

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐỒ ÁN TỐT NGHIỆP

KỸ THUẬT THÍCH ỨNG MIỀN DỮ LIỆU TRONG BÀI
TOÁN PHÂN VÙNG NGỮ NGHĨA ẢNH ĐƯỜNG PHỐ
ỨNG DỤNG CHO XE TỰ LÁI

NGUYỄN TIẾN TÀI

tai.nt164837@sis.hust.edu.vn

Ngành: Công nghệ thông tin

Chương trình Kỹ sư Tài năng Công nghệ Thông tin

Giảng viên hướng dẫn : TS. Đinh Viết Sang

Chữ ký của GVHD

Bộ môn

: Khoa học máy tính

Viện

: Công nghệ thông tin và Truyền thông

Hà Nội, 06/2021

Phiếu giao nhiệm vụ đồ án tốt nghiệp

Thông tin sinh viên

- Họ và tên: Nguyễn Tiến Tài
- Email: tai.nt164837@sis.hust.edu.vn
- Lớp: KSTN.CNTT K61
- Hệ đào tạo: Chương trình Kỹ sư tài năng Công nghệ thông tin
- Số điện thoại: 0941439925
- Đồ án này được viết tại: Trường Đại học Bách Khoa Hà Nội
- Đồ án này được viết từ: 22/01/2021

Mục đích nội dung của đồ án tốt nghiệp

- Thiết kế và cài đặt thuật toán thích ứng miền phân vùng ảnh ngữ nghĩa cho xe tự lái nhận diện ảnh đường phố.
- Sử dụng các giải thuật phù hợp, cải tiến và đánh giá hiệu năng.

Các nhiệm vụ cụ thể của đồ án tốt nghiệp

- Nghiên cứu các cấu trúc mạng phân vùng ảnh ngữ nghĩa dựa trên mạng nơ-ron tích chập.
- Nghiên cứu các kỹ thuật thích ứng miền cho dữ liệu ảnh đường phố.
- Cài đặt một framework có thể thay đổi và thử nghiệm các cấu trúc khác nhau một cách dễ dàng.
- So sánh đánh giá kết quả trên tập dữ liệu Cityscapes.
- Kết luận và lên kế hoạch phát triển trong tương lai.

Lời cam đoan của sinh viên

I - Nguyễn Tiến Tài - tất cả các kết quả và cài đặt được trình bày là công sức nghiên cứu dưới sự hướng dẫn của TS. Dinh Viết Sang.

*Hà Nội, 18 tháng Sáu, 2021
Tác giả*

Nguyễn Tiến Tài

Chứng nhận của giảng viên hướng dẫn về việc hoàn thành các yêu cầu đối với đồ án:

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

*Hà Nội, 18 tháng Sáu, 2021
Giảng viên hướng dẫn*

TS. Đinh Viết Sang

Lời cảm ơn

Đồ án này sẽ không thể hoàn thành nếu không có giảng viên hướng dẫn của tôi, Tiến sỹ Dinh Việt Sang. Thầy đã hướng dẫn tôi không chỉ bằng những lời khuyên và hiểu biết sâu sắc, mà còn bởi sự quyết tâm và niềm đam mê. Tôi đã học được rất nhiều sau khi hoàn thành đồ án này, và tôi tự hào khi là một trong những sinh viên của thầy.

Tôi cũng xin được gửi lời cảm ơn chân thành tới các thầy cô của tôi tại Đại học Bách Khoa Hà Nội, những người đã dạy cho tôi những kiến thức bổ ích trong suốt 5 năm qua khi tôi là một sinh viên trên giảng đường.

Xin gửi lời cảm ơn tới các đồng nghiệp và thầy cô ở VINIF, những người đã đóng góp thời gian, kiến thức và kinh nghiệm để giúp tôi hoàn thành đồ án này.

Cuối cùng, tôi cũng muốn gửi lời cảm ơn đến gia đình, bạn bè và những người tôi yêu quý, những người đã hỗ trợ và cho tôi niềm cảm hứng trong suốt 4 tháng qua. Tôi cảm thấy thật may mắn khi có những người tuyệt vời như vậy bên cạnh mình.

Tóm tắt đồ án

Những năm gần đây, cùng với sự gia tăng dữ liệu và sự phát triển của trí tuệ nhân tạo, đã có rất nhiều đột phá mạnh mẽ trong lĩnh vực thị giác máy tính và được áp dụng vào rất nhiều bài toán thực tế. Huấn luyện các mạng học sâu một cách hiệu quả là một chủ đề rất được quan tâm, đóng vai trò quan trọng trong việc thử nghiệm cũng như ứng dụng. Trong số đó, việc áp dụng các kết quả vào cho xe tự lái là một trong những ứng dụng có tính thực tiễn nhất đối với thế giới ngày càng số hóa như hiện nay. Đồ án này tập trung nghiên cứu và áp dụng các giải pháp nhằm tăng khả năng phân vùng vật thể của mô hình cho xe tự lái.

