis, machine learning và giáo dục. Colab không cần yêu cầu cài đặt hay cấu hình máy tính, mọi thứ có thể chạy thông qua trình duyệt, có thể sử dụng tài nguyên máy tính từ CPU tốc độ cao và cả GPUs và cả TPUs đều được cung cấp. Colab cung cấp nhiều loại GPU, thường là Nvidia K80s, T4s, P4s and P100s, tuy nhiên người dùng không thể chọn loại GPU trong Colab, GPU trong Colab thay đổi theo thời gian.

Vì là dịch vụ miễn phí, nên Colab sẽ có những thứ tự ưu tiên trong việc sử dụng tài nguyên hệ thống, cũng như giới hạn thời gian sử dụng, thời gian sử dụng tối đa là 10 tiếng sau đó sẽ kill toàn bộ session, xóa toàn bộ dữ liệu. Tuy nhiên chúng ta vẫn có cách khắc phục tình trạng này sẽ được đề cập sau đây. Cấu hình phần cứng Google Colab cung cấp

CPU	GPU	TPU
Intel Xeon Processor with two cores @ 2.30 GHz and 13 GB RAM	Up to Tesla K80 with 12 GB of GDDR5 VRAM, Intel Xeon Processor with two cores @ 2.20 GHz and 13 GB RAM	Cloud TPU with 180 teraflops of computation, Intel Xeon Processor with two cores @ 2.30 GHz and 13 GB RAM

Hình 3.4 Cấu hình phần cứng Google Colab cung cấp

Tiến hành huấn luyện dữ liệu trên google colab, về bản chất là huấn luyện trên google colab là môi trường máy ảo chạy Linux.

Bước 1: Để gia tốc thời gian quá trình huấn luyện model ta cần chuyển sang sử dụng GPU trên google colab

Bước 2: Để giải quyết vấn đề lưu dữ liệu trên Colab thì sau 10 tiếng, nó sẽ xóa trắng. Chúng ta sẽ phải sử dụng Google Drive để lưu trữ dữ liệu gồm: mã nguồn, file cấu hình và file backup weights

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

Bước 3: Tải mã nguồn darknet về google drive, cấu hình Opencv, GPU và CUDNN, đồng thời cấp phép quyền thực thi cho darknet.

```
!git clone https://github.com/AlexeyAB/darknet
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!make
```

Bước 4: Tiến hành tải file pretrained yolov3.weight ở trên trang "pjreddie.com" về thư mục lưu darknet. Sau đó mở file yolov3.cfg và sửa các dòng như sau:

- Ở các dòng thứ 8 và thứ 9 và 10 ta kiểm tra kích đầu vào của mô hình có đúng là 416x416x3
- Ở dòng thứ 11 và 18 đặt: momentum = 0.9 và learning rate=0.0005, đây là hai thông số sử dụng khi cập nhật mô hình
- Ở dòng thứ 20 và 22 chỉnh max batches = 8000 và steps = 7000, 7500 đây là hai thông số giúp định lượng số tập trọng số và từ bước 7000 đến 7500 thì mô hình sẽ học chậm lại để có thể hội tụ bằng cách giảm tốc độ học xuống
- Ở các dòng 610, 696 và 783 sửa lại thành classes=2, đây là số lượng đối tượng được huấn luyện
- Ở các dòng 603, 689, 776 ta chỉnh sửa filter = 21, được tính bằng $3 \times (5+2) = 21$ đã được đề cập trong phần lý thuyết về yolo, đây là số lượng bộ lọc ở các lớp convolution trước các lớp ngõ ra của yolo giúp định dạng đúng ngõ ra của yolo

Bước 5: Giờ ta tiếp tục với cấu hình file yolo.data, file này sẽ gồm 5 dòng

```
classes = 2 # Số lượng class, ở đây chỉ có 1 đối tượng lên classes=2
train = train.txt # trỏ đến file train
valid = val.txt # trỏ đến file val
```

```
names = yolo.names # trỏ đến file names chứa tên của các class
backup = backup # là đường dẫn sẽ lưu các file weights trong quá trình train
```

Bước 6: Thực hiện huấn luyện mô hình và kết quả sẽ được lưu vào file log.txt

```
!./darknet detector train data/obj.data cfg/yolov3-custom.cfg darknet53.conv.74 -map -dont_show 2>&1 > log.txt
```

```
class_id = 0, name = Car, ap = 43.33% (TP = 83, FP = 57)
class_id = 1, name = plate, ap = 45.21% (TP = 78, FP = 62)

for conf_thresh = 0.25, precision = 0.57, recall = 0.42, F1-score = 0.49
for conf_thresh = 0.25, TP = 161, FP = 119, FN = 219, average IoU = 42.91 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.442676, or 44.27 %

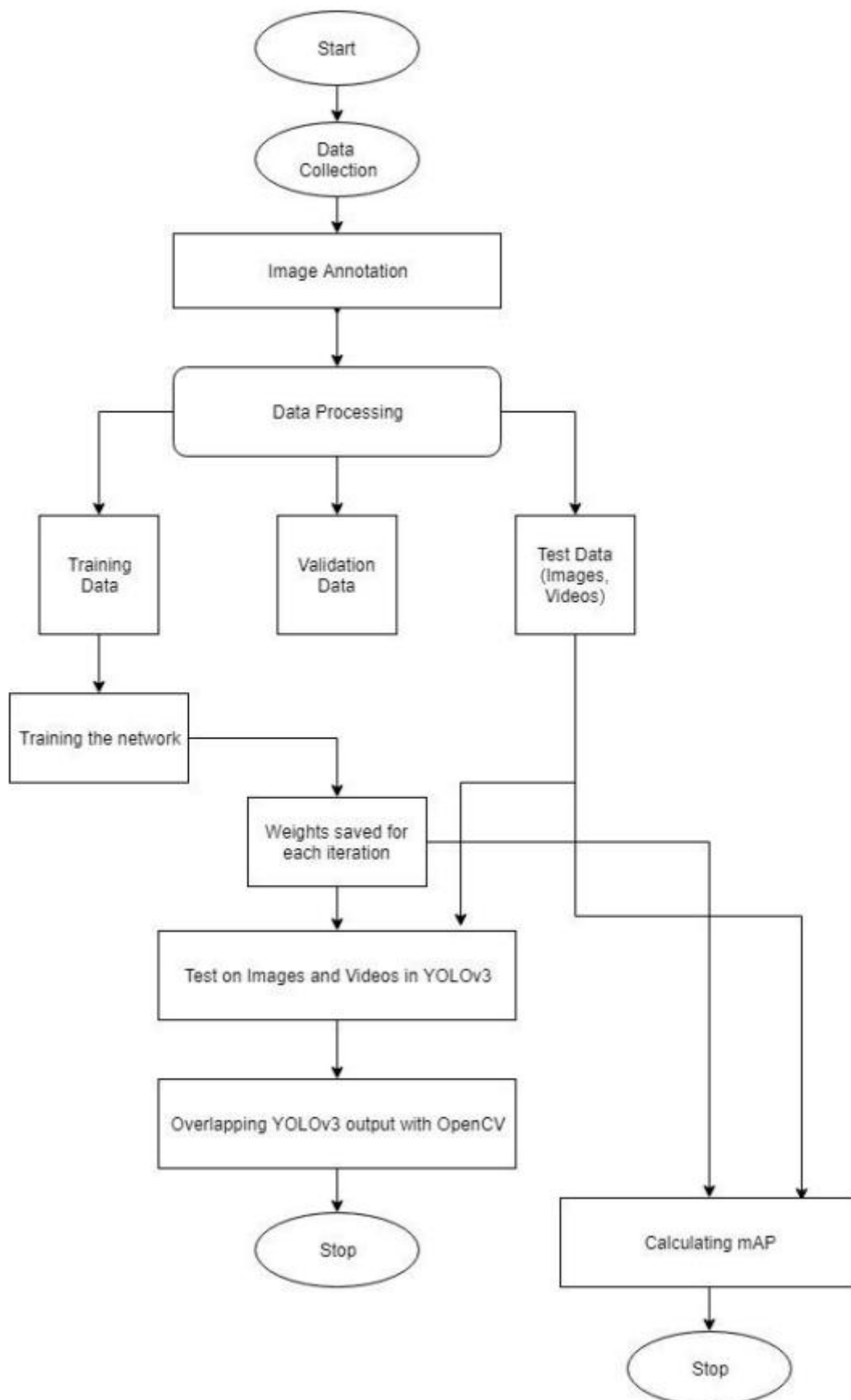
Set -points flag:
`-points 101` for MS COCO
`-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
`-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset

mean_average_precision (mAP@0.50) = 0.442676
Loaded: 0.000075 seconds

(next mAP calculation at 1760 iterations)
Last accuracy mAP@0.50 = 44.27 %, best = 46.30 %
1589: 0.688875, 0.643963 avg loss, 0.001000 rate, 11.079960 seconds, 101696 images, 30.517275 hours left
Loaded: 0.000074 seconds
```

Hình 3.5 Quá trình huấn luyện mô hình

Sau khoảng thời gian 30 tiếng . Kết quả ta thu được một mô hình có hệ số mAP@0.5 = 0.9, một kết quả chấp nhận được cho bài toán nhận diện đối tượng. Sau đó ta tiến hành kiểm thử trên tập test (tập dữ liệu hoàn toàn chưa được thấy bởi mô hình). Với 900 bức ảnh thuộc 2 đối tượng khác nhau (Car, License Plate). Cho kết quả phát hiện đúng trên 85% điều đó cho thấy mô hình đang hoạt động tốt.

