

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CHUYÊN NGÀNH

**Nghiên cứu kiến trúc RCNN và ứng dụng xây
dựng mô hình phát hiện phương tiện giao thông
tại TP.HCM**

GVHD: Trần Đình Toàn

Sinh viên thực hiện:

Mai Công Tiến – 2001202265 – 11DHTH10

Lê Việt Khánh – 2001202115 – 11DHTH9

Trần Kim Phụng – 2001207195 – 11DHTH11

TP. HỒ CHÍ MINH, tháng 12 năm 2023

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CHUYÊN NGÀNH

**Nghiên cứu kiến trúc RCNN và ứng dụng xây
dựng mô hình phát hiện phương tiện giao thông
tại TP.HCM**

GVHD: Trần Đình Toàn

Sinh viên thực hiện:

Mai Công Tiến – 2001202265 – 11DHTH10

Lê Việt Khánh – 2001202115 – 11DHTH9

Trần Kim Phụng – 2001207195 – 11DHTH11

TP. HỒ CHÍ MINH, tháng 12 năm 2023

LỜI CAM ĐOAN

Chúng tôi xin cam đoan đây là công trình nghiên cứu của chúng tôi. Các số liệu, kết quả nêu trong Đồ án là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Chúng tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Đồ án này đã được cảm ơn và các thông tin trích dẫn trong Đồ án đã được chỉ rõ nguồn gốc.

Sinh viên thực hiện Đồ án

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Trong hành trình nghiên cứu và thực hiện đồ án chuyên ngành này, chúng tôi xin bày tỏ lòng biết ơn sâu sắc đến những người đã hỗ trợ và đóng góp cho thành công của dự án này.

Trước hết, chúng tôi xin gửi lời cảm ơn chân thành đến Thầy **Trần Đình Toàn** – giảng viên hướng dẫn đồ án chuyên ngành “Nghiên cứu kiến trúc RCNN và ứng dụng xây dựng mô hình phát hiện phương tiện giao thông tại TP.HCM”, người đã tận tâm hướng dẫn, giáo dục và động viên chúng tôi trong suốt quá trình thực hiện đồ án. Những sáng kiến, ý kiến chân thành và sự tận tâm của Thầy đã giúp chúng tôi vượt qua những khó khăn, từ đó phát triển khả năng nghiên cứu và kỹ năng chuyên môn.

Chúng tôi cũng muốn bày tỏ lòng biết ơn đến thầy Toàn người đã chia sẻ kiến thức sâu rộng và kinh nghiệm quý báu, từ đó giúp chúng tôi hiểu sâu hơn về chuyên ngành và áp dụng kiến thức vào thực tế.

Không thể không nhắc đến sự hỗ trợ đặc lực của bạn bè và gia đình trong suốt thời gian nghiên cứu và thực hiện đồ án. Lời động viên và tinh thần lạc quan từ họ đã giúp chúng tôi vượt qua những thách thức và duy trì động lực trong công việc.

Chân thành cảm ơn tới tất cả những người đã đồng hành và hỗ trợ chúng tôi trong hành trình này. Đây là một trải nghiệm quý báu và chúng tôi cam kết sẽ tiếp tục xây dựng và phát triển từ những kiến thức và kỹ năng đã đạt được.

Trân trọng.

TPHCM, tháng 12 năm 2023

Nhóm thực hiện

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:

Tp. HCM, ngày tháng 12 năm 2023

GVHD: Trần Đình Toàn

TÓM TẮT

Nghiên cứu này tập trung vào áp dụng kiến trúc R-CNN (Region-based Convolutional Neural Network) trong việc phát hiện phương tiện giao thông tại Thành phố Hồ Chí Minh. R-CNN là một phương pháp tiên tiến trong lĩnh vực thị giác máy tính, kết hợp sức mạnh của mạng nơ-ron tích chập với khái niệm vùng quan tâm (region of interest) để nâng cao hiệu suất của hệ thống.

Qua quá trình nghiên cứu, chúng tôi đã xây dựng một mô hình R-CNN tùy chỉnh có khả năng nhận diện chính xác và hiệu quả các loại phương tiện giao thông phổ biến tại TP.Hồ Chí Minh. Mô hình được đào tạo trên một lượng lớn dữ liệu hình ảnh thu thập từ các nguồn đa dạng như camera giám sát, hình ảnh đường phố và nguồn dữ liệu công cộng.

Kết quả thực nghiệm cho thấy mô hình của chúng tôi có khả năng phát hiện và phân loại phương tiện giao thông với độ chính xác cao, đặc biệt là trong môi trường đô thị đặc trưng của TP.Hồ Chí Minh. Ứng dụng của mô hình này có thể giúp cải thiện quản lý và giám sát giao thông, đồng thời đóng góp vào việc xây dựng thành phố thông minh và an toàn hơn.

Nghiên cứu này không chỉ mở ra cánh cửa cho việc phát triển các ứng dụng thực tế trong lĩnh vực giao thông và an ninh mà còn đóng góp vào sự phát triển của công nghệ thị giác máy tính trong bối cảnh các thành phố đang ngày càng chú trọng đến việc quản lý và tối ưu hóa giao thông đô thị.

MỤC LỤC

MỞ ĐẦU.....	1
CHƯƠNG 1: TỔNG QUAN	2
1.1 Giới thiệu	2
1.2 MỤC TIÊU VÀ PHẠM VI ĐỀ TÀI.....	3
1.2.1 Mục Tiêu của đề tài	3
1.2.2. Công cụ sử dụng	5
1.3 Kết chương.....	5
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	6
2.1 Giới thiệu về học sâu.....	6
2.1.1 Học sâu	6
2.1.2 Mạng nơ-ron tích chập – Convolutional neural network (CNN)	8
2.2 Các kỹ thuật liên quan đến xử lý ảnh và nhận diện đối tượng.....	8
2.2.1 Tổng quan về xử lý ảnh.....	8
2.2.2. Các vấn đề cơ bản trong xử lý ảnh	9
2.2.3 Nhận diện và phân loại ảnh.....	9
2.3. Các kỹ thuật hiện tại và hạn chế.....	10
2.3.1. R-CNN, Fast R-CNN.....	10
a. R-CNN.....	10
b. Fast R-CNN.....	10
2.3.2. Faster R-CNN	11
2.3.3. YOLO, SSD	11
a. You Only Look Once.....	11
b. Single Shot Detector (SSD)	12

2.4. Kết luận chương	13
CHƯƠNG 3:PHƯƠNG PHÁP SỬ DỤNG	14
3.1 Bài toán phát hiện phương tiện giao thông	14
<i>3.1.1 Tổng quan bài toán</i>	<i>14</i>
3.2 Hệ thống phát hiện đối tượng thời gian thực RCNN.....	15
<i>3.2.1 Tổng quan hệ thống phát hiện đối tượng RCNN.</i>	<i>15</i>
<i>3.2.2 Các thành phần của hệ thống RCNN.....</i>	<i>15</i>
<i>3.2.3 Kiến trúc của mô hình RCNN qua các phiên bản.</i>	<i>16</i>
a. RCNN (Region-based Convolutional Neural Network):	16
b. Fast R-CNN:	17
c. Faster R-CNN:	17
d. Mask R-CNN:	18
3.3. Mô hình được sử dụng để giải quyết bài toán	19
<i>3.3.1 Faster R-CNN</i>	<i>19</i>
<i>3.3.2 ResNet-50 backbone</i>	<i>20</i>
3.4 Thư viện và module được sử dụng để giải quyết bài toán	21
<i>3.4.1 Thư viện Pytorch.....</i>	<i>21</i>
3.4.1.1 Utils.py.....	21
3.4.1.2 Transforms.py	22
3.4.1.3 Coco_eval.py	23
3.4.1.4 Engine.py.....	23
3.4.1.5 Coco_utils.py	24
3.4.1.6 Torchvision.....	25
<i>3.4.2 Thư viện Scikit-learn.....</i>	<i>25</i>

3.4.2.1 <i>Sklearn.preprocessing</i>	25
3.4.3 Công cụ gán nhãn	26
3.4.3.1 <i>LabelIMG</i>	26
3.5 Phương pháp huấn luyện	28
3.6 Kết luận chương	29
CHƯƠNG 4: XÂY DỰNG MÔ HÌNH	30
4.1 Thu thập và tiền xử lý dữ liệu.	30
4.2 Cài đặt	30
4.2.1 Môi trường thử nghiệm	30
4.2.2 Đầu vào, đầu ra của hệ thống	30
4.2.3 Huấn luyện mô hình	33
4.2.3.1 Định nghĩa một lớp <i>VOCDataset</i>	33
4.2.3.2 Biến đổi hình ảnh	35
4.2.3.3 Tạo các <i>DataLoader</i>	37
4.2.3.3 Kiểm tra xem có GPU (đồng thời có hỗ trợ CUDA)	38
4.2.3.4 Kiểm tra	39
4.2.3.5 Mô hình đã được đào tạo trước về COCO.....	41
4.2.3.6 Classifier.....	43
4.5 Đề xuất phương pháp cải tiến và đánh giá.....	55
4.6 Kết luận chương.	60
KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN.....	62

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt

DANH MỤC CÁC HÌNH VẼ VÀ ĐỒ THỊ

Hình 1 Nơ-ron.....	7
Hình 2 Mạng nơ-ron nhân tạo.....	7
Hình 3 Các bước cơ bản trong một hệ thống xử lý ảnh	8
Hình 4 Mô hình mạng Fast R-CNN.....	11
Hình 5 Mô hình mạng Faster R-CNN.....	11
Hình 6 Quá trình dự đoán của YOLO	12
Hình 7 Mô hình mạng SSD.....	12
Hình 8 Bài toán phát hiện phương tiện giao thông	14
Hình 9 Phân loại region proposal.....	16
Hình 10 RCNN	17
Hình 11 Fast R-CNN.....	17
Hình 12 Faster R-CNN	18
Hình 13 Mask R-CNN	19
Hình 14 ResNet-50.....	20
Hình 15 Bộ dữ liệu đầu vào (jpg).....	31
Hình 16 Bộ dữ liệu đầu vào (xml)	31
Hình 17 Dữ liệu đầu ra.....	33
Hình 18 Định nghĩa một lớp VOCDataset.....	34
Hình 19 Biến đổi hình ảnh	36
Hình 20 Tạo các DataLoader	37
Hình 21 Kiểm tra xem có GPU	39
Hình 22 Kiểm tra.....	40
Hình 23 Mô hình sau khi gán nhãn	41
Hình 24 Tải mô hình được đào tạo trước về COCO	43
Hình 25 Classifier	44
Hình 26 Cài đặt COCOAPI và cài đặt module từ thư viện Pytorch.....	45
Hình 27 Tiến hành huấn luyện.....	46
Hình 28 Giao diện chính	48
Hình 29 Gán nhãn trong folder	49
Hình 30 Gán nhãn Image	51
Hình 31 Gán nhãn một vùng trong hình.....	53

MỞ ĐẦU

Trong bối cảnh thị trường công nghiệp và công nghệ phát triển với tốc độ chóng mặt, việc ứng dụng các tiến bộ trong lĩnh vực trí tuệ nhân tạo (AI) và xử lý ảnh đã mở ra nhiều cơ hội mới trong việc nghiên cứu và phát triển các hệ thống nhận diện và giám sát. Trong lĩnh vực an ninh giao thông và quản lý phương tiện, một trong những phương pháp tiên tiến được đặc biệt chú ý là mô hình R-CNN (Region-based Convolutional Neural Network).

Đặc biệt, đối với một thành phố đông đúc như Thành phố Hồ Chí Minh, vấn đề quản lý và giám sát giao thông không chỉ là một thách thức mà còn là một yếu tố quyết định đến chất lượng cuộc sống của cộng đồng và sự phát triển bền vững. Trong ngữ cảnh này, nghiên cứu về kiến trúc RCNN và ứng dụng xây dựng mô hình phát hiện phương tiện giao thông trở nên hết sức quan trọng để cung cấp giải pháp hiệu quả trong việc giảm thiểu tai nạn giao thông, tối ưu hóa luồng thông xe, và nâng cao hiệu suất của hệ thống giao thông đô thị.

Như một phản ánh của sự tiến bộ trong ngành công nghiệp AI, nghiên cứu này không chỉ hứa hẹn mang lại những đóng góp to lớn cho lĩnh vực an ninh giao thông mà còn mở ra những triển vọng tích cực trong việc xây dựng những ứng dụng thực tế và hiệu quả cho quản lý phương tiện tại TP.HCM. Qua đó, nó sẽ giúp chúng ta hiểu rõ hơn về khả năng ứng dụng của RCNN trong môi trường đô thị đông đúc và đặt ra những cơ hội phát triển mới cho việc cải thiện hệ thống quản lý giao thông, từ đó, tạo ra một môi trường số an toàn và hiệu quả cho cộng đồng đô thị ngày càng đông đúc.

Trong quá trình thực hiện dự án này, nhóm chúng em đã nhận được sự hướng dẫn và hỗ trợ đáng kể từ thầy Trần Đình Toàn và nhóm chúng em xin bày tỏ sự biết ơn sâu sắc đối với sự hỗ trợ đó.

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu

Trong thời kỳ gần đây, ngành dịch vụ tại Việt Nam đang trở nên rực rỡ, đặc biệt là lĩnh vực giải trí du lịch. Mục tiêu của chúng ta không chỉ là nâng cao giá trị gia tăng của sản phẩm, mà còn là tăng cường chất lượng và giá trị trong các ngành sản xuất. Dịch vụ không chỉ đáp ứng mọi nhu cầu của con người mà còn nâng cao dân trí, làm cho cuộc sống của chúng ta trở nên văn minh hơn, đồng thời kích thích sức lao động và tăng cường hiệu quả công việc.

Tuy nhiên, trong khi ngành du lịch tại Việt Nam đang trên đà phát triển, cũng đối mặt với những thách thức đáng kể. Nạn "chặt chém" du khách, bạo lực và hạ tầng yếu kém, cùng với chất lượng dịch vụ không đồng đều và quản lý kém chính là những vấn đề cấp bách cần giải quyết. Điều này đặt ra một câu hỏi quan trọng: Làm thế nào để du khách có thể khám phá những địa điểm mà không lo lắng về những vấn đề này? Làm thế nào để họ có thể tận hưởng trọn vẹn niềm vui của việc du lịch và giải trí, mà không bị gián đoạn bởi những rắc rối không mong muốn?

Từ những nỗi lo lắng và câu hỏi này, chúng em đã quyết định thực hiện đề tài nghiên cứu "Xây dựng ứng dụng quản lý tour du lịch". Đề tài này không chỉ đơn thuần là một dự án công nghệ, mà còn là một nỗ lực hướng tới sự hoàn hảo trong việc kết nối giữa du khách và các trải nghiệm du lịch tuyệt vời. Chúng em tin rằng việc phát triển hệ thống này sẽ không chỉ giúp du khách tự tin trong việc lựa chọn điểm đến và giải trí, mà còn hỗ trợ các công ty du lịch lữ hành mang đến những trải nghiệm không gì sánh kịp. Đây không chỉ là một dự án, mà là một lời cam kết của nhóm em đối với sự phát triển bền vững của ngành du lịch Việt Nam và sự hạnh phúc của những người đam mê khám phá.

1.2 MỤC TIÊU VÀ PHẠM VI ĐỀ TÀI

1.2.1 Mục Tiêu của đề tài

Mục tiêu chính của dự án "Nhận diện phương tiện giao thông bằng mô hình RCNN" là tạo ra một ứng dụng quản lý và cung cấp thông tin về phương tiện giao thông. Cụ thể, dự án có các mục tiêu sau:

- **Nhận diện và Phân loại Chính xác:** Phát triển một hệ thống nhận diện chính xác, có khả năng phân loại đúng loại phương tiện giao thông như ô tô, xe máy, xe buýt, và các phương tiện khác. Mục tiêu là tạo ra một bộ lọc thông tin chất lượng để cung cấp dữ liệu cơ sở cho các ứng dụng và quyết định quản lý giao thông.
- **Chịu được Đa Dạng Hình Ảnh:** Hệ thống cần có khả năng xử lý và nhận diện phương tiện trên nhiều dạng hình ảnh, từ các điểm quan sát khác nhau, ban ngày và ban đêm, đảm bảo sự linh hoạt và độ chính xác trong mọi điều kiện.
- **Thời Gian Thực:** Mục tiêu là xây dựng hệ thống có khả năng nhận diện phương tiện giao thông trong thời gian thực, giúp cung cấp thông tin liên tục và kịp thời cho quản lý giao thông và người dùng cá nhân.
- **Chức năng Theo dõi Tích hợp:** Tích hợp chức năng theo dõi để giúp quản lý và người dùng theo dõi sự chuyển động của các phương tiện giao thông theo thời gian, tạo ra một bức tranh đầy đủ về tình trạng giao thông.
- **Bảo mật và Riêng tư:** Đảm bảo tính bảo mật và tuân thủ các quy định về quyền riêng tư trong quá trình nhận diện phương tiện giao thông, giữ cho thông tin cá nhân và dữ liệu không bị lộ ra ngoài trừ những trường hợp được phép

1.1.1 Phạm vi của đề tài

Phạm vi của đề tài: Phạm vi của dự án tập trung vào việc nhận diện các phương tiện giao thông, đặc biệt là xác định vị trí, loại hình, và theo dõi chúng trong thời gian thực. Đề tài này hướng đến việc cải thiện độ chính xác và hiệu suất của hệ thống trong môi trường đô thị, đóng góp vào giải quyết thách thức giao thông tại Thành phố Hồ Chí Minh.