Nội dung đồ án được chia thành 5 chương như sau:

- Chương 1: Tổng quan bài toán và đặt vấn đề.
- Chương 2: Cơ sở lý thuyết và các nghiên cứu liên quan.
- Chương 3: Phương pháp sử dụng
- Chương 4: Thử nghiệm và đánh giá.
- Chương 5: Kết luận và hướng phát triển

Mục lục

Danh sách hình vẽ	1
Danh sách bảng	4
Danh mục từ viết tắt	5
1 Tổng quan bài toán và đặt vấn đề	6
1.1 Giới thiệu bài toán	6
1.2 Phân vùng ngữ nghĩa	7
1.3 Thích ứng miền dữ liệu trong bài toán phân vùng ngữ nghĩa . . .	8
1.4 Đóng góp của đồ án	9
2 Cơ sở lý thuyết và các nghiên cứu liên quan	10
2.1 Học máy	10
2.1.1 Phân loại học máy	11
2.1.2 Đánh đổi giữa độ thiên lệch và phương sai	12
2.1.3 Phân vùng các tập dữ liệu	13
2.1.4 Các độ đo đánh giá	14
2.2 Giới thiệu về mạng nơ-ron và học sâu	15
2.2.1 Các mạng nơ-ron nhân tạo và mạng kết nối đầy đủ	15
2.2.2 Mạng nơ-ron tích chập	20

2.2.3	Hàm mục tiêu và giải thuật huấn luyện	23
3	Phương pháp sử dụng	26
3.1	Kiến trúc tổng thể	26
3.2	Bộ trích xuất đặc trưng	27
3.3	Bộ giải mã	29
3.4	Học đối nghịch	31
3.4.1	Tổng quan về mạng GAN	31
3.4.2	Hàm mất mát sử dụng	32
3.4.3	Kiến trúc bộ phân biệt	32
3.5	Học chắt lọc	33
3.5.1	Chắt lọc kiến thức	33
3.5.2	Tự chắt lọc	34
4	Thử nghiệm và đánh giá	36
4.1	Dữ liệu	36
4.2	Tiền xử lý dữ liệu	39
4.3	Thiết lập thử nghiệm	39
4.3.1	Huấn luyện mô hình cơ sở trên tập dữ liệu nguồn	40
4.3.2	Huấn luyện cạnh tranh trên tập dữ liệu trộn giữa tập nguồn và tập đích	42
4.3.3	Huấn luyện tự chắt lọc trên tập dữ liệu đích với nhãn giả được sinh từ giai đoạn 2	44
4.4	Kết quả thực nghiệm	44
4.4.1	Sau khi học trên tập nguồn	44
4.4.2	Sau khi học cạnh tranh	45
4.4.3	Sau khi học tự chắt lọc	45

5 Kết luận và hướng phát triển **48**

Tài liệu tham khảo **49**

Danh sách hình vẽ

1.1	Một ví dụ của phân vùng ngữ nghĩa ¹	7
1.2	Minh họa cách một mô hình phân vùng ngữ nghĩa cho kết quả ²	7
1.3	Sự khác nhau giữa 3 bài toán Phát hiện đối tượng, Phân vùng ngữ nghĩa và Phân vùng thực thể ³	8
1.4	Một ví dụ của thích ứng miền ⁴	8
2.1	Minh họa bias và variance) ⁵	12
2.2	Minh họa một ma trận hỗn loạn (Đúng tích cực: TP - true positive, Sai tích cực: FP - false positive, Sai tiêu cực: FN - false negative, Đúng tiêu cực: TN - true negative) ⁶	15
2.3	Architecture of a 4-layer neural network ⁷	17
2.4	Các hàm kích hoạt sigmoid và tanh ⁸	17
2.5	Dường đi nghiệm của linear regression với các learning rate khác nhau ⁹	19
2.6	An example convolution layer ¹⁰	21
2.7	Một ví dụ về maxpooling ¹¹	22
2.8	Minh họa cho cách dùng cross entropy ¹²	23
2.9	Phép nhân tương ứng từng điểm ảnh rồi cộng tổng kết quả thu được ¹³	24
2.10	Tổng các bình phương điểm ảnh định lượng cho tập A và tập B ¹⁴	25
2.11	Một hàm mất mát xác mềm được tính cho từng lớp riêng biệt và sau đó lấy trung bình cộng để thu được kết quả cuối cùng ¹⁵	25

3.1	Kiến trúc tổng thể của mô hình ¹⁶	27
3.2	Kiến trúc kết nối bỏ qua của Resnet ¹⁷	28
3.3	Kiến trúc khối cơ bản và khối cổ chai trong Resnet ¹⁸	28
3.4	Kiến trúc khối phần dư và khối dày đặc ¹⁹	29
3.5	Hình minh họa cho DenseNet, LogDenseNet, SparseNet và Harmonic DenseNet được đề xuất (HarDNet), trong đó mỗi lớp là một tích chập 3x3 ²⁰	29
3.6	Để phân loại các điểm ảnh trung tâm (màu cam), Atrous spatial pyramid pooling (ASPP) khai thác đặc trưng đa kích thước bằng cách sử dụng nhiều bộ lọc song song với các tỷ lệ khác nhau. Trường nhìn hiệu quả được hiển thị bằng các màu khác nhau ²¹	30
3.7	Kiến trúc Hardnet68 được kết hợp với GALD để tăng khả năng khớp dữ liệu của mô hình ²²	30
3.8	Kiến trúc tổng quát của mạng GAN được minh họa như sự cạnh tranh giữa kẻ làm tiền giả và cảnh sát ²³	31
3.9	Từ trái sang phải lần lượt là nhãn nhị phân, nhãn cứng với chỉ số 0 1, nhãn mềm ²⁴	33
3.10	Bộ phân biệt có tác dụng nhằm giúp cho bộ mã hoá căn chỉnh các đặc trưng của 2 miền ảnh nhằm tìm ra những điểm chung nhất giữa 2 miền ²⁵	33
3.11	Mô hình tổng quát của học cô đọng kiến thức ²⁶	34
3.12	"Nhãn mềm" linh hoạt và cung cấp nhiều thông tin hơn "nhãn cứng" ²⁷	35
4.1	Dữ liệu trong GTA5 được trích xuất từ trò chơi điện tử	36
4.2	Một kiểu gán nhãn trong Cityscapes	37
4.3	Tiền xử lý dữ liệu giúp cho mô hình học được những thông tin đa dạng hơn	39
4.4	Theo thứ tự lần lượt là: Ảnh đầu vào, nhãn thật, kết quả sau giai đoạn 1, kết quả sau giai đoạn 2, kết quả sau giai đoạn 3	40

