

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI TP. HỒ CHÍ MINH
BỘ MÔN CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: NGHIÊN CỨU TRIỂN KHAI THUẬT TOÁN YOLOV3
VÀ XÂY DỰNG HỆ THỐNG HƯỚNG DẪN ĐẠU XE Ô TÔ
PARKING VISION

Giảng viên hướng dẫn: ThS. PHẠM THỊ MIÊN

Sinh viên thực hiện : NGUYỄN VĂN HIỆP

Mã sinh viên : 5851071024

Lớp : CÔNG NGHỆ THÔNG TIN

Khoá : 58

TP. Hồ Chí Minh, tháng 06 năm 2021

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI TP. HỒ CHÍ MINH
BỘ MÔN CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: NGHIÊN CỨU TRIỂN KHAI THUẬT TOÁN YOLOV3
VÀ XÂY DỰNG HỆ THỐNG HƯỚNG DẪN ĐẠU XE Ô TÔ
PARKING VISION

Giảng viên hướng dẫn: ThS. PHẠM THỊ MIÊN

Sinh viên thực hiện : NGUYỄN VĂN HIỆP

Mã sinh viên : 5851071024

Lớp : CÔNG NGHỆ THÔNG TIN

Khoá : 58

TP. Hồ Chí Minh, tháng 06 năm 2021

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

BỘ MÔN: CÔNG NGHỆ THÔNG TIN

-----***-----

Mã sinh viên: 5851071024

Họ tên SV: Nguyễn Văn Hiệp

Khóa: 58

Lớp: Công Nghệ Thông Tin

1. Tên đề tài đồ án tốt nghiệp:

NGHIÊN CỨU TRIỂN KHAI THUẬT TOÁN YOLOV3 VÀ XÂY DỰNG HỆ THỐNG HƯỚNG DẪN ĐẠU XE Ô TÔ PARKING VISION.

2. Mục đích, yêu cầu:

a. Mục đích:

- Xây dựng website theo dõi các vị trí còn trống và đã đỗ trong bãi đỗ xe và hiển thị lên màn hình số vị trí còn trống, đã đỗ và tổng số lượng chỗ đỗ trong bãi đỗ xe ô tô theo thời gian thực.
- Xây dựng mô hình phát hiện các chỗ còn trống và đã đỗ trong bãi đỗ xe bằng hình ảnh, video, camera.
- Xây dựng website giới thiệu sản phẩm.
- Xây dựng giải pháp cho Hệ Thống Hướng Dẫn Bãi Đỗ Xe (Parking Guidance System) thông minh tiết kiệm, nhanh hơn, tiện dụng, chính xác và có thể mở rộng quy mô sau này.

b. Yêu cầu:

- Tìm hiểu về Thị Giác Máy Tính (Computer Vision) và Học Sâu (Deep Learning).

- Nghiên cứu về Mạng Neural Tích Chập (Convolution Neural Network) và những ứng dụng của nó trong Deep Learning.
- Nghiên cứu thuật toán YOLOv3 (You Only Look Once Version 3).
- Tìm hiểu về thuật toán hồi quy logistic.
- Thu thập dữ liệu hình ảnh về những chỗ trống và đã đỗ trong bãi đỗ xe. Gắn nhãn, tiền xử lý.
- Tìm hiểu về Transfer Learning, và ứng dụng vào huấn luyện mô hình.
- Ứng dụng kiến trúc Darknet huấn luyện mô hình trên tập dữ liệu lớn và tập dữ liệu mô phỏng bằng công cụ Google Colab.
- Ứng dụng thuật toán YOLOv3 (You Only Look Once version 3) để phát hiện những vị trí còn trống, đã đỗ trên hình ảnh, video, camera bằng Pytorch.
- Ứng dụng Nicepage thiết kế website giới thiệu sản phẩm và sử dụng Django để xử lý trang web, truyền hình ảnh đã xử lý lên website.

3. Nội dung và phạm vi đề tài:

a. Nội dung đề tài:

- Nghiên cứu và triển khai thuật toán YOLOv3 bằng Pytorch:
 - + Triển khai mô-đun phát hiện qua hình ảnh.
 - + Triển khai mô-đun phát hiện qua video/camera.
- Kiểm thử mô hình.
- Nghiên cứu các chỉ số đánh giá mô hình.
- Xây dựng trang web hiển thị màn hình xử lý bằng Django.
- Hiển thị song song màn hình xử lý trên website và trên desktop.

b. Phạm vi đề tài:

- Bài toán phát hiện đối tượng (Object Detection).
- Thuật toán YOLOv3.
- Ứng dụng Django để xây dựng website giới thiệu sản phẩm và sử dụng Django để xử lý trang web, truyền hình ảnh đã xử lý lên website.

4. Công nghệ, công cụ và ngôn ngữ lập trình:

a. Công nghệ:

Python, OpenCV, Pytorch, Django, Colab Notebook, CUDA, Darknet.

b. Công cụ:

- + Một số thư viện mã nguồn mở của Python:
Opencv-python, pandas, numpy, django, torch,...
- + Visual Studio Code
- + Darknet: Open Source Neural Networks
- + Google Colab

c. Ngôn ngữ lập trình: Python

5. Các kết quả chính dự kiến sẽ đạt được và ứng dụng:

- Sử dụng camera tiến hành phát hiện các vị trí còn trống và đã đỗ trong thời gian thực. Hiện thị lên màn hình vị trí còn trống và đã đỗ, đếm những vị trí còn trống, đã đỗ, tổng các vị trí hiện có.
- Sử dụng Django để xây dựng website.
- Mô phỏng mô hình trực tiếp.
- Hoàn chỉnh cuốn báo cáo đề tài.
- Nắm được kiến trúc thuật toán YOLOv3 và có thể ứng dụng vào mọi đề tài liên quan.
- Nắm được các ưu, nhược điểm của thuật toán và các phương pháp tối ưu cho thuật toán để cải thiện mô hình nhận diện.
- Nắm được những quy trình trong huấn luyện và kiểm tra mô hình trong các mô hình Deep Learning.

6. Giáo viên và cán bộ hướng dẫn

Họ tên: PHẠM THỊ MIÊN

Đơn vị công tác: Bộ môn Công Nghệ Thông Tin – Trường Đại học Giao thông Vận tải phân hiệu tại TP HCM

Điện thoại: 0961170638

Email: ptmien@utc2.edu.vn

Ngày ... tháng 03 năm 2021

Đã giao nhiệm vụ TKTN

BM Công Nghệ Thông Tin

Giáo viên hướng dẫn

ThS. Phạm Thị Miên

Đã nhận nhiệm vụ TKTN

Sinh viên: NGUYỄN VĂN HIỆP

Ký tên:

Điện thoại: 0973550322

Email: 5851071024@st.utc2.edu.vn

LỜI CẢM ƠN

Qua thời gian học tập và rèn luyện tại trường Trường Đại học Giao thông Vận tải phân hiệu tại TP HCM, đến nay chúng em đã kết thúc khoá học 4 năm và hoàn thành đồ án tốt nghiệp. Trong thời gian học tập tại trường để có được kết quả hiện tại em xin chân thành cảm ơn:

Cảm ơn tập thể các thầy cô giáo Bộ môn Công Nghệ Thông Tin và các thầy cô thỉnh giảng đã giảng dạy, quan tâm và tạo điều kiện thuận lợi để chúng em học tập rèn luyện trong suốt thời gian qua, giúp chúng em trang bị những kiến thức, kỹ năng cần thiết cho công việc thực tế sau này. Cảm ơn thầy cô giáo Bộ môn cũng như Ban Giám Hiệu đã cho phép em thực hiện đề tài tốt nghiệp: ***NGHIÊN CỨU TRIỂN KHAI THUẬT TOÁN YOLOV3 VÀ XÂY DỰNG HỆ THỐNG HƯỚNG DẪN ĐẠU XE Ô TÔ PARKING VISION.***

Và cảm ơn thạc sĩ Phạm Thị Miên đã luôn quan tâm nhiệt tình hướng dẫn, giúp đỡ em trong quá trình thực hiện đồ án tốt nghiệp. Cô cũng luôn nhắc nhở, động viên mỗi khi em gặp khó khăn, nhờ vậy mà em đã hoàn thành tốt đồ án tốt nghiệp của mình đúng thời hạn được giao. Nếu không có những lời hướng dẫn, dạy bảo của cô thì em nghĩ bài báo cáo này của em sẽ rất khó có thể hoàn thiện được.

Em cũng xin gửi lời cảm ơn tới gia đình, bạn bè, những người đã động viên, giúp đỡ em rất nhiều trong thời gian học tập và làm đồ án tốt nghiệp.

Mặc dù đã cố gắng nỗ lực học hỏi không ngừng để hoàn thành đề tài, vậy nhưng thời gian thực hiện đồ án có hạn, kiến thức của em còn hạn chế. Do vậy, không tránh khỏi những thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của thầy cô trong hội đồng bảo vệ đồ án tốt nghiệp để kiến thức của em được hoàn thiện hơn.

Đồng thời em xin cam đoan rằng nội dung đồ án của chính em nghiên cứu xây dựng nên, nếu có nội dung tham khảo đều được trích dẫn cụ thể, rõ ràng.

TP. Hồ Chí Minh, ngày 11 tháng 06 năm 2021

Sinh viên thực hiện

Nguyễn Văn Hiệp

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp. Hồ Chí Minh, ngày ... tháng 08 năm 2021

Giáo viên hướng dẫn

ThS. Phạm Thị Miên

MỤC LỤC

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP.....	i
LỜI CẢM ƠN.....	v
NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN	vi
MỤC LỤC	vii
DANH MỤC THUẬT NGỮ	xi
DANH MỤC HÌNH ẢNH.....	xiii
CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI.....	1
1.1 Sơ lược về Computer Vision và sự hỗ trợ của Deep Learning	1
1.1.1 Thị Giác Máy Tính (Computer Vision)	1
1.1.2 Đóng góp từ Deep Learning	1
1.1.3 Một số ứng dụng chính	2
1.2 Đặt vấn đề.....	2
1.2.1 Thực trạng	2
1.2.2 Các loại hình bãi đỗ xe phổ biến hiện nay	3
1.2.3 Hệ Thống Hướng Dẫn Bãi Đỗ Xe (Parking Guidance System - PGS)	4
1.2.4 Giải pháp mới sử dụng công nghệ Computer Vision và Deep Learning:.....	5
1.3 Tình hình nguyên cứu.....	6
1.4 Quá trình nguyên cứu	9
1.4.1 Các kiểu bãi đỗ xe ô tô thông dụng.....	9
1.4.2 Các trạng thái có thể có trong một chỗ đỗ xe ô tô	9
1.4.3 Các điều kiện bên ngoài ảnh hưởng đến chất lượng hình ảnh.....	9
1.5 Cấu trúc báo cáo đồ án tốt nghiệp	10
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	11

2.2 Giới thiệu về Mạng Neural và Mạng Neural lan truyền ngược.....	11
2.2.1 Định nghĩa.....	11
2.2.2 Cấu tạo Mạng Neural	11
2.2.3 Một số hàm kích hoạt phổ biến.....	12
2.2.4 Multi-layers Perceptron (MLP).....	13
2.2.3 Mạng lan truyền ngược (Backpropagation Network)	14
2.3 Mạng Neural Tích chập	15
2.3.1 Giới thiệu Mạng Neural Tích chập	15
2.3.2 Các thành phần của một Mạng Neural Tích chập.....	15
2.3.3 Mô hình Mạng Neural Tích chập.....	18
2.4 Hồi quy logistic (logistic regression)	19
2.4.1 Thiết lập bài toán trong phân loại ảnh	19
2.4.2 Xây dựng mô hình.....	20
2.7 Transfer Learning	24
2.8 Một số kiến trúc sử dụng	25
2.8.1 ResNet.....	25
2.8.2 Feature Pyramid Networks (FPN)	25
2.8.3 Darknet 53.....	26
CHƯƠNG 3. BÀI TOÁN PHÁT HIỆN ĐỐI TƯỢNG.....	28
3.1 Phân loại hình ảnh	28
3.2 Phân loại hình ảnh cùng với việc bản địa hóa.....	29
3.3 Phát hiện đối tượng.....	31
3.4 Lựa chọn mô hình object detection	32
3.5 YOLOv3	33
3.5.1 Giới thiệu YOLO	33

3.5.2 Kiến trúc của YOLOv3	34
3.5.3 Phát hiện đối tượng ở 3 tỷ lệ kích thước khác nhau	36
3.5.4 Anchor Boxes.....	39
3.5.5 Intersection Over Union (IOU).....	39
3.5.6 Bounding Box Prediction (dự đoán hộp giới hạn).....	40
3.5.7 Thresholding (ngưỡng)	41
3.5.8 Thuật toán Non-Maximum Suppression.....	41
3.5.9 Giải thích hàm mất mát.....	42
3.6 Một số chỉ số đánh giá bài toán phát hiện đối tượng.....	43
3.6.1 Precision và Recall.....	43
3.6.2 AP và mAP.....	44
CHƯƠNG 4. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH	46
4.1 Hiện trạng tổ chức	46
4.2 Yêu cầu hệ thống	46
4.2.1 Yêu cầu chức năng.....	46
4.2.2 Yêu cầu phi chức năng.....	46
4.3 Giải quyết bài toán phát hiện đối tượng bằng thuật toán YOLOv3	47
4.4 Huấn luyện mô hình	47
4.4.1 Thu thập dữ liệu	47
4.4.2 Gán nhãn cho ảnh.....	49
4.4.3 Cấu hình các tệp cần thiết	50
4.4.4 Huấn luyện mô hình.....	52
4.5 Triển khai thuật toán YOLOv3 bằng Pytorch và xử lý website	57
4.5.1 Triển khai thuật toán YOLOv3	57
4.5.2 Xử lý website	60

4.6 Sơ đồ use case của hệ thống	61
4.7 Sơ đồ hoạt động	62
4.7.1 Sơ đồ hoạt động truyền video lên website và Python App	63
4.7.2 Sơ đồ hoạt động xử lý ảnh / thư mục ảnh	64
4.8 Xây dựng giao diện chương trình	65
4.8.1 Giao diện trang chủ website – giới thiệu sản phẩm	65
4.8.2 Giao diện xem video camera theo thời gian thực	66
4.8.3 Giao diện thông tin liên hệ	67
KẾT LUẬN	68
Kết quả đạt được	68
Nhược điểm	68
Hướng phát triển	69
PHỤ LỤC	70
Phụ lục 1: Hướng dẫn cài đặt	70
Phụ lục 2: Hướng dẫn sử dụng	70
TÀI LIỆU THAM KHẢO	72

DANH MỤC THUẬT NGỮ

STT	THUẬT NGỮ	Ý NGHĨA TIẾNG VIỆT	TỪ VIẾT TẮT	GHI CHÚ
1		Cơ sở dữ liệu	CSDL	
2	Activation function	Hàm kích hoạt		
3	Anchor	Điểm neo		
4	Artificial Intelligence	Trí tuệ nhân tạo	AI	
5	Backbone	Lõi / xương sống		
6	Backpropagation	Lan truyền ngược		
7	Bias	Độ lệch		
8	Bounding box	Hộp giới hạn	bbox	
9	Central Processing Unit	Bộ Xử Lý Trung Tâm	CPU	
10	Classify/Regress	Phân loại/hồi quy		
11	Computer Vision	Thị Giác Máy Tính	CV	
12	Convolutional Neural Network	Mạng Neural Tích Chập	CNN	
13	Dataset	Tập dữ liệu		
14	Deep Learning	Học Sâu	DL	
15	Electronic Know Your Customer	Định Danh Khách Hàng Điện Tử	eKYC	
16	Extraction	Trích xuất		
17	Feature	Đặc trưng		
18	Feature map	Biểu đồ đặc trưng		
19	Feature Pyramid Networks	Mạng Kim Tự Tháp Đặc Trưng	FPN	
20	Frame	Khung hình		
21	Frames per second	Số hình ảnh xuất hiện trên khung hình trong một giây	FPS	
22	Framework	Bộ khung		
23	Fully Connected Layer	Lớp Kết Nối Đầy Đủ		
24	Fully Convolutional Network	Mạng Tích Chập Hoàn Toàn	FCN	
25	Gradient	Độ dốc		
26	Gradient Descent	Giảm độ dốc	GD	
27	Graphical Processing Unit	Bộ Xử Lý Đồ Họa	GPU	
28	Graphical User Interface	Giao diện đồ họa người dùng	GUI	
29	Ground truth	Thực sự/chân lý		

30	Image Processing	Xử lý ảnh	XLA	
31	Insight	Sự thật ngầm hiểu		
32	Internal Covariate Shift	Chuyển dịch nội hiệp phương sai	ICS	
33	Internet of Things	Internet Vạn Vật	IOT	
34	Intersection Over Union	Giao trong hợp	IOU	
35	Learning rate	Tốc độ học		
36	Machine Learning	Học Máy	ML	
37	Mean squared error	Sai số toàn phương trung bình	MSE	
38	Momentum	Gia tốc học		
39	Multi Layer Perceptron	Mạng Neural Nhiều Lớp	MLP	
40	Neural	Nơ-ron		
41	Object detection	Phát hiện đối tượng		
42	Offset	Độ lệch/khoảng trống		
43	Open source	Mã nguồn mở		
44	Open Source Computer Vision Library	Thư viện Thị giác Máy tính Nguồn mở	OpenCV	
45	Optical Character Recognition	Nhận Dạng Ký Tự Quang Học	OCR	
46	Overfitting	Quá khớp		
47	Package	Gói		
48	Picture Element	Điểm ảnh	pixel	
49	Predicted	Được dự đoán		
50	Pretrained model	Mô hình được đào tạo trước		
51	Real-time	Thời gian thực		
52	Responsive	Tương thích		
53	Single Shot Detector	Máy dò ảnh đơn	SSD	
54	Skip connection	Kỹ thuật bỏ qua kết nối		
55	Stride	Bước nhảy		
56	Training	Huấn luyện		
57	Transfer Learning	Học chuyển giao		
58	Weight	Trọng số		
59	You Only Look Once	Bạn Chỉ Nhìn Một Lần	YOLO	
60	You Only Look Once Version 3	Bạn Chỉ Nhìn Một Lần Phiên bản 3	YOLOv3	

DANH MỤC HÌNH ẢNH

Hình 1.1 Bãi đỗ xe truyền thống tại siêu thị Emart Gò Vấp TP. HCM. (Ảnh Internet)	3
Hình 1.2 Bãi đỗ xe tự động tại Đà Nẵng. (Ảnh Internet)	4
Hình 1.3 Minh họa cho hệ thống hướng dẫn bằng cảm biến. (Ảnh Internet)	5
Hình 1.4 Bãi đỗ xe góc chéo. (Ảnh Internet)	9
Hình 1.5 Bãi đỗ xe song song. (Ảnh Internet)	9
Hình 2.1 Cấu trúc của một neural nhân tạo được gán nhãn k	11
Hình 2.2 Các hàm kích hoạt phổ biến	13
Hình 2.3 Cấu trúc mạng đa lớp MLP	13
Hình 2.4 Bên trái là ma trận ảnh, bên phải là ma trận filter	15
Hình 2.5 Minh họa tích chập	16
Hình 2.6 Tích chập với bước nhảy (a) là 1 và (b) là 2	16
Hình 2.7 Minh họa padding khi thêm các số 0 vào biên của ảnh	16
Hình 2.8 Hàm max pooling với stride bằng 2	17
Hình 2.9 Minh họa quá trình phân loại ảnh bằng CNN	17
Hình 2.10 Minh họa dataset hồi quy logistic với 2 nhãn 0 và 1	19
Hình 2.11 Minh họa quá trình reshape ảnh thành vector	20
Hình 2.12 Mô tả quá trình cập nhật trọng số w	23
Hình 2.13 So sánh tương quan hiệu quả của model train từ đầu và transferred model	24
Hình 2.14 Sự khác biệt giữa một khối thông thường (trái) và một khối phần dư (phải)	25
Hình 2.15 Quan hệ tương quan giữa độ phân giải và giá trị ngưỡng	26
Hình 2.16 Thêm các skip connection giữa feature map và reconstruction layer	26
Hình 2.17 Kiến trúc của mô hình Darknet 53	27
Hình 3.1 Minh họa phân loại ảnh mèo	29
Hình 3.2 Minh họa phân loại và bản địa hoá ảnh	30
Hình 3.3 Minh họa quá trình phát hiện đối tượng	32
Hình 3.4 So sánh tốc độ của các mô hình phát hiện đối tượng tại thời điểm YOLOv3 mới ra mắt (2018)	33
Hình 3.5 Kiến trúc YOLOv3	35
Hình 3.6 Minh họa một ô chịu trách nhiệm phát hiện tại kích thước 13 x 13	38

Hình 3.7 Công thức tính IoU	39
Hình 3.8 Minh hoạ các dự đoán toạ độ bounding box dựa qua hộp neo.....	40
Hình 3.9 Hàm mất mát của YOLOv3.....	42
Hình 3.10 Giá trị AP là giá trị phía dưới đường biểu diễn Precision-Recall Curve.....	44
Hình 4.1 Một số hình ảnh trong tập dữ liệu PKLot Dataset.....	48
Hình 4.2 Dữ liệu mô phỏng tự thu thập.....	48
Hình 4.3 Gán nhãn bằng labelImg.....	49
Hình 4.4 Nội dung một tệp annotation	49
Hình 4.5 Nội dung trong tệp train.txt	50
Hình 4.6 Kết quả 9 điểm neo mới được sinh ra từ thuật toán k-mean	51
Hình 4.7 Trực quan hoá hàm mất mát sau gần 2000 batch của tập dữ liệu mô phỏng .	53
Hình 4.8 Trực quan hoá Region Avg IOU của tập dữ liệu mô phỏng	54
Hình 4.9 Các phương pháp data augmentation được sử dụng trong quá trình học	54
Hình 4.10 Kết quả tính toán $mAP@50 = 99.72\%$ trên 872 ảnh của tập dữ liệu lớn.....	55
Hình 4.11 Sơ đồ quá trình xử lý trong backbone	58
Hình 4.12 Output của trình phát hiện ảnh	59
Hình 4.13 Trình phát hiện đối tượng chạy trên Python App.....	60
Hình 4.14 Quá trình xử lý trong Django	61
Hình 4.15 Sơ đồ use case hệ thống.....	62
Hình 4.16 Sơ đồ hoạt động truyền ảnh lên website.....	63
Hình 4.17 Sơ đồ hoạt động xử lý ảnh / thư mục ảnh.....	64
Hình 4.18 Giao diện trang chủ	65
Hình 4.19 Giao diện giới thiệu sản phẩm dạng slide	65
Hình 4.20 Giao diện giới thiệu sản phẩm.....	66
Hình 4.21 Giao diện xem camera theo thời gian thực.....	66
Hình 4.22 Giao diện chú thích cho camera theo thời gian thực	66
Hình 4.23 Giao diện thông tin liên hệ	67

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1 Sơ lược về Computer Vision và sự hỗ trợ của Deep Learning

1.1.1 Thị Giác Máy Tính (Computer Vision)

Thị Giác Máy Tính (Computer Vision) đề cập đến toàn bộ quá trình mô phỏng tầm nhìn của con người trong một bộ máy phi sinh học. Điều này bao gồm việc chụp ảnh ban đầu, phát hiện và nhận dạng đối tượng, nhận biết bối cảnh tạm thời giữa các cảnh và phát triển sự hiểu biết ở mức độ cao về những gì đang xảy ra trong khoảng thời gian thích hợp. Trong thực tế, một hệ thống cung cấp khả năng Thị Giác Máy Tính đáng tin cậy, chính xác và trong thời gian thực là một vấn đề đầy thách thức vẫn chưa được phát triển một cách đầy đủ.

Khi các hệ thống này được phát triển hoàn chỉnh, sẽ có vô số ứng dụng dựa vào Thị Giác Máy Tính như một thành phần chính. Những ví dụ điển hình đó là xe hơi tự lái, robot tự động, máy bay không người lái, thiết bị hình ảnh y tế thông minh hỗ trợ phẫu thuật và cấy ghép phẫu thuật phục hồi thị lực của con người.

Có thể thấy Thị Giác Máy Tính mang nhiều hứa hẹn rất lớn trong tương lai, tuy vậy nó là một hệ thống phức tạp và luôn là thách thức đối với các hệ thống máy tính. Một phần của sự phức tạp là do thực tế Thị Giác Máy Tính không phải là một nhiệm vụ duy nhất. Thay vào đó, nó là một chuỗi các nhiệm vụ không đơn giản mà mỗi yêu cầu sử dụng các thuật toán phức tạp và đủ sức mạnh tính toán để hoạt động trong thời gian thực. Ở cấp độ cao, các tác vụ phụ của Thị Giác Máy Tính là phát hiện và phân đoạn đối tượng, phân loại hình ảnh, theo dõi đối tượng, gắn nhãn hình ảnh với các mô tả có ý nghĩa (ví dụ như chú thích hình ảnh) và cuối cùng, hiểu ý nghĩa của toàn bộ bối cảnh.