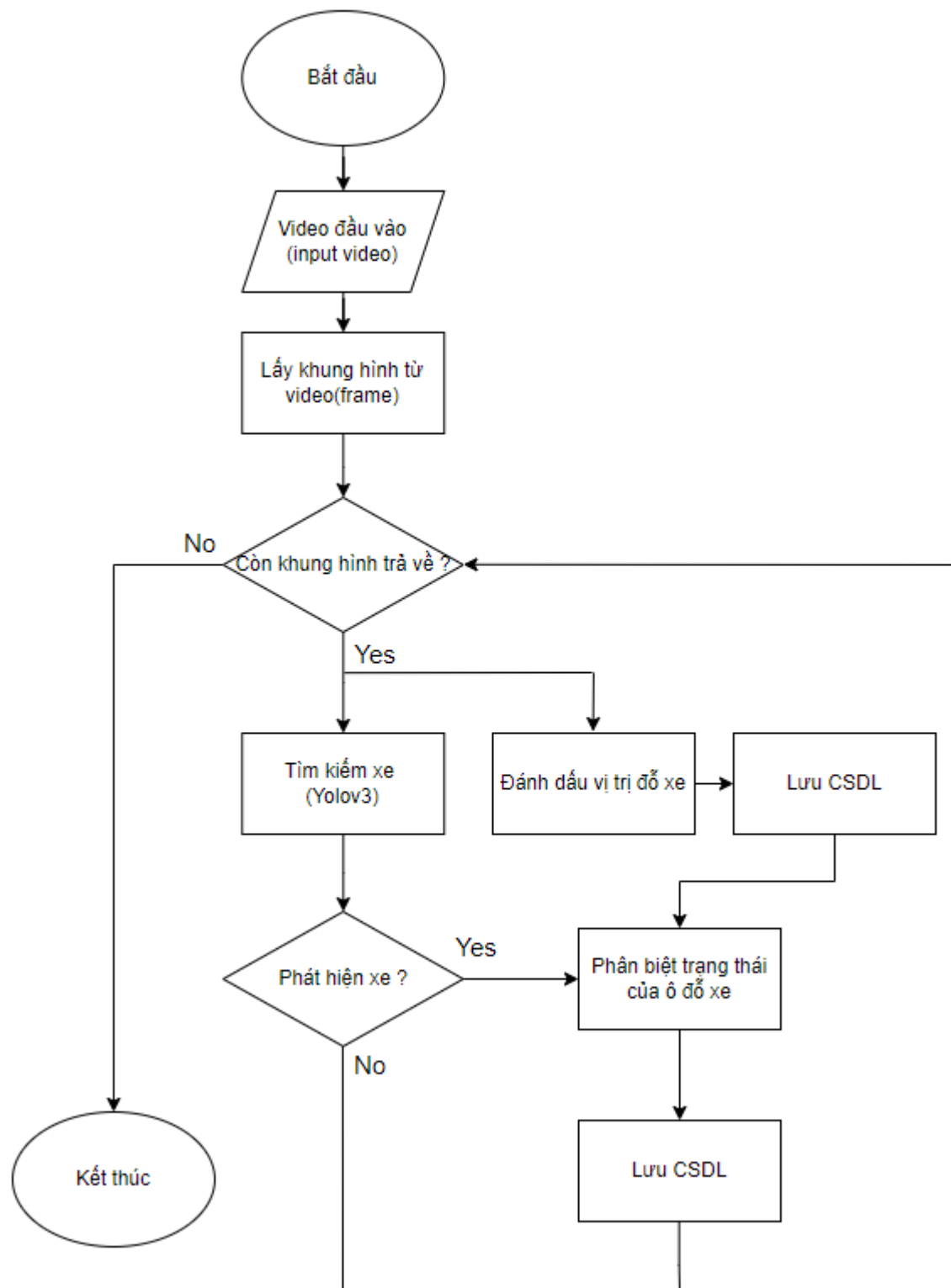


Hình 3.6 Sơ đồ giải thuật cho quá trình đào tạo và thử nghiệm mô hình YOLOv3

3.2 Xây dựng hệ thống xác định trạng thái của bãi đỗ xe

3.2.1 Xây dựng hệ thống ứng dụng mô hình học sâu Yolov3

Hệ thống xác định trạng thái của bãi đỗ xe được đề xuất theo mô hình như sơ đồ



Hình 3.7 Sơ đồ giải thuật hệ thống phát hiện trạng thái ô đỗ xe

Trong môi hình hệ thống đề xuất: Sau khi nhận dữ liệu từ camera, hệ thống sẽ xử lý 2 việc chính:

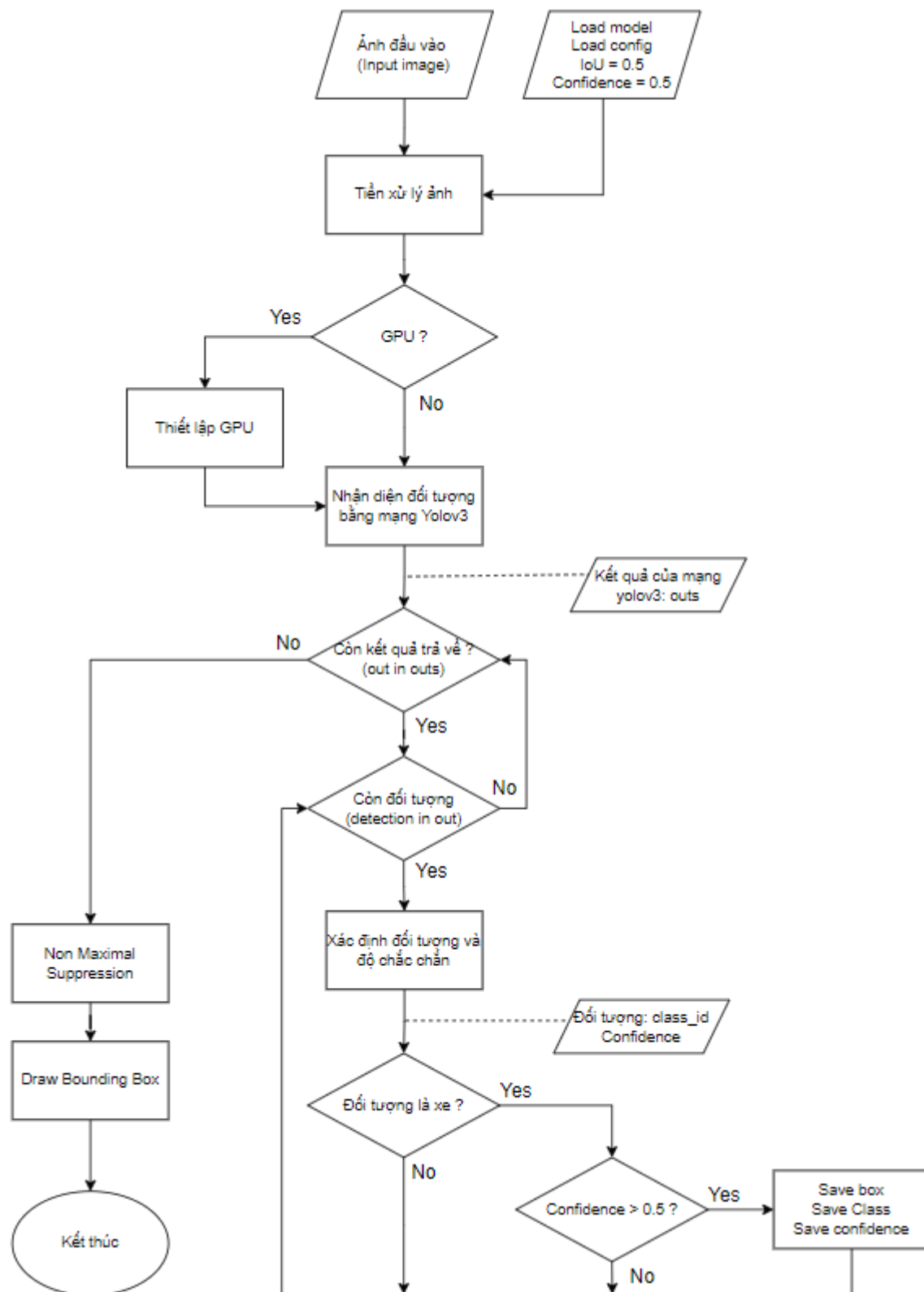
- Từ một khung ảnh cho phép người quản lý đánh dấu vị trí đỗ xe và lưu vào cơ sở dữ liệu
- Máy tính sẽ xử lý từng khung ảnh để phát hiện đối tượng là xe ô tô sử dụng thuật toán YOLOv3, trong trường hợp nếu phát hiện ra xe ô tô trong khung hình

Trong trường hợp phát hiện ra khung hình người và từ những tọa độ của các ô tô đỗ xe ta sẽ đi phân biệt trạng thái của ô tô đỗ xe là “Trống” hay “Bận” đồng thời tổng hợp và gửi thông tin tổng quát cập nhật về trạng thái có thể dừng đỗ của bãi đỗ về cơ sở dữ liệu.

3.2.2 Giải thuật phát hiện và định vị xe ô tô

Để tìm người bằng YOLOv3 trong luận văn này em sử dụng ngôn ngữ lập trình Python với môi trường Anaconda Navigator – Pycharm, đồng thời sử dụng các bộ thư viện mã nguồn mở: Opencv, Numpy,...

Để sử dụng thuật toán YOLOv3, trong bài luận văn này sử dụng một số tệp được tải xuống bao gồm: Tệp các trọng số đã được đào tạo ở phần trước của YOLOv3: yolo3.weights, tệp cấu hình: yolo3.cfg, tệp tên các đối tượng: yolo3.txt



Hình 3.8 Sơ đồ giải thuật phát hiện và định vị xe ô tô

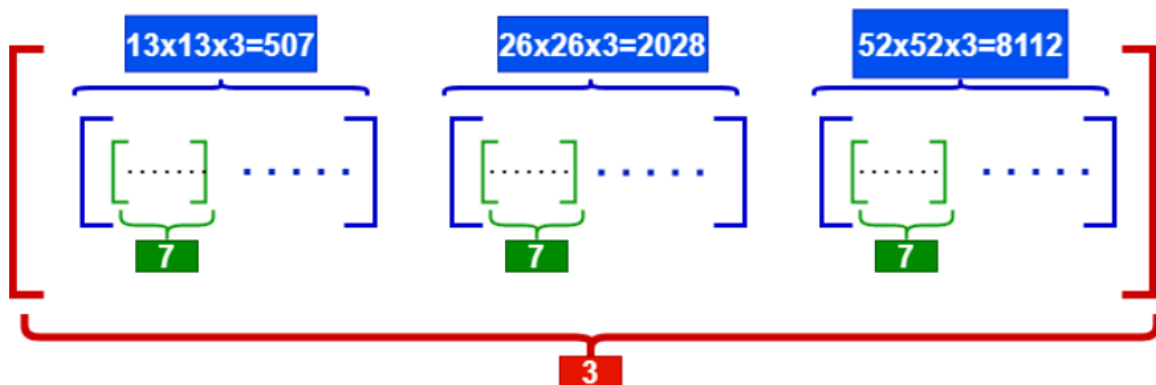
Ta sẽ đi tìm hiểu chi tiết về giải thuật trên:

Bước 1 :

- Load tệp tin cấu hình chứa các thông tin cấu từng lớp và tệp tin trong số đã học được từ việc huấn luyện mô hình nhận dạng. Khởi tạo hai giá trị ngưỡng cho mô hình. Thứ nhất, độ chính xác (Confidence) bằng 0.5, ngưỡng này cho xác định độ chắc chắn đó có thật sự là đối tượng hay không. Thứ hai, độ chồng lấn (IoU) bằng 0.5, ngưỡng này xác định mức chồng lấn của những bounding box nếu nhỏ 0.5 nó sẽ thuộc những đối tượng khác, còn ngược lại sẽ thuộc đối tượng do đó sẽ có rất nhiều bounding box bao quanh một đối tượng để loại bỏ thì ta sẽ đến bước Non-max Suppression.
- Xác định ảnh đầu vào là ảnh màu với bất kì size nào , sau đó sẽ được đưa vào khối preprocess để tinh chỉnh kích thước về 416x416

Bước 2: Kiểm tra xem máy đã hỗ trợ GPU chưa điều này sẽ tăng tốc độ xử lý lên rất nhiều, nếu chưa máy sẽ tự động quá trình dưới CPU. Sau đó ứng với ảnh đầu vào đã được xử lý cùng với các tệp cần thiết ở bước một, ta tiến hành tính toán forward và lưu vào biến outs. Biến out sẽ có 3 ngõ ra tương ứng (13x13x3), (26x26x3) và (52x52x3) vì mỗi ngõ ra sẽ tạo ra thêm 3 anchor box khác nhau. Và trong những ngõ ra này bao gồm 7 thông số cần thiết bao gồm tâm (x,y), kích thước (w,h) của đối tượng, p phần trăm xác suất xuất hiện đối tượng, và 2 thông số xác suất cho 2 đối tượng được train.