4.5	Bên trái là kết quả tốc độ học qua các iteration. Bên phải là kết quả hàm mất mát trong quá trình huấn luyện Resnet101 + DeeplabV2 trên tập nguồn. Được đánh giá trên từng iteration, trong đó ta sử dụng trung bình động với kích thước cửa sổ là 150	41
4.6	Bên trái là kết quả tốc độ học qua các iteration. Bên phải là kết quả hàm mất mát trong quá trình huấn luyện Hardnet68 + GALD trên tập nguồn. Được đánh giá trên từng iteration, trong đó ta sử dụng trung bình động với kích thước cửa sổ là 150	42
4.7	Bên trái là kết quả tốc độ học của mô hình cơ sở và bộ phân biệt qua các iteration. Bên phải là kết quả hàm mất mát trong quá trình huấn luyện cạnh tranh Resnet101 + DeeplabV2 trên tập nguồn trộn tập đích. Được đánh giá trên từng iteration, trong đó ta sử dụng trung bình động với kích thước cửa sổ là 150	43
4.8	Bên trái là kết quả tốc độ học của mô hình cơ sở và bộ phân biệt qua các iteration. Bên phải là kết quả hàm mất mát trong quá trình huấn luyện cạnh tranh Global Aggregate and Local Distribution trên tập nguồn trộn tập đích. Được đánh giá trên từng iteration, trong đó ta sử dụng trung bình động với kích thước cửa sổ là 150	44
4.9	Minh họa Confusion Matrix cho kết quả của mô hình Hardnet68 + GALD	46

Danh sách bảng

4.1	Dịnh nghĩa các lớp trong tập dữ liệu Cityscapes	37
4.2	Cấu hình trong bộ sinh ảnh Carla	38
4.3	Dịnh nghĩa các lớp trong tập dữ liệu được sinh từ Bộ mô phỏng Carla	38
4.4	So sánh kết quả trên mô hình Resnet101 + DeeplabV2 và mô hình Hardnet68 + GALD đánh giá trên fold thứ 10 của tập GTA5 và tập kiểm thử của Cityscapes với kích thước đầu vào là [1024, 512]	45
4.5	So sánh kết quả mô hình Resnet101 + DeeplabV2 và mô hình Hardnet68 + GALD đánh giá trên tập kiểm thử của Cityscapes với kích thước đầu vào là [1024, 512]	45
4.6	Bảng so sánh kết quả các mô hình trên 2 tập dữ liệu GTA5 và Cityscapes	46

Danh mục từ viết tắt

ASPP Atrous spatial pyramid pooling.

CNN Convolutional Neural Network.

GALD Global Aggregation Local Distribution.

SOTA State of the art.

SVM Support Vector Machine.

Chương 1

Tổng quan bài toán và đặt vấn đề

1.1 Giới thiệu bài toán

Bài toán mà chúng ta cần giải quyết ở đây như sau:

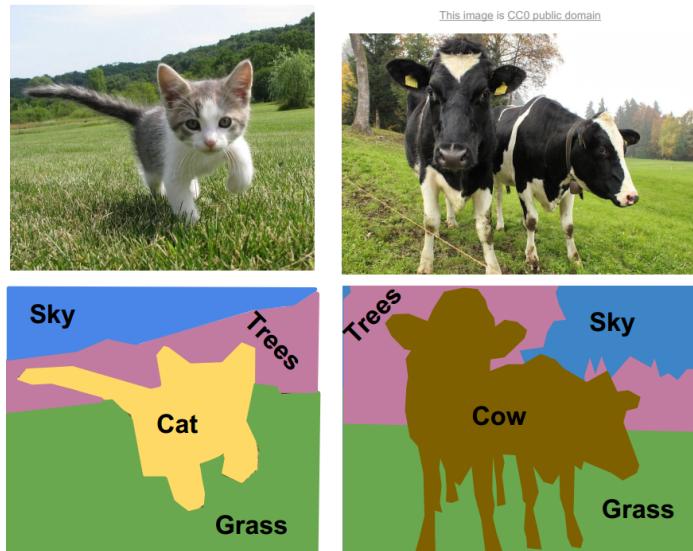
- Đầu vào: Ảnh chụp đường phố từ tập dữ liệu Cityscapes không nhãn trong lúc huấn luyện
- Đầu ra: Kết quả phân vùng ngữ nghĩa với 19 lớp bao gồm:
road-lòng đường; sidewalk-vỉa hè; building-toà nhà; wall-bức tường; fence-hàng rào
pole-cột; light-đèn; sign-biển báo; vegetation-cây cối; terrain-địa hình
sky-bầu trời; person-người; rider-người đạp xe; car-xe hơi; truck-xe tải
bus-xe buýt; train-đoàn tàu; motorcycle-xe máy; bicycle-xe đạp

Bài toán phân vùng ảnh ngữ nghĩa bán giám sát là một bài toán quan trọng, bao gồm nhiều bài toán con khác nhau như: bài toán phân vùng ngữ nghĩa, bài toán thích ứng miền, bài toán học cạnh tranh. Đây là một bài toán phức tạp và đòi hỏi độ chính xác cao bởi vì các thông tin trong ảnh đường phố khá đa dạng và phức tạp bao gồm nhiều lớp, nhiều thông tin cần xử lý. Hơn nữa, bài toán học bán giám sát còn khó khăn hơn khi mô hình không được tiếp cận trực tiếp với nhãn của ảnh mà phải thích nghi qua một tập dữ liệu khác gần tương tự mà ta sẽ trình bày ở phần sau.