1.1.2 Đóng góp từ Deep Learning

Mặc dù vẫn còn những trở ngại đáng kể trong con đường phát triển của Thị Giác Máy Tính đến “cấp độ con người”, các hệ thống Deep Learning đã đạt được tiến bộ đáng kể trong việc xử lý một số nhiệm vụ phụ có liên quan. Lý do cho sự thành công này một phần dựa trên trách nhiệm bổ sung được giao cho các hệ thống Deep Learning. Điều hợp lý để nói rằng sự khác biệt lớn nhất với các hệ thống Deep Learning là chúng không còn cần phải được lập trình để tìm kiếm các đặc điểm cụ thể. Thay vì tìm kiếm các đặc

điểm cụ thể bằng thuật toán được lập trình cẩn thận, các mạng lưới Neural bên trong các hệ thống Deep Learning được đào tạo để có khả năng tự học. Ví dụ: nếu ô tô trong hình ảnh bị phân loại sai thành xe máy thì ta không tinh chỉnh các tham số hoặc viết lại thuật toán. Thay vào đó, ta tiếp tục đào tạo cho đến khi hệ thống làm cho đúng. [17]

1.1.3 Một số ứng dụng chính

- Phân loại hình ảnh
- Phân loại hình ảnh cùng với việc bản địa hóa
- Phát hiện đối tượng
- Tái thiết hình ảnh
- Theo dõi đối tượng

Những tiến bộ trong hệ thống Deep Learning và sức mạnh tính toán đã giúp cải thiện tốc độ, độ chính xác và độ tin cậy tổng thể của hệ thống Thị Giác Máy Tính. Khi các mô hình Deep Learning cải thiện và sức mạnh tính toán trở nên dễ dàng hơn, chúng ta sẽ tiếp tục đạt được những tiến bộ và sự ổn định đối với các hệ thống tự vận hành có thể thực sự nắm bắt và phản ứng với những gì chúng cảm nhận.

1.2 Đặt vấn đề

1.2.1 Thực trạng

Cùng với sự phát triển vượt bậc về kinh tế xã hội, nhu cầu của con người ngày càng được nâng cao, do đó nhu cầu lưu thông hàng hoá và những đòi hỏi về đi lại ngày càng tăng. Nếu như trên không trung, máy bay là phương tiện chiếm ưu thế, thì trên mặt đất, ô tô và vận tải ô tô lại chiếm ưu thế về năng lực vận chuyển và khả năng cơ động.

“Theo Sở Giao thông vận tải Hà Nội, trên địa bàn Hà Nội hiện có khoảng 6,9 triệu phương tiện giao thông (ô-tô, xe máy), chưa kể lượng xe ngoại tỉnh ra, vào thành phố hằng ngày. Dự tính, với tốc độ tăng trưởng xe máy 7,66%/năm; ô-tô 16,15%/năm thì đến năm 2025 Hà Nội có 1,3 triệu ô-tô và 7,3 triệu xe máy; năm 2030 có 1,7 triệu ô-tô và 7,7 triệu xe máy. Thế nhưng, điều đáng nói là hệ thống giao thông tĩnh (bãi, điểm đỗ xe công cộng) hiện nay còn quá thiếu và yếu, khiến nhu cầu đỗ xe ngày càng cấp thiết.” (theo báo nhandan.vn số ra ngày 09/03/2021). [18]

Thực trạng cho ta thấy cùng với sự tăng lên của các phương tiện giao thông chỉ ở Hà Nội, với các thành phố lớn thì cũng đang gặp vấn đề tương tự, ô tô đang có tốc độ tăng trưởng về số lượng khá cao về số lượng, do đó nhu cầu đỗ xe cũng ngày càng cao. Việc tìm được một chỗ đỗ xe trong thời buổi hiện nay cũng đang còn khá khó khăn. Đôi khi, có thể có chỗ trống trong bãi đỗ xe nhưng người sử dụng sẽ không biết nó nằm ở đâu, hoặc là không biết trong bãi có còn chỗ trống hay không. Rất nhiều vụ đỗ xe trước nhà dân, vỉa hè, lòng đường, công viên các nơi không phép... dẫn đến nhiều bất cập, ví dụ tình trạng tắc nghẽn giao thông và mất mỹ quan đô thị gây cản trở đường đi của chủ nhà, thậm chí nguy hiểm cho các phương tiện giao thông khác.

1.2.2 Các loại hình bãi đỗ xe phổ biến hiện nay

- **Bãi đỗ xe truyền thống:** Hệ thống đỗ xe ô tô truyền thống có ở hầu hết các gia đình, văn phòng và trung tâm giải trí. Không gian đỗ xe riêng biệt được tạo ra để vào và ra trơn tru để tăng tổ chức và giảm lưu lượng. Chúng đòi hỏi ít năng lượng hơn với yêu cầu năng lượng duy nhất là chiếu sáng, chức năng không bị cản trở bởi sự cố mất điện. Nhà để xe truyền thống đang hoạt động suốt ngày đêm, một số bãi đỗ xe truyền thống có không gian hạn chế nên người sử dụng phải tìm chỗ trống trước khi đỗ xe. Nếu chúng có thiết kế đơn giản, số vốn bỏ ra không lớn. Chiến lược bảo vệ xe hơi cũng có thể được cung cấp bằng cách giao nhiều lô cho các cá nhân cụ thể quản lý.



Hình 1.1 Bãi đỗ xe truyền thống tại siêu thị Emart Gò Vấp TP. HCM. (Ảnh Internet)

- **Hệ thống bãi đỗ xe tự động:** đòi hỏi ít diện tích hơn, loại bỏ trình điều khiển khỏi quá trình đỗ xe. Nó có thể là hoàn toàn tự động hoặc bán tự động. Chiếc xe phải được lái đến một điểm nhập cảnh nơi người lái và hành khách thoát khỏi xe. Sau đó, nó được di chuyển tự động hoặc bán tự động (với một số hỗ trợ cần thiết) đến không gian được phân bổ của nó. Hệ thống bãi đỗ xe tự động tối đa hóa không gian hạn chế, một lợi thế trong khu vực không gian hạn chế có sẵn.



Hình 1.2 Bãi đỗ xe tự động tại Đà Nẵng. (Ảnh Internet)

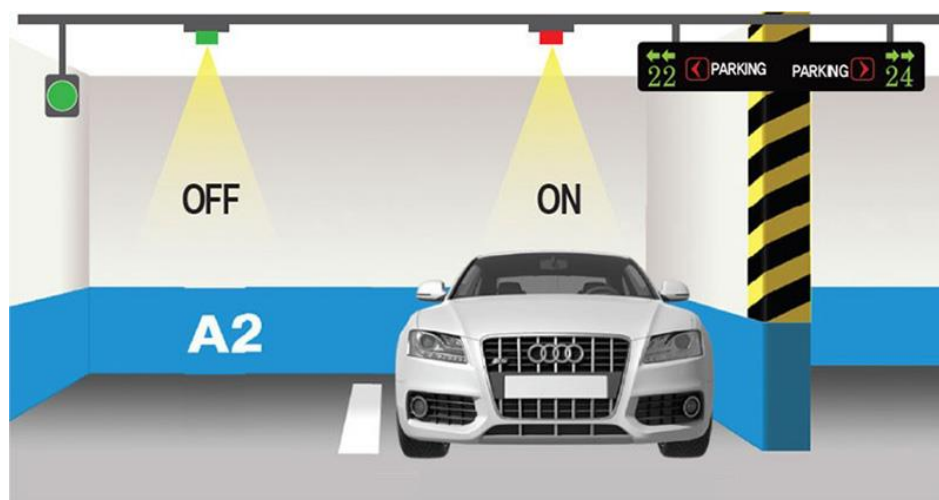
1.2.3 Hệ Thống Hướng Dẫn Bãi Đỗ Xe (Parking Guidance System - PGS)

Cho dù là bãi đỗ xe truyền thống hay tự động thì trong một bãi đỗ xe hoàn chỉnh không thể thiếu hệ thống hướng dẫn đỗ xe (Viết tắt là PGS), các loại hình phổ biến tại Việt Nam hiện nay:

- **Chỉ dẫn thủ công:** Đây là phương pháp chỉ dẫn truyền thống nhất, mọi công việc đều do con người trực tiếp tham gia vào làm, khi đó khi vào chỗ đỗ xe thì người sử dụng sẽ được một nhân viên trong bãi xe hướng dẫn đỗ xe, hoặc nhiều khi đó cũng chính là người ghi vé. Phương pháp này tốn rất nhiều nhân công và thiếu chính xác, ví dụ như một bãi đỗ xe lớn thì nhiều khi họ sẽ không nắm được số lượng chỗ trống còn lại và vị trí nào còn trống trong bãi để chỉ dẫn cho người sử dụng được. Hơn nữa còn mang lại

phiền toái khá nhiều cho người sử dụng khi phải mất thời gian tự đi tìm chỗ trống, dễ gây ra va chạm với các xe khác, tổn nhiên liệu tiêu thụ cho xe và tăng mức độ ô nhiễm môi trường, làm tâm trạng người sử dụng trở nên chán nản và bức bối, gây ùn ứ các phương tiện đi vào bãi, có thể làm giảm doanh thu của bãi xe.

- **Hệ thống hướng dẫn thông minh bằng cảm biến:** Hệ thống hướng dẫn đỗ xe thông minh cung cấp sự tiện lợi trong việc hướng dẫn đỗ xe một cách tự động cho bãi đỗ xe ô tô. Người sử dụng dễ dàng nhận biết vị trí còn trống tại từng khu vực trong bãi đỗ xe khi họ đi vào bãi. Tại mỗi vị trí xe đỗ được gắn một thiết bị siêu âm để phát hiện vị trí đỗ là có xe hay không có xe. Tình trạng của các vị trí báo cáo về trung tâm theo thời gian thực. Máy tính chủ sẽ thu thập tất cả trạng thái của từng vị trí từ bộ siêu âm và điều khiển các bảng quang báo hiển thị để hướng dẫn người sử dụng dễ dàng tìm đến vị trí còn trống để đỗ xe.



Hình 1.3 Minh họa cho hệ thống hướng dẫn bằng cảm biến. (Ảnh Internet)

Tuy nhiên giá mỗi cảm biến rất đắt đỏ, tốn khá nhiều chi phí triển khai, việc bảo trì cũng diễn ra khá khó khăn và tốn kém khi phải thuê nhân viên bên ngoài về vì yêu cầu tính chuyên môn cao, và giá linh kiện đắt đỏ, hơn nữa không phải lúc nào thì người sử dụng cũng biết là trong bãi còn chỗ trống hay không, đến khi vào nhưng hết chỗ thì cũng gây ra nhiều phiền toái. Hệ thống này chủ yếu được triển khai trong nhà có mái che, hoặc hầm.

1.2.4 Giải pháp mới sử dụng công nghệ Computer Vision và Deep Learning:

Hệ thống hướng dẫn đậu xe thông minh Parking Vision

Để giải quyết những nhược điểm của các phương pháp trên thì giải pháp được đề xuất để giải quyết cũng như là đề tài của đồ án này đó là *nguyên cứu triển khai thuật toán YOLOv3 và xây dựng hệ thống hướng dẫn đậu xe ô tô Parking Vision*. Ta sẽ sử dụng bài toán **phát hiện đối tượng** nằm trong lĩnh vực Computer Vision để phát hiện ra các vị trí đã đỗ và còn trống trong bãi, cùng với đó sử dụng công nghệ Deep Learning để huấn luyện và hỗ trợ cho Computer Vision trong việc kiểm tra dự đoán của mô hình. Một trong những sự kết hợp đó là thuật toán You Only Look Once (YOLO), là một mô hình CNN để phát hiện đối tượng mà một ưu điểm nổi trội là nhanh hơn nhiều so với những mô hình cũ. Thậm chí có thể chạy tốt trên những thiết bị IOT như Raspberry Pi¹. Khi sử dụng phương pháp này thì có những ưu điểm sau:

- **Đối với chủ đầu tư:** Giúp chủ đầu tư quản lý bãi giữ xe một cách chuyên nghiệp, khoa học với các công nghệ hiện đại, đồng thời giảm chi phí chủ đầu tư (bao gồm chi phí đầu tư máy chủ, tên miền, màn hình hiển thị và camera đối với giải pháp này), thích hợp triển khai ở những nơi ngoài trời như bãi đỗ xe của siêu thị, trường học, nhà hàng... có thể bao quát tầm nhìn của toàn bộ xe.

- **Đối với người sử dụng:** Thuận tiện cho người sử dụng, có thể nắm bắt số lượng chỗ trống và vị trí chính xác trong bãi, giúp chủ xe giảm thời gian tìm kiếm, tiết kiệm nhiên liệu, giúp tâm trạng thoải mái hơn thay vì tự mình tìm xem vị trí nào còn trống. Nhờ đó có thể biết trước bãi còn chỗ đỗ xe hay không, có thể tự tin đến nơi đỗ xe. Hơn nữa người sử dụng có thể theo dõi tình trạng xe mình ở mọi lúc mọi nơi, sử dụng như một camera an ninh.

- **Đối với người quản lý:** Thuận tiện cho người quản lý bãi xe, giúp cho họ nắm bắt thông tin số lượng chỗ trống hiện tại, chính xác vị trí còn trống để chỉ dẫn cho các tài xế vào bãi, tạo điều kiện thuận lợi cho việc điều hành các phương tiện di chuyển.

1.3 Tình hình nguyên cứu

Hiện nay ở Việt Nam phương pháp hướng dẫn ô tô dựa vào Computer Vision còn chưa phát triển nhiều, có lẽ vì ngành Computer Vision ở Việt Nam chưa được ứng dụng sâu và rộng, đây là một ngành mới nổi ở Việt Nam và có nhiều triển vọng phát triển, hiện tại chỉ được áp dụng nhiều vào các giải pháp nhận diện khuôn mặt, nhận diện ký tự

¹ Raspberry Pi là từ để chỉ các máy tính bo mạch đơn kích thước chỉ bằng một thẻ tín dụng.

- OCR. Những sản phẩm của các giải pháp đó có thể kể đến như: Định danh khách hàng
- eKYC, nhận dạng giấy tờ tùy thân, nhận dạng biển số xe. Nhưng công nghệ luôn phát triển không ngừng, và các bài toán đó đã được làm đi làm lại, cải tiến rất nhiều lần và cũng đã dần hoàn thiện, điều đó thúc đẩy mở ra các giải pháp khác cho chúng ta tìm hiểu. Ta nên có “insight” về xu thế phát triển công nghệ mới của tương lai và nghĩ ra giải pháp cho những vấn đề mới từ đó kiến tạo thế giới tiện dụng và tốt đẹp hơn.

Hiện tại dù YOLO đã ra tới phiên bản YOLOv5, nhưng phiên bản này đang còn nhiều tranh cãi về tốc độ cũng như độ chính xác, không phải do chính chủ tác giả phát hành. Phiên bản YOLOv4 cũng đã có rất nhiều cải tiến so với YOLOv3 nhưng hiện giờ YOLOv3 vẫn còn rất mạnh về cả tốc độ lẫn độ chính xác, hơn nữa với một người mới tìm hiểu về lĩnh vực này thì tôi chọn phiên bản YOLOv3 để tìm hiểu và triển khai, bởi vì phiên bản càng mới thì kiến trúc của thuật toán càng phức tạp, nếu quá phức tạp sẽ rất khó để tìm hiểu trong thời gian ngắn.

PyTorch là một package được xây dựng dựa trên Python để thay thế Numpy để tận dụng sức mạnh tính toán của GPU và cung cấp tính linh hoạt như một nền tảng phát triển Deep Learning. Pytorch là framework được phát triển bởi Facebook, Inc². Đây là một ông lớn về công nghệ đầu tư rất nhiều nguồn lực cho việc phát triển Trí Tuệ Nhân Tạo. Pytorch được phát triển với giấy phép mã nguồn mở do đó nó tạo được cho mình một cộng đồng rất lớn. Một cộng đồng lớn đồng nghĩa với nhiều tài nguyên để học và các vấn đề của ta có thể đã có ai đó giải quyết và chia sẻ với cộng đồng. Pytorch cùng với Tensorflow³ và Keras⁴ là một trong những framework phổ biến được sử dụng trong các bài toán về Deep Learning hiện nay. Đặc biệt, trong các lĩnh vực nghiên cứu, hầu như các tác giả đều sử dụng Pytorch để triển khai bài toán của mình.

Django là một trong số những web framework bậc cao miễn phí, là mã nguồn mở được tạo ra bởi ngôn ngữ Python dựa trên mô hình MTV (gồm Model-Template-Views). Hiện framework này được phát triển, quản lý bởi Django Software Foundation. Django ra đời với mục tiêu hỗ trợ thiết kế các website phức tạp dựa trên những CSDL có sẵn.

² Facebook, Inc. là một công ty truyền thông xã hội và công nghệ Mỹ có trụ sở tại Menlo Park, California.

³ TensorFlow là một thư viện phần mềm mã nguồn mở dành cho máy học trong nhiều loại hình tác vụ nhận thức và hiểu ngôn ngữ.

⁴ Keras là một thư viện phần mềm mã nguồn mở cung cấp giao diện Python cho các mạng nơ-ron nhân tạo.

Nó hoạt động dựa theo nguyên lý ‘cắm’ các thành phần và tái sử dụng để tạo nên các website với ít code, ít khớp nối, có khả năng phát triển và không bị trùng lặp. Lợi thế hàng đầu của Django là khả năng thiết kế, tạo lập website và các ứng dụng nhanh chóng. Vì vậy Django được sử dụng để thực hiện đồ án này.

Nicepage là một trình xây dựng trang web mới mẻ, khác biệt với các trình xây dựng phổ biến khác trên thị trường. Nó nhằm mục đích trả lại quyền tự do sáng tạo cho các nhà thiết kế web, với trình chỉnh sửa kéo và thả nâng cao cho phép nhà thiết kế web tự do sáng tạo tối đa.

OpenCV viết tắt cho Open Source Computer Vision Library. OpenCV là thư viện nguồn mở hàng đầu cho Computer Vision và Machine Learning, và hiện có thêm tính năng tăng tốc GPU cho các hoạt động theo real-time. OpenCV có một cộng đồng người dùng khá hùng hậu hoạt động trên khắp thế giới bởi nhu cầu cần đến nó ngày càng tăng theo xu hướng chạy đua về sử dụng Computer Vision của các công ty công nghệ.

Darknet là một framework open source chuyên biệt về phát hiện đối tượng được viết bằng ngôn ngữ C và CUDA. Các mô hình được huấn luyện trên Darknet nhanh hơn, đồng thời Darknet dễ cài đặt và hỗ trợ tính toán CPU và GPU. Cộng đồng sử dụng Darknet đông đảo, được hỗ trợ nhiệt tình. Để không tốn thời gian xây dựng lại mô-đun huấn luyện thuật toán YOLOv3 thì tôi sử dụng thư viện này để tận dụng những mã nguồn có sẵn để thực hiện huấn luyện mô hình song song với quá trình triển khai thuật toán, và tinh chỉnh các thông số trong lúc huấn luyện.

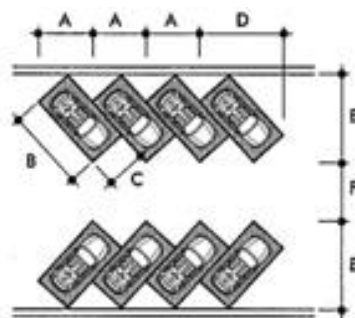
Google Colaboratory (gọi tắt là Google Colab hay Colab) là một sản phẩm của Google Research. Colab dựa trên Jupyter Notebook, người dùng có thể viết và thực thi đoạn mã python thông qua trình duyệt và đặc biệt rất phù hợp với Data Analysis, Machine Learning và giáo dục. Sự phát triển mạnh mẽ của Machine learning và Deep learning trong những năm gần đây không chỉ bởi các thuật toán, các mô hình tân tiến liên tiếp ra đời mà còn bởi sự phát triển không ngừng của phần cứng, đặc biệt là GPU. Như chúng ta đều biết, GPU rất đắt tiền. Ta có thể giải quyết vấn đề này bằng Colab, nó cung cấp một loạt các GPU miễn phí có sức mạnh to lớn về hiệu suất.

1.4 Quá trình nguyên cứu

Để hiểu rõ hơn về nghiệp vụ liên quan tới bãi đỗ xe, tôi sử dụng các mẫu bãi đỗ xe truyền thống và phổ biến nhất trên internet để tìm hiểu và triển khai giúp ích cho quá trình thu thập hình ảnh và đánh nhãn cũng như phát hiện sau này được dễ dàng hơn.

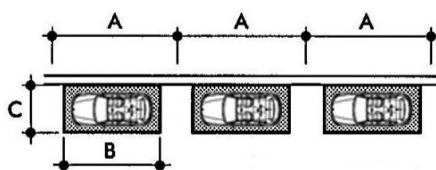
1.4.1 Các kiểu bãi đỗ xe ô tô thông dụng

- Bãi đỗ xe ô tô chéo góc 45 độ: [16]



Hình 1.4 Bãi đỗ xe góc chéo. (Ảnh Internet)

- Bãi đỗ xe ô tô song song:



Hình 1.5 Bãi đỗ xe song song. (Ảnh Internet)

1.4.2 Các trạng thái có thể có trong một chỗ đỗ xe ô tô

Chỉ có 2 trạng thái chính, đó là:

- Trống: Chỗ đó chưa có ô tô nào được đỗ
- Đã đỗ: Chỗ đó đã có ô tô đỗ

1.4.3 Các điều kiện bên ngoài ảnh hưởng đến chất lượng hình ảnh

- Thời tiết: Mưa, nắng
- Ánh sáng: Nắng đổ bóng, nắng gắt, nắng nhẹ, râm mát, trời tối bật đèn,...
- Các yếu tố gây nhiễu: Cây cối, bụi bặm bám trên camera,...

1.5 Cấu trúc báo cáo đồ án tốt nghiệp

Cấu trúc đồ án được chia thành các chương như sau:

Chương 1: Tổng quan đề tài – Giới thiệu tổng quan về đề tài đồ án tốt nghiệp.

Chương 2: Cơ sở lý thuyết

- Giới thiệu Mạng Neural và Mạng Neural lan truyền ngược.
- Giới thiệu Mạng Neural Tích Chập.
- Giới thiệu thuật toán hồi quy logistic.
- Giới thiệu về Transfer Learning.
- Giới thiệu một số kiến trúc mô hình được sử dụng.

Chương 3: Bài toán phát hiện vật thể

- Giới thiệu phân loại hình ảnh
- Giới thiệu phân loại và bản địa hoá hình ảnh.
- Giới thiệu phát hiện vật thể.
- Giới thiệu thuật toán YOLOv3.
- Giới thiệu các chỉ số đánh giá bài toán phát hiện đối tượng.

Chương 4: Triển khai thuật toán và thiết kế, xây dựng website

- Đưa ra cách giải quyết bài toán bằng YOLOv3.
- Thu thập dữ liệu, tiền xử lý, gán nhãn và huấn luyện thuật toán.
- Triển khai thuật toán YOLOv3 phát hiện qua hình ảnh, video, camera xem trực tiếp màn hình xử lý trên máy tính.
- Kiểm tra thuật toán.
- Xây dựng website giới thiệu sản phẩm và xem màn hình xử lý.

Kết luận

- Đưa ra kết quả đạt được, những thứ còn tồn tại và hướng phát triển về thuật toán lẫn website trong tương lai.

Phụ lục

Tài liệu tham khảo.

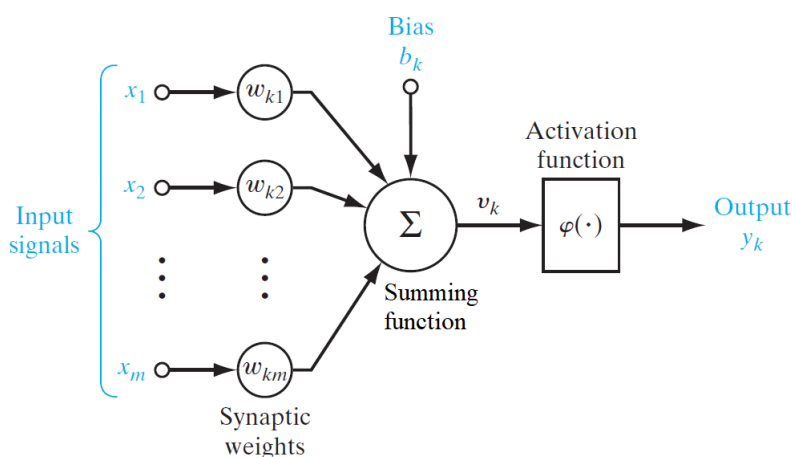
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.2 Giới thiệu về Mạng Neural và Mạng Neural lan truyền ngược

2.2.1 Định nghĩa

Mạng Neural Nhân Tạo, Artificial Neural Network (ANN) là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ neural sinh học. Nó được tạo nên từ một số lượng lớn các phần tử (neural) kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Một mạng neural nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu,...) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất học chính là quá trình hiệu chỉnh trọng số liên kết giữa các neural. Các neural đơn lẻ được gọi là các Perceptron.

2.2.2 Cấu tạo Mạng Neural



Hình 2.1 Cấu trúc của một neural nhân tạo được gán nhãn k

Các thành phần cơ bản của một neural nhân tạo bao gồm:

- Tập các đầu vào: Là các tín hiệu vào (input signals) của neural, các tín hiệu này thường được đưa vào dưới dạng một vector \mathbf{N} chiều.
- Tập các liên kết: Mỗi liên kết được thể hiện bởi một trọng số liên kết – Synaptic weight. Trọng số liên kết giữa tín hiệu vào thứ j với neural k thường được kí hiệu là w_{kj} . Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.

- Hàm tổng (Summing function) thường dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó.
- Bias (còn gọi là một độ lệch) là sự sai khác giữa trung bình dự đoán của mô hình chúng ta xây dựng với giá trị chính xác đang cố gắng để dự đoán.
- Một hàm kích hoạt (activation function) dùng để đưa các tín hiệu đầu ra của neural vào một miền giá trị nhất định hoặc vào một tập hợp các giá trị cố định.
- Đầu ra: Là tín hiệu đầu ra của neural, với mỗi neural sẽ có tối đa là một đầu ra.