Bước 3: Sử dụng vòng lặp để truy xuất còn đối tượng nào không (detection in out).

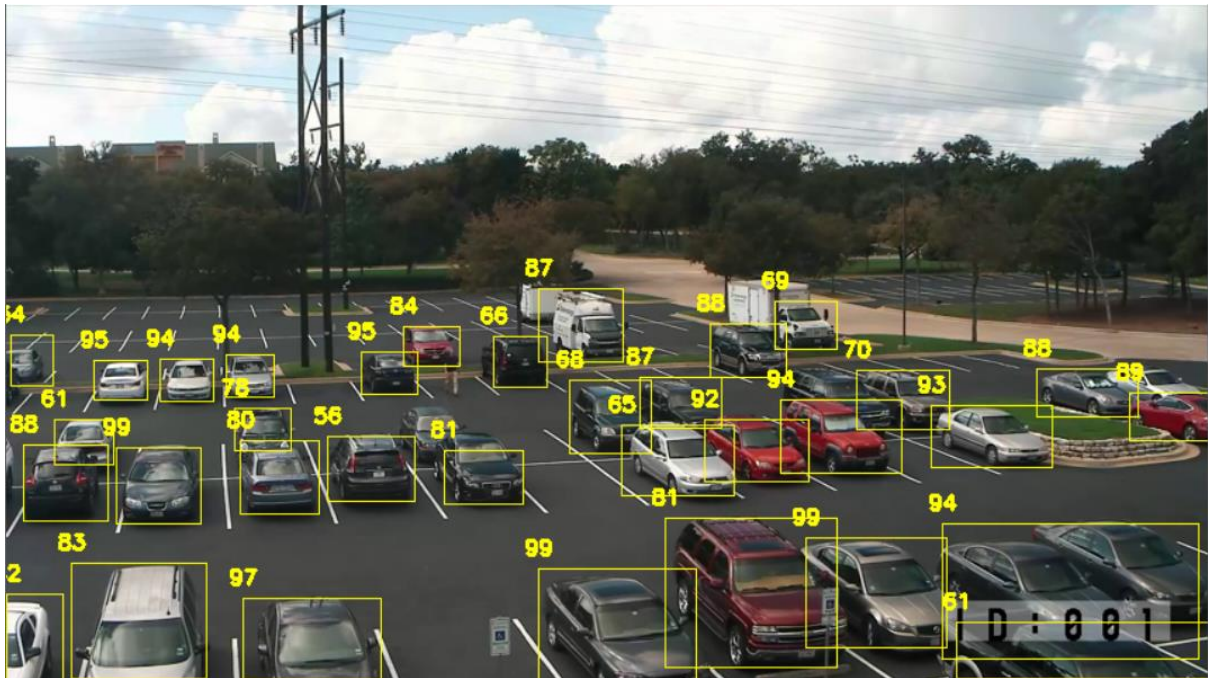


Hình 3.9 Định dạng ngõ ra outs

Đồng thời trích xuất các thông số về đối tượng: confidence và lớp phân loại . Nếu đối tượng phân loại là xe ô tô và confidence lớn hơn ngưỡng đã đặt ra là 0.5 thì tiến hành lưu các thông số kích thước, tâm của đối tượng, thông số xác suất và lớp phân loại của đối tượng. Ngược lại nếu nhỏ hơn thông số thì sẽ bỏ qua và tiến hành lặp lại bước 3.

Bước 4: Sau khi truy hết giá trị của mảng kết quả outs. Ta sẽ tiến hành xác định vị trí chính xác của đối tượng (Non Maximal Suppression) (đã đề cập 2.3.7) .Ta thu được kết quả là một mảng chứa những đối tượng độc lập và có độ tin cậy cao nhất

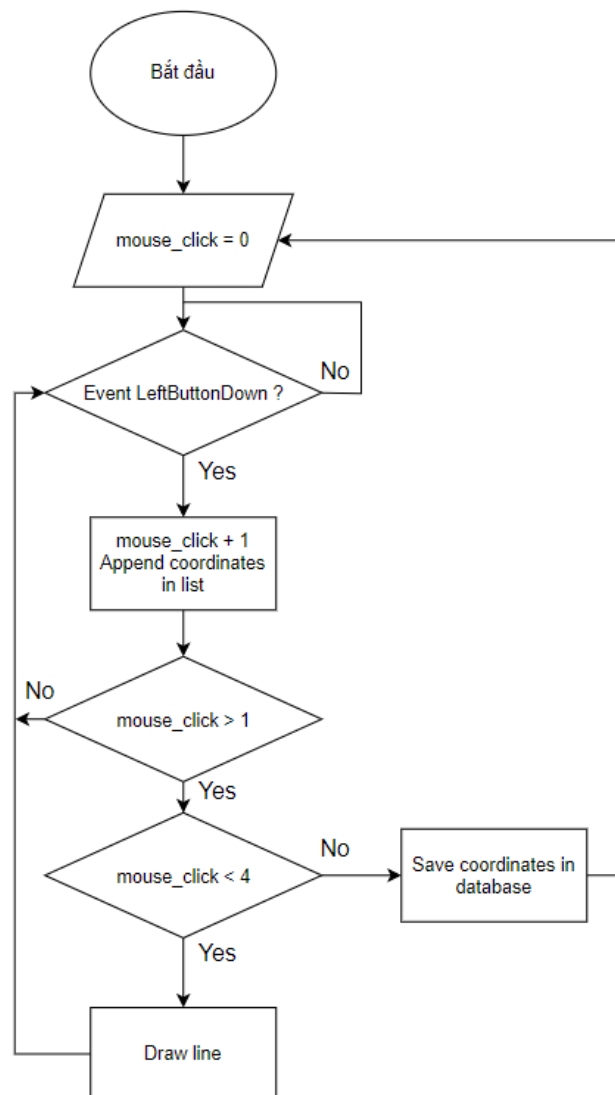
Bước 5 : Từ mảng kết quả ở bước 4 ta sẽ tiến hành vẽ khung cho các đối tượng và kết thúc



Hình 3.10 Kết quả nhận được

3.2.3 Giải thuật đánh dấu vị trí đỗ xe

Sơ đồ giải thuật:

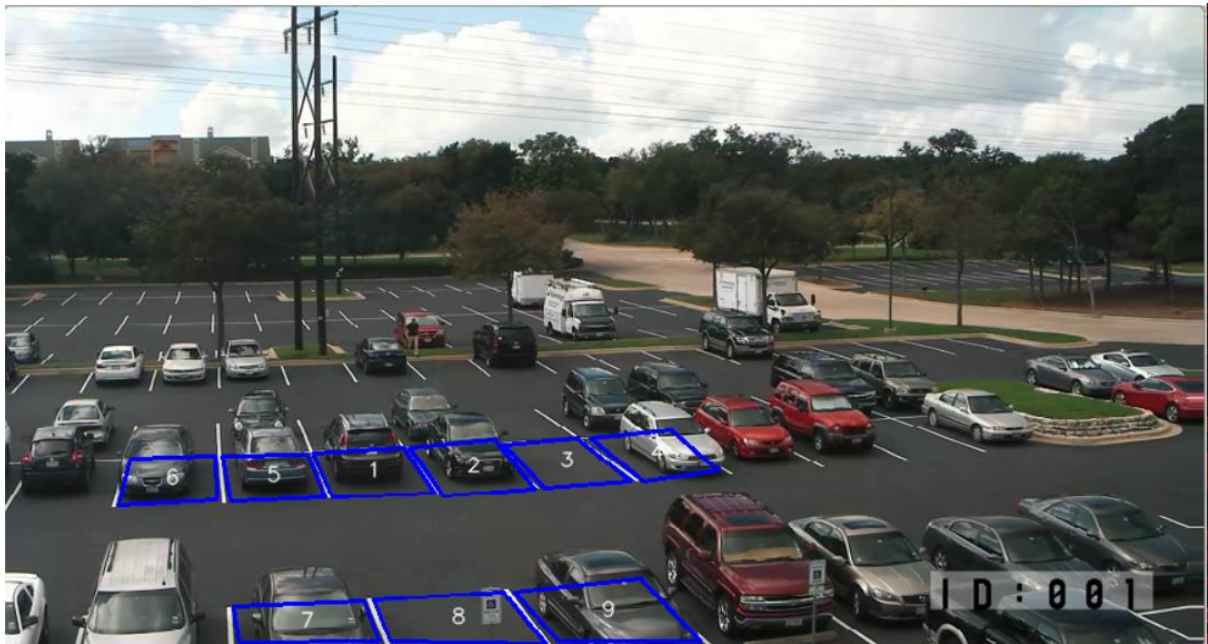


Hình 3.11 Giải thuật đánh dấu vị trí đỗ xe

Giải thuật đánh dấu vị trí đỗ xe được thực hiện qua các bước sau :

- *Bước 1:* Khởi tạo biến mouse_click bằng 0 dùng để lưu số lần nhấn chuột
- *Bước 2:* Kiểm tra sự kiện nhấn chuột trái. Nếu có sự kiện nhấn chuột thì mouse_click + 1 đồng thời lưu tọa độ tại điểm nhấn chuột vào vào list dùng để vẽ đường thẳng bao quanh vị trí đỗ xe.
- *Bước 3:* Kiểm tra xem nếu số lần nhấn chuột lớn hơn 1 và nhỏ hơn 4 thì vẽ những đường thẳng nối các điểm tọa độ đã lưu lại với nhau
- *Bước 4:* Nếu số lần nhấn chuột bằng 4 thì lưu tất cả những điểm nhấn chuột vào database