1.2 Phân vùng ngữ nghĩa

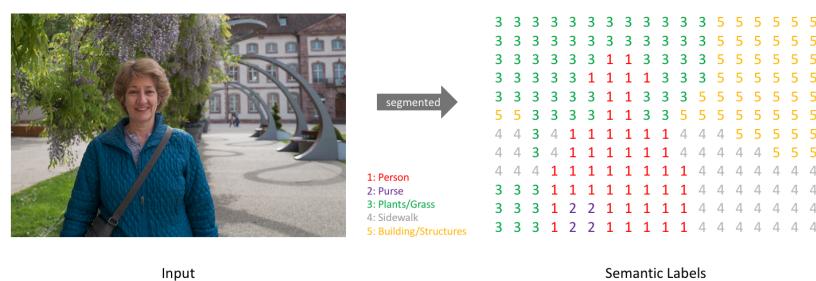
Phân vùng ngữ nghĩa là một nhiệm vụ nhằm phân loại từng điểm ảnh trong 1 ảnh từ một tập các lớp được định nghĩa trước. Hay đơn giản có thể hiểu đó là việc phân loại các đối tượng trong một ảnh vào các lớp đã định nghĩa trước đó.

Ví dụ như phân vùng ảnh để tìm con mèo hay con bò. Các lớp khác như bầu trời, cây, bã cỏ được thể hiện trong Hình 1.1.



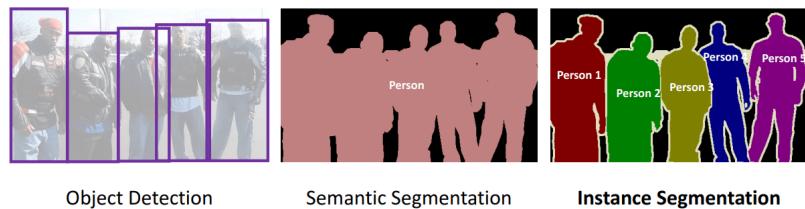
Hình 1.1: Một ví dụ của phân vùng ngữ nghĩa ¹

Mục tiêu của lớp bài toán này là từ ảnh đầu vào với $\text{shape} = (W, H, 3)$ để tạo ra ma trận $W \times H$ đầu ra với mỗi điểm ảnh sẽ là nhãn tương ứng được dự đoán. Ví dụ trong Hình 1.2 ta phân vùng các nhãn là "Người", "Cái ví", "Cây/cỏ", "Toà nhà/Công trình". Từng điểm ảnh sẽ được phân vào một trong các nhãn được minh họa như hình bên phải



Hình 1.2: Minh họa cách một mô hình phân vùng ngữ nghĩa cho kết quả²

Bài toán Phân vùng ngữ nghĩa khác với bài toán Phát hiện đối tượng ở chỗ nó sẽ không dự đoán "hình vuông bao quanh" đối tượng và cũng không phân biệt giữa các đối tượng khác nhau của một lớp (như với bài toán Phân vùng thực thể)



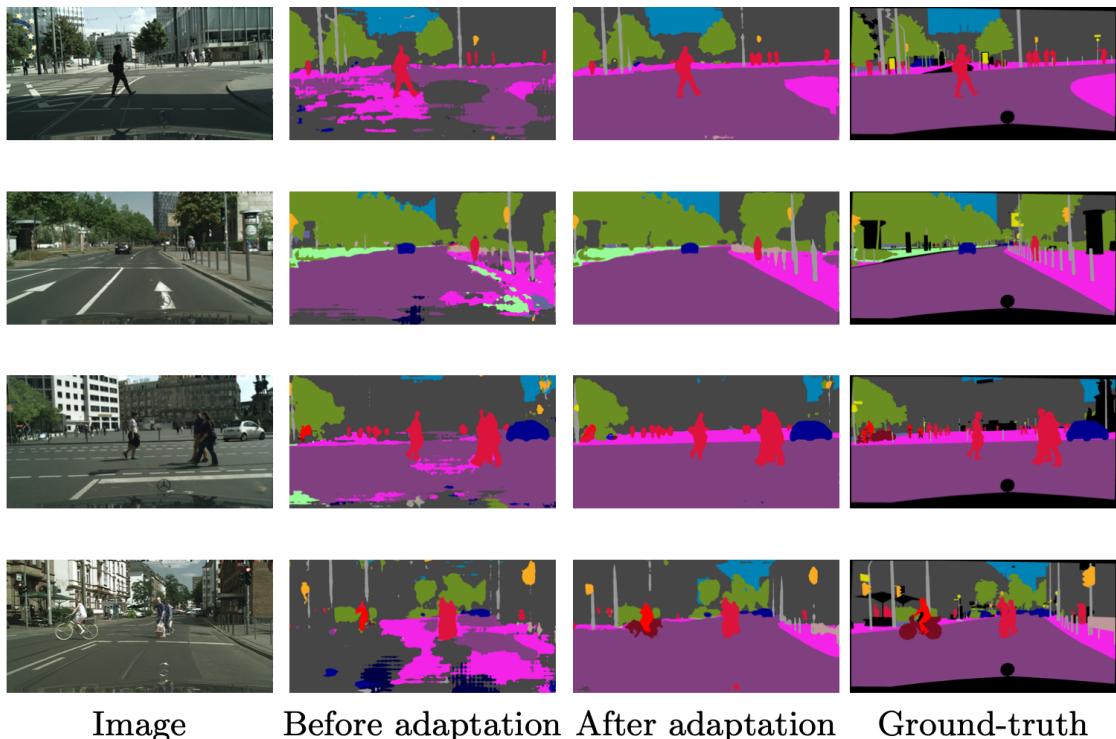
Hình 1.3: Sự khác nhau giữa 3 bài toán Phát hiện đối tượng, Phân vùng ngữ nghĩa và Phân vùng thực thể³

1.3 Thích ứng miền dữ liệu trong bài toán phân vùng ngữ nghĩa

Thành công của bài toán phân vùng ngữ nghĩa trong những năm gần đây chủ yếu được thúc đẩy bởi một lượng lớn dữ liệu được gán nhãn có thể truy cập được.