Xét về mặt toán học, cấu trúc của một neural \mathbf{k} , được mô tả bằng biểu thức sau:

$$\begin{aligned}u_k &= \sum_{j=1}^m w_{kj} x_j \\v_k &= u_k + b_k \\y_k &= \varphi(v_k)\end{aligned}$$

Trong đó: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$: là các tín hiệu vào; $(\mathbf{w}_{k1}, \mathbf{w}_{k2}, \dots, \mathbf{w}_{km})$ là các trọng số liên kết của neural thứ \mathbf{k} ; \mathbf{u}_k là hàm tổng; \mathbf{b}_k là độ lệch; $\varphi(.)$ là hàm kích hoạt và \mathbf{y}_k là tín hiệu đầu ra của neural có nhãn \mathbf{k} .

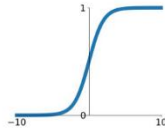
Cách thức kết nối các neural trong mạng xác định kiến trúc của mạng. Các neural trong mạng có thể kết nối đầy đủ (fully connected), hoặc kết nối cục bộ (partially connected).

2.2.3 Một số hàm kích hoạt phổ biến

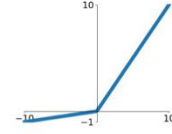
Hàm kích hoạt (activation function) mô phỏng tỷ lệ truyền xung qua axon của một neuron thần kinh. Trong một mạng nơ-ron nhân tạo, hàm kích hoạt đóng vai trò là thành phần phi tuyến tại output của các nơ-ron. Trong bài viết này, chúng ta sẽ cùng tìm hiểu các hàm kích hoạt phổ biến nhất và các ưu, nhược điểm của chúng.

Sigmoid

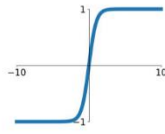
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

**Leaky ReLU**

$$\max(0.1x, x)$$

**tanh**

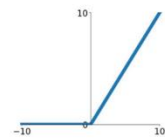
$$\tanh(x)$$

**Maxout**

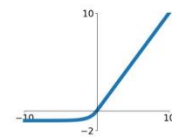
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



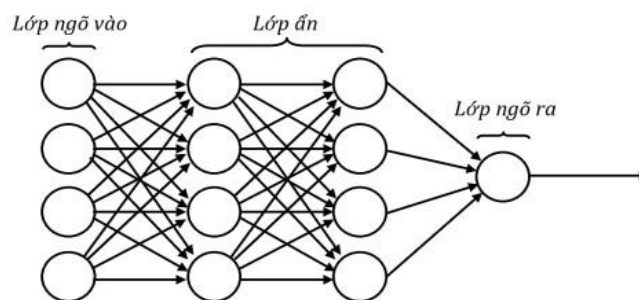
Hình 2.2 Các hàm kích hoạt phổ biến

2.2.4 Multi-layers Perceptron (MLP)

2.2.4.1 Giới thiệu mạng MLP

Mạng nhiều lớp truyền thẳng (MLP: Multi Layer Perceptron) là mạng có **n** ($n \geq 2$) lớp (thông thường lớp đầu vào không được tính đến): Trong đó gồm một lớp đầu ra (lớp thứ **n**) và (**n**-1) lớp ẩn, bao gồm:

- Lớp ngõ vào (input layer), cung cấp dữ liệu từ ngoài vào mạng.
- Lớp ngõ ra (output layer) có nhiệm vụ thực hiện tính toán và truyền thông tin ra bên ngoài.
- Lớp ẩn (hidden layer) thực hiện nhận thông tin từ các lớp phía trước, tính toán và chuyển thông tin đến các lớp phía sau.



Hình 2.3 Cấu trúc mạng đa lớp MLP

Hoạt động của mạng MLP như sau: Tại lớp ngõ vào các neural nhận tín hiệu vào xử lý (tính tổng trọng số, gửi tới hàm kích hoạt rồi cho ra kết quả (là kết quả của hàm kích hoạt); kết quả này sẽ được truyền tới các neural thuộc lớp ẩn thứ nhất; các neural

tại đây tiếp nhận như là tín hiệu đầu vào, xử lý và gửi kết quả đến lớp ẩn thứ 2, quá trình tiếp tục cho đến khi các neural thuộc lớp ra cho kết quả.

2.2.4.2 Học có giám sát trong các mạng neural:

Học có giám sát có thể được xem như việc xấp xỉ một ánh xạ: $\mathbf{X} \rightarrow \mathbf{Y}$, trong đó \mathbf{X} là tập các vấn đề và \mathbf{Y} là tập các lời giải tương ứng cho vấn đề đó. Các mẫu (\mathbf{x}, \mathbf{y}) với $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \in \mathbf{X}$, $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) \in \mathbf{Y}$ được cho trước. Học có giám sát trong các mạng neural thường được thực hiện theo các bước sau:

- Bước 1: Xây dựng cấu trúc thích hợp cho mạng neural, chẳng hạn có $(n + 1)$ neural vào (n neural cho biến vào và 1 neural cho ngưỡng \mathbf{x}_0), m neural đầu ra, và khởi tạo các trọng số liên kết của mạng.
- Bước 2: Đưa một vector \mathbf{x} trong tập mẫu huấn luyện \mathbf{X} vào mạng
- Bước 3: Tính vector đầu ra \mathbf{o} của mạng
- Bước 4: So sánh vector đầu ra mong muốn \mathbf{y} (là kết quả được cho trong tập huấn luyện) với vector đầu ra \mathbf{o} do mạng tạo ra; nếu có thể thì đánh giá lỗi.
- Bước 5: Hiệu chỉnh các trọng số liên kết theo một cách nào đó sao cho ở lần tiếp theo khi đưa vector \mathbf{x} vào mạng, vector đầu ra \mathbf{o} sẽ giống với \mathbf{y} hơn.
- Bước 6: Nếu cần, lặp lại các bước từ 2 đến 5 cho tới khi mạng đạt tới trạng thái hội tụ. Việc đánh giá lỗi có thể thực hiện theo nhiều cách, cách dùng nhiều nhất là sử dụng lỗi tức thời: $\mathbf{Err} = (\mathbf{o} - \mathbf{y})$, hoặc $\mathbf{Err} = |\mathbf{o} - \mathbf{y}|$; sai số toàn phương trung bình (MSE): $\mathbf{Err} = (\mathbf{o} - \mathbf{y})^2/2$.

Có hai loại lỗi trong đánh giá một mạng neural. Thứ nhất, gọi là lỗi rõ ràng (apparent error), đánh giá khả năng xấp xỉ các mẫu huấn luyện của một mạng đã được huấn luyện. Thứ hai, gọi là lỗi kiểm tra (test error), đánh giá khả năng tổng quát hóa của một mạng đã được huấn luyện, tức khả năng phản ứng với các vector đầu vào mới. Để đánh giá lỗi kiểm tra chúng ta phải biết đầu ra mong muốn cho các mẫu kiểm tra.

2.2.3 Mạng lan truyền ngược (Backpropagation Network)

Các neural ở các lớp trong thường được kết nối đầy đủ với tất cả các neural lớp ngoài, trên mỗi đường kết nối giữa 2 neural ở 2 lớp khác nhau có 1 trọng số mạng (weight). Các trọng số này đóng vai trò là các giá trị ẩn số mà mạng cần phải tìm ra

(học) sao cho với các giá trị đầu vào, thông qua mạng ta nhận được kết quả xấp xỉ với đầu ra mong muốn tương ứng của mẫu học.

2.3 Mạng Neural Tích chập

2.3.1 Giới thiệu Mạng Neural Tích chập

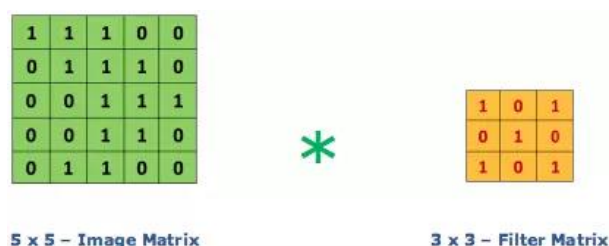
Những năm gần đây, ta đã chứng kiến được nhiều thành tựu vượt bậc trong ngành Thị giác máy tính (Computer Vision). Các hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như phát hiện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone (thiết bị bay không người lái) giao hàng tự động. Convolutional Neural Network (CNN – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

2.3.2 Các thành phần của một Mạng Neural Tích chập

2.3.2.1 Convolution Layer (Lớp tích chập)

Tích chập được sử dụng đầu tiên trong xử lý tín hiệu số (Signal processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kỹ thuật này vào xử lý ảnh và video số.

Để dễ hình dung, ta có thể xem tích chập như một cửa sổ trượt (sliding window) áp đặt lên một ma trận. Ta có thể theo dõi cơ chế của tích chập qua những hình ảnh minh họa bên dưới.

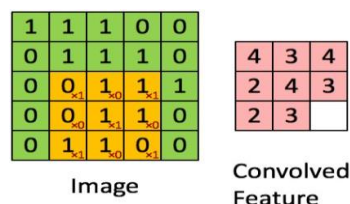


Hình 2.4 Bên trái là ma trận ảnh, bên phải là ma trận filter

Ma trận bên trái là một bức ảnh đen trắng. Mỗi giá trị của ma trận tương đương với một điểm ảnh (pixel), 0 là màu đen, 1 là màu trắng.

Sliding window còn có tên gọi là kernel, filter hay feature detector. Ở đây, ta dùng một ma trận filter 3×3 nhân từng thành phần tương ứng (element-wise) với ma trận

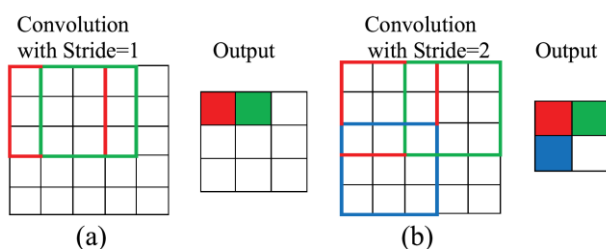
ảnh bên trái. Giá trị đầu ra do tích của các thành phần này cộng lại. Kết quả của tích chập là một ma trận (convolved feature) sinh ra từ việc trượt ma trận filter và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái. Như hình minh hoạ bên dưới:



Hình 2.5 Minh họa tích chập

Stride (bước nhảy)

Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy. Hình dưới là lớp tích chập hoạt động với stride 1 và stride 2.

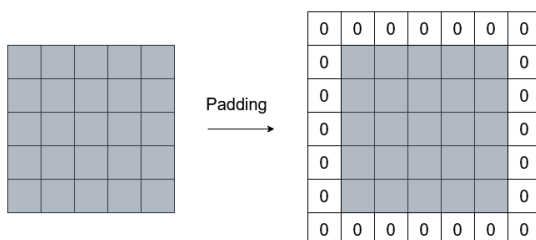


Hình 2.6 Tích chập với bước nhảy (a) là 1 và (b) là 2

Padding (đường viền)

Đôi khi kernel không phù hợp với hình ảnh đầu vào. Ta có 2 lựa chọn:

- Chèn thêm các số 0 vào 4 đường biên của hình ảnh (padding).
- Cắt bớt hình ảnh tại những điểm không phù hợp với kernel.



Hình 2.7 Minh họa padding khi thêm các số 0 vào biên của ảnh

Feature map (biểu đồ đặc trưng):

Thể hiện kết quả mỗi lần ma trận filter quét qua input. Mỗi lần quét như thế sẽ xảy ra quá trình tính toán.

2.3.2.2 Nonlinear activation function (hàm kích hoạt phi tuyến)

Trong một mạng nơ-ron nhân tạo, hàm kích hoạt đóng vai trò là thành phần phi tuyến tại output của các nơ-ron.

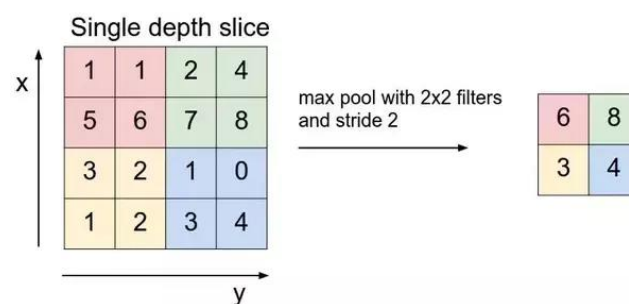
Có thể xem phần 3.1.3 *Một số hàm kích hoạt phổ biến* để biết thêm chi tiết vì chức năng như nhau.

2.3.2.3 Pooling Layer (lớp tổng hợp)

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau:

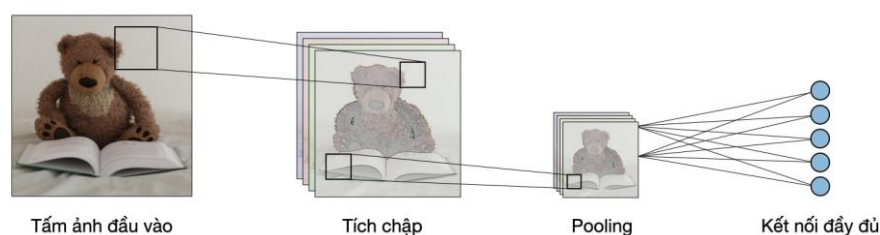
- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình (average pooling), tổng tất cả các phần tử trong map gọi là sum pooling.



Hình 2.8 Hàm max pooling với stride bằng 2

2.3.2.4 Fully connected layer (lớp kết nối đầy đủ)



Hình 2.9 Minh họa quá trình phân loại ảnh bằng CNN

Dùng để đưa ra kết quả. Ví dụ, sau khi các lớp convolutional layer và pooling layer đã nhận được các ảnh đã truyền qua nó, thì lúc đó ta sẽ thu được kết quả là mô hình đã đọc được khá nhiều thông tin về ảnh. Vì vậy, để liên kết các đặc điểm đó lại và cho ra output chúng ta dùng fully connected layer.

2.3.3 Mô hình Mạng Neural Tích chập

Mạng CNN gồm nhiều lớp Convolution chồng lên nhau, sử dụng các hàm kích hoạt phi tuyến để kích hoạt các trọng số. Mỗi một lớp sau khi được kích hoạt sẽ cho ra kết quả trừu tượng cho các lớp tiếp theo. Mỗi layer kế tiếp chính là thể hiện kết quả của layer trước đó. Thông qua quá trình training, các lớp CNN tự động học các giá trị được thể hiện qua các lớp filter.

CNN có tính bất biến và tính kết hợp cục bộ (Location Invariance and Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các góc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể. Pooling layer sẽ cho ta tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling).

Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNN cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các đối tượng trong tự nhiên, ta phân biệt được một con chó với một con mèo nhờ vào các đặc trưng từ mức độ thấp (có 4 chân, có đuôi) đến mức độ cao (dáng đi, hình thể, màu lông).

Cấu trúc cơ bản của CNN gồm 3 phần chính: Local receptive field, shared weights and bias, pooling:

- **Local receptive field:** hay còn gọi là các trường cục bộ. Tác dụng của lớp này chính là nó giúp chúng ta tách lọc các dữ liệu, thông tin của ảnh và chọn được những vùng ảnh có giá trị sử dụng nhất.

- **Shared weights and bias** (trọng số và độ lệch chia sẻ): Làm giảm tối đa số lượng các tham số là tác dụng chính của yếu tố này trong mạng CNN hiện nay. Bởi

trong mỗi convolution có những feature map khác nhau, mỗi feature map lại giúp nhận diện một vài feature trong ảnh. [6]

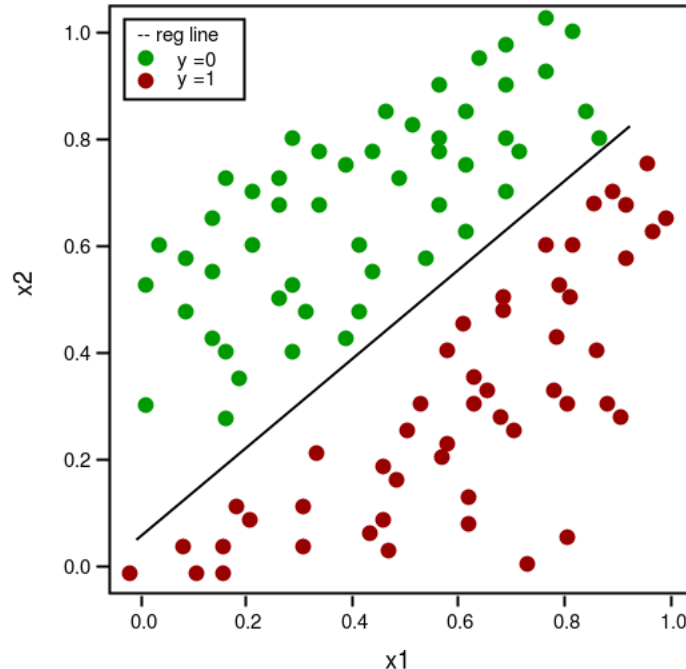
- **Pooling layer:** Như đã tìm hiểu ở phần **2.3.2.3 Pooling Layer (lớp tổng hợp)**.

2.4 Hồi quy logistic (logistic regression)

Hồi quy logistic là một phương pháp phân tích thống kê được sử dụng để dự đoán giá trị dữ liệu dựa trên các quan sát trước đó của tập dữ liệu, thuộc thuật toán học có giám sát, đây là thuật toán đơn giản nhưng lại rất hiệu quả trong bài toán phân loại (Classification), được áp dụng trong bài toán phân loại nhị phân (Binary classification) tức ta sẽ có hai output, hoặc có thể gọi là hai nhãn. [9]

Mục đích của hồi quy logistic là ước tính xác suất của các sự kiện, bao gồm xác định mối quan hệ giữa các tính năng từ đó dự đoán xác suất của các kết quả, nên đối với hồi quy logistic ta sẽ có:

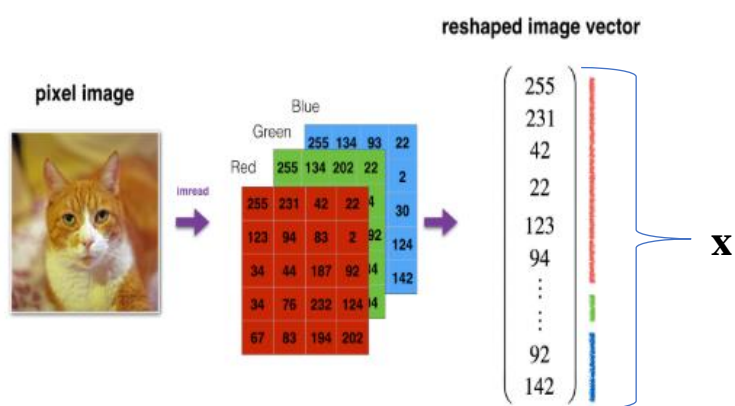
- **Input:** dữ liệu input (ta sẽ coi có hai nhãn là 0 và 1).
- **Output:** Xác suất dữ liệu input rơi vào nhãn 0 hoặc nhãn 1.



Hình 2.10 Minh họa dataset hồi quy logistic với 2 nhãn 0 và 1

2.4.1 Thiết lập bài toán trong phân loại ảnh

Trong bài toán này sẽ có hai nhãn là 0 và 1, input sẽ là bức ảnh. Như ta đã biết một bức ảnh màu sẽ có ba kênh màu là R, G, B với một bức ảnh có kích cỡ là $w * h$ thì sẽ có tất cả $w * h * 3$ điểm ảnh (nhân với 3 vì có ba kênh màu), số điểm ảnh chính là số feature của vector đầu vào \mathbf{x} . Ở hình dưới ta có thể thấy từ một bức ảnh màu ta sẽ reshape về một vector cột \mathbf{x} để làm input cho mô hình logistic.



Hình 2.11 Minh họa quá trình reshape ảnh thành vector

Với mỗi bức ảnh ta sẽ có một input \mathbf{x} nên với \mathbf{m} bức ảnh ta sẽ có \mathbf{m} input, việc lấy \mathbf{m} bức ảnh làm input là bởi vì bài toán học máy có giám sát cần nhiều dữ liệu đầu vào để huấn luyện mô hình, một bức ảnh là quá ít để cho mô hình có thể học được.

2.4.2 Xây dựng mô hình

2.4.2.1 Kí hiệu

Xét trên một điểm dữ liệu ta có input $\mathbf{x} = [x_1; x_2; \dots; x_n]$ sẽ là một vector cột, ta sẽ ngăn cách các feature (x_i) bằng dấu “;”. input \mathbf{x} sẽ có dạng như bên dưới:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Xét trên toàn bộ tập dữ liệu (\mathbf{m} điểm dữ liệu) ta sẽ có một vector hàng $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}]$ với mỗi cột là một điểm dữ liệu \mathbf{x}^i ngăn cách nhau bởi dấu “,”:

$$\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}] = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(m)} \end{bmatrix}$$

Ta có thể thấy \mathbf{X} có được bằng cách xếp thành cột các các $\mathbf{x}^{(i)}$.

Bộ trọng số $\mathbf{w} = [\mathbf{w}_1; \mathbf{w}_2; \dots; \mathbf{w}_n]$ cũng sẽ là một vector cột, ta sẽ ngăn cách các trọng số \mathbf{w}_i bằng dấu “;” bộ trọng số \mathbf{w} sẽ có dạng như dưới:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Ta có chuyển vị của vector \mathbf{w} sẽ là một vector hàng, các \mathbf{w}_i ngăn cách nhau bằng dấu “,”:

$$\mathbf{w}^T = [w_1, w_2, \dots, w_n]$$

Xét trên một điểm dữ liệu ta đặt $\mathbf{z} = \mathbf{w}^T \mathbf{x} + \mathbf{b}$ (\mathbf{b} gọi là bias), và $\mathbf{a} = \sigma(\mathbf{z})$:

$$\begin{aligned} z &= \mathbf{w}^T \mathbf{x} + b \\ a &= \sigma(z) \end{aligned}$$

Xét trên toàn bộ tập dữ liệu ta đặt $\mathbf{Z} = [\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(m)}]$ và $\mathbf{A} = [\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(m)}]$, đây đều là các vector hàng với \mathbf{m} là số điểm dữ liệu:

$$\begin{aligned} \mathbf{Z} &= [z^{(1)}, z^{(2)}, \dots, z^{(m)}] \\ \mathbf{A} &= [a^{(1)}, a^{(2)}, \dots, a^{(m)}] \end{aligned}$$

2.4.2.2 Sigmoid function

Hồi quy logistic sử dụng hàm kích hoạt Sigmoid để kiểm soát đầu ra.

Có thể xem phần **2.2.3 Một số hàm kích hoạt phổ biến** để biết chi tiết.

Tính chất:

- Là một hàm số liên tục và nhận giá trị trong khoảng (0;1)
- Chính vì là hàm liên tục nên hàm sigmoid sẽ có đạo hàm tại mọi điểm.

Đạo hàm:

$$\frac{\partial \sigma(x)}{\partial x} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left(\frac{1}{1 + e^{-x}} \right) \left(1 - \frac{1}{1 + e^{-x}} \right) = (\sigma(x))(1 - \sigma(x))$$

2.4.2.3 Mô hình

Quan hệ giữa nhiều biến độc lập \mathbf{x}_i với biến mục tiêu (biến phụ thuộc), phương trình tổng quát để ước lượng biến mục tiêu \mathbf{y} :

$$E(y) = w_n x_n + w_{n-1} x_{n-1} + \dots + w_1 x_1 + w_0 = \mathbf{w}^T \mathbf{x} + w_0$$

Trong đó: \mathbf{w}_0 là giá trị ước lượng của y khi tất cả các x_i đều đạt giá trị 0.

$E(y)$ trong phương trình logistic regression là xác suất để kết luận giá trị của biến y không phải giá trị thực của biến y :

$$E(y) = P(y = 0 | 1 | x_1, x_2, \dots, x_n)$$

Áp dụng hàm sigmoid để chuyển giá trị $\mathbf{w}^T \mathbf{x} + w_0$ thành xác suất để kết luận giá trị của biến y từ đó để xác định được nhãn của input \mathbf{x} :

$$P = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + w_0)}}$$

Đây chính là mô hình của logistic regression.

2.4.2.4 Loss function (hàm mất mát):

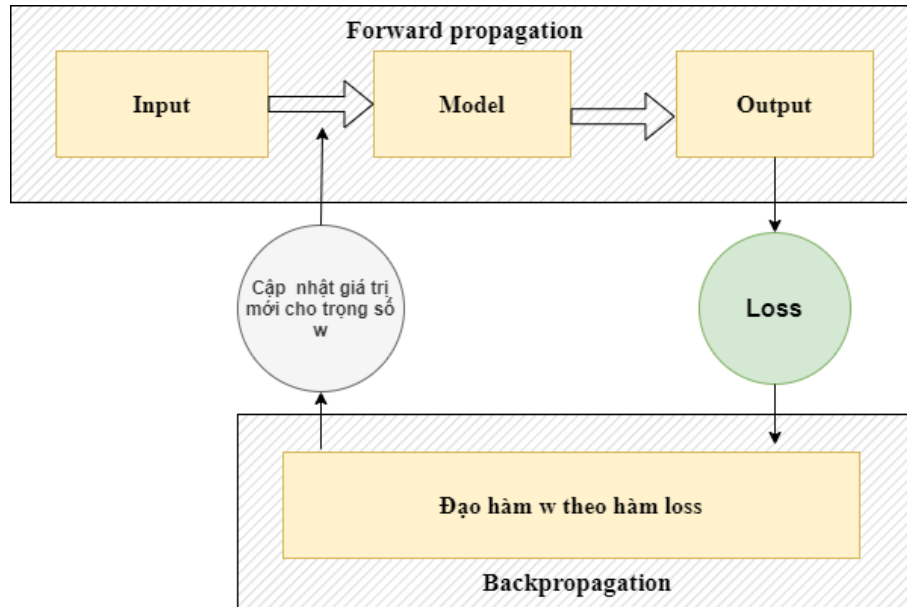
Hàm mất mát trong thuật toán học có giám sát (supervised learning) nói chung là hàm số để đo sự sai khác giữa giá trị thực tế và giá trị mô hình dự đoán, sự sai khác có thể sai khác về giá trị thực (như trong hồi quy tuyến tính) hoặc sai khác về nhãn (như trong bài toán phân loại) việc định nghĩa sự sai khác như nào là tùy vào từng bài toán cụ thể để từ đó xây dựng hàm mất mát.