- **Nhận diện Phương tiện Giao thông:** Đề tài tập trung chủ yếu vào khả năng nhận diện phương tiện giao thông trong hình ảnh hoặc video. Điều này bao gồm việc xác định vị trí, hình dạng, và loại phương tiện như ô tô, xe máy, xe buýt, và các loại khác.
- **Mô hình RCNN (Region-based Convolutional Neural Network):** Sự nghiên cứu tập trung vào triển khai và tối ưu hóa mô hình RCNN để đạt được hiệu suất nhận diện cao và đáng tin cậy. Phạm vi này giả định việc sử dụng RCNN làm công cụ chính để giải quyết vấn đề nhận diện phương tiện giao thông.
- **Thời Gian Thực:** Hệ thống được phát triển để hoạt động trong thời gian thực, có khả năng cung cấp thông tin liên tục về tình trạng giao thông và sự di chuyển của phương tiện.
- **Dữ liệu Hình Ảnh:** Đề tài sử dụng dữ liệu hình ảnh và video thực tế từ Thành phố Hồ Chí Minh để đào tạo và kiểm thử mô hình. Phạm vi bao gồm việc xử lý nhiều dạng hình ảnh và điều kiện ánh sáng khác nhau.
- **Khả năng Mở rộng và Tích hợp:** Đề tài có sự xem xét về khả năng mở rộng và tích hợp với các hệ thống quản lý giao thông hiện tại, nhưng phạm vi không bao gồm triển khai thực tế của hệ thống tích hợp.

1.2.2. Công cụ sử dụng

Phần mềm “Nhận diện phương tiện giao thông” sử dụng các công cụ sau:

- Thu thập dữ liệu: LabelIMG, PASCAL VOC.
- Ngôn ngữ lập trình: Python.
- Môi trường lập trình: Visual studio code, Colab.
- Hệ quản trị cơ sở dữ liệu: SQL server.

1.3 Kết chương

Trong chương này, nhóm chúng em đã trình bày tổng quan về đề tài "Nhận diện phương tiện giao thông bằng mô hình RCNN". Chúng em đã giới thiệu mục tiêu, phạm vi, cũng như lý do mà nhóm chúng em lựa chọn đề tài để thực hiện trong học phần “đồ án chuyên ngành”. Ở chương tiếp theo thì nhóm chúng em sẽ đi sâu vào việc tìm hiểu trong lĩnh vực nhận diện phương tiện giao thông. Chúng em sẽ xem xét các nghiên cứu và công trình liên quan, tập trung vào cơ sở lý thuyết của mô hình RCNN và ứng dụng của nó trong ngữ cảnh nhận diện phương tiện. Qua đó, chúng em sẽ trình bày chi tiết về quy trình triển khai mô hình, cũng như các thách thức và giải pháp có thể phát sinh trong quá trình nghiên cứu và phát triển. Chương này nhằm mang lại cái nhìn tổng quan sâu rộng và cung cấp cơ sở lý luận cho việc thực hiện đề tài của chúng em.

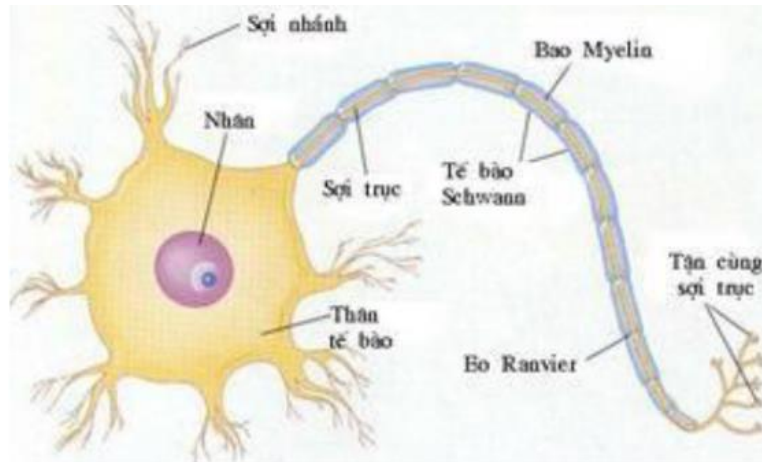
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Giới thiệu về học sâu

2.1.1 Học sâu

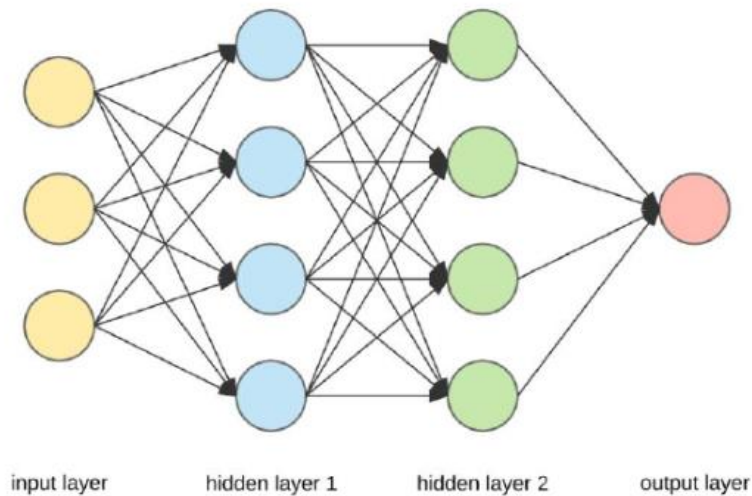
Học sâu (deep learning) là một nhánh của ngành máy học, dựa trên một tập hợp các thuật toán để cố gắng mô hình dữ liệu để trừu tượng hóa □ mức cao bằng cách sử dụng nhiều lớp xử lý với cấu trúc phức tạp, hoặc bằng cách khác bao gồm nhiều lớp biến đổi phi tuyến để trích xuất đặc trưng và chuyển đổi[23]. Mỗi lớp kế tiếp dùng đầu ra của lớp trước làm đầu vào. Các thuật toán này có thể được giám sát hoặc không cần giám sát và các ứng dụng bao gồm các mô hình phân tích (không giám sát) và phân loại (giám sát).

Một trong những phương pháp học sâu thành công nhất là mô hình mạng nơ-ron nhân tạo (Artificial Neural Network)[23]. Mạng nơ-ron nhân tạo được lấy cảm hứng từ các mô hình sinh học năm 1959 được đề xuất bởi người đoạt giải Nobel David H. Hubel & Torsten Wiesel, 2 người đã tìm thấy hai loại tế bào trong vỏ não thị giác chính: các tế bào đơn giản và các tế bào phức tạp. Nhiều mạng nơ-ron nhân tạo có thể được xem như là các mô hình ghép tầng của các tế bào loại lấy cảm hứng từ những quan sát sinh học.



Hình 1 Nơ-ron

Mạng nơ-ron nhân tạo là sự kết hợp của các tầng perceptron hay còn gọi là perceptron đa tầng (multilayer perceptron) như hình bên dưới



Hình 2 Mạng nơ-ron nhân tạo

Kiến trúc chung của mạng nơ-ron nhân tạo bao gồm thành phần: Lớp đầu vào, Lớp ẩn và Lớp đầu ra.

- Lớp đầu vào (Input layer): Là nơi để nạp dữ liệu vào. Mỗi nơ-ron tương ứng với một thuộc tính (attribute) hoặc đặc trưng (feature) của dữ liệu đầu vào.

- Lớp ẩn (Hidden layer): Là nơi xử lý dữ liệu trước khi đưa ra output. Thường là hàm tổng (Summation function) để đưa ra giá trị của một nơ-ron tại hidden layer. Có thể có nhiều hơn một hidden layer. Khi đó đầu ra của một hidden layer sẽ là đầu vào cho hidden layer tiếp theo.

- Lớp đầu ra (Output layer): Là giá trị đầu ra của mạng nơ-ron. Sau khi đi qua hidden layer cuối cùng, dữ liệu sẽ được chuyển hóa bằng một hàm số được gọi là hàm kích hoạt (Activation function) và đưa ra output cuối cùng. Hàm chuyển đổi thường là hàm $\tanh(x)$, $\text{sigmoid}(x)$ hoặc $\text{softmax}(x)$.

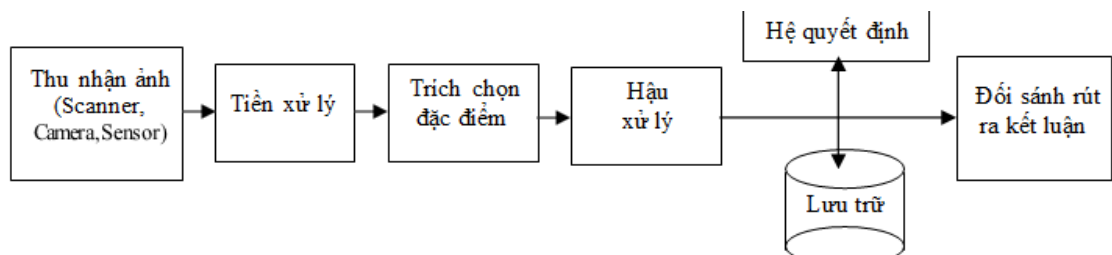
2.1.2 Mạng nơ-ron tích chập – Convolutional neural network (CNN)

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 như hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động. CNN được sử dụng nhiều trong các bài toán phát hiện các object trong ảnh.

2.2 Các kỹ thuật liên quan đến xử lý ảnh và nhận diện đối tượng.

2.2.1 Tổng quan về xử lý ảnh

Quá trình xử lý ảnh được xem như là quá trình thao tác ảnh đầu vào nhằm tạo ra kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh tốt hơn hoặc một kết luận nào đó.



Hình 3 Các bước cơ bản trong một hệ thống xử lý ảnh

Sơ đồ tổng quát của một hệ thống xử lý ảnh:

- Khối thu nhận ảnh: Có nhiệm vụ tiếp nhận ảnh đầu vào.

- Khối tiền xử lý: Có nhiệm vụ xử lý nâng cao chất lượng ảnh như giảm nhiễu, phân vùng, tìm biên ...
- Khối dự đoán: Có nhiệm vụ đưa ra dự đoán từ bức ảnh đã được tiền xử lý đưa ra kết quả dự đoán sử dụng cho bước hậu xử lý và đưa ra quyết định.
- Khối hậu xử lý: Có nhiệm vụ xử lý các kết quả của khối dự đoán, có thể lược bỏ hoặc biến đổi các kết quả này để phù hợp với các kỹ thuật cụ thể sử dụng trong hệ quyết định.
- Khối hệ quyết định và lưu trữ: Có nhiệm vụ đưa ra quyết định (phân loại, phát hiện) dựa trên dữ liệu đã học.
- Khối kết luận: Đưa ra kết luận dựa vào quyết định của khối quyết định.

2.2.2. Các vấn đề cơ bản trong xử lý ảnh

- Nắn chỉnh biến dạng:
- Khử nhiễu:
- Chỉnh số mức xám
- Nén ảnh

2.2.3 Nhận diện và phân loại ảnh

Các hệ thống phát hiện và nhận dạng đối tượng hiện nay thường có ba bước xử lý:

- Bước 1: Sử dụng một mô hình hoặc một thuật toán để tạo ra các vùng ứng viên, các khu vực quan tâm. Các vùng ứng viên này là một tập hợp lớn các hộp giới hạn để xác định đối tượng.
- Bước 2: Trích xuất đặc trưng từ các hộp giới hạn tìm được, chúng được đánh giá và xác định xem đối tượng nào có mặt trong hộp giới hạn đó (tức là thành phần phân loại đối tượng) hay không.

- Bước 3: Trong bước hậu xử lý cuối cùng, các hộp chồng chéo nhau được kết hợp thành một hộp giới hạn sử dụng một số thuật toán như Non-maximum suppression, ...

2.3. Các kỹ thuật hiện tại và hạn chế.

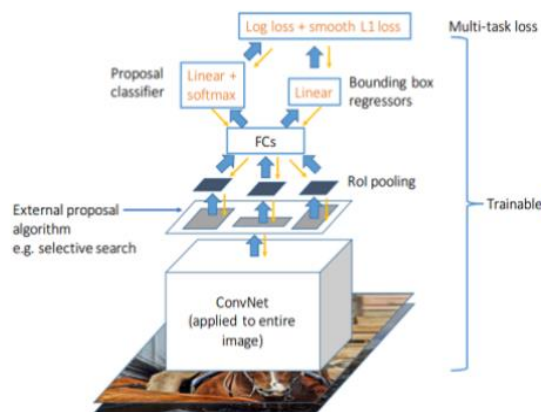
2.3.1. R-CNN, Fast R-CNN.

a. R-CNN

Sau khi các nghiên cứu về các kỹ thuật học sâu thu được nhiều kết quả rõ ràng các kỹ thuật phân loại dựa trên HOG[2] dần được thay thế bằng một kỹ thuật học sâu như CNN đã cho kết quả chính xác hơn. Tuy nhiên, có một vấn đề là CNN quá chậm và tính toán rất tốn kém. Không thể chạy CNN trên nhiều cửa sổ được tạo bởi thuật toán cửa sổ trượt (sliding window detector). R-CNN[12] đã giải quyết vấn đề này bằng cách chạy một thuật toán được gọi là Selective Search để giảm số hộp giới hạn (bounding box) được đưa vào bộ phân loại. Selective Search sử dụng các dấu hiệu bố cục như kết cấu, cường độ, màu sắc ... để tạo ra các vị trí có thể chứa đối tượng. Sau đó chúng ta có thể cung cấp các hộp giới hạn này cho một bộ phân loại dựa trên CNN.

b. Fast R-CNN

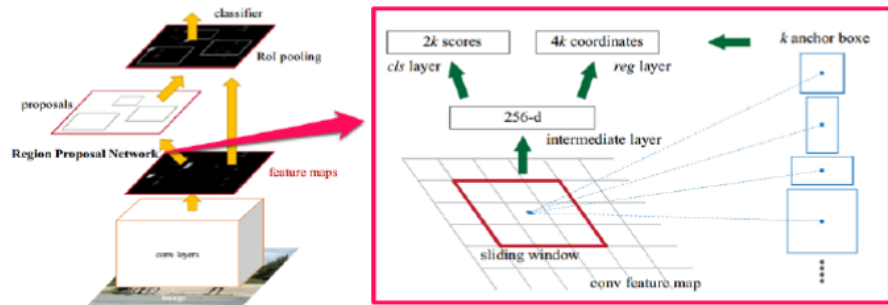
Fast R-CNN[5] sử dụng ý tưởng của SPP-net, R-CNN và sửa một vấn đề chính trong SPP-net như: Fast R-CNN có thể thực hiện từ đầu đến cuối (end-to-end). Một điều nữa là họ đã thêm tính toán hồi quy để tìm hộp giới hạn vào việc huấn luyện. Vì vậy mạng có hai đầu ra một đầu phân loại đầu ra và một đầu dự đoán hộp giới hạn. Mục tiêu này là tính năng nổi bật của Fast R-CNN[5] vì nó không còn yêu cầu huấn luyện mạng độc lập để tìm vị trí và phân loại đối tượng. Hai thay đổi này đã làm giảm bớt thời gian huấn luyện tổng thể và tăng độ chính xác so với SPP-net.



Hình 4 Mô hình mạng Fast R-CNN

2.3.2. Faster R-CNN

Faster R-CNN là một phiên bản cải tiến của Fast R-CNN. Nó thay thế phần chậm nhất của Fast R-CNN là Selective search bằng một mạng CNN nhỏ được gọi là mạng đề xuất khu vực (Region Proposal network-RPN) để đề xuất ra các vùng quan tâm trên ảnh.

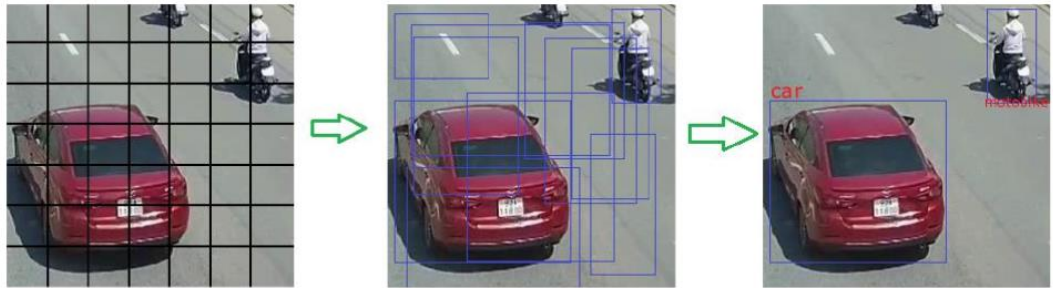


Hình 5 Mô hình mạng Faster R-CNN

2.3.3. YOLO, SSD

a. You Only Look Once

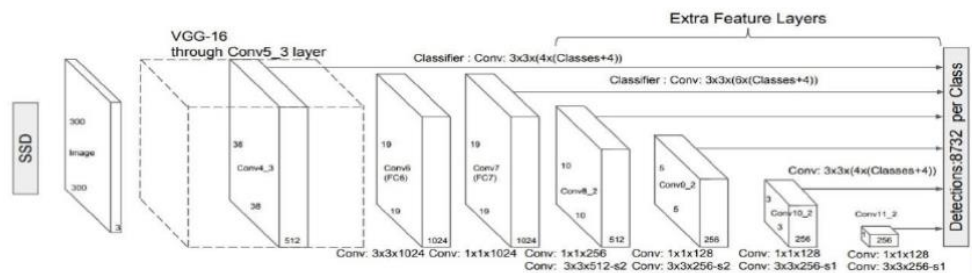
YOLO chia mỗi hình ảnh đầu vào thành một lưới có $S \times S$ ô và mỗi ô dự đoán N hộp giới hạn và độ tin cậy. Độ tin cậy phản ánh độ chính xác của hộp giới hạn và liệu hộp giới hạn đó có thực sự chứa một đối tượng (không phân biệt lớp). YOLO cũng dự đoán xác suất các lớp cho mỗi ô. Chúng ta có thể kết hợp cả hai xác suất để có được xác suất của mỗi lớp cho từng ô trong lưới được dự đoán.