Tuy nhiên, thu thập một lượng lớn dữ liệu được gán nhãn cho việc huấn luyện là một tác vụ tốn kém. Gần đây, những tiến bộ trong đồ họa máy tính đã cung cấp thêm giải pháp thay thế cho việc gán nhãn thủ công đắt đỏ.

Một ví dụ được thể hiện trong Hình 1.4.



Hình 1.4: Một ví dụ của thích ứng miền⁴

Gần đây, các phương pháp huấn luyện cạnh tranh (adversarial training - AT)

đang nhận được nhiều sự quan tâm cho bài toán phân vùng ngữ nghĩa. Những phương pháp này nhằm tối ưu một chuỗi những hàm mục tiêu cạnh tranh để căn chỉnh các phân phối đặc trưng của tập đích và tập nguồn. Bên cạnh đó, một vài nghiên cứu khác tập trung vào xây dựng các mô hình dựa trên phương pháp tự huấn luyện (self-training - ST). Những phương pháp này đều lặp lại việc huấn luyện mô hình bằng cách sử dụng cả dữ liệu nguồn có nhãn và nhãn giả được sinh ra cho dữ liệu đích, từ đó đạt được sự cân bằng giữa tập nguồn và tập đích.

1.4 Đóng góp của đồ án

Trong kỷ nguyên công nghệ số như hiện nay, các sản phẩm mới, các kỹ thuật mới đang ngày ngày được sử dụng hoặc tích hợp, bổ sung cho các sản phẩm sẵn có trong sản xuất công nghiệp. Trong đó, lĩnh vực ô tô tự lái là một trong những mảnh đất màu mỡ cho các tập đoàn lớn và các nhà khoa học chinh phục. Có thể kể đến xe tự lái của Tesla (Hoa Kỳ) hay xa hơn có thể là sản phẩm xe tự lái của Vinfast (Việt Nam). Vậy làm sao để một chiếc xe có thể nhận dạng và phân tích các chướng ngại vật mà nó gặp phải trên đường ?

Ta thấy rằng điều kiện làm việc của một chiếc ô tô là rất đa dạng về thời tiết, khí hậu, phong cảnh, ánh sáng, ... Để mô hình có thể hoạt động tốt thì đòi hỏi dữ liệu cũng phải đa dạng phù hợp với yêu cầu thực tế. Trong khi đó, để chuẩn bị dữ liệu ảnh đường phố thật đa dạng như vậy sẽ tốn kém chi phí và thời gian. Do đó, tác giả đã sinh dữ liệu ảnh đường phố từ trò chơi điện tử. Bên cạnh đó, ta không thể chỉ huấn luyện dữ liệu ảnh từ trò chơi rồi áp dụng vào ảnh thực tế được. Vậy nên, tác giả cần sử dụng thêm các kỹ thuật học máy nhằm tận dụng tốt dữ liệu. Tóm lại, đồ án này đóng góp một hướng tiếp cận với vấn đề trên bằng 2 tác vụ bao gồm:

- Sinh ảnh đường phố với các điều kiện thời tiết, ánh sáng, xe cộ, con người, ... khác nhau để cung cấp dữ liệu cho mô hình.
- Áp dụng các kỹ thuật thích ứng miền và các mô hình cơ sở phù hợp với dữ liệu và yêu cầu bài toán nhằm tổng quát hoá khả năng nhận biết của mô hình.

Chương 2

Cơ sở lý thuyết và các nghiên cứu liên quan

2.1 Học máy

Học máy là một phần trong lĩnh vực Trí Tuệ Nhân Tạo, bao gồm cung cấp kiến thức cho máy tính thông qua dữ liệu, quan sát và tương tác với thế giới (Yoshua Bengio [1]). Các giải thuật Học máy sinh ra các "mô hình" từ dữ liệu trong suốt quá trình học hỏi của nó, và suy luận ra kết quả của mô hình trong suốt thời gian chạy nhằm thu được kết quả trên tập kiểm thử mà "mô hình" chưa nhìn thấy bao giờ.

Học máy hay học sâu là một lĩnh vực nằm trong lĩnh vực lớn hơn là Trí tuệ nhân tạo. Hầu hết các giải thuật học máy đều cố gắng giải quyết các tác vụ của con người, ví dụ như nhận diện thị giác, hay là đọc hiểu ngôn ngữ, vân vân,... Vì Học máy vốn là các thuật toán xấp xỉ, các bài toán đòi hỏi giải thuật Học máy thường thuộc lớp bài toán NP hoặc rất khó để tính toán.

Học máy có liên quan mật thiết đến khoa học thống kê. Một mô hình Học máy cung cấp các dự đoán dựa trên một mô hình thống kê của các dữ liệu huấn luyện. Mặc dù các mô hình tập trung vào việc tổng quát hóa những dữ liệu mới, chúng vẫn xoay quanh phân phối của tập dữ liệu huấn luyện. Hiểu rõ được các đặc điểm thống kê đối với mỗi bài toán là một bước quan trọng trong việc tạo ra các mô hình tốt.

Học máy cũng có mối quan hệ mật thiết với bài toán tối ưu: nhiều bài toán học máy được mô hình hóa như việc tối ưu hàm mất mát trên một tập dữ liệu huấn luyện. Hàm mất mát thể hiện sự khác biệt giữa kết quả dự đoán của mô hình với kết quả của dữ liệu thực tế.