Logistic regression sử dụng hàm mất mát **Binary Cross-Entropy**:

$$\mathcal{L}(\mathbf{w}) = - \sum_{i=1}^m (y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log (1 - a^{(i)}))$$

2.4.2.5 Backpropagation (lan truyền ngược):

Ngược với Forward propagation là Backpropagation, đây là quá trình tính toán ngược trong thuật toán. Ngược ở đây có nghĩa là chúng ta sẽ từ đi từ loss function để điều chỉnh bộ trọng số w nhằm giảm được giá trị loss ở các lần tính toán xuôi tiếp theo.



Hình 2.12 Mô tả quá trình cập nhật trọng số w

Xét tại một điểm dữ liệu, ta có hàm mất mát như sau:

$$J(\mathbf{w}) = -(y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log (1 - a^{(i)}))$$

Ta có đạo hàm:

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= - \left(y^{(i)} \frac{\mathbf{x}^{(i)} a^{(i)} (1 - a^{(i)})}{a^{(i)}} + (1 - y^{(i)}) \frac{-\mathbf{x}^{(i)} a^{(i)} (1 - a^{(i)})}{1 - a^{(i)}} \right) \\ &= - (y^{(i)} \mathbf{x}^{(i)} (1 - a^{(i)}) - \mathbf{x}^{(i)} a^{(i)} (1 - y^{(i)})) \\ &= - (\mathbf{x}^{(i)} (y^{(i)} - a^{(i)})) \\ &= \mathbf{x}^{(i)} (a^{(i)} - y^{(i)}) \end{aligned}$$

Vậy ta có công thức cập nhật cho \mathbf{w} :

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w} - \eta \mathbf{x}^{(i)} (a^{(i)} - y^{(i)})$$

Nếu biểu diễn tổng quát cho toàn bộ dữ liệu ta có:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\mathbf{X}(\mathbf{A} - \mathbf{Y})^T}{m}$$

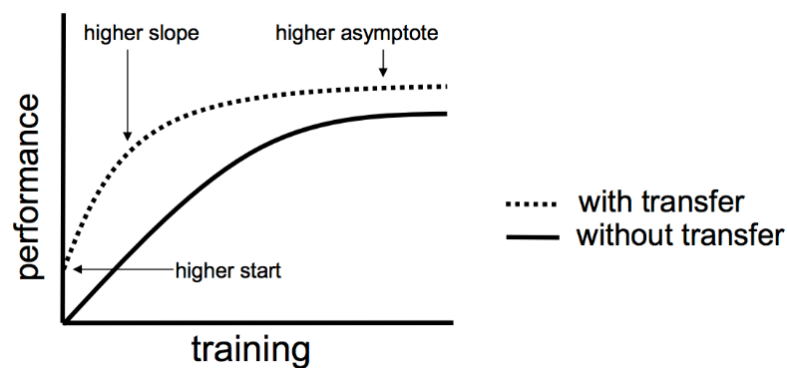
Trong đó: η : là tốc độ học (learning rate)

- $\mathbf{y}^{(i)}$: giá trị đúng của input $\mathbf{x}^{(i)}$, $\mathbf{y}^{(i)}$ nhận giá trị 0 hoặc 1.
- $\mathbf{a}^{(i)}$: giá trị mô hình dự đoán ứng với input $\mathbf{x}^{(i)}$.
- \mathbf{m} : là số điểm dữ liệu.
- \mathbf{X} : dạng biểu diễn tất cả điểm dữ liệu, ở phần kí hiệu mình đã chỉ rõ.
- \mathbf{A} : dạng biểu diễn giá trị mô hình dự đoán được cho tất cả điểm dữ liệu.
- \mathbf{Y} : dạng biểu diễn nhãn của tất cả điểm dữ liệu.

2.7 Transfer Learning

Transfer Learning trong Deep Learning là một kỹ thuật mà trong đó:

- Một mô hình được huấn luyện sẵn để làm một công việc cụ thể nào đó, khi đó một phần hay toàn bộ pretrained model có thể được tái sử dụng phụ thuộc vào nhiệm vụ của mỗi layer trong mô hình đó.
- Một mô hình mới sử dụng một phần hay toàn bộ pretrained model để học một công việc đích và tùy vào nhiệm vụ của mỗi layer mà mô hình mới có thể thêm các layer khác dựa trên pretrained model sẵn có.



Hình 2.13 So sánh tương quan hiệu quả của model train từ đầu và transferred model

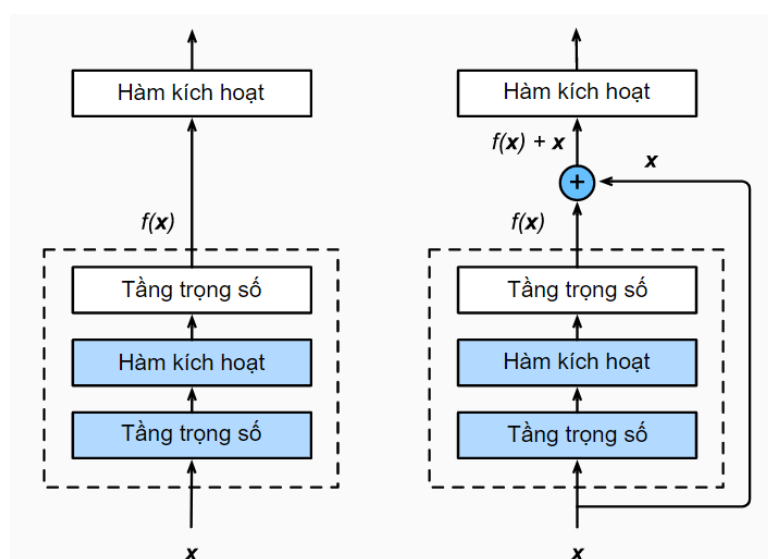
Fine-tuning:

Fine-tuning được hiểu là lấy một pre-trained model, tận dụng 1 phần hoặc toàn bộ các layer, thêm/sửa/xoá 1 vài layer/nhánh để tạo ra một model mới. Thường các layer đầu của model được freeze (đóng băng) lại - tức weight (trọng số) các layer này sẽ không bị thay đổi giá trị trong quá trình train. Lý do bởi các layer này đã có khả năng trích xuất

thông tin mức trừu tượng thấp, khả năng này được học từ quá trình training trước đó. Ta freeze lại để tận dụng được khả năng này và giúp việc train diễn ra nhanh hơn (model chỉ phải update weight ở các layer cao). Có rất nhiều các Object detection model được xây dựng dựa trên các Classifier model. Ví dụ mô hình YOLOv3 được xây dựng với backbone là Darknet 53.

2.8 Một số kiến trúc sử dụng

2.8.1 ResNet



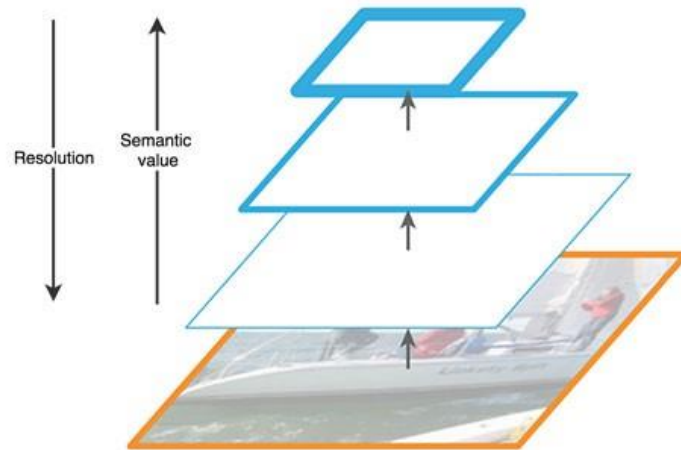
Hình 2.14 Sự khác biệt giữa một khối thông thường (trái) và một khối phần dư (phải)

ResNet đưa ra là sử dụng kết nối "tắt" đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block (khối phần dư). Ví dụ, ký hiệu đầu vào là \mathbf{x} . Giả sử ánh xạ lý tưởng muốn học được là $\mathbf{f}(\mathbf{x})$, và được dùng làm đầu vào của hàm kích hoạt. Phần nằm trong viền nét đứt bên trái phải khớp trực tiếp với ánh xạ $\mathbf{f}(\mathbf{x})$. Điều này có thể không đơn giản nếu chúng ta không cần khối đó và muốn giữ lại đầu vào \mathbf{x} . Khi đó, phần nằm trong viền nét đứt bên phải chỉ cần tham số hoá độ lệch khỏi giá trị \mathbf{x} , bởi vì ta đã trả về $\mathbf{x} + \mathbf{f}(\mathbf{x})$ (được gọi là một skip connection). Trên thực tế, ánh xạ phần dư thường dễ tối ưu hơn, vì chỉ cần đặt $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.

2.8.2 Feature Pyramid Networks (FPN)

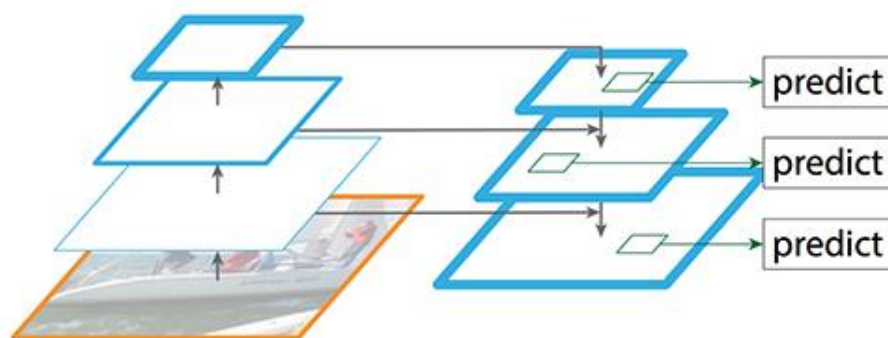
Dò tìm các đối tượng có kích thước nhỏ là một vấn đề đáng được giải quyết để nâng cao độ chính xác. Và FPN là mô hình mạng được thiết kế ra dựa trên khái niệm pyramid (kim tự tháp) để giải quyết vấn đề này. Mô hình FPN kết hợp thông tin của mô

hình theo hướng bottom-up kết hợp với top-down để dò tìm đối tượng (trong khi đó, các thuật toán khác chỉ thường sử dụng bottom-up). Khi chúng ta ở bottom và đi lên (up), độ phân giải sẽ giảm, nhưng giá trị ngữ nghĩa sẽ tăng lên. Xem hình mô phỏng bên dưới.



Hình 2.15 Quan hệ tương quan giữa độ phân giải và giá trị ngữ nghĩa

Trong quá trình xây dựng lại các layer từ top xuống bottom, chúng ta sẽ gặp một vấn đề khá nghiêm trọng là bị mất mát thông tin của các đối tượng. Ví dụ một đối tượng nhỏ khi lên top sẽ không thấy nó, và từ top đi ngược lại sẽ không thể tái tạo lại đối tượng nhỏ đó. Để giải quyết vấn đề này, chúng ta sẽ tạo các kết nối (skip connection) giữa các reconstruction layer và các feature map để giúp quá trình detector dự đoán các vị trí của đối tượng thực hiện tốt hơn (hạn chế tốt nhất việc mất mát thông tin).



Hình 2.16 Thêm các skip connection giữa feature map và reconstruction layer

2.8.3 Darknet 53

DarkNet-53 là một mạng nơ-ron phức tạp có độ sâu 53 lớp. Vì các mạng neural sâu (deep neural network) rất khó đào tạo, và với độ sâu ngày càng tăng, đôi khi độ chính xác của mạng bị bão hòa dẫn đến lỗi huấn luyện cao hơn. Để giải quyết vấn đề

này, residual block đã được sử dụng. Sự khác biệt về kiến trúc giữa khối tích chập thông thường và residual block là việc bổ sung skip connection. Skip connection mang đầu vào cho các lớp sâu hơn.

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

Hình 2.17 Kiến trúc của mô hình Darknet 53

CHƯƠNG 3. BÀI TOÁN PHÁT HIỆN ĐỐI TƯỢNG

3.1 Phân loại hình ảnh

Phân loại hình ảnh (Image Classification) là quá trình dự đoán một lớp hoặc nhãn cụ thể cho một thứ được xác định bởi một tập hợp các điểm dữ liệu. Các hệ thống machine learning xây dựng các mô hình dự đoán có lợi ích to lớn nhưng thường không thấy được cho mọi người. Ví dụ: phân loại email spam đáng tin cậy có nghĩa là hộp thư đến trung bình ít gánh nặng hơn và dễ quản lý hơn. Mặc dù người dùng cuối trung bình dường như không nhận thức được sự phức tạp của vấn đề và số lượng lớn xử lý cần thiết để giảm thiểu vấn đề, nhưng lợi ích rất rõ ràng. [17]

Phân loại hình ảnh là một tập hợp con của vấn đề phân loại, trong đó toàn bộ hình ảnh được gán nhãn. Có lẽ một bức ảnh sẽ được phân loại là một bức ảnh ban ngày hoặc ban đêm. Hoặc, theo cách tương tự, hình ảnh của ô tô và xe máy sẽ được tự động đặt vào các nhóm của riêng họ.

Có vô số danh mục, hoặc các lớp, trong đó một hình ảnh cụ thể có thể được phân loại. Xem xét một quy trình thủ công trong đó các hình ảnh được so sánh và các hình ảnh tương tự được nhóm theo các đặc điểm tương tự, nhưng không nhất thiết phải biết trước những gì bạn đang tìm kiếm. Rõ ràng, đây là một nhiệm vụ khó khăn. Để làm cho nó thậm chí nhiều hơn như vậy, giả sử rằng tập hợp các số hình ảnh trong hàng trăm ngàn. Rõ ràng là cần có một hệ thống tự động để thực hiện việc này nhanh chóng và hiệu quả.

Kiến trúc deep learning để phân loại hình ảnh thường bao gồm các lớp chập, làm cho nó trở thành một mạng nơ ron tích chập (CNN). Một số siêu đường kính, như số lượng các lớp chập và hàm kích hoạt cho mỗi lớp, sẽ phải được đặt. Đây là một phần không tầm thường của quá trình mà nó nằm ngoài phạm vi của cuộc thảo luận này. Tuy nhiên, là điểm khởi đầu, người ta thường có thể chọn các giá trị này dựa trên nghiên cứu hiện có.

Trên hệ thống như vậy là AlexNet, một CNN đã thu hút sự chú ý khi chiến thắng Thử thách nhận dạng hình ảnh quy mô lớn ImageNet (ILSVRC) năm 2012. Một mô hình được nghiên cứu kỹ lưỡng khác là Mạng thần kinh dư (ResNet), sau này đã

chiến thắng thử thách tương tự, cũng như cuộc thi Đối tượng chung của Microsoft trong bối cảnh (MS COCO) vào năm 2015.

Quá trình phân loại bao gồm:

- Input: Một hình ảnh với một đối tượng, chẳng hạn như một bức ảnh.
- Output: Nhãn lớp (ví dụ: Một hoặc nhiều số nguyên được ánh xạ tới nhãn lớp).

Classification



CAT

Hình 3.1 Minh họa phân loại ảnh mèo

3.2 Phân loại hình ảnh cùng với việc bản địa hóa

Phân loại hình ảnh với bản địa hóa (Classification + Localization Image) là một chuyên môn của phân loại hình ảnh, với yêu cầu bổ sung rằng đối tượng trong ảnh được đặt đầu tiên, và sau đó một hộp giới hạn được vẽ xung quanh nó.

Đây là một vấn đề khó khăn hơn so với phân loại hình ảnh và nó bắt đầu bằng việc xác định liệu chỉ có một đối tượng duy nhất được mô tả. Nếu vậy, hoặc nếu số lượng đối tượng được biết, thì mục tiêu là xác định vị trí của từng đối tượng và xác định bốn góc của hộp giới hạn tương ứng.

Quá trình này sẽ là một bước cần thiết trong một hệ thống chịu trách nhiệm nhận dạng xe. Hãy xem xét một hệ thống tự động duyệt hình ảnh của xe ô tô và được

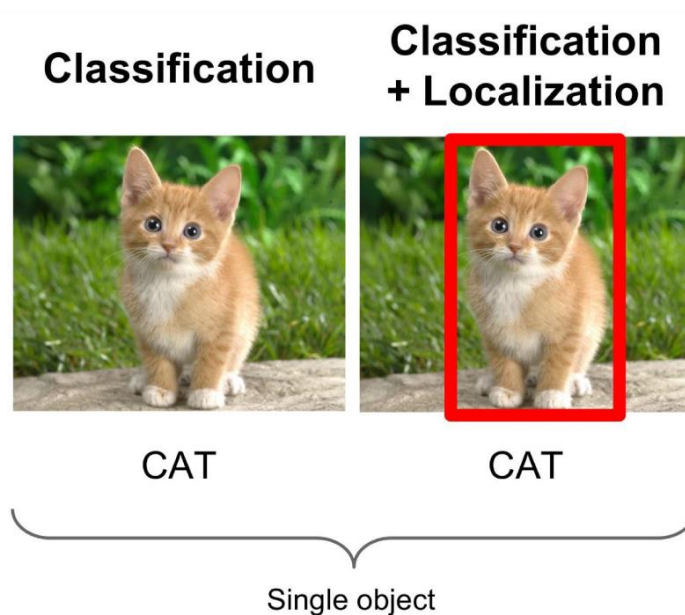
đảm bảo rằng có một chiếc xe duy nhất chứa trong cảnh. Khi chiếc xe đã được định vị, các thuộc tính như nhãn hiệu, kiểu dáng và màu sắc có thể được xác định.

Nhiệm vụ này có thể được thực hiện bằng cách sử dụng một mô hình deep learning phổ biến, chẳng hạn như AlexNet hoặc ResNet, và sửa đổi lớp được kết nối đầy đủ để tạo hộp giới hạn. Như đã đề cập trước đây, có thể có một số tinh chỉnh để thực hiện về mặt cài đặt siêu đường kính hoặc sửa đổi kiến trúc cho hiệu quả trong một miền cụ thể, nhưng trong thực tế, các kiến trúc cơ bản hoạt động tốt. Sẽ cần phải có đủ dữ liệu huấn luyện bao gồm các ví dụ với cả mô tả đối tượng và hộp giới hạn được xác định rõ ràng, mặc dù các bộ dữ liệu mẫu có sẵn cho mục đích này.

Khó khăn với nhiệm vụ này xảy ra khi có một số lượng đối tượng không xác định trong hình. Trong phần lớn các hình ảnh, đặc biệt là những hình ảnh được chụp ở khu vực công cộng, sẽ có nhiều khả năng như người, phương tiện, cây cối và động vật khác nhau. Đối với loại môi trường này, vấn đề trở thành một trong những phát hiện đối tượng.

Quá trình bản địa hoá bao gồm:

- Input: Một hình ảnh có một hoặc nhiều đối tượng, chẳng hạn như một bức ảnh.
- Output: Một hoặc nhiều bounding box được xác định bởi tọa độ tâm, chiều rộng và chiều cao.



Hình 3.2 Minh họa phân loại và bản địa hoá ảnh

3.3 Phát hiện đối tượng

Phát hiện đối tượng (object detection) là phân loại hình ảnh với nội địa hóa, nhưng trong các hình ảnh có thể chứa nhiều đối tượng. Đây là một lĩnh vực nghiên cứu tích cực và quan trọng bởi vì các hệ thống thị giác máy tính sẽ được sử dụng trong robot và xe tự lái sẽ phải chịu những hình ảnh rất phức tạp. Xác định vị trí và xác định mọi đối tượng chắc chắn sẽ là một phần quan trọng trong quyền tự chủ của họ.

Kiến trúc cần thiết để phát hiện đối tượng khác nhau theo một cách quan trọng. Cụ thể, kích thước của vector đầu ra không cố định. Ví dụ, nếu có một đối tượng trong hình, thì sẽ có bốn tọa độ xác định hộp giới hạn. Giá trị tĩnh và được xác định trước này hoạt động bằng cách sử dụng các kiến trúc đã đề cập trước đó. Tuy nhiên, khi số lượng đối tượng tăng lên, số lượng tọa độ cũng tăng theo. Đặc biệt là số lượng đối tượng không được biết trước, điều này đòi hỏi phải điều chỉnh trong trang điểm của mạng lưới thần kinh.

Một kiến trúc được sửa đổi như vậy là R-CNN: Khu vực có các tính năng CNN. Cách tiếp cận này liên quan đến việc tạo các vùng quan tâm được thu nhỏ theo kích thước cố định và sau đó chuyển tiếp các vùng này thành một mô hình như AlexNet. Trong khi hệ thống này tạo ra kết quả tốt, nó đắt tiền về mặt tính toán và quá chậm đối với hệ thống thị giác máy tính thời gian thực.

Với mục tiêu tăng tốc R-CNN, đã có nhiều điều chỉnh khác nhau đối với kiến trúc. Đầu tiên là Fast R-CNN, chứa tối ưu hóa và các cải tiến khác giúp cải thiện cả tốc độ và độ chính xác phát hiện. Tiến lên một bước nữa, thế hệ tiếp theo, mô hình R-CNN nhanh hơn, bao gồm một CNN bổ sung có tên Mạng đề xuất khu vực (RPN).

RPN được đào tạo để tạo các vùng chất lượng cao được gửi tới mô hình R-CNN nhanh. Sự kết hợp của các thuật toán này dẫn đến sự gia tăng tốc độ ấn tượng và thực sự trên con đường hướng tới phát hiện đối tượng thời gian thực trong các hệ thống thị giác máy tính.

Quá trình phát hiện ảnh bao gồm:

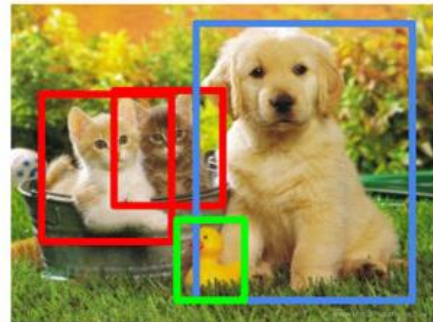
- Input: Một hình ảnh có một hoặc nhiều đối tượng, chẳng hạn như một bức ảnh.
- Output: Một hoặc nhiều bounding box và nhãn cho mỗi bounding box.

Classification



CAT

Object Detection



CAT, DOG, DUCK

Hình 3.3 Minh họa quá trình phát hiện đối tượng

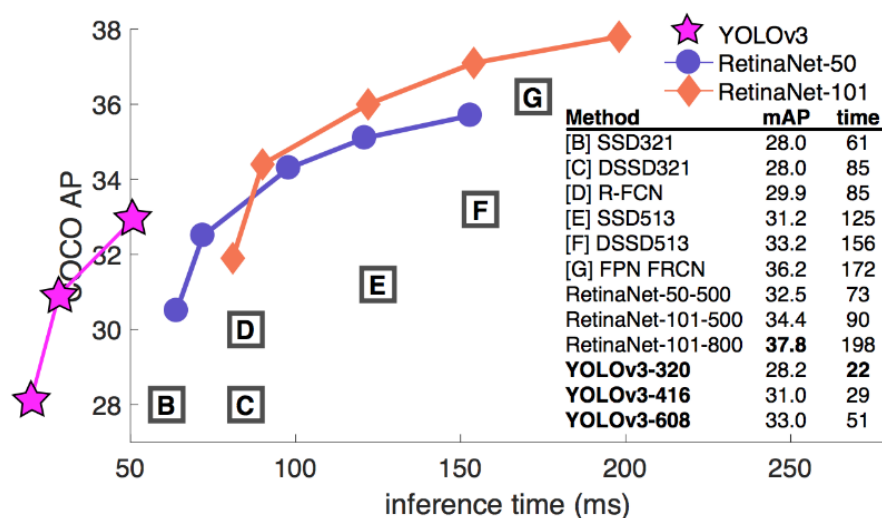
Lịch sử hình thành, phát triển và đặc điểm cấu trúc của các thuật toán object detection bao gồm 3 nhóm chính:

- Họ các mô hình RCNN (Region-Based Convolutional Neural Networks) giải quyết các nhiệm vụ định vị đối tượng và phát hiện đối tượng.
- Họ các mô hình SSD (Single Shot Detector) mô hình phát hiện đối tượng thời gian thực dựa trên kiến trúc VGG-16.
- Họ các mô hình YOLO (You Only Look Once), là một nhóm kỹ thuật thứ ba để nhận dạng đối tượng được thiết kế để phát hiện đối tượng theo thời gian thực.

3.4 Lựa chọn mô hình object detection

Trong phạm vi đề án này, tôi chỉ quan tâm đến họ các mô hình thứ 3 là YOLOv3, bởi vì những lý do sau:

- Về mặt tốc độ và độ chính xác: Các mô hình họ RCNN tuy độ chính xác cao nhưng tốc độ không thể thực hiện theo thời gian thực, vậy ta loại bỏ RCNN.