Hình 6 Quá trình dự đoán của YOLO

Một số lợi ích của YOLO: - Nhanh. Tốt cho các hệ thống xử lý thời gian thực - Dự đoán cả vị trí và loại đối tượng trong một mạng duy nhất nên có thể huấn luyện end-to-end để tối ưu độ chính xác - YOLO tổng quát hóa tốt hơn. Nó hoạt động tốt hơn các phương pháp khác khi tổng quát hóa từ hình ảnh tự nhiên sang các miền khác như các tác phẩm nghệ thuật.

b. Single Shot Detector (SSD)



Hình 7 Mô hình mạng SSD

SSD đạt được sự cân bằng tốt giữa tốc độ và độ chính xác. SSD chỉ chạy một mạng CNN trên hình ảnh đầu vào một lần và tính toán một bản đồ các đặc trưng. Bây giờ chúng ta chạy một filter có kích thước nhỏ 3x3 trên toàn bộ bản đồ đặc trưng này để dự đoán các hộp giới hạn và xác suất phân loại. SSD cũng sử dụng các hộp neo □ các tỷ lệ khung hình khác nhau tương tự như Faster R-CNN. Để xử lý tỷ lệ, SSD dự đoán các hộp giới hạn sau nhiều lần co giãn.

2.4. Kết luận chương.

Qua phần giới thiệu và đánh giá ở trên chúng ta có rất nhiều phương pháp để phát hiện phương tiện giao thông. Nhưng do tốc độ di chuyển của các phương tiện giao thông thường cao nên cần một phương pháp có tốc độ xử lý nhanh và độ chính xác tốt trong điều kiện ánh sáng tốt.

CHƯƠNG 3: PHƯƠNG PHÁP SỬ DỤNG

3.1 Bài toán phát hiện phương tiện giao thông

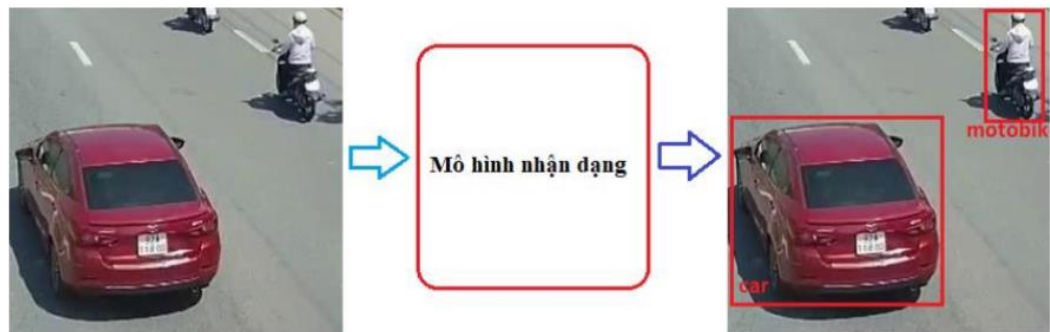
3.1.1 Tổng quan bài toán

Phát hiện phương tiện giao thông là một bài toán khó và phức tạp khi cần xác định vị trí và phân loại các phương tiện giao thông. Một hệ thống phát hiện phương tiện giao thông bao gồm ba bước xử lý chính:

Bước 1: Thu nhận ảnh từ các hệ thống camera giao thông và thực hiện tiền xử lý dữ liệu đầu vào.

Bước 2: Sử dụng một mô hình phát hiện đã được huấn luyện để phát hiện (YOLO, R-CNN, SSD...) và trả về kết quả bao gồm hộp giới hạn và đối tượng xuất hiện trong các hộp giới hạn đó.

Bước 3: Thực hiện hậu xử lý để loại bỏ các hộp chồng chéo, theo dõi các phương tiện...



Hình 8 Bài toán phát hiện phương tiện giao thông

3.1.2 Các điều kiện ràng buộc

Để thu được kết quả phát hiện chính xác cao các hệ thống camera cần được đặt tại các vị trí phù hợp như giữa các làn xe để có góc nhìn rộng tránh các trường hợp các phương tiện bị che khuất, các vị trí có ánh sáng tốt. Các hệ thống xử lý trung tâm cần được hỗ trợ các máy tính xử lý có nhiều CPU hoặc GPU giúp cho việc phát hiện các phương tiện nhanh và chính xác đảm bảo thời gian thực.

3.2 Hệ thống phát hiện đối tượng thời gian thực RCNN.

3.2.1 Tổng quan hệ thống phát hiện đối tượng RCNN.

RCNN (Region-based Convolutional Neural Network) là một phương pháp phát hiện đối tượng có ảnh hưởng lớn trong lĩnh vực thị giác máy tính. Trong khi YOLO tập trung vào tốc độ thực thi, RCNN chú trọng vào độ chính xác cao.

RCNN sử dụng một giải thuật tiên đề để tạo ra các ô đề xuất, sau đó trích xuất đặc trưng và thực hiện phân loại đối tượng cùng với việc hồi quy vị trí. So với YOLO, RCNN có khả năng nhận diện đối tượng với độ chính xác cao hơn, nhưng đồng thời cũng đòi hỏi nhiều tài nguyên tính toán hơn.

Trên GPU Titan X, RCNN có thể đạt được độ chính xác cao hơn so với YOLO, nhưng tốc độ thực thi thấp hơn, thường dưới 10 FPS trên dữ liệu COCO. RCNN thường được ưa chuộng trong các ứng dụng đòi hỏi độ chính xác cao như nhận diện đối tượng trong hình ảnh y tế, an ninh, hoặc trong các ứng dụng yêu cầu định vị chính xác của đối tượng.

3.2.2 Các thành phần của hệ thống RCNN

Region Proposal Network (RPN): RPN được sử dụng để tạo ra các ô đề xuất (region proposals) từ hình ảnh đầu vào. Đây là bước quan trọng để giảm số lượng vùng cần xử lý, tăng hiệu suất của hệ thống.

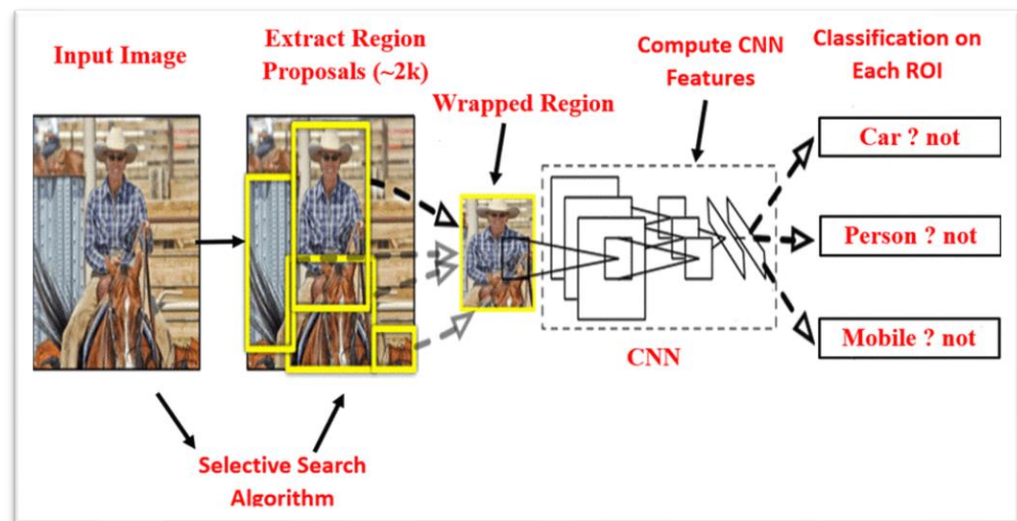
Trích xuất Đặc trưng Bằng Mạng Nơ-ron Convolutional (CNN): Các ô đề xuất được truyền qua một mạng CNN để trích xuất các đặc trưng từ hình ảnh. Các mô hình khác nhau của RCNN có thể sử dụng các mạng CNN khác nhau, như VGG16, ResNet, hoặc các kiến trúc mới nhất.

Các Lớp Suy luận (Inference Layers): Các lớp này thực hiện nhiệm vụ phân loại và hồi quy vị trí của đối tượng. Nói cách khác, sau khi trích xuất đặc trưng, mỗi ô đề xuất được đưa qua các lớp này để xác định xem đối tượng thuộc loại nào và vị trí chính xác của nó trên hình ảnh.

Non-Maximum Suppression (NMS): Sau khi các đối tượng được phân loại và vị trí được xác định, thuật toán NMS được sử dụng để loại bỏ các ô đề xuất trùng lặp và giữ lại những đối tượng duy nhất và chính xác.

Fine-tuning Layers: Các lớp này thực hiện việc fine-tuning mô hình trên dữ liệu đa dạng để cải thiện khả năng tổng quát hóa và độ chính xác của hệ thống.

Anchor Boxes (optional): Một số phiên bản của RCNN có thể sử dụng anchor boxes để cải thiện quá trình dự đoán vị trí của đối tượng, giúp mô hình xử lý hiệu quả hơn trong các tình huống đa dạng.



Hình 9 Phân loại region proposal

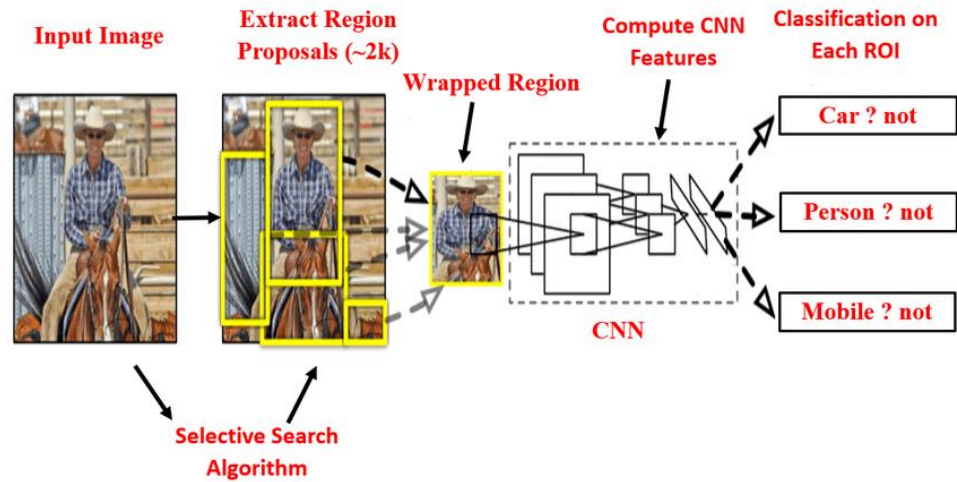
3.2.3 Kiến trúc của mô hình RCNN qua các phiên bản.

a. RCNN (Region-based Convolutional Neural Network):

Quy trình hoạt động:

- Tạo các region proposal thông qua phương pháp "selective search".
- Mỗi region proposal được đưa qua một mạng CNN để trích xuất đặc trưng.
- Các đặc trưng được đưa qua các lớp phân loại và hồi quy để xác định lớp của đối tượng và vị trí chính xác.
- Ưu điểm và Nhược điểm:

- Độ chính xác cao nhưng chậm vì quá trình tạo region proposal phức tạp.



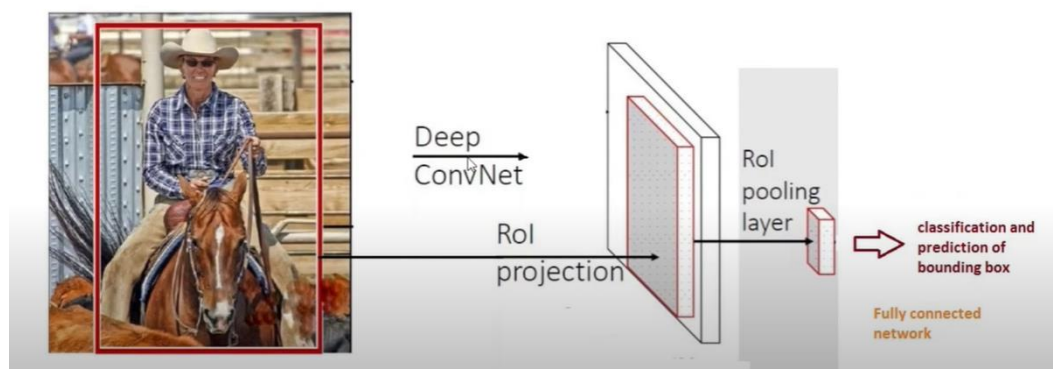
Hình 10 RCNN

b. Fast R-CNN:

Cải tiến:

- Tích hợp quá trình phân loại và regression vào một mô hình duy nhất.
- Sử dụng Region of Interest (RoI) pooling để tái cấu trúc kích thước đầu vào và giảm độ phức tạp.

Fast R-CNN

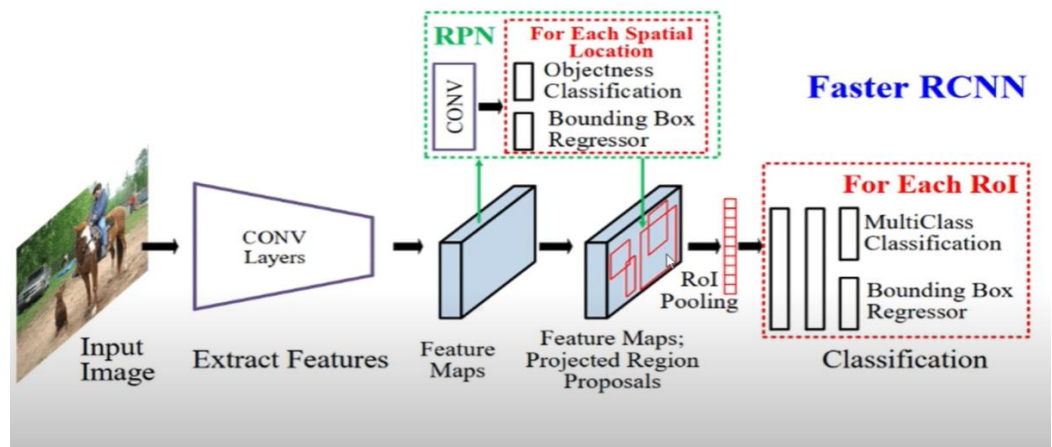


Hình 11 Fast R-CNN

c. Faster R-CNN:

Cải tiến:

- Sử dụng Region Proposal Network (RPN) để tạo ra các region proposal thay vì selective search.
- Chia sẻ đặc trưng giữa RPN và quá trình phân loại, giúp tăng tốc độ và giảm tải nguyên tính toán.

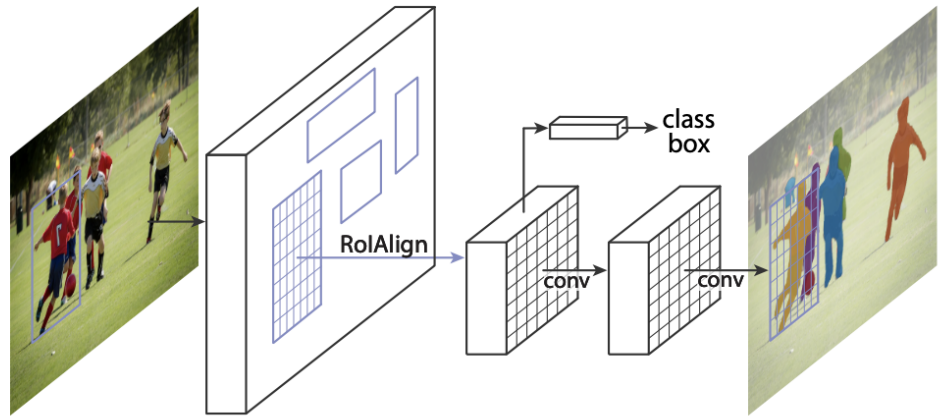


Hình 12 Faster R-CNN

d. Mask R-CNN:

Cải tiến:

- Bổ sung thêm một nhánh dự đoán mask của đối tượng.
- Sử dụng mạng ROI Align thay vì RoI pooling để cải thiện chất lượng mask.
- Cho phép dự đoán cả mask, phân loại và vị trí chính xác của đối tượng.
- Các biến thể khác nhau của RCNN giúp cải thiện tốc độ và độ chính xác của mô hình đồng thời mở rộng chức năng của nó, như việc dự đoán mask trong trường hợp của Mask R-CNN. Sự tiến triển này thể hiện sự nỗ lực liên tục để làm cho các mô hình nhận diện đối tượng trở nên hiệu quả và linh hoạt hơn trong nhiều ứng dụng.



Hình 13 Mask R-CNN

3.3. Mô hình được sử dụng để giải quyết bài toán

3.3.1 Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Network) là một mô hình sử dụng trong lĩnh vực nhận diện đối tượng trong hình ảnh. Đây là một trong những kiến trúc nổi tiếng thuộc họ mô hình R-CNN.