Sự khác biệt chính giữa Học máy và các lĩnh vực đã đề cập ở trên nằm ở mục đích của chúng. Cả thông kê lẫn tối ưu đều tập trung vào việc trích xuất kết quả từ những dữ liệu huấn luyện đã cho, trong khi đó Học máy được nghiên cứu nhằm tổng quát hoá những dữ liệu chưa được nhìn thấy.

2.1.1 Phân loại học máy

Có nhiều loại giải thuật học máy, bao gồm: Học có giám sát, Học không giám sát, Học bán giám sát, và Học củng cố.

Các giải thuật Học có giám sát đều học từ một tập dữ liệu bao gồm cả đầu vào và đầu ra mong muốn (hay còn gọi là nhãn). Mô hình được huấn luyện bao gồm một hàm số nhằm ánh xạ bất kỳ một đầu vào cho trước nào với một đầu ra tương ứng, và nhằm vào việc xuất ra những kết quả đúng cho những dữ liệu chưa thấy và cả những dữ liệu đã thấy. Hai bài toán con của Học có giám sát là bài toán Hồi quy (với đầu ra là một giá trị liên tục) và phân lớp (với đầu ra là một hoặc nhiều giá trị rời rạc). Các giải thuật Học có giám sát phổ biến bao gồm Naive Bayes classification, Support Vector Machine (SVM), Decision Trees và Neural Networks. Đây là các thuật toán phổ biến nhất trong Học máy, được sử dụng trong các lĩnh vực như bài toán phân loại ảnh, phân tích cảm giác, hay nhận diện giọng nói, vân vân,...

Các giải thuật học bán giám sát cũng tương tự, tuy nhiên chúng được thiết kế để xử lý các dữ liệu không có nhãn hoặc các tập dữ liệu rất nhỏ, bằng việc tạo ra các giả thiết heuristic cho bài toán.

Mặt khác, các giải thuật không giám sát, xử lý các dữ liệu không nhãn, chỉ biết mỗi đầu vào. Thường thì mục đích của những mô hình này là học một mối quan hệ giữa các mẫu với nhau. Thay vì trả lời phản hồi, các thuật toán học tập không được giám sát xác định các điểm chung trong dữ liệu và phản ứng dựa trên sự vắng mặt của các điểm tương đồng như vậy trong mỗi phần dữ liệu mới. Các thuật toán không giám sát phổ biến bao gồm: K-means Clustering, và Neural Autoencoders. Một ứng dụng chính của thuật toán không giám sát là phân tích cụm, các thực thể (như người dùng hay sản phẩm, vân vân,...) có thể được phân loại thành các nhóm dựa trên các đặc trưng của chúng.

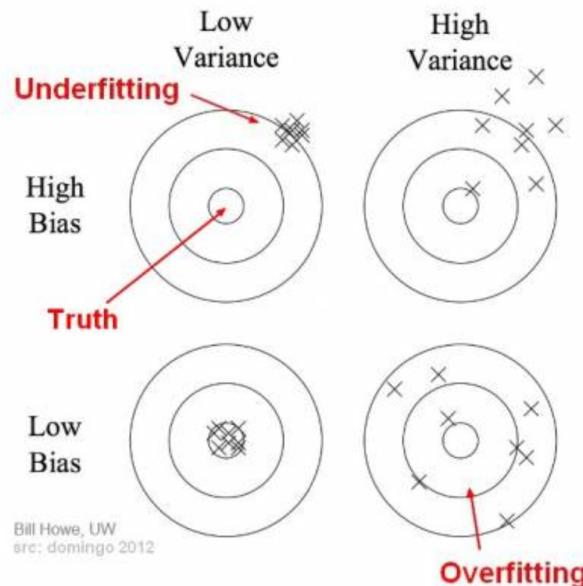
Cuối cùng, các thuật toán Học tăng cường được mô hình hóa như các tác nhân trong một môi trường. Môi trường cung cấp các phản hồi tới các hành động của tác nhân, được hiểu như những giá trị phần thưởng. Mục tiêu trong Học tăng cường là học một chính sách để các tác nhân thực hiện nhằm tối ưu hóa phần

thường của chúng, và hoàn thành nhiệm vụ của chúng. Các ứng dụng đáng chú ý của Học tăng cường gồm có xe tự lái và các phần mềm AI trong trò chơi (eg. Google's AlphaGo [21]).

2.1.2 Đánh đổi giữa độ thiên lệch và phương sai

Độ thiên lệch: nghĩa là độ lệch, biểu thị sự chênh lệch giữa giá trị trung bình mà mô hình dự đoán và giá trị thực tế của dữ liệu.

Phương sai: nghĩa là phương sai, biểu thị độ phân tán của các giá trị mà mô hình dự đoán so với giá trị thực tế.



Hình 2.1: Minh họa bias và variance)¹

Giá trị thật dữ liệu ở giữa tâm các đường tròn. Các dấu X là các giá trị dự đoán. Ta thấy nếu bias cao thì giá trị dự đoán rất xa tâm. Tuy nhiên nếu variance cao thì các giá trị dự đoán phân tán rộng dẫn đến việc xa giá trị thực tế. Vậy nên, ta mong muốn bias thấp và variance thấp.