Hình 3.4 So sánh tốc độ của các mô hình phát hiện đối tượng tại thời điểm YOLOv3 mới ra mắt (2018)⁵

Ta có thể thấy thì tốc độ của YOLOv3 vượt xa so với các phiên bản SSD, và độ chính xác (mAP) cũng cao hơn ở chế độ suy luận mô hình.

- YOLOv3 có FPN sử dụng các feature map nhiều thông tin hơn để cải thiện độ chính xác, có thể phát hiện đối tượng nhỏ với độ chính xác cao. SSD có layer ở bottom không được sử dụng để phát hiện đối tượng. Vì những layer này có độ phân giải cao nhưng giá trị ngưỡng của chúng lại không đủ cao (thấp) nên những nhà nghiên cứu bỏ chúng đi để tăng tốc độ xử lý. Cho nên, SSD chỉ sử dụng các layer ở lớp trên, và do đó sẽ không phát hiện được các đối tượng có kích thước nhỏ.

- YOLOv3 sử dụng batch normalization và gradient descent để chuẩn hoá và tối ưu dữ liệu.

- YOLOv3 sử dụng nhiều data augmentation trong quá trình training.
- YOLOv3 Sử dụng 9 anchors (điểm neo) để phát hiện đối tượng.

3.5 YOLOv3

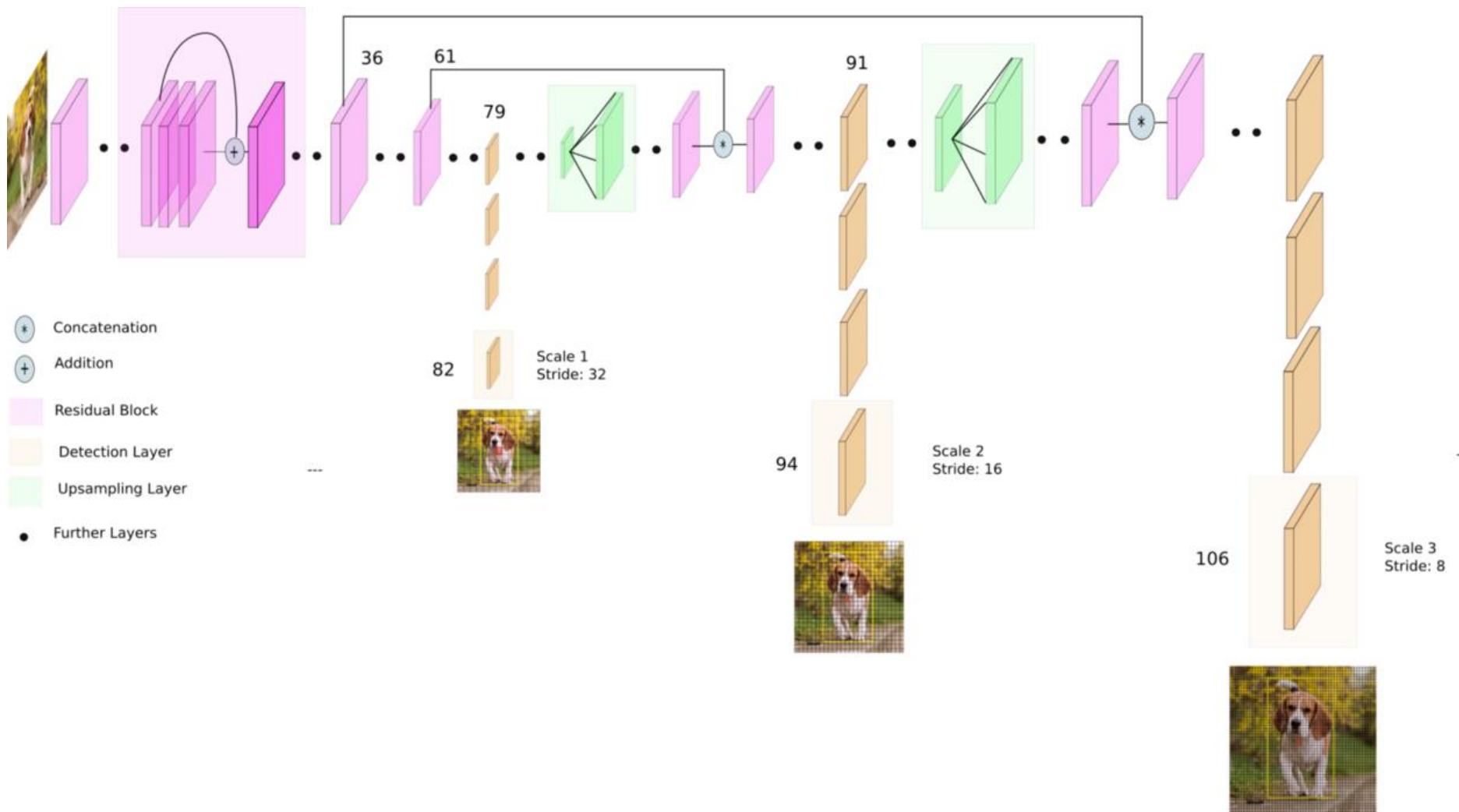
3.5.1 Giới thiệu YOLO

YOLO - You Only Look Once (tạm dịch Bạn Chỉ Nhìn Một Lần) là một trong những mô hình phát hiện đối tượng tốt nhất ở thời điểm hiện tại, phiên bản của mô hình

⁵ <https://pjreddie.com/darknet/yolo/>

này đều có những cải tiến rất đáng kể sau mỗi phiên bản. Sau 3 phiên bản của tác giả chính Joseph Redmon là YOLOv1 đến v3, tính đến thời điểm hiện tại có thêm một paper YOLOv4 của Alexey Bochkovskiy được dẫn link trực tiếp từ repository gốc của Joseph Redmon và YOLOv5 đang được phát triển. Nhiều hệ thống phát hiện đối tượng cần phải “nhìn” qua ảnh nhiều hơn một lần để có thể phát hiện tất cả các đối tượng trong ảnh, hoặc nó phải trải qua hai giai đoạn để phát hiện đối tượng. YOLO không cần phải trải qua những quy trình nhàm chán này. Nó chỉ cần “nhìn” một lần vào hình ảnh để phát hiện tất cả các đối tượng và đó là lý do tại sao họ chọn tên You Only Look Once và đó thực sự là lý do tại sao YOLO là một mô hình rất nhanh. Trong phạm vi đồ án này chỉ giới thiệu phiên bản YOLOv3.

3.5.2 Kiến trúc của YOLOv3



Hình 3.5 Kiến trúc YOLOv3

YOLOv3 có thể gọi là thuật toán hay mô hình (nó là một mô hình Deep Learning hoàn chỉnh gồm nhiều thuật toán khác nhau), sử dụng một biến thể của Darknet, ban đầu là mạng 53 lớp được huấn luyện trên tập dữ liệu Imagenet. Đối với nhiệm vụ phát hiện, 53 lớp khác được xếp chồng lên nó, tạo nên một kiến trúc mạng tích chập hoàn toàn (Fully Convolutional Network - FCN) gồm 106 lớp, với các residual block (khối phần dư), skip connection (bỏ qua kết nối) và các lớp upsampling, tương tự với mô hình FPN và ResNet. Không có lớp tổng hợp nào được sử dụng và một lớp tích chập có bước nhảy bằng 2 được sử dụng để downsampling cho các biểu đồ đặc trưng (feature map). Điều này giúp ngăn ngừa mất các đặc trưng cấp thấp thường được cho là do các lớp tổng hợp. YOLOv3 bỏ qua ba lớp cuối cùng của Darknet 53 (lớp avgpool, lớp full connected, lớp softmax) vì các lớp này chủ yếu được sử dụng để phân loại hình ảnh. Trong nhiệm vụ phát hiện đối tượng, YOLOv3 sử dụng Darknet 53 chỉ để trích xuất các đặc trưng hình ảnh nên ba lớp này sẽ không cần thiết.

Là một FCN, kích thước của hình ảnh đầu vào luôn bất biến và có thể chỉnh các giá trị này trong tệp cấu hình mạng. Một vấn đề lớn là nếu chúng ta muốn xử lý hình ảnh theo batch (hình ảnh theo batch có thể được xử lý song song bởi GPU, tăng tốc độ xử lý), chúng ta cần có tất cả hình ảnh có chiều cao và chiều rộng cố định. Điều này là cần thiết để nối nhiều hình ảnh thành một batch lớn (ghép nhiều PyTorch tensor thành một để thuận tiện cho quá trình triển khai).

Các lớp downsampling hình ảnh bằng một hệ số được gọi là bước nhảy (stride).

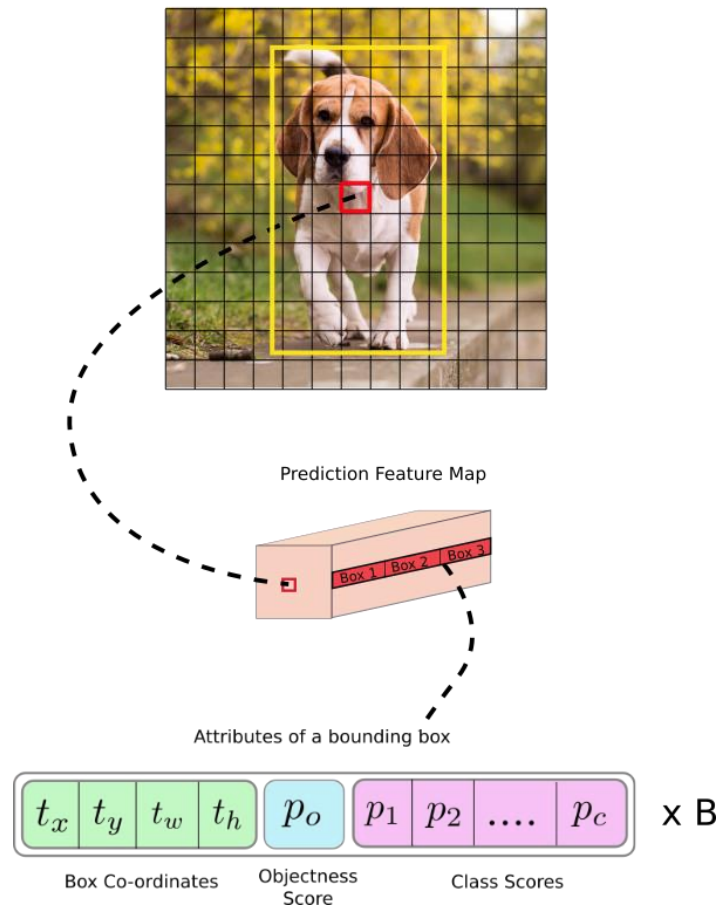
3.5.3 Phát hiện đối tượng ở 3 tỷ lệ kích thước khác nhau

Tính năng nổi bật nhất của YOLOv3 là nó thực hiện phát hiện ở ba tỷ lệ kích thước khác nhau. Trong YOLO v3, việc phát hiện được thực hiện bằng cách áp dụng các kernel phát hiện 1×1 lên các biểu đồ đặc trưng có ba kích thước khác nhau tại ba vị trí khác nhau trong mạng.

Lưu ý, mặc dù thuật ngữ chính xác để mô tả một đơn vị trong biểu đồ đặc trưng sẽ là một neural, nhưng ở đây gọi nó là một ô (cell) làm cho nó trực quan hơn trong ngữ cảnh bài toán này.

Hình dạng (shape) của kernel phát hiện là $1 \times 1 \times (\mathbf{B} \times (\mathbf{5} + \mathbf{C}))$. Ở đây \mathbf{B} là số hộp giới hạn (bounding box) mà một ô (cell) trên feature map có thể dự đoán, “5” là bởi có 4 thuộc tính hộp giới hạn (toạ độ, kích thước tâm) và độ tin cậy của một đối tượng (hay **objectness score**), và \mathbf{C} là số lớp (bao gồm cả **class scores**) cần dự đoán. Ví dụ, trong YOLO v3 được đào tạo trên dataset COCO, có $\mathbf{B} = 3$ và $\mathbf{C} = 80$, do đó shape của kernel là $1 \times 1 \times 255$. Biểu đồ đặc trưng do kernel này tạo ra có chiều cao và chiều rộng giống hệt biểu đồ đặc trưng trước đó và có một dãy các thuộc tính phát hiện được gọi là độ sâu (**depth**) như đã mô tả ở trên. YOLOv3 dự đoán 3 hộp giới hạn cho mỗi ô.

Mỗi ô của feature map dự đoán một đối tượng thông qua một trong các hộp giới hạn của nó nếu tâm của đối tượng nằm trong trường tiếp nhận (receptive field) của ô đó. (Trường tiếp nhận là vùng của hình ảnh đầu vào có thể nhìn thấy đối với ô). Điều này liên quan đến cách YOLO được huấn luyện, trong đó chỉ có một hộp giới hạn chịu trách nhiệm phát hiện bất kỳ đối tượng nhất định nào. Đầu tiên, chúng ta phải xác định chắc chắn hộp giới hạn này thuộc về ô nào. Để làm điều đó, ta chia hình ảnh đầu vào thành một lưới có kích thước bằng với kích thước của biểu đồ đặc trưng cuối cùng. YOLO v3 đưa ra dự đoán ở ba tỷ lệ kích thước, bằng cách downsampling kích thước của hình ảnh đầu vào tại các lớp lần lượt với 32, 16 và 8.



Hình 3.6 Minh họa một ô chịu trách nhiệm phát hiện tại kích thước 13×13

Ở đây ta lấy ví dụ bằng ảnh có kích thước 416×416 . Lần phát hiện đầu tiên được thực hiện bởi lớp thứ 82. Đối với 81 lớp đầu tiên, hình ảnh được mạng downsampling sao cho lớp thứ 81 có stride là 32. Lúc này, biểu đồ đặc trưng sẽ có kích thước 13×13 . Một lần phát hiện được tạo ở đây bằng cách sử dụng kernel phát hiện 1×1 , tạo nên biểu đồ đặc trưng phát hiện có shape là $13 \times 13 \times 255$.

Sau đó, ô (trên hình ảnh đầu vào) chứa tâm của hộp ground truth của một đối tượng được chọn làm ô chịu trách nhiệm dự đoán đối tượng. Trong ảnh trên, nó là ô được đánh dấu màu đỏ, chứa trung tâm của hộp ground truth (được đánh dấu màu vàng). Bây giờ, ô này có thể dự đoán ba hộp giới hạn. Cái nào sẽ được gán cho nhãn ground truth? Để hiểu được điều đó, chúng ta phải tìm hiểu khái niệm neo (anchors) ở phần sau.

Tiếp theo, feature map từ lớp 79 phải được tích chập một vài lớp trước khi upsampling 2 lần đến kích thước 26×26 . Feature map này sau đó được ghép độ sâu (depth) với feature map từ lớp 61. Sau đó, feature map mới được ghép nối phải được

tích chập một vài lớp 1 x 1 để hợp nhất các đặc trưng từ lớp trước đó. Sau đó, phát hiện thứ hai được thực hiện bởi lớp thứ 94, tạo nên feature map có shape là 26 x 26 x 255.

Cuối cùng, quy trình tương tự lại được thực hiện một lần nữa, trong đó feature map từ lớp 91 phải được tích chập một vài lớp trước khi được nối theo độ sâu với feature map từ lớp 36. Giống như trên, một vài lớp tích chập 1 x 1 theo sau để kết hợp thông tin từ trước lớp (36). Sau đó, phát hiện cuối cùng ở lớp thứ 106, tạo ra feature map có kích thước shape là 52 x 52 x 255.

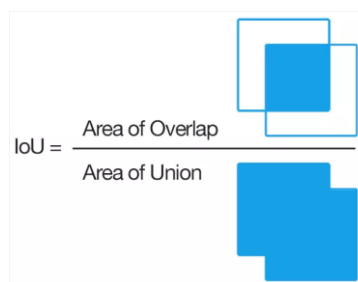
Kích thước 13 x 13 chịu trách nhiệm phát hiện các vật thể lớn, trong khi kích thước 52 x 52 phát hiện các vật thể nhỏ, với kích thước 26 x 26 phát hiện các vật thể trung bình. Khắc phục nhược điểm của các phiên bản YOLO trước đó.

3.5.4 Anchor Boxes

Dự đoán chiều rộng và chiều cao của hộp giới hạn trên thực tế dẫn đến gradients không ổn định trong quá trình huấn luyện. Thay vào đó, YOLOv3 dự đoán các phép biến đổi không gian logarit, hoặc đơn giản là độ lệch (offset) của các hộp giới hạn mặc định được xác định trước gọi là neo (anchor). Sau đó, các biến đổi này được áp dụng cho các hộp neo để thu được dự đoán. YOLO v3 dự đoán ba hộp giới hạn trên mỗi ô (3 neo) ở cả 3 tỷ lệ vì vậy YOLOv3 dự đoán tổng cộng 10647 bounding box. Hộp giới hạn chịu trách nhiệm phát hiện đối tượng sẽ là hộp neo có **IoU** cao nhất với hộp ground truth. Các hộp neo này được thực hiện bằng cách chạy phân cụm **K-mean** trên tập dữ liệu.

3.5.5 Intersection Over Union (IOU)

IoU (Intersection Over Union) là hàm đánh giá độ chính xác của object detector trên tập dữ liệu cụ thể. IoU được tính bằng:


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

Hình 3.7 Công thức tính IoU

Trong đó Area of Overlap là diện tích phần giao nhau giữa predicted bounding box với growth truth box, còn Area of Union là diện tích phần hợp giữa predicted bounding box với hộp growth truth. Nếu $IOU > 0.5$ thì prediction được đánh giá là tốt.

3.5.6 Bounding Box Prediction (dự đoán hộp giới hạn)

Các công thức sau đây mô tả cách chuyển đổi đầu ra của mạng để có được các dự đoán hộp giới hạn:

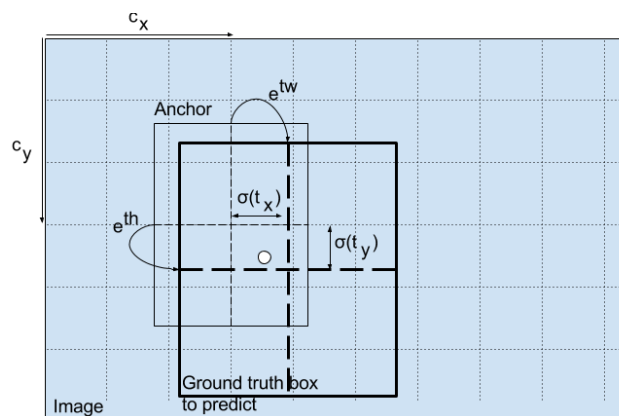
$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

b_x, b_y, b_w, b_h lần lượt là tọa độ tâm (x, y) và chiều rộng, chiều cao bounding box dự đoán. t_x, t_y, t_w, t_h là đầu ra của mạng. c_x và c_y là tọa độ trên cùng bên trái của lưới ảnh. p_w và p_h là kích thước hộp neo. YOLOv3 dự đoán tọa độ tâm thông qua một hàm **sigmoid**. Điều này buộc giá trị đầu ra phải nằm trong khoảng từ 0 đến 1 (để nằm trong phạm vi một ô). Sau đó, ta thêm các tọa độ trên cùng bên trái c_x, c_y để dự đoán tọa độ thực của hộp giới hạn.



Hình 3.8 Minh họa các dự đoán tọa độ bounding box dựa qua hộp neo

Kích thước của hộp giới hạn được dự đoán bằng cách áp dụng phép biến đổi không gian logarit cho đầu ra và sau đó nhân với kích thước hộp neo. Trong kết quả của dự đoán, b_w và b_h , được chuẩn hóa theo chiều cao và chiều rộng của hình ảnh. (Nhãn huấn luyện được chọn theo cách này).

Đối với **objectness score (điểm đối tượng)**: Trong YOLO v3 đưa ra điểm cho các đối tượng cho mỗi hộp giới hạn. Nó sử dụng hồi quy logistic để dự đoán điểm đối tượng để mang tính khách quan, sử dụng hàm kích hoạt leaky ReLU.

Đối với **class scores**: YOLOv3 không sử dụng softmax; thay vào đó sử dụng logistic classifier rời rạc (trong lúc huấn luyện sử dụng hàm mất mát binary cross-entropy), vốn dùng để phân biệt các đối tượng bao hàm nhau ví dụ như con người và phụ nữ, sử dụng softmax cho dự đoán chỉ giả định rằng mỗi hộp có đúng một lớp, điều này thường không đúng.

3.5.7 Thresholding (ngưỡng)

Ta lọc các hộp dựa trên điểm đối tượng của chúng, các hộp có điểm dưới ngưỡng sẽ bị bỏ qua.

3.5.8 Thuật toán Non-Maximum Suppression

Sau khi thực hiện object detection một ảnh qua mạng YOLOv3, chúng ta sẽ thu được rất nhiều proposals (hộp giới hạn đề xuất). Ở trạng thái này, có rất nhiều proposals là bounding box cho một đối tượng duy nhất, điều này dẫn tới việc dư thừa. Vì vậy, chúng ta cần giữ một hộp có điểm tin cậy cao nhất và loại bỏ các hộp giới hạn khác có tỷ lệ trùng lặp cao và điểm thấp. Kỹ thuật này được gọi là NMS.

Kỹ thuật này hoạt động theo ba bước:

- Bước 1: Xác định xem số hộp giới hạn có lớn hơn 0 hay không. Nếu không, hãy kết thúc quá trình.
- Bước 2: Chọn ô giới hạn có độ tin cậy cao nhất và lấy nó ra.
- Bước 3: Tính IoU giữa hộp giới hạn đã chọn và các hộp giới hạn còn lại. Loại bỏ tất cả các hộp giới hạn có giá trị IoU cao hơn giá trị ngưỡng. Chuyển sang bước 1.

3.5.9 Giải thích hàm mất mát

Hàm mất mát trong YOLO được tính trên việc dự đoán và nhãn mô hình để tính. Cụ thể hơn nó là tổng mất mát của 3 thành phần con sau:

- Mất mát của việc dự đoán lớp - Classification loss được tính theo hàm BCE (binary cross entropy).
- Mất mát của dự đoán tọa độ tâm, chiều dài, rộng của boundary box (x, y, w, h) - Localization loss được tính theo hàm lỗi toàn phương trung bình MSE.
- Mất mát của việc dự đoán bounding box đó chứa đối tượng so với nhãn thực tế tại ô vuông đó – Confidence loss được tính theo hàm BCE.

$$\begin{aligned}
 loss(object) = & \lambda_{coord} \sum_{i=0}^{K \times K} \sum_{j=0}^M I_{ij}^{obj} (2 - w_i \times h_i) [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \\
 & \lambda_{coord} \sum_{i=0}^{K \times K} \sum_{j=0}^M I_{ij}^{obj} (2 - w_i \times h_i) [(w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2] - \\
 & \sum_{i=0}^{K \times K} \sum_{j=0}^M I_{ij}^{obj} [\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] - \\
 & \lambda_{noobj} \sum_{i=0}^{K \times K} \sum_{j=0}^M I_{ij}^{noobj} [\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] - \\
 & \sum_{i=0}^{K \times K} I_{ij}^{obj} \sum_{c \in classes} [\hat{p}_i(c) \log(p_i(c)) + (1 - \hat{p}_i(c)) \log(1 - p_i(c))]
 \end{aligned}$$

Mất mát của dự đoán Bounding Box

Mất mát của dự đoán đối tượng

Mất mát của dự đoán lớp

Hình 3.9 Hàm mất mát của YOLOv3

Trong đó: λ_{coord} , $\lambda_{noobject}$: Là trọng số thành phần trong paper gốc tác giả lấy giá trị lần lượt là 5 và 0.5.

- $K \times K$: Là tỷ lệ ô lưới tại ảnh đó (với một ảnh kích thước 416 thì có 3 tỷ lệ là 13×13 , 26×26 và 52×52).
- M : Số bounding box cần dự đoán tại một ô, ở YOLOv3 là bằng 3.
- $I_{ij}^{obj} = 1$: Nếu box thứ j của ô thứ i có chứa object. Vì huấn luyện cần các image với ground-truth (vị trí của các objects) nên YOLO biết điểm trung tâm của từng object rơi vào ô nào trong grid $K \times K$.
- $I_{ij}^{noobj} = 1$: Nếu box thứ j của ô thứ i không chứa object.
- $(2 - w_i \times h_i)$: Hệ số tỷ lệ để tăng tổn thất của hộp nhỏ thứ i .

- (x_i, y_i) : Là tọa độ tâm của ground-truth bounding box thứ i và (\hat{x}_i, \hat{y}_i) là tọa độ tâm của predicted bounding box thứ i .
- (w_i, h_i) : Là kích thước của ground-truth bounding box thứ i và (\hat{w}_i, \hat{h}_i) là kích thước của predicted bounding box thứ i .
- C_i : Điểm tin cậy của ô i . Đối với các hộp j của ô i nơi object tồn tại C_i luôn = 1. Và ngược lại.
- \hat{C}_i : Điểm tin cậy dự đoán của ô i . $\hat{C}_i = P(\text{contain object}) * IoU(\text{predict bbox}, \text{ground truth bbox})$.
- $p_i(c)$: Xác suất có điều kiện, có hay không ô i có chứa một đối tượng của lớp $c \in \text{classes}$. Chú ý $p_i(c)$ luôn = 1 nếu đúng lớp c với ground-truth, ngược lại thì $p_i(c)$ luôn = 0.
- $\hat{p}_i(c)$: Xác suất có điều kiện dự đoán của ô i .
- classes : Các lớp đối tượng cần được nhận dạng, ví dụ chó, mèo, oto...