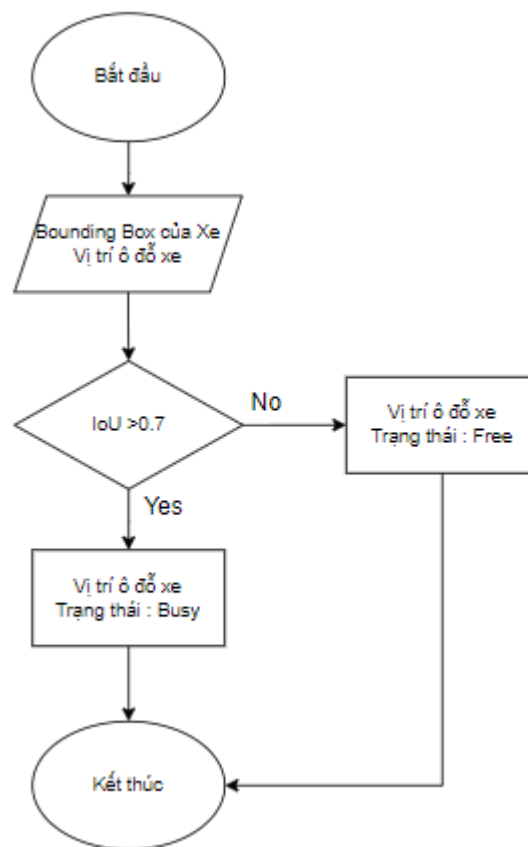
- Lặp lại bước 1



Hình 3.12 Kết quả thực hiện

3.2.4 Giải thuật phân biệt trạng thái ô đỗ xe

Sơ đồ giải thuật :

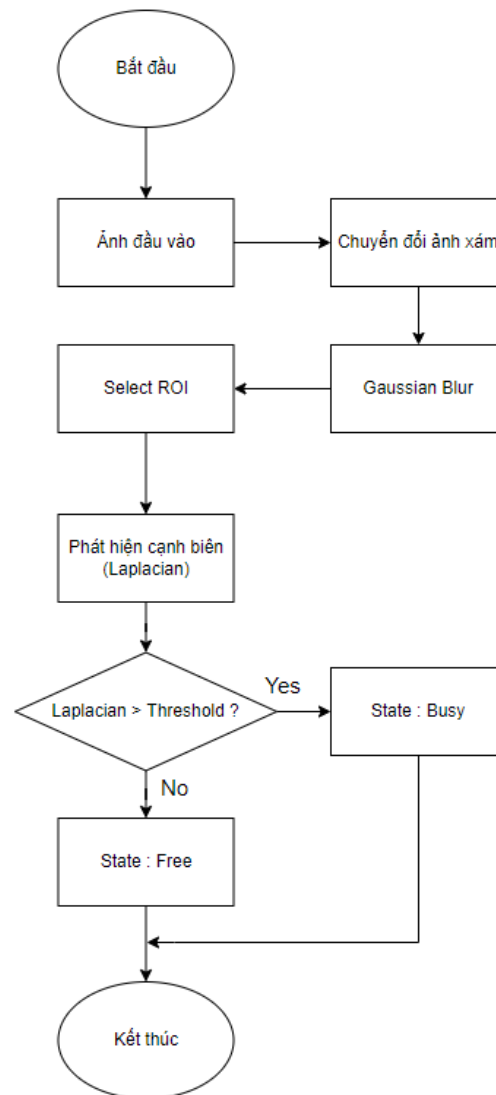


Hình 3.13 Giải thuật phân biệt trạng thái ô đỗ xe

Giải thuật phân biệt trạng thái xe ô tô được thực hiện qua các bước chính sau:

- Nhận Bounding box của tất cả xe ô tô trong trong bãi và vị trí đỗ xe
- Tính IoU giữa bounding box của xe ô tô và vị trí đỗ xe
- Nếu IoU lớn hơn 0.7 thì vị trí trả về vị trí đỗ xe, trạng thái Busy và ngược lại

3.2.5 Giải thuật phân biệt trạng thái ô đỗ xe dùng xử lý ảnh



Hình 3.14 Giải thuật phân biệt trạng thái ô đỗ xe bằng xử lý ảnh

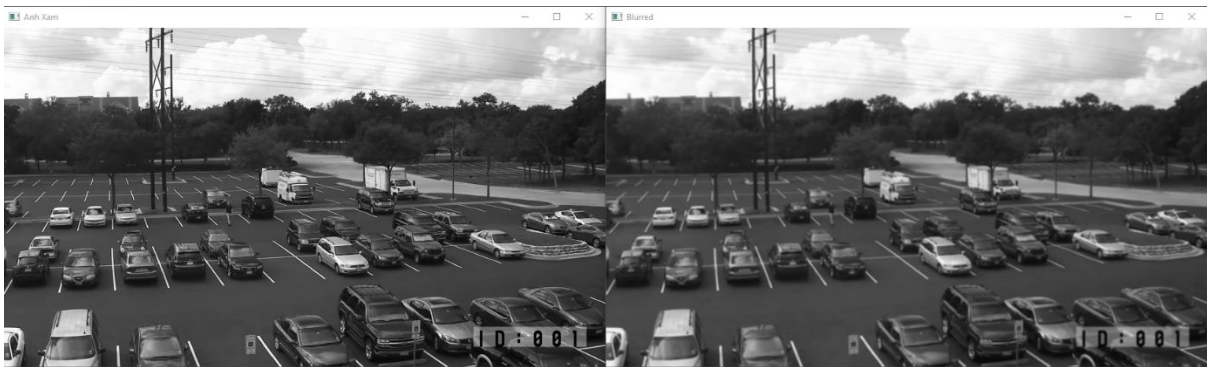
Trong phần này ta sẽ đi tiến hành xây dựng giải thuật phân biệt trạng thái vị trí đỗ xe bằng phương pháp xử lý ảnh

Với ảnh đầu vào là các ảnh màu. Ảnh màu thực chất chỉ là tập hợp của những ma trận số có cùng kích thước. Do đó khi xử lý thông tin trên ảnh, sẽ dễ dàng hơn nếu ta chỉ xử lý dữ liệu trên một ma trận số thay vì nhiều ma trận số. Việc biến đổi ảnh màu về ảnh số (Grayscale converting) xuất hiện vì mục đích trên - biến đổi thông tin ảnh về một ma trận số hai chiều duy nhất.



Hình 3.15 Chuyển đổi ảnh xám

Ảnh thu nhận được từ thiết bị thu nhận thường có nhiễu nhiều, để nhiễu không ảnh hưởng đến quá trình tách cạnh nó cần được giảm thiểu. Làm mờ ảnh, giảm nhiễu dùng bộ lọc Gaussian kích thước 5×5 . Kích thước 5×5 thường hoạt động tốt cho giải thuật Canny và Laplacian



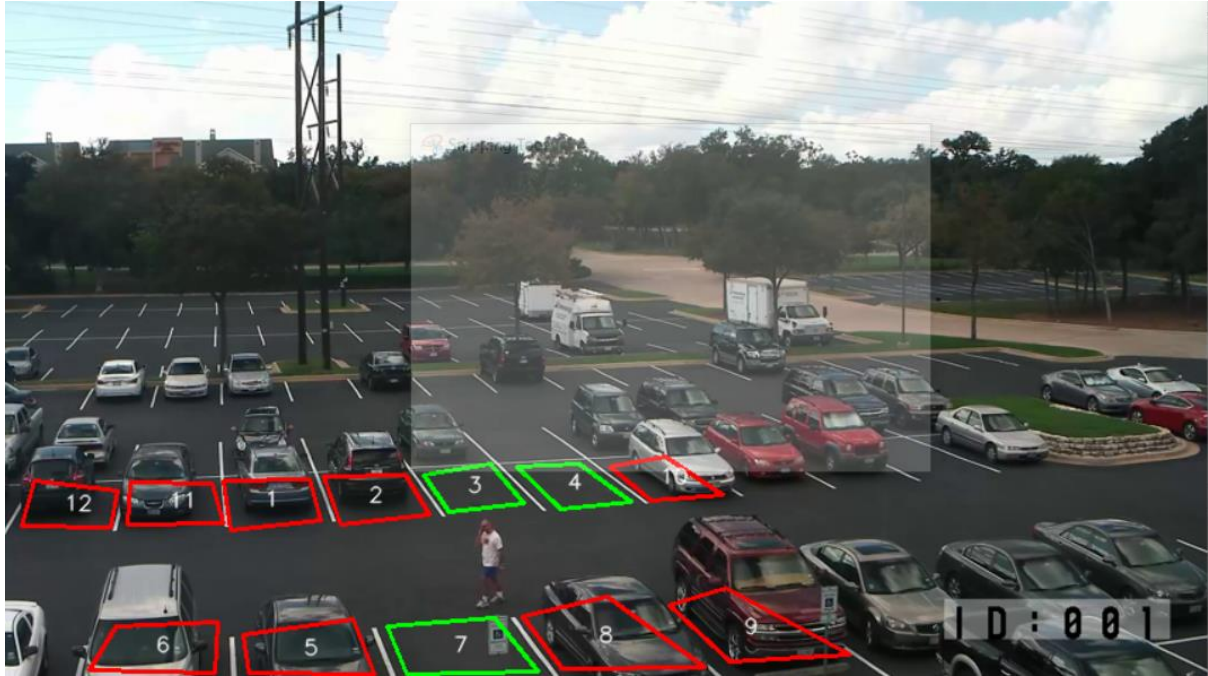
Hình 3.16 Lọc nhiễu và làm mịn ảnh

Từ các tọa độ vị trí đỗ xe đã thu được ta sẽ tiến hành trích xuất những ô đỗ xe (Select ROI). Như trong hình 3.15 ta thấy những vị trí được đánh dấu là những vị trí được trích xuất cho quá trình tiếp theo.

Để xác định được vị trí cần quan tâm (ô đỗ xe) có vật thể bị chiếm chỗ hay chưa. Với một ô đỗ xe bị chiếm chỗ có nghĩa là sẽ xuất hiện những vật thể chắn nằm trên ô đỗ xe. Với một chiếc ô tô thì nó sẽ có nhiều góc cạnh do đó để phát hiện vật thể có bị chiếm chỗ hay không ta sẽ sử dụng thuật toán phát hiện các cạnh biên (Laplacian). Do đó ta sẽ thực hiện tính toán laplacian cho các vị trí ô đỗ xe. Nếu giá trị laplacian của vị trí ô đỗ xe đó lớn hơn một ngưỡng cho phép (ngưỡng giá trị này được xác định thông qua quá trình thực nghiệm) thì có nghĩa là có những cạnh biên được phát hiện có nhiều

các cạnh biên do đó vị trí này đang bị chiếm dụng từ đó ta sẽ đánh giá được trạng thái của ô đỗ xe.