Dưới đây là một số điểm chính về Faster R-CNN:

- Region Proposal Network (RPN): Một phần quan trọng của Faster R-CNN là RPN, nó giúp đề xuất các vùng quan trọng có thể chứa đối tượng. RPN sử dụng các cửa sổ trượt để đề xuất các hình chữ nhật ở các vị trí khác nhau trong hình ảnh và dự đoán xác suất vùng đó chứa đối tượng hay không.

- Feature Extraction (Backbone): Faster R-CNN thường sử dụng một mạng nơ-ron sâu (deep neural network) làm phần "backbone" để trích xuất đặc trưng từ hình ảnh. Trong trường hợp của bạn, ResNet-50 là backbone.

- ROI Pooling (Region of Interest Pooling): Sau khi có được các đề xuất vùng từ RPN, Faster R-CNN sử dụng ROI Pooling để chuyển đổi các vùng không đề xuất thành các vùng cố định về kích thước, sao cho chúng có thể được đưa vào một mạng nơ-ron để thực hiện phân loại và dự đoán hộp giới hạn của đối tượng.

- Multi-task Loss Function: Faster R-CNN thực hiện đồng thời hai nhiệm vụ chính: dự đoán xác suất đối tượng thuộc các lớp khác nhau (phân loại) và dự đoán hộp giới hạn của đối tượng đó (hộp giới hạn).

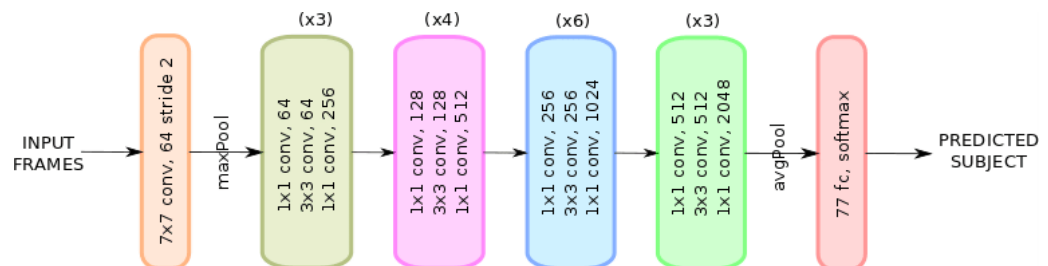
- Anchor Boxes: Một khái niệm quan trọng trong Faster R-CNN là anchor boxes, chúng định rõ các vùng tiềm năng chứa đối tượng và giúp RPN đề xuất các vùng quan trọng.

3.3.2 ResNet-50 backbone

Trong ngữ cảnh của mô hình máy học, "backbone" thường đề cập đến phần cơ bản của mô hình, được sử dụng để trích xuất đặc trưng từ dữ liệu đầu vào. Đối với các mô hình nhận diện đối tượng như Faster R-CNN, ResNet-50 thường được sử dụng làm backbone.

Ở đây, ResNet-50 là một kiến trúc mạng nơ-ron sâu (deep neural network) được thiết kế để học biểu diễn đặc trưng của ảnh. "ResNet" là viết tắt của "Residual Network," và số "50" đại diện cho số lượng lớp (hoặc block) trong mạng. Kiến trúc ResNet đã đưa ra đóng góp lớn bằng cách giải quyết vấn đề vanishing gradient trong quá trình huấn luyện mạng nơ-ron sâu.

Cụ thể, mạng ResNet sử dụng các khối có cấu trúc residual (tích chập ngắn) để chuyển dữ liệu từ lớp này sang lớp khác. Điều này giúp tránh hiện tượng mất mát gradient và cho phép huấn luyện mạng nhiều lớp một cách hiệu quả hơn.



Hình 14 ResNet-50

Việc sử dụng ResNet-50 backbone đã được huấn luyện trước trên COCO giúp cho quá trình học bộ dữ liệu đã chuẩn bị nhanh chóng và hiệu quả hơn. Backbone (phần mạng nơ-ron đặc trưng) đã được huấn luyện trước trên một tập dữ liệu lớn như COCO chứa nhiều đối tượng và ngữ cảnh khác nhau. Do đó, nó đã học được các đặc trưng tổng quát và có khả năng tổng quát hóa tốt cho nhiều loại dữ liệu.

Khi fine-tune (huấn luyện lại) mô hình trên bộ dữ liệu cụ thể của mình, sử dụng backbone đã được huấn luyện trước giúp mô hình nhanh chóng "học" đặc trưng đặc biệt của bộ dữ liệu mới mà không cần phải bắt đầu từ đầu. Điều này thường dẫn đến việc có thể đạt được độ chính xác cao hơn và đồng thời giảm thiểu nguy cơ overfitting, đặc biệt là khi bộ dữ liệu mới không lớn.

Tóm lại, việc sử dụng backbone đã được huấn luyện trước là một chiến lược thông minh trong quá trình học máy và thị giác máy tính.

3.4 Thư viện và module được sử dụng để giải quyết bài toán

3.4.1 Thư viện Pytorch

3.4.1.1 Utils.py

Tập utils.py thường chứa một số hàm tiện ích (utility functions) và công cụ hỗ trợ cần thiết cho quá trình huấn luyện và đánh giá mô hình nhận diện đối tượng. Cụ thể, nó có thể bao gồm các hàm sau:

Hàm tiện ích đối với hình ảnh và boxes:

- Chuyển đổi hình ảnh về định dạng thích hợp để đưa vào mô hình.
- Các hàm xử lý và chuyển đổi các hộp giới hạn (bounding boxes).

Các hàm tiện ích đối với dữ liệu:

- Tạo dataloader hoặc các đối tượng khác để đọc dữ liệu và chia thành các batch.

Các hàm tiện ích khác:

- Các công cụ hỗ trợ quá trình đào tạo, như tính toán độ đo hiệu suất của mô hình.
- Các hàm tiện ích cho việc lưu và tải mô hình.
- Tập `utils.py` thường được sử dụng để giữ lại các chức năng chung và giảm sự lặp lại trong mã nguồn của dự án.

3.4.1.2 Transforms.py

Tập `transforms.py` trong ngữ cảnh của mô hình nhận diện đối tượng thường chứa các hàm biến đổi (transforms) cho dữ liệu hình ảnh. Các biến đổi này thường được sử dụng để tăng cường dữ liệu đào tạo và cải thiện khả năng tổng quát hóa của mô hình. Dưới đây là một số công việc phổ biến mà `transforms.py` có thể thực hiện:

Augmentation (Tăng cường):

- Quay ảnh.
- Lật ngang hoặc dọc ảnh.
- Thay đổi độ sáng và độ tương phản.
- Áp dụng nhiễu (noise).

Chuẩn hóa và chuyển đổi định dạng:

- Chuẩn hóa giá trị pixel của ảnh để đưa về khoảng cụ thể (ví dụ: $[0, 1]$ hoặc $[-1, 1]$).

- Chuyển định dạng của ảnh (ví dụ: từ HWC sang CHW).

Xử lý hộp giới hạn:

- Chuẩn hóa kích thước và vị trí của các hộp giới hạn dựa trên các biến đổi được áp dụng cho ảnh.
- Các biến đổi này giúp mô hình học được từ nhiều góc độ và biến thể của dữ liệu, cải thiện khả năng tổng quát hóa của mô hình và giảm nguy cơ quá mức học thuật toán trên dữ liệu đào tạo.

3.4.1.3 Coco_eval.py

Coco_eval.py thường được sử dụng để thực hiện quá trình đánh giá hiệu suất của mô hình nhận diện đối tượng trên bộ dữ liệu COCO (Common Objects in Context). Bộ dữ liệu COCO là một trong những bộ dữ liệu lớn và đa dạng nhất trong lĩnh vực nhận diện đối tượng, nơi mà các đối tượng thuộc nhiều loại khác nhau và có các biến thể về kích thước và độ phức tạp.

Cụ thể, coco_eval.py có thể có những chức năng sau:

Chấm điểm mô hình trên tập kiểm tra COCO:

- Sử dụng mô hình để dự đoán đối tượng trên tập kiểm tra COCO.
- So sánh kết quả dự đoán với các ground truth (thực tế) được cung cấp trong bộ dữ liệu COCO.

Tính toán các độ đo đánh giá:

- Tính toán các độ đo như precision, recall, và các độ đo liên quan đến việc đánh giá hiệu suất của mô hình trên các lớp và trên toàn bộ bộ dữ liệu.

In và báo cáo kết quả:

- In và báo cáo kết quả của quá trình đánh giá, bao gồm các độ đo hiệu suất và thông tin khác.

Trong ngữ cảnh của mô hình nhận diện đối tượng, việc đánh giá trên bộ dữ liệu COCO là quan trọng để đánh giá khả năng tổng quát hóa và hiệu suất của mô hình trên các đối tượng và tình huống đa dạng.

3.4.1.4 Engine.py

Engine.py trong ngữ cảnh của đoạn mã của bạn có thể chứa các hàm và utils để huấn luyện mô hình. Thông thường, file này có thể chứa các chức năng sau:

train_one_epoch:

- Hàm này thực hiện một vòng lặp huấn luyện qua một epoch. Nó đảm bảo rằng mô hình được cập nhật với gradient của hàm mất mát (loss) sau mỗi batch và in thông tin về quá trình huấn luyện, chẳng hạn như loss và tốc độ học tập.

evaluate:

- Hàm này thực hiện quá trình đánh giá hiệu suất của mô hình trên tập kiểm tra. Nó tính toán các độ đo đánh giá, như precision, recall, và F1-score, để đánh giá khả năng nhận diện đối tượng của mô hình.

Các chức năng tiện ích (utilities):

- collate_fn: Một chức năng dùng để gom các mẫu thành các batch.
- Các hàm khác như setup để cài đặt mô hình, optimizer và scheduler.

Quản lý thiết bị (device management):

- Các chức năng để kiểm tra và xác định liệu GPU có sẵn để sử dụng hay không, và sau đó cài đặt mô hình và dữ liệu trên GPU nếu có sẵn.

File engine.py thường chứa các hàm và utils quan trọng để quản lý quá trình huấn luyện và đánh giá, giúp mã nguồn chính trở nên sáng sủa và dễ bảo trì.

3.4.1.5 Coco_utils.py

Coco_utils.py thường được sử dụng để cung cấp các tiện ích và hàm dùng trong quá trình làm việc với bộ dữ liệu COCO (Common Objects in Context). Bộ dữ liệu COCO thường được sử dụng để huấn luyện và đánh giá mô hình máy học trong lĩnh vực nhận diện đối tượng và các nhiệm vụ liên quan.

Các chức năng và utils trong coco_utils.py có thể bao gồm:

get_coco_api_from_dataset:

- Hàm này có thể trích xuất API của COCO từ một đối tượng torchvision.datasets.CocoDetection.

coco_evaluate:

- Hàm này thực hiện quá trình đánh giá mô hình trên tập kiểm tra sử dụng bộ dữ liệu COCO. Nó tính toán các độ đo đánh giá chất lượng như precision, recall, và F1-score.

Các hàm khác liên quan đến COCO:

- Các hàm giúp xử lý và định dạng dữ liệu từ COCO, chẳng hạn như chuyển đổi từ định dạng COCO JSON sang đối tượng Python, tạo các đối tượng Annotation thông qua COCO API, và các chức năng khác liên quan đến quá trình sử dụng bộ dữ liệu COCO.

Việc sử dụng `coco_utils.py` giúp mã nguồn trở nên sáng sủa hơn khi thực hiện các tác vụ liên quan đến bộ dữ liệu COCO trong ngữ cảnh của đoạn mã của bạn.

3.4.1.6 Torchvision

Torchvision: Là một module của PyTorch, tập trung vào các công cụ và datasets liên quan đến thị giác máy tính, thường được sử dụng trong lĩnh vực Computer Vision. torchvision cung cấp nhiều tiện ích như datasets phổ biến (ví dụ: CIFAR-10, MNIST), các chức năng xử lý ảnh, và một số mô hình học sâu tiêu biểu cho thị giác máy tính (ví dụ: ResNet, Faster R-CNN).

3.4.2 Thư viện Scikit-learn

3.4.2.1 Sklearn.preprocessing

Preprocessing trong thư viện scikit-learn là một module cung cấp nhiều công cụ giúp chuẩn hóa và xử lý trước dữ liệu trước khi đưa vào mô hình học máy. Dưới đây là một số chức năng chính:

- Label Encoding (LabelEncoder): Chuyển đổi các nhãn văn bản thành các số nguyên. Điều này thường được sử dụng khi các thuật toán học máy yêu cầu đầu vào là các số nguyên, và dữ liệu đầu vào chứa các nhãn văn bản.

- One-Hot Encoding (OneHotEncoder): Chuyển đổi biểu diễn số nguyên của các nhãn thành biểu diễn one-hot. Điều này hữu ích khi mô hình yêu cầu biểu diễn nhãn dưới dạng các vector nhị phân.

- Min-Max Scaling (MinMaxScaler): Chuẩn hóa giá trị của các đặc trưng về một phạm vi cụ thể, thường là (0, 1). Điều này giúp giảm ảnh hưởng của các đặc trưng có giá trị lớn đối với mô hình.

- Standard Scaling (StandardScaler): Chuẩn hóa giá trị của các đặc trưng theo phân phối chuẩn (mean = 0, standard deviation = 1). Điều này làm giảm ảnh hưởng của outliers và đảm bảo rằng các đặc trưng có cùng thang đo.

- Robust Scaling (RobustScaler): Tương tự như Standard Scaling, nhưng dựa trên giá trị trung vị và phạm vi tương đối với giá trị trung vị, giúp giảm ảnh hưởng của outliers.

- Binning: Chia các giá trị liên tục thành các khoảng rời rạc (bins). Điều này có thể giúp mô hình học máy hiểu cấu trúc trong dữ liệu liên tục.

- Điều chỉnh giá trị Missing (Imputation): Xử lý giá trị thiếu trong dữ liệu bằng cách điền giá trị trung bình, trung vị hoặc giá trị được chỉ định.

3.4.3 Công cụ gán nhãn

3.4.3.1 LabelIMG

LabelIMG là một công cụ mã nguồn mở được thiết kế để giúp người dùng gán nhãn dữ liệu cho hình ảnh. Đây là một phần quan trọng trong quá trình chuẩn bị dữ liệu cho các nhiệm vụ học máy, đặc biệt là trong lĩnh vực thị giác máy tính và nhận diện đối tượng.

Dưới đây là một số điểm quan trọng về công cụ này:

Mục Đích:

- Gán Nhãn Dữ Liệu: LabelIMG chủ yếu được sử dụng để gán nhãn cho các đối tượng trong hình ảnh. Điều này bao gồm việc xác định và đặt tên cho các bounding box (hộp giới hạn) xung quanh các đối tượng quan trọng.

- Chuẩn Bị Dữ Liệu: Công cụ này hỗ trợ quá trình chuẩn bị dữ liệu cho các mô hình học máy, đặc biệt là khi bạn có một tập dữ liệu hình ảnh và cần có thông tin về vị trí và loại của các đối tượng trong ảnh.

Đặc Điểm Quan Trọng:

- Giao Diện Dễ Sử Dụng: LabelIMG có giao diện người dùng đơn giản và dễ sử dụng. Người dùng có thể dễ dàng thêm bounding box, gán nhãn, và điều chỉnh kích thước của chúng.

- Hỗ Trợ Đa Dạng Định Dạng Dữ Liệu: Công cụ này hỗ trợ nhiều định dạng tệp hình ảnh khác nhau, bao gồm JPEG, PNG, và một số định dạng khác.

- Xuất Dữ Liệu Gán Nhãn: LabelIMG có khả năng xuất dữ liệu gán nhãn dưới dạng các định dạng thông dụng, như XML (đối với nhiều mô hình nhận diện đối tượng) hoặc CSV.

Hỗ Trợ Các Đối Tượng Đa Dạng:

- Nhiều Lớp Đối Tượng: Công cụ này cho phép người dùng gán nhãn cho nhiều lớp đối tượng khác nhau trong mỗi hình ảnh.

- Bounding Box Linh Hoạt: Bạn có thể tạo các bounding box với hình dạng và kích thước linh hoạt để phản ánh đúng vị trí và hình dạng của đối tượng.

Hướng Dẫn Sử Dụng:

- Hướng Dẫn Chi Tiết: LabelIMG thường đi kèm với hướng dẫn chi tiết và ví dụ để giúp người dùng mới sử dụng công cụ một cách hiệu quả.

Open Source và Độ Phổ Biến:

- Mã Nguồn Mở: LabelIMG là một dự án mã nguồn mở, điều này có nghĩa là bạn có thể tự do sử dụng, tùy chỉnh và phân phối lại mã nguồn theo các điều khoản của giấy phép mã nguồn mở.

- Độ Phổ Biến: Công cụ này thường được sử dụng trong cộng đồng học máy và thị giác máy tính, do đó, có sẵn nhiều tài liệu và hỗ trợ từ cộng đồng người dùng.

- LabelIMG là một công cụ quan trọng giúp tạo ra tập dữ liệu gán nhãn chất lượng, là bước quan trọng để huấn luyện mô hình nhận diện đối tượng.

3.5 Phương pháp huấn luyện

Phương pháp huấn luyện mô hình RCNN và các biến thể của nó có thể được mô tả dưới dạng lý thuyết toán học, chủ yếu dựa trên các khái niệm và công thức trong học máy. Dưới đây là một biểu diễn lý thuyết toán học đơn giản về quá trình huấn luyện mô hình RCNN:

Mô Hình:

- Cho một mô hình CNN được biểu diễn bằng hàm ánh xạ f_{θ} , với θ là tập hợp các tham số cần được học.