3.6 Một số chỉ số đánh giá bài toán phát hiện đối tượng

3.6.1 Precision và Recall

Quy ước: Giả sử chia một số kết quả khi kiểm tra mô hình phát hiện đối tượng trong một bức ảnh thành 3 nhóm được hiểu như sau:

- False negative (FN): Mô hình báo không có đối tượng, nhưng trong ảnh có.
- False positive (FP): Mô hình báo có đối tượng, nhưng trong ảnh không có.
- True positive (TP): Mô hình báo có đối tượng và trong ảnh cũng có.

- **Precision**: Đánh giá độ tin cậy của kết luận đưa ra (bao nhiêu % lời kết luận của mô hình là chính xác).

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

- **Recall**: Đánh giá khả năng tìm kiếm toàn bộ các ground truth của mô hình (bao nhiêu % positive samples mà mô hình nhận diện được).

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

- **IoU**: ta đã tìm hiểu ở phần trước.

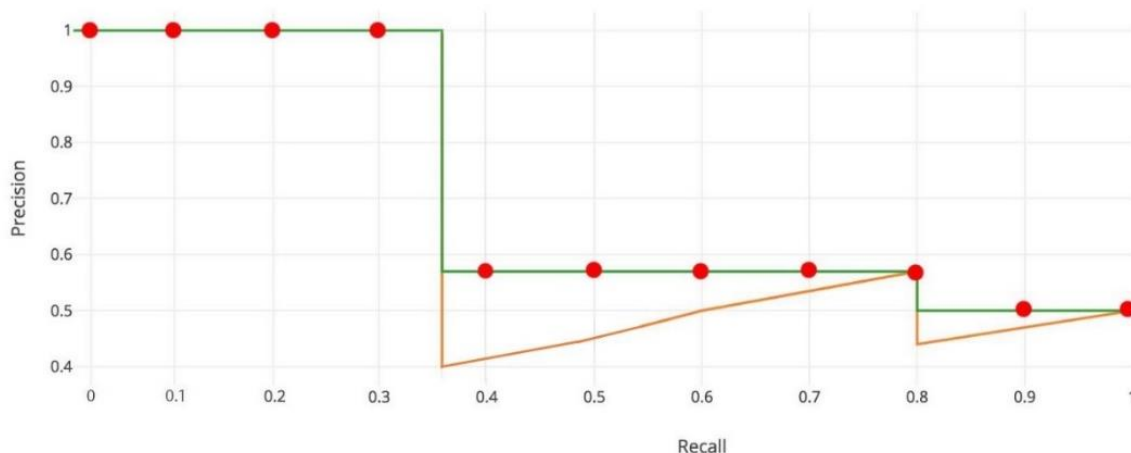
- **Precision-Recall Curve:** Đường biểu diễn mối liên hệ giữa precision và recall.

3.6.2 AP và mAP

Với Precision-Recall Curve, **Area Under the Curve (AUC)** còn có một tên khác là **Average precision (AP)**. Giả sử có N ngưỡng (ngưỡng thường là IoU) để tính precision và recall, với mỗi ngưỡng cho một cặp giá trị precision, recall là $P_n, R_n, n = 1, 2, \dots, N$. Precision-Recall curve được vẽ bằng cách vẽ từng điểm có tọa độ (R_n, P_n) trên trục tọa độ và nối chúng với nhau. AP được xác định bằng:

$$AP = \sum n(R_n - R_{n-1})P_n$$

Ở đó $(R_n - R_{n-1})P_n$ chính là diện tích hình chữ nhật có chiều rộng $(R_n - R_{n-1})$ và chiều cao P_n , đây cũng gần với cách tính tích phân dựa trên cách tính diện tích của từng hình chữ nhật nhỏ.



Hình 3.10 Giá trị AP là giá trị phía dưới đường biểu diễn Precision-Recall Curve

mAP (mean Average Precision) là trung bình của AP được tính cho tất cả các lớp. Trong một vài trường hợp, mAP chính là AP.

Kí hiệu:

- **AP@[0.5:0.05:0.95]** : Giá trị AP trung bình của các AP khi IoU có giá trị (0.5, 0.55, 0.6, ..., 0.95)
- **AP@0.5**: Giá trị AP khi IoU = 0.5
- **AP (small/medium/large)**: AP cho small/medium/large object (object có kích thước: nhỏ hơn 32×32 ; từ 32×32 đến 96×96 ; lớn hơn 96×96).

Mối quan hệ giữa precision – recall giúp mAP đánh giá được về độ chính xác của classification task. Precision – recall thay đổi khi ngưỡng IoU thay đổi, do đó, tại một giá trị IoU xác định, có thể đo hoặc so sánh độ tốt của các mô hình.

CHƯƠNG 4. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

4.1 Hiện trạng tổ chức

Các vấn đề mà một hệ thống hướng dẫn bãi đỗ xe PGS cần đó là:

- Phát hiện những vị trí còn trống và đã đỗ trong bãi.
- Đếm những vị trí còn trống và đã đỗ.

4.2 Yêu cầu hệ thống

4.2.1 Yêu cầu chức năng

- Giới thiệu sản phẩm công nghệ hiện tại.
- Thông tin liên hệ.
- Hiện thị video camera theo thời gian thực tại bãi đỗ xe bao gồm:
 - Hiện thị những phát hiện gồm nhãn và vị trí còn trống, đã đỗ.
 - Hiện thị số lượng vị trí còn trống, đã đỗ và tổng các vị trí hiện có.

4.2.2 Yêu cầu phi chức năng

- Độ chính xác của mô hình cần tinh chỉnh phù hợp để quá trình nhận diện diễn ra thuận lợi trong thời gian dài.
- Giao diện và hiệu ứng đẹp, đơn giản, thân thiện để sử dụng với mọi đối tượng người dùng.
- Cần chú thích rõ ràng, dễ hiểu với người sử dụng.
- Thông tin giới thiệu các lợi ích và chức năng ngắn gọn, dễ hiểu, đầy đủ.
- Tốc độ của video camera phải ổn định có thể nhìn thấy rõ các vị trí phát hiện được và số lượng vị trí đó trong bãi.
- Thông tin giới thiệu ngắn gọn rõ ràng.
- Video camera phải hiển thị chính xác bất kể điều kiện thời tiết và ánh sáng khác nhau làm nhiễu hình ảnh.
- Nhãn của các vị trí còn trống và đã đỗ có màu khác nhau để người dùng dễ phân biệt.

4.3 Giải quyết bài toán phát hiện đối tượng bằng thuật toán YOLOv3

Có 2 phương án để giải quyết bài toán phát hiện chỗ còn trống và đã đỗ với thuật toán YOLOv3, đó là:

- Thứ nhất, sử dụng tệp weights được tác giả thuật toán YOLOv3 huấn luyện sẵn với tập dữ liệu COCO sau đó phát hiện các vị trí của ô tô, và lấy tổng các vị trí hiện có trong bãi trừ cho số lượng vị trí ô tô đó. Tuy phương án này rút ngắn thời gian huấn luyện mô hình nhưng độ chính xác không cao, dễ bị nhiễu bởi các đối tượng khác bởi vì tập dữ liệu COCO được huấn luyện để nhận dạng lên đến 80 đối tượng, và trong lúc phát hiện có thể có ô tô đang di chuyển sẽ khiến thuật toán nhận diện nhầm. Hơn nữa các vị trí còn trống sẽ không được phát hiện, cho nên thuật toán sẽ không biết khi nào ô tô được đậu đúng vị trí, vì khi ô tô đậu bên ngoài vạch cũng được phát hiện. Cuối cùng phương án này chúng ta cần phải biết trước số lượng tổng các vị trí trong bãi. Đây là phương án tiếp cận quá nhiều bước.

- Thứ hai, ta sẽ huấn luyện mô hình để phát hiện được cả 2 trạng thái trống và đã đỗ trong bãi, sau đó tính tổng bằng cách lấy tổng các đối tượng ta phát hiện được. Ta sẽ chọn phương án này để triển khai.

4.4 Huấn luyện mô hình

Huấn luyện mô hình là bước đầu tiên trong một trình phát hiện đối tượng, lúc này mô hình sẽ được học có giám sát và tự rút trích ra các đặc trưng thành một tập các trọng số (weights) từ tập dữ liệu đầu vào.

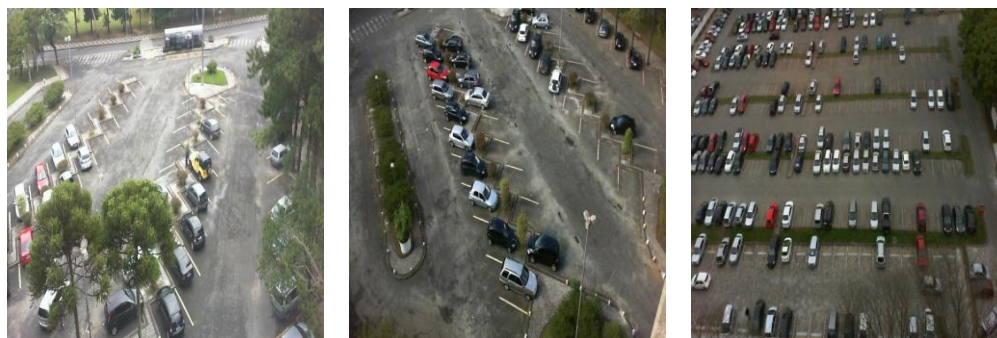
4.4.1 Thu thập dữ liệu

4.4.1.1 Tập dữ liệu lớn

Đối với tập dữ liệu lớn dùng để phát hiện nhiều đối tượng trong một ảnh, số lượng ảnh rất lớn phù hợp để huấn luyện mô hình YOLOv3. Tập dữ liệu này được chia sẻ công khai trên Roboflow, đây là trang web chia sẻ những tập dữ liệu được gán nhãn sẵn và đầy đủ các định dạng cần thiết cho huấn luyện các mô hình CNN. Tập dữ liệu đó tên là PKLot Dataset⁶, sau đó tải với định dạng YOLO Darknet để phù hợp với công cụ Darknet ta sẽ dùng để huấn luyện. Định dạng này bao gồm các ảnh và tệp văn bản trong

⁶ <https://public.roboflow.com/object-detection/pklot>

đó ảnh và tệp annotation chứa nhãn có định dạng *txt* cùng tên với nhau, tập dữ liệu này bao gồm 8169 ảnh khác nhau với kích thước 640 x 640.



Hình 4.1 Một số hình ảnh trong tập dữ liệu PKLot Dataset

4.4.1.2 Tập dữ liệu mô phỏng

Đối với tập dữ liệu mô phỏng dùng để mô phỏng mô hình trực tiếp, tự thu thập và gán nhãn cho ảnh, cũng như kiểm tra xem với dữ liệu ít ỏi thì mô hình còn chính xác hay không. Để dữ liệu chính xác và tiện trong quá trình gán nhãn, ta phải thu thập dữ liệu từ một góc nhìn nhất định trong suốt quá trình và các điều kiện ánh sáng khác nhau từ camera, tổng cộng ảnh thu thập được là 202 ảnh với kích thước 1024 x 768.

Tuy nhiên để tối ưu tốc độ và độ chính xác của mô hình lúc huấn luyện thì ta sẽ dùng công cụ Photoshop⁷ với tính năng Actions để resize hàng loạt các ảnh về kích



Hình 4.2 Dữ liệu mô phỏng tự thu thập

thước 640 x 640, và giảm chất lượng lần kích thước tệp ảnh xuống thấp (tuy vậy vẫn giữ được những đặc trưng ban đầu) giảm áp lực học cho mô hình.

⁷ Photoshop là một phần mềm chỉnh sửa đồ họa được phát triển và phát hành bởi hãng Adobe Systems

4.4.2 Gán nhãn cho ảnh

Hiện tại có rất nhiều các công cụ mã nguồn mở khác nhau được sử dụng để gán nhãn cho mô hình YOLO, tuy nhiên sau khi dùng thử một số phần mềm thì tôi sử dụng công cụ *labelImg*⁸ của **pypi**.



Hình 4.3 Gán nhãn bằng labelImg

Bên dưới là ví dụ nội dung của một tệp *annotation.txt* của một ảnh:

4 bbox	1	2	0.414500	0.574667	0.361000	0.578667
	2	2	0.517000	0.482667	0.284000	0.589333
	3	2	0.560000	0.409333	0.264000	0.536000
	4	2	0.630500	0.324000	0.303000	0.397333
		class index	center <x, y>		scale <width, height>	

Hình 4.4 Nội dung một tệp annotation

Nội dung của file annotation sẽ bao gồm:

<class-index> <center-x> <center-y> <bbbox-width> <bbbox-height>

Trong đó: các giá trị *<center-x> <center-y> <bbbox-width> <bbbox-height>* là tâm và kích thước width (chiều rộng), height (chiều cao) của bounding box đã được chuẩn hóa bằng cách chia cho chiều rộng và chiều cao của ảnh, do đó các giá trị này luôn nằm trong khoảng [0, 1]. *<class-index>* là giá trị index đánh dấu các classes. Trong

⁸ <https://pypi.org/project/labelImg/>

trường hợp một ảnh có nhiều bounding box thì tệp annotation sẽ gồm nhiều dòng, mỗi một bounding box là một dòng.

4.4.3 Cấu hình các tệp cần thiết

4.4.3.1 Tạo tệp train/validation

Ta sẽ tạo ra 2 tệp **train.txt** và **val.txt** chứa dữ liệu đường dẫn tới các file hình ảnh nằm trong tập train (chứa dữ liệu huấn luyện) và validation (chứa dữ liệu kiểm thử) bằng một đoạn script viết bằng Python. Khi đó đối với mỗi ảnh thì đường dẫn của nó nằm trên một dòng khác nhau.

```
data/images/IMG_20210601_101240.jpg  
data/images/IMG_20210601_101247.jpg  
data/images/IMG_20210601_101309.jpg  
data/images/IMG_20210601_101319.jpg  
data/images/IMG_20210601_101324.jpg
```

Hình 4.5 Nội dung trong tệp train.txt

4.4.3.2 Tạo file object name

Đây là tệp chứa tên các lớp mà chúng ta sẽ huấn luyện mô hình. Trên file này, thứ tự các tên class cần phải đặt đúng với index của nó trong các tệp nhãn của vật thể, ta tạo tệp **obj.names** và ghi tên 2 lớp ta cần huấn luyện là **empty** (còn trống) và **occupied** (đã đỗ), với mỗi lớp nằm trên mỗi dòng khác nhau.

4.4.3.3 Tạo tệp config data

Tệp config data (tên tệp là **obj.data**) sẽ khai báo một số thông tin như:

- Số lượng các lớp
- Đường dẫn tới các tệp **train.txt**, **val.txt**
- Đường dẫn tới file **obj.names**
- Thư mục backup mô hình huấn luyện.

4.4.3.4 Tạo anchor

Ta sẽ tạo anchor mới cho tập dữ liệu mô phỏng bằng thuật toán phân cụm k-mean bằng đoạn script được tác giả cung cấp sẵn, kết quả phân tích tạo ra kết quả như sau:

```
C:\WINDOWS\system32\cmd.exe - darknet.exe detector calc_anchors data/obj.data -num_of_clusters 9 -width 640 -height 640
C:\Users\Hiep\Videos\Project\darknet-master\build\darknet\x64>darknet.exe detector calc_anchors data/obj.data
-num_of_clusters 9 -width 640 -height 640
CUDA-version: 11030 (11030), cuDNN: 8.1.1, CUDNN_HALF=1, GPU count: 1
CUDNN_HALF=1
OpenCV version: 3.4.0

num_of_clusters = 9, width = 640, height = 640
read labels from 202 images
loaded      image: 202      box: 1818
all loaded.

calculating k-means++ ...

iterations = 6

counters_per_class = 1335, 483

avg IoU = 98.41 %

Saving anchors to the file: anchors.txt
anchors = 71,137, 71,137, 71,137, 71,137, 71,137, 85,163, 125,120, 99,164, 107,161
```

Hình 4.6 Kết quả 9 điểm neo mới được sinh ra từ thuật toán k-mean

4.4.3.5 Chỉnh sửa tệp config model

Để sử dụng mô hình Darknet, ta clone repository của tác giả Joseph Redmon tại địa chỉ <https://github.com/pjreddie/darknet> về máy.

Ta chỉnh sửa tệp **yolov3.cfg** tại thư mục **cfg**, đây là tệp cấu hình của YOLOv3, chứa thông tin về số lớp, filter, các lớp tích chập, các thông số học,... Ta chỉnh sửa lại như sau:

- Tại dòng 6: Thay đổi *batch=64*. Nghĩa là: batch = số ảnh (cả tệp annotation) được đưa vào huấn luyện trong một batch.
- Tại dòng 7: Thay đổi *subdivisions=16*. Trong một batch được chia thành nhiều block, mỗi block chứa batch/subdivisions ảnh được đưa vào GPU xử lý tại một thời điểm. Weights của mô hình được update sau mỗi batch.
- Tại các dòng 8, 9: Thay *width, height* thành *640* đối với tập dữ liệu lớn và *416* đối với tập dữ liệu mô phỏng (đây là giá trị trung bình mang giá trị ngữ nghĩa cao kể cả sau này khi tăng hay giảm đi kích thước ảnh và phù hợp cho cấu hình máy tính hiện tại, trong đó giá trị kích thước càng cao thì độ chính xác càng cao và ngược lại).
- Tại các dòng 610, 696, 783: Thay thành *classes=2* là số lượng classes (lớp) chúng ta huấn luyện.

- Tại các dòng 603, 689, 776: Thay số lượng *filter=21*. Đây chính là layer cuối cùng của base network. Do đó chúng có output shape thay đổi theo số lượng classes theo đúng công thức của bài trước đó là: $(n_classes + 5) \times 3 = (2+5) \times 3 = 21$.

- *max_batches*: tại dòng 20 là số lượng batch tối đa để huấn luyện mô hình YOLO. Theo như ý kiến của tác giả thì $max_batches = n_classes \times 2000$, và không được nhỏ hơn số lượng ảnh. Đối với tập dữ liệu ảnh có số lượng ảnh gần 9000 nên ta đặt $max_batches = 9000$, và đối với tập dữ liệu mô phỏng thì $max_batches = 4000$.

- *burn_in*: Tại dòng 19 là số lượng batch ban đầu được giữ sao cho *learning_rate* rất bé. Giá trị này sẽ tăng dần từ 0 đến *learning_rate*. Sau đó *learning_rate* sẽ được giữ ổn định. Thực nghiệm cho thấy thiết lập *learning_rate* bé ở những steps đầu sẽ giúp cho thuật toán hội tụ nhanh hơn. Đối với tập dữ liệu lớn *max_batches* chỉ là 9000 nên cần điều chỉnh giảm $burn_in = 200$, còn đối với tập dữ liệu mô phỏng $burn_in = 80$.

- *steps*: Tại dòng 22. Theo ý kiến của tác giả, thì số lượng *steps* chiếm 80% và 90% của *max_batches*, đối với tập dữ liệu lớn điều chỉnh về $steps=7200, 8100$. Còn đối với tập dữ liệu mô phỏng $steps=3200, 3600$. Đây là các vị trí step mà chúng ta sẽ bắt đầu giảm dần *learning_rate* vì thuật toán đã đạt tới điểm hội tụ nên không cần thiết lập *learning_rate* quá cao.

- Tại các dòng 609, 695, 782: Thay thành *anchors = 71,137, 71,137, 71,137, 71,137, 71,137, 85,163, 125,120, 99,164, 107,161* (chỉ tập dữ liệu mô phỏng).

Ngoài những thuộc tính được chỉnh sửa như trên thì còn nhiều thuộc tính khác được để giá trị mặc định như: *momentum=0.9, learning_rate=0.001, channels=3, ...*

4.4.4 Huấn luyện mô hình

4.4.4.1 Tiến hành huấn luyện

Tiếp theo ta sẽ tải toàn bộ mô hình Darknet lên Google Drive sau đó dùng Google Colab mount dữ liệu qua thư mục ở Google Drive (vì dữ liệu lưu trên Colab chỉ lưu được khá ngắn, và vì quá trình train lâu nên dễ bị lỗi gián đoạn làm dừng quá trình train, dễ bị mất dữ liệu khi bị ngừng session) và tiến hành quá trình *make* các tệp cần thiết trên mô hình Darknet. Cuối cùng ta huấn luyện mô hình bằng cú pháp:

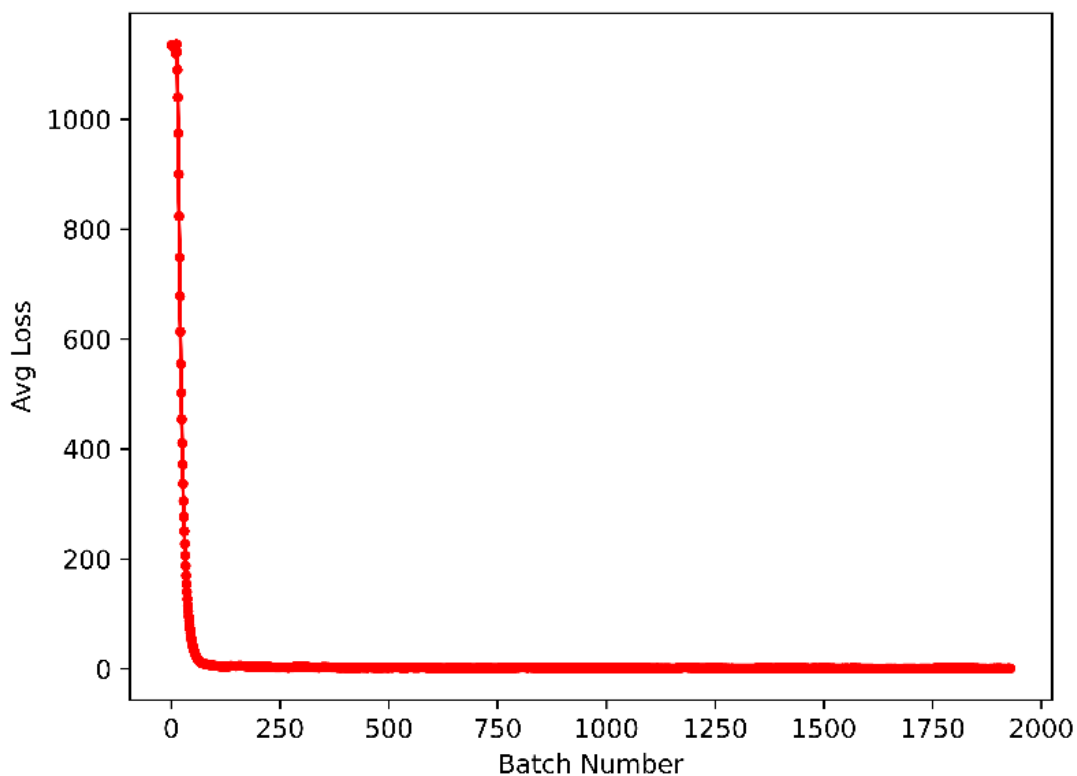
```
!./darknet detector train [tệp data config] [tệp model config] [pretrain-model weights]  
> [tệp lưu log]
```

Trong đó: Tập *data config*, *model config* đã được mô tả ở trên.

- *pretrain-model weights*: Chúng ta sẽ sử dụng pretrain model darknet53.conv.74 được huấn luyện từ bộ dữ liệu ImageNet để khởi tạo các trọng số sử dụng *Fine-tuning*.
- *tệp lưu log*: Tập lưu giữ các giá trị output của mô hình.