Một số kết quả thực hiện :

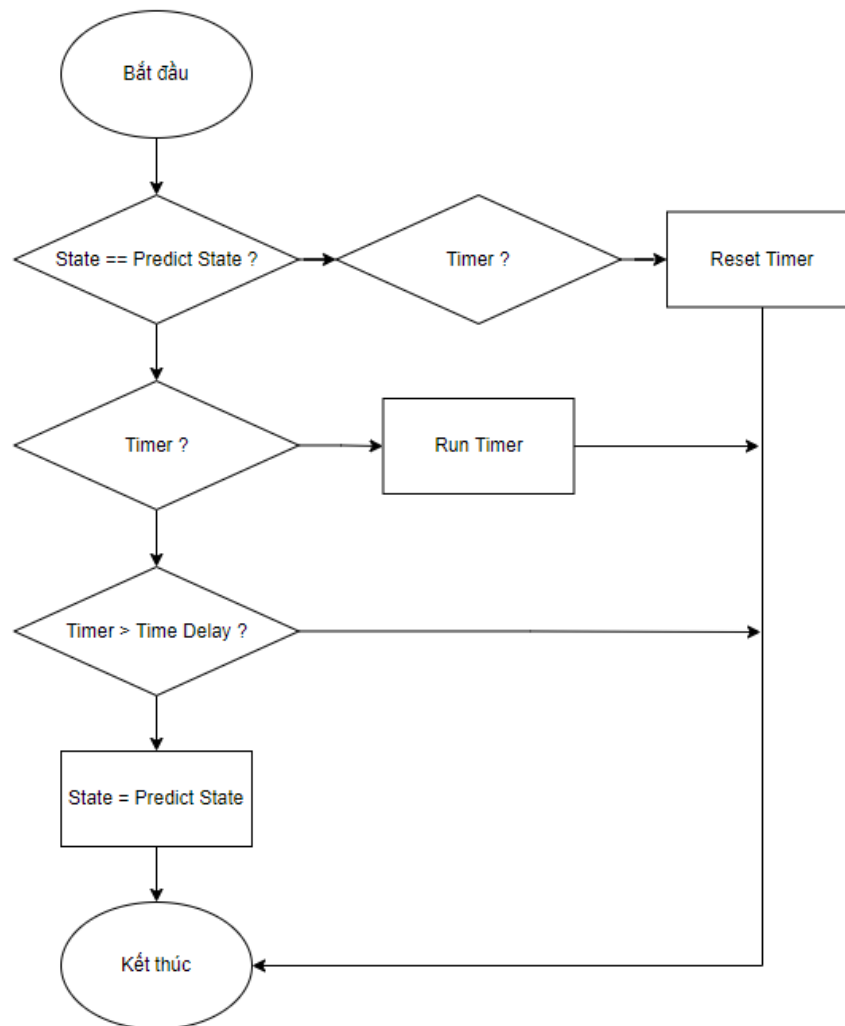


Hình 3.17 Kết quả thực hiện

Ta thấy kết quả thu được rất khả quan tuy nhiên vẫn còn nhiều nhược điểm cần lưu ý. Phụ thuộc rất nhiều vào việc lựa chọn thông số, mà việc này lại làm bằng thủ công nên độ tin cậy không cao. Ngoài ra, những thông số này không có tính chất tổng quát mà chỉ áp dụng được cho một đầu vào cụ thể, nếu thay đổi môi trường hay tác động nhỏ sẽ làm ảnh hưởng nhiều đến kết quả đầu ra. Nhưng bù lại thì tốc độ xử lý cho một ảnh như này sẽ rất nhanh. Và đó là một lý do chính mà em quyết định dùng mô hình học sâu (deep learning) thay cho mô hình xử lý ảnh cổ điển.

3.2.6 Đánh giá việc dừng đỗ của xe ô tô

Việc các xe di chuyển trong bãi khi đi qua các ô đỗ xe sẽ dẫn đến tình trạng thay đổi trạng thái ô đỗ xe do đó để tránh tình trạng đó diễn ra em sẽ sử dụng thêm một máy đếm thời gian kiểm tra việc thay đổi trạng thái ô đỗ xe



Hình 3.18 Lưu đồ sử dụng timer đánh giá trạng thái ô đỗ xe

Ta thấy nếu trạng thái dự đoán khác trạng thái hiện tại. Kiểm tra timer nếu timer chưa được bật ta sẽ bật timer. Nếu timer lớn hơn thời gian trì hoãn dự kiến (lựa chọn thông qua quá trình thực nghiệm) gán giá trị dự đoán cho giá trị hiện tại. Nếu trạng thái dự đoán giống trạng thái hiện tại thì ta reset timer

3.3 Kết luận

Trong chương này, chúng ta đã hoàn thành huấn luyện mô hình yolov3 ứng dụng cho việc nhận xe ô tô, đồng thời đánh giá chi tiết về độ chính xác mô hình . Đồng thời có cái nhìn tổng quát về hệ thống trên cũng như giải thuật chi tiết cách mà hệ thống nhận diện trạng thái bãi đỗ xe hoạt động.

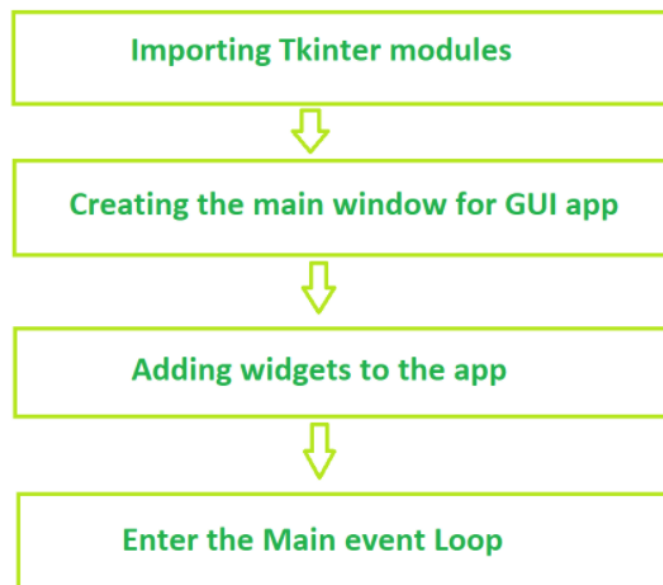
Trong chương tiếp theo, để hệ thống trở nên hoàn thiện hơn cũng như dễ dàng vận hành em sẽ xây dựng một ứng dụng giao diện cho việc quản lý bãi đỗ xe.

Chương 4 XÂY DỰNG ỨNG DỤNG QUẢN LÝ BÃI ĐỖ XE

4.1 Giới thiệu về tkinter

Giao diện người dùng đồ họa (GUI) là một dạng giao diện người dùng cho phép người dùng tương tác với máy tính thông qua các chỉ báo trực quan bằng cách sử dụng các mục như biểu tượng, menu, cửa sổ, v.v. Nó có ưu điểm hơn Giao diện dòng lệnh (CLI) nơi người dùng tương tác với máy tính bằng cách viết lệnh chỉ bằng bàn phím và cách sử dụng khó hơn GUI.

Tkinter là mô-đun python có sẵn được sử dụng để tạo các ứng dụng GUI. Đây là một trong những mô-đun được sử dụng phổ biến nhất để tạo các ứng dụng GUI bằng Python vì nó đơn giản và dễ làm việc. Đồng thời Python cũng cung cấp giao diện hướng đối tượng cho bộ công cụ Tk GUI. Ngoài ra còn có một số thư viện Python khác có sẵn để tạo ứng dụng GUI như Kivy, Python Qt, wxPython. Tuy nhiên trong luận văn này em sẽ sử dụng Tkinter để tạo GUI cho hệ thống.



Hình 4.1 Nguyên tắc hoạt động của Tkinter

Nguyên tắc hoạt động của Tkinter như trên hình. Để bắt đầu, trước tiên chúng ta sẽ thêm thư viện Tkinter. Tiếp theo đó sẽ tạo một cửa sổ chính. Chính trong cửa sổ chính này ta sẽ thực hiện các thao tác và hiển thị mọi thứ mà ta sẽ cài đặt. Sau đó, ta thêm các widget (tiện ích) và cuối cùng là vào vòng lặp sự kiện chính. Các widget trong Tkinter là các phần tử của ứng dụng GUI, cung cấp các điều khiển khác nhau (chẳng hạn như

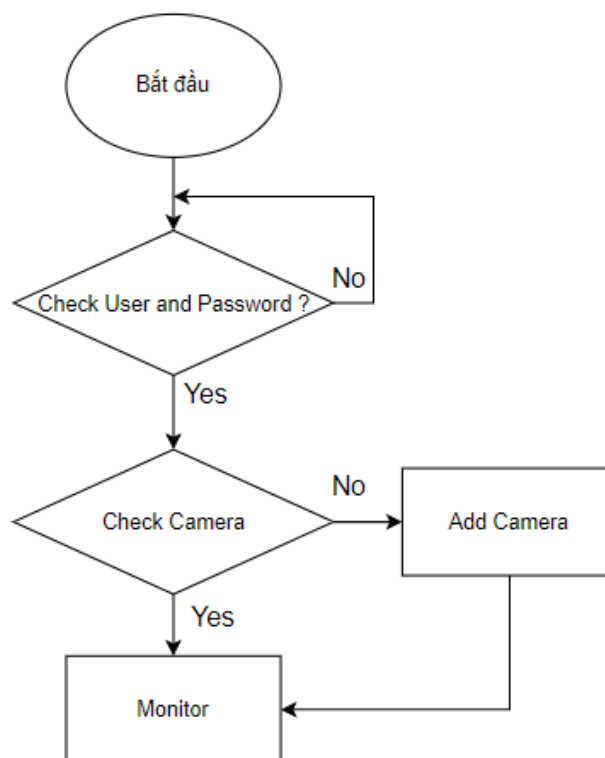
Nhấn, Nút, ComboBox, Hộp kiểm, Thanh menu, RadioButtons và nhiều thứ khác) để người dùng tương tác với ứng dụng.