Nếu chúng ta biểu thị biến số mà chúng ta đang cố gắng dự đoán là Y và các hiệp biến của chúng ta là X. Ta giả sử có một mối quan hệ tương quan ví dụ như $Y = f(X) + \epsilon$ trong đó ϵ là phân phối chuẩn với kỳ vọng là không, tức là

$$\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$$

Ta có thể ước lượng một mô hình $f(X)$ sử dụng hồi quy tuyến tính hoặc là một

¹<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>

kỹ thuật mô hình nào đó. Trong trường hợp này sai số bình phương kỳ vọng tại điểm x là

$$\text{Err}(x) = E[(Y - \hat{f}(x))^2]$$

Sai số này có thể được phân tách thành các thành phần bias và phuơng sai:

$$\begin{aligned}\text{Err}(x) &= (E[\hat{f}(x)] - f(x))^2 + E[(f(\hat{x}) - E[\hat{f}(x)])^2] + \sigma_e^2 \\ \text{Err}(x) &= \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}\end{aligned}$$

Ở đây, Irreducible Error, là thuật ngữ nhiều trong mỗi quan hệ thực sự mà về cơ bản không thể bị giảm bởi bất kỳ mô hình nào. Cho trước mô hình chính xác và dữ liệu không giới hạn, ta có thể giảm thiểu cả bias và phuơng sai về 0. Tuy nhiên thực tế thì với các mô hình chỉ gần đúng và dữ liệu hạn chế, ta cần phải đánh đổi giữa tối ưu bias hay tối ưu phuơng sai.

2.1.3 Phân vùng các tập dữ liệu

Đánh giá tính hiệu quả của các mô hình Học máy khác với các mô hình và giải thuật thống kê khác. Trong phần này, chúng ta tập trung vào đánh giá các mô hình có giám sát.

Các mô hình có giám sát được huấn luyện trên tập dữ liệu huấn luyện, tuy nhiên mục tiêu nhằm xuất ra các dự đoán chính xác trên tập dữ liệu chưa nhìn thấy bên ngoài tập huấn luyện. Vì vậy ta không thể đánh giá trên dữ liệu huấn luyện. Một khía cạnh khác của mô hình mà chỉ "học" bằng việc ghi nhớ những dữ liệu huấn luyện một cách hoàn hảo thì sẽ không thể sử dụng được. Vậy nên một tập con của dữ liệu được đánh nhãn sẽ được lấy ra từ tập huấn luyện và coi đó như một tập kiểm thử mới, chỉ phục vụ cho việc đánh giá mô hình.

Thêm vào đó, có nhiều mô hình với các siêu tham số mà ta buộc phải lựa chọn dựa trên kinh nghiệm và thực nghiệm. lựa chọn này cũng nên được thực hiện trên dữ liệu không nhìn thấy, nhưng không sử dụng tập hợp thử nghiệm vì điều này chỉ đơn giản là chọn ra bất kỳ tập thông số nào xảy ra phù hợp với phân phối thử nghiệm. Thay vào đó một tập dữ liệu thẩm định sẽ được tạo ra nhằm mục đích này.

Đối với các tập dữ liệu lớn, ta thường chia tập dữ liệu theo tỷ lệ 80/20 cho huấn luyện và kiểm thử. Trong những tập dữ liệu nhỏ hơn, một hướng tiếp cận thay thế được gọi là k -fold cross validation, khi mà ta chia tập dữ liệu thành k phần bằng nhau được gọi là các fold. Việc huấn luyện khi đó được thực hiện k lần,

mỗi lần sử dụng một fold như là tập kiểm thử, và tập còn lại như là tập huấn luyện. Các kết quả đánh giá khi đó được lấy trung bình trên tất cả các lần chạy.

Khi chia các tập dữ liệu, đôi khi việc duy trì phân phối của các lớp hay các thuộc tính khác rất quan trọng trong dữ liệu gốc. Kỹ thuật này được gọi là lấy mẫu phân tầng, và đặc biệt quan trọng đối với các tập dữ liệu được phân phối không đồng đều.

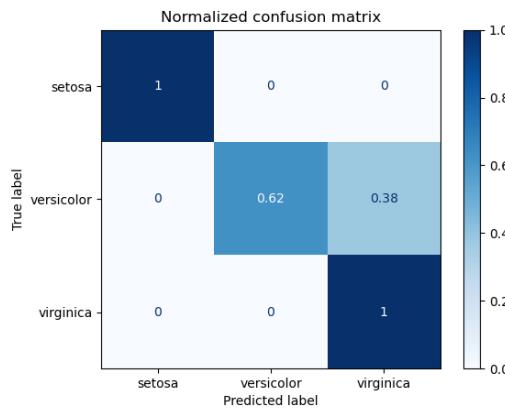
Đánh giá mô hình trên nhiều tập dữ liệu khác nhau có thể bộc lộ ra nhiều những thiếu sót khác nhau. Một mô hình gọi là *underfits* khi nó đạt điểm đánh giá thấp trên tập huấn luyện, điều đó có nghĩa là mô hình không khớp được với phân phối của dữ liệu bài toán đưa ra. Sử dụng những mô hình mạnh mẽ và phức tạp hơn có thể giải quyết được vấn đề này. Mặt khác, một mô hình gọi là *overfits* khi nó đạt được điểm đánh giá cao trên tập huấn luyện, nhưng lại đạt điểm thấp trên tập thẩm định. Điều này có nghĩa rằng mô hình chỉ học được các mẫu có trong tập huấn luyện mà không phải các đặc trưng ta mong muốn. Các kỹ thuật chuẩn hoá và tăng cường dữ liệu sẽ được sử dụng nhằm cải thiện vấn đề overfitting này.

2.1.4 Các độ đo đánh giá

Các metric khác nhau được dùng để đánh giá các mô hình học máy khác nhau, tùy thuộc vào từng lớp bài toán. Hầu hết các metric tập trung vào việc đánh giá xem mô hình thực thi các tác vụ đã cho chính xác đến mức nào, hoặc cũng có thể bao gồm các yêu cầu khác như tính hiệu quả, sức mạnh hay khả năng mở rộng.