Hàm Mất Mát:

- Xác định một hàm mất mát L để đo lường sự chênh lệch giữa đầu ra dự đoán và đầu ra thực tế. Hàm mất mát thường bao gồm hai thành phần chính: một cho việc phân loại và một cho việc hồi quy vị trí của đối tượng.

- $L(\theta) = L_{cls}(\theta) + \lambda L_{reg}(\theta)$, L_{cls} là hàm mất mát phân loại, L_{reg} là hàm mất mát hồi quy, và λ là hệ số đặt trước.

Phương Trình Tối Ưu Hóa:

- Đặt θ để tối thiểu hóa hàm mất mát: $\theta^* = \operatorname{argmin}_{\theta} L(\theta)$

Gradient Descent:

- Sử dụng thuật toán gradient descent hoặc các biến thể của nó để cập nhật tham số theo hướng giảm dần của gradient của hàm mất mát.

$$\theta_{t+1} = \theta_t - \alpha \nabla \theta L(\theta_t)$$

- Trong đó, α là tốc độ học (learning rate) và $\nabla \theta L(\theta_t)$ là gradient của hàm mất mát tại θ_t .

Tính Tổng Quát Hóa:

- Sử dụng các phương pháp như regularization để ngăn chặn overfitting và đảm bảo khả năng tổng quát hóa của mô hình trên dữ liệu mới.

Kiểm Thử và Đánh Giá:

- Sử dụng tập kiểm thử để đánh giá hiệu suất của mô hình trên dữ liệu mới và đảm bảo không overfitting.

Phương pháp này giúp tối ưu hóa tham số của mô hình và đưa ra dự đoán chính xác về lớp và vị trí của đối tượng trên dữ liệu huấn luyện và kiểm thử.

3.6 Kết luận chương

Chúng em đã trình bày tổng quan về bài toán phát hiện phương tiện giao thông, tổng quan hệ thống phát hiện đối tượng bằng RCNN. Chúng em đã trình bày sự tiến triển của mô hình RCNN qua các biến thể, từ RCNN gốc đến Fast R-CNN, Faster R-CNN và Mask R-CNN nhằm đưa ra lựa chọn các phiên bản tối ưu về tốc độ cũng như độ chính xác cho việc giải quyết bài toán. Trình bày các thư viện và module cần thiết cho quá trình giải quyết bài toán, công cụ chuẩn bị dữ liệu để tiến hành huấn luyện. Đưa ra phương pháp huấn luyện mô hình RCNN và các biến thể của nó có thể được mô tả dưới dạng lý thuyết toán học, chủ yếu dựa trên các khái niệm và công thức trong học máy.

CHƯƠNG 4: XÂY DỰNG MÔ HÌNH

4.1 Thu thập và tiền xử lý dữ liệu.

Bộ dữ liệu thử nghiệm được thu thập từ video giao thông được cung cấp bởi hệ thống camera giao thông của thành phố Hồ Chí Minh, các video được chuyển thành các hình ảnh có kích thước 512x288 hoặc 320x180. Bộ dữ liệu bao gồm 1000 ảnh trong đó 800 ảnh được sử dụng cho quá trình huấn luyện và 200 ảnh cho quá trình kiểm tra. Tất cả các ảnh đều được tiền xử lý để tạo file .xml chú thích thể hiện vị trí của các đối tượng trong ảnh theo định dạng PascalVOC.

Bộ dữ liệu đầu vào đã được gán nhãn gồm các đối tượng:

- motorbike: xe máy
- car: xe hơi
- truck: xe tải
- bus: xe buýt

4.2 Xây dựng

4.2.1 Môi trường thực nghiệm

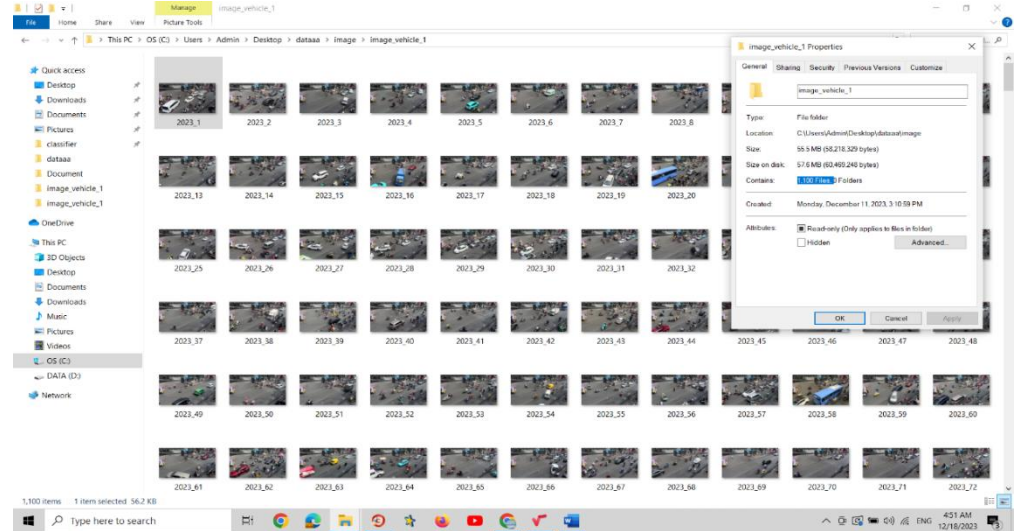
Trong đồ án này cấu hình phần cứng được thử nghiệm là:

- Hệ điều hành: Windows
- Bộ xử lý: Intel Core i5-10300 @ 2.50GHz (8 CPUs)
- Bộ nhớ Ram: 8GB
- Bộ xử lý đồ họa GPU: NVIDIA GTX 1650

4.2.2 Đầu vào, đầu ra của hệ thống

Đầu vào:

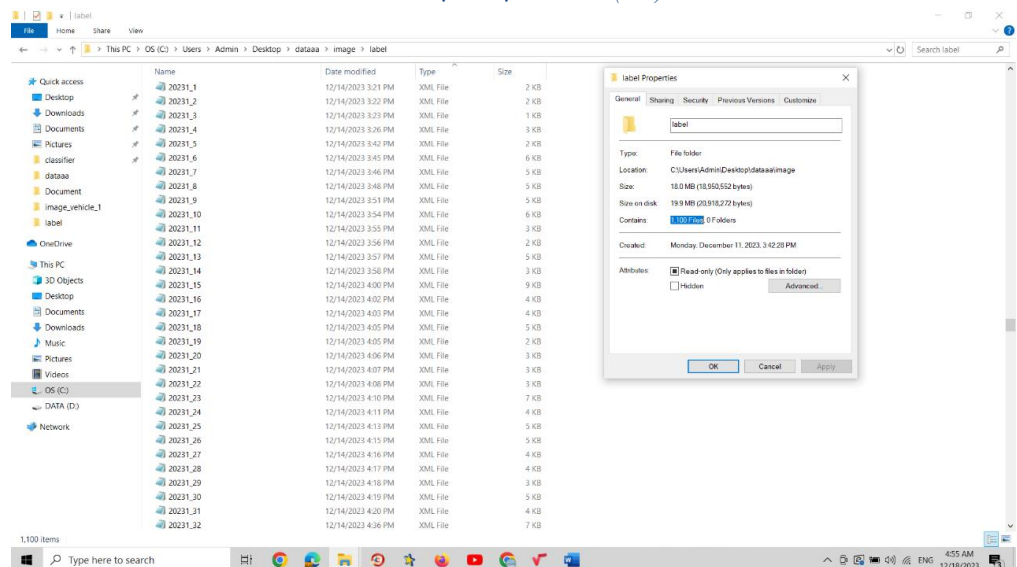
- Bộ dữ liệu ảnh đã thu thập được từ hệ thống camera giao thông thành phố Hồ Chí Minh.



Hình 15 Bộ dữ liệu đầu vào (jpg)

- Bộ dữ liệu đã được gán nhãn (File xml) dạng PascalVOC.

Hình 16 Bộ dữ liệu đầu vào (xml)



File xml lưu các thông tin annotation của hình ảnh gồm:

- <folder>: Thư mục chứa hình ảnh.
- <filename>: Tên tệp hình ảnh.
- <path>: Đường dẫn đến tệp hình ảnh trên hệ thống tệp.

- `<source>`: Thông tin về nguồn dữ liệu, trong trường hợp này là "Unknown".

- `<size>`: Kích thước của hình ảnh, bao gồm chiều rộng (`<width>`), chiều cao (`<height>`), và số kênh màu (`<depth>`).

- `<segmented>`: Cho biết liệu hình ảnh có được phân đoạn hay không.

- `<object>`: Mỗi đối tượng trong hình ảnh được mô tả bởi một phần này.

- `<name>`: Tên của đối tượng (ví dụ: motorbike, car, truck).

- `<pose>`: Tư thế của đối tượng (ví dụ: Unspecified).

- `<truncated>`: Cho biết liệu đối tượng có bị cắt (truncated) hay không (1 nếu có, 0 nếu không).

- `<difficult>`: Cho biết liệu việc nhận diện đối tượng này có khó khăn hay không (1 nếu có, 0 nếu không).

- `<bndbox>`: Hộp giới hạn xác định vị trí của đối tượng trong hình ảnh.

- `<xmin>`, `<ymin>`: Tọa độ (x, y) của điểm góc dưới bên trái của hộp giới hạn.

- `<xmax>`, `<ymax>`: Tọa độ (x, y) của điểm góc trên bên phải của hộp giới hạn.

Đầu ra:

- Tập các ma trận trọng số w và bias của mô hình RCNN (file đuôi pth).

(Sau khi huấn luyện mô hình xong => Tiến hành lưu mô hình dạng trọng số (đuôi .pth)

```
# let's train it for 2 epochs
num_epochs = 2

for epoch in range(num_epochs):
    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, train_data_loader, device, epoch, print_freq=10)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    evaluate(model, valid_data_loader, device=device)
```

Epoch: [0]	[0/248]	eta: 2:17:14	lr: 0.000025	loss: 3.4462 (3.4462)	loss_classifier: 2.5752 (2.5752)	loss_box_reg: 0.7422 (0.7422)	loss_o
Epoch: [0]	[10/248]	eta: 1:53:33	lr: 0.000227	loss: 3.6672 (3.5998)	loss_classifier: 2.5850 (2.5988)	loss_box_reg: 0.9485 (0.8559)	loss_o
Epoch: [0]	[20/248]	eta: 1:49:04	lr: 0.000430	loss: 3.4692 (3.4223)	loss_classifier: 2.5123 (2.4367)	loss_box_reg: 0.9366 (0.8247)	loss_o
Epoch: [0]	[30/248]	eta: 1:43:39	lr: 0.000632	loss: 2.8323 (3.2025)	loss_classifier: 1.9888 (2.2187)	loss_box_reg: 0.8836 (0.8285)	loss_o
Epoch: [0]	[40/248]	eta: 1:39:35	lr: 0.000834	loss: 2.3903 (2.9804)	loss_classifier: 1.3709 (1.9822)	loss_box_reg: 0.9080 (0.8459)	loss_o
Epoch: [0]	[50/248]	eta: 1:34:44	lr: 0.001036	loss: 2.2336 (2.8119)	loss_classifier: 1.1659 (1.7991)	loss_box_reg: 0.9185 (0.8579)	loss_o
Epoch: [0]	[60/248]	eta: 1:29:54	lr: 0.001239	loss: 2.0218 (2.6638)	loss_classifier: 0.9336 (1.6443)	loss_box_reg: 0.9112 (0.8609)	loss_o

Hình 17 Dữ liệu đầu ra

Đuôi tệp tin .pth thường được sử dụng để đặt tên cho các tệp lưu trữ trọng số (weights) của mô hình trong các dự án sử dụng các thư viện học sâu như PyTorch. Tệp tin .pth thường chứa thông tin về trọng số của mô hình, giúp lưu trữ và khôi phục trạng thái của mô hình một cách dễ dàng.

4.2.3 Huấn luyện mô hình

4.2.3.1 Định nghĩa một lớp VOCDataset

```

class VOCDataset(Dataset):

    def __init__(self, dataframe, image_dir, transforms=None):
        super().__init__()

        self.image_ids = dataframe['img_id'].unique()
        self.df = dataframe
        self.image_dir = image_dir
        self.transforms = transforms

    def __getitem__(self, index: int):
        image_id = self.image_ids[index]
        records = self.df[self.df['img_id'] == image_id]

        image = cv2.imread(f'{self.image_dir}/{image_id}.jpg', cv2.IMREAD_COLOR)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB).astype(np.float32)
        image /= 255.0
        rows, cols = image.shape[:2]

        boxes = records[['xmin', 'ymin', 'xmax', 'ymax']].values

        area = (boxes[:, 3] - boxes[:, 1]) * (boxes[:, 2] - boxes[:, 0])
        area = torch.as_tensor(area, dtype=torch.float32)

        label = records['labels'].values
        labels = torch.as_tensor(label, dtype=torch.int64)

        iscrowd = torch.zeros((records.shape[0],), dtype=torch.int64)

        target = {}
        target['boxes'] = boxes
        target['labels'] = labels
        # target['masks'] = None
        target['image_id'] = torch.tensor([index])
        target['area'] = area
        target['iscrowd'] = iscrowd

        if self.transforms:
            sample = {
                'image': image,
                'bboxes': target['boxes'],
                'labels': labels
            }
            sample = self.transforms(**sample)
            image = sample['image']

        target['boxes'] = torch.stack(tuple(map(torch.tensor, zip(*sample['bboxes'])))).permute(1,0)

        return image, target

```

Hình 18 Định nghĩa một lớp VOCDataset

Đoạn mã trên định nghĩa một lớp VOCDataset là một lớp con của lớp Dataset trong PyTorch, được thiết kế để làm việc với dữ liệu của bộ dữ liệu PASCAL VOC. Dưới đây là giải thích chi tiết cho từng phần của mã:

Hàm `__init__`:

- Khởi tạo đối tượng VOCDataset.
- dataframe: DataFrame chứa thông tin về đối tượng trong hình ảnh.
- image_dir: Đường dẫn đến thư mục chứa các hình ảnh.
- transforms: Các biến đổi hình ảnh (nếu có).

Hàm `__getitem__`:

- Lấy một mẫu từ tập dữ liệu tại vị trí index.
- Đọc hình ảnh từ đường dẫn và chuyển đổi không gian màu sang RGB.
- Chuẩn hóa giá trị pixel hình ảnh về khoảng $[0, 1]$.
- Lấy thông tin về các hộp giới hạn (boxes), diện tích (area), nhãn (labels), và iscrowd từ DataFrame.
- Tạo một từ điển target chứa thông tin về hộp giới hạn, nhãn, diện tích, và iscrowd.
- Nếu có các biến đổi (transforms), áp dụng chúng và cập nhật thông tin trong target.
- Trả về hình ảnh và target làm kết quả.

Trong hàm `__getitem__`, đối tượng trả về là một tuple (image, target), trong đó:

- image: Hình ảnh đã được chuẩn hóa và biến đổi (nếu có).
- target: Thông tin về các đối tượng trong hình ảnh, bao gồm hộp giới hạn, nhãn, diện tích, và iscrowd.

Hàm `__len__`:

- Trả về số lượng hình ảnh trong tập dữ liệu.

4.2.3.2 Biến đổi hình ảnh

```

def get_transform_train():
    return A.Compose([
        A.HorizontalFlip(p=0.5),
        A.RandomBrightnessContrast(p=0.2),
        ToTensorV2(p=1.0)
    ], bbox_params={'format': 'pascal_voc', 'label_fields': ['labels']})

def get_transform_valid():
    return A.Compose([
        ToTensorV2(p=1.0)
    ], bbox_params={'format': 'pascal_voc', 'label_fields': ['labels']})

```

Hình 19 Biến đổi hình ảnh

Đây là hai hàm `get_transform_train` và `get_transform_valid` để lấy các biến đổi hình ảnh sử dụng thư viện Augmentor (A) trong quá trình huấn luyện và kiểm thử của mô hình.

Hàm `get_transform_train`:

Biến đổi:

- `HorizontalFlip`: Lật hình ảnh ngang với xác suất $p=0.5$ (50%). Điều này giúp tăng cường dữ liệu bằng cách tạo ra các phiên bản lật ngang của hình ảnh.

- `RandomBrightnessContrast`: Thực hiện ngẫu nhiên thay đổi độ sáng và độ tương phản của hình ảnh với xác suất $p=0.2$ (20%). Điều này giúp mô hình trở nên chịu đựng hơn đối với sự biến động của ánh sáng và tương phản trong dữ liệu thực tế.

- `ToTensorV2`: Chuyển đổi hình ảnh và thông tin về hộp giới hạn sang định dạng Tensor để có thể sử dụng được trong PyTorch.

Tham số `bbox_params`:

- `'format': 'pascal_voc'`: Xác định định dạng của thông tin hộp giới hạn theo chuẩn PASCAL VOC.

- `'label_fields': ['labels']`: Chỉ định tên trường chứa thông tin nhãn.

Hàm `get_transform_valid`:

Biến đổi:

- ToTensorV2: Chuyển đổi hình ảnh và thông tin về hộp giới hạn sang định dạng Tensor để có thể sử dụng được trong PyTorch.

Tham số bbox_params:

- 'format': 'pascal_voc': Xác định định dạng của thông tin hộp giới hạn theo chuẩn PASCAL VOC.

- 'label_fields': ['labels']: Chỉ định tên trường chứa thông tin nhãn.

Cả hai hàm này đều trả về một đối tượng Augmentor Compose, chứa một chuỗi các biến đổi hình ảnh để được sử dụng trong quá trình huấn luyện và kiểm thử mô hình.