4.2 Xây dựng ứng dụng

Từ những yêu cầu và mục tiêu của bài toán đặt ra, do đó ta cần xây dựng một ứng dụng đáp ứng được những nhu cầu sau :

- Cho phép quản lý trên nhiều camera ở các khu vực khác nhau trong bãi đỗ xe
- Cho phép căn chỉnh vị trí đỗ xe phù hợp với không gian bãi đỗ xe hiện tại
- Hiển thị thông tin về trạng thái bãi đỗ xe hiện tại : Số lượng chỗ đỗ xe bãi đỗ xe quản lý, số vị trí trống trong bãi đỗ xe.
- Dễ dàng vận hành, chuyển giao trong việc sử dụng
- Có hệ thống đăng nhập để tránh hiện tượng phá hoại
- Hệ thống hiển thị cho phép người quản lý giám sát bãi đỗ xe theo thời gian thực

Từ những yêu cầu trên, ta có lưu đồ giải thuật chính cho ứng dụng như hình

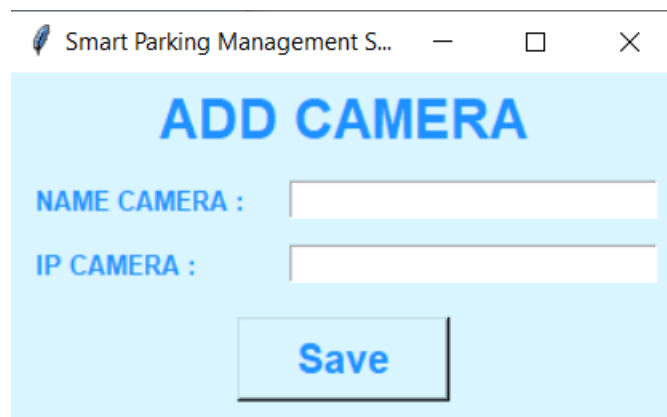


Hình 4.2 Sơ đồ giải thuật của ứng dụng

Quá trình xử lý của ứng dụng : Khi vào ứng dụng sẽ yêu cầu nhập use và password đã được cấp từ trước. Nếu đúng ta mới được phép cấp quyền quản lý bãi đỗ xe. Khi đó ứng dụng sẽ kiểm tra đã có camera thêm vào chưa nếu chưa thì tiến hành thêm camera và cho phép đi vào màn hình (Monitor) giám sát bãi xe thông qua camera được thêm vào

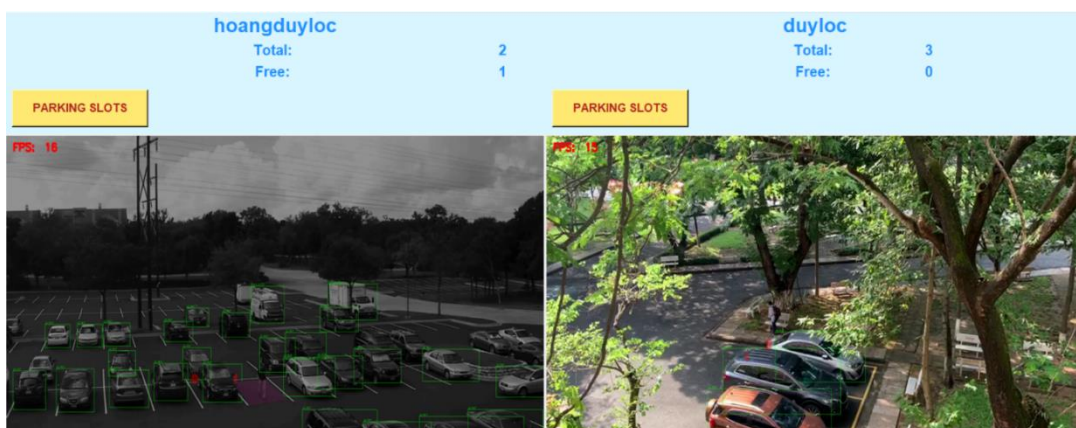
Các chức năng của ứng dụng:

- Chức năng ADD CAMERA : Cho phép ứng dụng thêm camera quản lý bãi đỗ xe. Các thông số cần thêm
 - Name Camera : Tên của khu vực mà camera quản lý
 - Ip Camera : Là địa chỉ nhận luồng thông tin từ camera truyền về



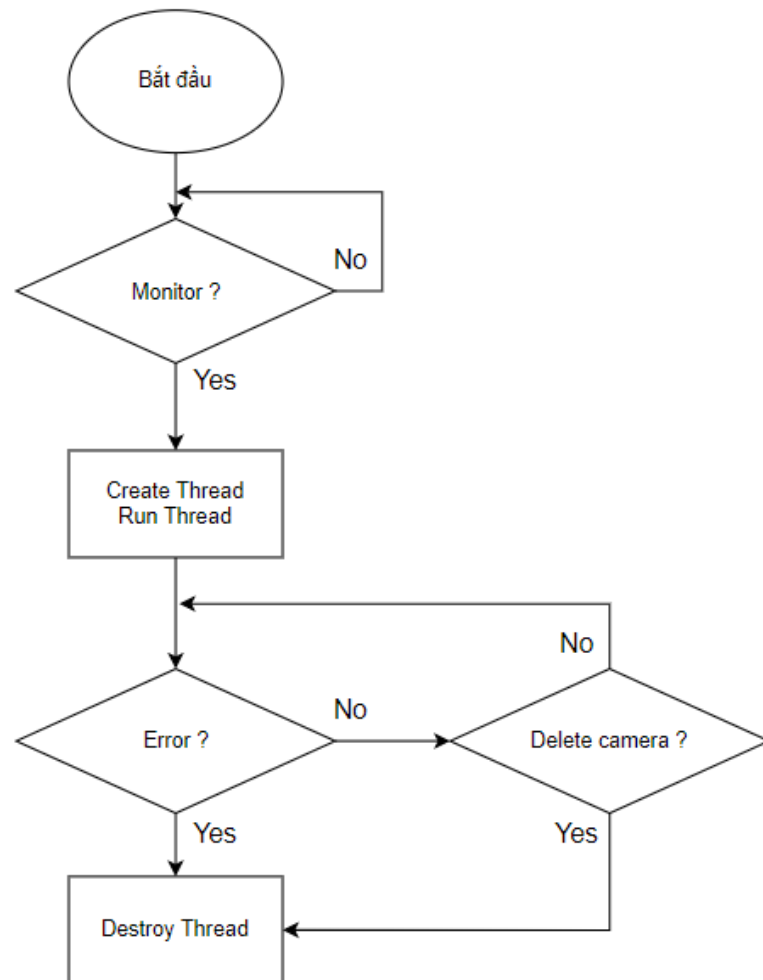
Hình 4.3 Màn hình chức năng add camera

- Chức năng MONITOR : Dùng để hiển thị thông tin từ camera, nơi camera quản lý, thông tin về trạng thái bãi đỗ xe hiện tại. Tuy nhiên hiện tại do vấn đề về cấu hình của laptop nên hiện tại ứng dụng của em chỉ có thể hiển thị tối đa là 2 camera ở 2 khu vực khác nhau.



Hình 4.4 Màn hình chức năng monitor

- Thông qua việc lập trình đa luồng, mỗi camera sẽ được quản lý bởi một luồng và các camera này hoạt động độc lập với nhau. Cụ thể cho giải thuật của một luồng camera. Khi màn hình (MONITOR) giám sát được bật. Một luồng camera kèm theo thuật toán yolov3 sẽ được tạo ra và chạy liên tục cho tới khi ta tắt màn hình hoặc có lỗi xảy ra trong quá trình hoạt động



Hình 4.5 Giải thuật của một luồng xử lý camera

4.3 Xây dựng nền tảng máy chủ Node.js

4.3.1 Giới thiệu Node.js

Node.js là một môi trường mã nguồn mở, được phát triển vào năm 2009 bởi Ryan Dahl để chạy các đoạn mã viết bằng JavaScript tại máy chủ, với nhiệm vụ giải quyết các vấn đề mà các nền tảng có thể gặp phải về hiệu suất như thời gian truyền thông mạng, thời gian xử lý các yêu cầu cũng như phản hồi web. Node.js là một nền tảng được xây dựng dựa trên Chrome's V8 JavaScript runtime, với mục đích xây dựng các ứng dụng mạng nhanh và có khả năng mở rộng. Node.js sử dụng event-driven, I/O non-

blocking để làm ứng dụng nhẹ và có hiệu năng cao hơn, phù hợp với những ứng dụng thời gian thực chạy trên các thiết bị phân tán.

Các đặc trưng của Node.js:

- Đơn luồng (single thread): Node.js chủ yếu được sử dụng để xây dựng các ứng dụng máy chủ Web. Node.js có thể xử lý nhiều kết nối đồng thời chỉ với một single-thread (đơn luồng). Điều này giúp hệ thống tốn ít RAM nhất và chạy nhanh nhất khi không phải tạo thread mới cho mỗi truy vấn như khi sử dụng PHP.
- Non-blocking: Sự khác biệt lớn nhất giữa Node.js và PHP là hầu hết các hàm chức năng của PHP đều bị block (các lệnh thực hiện chỉ sau khi các lệnh trước đó đã hoàn thành), trong khi các hàm chức năng trong Node.js được thiết kế non-blocking (các lệnh được thực thi song song mà không cần chờ một thao tác trước thực hiện xong). Điều này đồng nghĩa với việc Node.js có thể thực thi nhiều yêu cầu đồng thời, không lãng phí chu kỳ tuần hoàn (clock cycle) mà máy chủ có thể thực hiện. Cụ thể, khi một ứng dụng Node.js cần thực hiện một thao tác, nó sẽ gửi nhiệm vụ này đến vòng lặp sự kiện, tạo một hàm callback, sau đó tiếp tục xử lý các thao tác khác. Vòng lặp sự kiện sẽ theo dõi những thao tác này, thực thi hàm callback và khi thao tác hoàn thành sẽ trả lại kết quả cho ứng dụng.
- Hệ quản trị cơ sở dữ liệu: Node.js hỗ trợ kết nối tốt với các loại cơ sở dữ liệu khác nhau (Sqlite3, Postgresql, Oracle, SQL server,...) thông qua các module được hỗ trợ như Anydb, pg-promise, node-oracle,... Các module này đều có thể được cài đặt thông qua npm (node package manager) một cách đơn giản.

Hiện nay, Node.js đã trở nên rất phổ biến và thường được các nhà phát triển lựa chọn làm giải pháp phát triển backend. Có nhiều lý do cho sự phổ biến của Node.js và tại sao nên sử dụng Node.js để phát triển server-side :

- Hiệu năng cao: Với đặc điểm đơn luồng và non-blocking, máy chủ Node.js có thể phục vụ nhiều kết nối cùng lúc mà vẫn có thời gian phản hồi nhanh. Với cách tiếp cận sử dụng đơn luồng, Node.js cung cấp môi trường phát triển ứng dụng có hiệu suất cao, ít tốn kém và có khả năng mở rộng.

- Ngôn ngữ JavaScript thông dụng: Với sự phổ biến của JavaScript, cộng đồng nhà phát triển lớn và ngày càng tăng trưởng, hàng ngàn module có sẵn miễn phí, việc phát triển ứng dụng Node.js phía máy chủ sẽ đơn giản và mang lại nhiều hiệu quả hơn với nhà phát triển.
- Môi trường phát triển đơn giản, thời gian phát triển nhanh: Đối với Node.js, các nhà phát triển chỉ cần tải Node qua npm, sau đó có thể xây dựng ứng dụng luôn một cách dễ dàng. Node.js đơn thuần chạy trên các nền tảng mở như HTML và Javascript nên khi xây dựng ứng dụng Node.js, không cần cài thêm bất cứ phần mềm của bên thứ ba nào để có thể chạy được các ứng dụng.