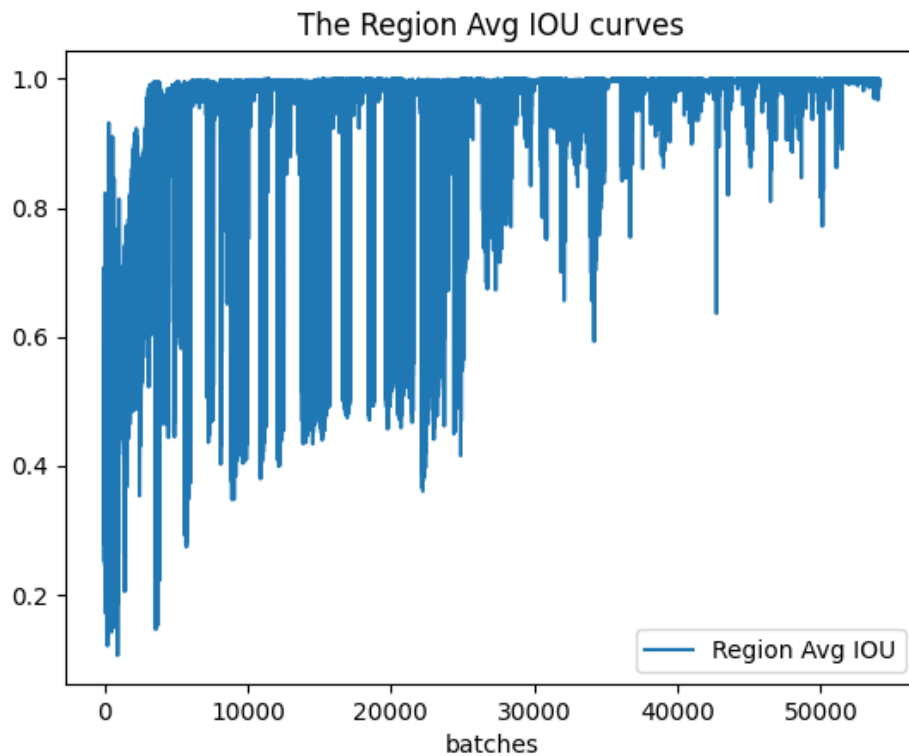
Bất cứ khi nào quá trình huấn luyện bị dừng đột ngột, ta có thể train tiếp từ điểm checkpoint được lưu từ tệp *.weights* hay *.backup* cuối cùng trong thư mục **backup** được sinh ra sau mỗi 100 batch cập nhật sau 1000 vòng lặp, bằng cú pháp sau:

```
!./darknet detector train [tệp data config] [tệp model config] backup/[last weights] > [tệp lưu log]
```



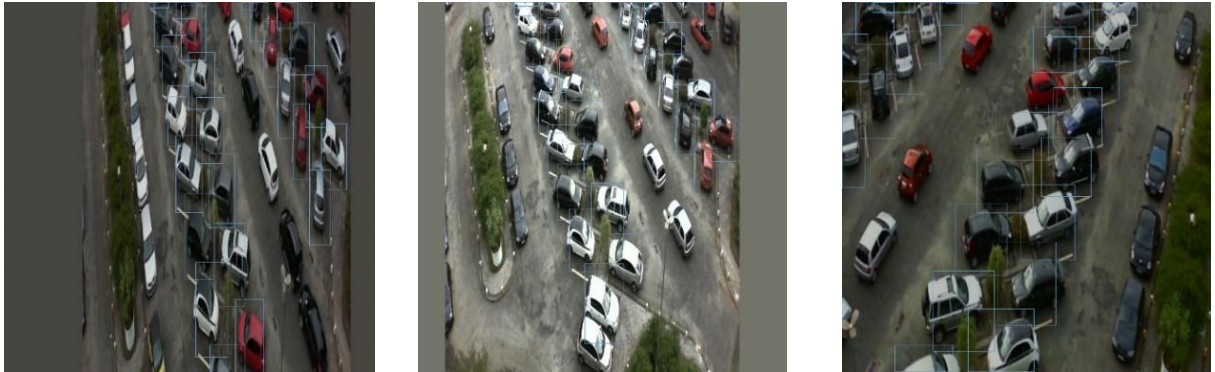
Hình 4.7 Trực quan hoá hàm mất mát sau gần 2000 batch của tập dữ liệu mô phỏng

Ta có thể thấy sau khoảng 80 batch thì thuật toán đã bắt đầu hội tụ đúng như giá trị *burn_in* trong cấu hình tệp *yolov3.cfg*. Sau khoảng vài ngàn lần lặp thì giá trị *avg_loss* không thấy tăng và đã rất nhỏ thì ta có thể dừng quá trình train sớm bởi vì lúc này thuật toán đã hội tụ và giá trị hàm mất mát đã tiệm cận 0.



Hình 4.8 Trực quan hoá Region Avg IOU của tập dữ liệu mô phỏng

Region Avg IOU: Cho biết IOU trung bình của hình ảnh trong subdivision hiện tại, ta có thể thấy thì chỉ số IOU dần chính xác hơn ở các số batch lớn hơn bởi vì quá trình học diễn ra thuận lợi và đúng với dự định ban đầu.



Hình 4.9 Các phương pháp data augmentation được sử dụng trong quá trình học

Tạo ra nhiều mẫu huấn luyện hơn bằng cách sử dụng các thông số *saturation*, *exposure*, *hue* (trong hệ màu HSV), và trong đó có thông số *random* sẽ huấn luyện mạng ở các độ phân giải khác nhau, từ đó tăng độ chính xác của mô hình.

```

C:\WINDOWS\system32\cmd.exe
97 upsample          2x   40 x 40 x 128 -> 80 x 80 x 128
98 route 97 36              -> 80 x 80 x 384
99 conv 128             1 x 1/ 1   80 x 80 x 384 -> 80 x 80 x 128 0.629 BF
100 conv 256           3 x 3/ 1   80 x 80 x 128 -> 80 x 80 x 256 3.775 BF
101 conv 128           1 x 1/ 1   80 x 80 x 256 -> 80 x 80 x 128 0.419 BF
102 conv 256           3 x 3/ 1   80 x 80 x 128 -> 80 x 80 x 256 3.775 BF
103 conv 128           1 x 1/ 1   80 x 80 x 256 -> 80 x 80 x 128 0.419 BF
104 conv 256           3 x 3/ 1   80 x 80 x 128 -> 80 x 80 x 256 3.775 BF
105 conv 21            1 x 1/ 1   80 x 80 x 256 -> 80 x 80 x 21 0.069 BF
106 yolo

[yolo] params: iou_loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 154.584
avg_outputs = 1223484
Allocate additional workspace_size = 14.75 MB
loading weights from yolov3_final.weights...
seen 64, trained: 576 K-images (9 Kilo-batches_64)
Done! Loaded 107 layers from weights-file

calculation mAP (mean average precision)...
Detection layer: 82 - type = 28
Detection layer: 94 - type = 28
Detection layer: 106 - type = 28
872
detections_count = 57917, unique_truth_count = 49732
class_id = 0, name = space-empty, ap = 99.76% (TP = 26353, FP = 300)
class_id = 1, name = space-occupied, ap = 99.67% (TP = 23053, FP = 721)

for conf_thresh = 0.25, precision = 0.98, recall = 0.99, F1-score = 0.99
for conf_thresh = 0.25, TP = 49406, FP = 1021, FN = 326, average IoU = 84.26 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.997165, or 99.72 %
Total Detection Time: 653 Seconds

```

Hình 4.10 Kết quả tính toán $mAP@50 = 99.72\%$ trên 872 ảnh của tập dữ liệu lớn

Tập *weights*: Một số định dạng của các tập lưu trọng số đó là *.weights*, *.backup* hay có thể là số như tập pretrained model *darknet53.conv.74* như ở trên đã giới thiệu. Về bản chất thì các loại định dạng tập này chỉ dùng để phân biệt chúng với nhau vì chúng đều là những tập nhị phân. Nhưng tập pretrained chỉ lưu giữ kiến trúc backbone, được sử dụng 1 lần cho kỹ thuật fine-tuning. Cấu trúc tập *.weight* bao gồm:

- Tham số phiên bản major (kiểu int, 4 byte)
- Tham số số phiên bản minor (kiểu int, 4 byte)
- Bản sửa đổi tham số số phiên bản (kiểu int, 4 byte)
- Số lượng ảnh được huấn luyện đã nhìn thấy (kiểu size_t, loại 8 byte trong hệ điều hành 64 bit)
- Tham số lớp chuyển đổi: trọng số chập (kiểu float, mỗi tham số 4 byte).

4.4.4.2 Một số thông tin về output của mô hình

- Với tập dữ liệu lớn tổng cộng mô hình cần dự đoán số bounding box tối đa ở 3 tỷ lệ là: $(20 \times 20 + 40 \times 40 + 80 \times 80) \times 3 = 25200$ mỗi ảnh; với mỗi tỷ lệ dự đoán tổng cộng $5 + 2 = 7$ tham số đối với mỗi đối tượng trên mỗi bounding box trong mỗi ảnh. Ta sử dụng batch = 9000, với mỗi batch ta có 64 ảnh, vậy số bounding box

tối đa cần dùng để dự đoán trong toàn bộ quá trình huấn luyện là $25200 * 9000 * 64 = 1.45152E10$ (với $E10 = 10^{10}$). Số ảnh được tạo ra trong quá trình huấn luyện tối đa là $9000 * 64 = 576000$ ảnh.

- Với tập dữ liệu mô phỏng tổng cộng mô hình cần dự đoán số bounding box tối đa ở 3 tỷ lệ là: $(13*13 + 26*26 + 52*52) * 3 = 10647$ mỗi ảnh; với mỗi tỷ lệ dự đoán tổng cộng $5 + 2 = 7$ tham số đối với mỗi đối tượng trên mỗi bounding box trong mỗi ảnh. Ta sử dụng batch = 4000, với mỗi batch ta có 64 ảnh, vậy số bounding box tối đa cần dùng để dự đoán trong toàn bộ quá trình huấn luyện là $10647 * 4000 * 64 = 2.725632E9$ (với $E9 = 10^9$). Số ảnh được tạo ra trong quá trình huấn luyện tối đa là $4000 * 64 = 256000$ ảnh.
- Ví dụ output của quá trình huấn luyện có dạng:

“19: 0.783273, 0.827796 avg, 0.000020 rate, 1.650999 seconds, 339 images

*Region 82 Avg IOU: 0.759600, Class: 0.809470, Obj: 0.732717, No Obj: 0.002799,
.5R: 1.000000, .75R: 0.500000, count: 4*

*Region 94 Avg IOU: 0.699416, Class: 0.917663, Obj: 0.226457, No Obj: 0.000643,
.5R: 1.000000, .75R: 0.333333, count: 6*

*Region 106 Avg IOU: 0.667185, Class: 0.959919, Obj: 0.089371, No Obj: 0.000099,
.5R: 1.000000, .75R: 0.000000, count: 1”*

Giải thích:

- *19*: cho biết số lần lặp của quá trình huấn luyện hiện tại;
- *0.783273* là giá trị mất mát tổng thể (hàm mất mát);
- *0.827796 avg* là giá trị mất mát trung bình;
- *0.000020 rate* thể hiện tốc độ học (learning rate) hiện tại.
- *1.650999 seconds* cho biết tổng thời gian huấn luyện theo batch hiện tại;
- *339 images* là tổng số ảnh được tạo ra tính tới hiện tại.
- *Region 82 Avg IOU: 0.759600* cho biết IOU trung bình của hình ảnh trong subdivision hiện tại (bằng 75%), ở lớp thứ 82, dự đoán đối tượng có kích thước nhỏ.

- *Region 94 Avg IOU: 0.699416* Cho biết IOU trung bình của hình ảnh trong subdivision hiện tại (bằng 69%), ở lớp thứ 94, dự đoán đối tượng có kích thước trung bình.
- *Region 106 Avg IOU: 0.667185* cho biết IOU trung bình của hình ảnh trong subdivision hiện tại (bằng 66%), ở lớp thứ 106, dự đoán đối tượng có kích thước lớn.
- *Class: 0.809470* là tỷ lệ chính xác phân loại nhãn của đối tượng.
- *Obj: 0.732717* là giá trị trung bình của điểm đối tượng.
- *No Obj: 0.002799* là giá trị trung bình của điểm không có đối tượng.
- *.5R: 1.000000* là tỷ lệ các mẫu positive có IOU lớn hơn 50%.
- *.75R: 0.500000* là tỷ lệ các mẫu positive có IOU lớn hơn 75%.
- *count: 4* cho biết số lượng hình ảnh huấn luyện có các mẫu positive thực sự trong tất cả các hình của subdivision hiện tại.

4.5 Triển khai thuật toán YOLOv3 bằng Pytorch và xử lý website

4.5.1 Triển khai thuật toán YOLOv3

4.5.1.1 Xây dựng backbone

Quá trình triển khai thuật toán bằng Pytorch có sử dụng một số lớp cơ bản như *nn.Module*, *nn.Sequential* và *torch.nn.parameter*. Trong đó các mạng neural sẽ được xây dựng dựa trên package *torch.nn* và *nn.Module* sẽ bao gồm các layers và một phương thức **forward (input)** để trả ra kết quả **output**. *nn.Sequential* là một lớp container (chứa) mở rộng lớp cơ sở *nn.Module* và cho phép chúng ta compose (dàn xếp, kết hợp) các module với nhau. *torch.nn.parameter* khi được sử dụng với *nn.Module* sẽ tự động được thêm vào danh sách các tham số của nó.

Cấu trúc thư mục mô hình của mô hình YOLOv3 được triển khai như sau:

- **colors:** Chứa tệp nhị phân định dạng màu
pallette
- **config:** Chứa tệp cấu hình mạng
yolov3.cfg
- **data:** Chứa tệp danh sách tên lớp

parking.names

- **outputs:** Chứa ảnh đã được xử lý khi dùng module phát hiện ảnh
- **util:** Chứa kiến trúc backbone của mô hình

__init__.py

image_processor.py

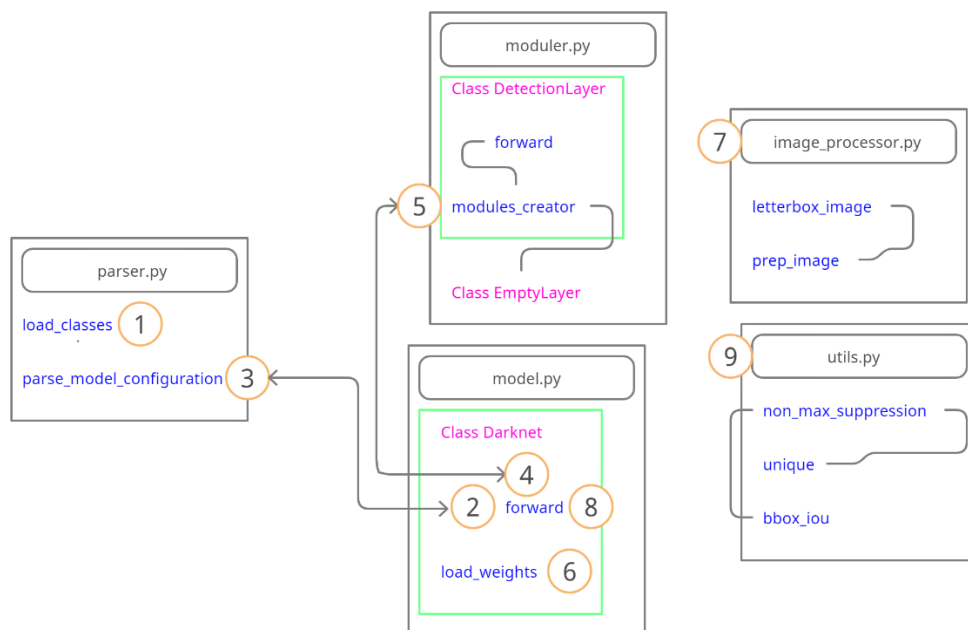
model.py

moduler.py

parser.py

utils.py

- **weights:** Chứa tệp weights của mô hình



Hình 4.11 Sơ đồ quá trình xử lý trong backbone

4.5.1.2 Xây dựng trình phát hiện kiểm tra suy luận của mô hình

Đối với module phát hiện đối tượng trong ảnh:

Ta tạo tệp *images.py*, đầu tiên ta import các module cần thiết ở phần backbone và thư viện opencv, sử dụng module ArgParse của Python để nhận vào đối số đường dẫn tới tệp hoặc thư mục ảnh cần phát hiện. Khởi tạo các siêu tham số như *outputs*, *batch_size*, *confidence*, *nms_thesh*, *classes* tương ứng với thư mục sẽ xuất ảnh, kích thước batch, độ tự tin, ngưỡng của NMS, các lớp cần phát hiện. Sau đó tải tệp cấu hình và tệp weights. Kiểm tra xem máy có hỗ trợ CUDA không, nếu có thì đặt mô hình vào

CUDA để GPU xử lý, gán một số cờ (flag) cần thiết ở các vị trí khác nhau để đo tốc độ đọc ảnh. Dùng opencv đọc hình ảnh từ đĩa hoặc hình ảnh từ một thư mục, nếu sai đường dẫn hay tên tệp thì thông báo lỗi và thoát ra khỏi chương trình. Các đường dẫn của ảnh được lưu trữ trong một danh sách được gọi là *imlist*. Tạo batch và bắt đầu vòng lặp phát hiện cho toàn bộ số ảnh. Sử dụng hàm Counter từ module collections của Python để đếm các đối tượng được nhận diện được để tiến hành đếm các chỗ trống và đã đỗ. Còn một cách nữa để đếm các đối tượng đó là lưu các đối tượng nhận diện được vào một list, sau đó dùng vòng lặp để đếm tần suất xuất hiện của các đối tượng để tính tổng. Tiếp đó vẽ bounding box, in ra nhãn và xuất ảnh bằng *cv2.imwrite*. Nếu không có bất kì phát hiện nào thì thông báo cho người dùng và thoát khỏi chương trình. Cuối cùng in ra những thông tin log tổng hợp về quá trình phát hiện, sử dụng tính năng đồng bộ hoá của CUDA để tính thời gian chạy.

Kết quả dưới đây được chạy trên laptop với GPU NVIDIA GeForce 940MX (2GB VRAM), CPU Intel i5-8250U (8GB RAM). Đây là lần đầu quá trình phát hiện cho tệp ảnh được chạy nên tốc độ có thể chậm hơn một chút so với chạy các lần sau nếu có nhiều ảnh vì mô hình mới được khởi tạo lần đầu.



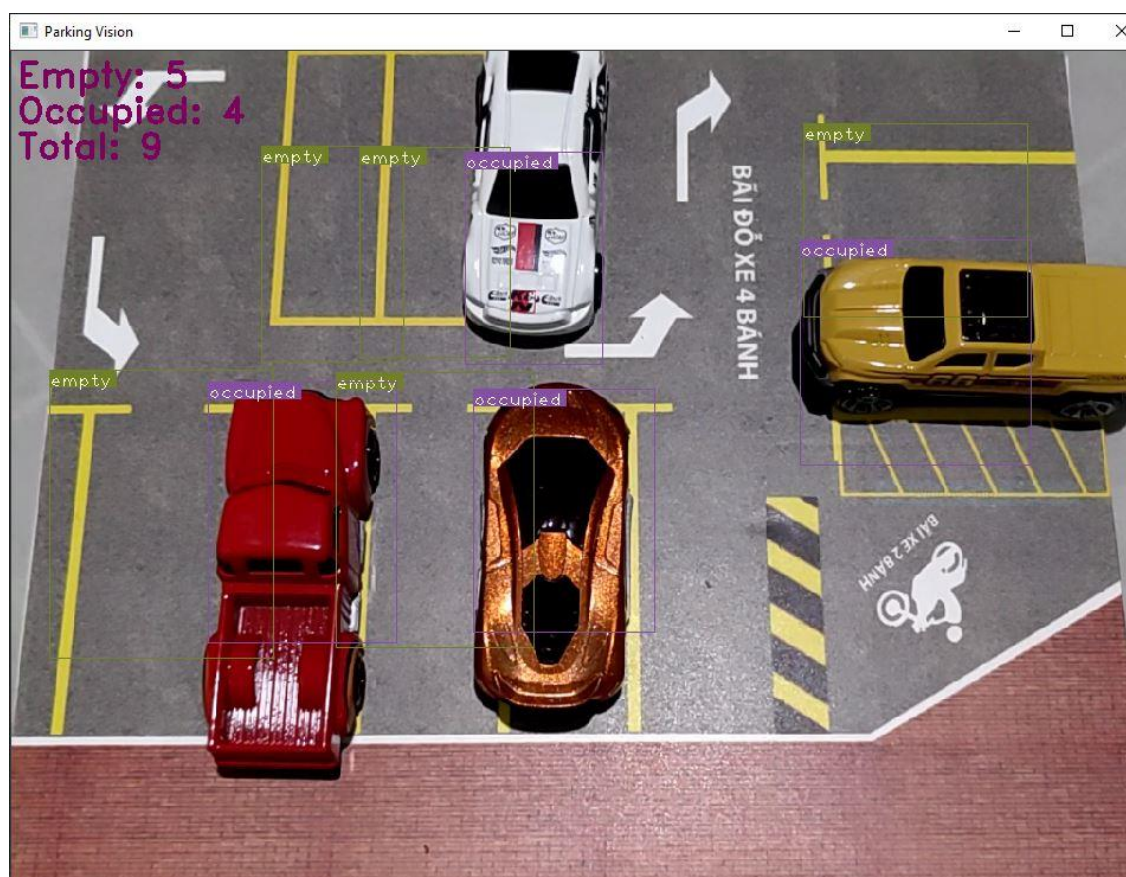
Hình 4.12 Output của trình phát hiện ảnh

Đối với module phát hiện đối tượng trong video/camera:

Đối với trình phát hiện đối tượng trong video/camera thì mã nguồn không thay đổi nhiều, thay vì lặp qua thư mục thông qua các batch, thì ở đây ta sẽ làm việc với

frame (khung hình) của video, vì bản chất thì video cũng chỉ là một dãy các *frame* hay có thể xem là các bức ảnh được nối tiếp nhau trong thời gian nhất định.

Ta tạo tệp *cam.py* và gọi nó là một *Python App*, đầu tiên ta tiến hành đọc video/camera bằng *opencv*, ta lặp qua các khung hình giống như cách đã làm với thư mục ảnh. Các mã được đơn giản hoá ở nhiều nơi vì không cần phải xử lý theo batch mà chỉ có một ảnh (khung hình) tại một thời điểm. Sau mỗi lần lặp lại ta theo dõi số lượng khung hình (*frame*) bằng biến *frames*, sau đó ta chia số này cho thời gian trôi qua kể từ khung hình đầu tiên để in tỷ lệ khung hình trên giây (*frames per second* - FPS) của video. Cuối cùng ta sử dụng *cv2.imshow* để hiện khung hình này lên *Python App* với những bounding box được vẽ trên đó, vòng lặp sẽ liên tục cho đến khi người dùng dừng chương trình.



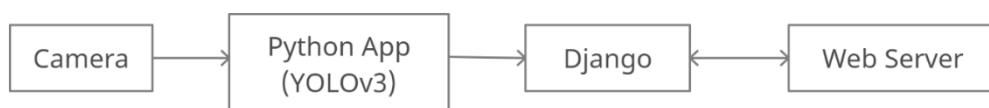
Hình 4.13 Trình phát hiện đối tượng chạy trên *Python App*

4.5.2 Xử lý website

Đầu tiên ta sẽ tạo trước các template gồm các tệp HTML được chia thành 4 tệp đó là: *home.html* chứa nội dung trang chủ và giới thiệu sản phẩm, *camera.html* chứa

phần stream camera và hình ảnh trả về từ hàm **video_feed()**, *contact.html* chứa thông tin liên hệ và cuối cùng là *base.html* chứa header và footer chung của các trang trên. Về phần xử lý stream có các phương án xử lý khác nhau. Phương án đầu tiên ta xử lý tệp *view.py* của Django bằng hàm **stream()** với mã gần giống hoàn toàn tệp *cam.py* của thuật toán YOLOv3, tạo hình ảnh và sử dụng phương thức *yield* để trả lại các frame ảnh, tiếp đó dùng hàm **video_feed()** với phương thức *StreamingHttpResponse* để stream các frame ảnh đó lên website. Tuy vậy phương án này tồn tại một lỗ hổng lớn đó là tiêu tốn quá nhiều tài nguyên hệ thống, vì với mỗi yêu cầu từ người dùng thì Web Server phải chạy lại từ đầu gây lãng phí, điều đó dẫn đến vấn đề là không cần một cuộc tấn công từ chối dịch vụ (DDOS)⁹ và với một vài người dùng truy cập vào website thì website của chúng ta cũng tự sập vì tài nguyên hệ thống không đủ để đáp ứng cho thuật toán xử lý nặng như YOLOv3 với mỗi yêu cầu lặp lại như vậy. Điều đó cần một phương án giải quyết thứ 2 đó là tách YOLOv3 ra khỏi Django, sử dụng Python App của tệp *cam.py* chạy trước để liên tục sinh ra các hình ảnh từ các *frame*, Django liên tục nhận các hình ảnh đó và hiển thị lên website bằng các hàm tương tự với phương án đầu nói trên, sử dụng *gzip* với hàm **video_feed()** nén các tệp để website tải nhanh hơn. Phương án này đáp ứng cả 2 yêu cầu đó là vừa chạy được Python App song song với Django giúp người quản lý có thể vừa xem được hình ảnh trực tiếp từ Python App và người dùng có thể xem trên website. Cuối cùng thêm các URL đã được định nghĩa từ hàm *view.py* vào tệp *url.py* của Django.