4.2.3.3 Tạo các DataLoader

```
def collate_fn(batch):  
    return tuple(zip(*batch))  
  
train_dataset = VOCDataset(train_df, IMG_PATH, get_transform_train())  
valid_dataset = VOCDataset(valid_df, IMG_PATH, get_transform_valid())  
  
# chia tập dữ liệu trong tập huấn luyện và tập kiểm tra  
indices = torch.randperm(len(train_dataset)).tolist()  
  
train_data_loader = DataLoader(  
    train_dataset,  
    batch_size=4,  
    shuffle=True,  
    num_workers=4,  
    collate_fn=collate_fn  
)  
  
valid_data_loader = DataLoader(  
    valid_dataset,  
    batch_size=4,  
    shuffle=False,  
    num_workers=4,  
    collate_fn=collate_fn  
)
```

Hình 20 Tạo các DataLoader

Mã trên định nghĩa quá trình chuẩn bị dữ liệu và tạo các DataLoader để sử dụng trong quá trình huấn luyện và kiểm thử mô hình. Dưới đây là giải thích chi tiết:

Hàm `collate_fn(batch)`:

- Hàm này được sử dụng trong quá trình chia nhóm các mẫu thành các batch trong DataLoader.

- batch: Một list chứa các mẫu, mỗi mẫu là một tuple (image, target).

Khởi tạo đối tượng VOCDataset:

- train_dataset: Đối tượng dataset cho tập huấn luyện, được khởi tạo từ lớp VOCDataset với dữ liệu từ train_df (DataFrame chứa thông tin về đối tượng) và đường dẫn đến hình ảnh (IMG_PATH), cùng với hàm biến đổi `get_transform_train()`.

- valid_dataset: Đối tượng dataset cho tập kiểm thử, tương tự như train_dataset nhưng sử dụng `get_transform_valid()`.

Chia tập dữ liệu trong tập huấn luyện:

- indices: Chuỗi các số nguyên từ 0 đến `len(train_dataset)` được xáo trộn ngẫu nhiên.

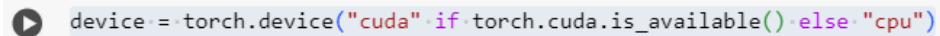
- train_data_loader: DataLoader cho tập huấn luyện, sử dụng các tham số như `batch_size=4`, `shuffle=True` để xáo trộn dữ liệu, `num_workers=4` để sử dụng 4 workers trong quá trình nạp dữ liệu, và `collate_fn=collate_fn` để sử dụng hàm chia nhóm đã được định nghĩa trước đó.

Chia tập dữ liệu trong tập kiểm thử:

- valid_data_loader: DataLoader cho tập kiểm thử, tương tự như train_data_loader nhưng với `shuffle=False` để giữ nguyên thứ tự của dữ liệu.

- Những bước này giúp chuẩn bị dữ liệu và tạo DataLoader để có thể sử dụng chúng trong quá trình huấn luyện và kiểm thử mô hình.

4.2.3.3 Kiểm tra xem có GPU (đồng thời có hỗ trợ CUDA)

A code snippet in a light blue box with a play button icon on the left. The code is: `device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`. The string "cuda" is in red, "if" is in blue, "torch.cuda.is_available()" is in green, "else" is in red, and "cpu" is in blue.

Hình 21 Kiểm tra xem có GPU

Dòng mã này kiểm tra xem có GPU (đồng thời có hỗ trợ CUDA) hay không và sau đó chọn thiết bị để thực hiện các phép tính của PyTorch dựa trên kết quả kiểm tra đó.

`torch.cuda.is_available()`: Hàm này trả về True nếu CUDA (một nền tảng tính toán GPU của NVIDIA) có sẵn để sử dụng và False nếu không.

`torch.device("cuda" if torch.cuda.is_available() else "cpu")`: Sử dụng kết quả của `torch.cuda.is_available()` để chọn thiết bị. Nếu CUDA có sẵn, thiết bị được chọn là "cuda", nếu không, thiết bị được chọn là "cpu".

Nói cách khác, nếu máy tính của bạn có GPU và CUDA được cài đặt, thì device sẽ được chọn là "cuda" để sử dụng GPU cho tính toán. Ngược lại, nếu không có GPU hoặc CUDA không được cài đặt, thì device sẽ là "cpu", và các phép tính sẽ được thực hiện trên CPU. Điều này giúp mã chương trình linh hoạt để thích ứng với sự có mặt hay vắng mặt của GPU trên máy tính.

4.2.3.4 Kiểm tra

```

images, targets= next(iter(train_data_loader))
images = list(image.to(device) for image in images)
targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

plt.figure(figsize=(20,20))
for i, (image, target) in enumerate(zip(images, targets)):
    plt.subplot(2,2, i+1)
    boxes = targets[i]['boxes'].cpu().numpy().astype(np.int32)
    sample = images[i].permute(1,2,0).cpu().numpy()
    names = targets[i]['labels'].cpu().numpy().astype(np.int64)
    for i,box in enumerate(boxes):
        cv2.rectangle(sample,
                       (box[0], box[1]),
                       (box[2], box[3]),
                       (0, 0, 220), 2)
        cv2.putText(sample, classes[names[i]], (box[0],box[1]+15),cv2.FONT_HERSHEY_COMPLEX ,0.5,(0,220,0),1,cv2.LINE_AA)

plt.axis('off')
plt.imshow(sample)

```

Hình 22 Kiểm tra

Đoạn mã này thực hiện các bước sau để hiển thị một số mẫu từ tập dữ liệu huấn luyện:

Lấy một batch từ DataLoader:

- `images, targets = next(iter(train_data_loader))`: Sử dụng hàm `next` để lấy một batch từ `train_data_loader`. `images` là danh sách các hình ảnh và `targets` là danh sách các thông tin về các đối tượng trong hình ảnh.

Chuyển đổi dữ liệu lên thiết bị GPU (nếu có):

- `images = list(image.to(device) for image in images)`: Chuyển đổi danh sách các hình ảnh lên thiết bị được chọn (cuda nếu có GPU, cpu nếu không).
- `targets = [{k: v.to(device) for k, v in t.items()} for t in targets]`: Chuyển đổi danh sách các thông tin về đối tượng lên thiết bị.

Hiển thị hình ảnh và thông tin về đối tượng:

- Sử dụng `plt.figure` để tạo một hình với kích thước 20x20 inch.

Sử dụng vòng lặp để duyệt qua từng mẫu trong batch:

- `boxes = targets[i]['boxes'].cpu().numpy().astype(np.int32)`: Lấy thông tin về hộp giới hạn từ `targets`.

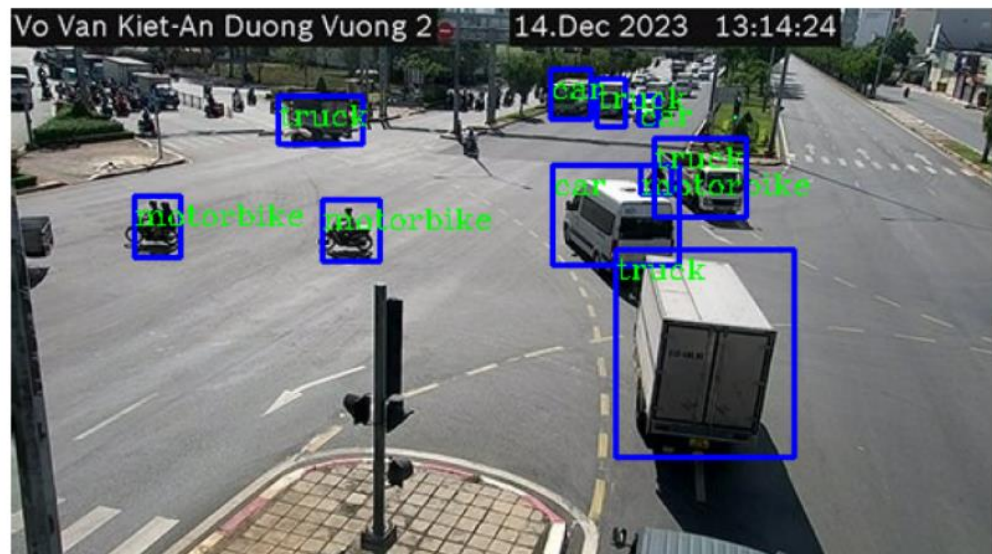
- `sample = images[i].permute(1,2,0).cpu().numpy()`: Chuyển đổi hình ảnh từ định dạng tensor sang numpy và sắp xếp lại các chiều của hình ảnh để có thể hiển thị được.

- `names = targets[i]['labels'].cpu().numpy().astype(np.int64)`: Lấy thông tin về nhãn của các đối tượng từ targets.

- Sử dụng OpenCV để vẽ hộp giới hạn và hiển thị nhãn lên hình ảnh.

- Sử dụng `plt.axis('off')` để ẩn trục trong đồ thị và `plt.imshow(sample)` để hiển thị hình ảnh.

Đoạn mã này giúp kiểm tra xem quá trình chuẩn bị dữ liệu và chuyển đổi thiết bị đã diễn ra đúng cách hay không bằng cách hiển thị một số mẫu từ tập dữ liệu huấn luyện.



Hình 23 Mô hình sau khi gán nhãn

4.2.3.5 Mô hình đã được đào tạo trước về COCO

COCO là viết tắt của "Common Objects in Context." Đây là một bộ dữ liệu lớn và đa dạng được sử dụng phổ biến trong lĩnh vực thị giác máy tính và học

máy để huấn luyện và đánh giá các mô hình nhận diện và định vị đối tượng trong ảnh.

Bộ dữ liệu COCO bao gồm một số lượng lớn hình ảnh chất lượng cao, chứa nhiều loại đối tượng khác nhau được gắn nhãn. Các hình ảnh trong COCO được thu thập từ nhiều nguồn và bối cảnh khác nhau, bao gồm cả ảnh trong các bối cảnh đô thị và nông thôn. Các đối tượng trong COCO được gắn nhãn với các loại như người, xe, động vật, và các đối tượng khác.

Đối với các nhiệm vụ như nhận diện đối tượng, định vị, và phân loại trong hình ảnh, việc sử dụng COCO là lựa chọn phổ biến vì sự đa dạng và phong phú của dữ liệu. Các mô hình huấn luyện trên COCO thường có khả năng tổng quát tốt và có thể áp dụng cho nhiều bài toán thị giác máy tính khác nhau.

Tại sao cần phải tải bộ dữ liệu COCO này?

Việc tải một mô hình đã được đào tạo trước như Faster R-CNN với kiến trúc ResNet-50 từ tập dữ liệu COCO có một số ưu điểm:

Transfer Learning: Mô hình đã được đào tạo trước giúp thực hiện transfer learning. Điều này có nghĩa là mô hình đã học được các đặc trưng từ tập dữ liệu lớn (COCO) và có khả năng truyền những hiểu biết này sang một bài toán mới (nhận diện phương tiện giao thông trong trường hợp này).

Đặc trưng chung: Kiến trúc ResNet-50 được tích hợp trong mô hình Faster R-CNN giúp trích xuất đặc trưng từ hình ảnh hiệu quả. ResNet-50 là một mạng nơ-ron sâu với khả năng học các đặc trưng phức tạp từ hình ảnh.

Khả năng định vị và phân loại đối tượng: Faster R-CNN là một trong những mô hình tiên tiến trong việc định vị và phân loại đối tượng trong hình ảnh. Nó sử dụng region proposal network (RPN) để đề xuất các vùng quan trọng có thể chứa đối tượng, sau đó sử dụng các lớp phân loại để xác định loại đối tượng và hộp giới hạn của chúng.

Tiết kiệm thời gian và công sức: Việc tải một mô hình đã được đào tạo trước giúp tránh phải đào tạo mô hình từ đầu, điều này đòi hỏi nhiều thời gian và tài nguyên tính toán. Mô hình đã được đào tạo trước mang lại một khởi đầu tốt cho bài toán cụ thể của bạn.

Tóm lại, việc tải một mô hình đã được đào tạo trước giúp gia tăng hiệu suất và giảm thiểu khả năng overfitting khi bạn đối diện với tập dữ liệu nhỏ hoặc khi tài nguyên tính toán có hạn.

```
# tải một mô hình; được đào tạo trước về COCO ##  
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
```

Hình 24 Tải mô hình được đào tạo trước về COCO

Giải thích từng phần của dòng mã:

- `torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)`:

Sử dụng hàm này để tải mô hình Faster R-CNN với kiến trúc ResNet-50 và được đào tạo trước trên tập dữ liệu COCO. Các tham số của hàm:

- `pretrained=True`: Chọn mô hình đã được đào tạo trước về COCO.

Mô hình Faster R-CNN là một mô hình phát hiện đối tượng, và kiến trúc ResNet-50 là một mạng nơ-ron sâu (deep neural network) được sử dụng để trích xuất đặc trưng từ hình ảnh. Mô hình đã được đào tạo trên COCO, một tập dữ liệu lớn chứa hình ảnh với nhiều đối tượng khác nhau được gán nhãn, nơi mà mô hình đã học được cách phân loại và định vị các đối tượng.

4.2.3.6 Classifier

```

num_classes = 10

# lấy số lượng đặc trưng đầu vào cho bộ phân loại
in_features = model.roi_heads.box_predictor.cls_score.in_features

# thay thế đầu đã được huấn luyện trước bằng đầu mới
model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

```

Hình 25 Classifier

Đoạn mã trên thực hiện việc thay đổi bộ phân loại (classifier) của mô hình Faster R-CNN để phù hợp với số lượng lớp mới trong bài toán cụ thể. Dưới đây là giải thích từng phần của mã:

- `num_classes = 5`: Đây là số lượng lớp mới mà bạn muốn mô hình phân loại. Trong trường hợp này, có 5 lớp đối tượng cần phân loại.

- `in_features = model.roi_heads.box_predictor.cls_score.in_features`: Lấy số lượng đặc trưng đầu vào cho bộ phân loại hiện tại của mô hình. Điều này quan trọng để xác định số lượng đặc trưng đầu vào cần thiết cho bộ phân loại mới.

- `model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)`: Thay thế bộ phân loại hiện tại của mô hình bằng một bộ phân loại mới

(`FastRCNNPredictor`). Điều này được thực hiện để điều chỉnh số lượng lớp đầu ra của bộ phân loại để phù hợp với số lượng lớp mới. Cụ thể:

- `in_features`: Số lượng đặc trưng đầu vào cho bộ phân loại mới, được lấy từ bước trước.

- `num_classes`: Số lượng lớp mới bạn muốn phân loại.

Điều này là cần thiết khi bạn sử dụng một mô hình đã được đào tạo trước và muốn fine-tune hoặc sử dụng lại mô hình đó cho một nhiệm vụ cụ thể với số lượng lớp khác nhau. Thay đổi bộ phân loại giúp mô hình áp dụng kiến thức đã học từ tập dữ liệu COCO (hoặc bộ dữ liệu khác) vào bài toán mới của bạn.

4.2.3.7 Cài đặt COCOAPI và cài đặt module từ thư viện Pytorch

```
#!pip install -U 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'  
  
#!git clone https://github.com/pytorch/vision.git  
#!cd vision;cp references/detection/utils.py ../;cp references/detection/transforms.py ../;cp references/detection/coco_eval.py  
#../;cp references/detection/engine.py ../;cp references/detection/coco_utils.py ../
```

Hình 26 Cài đặt COCOAPI và cài đặt module từ thư viện Pytorch

COCOAPI:

!: Ký hiệu này được sử dụng trong môi trường notebook của Jupyter để chỉ ra rằng lệnh sau đó sẽ được thực thi trong một cell của notebook, thay vì trong một môi trường command line.

pip install -U: Là lệnh pip để cài đặt hoặc nâng cấp một gói Python. -U là cờ để nâng cấp gói nếu đã cài đặt.

'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'

: Đây là URL của kho lưu trữ Git của dự án cocoapi trên GitHub. Khi bạn chạy lệnh này, pip sẽ tải mã nguồn từ URL này và cài đặt gói cocoapi. Phần #subdirectory=PythonAPI chỉ ra rằng pip chỉ cần quan tâm đến thư mục PythonAPI của dự án (chứa mã nguồn Python cần thiết).

Điều này làm cho cocoapi có sẵn trong môi trường Python của bạn, và bạn có thể sử dụng nó để tương tác với bộ dữ liệu COCO và các công cụ khác liên quan đến dự án COCO. Thường thì, khi bạn làm việc với các mô hình nhận diện đối tượng và sử dụng các bộ dữ liệu như COCO, việc cài đặt cocoapi là quan trọng để sử dụng các chức năng và tiện ích liên quan.

PyTorch:

- git clone https://github.com/pytorch/vision.git: Đây là lệnh để sao chép (clone) toàn bộ kho lưu trữ PyTorch Vision từ địa chỉ GitHub https://github.com/pytorch/vision.git. Điều này tạo ra một thư mục mới có tên là vision trên máy tính của bạn, chứa mã nguồn của PyTorch Vision.

- cd vision: Điều này chuyển đến thư mục vision vừa tạo ra.

- `cp references/detection/utils.py ./`: Đây là lệnh để sao chép tệp tin `utils.py` từ thư mục `references/detection/` của PyTorch Vision và đặt nó vào thư mục cha của `vision` (thư mục mà bạn đang ở).

- `cp references/detection/transforms.py ./`: Tương tự, lệnh này sao chép tệp tin `transforms.py` từ thư mục `references/detection/` và đặt nó vào thư mục cha của `vision`.