Với những ưu điểm về tốc độ xử lý, hiệu năng hệ thống, sự đa dạng về các module mở rộng, là giải pháp tối ưu đối với những hệ thống thời gian thực, thời gian phát triển nhanh, sử dụng ngôn ngữ JavaScript thông dụng Node.JS được lựa chọn làm môi trường phát triển phía máy chủ

4.3.2 Giới thiệu RESTful Webservice

RESTFUL Webservice là các Web Service được viết dựa vào kiến trúc REST. REST đã được sử dụng rộng rãi thay thế cho các Web Service dựa trên SOAP và WSDL. RESTful Webservice nhẹ, dễ dàng mở rộng và bảo trì. REST định nghĩa các quy tắc kiến trúc để bạn thiết kế Web Service, chú trọng vào tài nguyên hệ thống, bao gồm các trạng thái tài nguyên được định dạng như thế nào và được truyền tải qua HTTP, và được viết bởi nhiều ngôn ngữ khác nhau. Nếu tính theo số dịch vụ mạng sử dụng, REST nổi lên trong vài năm qua như là một mô hình thiết kế dịch vụ chiếm ưu thế. Trong thực tế, REST đã có những ảnh hưởng lên và gần như thay thế SOAP và WSDL vì nó đơn giản và dễ sử dụng hơn rất nhiều.

Quy tắc để tạo ra một ứng dụng Webservice tuân thủ 4 quy tắc thiết kế cơ bản sau:

- Sử dụng các phương thức HTTP một cách rõ ràng.
 - Để tạo một tài nguyên trên server ta dùng phương thức POST.
 - Để truy xuất tài nguyên trên server ta dùng phương thức GET.
 - Để thay đổi tài nguyên trên server ta dùng phương thức PUT.
 - Để xóa tài nguyên trên server ta dùng phương thức DELETE.

- Một đặc điểm của REST là phi trạng thái (stateless), có nghĩa là nó không lưu giữ thông tin của client. Chẳng hạn bạn vừa gửi yêu cầu để xem trang thứ 2 của một tài liệu, và bây giờ bạn muốn xem trang tiếp theo. REST không lưu trữ thông tin rằng trước đó nó đã phục vụ bạn. Điều đó có nghĩa REST không quản lý phiên làm việc (Session)
- Hiển thị cấu trúc thư mục như các URIs. REST đưa ra một cấu trúc để người dùng có thể truy cập vào tài nguyên của nó thông qua các URL, tài nguyên ở đây là tất cả những cái mà bạn có thể gọi tên được (Video, ảnh, ...).
- Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai. Khi Client gửi một yêu cầu tới Webservice nó thường được truyền tải dưới dạng XML hoặc JSON và thông thường nhận về với hình thức tương tự. Đôi khi Client cũng có thể chỉ định kiểu dữ liệu nhận về mà nó mong muốn (JSON, XML,...) các chỉ định này được gọi là các kiểu MIME, nó được gửi kèm trên phần header của request

MIME-Type	Content-Type
JSON	application/json
XML	application/xml
XHTML	application/xhtml+xml

Hình 4.6 Các kiểu MIME phổ biến thường dùng với REST service

4.3.3 Thiết kế webserver

Để hoàn thiện mô hình giao tiếp với người dùng, ta sẽ thiết kế backend webserver triển khai dưới dạng RESTful API bằng cách sử dụng Node.js với ngôn ngữ lập trình JavaScript.

Xây dựng các API :

- Cho phép nhận dữ liệu thông tin trạng thái về vị trí đỗ xe ô tô từ mô hình đánh giá và lưu trữ vào cơ sở dữ liệu
- Đồng bộ hóa dữ liệu với ứng dụng di động trong thời gian thực
- Lưu trữ các thông tin về tọa độ vị trí đỗ xe vào cơ sở dữ liệu

4.4 Hệ quản trị cơ sở dữ liệu Sqlite3

Hệ thống sử dụng 3 bảng dữ liệu chứa các thông tin phục vụ quá trình vận hành của hệ thống gồm :

Bảng chứa thông tin về quản lý user :

Tên thuộc tính	Kiểu dữ liệu	Mô tả
AdminID	Text	ID của admin
username	Text	Tên đăng nhập
Email	Text	Email
password	Text	Mật khẩu
role	Integer	Vai trò

Hình 4.7 Bảng chứa thông tin về quản lý user

Bảng chứa thông tin về tọa độ của vị trí đỗ xe :

Tên thuộc tính	Kiểu dữ liệu	Mô tả
spaceID	Text	ID của ô đỗ xe
x_coordinate1	Integer	Tọa độ ô đỗ xe
y_coordinate1	Integer	Tọa độ ô đỗ xe
x_coordinate2	Integer	Tọa độ ô đỗ xe
y_coordinate2	Integer	Tọa độ ô đỗ xe
x_coordinate3	Integer	Tọa độ ô đỗ xe
y_coordinate3	Integer	Tọa độ ô đỗ xe
x_coordinate4	Integer	Tọa độ ô đỗ xe
y_coordinate4	Integer	Tọa độ ô đỗ xe

Hình 4.8 Bảng chứa thông tin về tọa độ của vị trí đỗ xe

Bảng chứa thông tin trạng thái về vị trí đỗ xe :

Tên thuộc tính	Kiểu dữ liệu	Mô tả
name	Text	Tên ô đỗ xe
location	Text	Địa điểm đặt camera
status	Boolean	Trạng thái hiện tại của vị trí đỗ xe
reserve_status	Boolean	Trạng thái đặt chỗ của ô đỗ xe

Hình 4.9 Bảng chứa thông tin trạng thái về vị trí đỗ xe

4.5 Kết luận

Trong chương này, ta đã hoàn thành việc thiết kế và xây dựng giao diện quản lý bãi đỗ xe đáp ứng các nhu cầu cơ bản của vấn đề mà một bãi đỗ xe thường gặp phải. Tiếp theo em sẽ trình bày một số kết quả thực hiện được, từ đó đánh giá hệ thống ưu nhược điểm đồng thời đưa ra hướng phát triển.

Chương 5 KẾT QUẢ THỰC HIỆN

5.1 Kết quả thực hiện

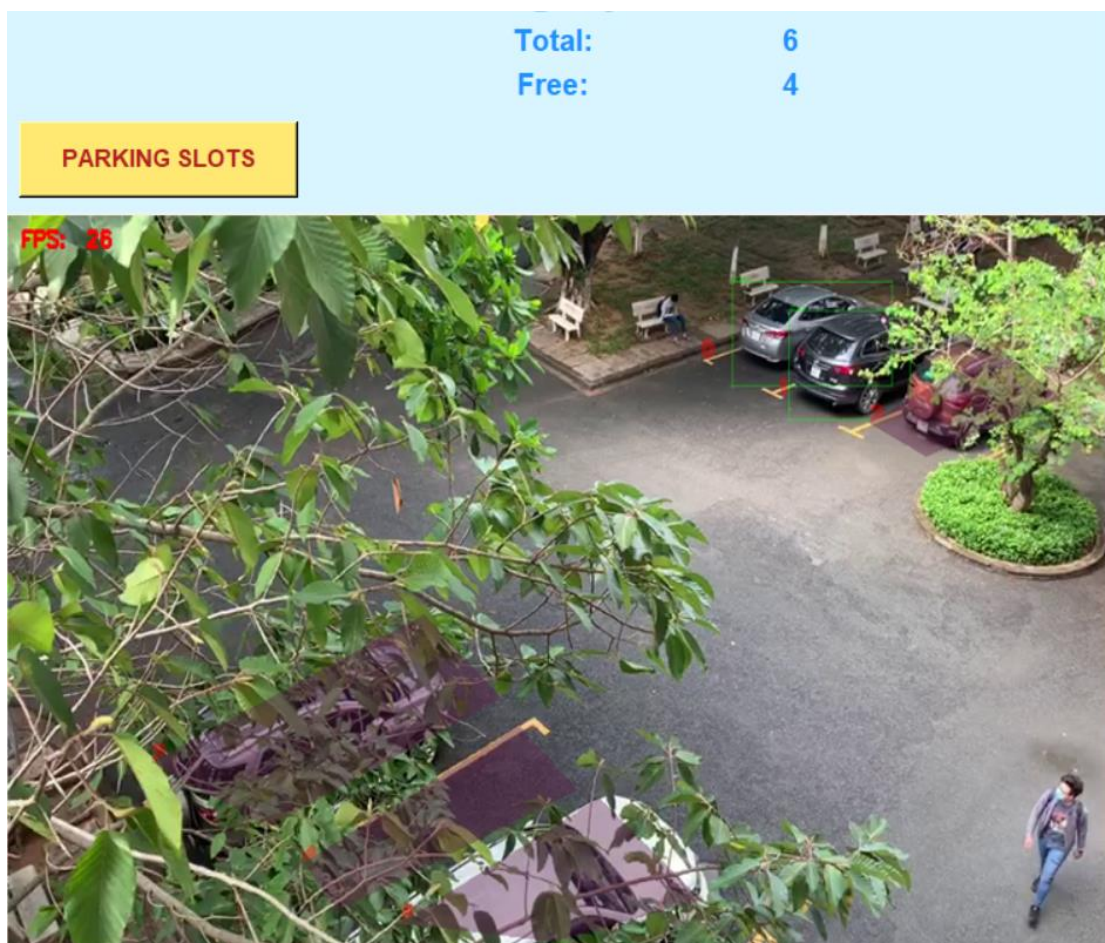
Sau khi đã hoàn tất quá trình xây dựng hệ thống trong chương 4, ta xem qua một số kết quả thực hiện được cũng như đi đánh giá lại toàn bộ hệ thống một cách trực quan

Môi trường thực hiện: Hệ thống được chạy trên máy tính Asus GL553VD với bộ vi xử lý Intel Core i7 thế hệ thứ 7, RAM DDR4 8GB, card rời NVIDIA GeForce GTX 1050