Quá trình xử lý trong Django biểu diễn bằng sơ đồ sau:



Hình 4.14 Quá trình xử lý trong Django

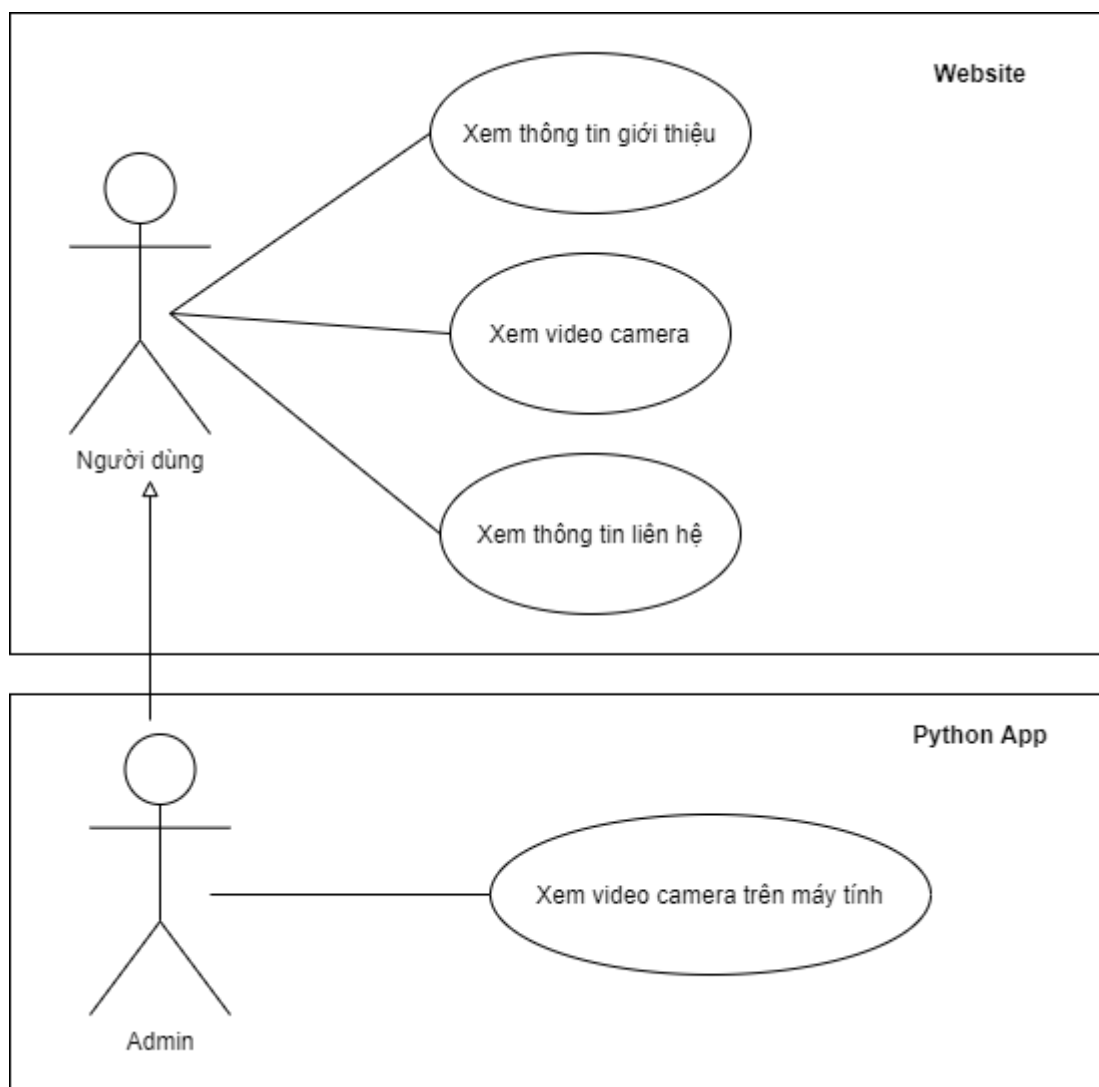
4.6 Sơ đồ use case của hệ thống

Biểu đồ usecase mô tả các chức năng của hệ thống, từ đó chỉ ra những công việc mà hệ thống cần làm để đáp ứng nhu cầu của người sử dụng.

⁹ DDOS (Denial Of Service) là hình thức tấn công từ chối dịch vụ khá phổ biến, nó khiến cho máy tính mục tiêu không thể xử lý kịp các tác vụ và dẫn đến quá tải.

Người quản trị (Admin) có thể chạy Website và Python App song song với nhau và có thể xem được cả 2.

Người dùng có thể xem thông tin giới thiệu, video camera và thông tin liên hệ.

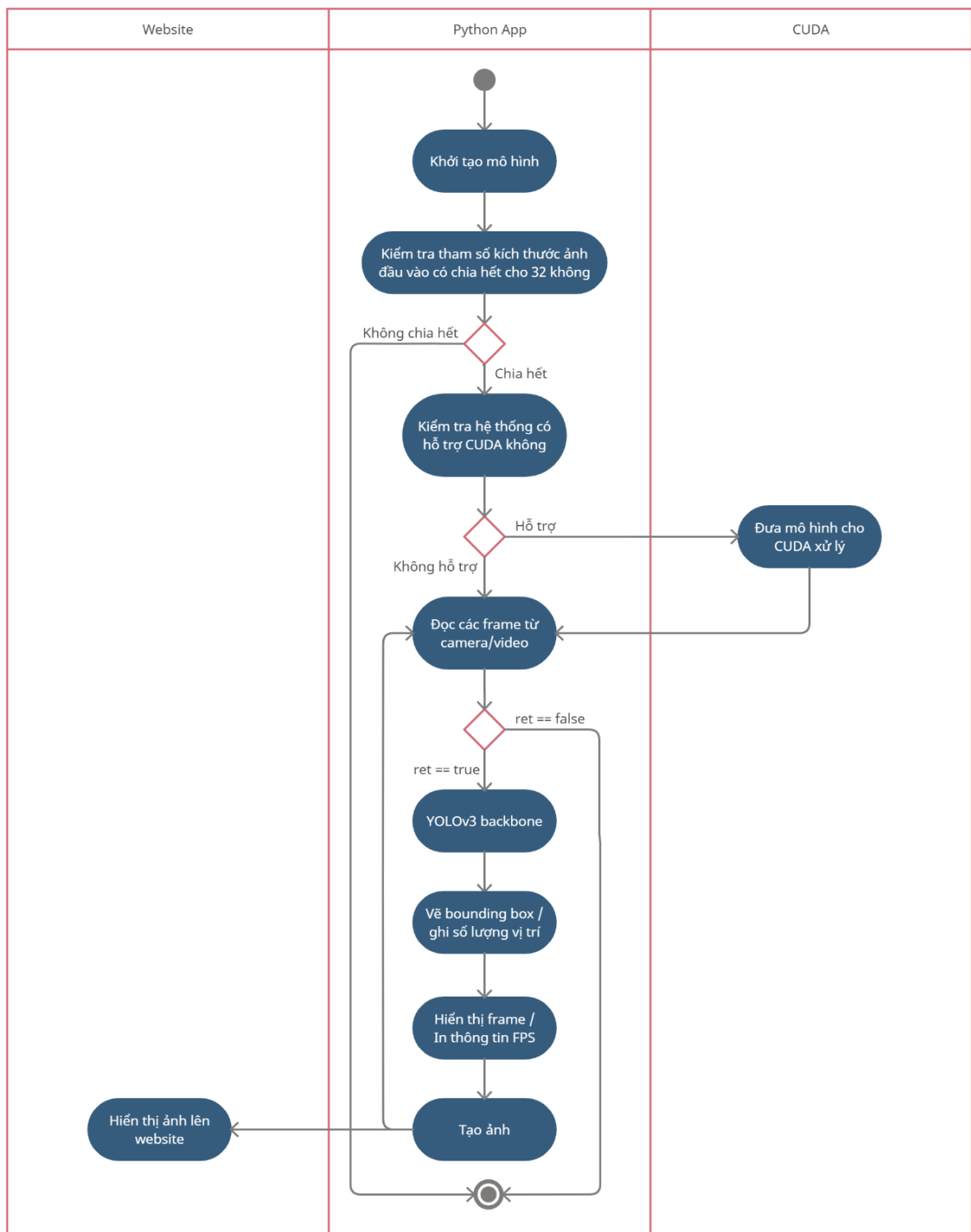


Hình 4.15 Sơ đồ use case hệ thống

4.7 Sơ đồ hoạt động

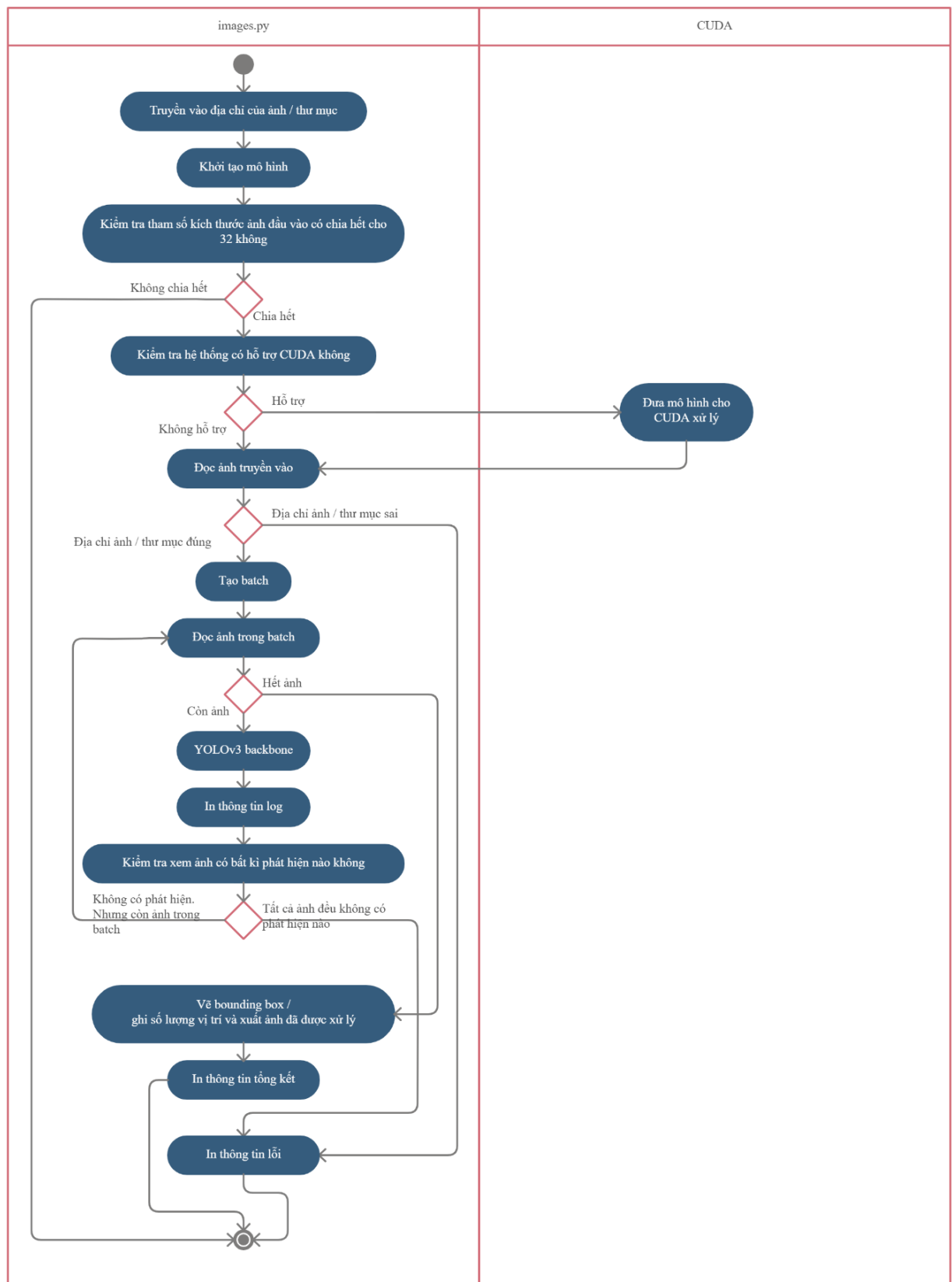
Sơ đồ hoạt động là một mô hình logic được dùng để mô hình hoá cho các hoạt động trong một quy trình nghiệp vụ. Nó chỉ ra luồng đi từ hoạt động này sang hoạt động khác trong một hệ thống. Nó đặc biệt quan trọng trong việc xây dựng mô hình chức năng của hệ thống và nhấn mạnh tới việc chuyển đổi quyền kiểm soát giữa các đối tượng.

4.7.1 Sơ đồ hoạt động truyền video lên website và Python App



Hình 4.16 Sơ đồ hoạt động truyền ảnh lên website

4.7.2 Sơ đồ hoạt động xử lý ảnh / thư mục ảnh



Hình 4.17 Sơ đồ hoạt động xử lý ảnh / thư mục ảnh

4.8 Xây dựng giao diện chương trình

4.8.1 Giao diện trang chủ website – giới thiệu sản phẩm



Hình 4.18 Giao diện trang chủ

Các thông tin giới thiệu sản phẩm được thiết kế trên cùng một trang với trang chủ, thông tin ngắn gọn súc tích.



Hình 4.19 Giao diện giới thiệu sản phẩm dạng slide

[HOME](#)
[CAMERA](#)
[LIÊN HỆ](#)

Tại sao cần bãi đỗ xe thông minh?

Giải pháp bãi đỗ xe thông minh áp dụng công nghệ tự động là một trong những giải pháp đang rất được quan tâm hiện nay, đặc biệt là tại các thành phố lớn nơi có lượng phương tiện giao thông đang tăng nhanh một cách chóng mặt, tìm được một chỗ đậu xe ô tô trong thời buổi hiện tại cũng là vấn đề khá khó khăn. Đôi khi, có thể có chỗ trống trong bãi đỗ xe nhưng tài xế sẽ không biết nó nằm ở đâu, hoặc là không biết trong bãi có còn chỗ trống hay không. Rất nhiều vụ ão xe trước nhà dân dẫn đến nhiều bất cập.

Đối với chủ đầu tư	Giúp chủ đầu tư quản lý bãi giữ xe một cách chuyên nghiệp, khoa học với các công nghệ hiện đại, đồng thời giảm chi phí chủ đầu tư.
Đối với người sử dụng	Thuận tiện cho người sử dụng, có thể nắm bắt số lượng chỗ trống và vị trí chính xác trong bãi, giúp chủ xe giảm thời gian tìm kiếm, tiết kiệm nhiên liệu, giúp tâm trạng thoải mái hơn thay vì tự mình tìm xem vị trí nào còn trống. Hơn nữa người sử dụng có thể theo dõi tình trạng xe mình ở mọi lúc mọi nơi.
Đối với người quản lý	Thuận tiện cho người quản lý bãi xe, giúp cho họ nắm bắt thông tin số lượng chỗ trống hiện tại, chính xác vị trí còn trống để chỉ dẫn cho các tài xế vào bãi, tạo điều kiện thuận lợi cho việc điều hành các phương tiện di chuyển.

Hình 4.20 Giao diện giới thiệu sản phẩm

4.8.2 Giao diện xem video camera theo thời gian thực

Sau khi người dùng chọn chức năng xem camera thì sẽ gửi yêu cầu cho máy chủ xử lý và trả về dữ liệu video đang diễn ra tại bãi đỗ xe theo thời gian thực.

[HOME](#)
[CAMERA](#)
[LIÊN HỆ](#)

Camera Realtime

Empty: 5
Occupied: 4
Total: 9

Bãi Đỗ Xe 4 Bánh

! Lưu ý: Nếu không thấy ảnh hay ảnh không đổi thì vui lòng tải lại trang

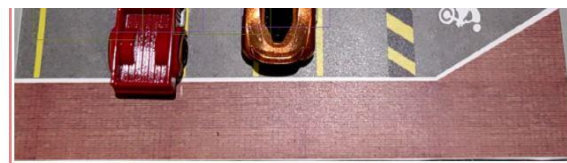
CHÚ THÍCH:

- Empty - Tương ứng với vị trí còn trống
- Occupied - Tương ứng với vị trí đã ão
- Total - Tương ứng với tổng vị trí trong bãi

[Home](#)
[Camera](#)
[Liên Hệ](#)

© 2021 All rights reserved

Hình 4.21 Giao diện xem camera theo thời gian thực



! Lưu ý: Nếu không thấy ảnh hay ảnh không đổi thì vui lòng tải lại trang

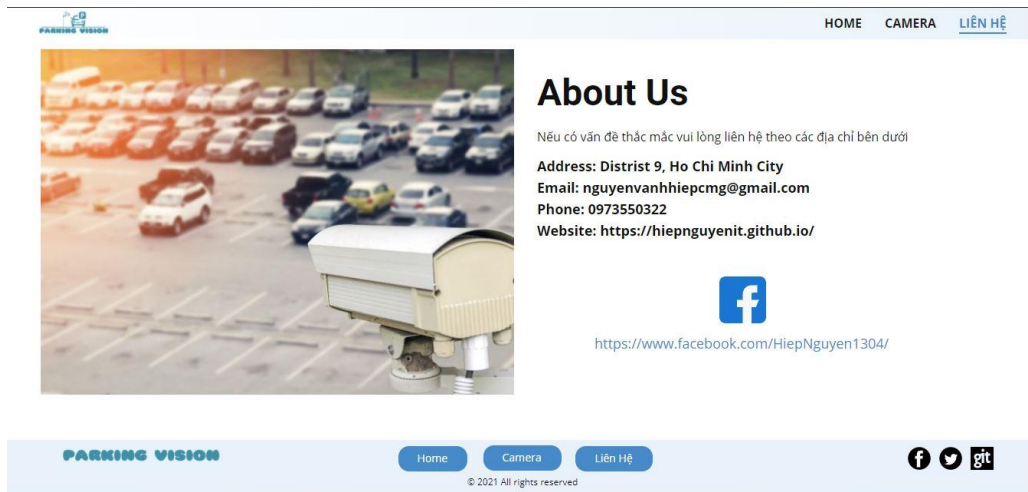
CHÚ THÍCH:

- Empty - Tương ứng với vị trí còn trống
- Occupied - Tương ứng với vị trí đã ão
- Total - Tương ứng với tổng vị trí trong bãi

Hình 4.22 Giao diện chú thích cho camera theo thời gian thực

4.8.3 Giao diện thông tin liên hệ

Chứa thông tin liên hệ của chúng ta.



Hình 4.23 Giao diện thông tin liên hệ

KẾT LUẬN

Kết quả đạt được

Sau khi tìm hiểu, nguyên cứu, phân tích, thực hiện và kiểm nghiệm trên thực tế, đề tài đã được những yêu cầu ở mục tiêu đã đề ra:

- Xây dựng được hệ thống hướng dẫn bãi đỗ xe ô tô với các chức năng tiện lợi và cần thiết như:

- Cho phép người dùng theo dõi trực tiếp tình hình hiện tại tại bãi đỗ xe trên website. Người quản lý có thể xem trực tiếp tình hình trên màn hình Python App cho tốc độ cao và dễ quan sát hơn.
- Admin có thể chạy song song cả Python App và website. Và có thể dùng module xử lý ảnh để kiểm tra tính chính xác của mô hình khi huấn luyện.

- Bộ phát hiện đối tượng có thể phát hiện ở các điều kiện thời tiết, ánh sáng với độ chính xác cao, có thể chạy ổn định chuyển đổi linh hoạt giữa CPU và GPU. Website có giao diện thân thiện, tính tương thích với điện thoại, máy tính cao, dễ sử dụng, đồng thời website cũng thực hiện tốt trên hầu hết các trình duyệt web như: Chrome, Edge,...

Nhược điểm

- + Do website được xử lý theo dạng lấy kết quả từ quá trình sinh ảnh từ Python application và sau đó hiển thị lên bằng thẻ html *img* không cần tải lại trang và quá trình này diễn ra liên tục nên có thể có một số thời điểm website sẽ lấy ảnh đúng lúc ảnh chưa sinh ra khiến website lỗi không hiện ảnh.

- + Module phát hiện vật thể trong ảnh / thư mục chưa ghi được tệp khi đối số truyền vào là thư mục.

- + Website còn ít chức năng và Python App không được viết bởi các framework hỗ trợ lập trình GUI nên giao diện đơn giản và không có chức năng.

- + Chưa chạy module nhận diện qua camera/video tại các bãi đỗ xe thật.

- + Đây là mô hình khá nặng nên yêu cầu máy chủ có cấu hình cao và tích hợp GPU. Đồng thời mã nguồn có thể còn chưa tối ưu và trong lớp upsampling sử dụng kỹ thuật Nearest-Neighbor để đổi lại tốc độ xử lý nên độ chính xác sẽ không cao bằng Bilinear.

+ Cần thu thập khá nhiều dữ liệu ảnh về đối tượng cần nhận diện và xử lý chúng tốn khá nhiều thời gian sau đó huấn luyện để đạt được sự khái quát hoá cao.

Hướng phát triển

- Khắc phục những nhược điểm trên.
- Trang website có thể thêm chức năng đặt trước chỗ để xe.
- Tìm kiếm đối tác ngành Robotics kết hợp với mô hình trên để có thể hướng dẫn cho các xe tự hành có thể tự nhận diện được vị trí chưa đỗ trong bãi đỗ xe.
- Thêm khả năng nhận diện các phương tiện khác như xe máy, container,...
- Phân quyền thêm cho website bằng một trang chỉ cho admin truy cập để xem stream, lấy dữ liệu thu được lưu vào cơ sở dữ liệu.

PHỤ LỤC

Phụ lục 1: Hướng dẫn cài đặt

- Tải tệp weights tại địa chỉ:

https://drive.google.com/drive/folders/1sne3rveUK3j0QYPTrdc96MB6OT_5z0jf?usp=sharing

Sau đó đặt tệp weights tại thư mục **weights**.

- Tinh chỉnh các siêu tham số *confidence*, *nms_thesh* ở mức vừa đủ nhận diện được tất cả các đối tượng, càng tăng các thông số này thì độ chính xác càng cao. Tinh chỉnh siêu tham số *inp_dim* và các tham số *weight*, *height* trong tệp *yolov3.cfs* ở độ phân giải vừa đủ với cấu hình máy tính (thường trung bình là 416). Thay đổi tên tệp weights tại phương thức *model.load_weights()* với **yolov3.weights** là tệp weights cho bộ dữ liệu mô phỏng, **bigyolov3.weights** là tệp weights cho bộ dữ liệu lớn. Đối với module nhận diện ảnh, có thể kiểm tra bằng những hình ảnh tại thư mục **imgs**.
- Cài đặt các môi trường cần thiết trong tệp **requirements.txt**.

Phụ lục 2: Hướng dẫn sử dụng

- Đối với module nhận diện ảnh, chạy theo lệnh sau để kiểm tra:

```
python images.py --images [tên ảnh]
```

- Đối với module nhận diện qua camera, chạy bằng lệnh sau để kiểm tra:

```
python cam.py
```

- Đối với website, chạy bằng lệnh sau để kiểm tra:

```
python manage.py runserver
```

- Website sau khi triển khai, kiểm tra tại địa chỉ:

<https://parkingvision.herokuapp.com/>

(Do máy chủ web cần cấu hình cao và camera để chạy liên tục nên không thể triển khai cùng được, vì vậy đồ án này chỉ triển khai trước phần web Front End, phần demo trực tiếp sẽ sử dụng **ngrok** để tạo đường hầm (tunnel) từ localhost ra internet).

- Xem video demo chạy website và 2 module nhận diện qua camera và ảnh tại địa chỉ website: https://youtu.be/IHwrNHjOy_Y

- Link Github: <https://github.com/hiepnguyenit/DATN-ParkingYolo>

TÀI LIỆU THAM KHẢO

- [1]. Andrew Ng, *Machine Learning Yearning*, 2018.
- [2]. Alberto Fernández Villán, *Mastering OpenCV 4 with Python*, Packt, 2019.
- [3]. Joseph Redmon & Ali Farhadi, *YOLOv3: An Incremental Improvement*, University of Washington, 2018.
- [4]. Samuel Ordonia, *Detecting Cars in a Parking Lot using Deep Learning*, A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science, San Jose State University, 2019.
- [5]. Arepalli Rama Venkata Naga Sai, *Car Parking Occupancy Detection using YOLOv3*, A thesis submitted in partial fulfillment of the requirements for the degree of Master of Engineering in Microelectronics and Embedded Systems, Asian Institute of Technology, 2019.
- [6]. Lê Thị Thu Hằng, nghiên cứu về mạng neural tích chập và ứng dụng cho bài toán nhận dạng biển số xe, Luận văn thạc sĩ trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, 2016.
- [7]. Vũ Hữu Tiệp's Blog, "*Machine Learning cơ bản*". 30 03 2021. [Online]. <https://machinelearningcoban.com>, [Accessed 06 07 2021].
- [8]. How to implement a YOLO (v3) object detector from scratch in PyTorch. 25 04 2021. [Online]. <https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/>, [Accessed 06 07 2021].
- [9]. Logistic regression. 01 04 2021. <http://maitrongnghia.com/2020/04/logistic-regression/>, [Accessed 06 07 2021].
- [10]. Hướng dẫn tắt tần tật về Pytorch để làm các bài toán về AI. 20 04 2021. [Online]. <https://viblo.asia/p/huong-dan-tat-tan-tat-ve-pytorch-de-lam-cac-bai-toan-ve-ai-YWOZrNkNZQ0>, [Accessed 06 07 2021].
- [11]. Deep Learning - Cách làm việc với CUDA trong PyTorch. 23 04 2021. [Online] <https://gpubub.net/deep-learning-cach-lam-viec-voi-cuda-trong-pytorch/>, [Accessed 06 07 2021].

- [12]. Wikipedia, “*Sai số toàn phương trung bình*”. 15 04 2021. [Online].
https://vi.wikipedia.org/wiki/Sai_số_toàn_phương_trung_bình, [Accessed 06 07 2021].
- [13]. How to train YOLOv3 on the custom dataset, 20 04 2021 [Online]
<https://thebinarynotes.com/how-to-train-yolov3-custom-dataset/>, [Accessed 06 07 2021].
- [14]. Training AlekseyAB YOLOv3 on own dataset in Google Colab. 20 04 2021.
[Online]. <https://vovaprivalov.medium.com/training-alekseyab-yolov3-on-own-dataset-in-google-colab-8f3de8105d86>, [Accessed 06 07 2021].
- [15]. Django documentation. 25 05 2021. [Online].
<https://docs.djangoproject.com/en/3.2/>, [Accessed 06 07 2021].
- [16]. Tiêu chuẩn thiết kế bãi đỗ xe ô tô mới nhất 2021. 31 01 2021 [Online]
<https://bilparking.com.vn/article/kich-thuoc-tieu-chuan-bai-do-xe-oto-2020>,
[Accessed 06 07 2021].
- [17]. 5 ứng dụng của thị giác máy tính cho Deep Learning. 23 06 2021. [Online].
<https://www.thegioimaychu.vn/blog/ai-deep-learning/5-ung-dung-cua-thi-giac-may-tinh-cho-deep-learning-p3205/>, [Accessed 06 07 2021].
- [18]. Tìm cơ chế cho các điểm đỗ xe. 15 03 2021. [Online].
<https://nhandan.vn/baothoinay-dothi/tim-co-che-cho-cac-diem-do-xe-637883/>,
[Accessed 06 07 2021].