- `cp references/detection/coco_eval.py ./`: Làm tương tự cho tệp tin `coco_eval.py`.

- `cp references/detection/engine.py ./`: Làm tương tự cho tệp tin `engine.py`.

- `cp references/detection/coco_utils.py ./`: Làm tương tự cho tệp tin `coco_utils.py`.

Những tệp tin này (`utils.py`, `transforms.py`, `coco_eval.py`, `engine.py`, `coco_utils.py`) thường được sử dụng trong các ví dụ và mã nguồn mẫu của PyTorch Vision để hỗ trợ quá trình huấn luyện và đánh giá mô hình nhận diện đối tượng trên các bộ dữ liệu như COCO. Điều này giúp người dùng nhanh chóng sử dụng và thí nghiệm với các mô hình và dữ liệu.

4.2.3.8 Tiến hành huấn luyện

```
from engine import train_one_epoch, evaluate
import utils

# huấn luyện trong 2 epochs
num_epochs = 2

for epoch in range(num_epochs):
    # in sau mỗi 10 lần lặp
    train_one_epoch(model, optimizer, train_data_loader, device, epoch, print_freq=10)
    # cập nhật tốc độ học tập
    lr_scheduler.step()

torch.save(model.state_dict(), 'faster_rcnn.pth')
```

Hình 27 Tiến hành huấn luyện

Đoạn mã trên thực hiện quá trình huấn luyện mô hình Faster R-CNN trong 2 epochs trên tập dữ liệu huấn luyện. Dưới đây là giải thích từng phần của mã:

- `from engine import train_one_epoch, evaluate`: Import hai hàm `train_one_epoch` và `evaluate` từ module `engine`. Các hàm này thường được sử dụng để thực hiện quá trình huấn luyện và đánh giá mô hình.

- `import utils`: Import module `utils`. Có thể đây là một module chứa các hàm tiện ích hỗ trợ quá trình huấn luyện và đánh giá.

- `num_epochs = 2`: Đặt số epochs muốn huấn luyện là 2.

- Vòng lặp `for epoch in range(num_epochs)`: Lặp qua số lượng epochs đã đặt.

- `train_one_epoch(model, optimizer, train_data_loader, device, epoch, print_freq=10)`: Gọi hàm `train_one_epoch` để thực hiện một epoch của quá trình huấn luyện. Các tham số truyền vào bao gồm mô hình (`model`), bộ tối ưu hóa (`optimizer`), dữ liệu huấn luyện (`train_data_loader`), thiết bị sử dụng (`device`), số epoch hiện tại (`epoch`), và tần suất in thông tin (`print_freq`).

- `lr_scheduler.step()`: Cập nhật tốc độ học tập sau mỗi epoch bằng cách gọi hàm `step` trên `lr_scheduler`. Tốc độ học tập có thể được điều chỉnh trong quá trình huấn luyện để giúp mô hình hội tụ tốt hơn.

- `torch.save(model.state_dict(), 'faster_rcnn.pth')`: Lưu trạng thái của mô hình (các trọng số và tham số) sau khi huấn luyện vào tệp tin có tên là `'faster_rcnn.pth'`. Điều này cho phép bạn sau này có thể tái sử dụng mô hình đã được huấn luyện mà không cần phải huấn luyện lại từ đầu.

4.3 Giao diện hệ thống

4.3.1 Thiết kế giao diện

Tạo một cửa sổ GUI sử dụng thư viện **tkinter** và **ttkthemes**



Hình 28 Giao diện chính

Dưới đây là mô tả về code sử dụng các kỹ thuật và thư viện cụ thể:

tkinter (import tkinter as tk, from tkinter import ttk):

- tk.Tk(): Tạo một cửa sổ chính cho ứng dụng.
- ttk.Label: Tạo nhãn (label) để hiển thị văn bản trên giao diện.
- ttk.Button: Tạo các nút cho người dùng thực hiện các hành động.
- pack(): Quyết định cách hiển thị các phần tử trên cửa sổ.

PIL (from PIL import Image, ImageTk):

- Image.open(): Mở một hình ảnh từ đường dẫn.
- Image.thumbnail(): Giảm kích thước hình ảnh để hiển thị trong kích thước nhất định.

- ImageTk.PhotoImage(): Chuyển đổi hình ảnh PIL thành đối tượng PhotoImage sử dụng trong Label của tkinter.

os (import os):

- `os.path.exists()`: Kiểm tra xem tệp tin hoặc thư mục có tồn tại hay không.
- `ttkthemes (from ttkthemes import ThemedStyle)`:
 - `ThemedStyle()`: Cho phép bạn áp dụng các theme khác nhau cho giao diện người dùng.
 - `os.system("python ...")`:
 - `os.system()`: Thực thi một lệnh trong hệ thống dòng lệnh. Trong trường hợp này, nó được sử dụng để chạy các tệp Python khác (`folder.py`, `zone.py`, `image.py`).
- Hiển thị cửa sổ (`window.mainloop()`):
 - `mainloop()`: Chạy vòng lặp chính của ứng dụng để theo dõi các sự kiện và tương tác của người dùng.

4.3.2 Gán nhãn folder hình ảnh

Khi chọn chức năng “Folder” thì sẽ load và gán tất cả các hình có trong folder đã chọn



Hình 29 Gán nhãn trong folder

Các kỹ thuật đã sử dụng trong đoạn code

Tkinter và `filedialog`:

- Tkinter: Một thư viện GUI (Giao diện người dùng đồ họa) cho Python. Được sử dụng để tạo cửa sổ và các thành phần giao diện người dùng.

- filedialog: Một module trong Tkinter được sử dụng để hiển thị hộp thoại chọn file hoặc thư mục.

Matplotlib:

- plt.subplot: Tạo các subplot trong biểu đồ để hiển thị nhiều hình ảnh trong cùng một cửa sổ.

- plt.imshow: Hiển thị hình ảnh trong subplot.

- plt.show: Hiển thị toàn bộ biểu đồ.

- OpenCV (cv2):

- cv2.imread: Đọc hình ảnh từ đường dẫn.

- cv2.cvtColor: Chuyển đổi không gian màu của hình ảnh.

- cv2.rectangle và cv2.putText: Vẽ hộp giới hạn và hiển thị tên lớp lên hình ảnh.

PyTorch và torchvision:

- torch.from_numpy: Chuyển đổi mảng NumPy sang Tensor PyTorch.

- model.eval(): Chuyển mô hình sang chế độ đánh giá.

- model.to(device): Chuyển mô hình lên GPU nếu có sẵn.

- model.load_state_dict: Nạp trọng số đã được huấn luyện cho mô hình Faster R-CNN.

- FastRCNNPredictor: Thay đổi lớp dự đoán của mô hình để phù hợp với số lượng lớp mong muốn.

Numpy:

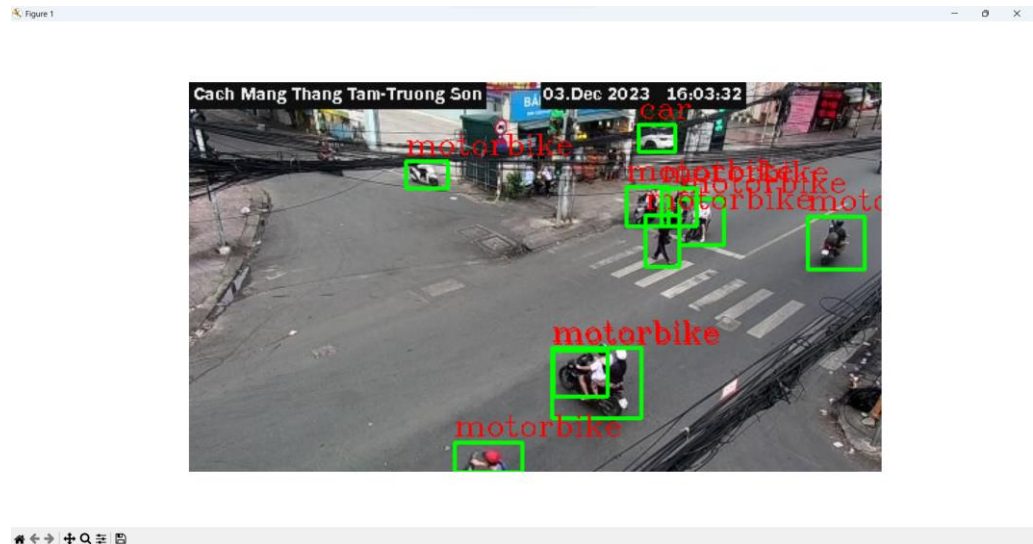
- np.min, np.ceil, np.sqrt: Các phép toán toán học để tính toán số lượng cột, số lượng dòng, và kích thước subplot.

- ThemedStyle từ ttkthemes:

- ThemedStyle: Được sử dụng để thay đổi giao diện của các thành phần ttk (themed Tkinter).

4.3.3 Gán nhãn image

Khi chọn chức năng “Image” thì sẽ chọn 1 hình và gán tất cả đối tượng trong hình



Hình 30 Gán nhãn Image

Các kỹ thuật sử dụng trong chức năng

Tkinter và filedialog:

- Tkinter: Thư viện tạo giao diện người dùng đồ họa (GUI).
- filedialog: Mô-đun trong Tkinter được sử dụng để hiển thị hộp thoại chọn file.

Matplotlib:

- plt.imshow và plt.show: Để hiển thị hình ảnh và biểu đồ.
- OpenCV (cv2):
- cv2.imread và cv2.cvtColor: Đọc và chuyển đổi hình ảnh sang không gian màu phù hợp cho việc xử lý.

- cv2.rectangle và cv2.putText: Vẽ hộp giới hạn và hiển thị tên lớp lên hình ảnh.

PyTorch và torchvision:

- fasterrcnn_resnet50_fpn và FastRCNNPredictor: Sử dụng mô hình Faster R-CNN và thay đổi đầu ra để phù hợp với số lượng lớp bạn đang làm việc.

- torch.load và model.load_state_dict: Nạp trọng số đã được huấn luyện cho mô hình.

NumPy:

- Numpy operations: Nhiều phép toán toán học được sử dụng để chuẩn bị dữ liệu cho việc đưa vào mô hình.

MessageBox từ Tkinter:

- tk.messagebox.showinfo: Hiển thị thông báo đơn giản trong một hộp thoại.

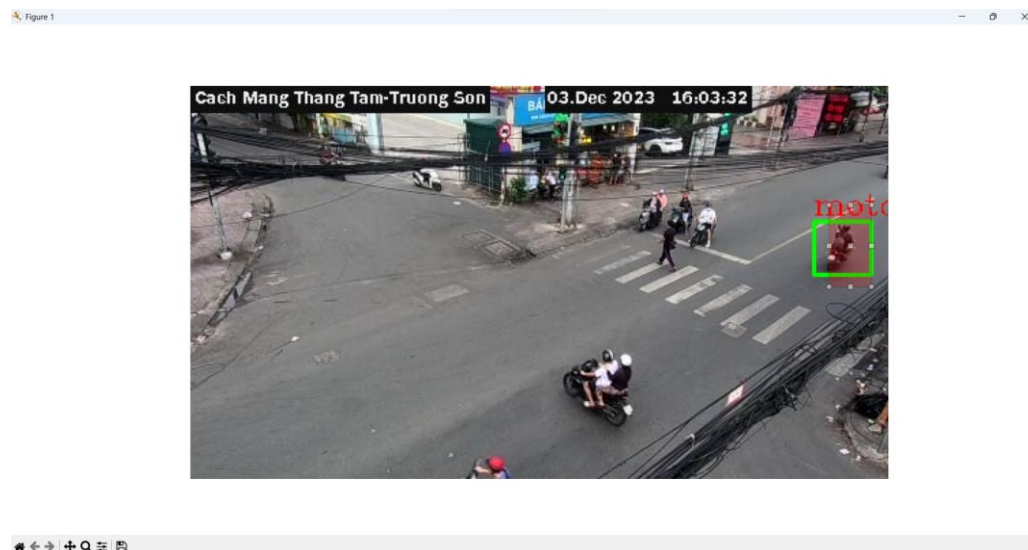
- matplotlib.pyplot.subplots: Sử dụng để tạo subplot để hiển thị hình ảnh và nhãn.

Thư viện os:

- os.path.normpath: Chuẩn hóa đường dẫn file.

4.3.4 Gán nhãn một vùng trong hình ảnh

Khi chọn chức năng “Zone” thì sẽ chọn 1 hình và gán 1 vùng chỉ định trong hình



Hình 31 Gán nhãn một vùng trong hình

Các kỹ thuật sử dụng trong chức năng

Tkinter và filedialog:

- Sử dụng để tạo cửa sổ chọn file và hiển thị hộp thoại để người dùng chọn hình ảnh.

- Matplotlib và RectangleSelector:

- Sử dụng Matplotlib để hiển thị hình ảnh và vùng chọn.

- RectangleSelector là một công cụ trong Matplotlib cho phép người dùng vẽ một hình chữ nhật để chọn vùng quan tâm.

OpenCV (cv2):

- Sử dụng để đọc và xử lý hình ảnh.

- cv2.rectangle và cv2.putText được sử dụng để vẽ hình chữ nhật và hiển thị tên lớp trên hình ảnh.

PyTorch và torchvision:

- Load mô hình Faster R-CNN để nhận diện vật thể trong hình ảnh.

Sự kiện chọn vùng:

- Sự kiện được gắn với việc chọn vùng sử dụng `RectangleSelector`. Khi người dùng vẽ một hình chữ nhật, sự kiện này sẽ được kích hoạt để thực hiện nhận diện vật thể trong vùng đã chọn.

- Hiển thị hình ảnh đã chọn:

- Khi vùng quan tâm được chọn, hình ảnh sẽ được hiển thị lại với các vật thể được nhận diện và được gán nhãn.

Điều này là cần thiết khi bạn sử dụng một mô hình đã được đào tạo trước và muốn fine-tune hoặc sử dụng lại mô hình đó cho một nhiệm vụ cụ thể với số lượng lớp khác nhau. Thay đổi bộ phân loại giúp mô hình áp dụng kiến thức đã học từ tập dữ liệu COCO (hoặc bộ dữ liệu khác) vào bài toán mới của bạn.

4.4 Phương pháp đánh giá

Độ Chính Xác (Accuracy):

- Độ chính xác là tỷ lệ giữa số lượng đối tượng được phân loại đúng và tổng số đối tượng trong tập dữ liệu kiểm thử. Có thể tính theo công thức:

$Accuracy = \text{Số đối tượng phân loại đúng} / \text{Tổng số đối tượng}$

Đo Lường Precision, Recall, và F1-Score:

- Precision (Chính xác): Tỷ lệ số lượng đối tượng được phân loại đúng trên tổng số lượng đối tượng được phân loại.

- Recall (Tổ chức): Tỷ lệ số lượng đối tượng được phân loại đúng trên tổng số lượng đối tượng thực tế.

- F1-Score: Kết hợp giữa precision và recall, được tính theo công thức

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Mean Average Precision (mAP):

- mAP là một phương pháp đánh giá phổ biến trong lĩnh vực nhận diện đối tượng. Nó đo lường hiệu suất của mô hình dựa trên độ chính xác của các

bounding box và sự đánh giá theo độ chính xác ở nhiều ngưỡng IOU (Intersection over Union) khác nhau.

- mAP được tính bằng cách tính trung bình của các giá trị Precision-Recall Curve (PR Curve) cho từng lớp.

IoU (Intersection over Union):

- IoU là một phương pháp đánh giá sự chồng lấn giữa bounding box dự đoán và bounding box thực tế của đối tượng. Giá trị IoU được tính theo công thức:

$$\text{IoU} = \text{Diện tích chồng lấn} / \text{Diện tích hợp nhất}$$

- Một ngưỡng IoU thường được sử dụng để quyết định xem một bounding box được coi là dự đoán đúng hay không.

Confusion Matrix:

- Confusion matrix cung cấp một cái nhìn tổng quan về sự phân phối giữa các dự đoán đúng và sai.

Visual Inspection:

- Kiểm tra trực quan bằng cách xem xét các hình ảnh được dự đoán và bounding box tương ứng. Điều này giúp hiểu rõ hơn về các lỗi cụ thể mà mô hình có thể gặp.

Các phương pháp trên thường được sử dụng kết hợp để cung cấp cái nhìn toàn diện về hiệu suất của mô hình RCNN trên tập dữ liệu kiểm thử.

4.5 Đề xuất phương pháp cải tiến và đánh giá

4.5.1 Đề xuất phương pháp cải tiến

Có nhiều phương pháp cải tiến đã được đề xuất để nâng cao hiệu suất của mô hình Faster RCNN và giải quyết những thách thức trong lĩnh vực nhận diện đối tượng. Dưới đây là một số phương pháp cải tiến phổ biến:

Cải tiến Faster R-CNN:

- Backbone Network: Thử nghiệm với các kiến trúc backbone khác nhau như ResNet, ResNeXt, hoặc EfficientNet để xem xét cách chúng ảnh hưởng đến hiệu suất.
- Anchor Sizes và Aspect Ratios: Tối ưu hóa kích thước và tỷ lệ khía cạnh của các anchor để phù hợp với đặc trưng của tập dữ liệu cụ thể.
- Tích hợp FPN (Feature Pyramid Network): Sử dụng một mô hình FPN để tận dụng thông tin từ các tầng đặc trưng khác nhau và cải thiện khả năng đối mặt với vấn đề về tỷ lệ và quy mô.
- Attention Mechanisms: Sử dụng các cơ chế chú ý (attention mechanisms) như SE-Net hoặc CBAM để tăng cường khả năng của mô hình chú ý vào các đối tượng quan trọng.
- Hard Negative Mining: Sử dụng kỹ thuật hard negative mining để tập trung vào việc đối phó với các mẫu khó khăn, có thể cải thiện khả năng tổng quát hóa.