Đầu vào : Giả lập camera thông qua các luồng stream trên VLC

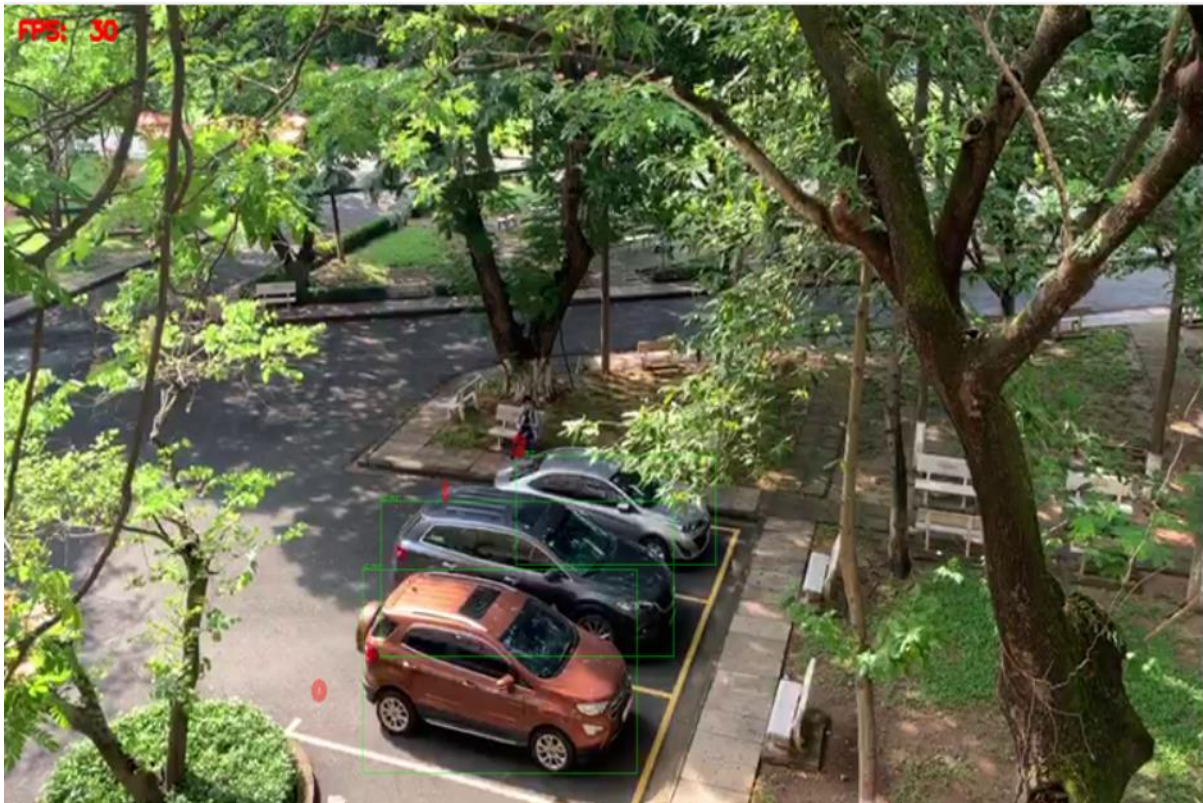
Một số kết quả thu nhận được ở các bãi xe khác nhau.

Tại bãi đỗ xe 1: Camera đặt tại vị trí có nhiều cây che phủ, vị trí camera ở trên cao và cho một hình ảnh xa. Ở những vị trí đỗ xe có tán cây che phủ hệ thống không phát hiện được xe dẫn đến sai sót rất lớn.



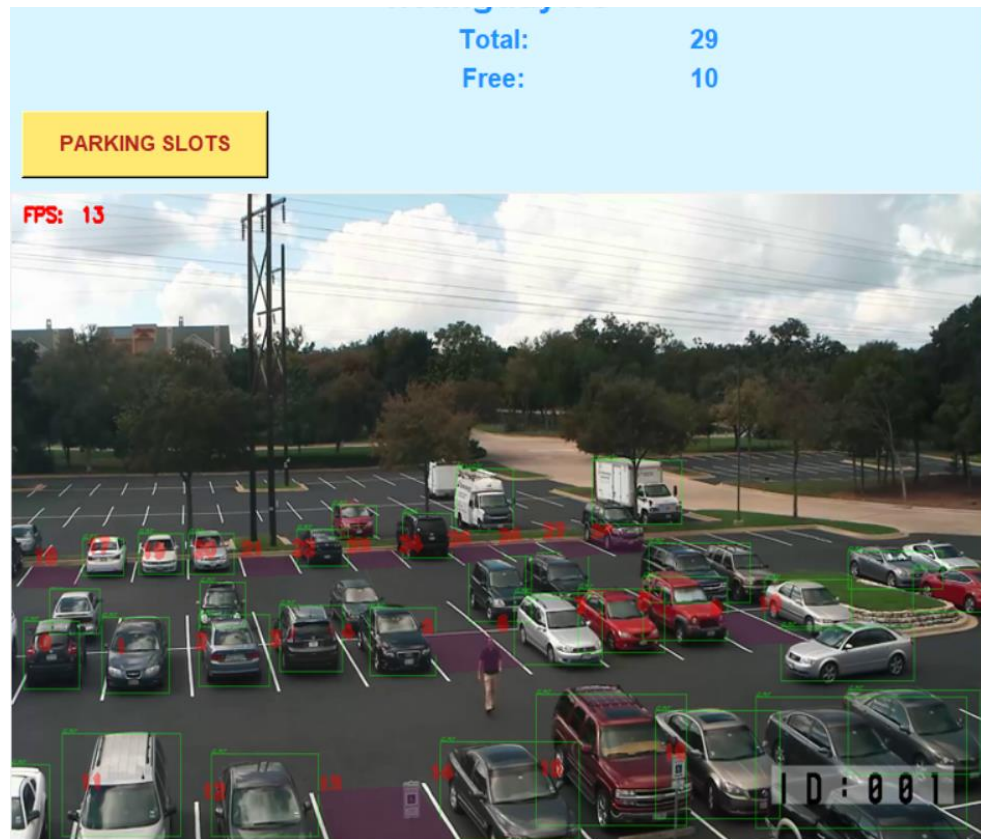
Hình 5.1 Một góc camera tại bãi đỗ xe 1

Tại góc camera này ta thấy được một phần của bãi đỗ xe tuy nhiên hình ảnh rõ nét và ít bị tán cây che phủ nên tình trạng phát hiện tốt, độ chính xác cao



Hình 5.2 Tại một góc camera khác tại bãi đỗ xe 1

Tại bãi đỗ xe 2 : Đây là một bãi đỗ xe lớn ngoài trời, góc camera bao quát toàn bộ bãi đỗ xe. Với việc quản lí 29 chỗ đỗ xe ta thấy được hệ thống hoạt động rất tốt với độ chính xác 100% tuy nhiên do việc quản lý quá nhiều vị trí đỗ xe dẫn đến tình trạng giảm tốc độ xử lý.



Hình 5.3 Tại bãi đỗ xe 2

5.2 Đánh giá hệ thống

Sau khi nhận được các kết quả ở một số bãi xe ta tiến hành việc đánh giá lại toàn bộ hệ thống:

- Trong điều kiện ánh sáng bình thường và vị trí đặt camera không bị che phủ hệ thống cho kết quả phát hiện trạng thái đỗ xe lên tới 95% trở lên
- Hoạt động ổn định trong điều kiện bãi đỗ xe ban đêm hệ thống cho kết quả chính xác lên tới 80% trở lên
- Giao diện tương tác người dùng dễ quản lý
- Cho phép thực hiện trên nhiều camera ở những khu vực khác nhau
- Tuy nhiên bên cạnh đó vẫn tồn tại nhiều nhược điểm về tốc độ xử lý camera. Nếu số lượng camera quá nhiều thì sẽ dẫn đến giảm tốc độ xử lý cũng như việc quản lý quá nhiều khu vực
- Việc đánh dấu vị trí đỗ xe một cách thủ công có thể đem đến nhiều vấn đề sai sót

- Tuy nhiên hệ thống vẫn chưa đóng gói thành một ứng dụng hoàn thiện, nên còn phải khởi động chương trình qua command line

Chương 6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Hiện nay, với các bãi đỗ xe thông thường dành cho ô tô thì mất khá nhiều thời gian ở khâu đi tìm vị trí đỗ xe, hoặc xảy ra tình trạng xe vào bãi nhưng không còn vị trí trống. Tình trạng này gây mất thời gian và phiền toái cho phương tiện cũng như người quản lý bãi xe. Xuất phát từ thực trạng này, em đã thiết kế và xây dựng lên mô hình nhận dạng trạng thái của bãi đỗ. Giải pháp sử dụng các camera giám sát trong bãi đỗ xe để nhận dạng vị trí đỗ xe. Mang lại hiệu quả cao cho các hoạt động ra vào tại bãi xe. Dự án mặc dù mới được phát triển nhưng đem lại kết quả khá chính xác trong việc nhận dạng vị trí. Tuy nhiên, cũng không thể tránh được những hạn chế khi bắt tay vào xây dựng một hệ thống mới như yếu tố ngoại cảnh có thể ảnh hưởng đến việc xác nhận sai của từng vị trí đỗ xe. Trong đề tài nghiên cứu đã đề xuất áp dụng các giải thuật mang tính ứng dụng công nghệ tiên tiến, hiện đại hiện nay nên đòi hỏi hệ thống trang thiết bị máy móc (máy tính) phải hỗ trợ tính năng GPU rất mạnh, trường hợp GPU yếu thì sẽ mất rất nhiều thời gian để hệ thống xử lý nhằm hoàn thành quá trình nhận dạng và phân loại đối tượng.

6.2 Hướng phát triển của luận văn trong tương lai

Từ những mục tiêu ban đầu đề ra, tuy gặp nhiều khó khăn thiếu sót trong quá trình làm luận văn nhưng qua quá trình học hỏi, cũng như nhờ sự hướng dẫn từ thầy Bùi Quốc Bảo để hoàn thành cơ bản các nội dung đề ra. Tuy nhiên để áp dụng được kết quả đề tài này vào thực tế cần gọt giũa một số mã nguồn chương trình đảm bảo tối ưu từ đó em xin đề xuất một số hướng phát triển cho luận văn trong tương lai :

- Thay đổi quy trình đánh dấu vị trí đỗ xe thủ công hiện tại bằng một cách tự động
- Xây dựng các ứng dụng trên điện thoại thông minh tương tác với người dùng với nhiều chức năng : đặt chỗ, thanh toán, thông báo trạng thái bãi đỗ xe,.
- Định vị GPS cho các bãi đỗ xe trong thành phố thực hiện hệ thống trên nhiều bãi đỗ xe và liên kết thành một hệ thống hoàn chỉnh
- Từ các thông tin về vị trí ô tô xe xây dựng hệ thống chỉ đường thông qua các bảng đèn led giúp giảm việc mất thời gian trong việc tìm kiếm vị trí đỗ xe

TÀI LIỆU THAM KHẢO

- [1] Đỗ Hữu Hiền, “NGHIÊN CỨU ỨNG DỤNG KỸ THUẬT MÁY HỌC ĐỂ PHÂN TÍCH HÌNH ẢNH VÀ NHẬN DẠNG PHƯƠNG TIỆN VỀ HÀNH VI VI PHẠM LUẬT GIAO THÔNG ĐƯỜNG BỘ”, Luận văn thạc sĩ Công nghệ thông tin, 2021
- [2] “Convolutional Neural Networks for visual Recognition”, CS231n
- [3] D. Thanh, “ mAP (mean Average Precision), dothanhblog, 24/4/2020
- [4] Adrian Rosebrock, “ How to use OpenCv’s ‘dnn’ module with NVIDIA GPUs, CUDA, and CuDNN”, pyimagesearch, 2/2020.
- [5] Jim Anderson, “An Intro to Threading in Python”, Real Python, 2018
- [6] Joseph Redmon, “YOLO: Real-Time Object Detection”, pjreddie
- [7] Giuseppe Amato, “Deep Learning for Decentralized Parking Lot Occupancy Detection”
- [8] V. H. Tiệp, “Bài 8: Gradient Descent”, Machine Learning Cơ bản, 2018
- [9] Thanh Tùng, “Khắc phục tình trạng thiếu bãi đỗ xe”, nhandan, 2020
- [10] Shuvashish Chatterjee, “Parking Slot Detection”, github, 2020