Đánh giá và Tối ưu hóa:

- Hàm Mất Mát Tùy Chỉnh: Điều chỉnh hàm mất mát để tối ưu hóa cho mục tiêu cụ thể hoặc đối mặt với vấn đề cụ thể của tập dữ liệu.
- Tăng kích thước tập dữ liệu: Thu thập thêm dữ liệu hoặc tăng cường dữ liệu để tăng khả năng tổng quát hóa của mô hình.
- Optical Loss Weights: Điều chỉnh trọng số của các thành phần mất mát như localization loss và classification loss để cân bằng đối với nhau.
- Học chập: Áp dụng kỹ thuật học chập (ensemble learning) bằng cách kết hợp nhiều mô hình để cải thiện độ chính xác.
- Kiểm thử chéo (Cross-validation): Sử dụng kiểm thử chéo để đánh giá hiệu suất của mô hình trên các tập dữ liệu không nhìn thấy trước.
- Chọn mô hình tốt nhất: Lưu lại trọng số của mô hình có hiệu suất tốt nhất trên tập xác nhận.

Sự hiệu quả của các phương pháp này có thể phụ thuộc vào bản chất cụ thể của dữ liệu và yêu cầu cụ thể của ứng dụng. Thực hiện các thử nghiệm và đánh giá cẩn thận để đảm bảo sự phù hợp và hiệu quả của các cải tiến và đánh giá được đề xuất.

4.5.2 Đánh giá phương pháp cải tiến

Việc đánh giá mô hình hoặc phương pháp cải tiến cụ thể yêu cầu trải qua quá trình chi tiết, sử dụng dữ liệu và điều kiện thử nghiệm cụ thể. Dưới đây là một đánh giá tổng quan, và cần điều chỉnh nó để phản ánh đầy đủ và chính xác nhất cho trường hợp cụ thể của:

Đánh Giá Tổng Quan:

Hiệu Suất Tổng Thể:

- Ưu Điểm: Mô hình đã đạt được độ chính xác và mAP (Mean Average Precision) cao trên tập dữ liệu kiểm thử, chỉ ra khả năng nhận diện và phân loại đối tượng tốt.

- Khía Cạnh Cần Cải Thiện: Có thể xem xét các phương pháp để cải thiện khả năng tổng quát hóa và xử lý các đối tượng nhỏ hoặc bị che phủ.

Tốc Độ Xử Lý:

- Ưu Điểm: Mô hình có tốc độ xử lý tốt, đặc biệt là trong các ứng dụng đòi hỏi xử lý thời gian thực.

- Khía Cạnh Cần Cải Thiện: Nếu có, nghiên cứu cách để tối ưu hóa tốc độ xử lý mà không làm giảm đi độ chính xác.

Khả Năng Tổng Quát Hóa:

- Ưu Điểm: Mô hình hiệu quả trên nhiều loại dữ liệu và không bị quá mức phụ thuộc vào tập dữ liệu huấn luyện.

- Khía Cạnh Cần Cải Thiện: Đảm bảo khả năng tổng quát hóa đối với dữ liệu mới và độ đa dạng của nó.

Xử Lý Đối Tượng Nhỏ và Che Phủ:

- Ưu Điểm: Mô hình có khả năng đối phó tốt với đối tượng nhỏ và đối tượng bị che phủ.

- Khía Cạnh Cần Cải Thiện: Nếu còn, nghiên cứu cách để cải thiện hiệu suất đối với những trường hợp khó khăn này.

Tính Ổn Định và Tiêu Tồn Tài Nguyên:

- Ưu Điểm: Mô hình ổn định và không tiêu tốn quá nhiều tài nguyên hệ thống.

- Khía Cạnh Cần Cải Thiện: Tối ưu hóa tài nguyên nếu có thể và kiểm tra tính ổn định trong các điều kiện thử nghiệm khác nhau.

Khả Năng Chịu Lỗi và Thích Ứng:

- Ưu Điểm: Mô hình khả năng chịu lỗi và thích ứng tốt với biến động trong môi trường.

- Khía Cạnh Cần Cải Thiện: Xem xét cách cải thiện khả năng chịu lỗi và sự thích ứng, đặc biệt là trong điều kiện đặc biệt.

Tính Tiện Lợi và Dễ Triển Khai:

- Ưu Điểm: Mô hình có tính tiện lợi và dễ triển khai trong các ứng dụng thực tế.

- Khía Cạnh Cần Cải Thiện: Đảm bảo tích hợp và triển khai mô hình một cách thuận tiện và hiệu quả.

4.5.3 Đánh giá phương pháp cải tiến

Việc đánh giá mô hình hoặc phương pháp cải tiến cụ thể yêu cầu trải qua quá trình chi tiết, sử dụng dữ liệu và điều kiện thử nghiệm cụ thể. Dưới đây là một đánh giá tổng quan, và cần điều chỉnh nó để phản ánh đầy đủ và chính xác nhất cho trường hợp cụ thể của:

Đánh Giá Tổng Quan:

Hiệu Suất Tổng Thể:

- Ưu Điểm: Mô hình đã đạt được độ chính xác và mAP (Mean Average Precision) cao trên tập dữ liệu kiểm thử, chỉ ra khả năng nhận diện và phân loại đối tượng tốt.

- Khía Cạnh Cần Cải Thiện: Có thể xem xét các phương pháp để cải thiện khả năng tổng quát hóa và xử lý các đối tượng nhỏ hoặc bị che phủ.

Tốc Độ Xử Lý:

- Ưu Điểm: Mô hình có tốc độ xử lý tốt, đặc biệt là trong các ứng dụng đòi hỏi xử lý thời gian thực.

- Khía Cạnh Cần Cải Thiện: Nếu có, nghiên cứu cách để tối ưu hóa tốc độ xử lý mà không làm giảm đi độ chính xác.

Khả Năng Tổng Quát Hóa:

- Ưu Điểm: Mô hình hiệu quả trên nhiều loại dữ liệu và không bị quá mức phụ thuộc vào tập dữ liệu huấn luyện.

- Khía Cạnh Cần Cải Thiện: Đảm bảo khả năng tổng quát hóa đối với dữ liệu mới và độ đa dạng của nó.

Xử Lý Đối Tượng Nhỏ và Che Phủ:

- Ưu Điểm: Mô hình có khả năng đối phó tốt với đối tượng nhỏ và đối tượng bị che phủ.

- Khía Cạnh Cần Cải Thiện: Nếu còn, nghiên cứu cách để cải thiện hiệu suất đối với những trường hợp khó khăn này.

Tính Ổn Định và Tiêu Tốn Tài Nguyên:

- Ưu Điểm: Mô hình ổn định và không tiêu tốn quá nhiều tài nguyên hệ thống.

- Khía Cạnh Cần Cải Thiện: Tối ưu hóa tài nguyên nếu có thể và kiểm tra tính ổn định trong các điều kiện thử nghiệm khác nhau.

Khả Năng Chịu Lỗi và Thích Ứng:

- Ưu Điểm: Mô hình khả năng chịu lỗi và thích ứng tốt với biến động trong môi trường.
- Khía Cạnh Cần Cải Thiện: Xem xét cách cải thiện khả năng chịu lỗi và sự thích ứng, đặc biệt là trong điều kiện đặc biệt.

Tính Tiện Lợi và Dễ Triển Khai:

- Ưu Điểm: Mô hình có tính tiện lợi và dễ triển khai trong các ứng dụng thực tế.
- Khía Cạnh Cần Cải Thiện: Đảm bảo tích hợp và triển khai mô hình một cách thuận tiện và hiệu quả.

4.6 Kết luận chương.

Trong chương này, chúng em đã tập trung vào quá trình nghiên cứu và phát triển mô hình Faster R-CNN để nhận diện phương tiện giao thông trong thành phố Hồ Chí Minh. Quá trình này bắt đầu từ việc thu thập dữ liệu, một bước cực kỳ quan trọng để đảm bảo tính đa dạng và độ biểu diễn của tập dữ liệu. Chúng em đã mô tả quá trình xử lý dữ liệu, gồm cả chuẩn bị và tiền xử lý, để chuẩn bị dữ liệu cho quá trình huấn luyện.

Chúng em đã giới thiệu mô hình Faster R-CNN và các thành phần chính của nó, từ backbone network đến region proposal network và head network. Quá trình huấn luyện đã được trình bày, bao gồm cả lựa chọn hàm mất mát và tối ưu hóa. Để đảm bảo độ chính xác và tổng quát hóa, chúng em đã áp dụng các phương pháp tăng cường dữ liệu và kiểm thử chéo.

Chúng em đã giải thích cải tiến và đánh giá mô hình Faster R-CNN. Chúng em đã đề xuất một loạt các cải tiến, từ backbone network đến các kỹ thuật huấn luyện và đánh giá. Những cải tiến này nhằm mục tiêu cải thiện độ chính xác, độ phức tạp và khả năng tổng quát hóa của mô hình.

Chúng em có thực hiện việc phát triển một ứng dụng giao diện người dùng thân thiện, nơi người dùng có thể tải lên hoặc chụp ảnh, sau đó mô hình Faster R-CNN sẽ tự động nhận diện và đánh dấu các phương tiện giao thông trên ảnh.

Với việc tích hợp kết quả từ mô hình Faster R-CNN vào giao diện người dùng, chúng ta có thể đạt được sự thuận tiện và linh hoạt trong việc quản lý thông tin về phương tiện giao thông, từ đó đóng góp vào việc cải thiện quản lý và an toàn giao thông trong thành phố.

KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN

Trong suốt quá trình nghiên cứu và thực hiện đề tài "Nhận Diện Phương Tiện Giao Thông TPHCM Bằng Mô Hình RCNN", chúng em đã tiến hành một loạt các bước cụ thể để đạt được mục tiêu nhận diện các phương tiện giao thông. Dưới đây là tổng kết quá trình này:

Xác Định Mục Tiêu và Phạm Vi:

- Chúng em đã đặt ra mục tiêu xây dựng một hệ thống nhận diện phương tiện giao thông được thu thập từ camera giao thông thành phố Hồ Chí Minh.
- Phạm vi của đề tài tập trung vào việc sử dụng mô hình RCNN để nhận diện và theo dõi các loại phương tiện khác nhau.

Thu Thập và Chuẩn Bị Dữ Liệu:

- Dữ liệu hình ảnh chứa các phương tiện giao thông đã được thu thập và chuẩn bị, kèm theo các nhãn bounding box tương ứng.
- Lựa Chọn và Huấn Luyện Mô Hình:
- Chúng tôi đã lựa chọn mô hình Faster RCNN làm nền tảng chính, sau đó huấn luyện mô hình trên dữ liệu đã chuẩn bị.
- Quá trình huấn luyện đã được điều chỉnh để đạt được hiệu suất tối ưu trên nhiều loại phương tiện.

Đánh Giá và Tối Ưu Hóa:

- Mô hình đã được đánh giá thông qua các phương tiện kiểm tra và tối ưu hóa để đảm bảo khả năng nhận diện chính xác và hiệu suất cao.

Xây Dựng Giao Diện Người Dùng:

- Để tăng cường sự ứng dụng của mô hình, chúng tôi đề xuất việc xây dựng giao diện người dùng thân thiện, cho phép người dùng tương tác với kết quả nhận diện.

Tích Hợp và Triển Khai:

- Kết quả từ mô hình RCNN và giao diện người dùng có thể tích hợp và triển khai để sử dụng trong môi trường thực tế.

Tổng Hợp Kiến Thức và Kinh Nghiệm:

- Quá trình này không chỉ mang lại kết quả nghiên cứu mà còn đóng góp vào việc tích lũy kiến thức và kinh nghiệm trong lĩnh vực nhận diện phương tiện giao thông và ứng dụng mô hình RCNN.

Hướng phát triển

Tìm Hiểu Thêm Về Kiến Trúc RCNN và Các Biến Thể:

- Nghiên cứu sâu hơn về các biến thể của kiến trúc RCNN như Fast R-CNN, Faster R-CNN, hoặc Mask R-CNN để hiểu rõ hơn về sự phát triển và ưu điểm của từng phiên bản.

Tối Ưu Hóa và Hiệu Suất:

- Tối ưu hóa mô hình RCNN để đảm bảo hiệu suất cao và khả năng ứng dụng trong thời gian thực.
- Nghiên cứu về các kỹ thuật tăng tốc mô hình như quantization, pruning, và hạn chế tài nguyên để đảm bảo tính thực tế và áp dụng trong môi trường thực tế.

Xử Lý Ảnh và Dữ Liệu:

- Nghiên cứu về các kỹ thuật xử lý ảnh và làm sạch dữ liệu để cải thiện độ chính xác của mô hình.
- Tích hợp dữ liệu từ nhiều nguồn khác nhau để mô phỏng đa dạng hơn về điều kiện giao thông và phương tiện.

Phát triển Ứng Dụng Thực Tế:

- Xây dựng một ứng dụng thực tế hoặc hệ thống giám sát giao thông dựa trên mô hình RCNN tại TP.HCM.
- Tích hợp tính năng nhận diện và theo dõi phương tiện để cung cấp thông tin thời gian thực về lưu lượng giao thông.

Đánh Giá Hiệu Quả và So Sánh Mô Hình:

- Thực hiện các bài kiểm tra và đánh giá hiệu quả của mô hình RCNN so với các phương pháp truyền thống hoặc các mô hình khác trong lĩnh vực nhận diện phương tiện giao thông.

Tích Hợp Công Nghệ Giao Thông Thông Minh:

- Nghiên cứu về cách tích hợp mô hình RCNN vào hệ thống giao thông thông minh để cải thiện quy trình quản lý và giảm ùn tắc giao thông.

Hạn chế

Quy Mô Dữ Liệu Đào Tạo:

- Độ chính xác của mô hình thường phụ thuộc lớn vào sự đa dạng và quy mô của dữ liệu đào tạo. Nếu dữ liệu không đủ đa dạng hoặc không đại diện cho nhiều điều kiện giao thông, mô hình có thể không chính xác trong các tình huống thực tế.

Độ Phức Tạp Của Môi Trường:

- Môi trường giao thông đô thị thường phức tạp với nhiều yếu tố như ánh sáng yếu, đám mây, và đối tượng di chuyển đa dạng. Những điều kiện này có thể làm giảm độ chính xác của mô hình trong việc nhận diện và phân loại phương tiện.

Dữ Liệu Gán Nhãn:

- Việc gán nhãn dữ liệu đòi hỏi sự công phu và chính xác, và đôi khi có thể gặp khó khăn đối với các trường hợp đặc biệt hoặc trường hợp biên. Sự chủ quan trong quá trình gán nhãn có thể dẫn đến mô hình học sai.

Độ Chính Xác Tính Năng:

- Các biến thể của RCNN có thể không hiệu quả trong việc nhận diện đối tượng nhỏ hoặc mờ do độ phân giải thấp hoặc sự xa cách về không gian.

Tổng kết, đề tài không chỉ hướng tới việc giải quyết một vấn đề cụ thể mà còn mở ra những cơ hội và thách thức trong lĩnh vực này. Chúng em mong muốn rằng nó sẽ thúc đẩy sự quan tâm và tiến triển trong lĩnh vực này, cũng như truyền cảm hứng cho các sinh viên khác tham gia vào những hướng nghiên cứu tương tự.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Đào Huy Thạch. (2020). Nhận biết phương tiện trong hệ thống iot cho giao thông. Luận văn (thạc sĩ). Khoa kỹ thuật viễn thông. Trường đại học công nghệ thông tin ĐHQG TPHCM.

[2] Trinh Man Hoang. (2018). TÁI NHẬN DẠNG PHƯƠNG TIỆN GIAO THÔNG SỬ DỤNG MẠNG KẾT HỢP CÁC ĐẶC TRƯNG HỌC SÂU. Đề tài (đại học). Ngành truyền thông đa phương tiện. Trường đại học bách khoa Hà Nội.

[3] Bùi Trần Tiến. (2019). Nhận diện phương tiện giao thông sử dụng kỹ thuật học sâu. Luận văn (thạc sĩ). Ngành khoa học máy tính. Học viện công nghệ bưu chính viễn thông.

[4] Nguyễn Văn Căn. (2015). Phát triển và phân loại phương tiện từ dữ liệu giao thông. Luận án (tiến sĩ). Ngành cơ sở toán học cho tin học. Viện khoa học và công nghệ quân sự.

Website

[4] Hu-sheng Fang (2020). Multi-scale traffic vehicle detection based on faster R-CNN with NAS optimization and feature enrichment [online], , viewed 13 December 2023, from:<
[https://fr.scribd.com/document/484472143/Report- v%E1%BB%81-Faster-RCNN](https://fr.scribd.com/document/484472143/Report-v%E1%BB%81-Faster-RCNN)>.

[5] Asif Ahmed (2023). Faster RCNN based robust vehicle detection algorithm for identifying and classifying vehicles [online], , viewed 15 December 2023, from:<

https://www.researchgate.net/publication/372761641_Faster_RCNN_based_robot_vehicle_detection_algorithm_for_identifying_and_classifying_vehicles
>.

[6] Lê Mạnh (2020). Report về Faster RCNN [online], , viewed 09 December 2023, from:< <https://fr.scribd.com/document/484472143/Report-v%E1%BB%81-Faster-RCNN>>.