Đối với các mô hình phân lớp, metric phổ biến nhất thường là để đo độ chính xác, tức là tỷ lệ giữa số mẫu được phân lớp đúng trên số mẫu mà mô hình đoán. Tuy nhiên metric này có thể bị nhiễu trên các tập dữ liệu không cân bằng. Đặc biệt, nếu như 90% dữ liệu thuộc vào lớp A, thì một mô hình phân loại luôn cho kết quả là lớp A sẽ đạt được 90% về độ chính xác. Vậy nên, 3 metric dưới đây thường xuyên được thêm vào để đánh giá mô hình bao gồm: precision, recall và F1 score.

Precision đo tỷ lệ giữa Đúng Tích Cực (số mẫu được mô hình dự đoán chính xác cho một lớp) trên tất cả dự đoán của lớp đó. Recall đo tỷ lệ giữa Đúng Tích Cực đối với một lớp và tất cả các mẫu được gán nhãn của lớp đó. Cuối cùng, F1 score là trung bình điều hòa giữa precision và recall. Tính toán những metric này thường sử dụng một ma trận hỗn loạn để biểu diễn (Figure 2.2).



Hình 2.2: Minh họa một ma trận hỗn loạn (Dúng tích cực: TP - true positive, Sai tích cực: FP - false positive, Sai tiêu cực: FN - false negative, Dúng tiêu cực: TN - true negative)²

Cụ thể ta có các công thức như dưới đây:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \quad (2.1)$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \quad (2.2)$$

$$F_1^i = 2 \frac{Precision_i \cdot Recall_i}{Precision_i + Recall_i} \quad (2.3)$$

Precision trừng phạt một mô hình khi nó đưa ra những dự đoán sai đối với một lớp cho trước, trong khi recall trừng phạt sự kém cỏi của mô hình nhằm a model's inability to bao gồm tất cả các mẫu của lớp đó. Một mô hình mà cố gắng "gian lận" trong một metric hay lớp A model that tries to "cheat" in one metric or class (like how our hypothetical "always A" model would have perfect recall for class A) thì sẽ nhận điểm thấp trên các metric khác (ví dụ một mô hình giả định "luôn là A" thì sẽ cho recall bằng 1 đối với lớp A, nhưng precision và recall của nó đối với các lớp khác A sẽ bằng 0), khi đó thì F1 score vẫn sẽ thấp.

2.2 Giới thiệu về mạng nơ-ron và học sâu

2.2.1 Các mạng nơ-ron nhân tạo và mạng kết nối đầy đủ

Các mạng nơ-ron nhân tạo (hay được gọi tắt là mạng nơ-ron) là một lớp của các mô hình học máy được lấy cảm hứng từ các hệ nơ-ron sinh học xử lý dữ liệu.

²<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>

Mạng nơ-ron sinh học bao gồm một nhóm hoặc nhiều nhóm nơ-ron được kết nối về mặt hóa học hoặc liên kết về mặt chức năng. Một nơ-ron có thể được kết nối đến nhiều nơ-ron khác, tổng số nơ-ron và kết nối trong một mạng có thể rất lớn. Các kết nối hay còn được gọi là các khớp thần kinh, thường được cấu tạo từ sợi trực đến đuôi gai.

Các mạng nơron nhân tạo sử dụng một quy trình gần đúng được đơn giản hóa hơn nhiều, có các nơ-ron được thay thế bởi các đơn vị tính toán đơn giản nhằm thực hiện biến đổi trên dữ liệu đầu vào. Mặc dù đơn giản như vậy, nhưng với số lượng nơ-ron và kết nối lớn cho phép các mạng nơ-ron thể hiện được những hàm số cực kỳ phức tạp.

Mạng nơ-ron đầu tiên được đề xuất vào năm 1958 bởi Frank Rosenblatt [19], được gọi là Perceptron. Một perceptron về cơ bản là một nơ-ron bao gồm một tập các trọng số w , một trọng số $bias$ b và một hàm kích hoạt $g(x)$. Cho một vector liên tục x là đầu vào, đầu ra của một perceptron được thể hiện bởi hàm sau đây:

$$f(x; w, b) = g(w^T x + b) \quad (2.4)$$

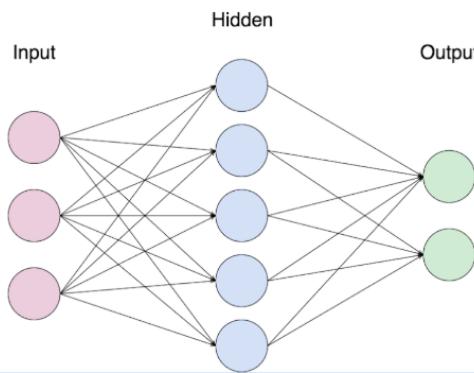
Ý tưởng của perceptron thậm chí đã được mở rộng thành perceptron đa tầng, hay còn được biết đến nhiều hơn với cái tên mạng nơ-ron nhân tạo. Nhìn chung, chúng chỉ khác ở chỗ có nhiều nơ-ron tính toán hơn thay vì chỉ có một, và chúng được chia thành các tầng khác nhau (Hình 2.3).

Có 3 loại tầng được thể hiện trong một mạng nơ-ron: tầng đầu vào, các tầng ẩn và tầng đầu ra. Tầng đầu vào biểu diễn cho dữ liệu đầu vào, mỗi đơn vị chứa giá trị đặc trưng của một mẫu. Các tầng ẩn bao gồm các biến đổi được thực hiện bởi mạng trên dữ liệu đầu vào, mỗi đơn vị làm việc tương tự như một perceptron. Cuối cùng là tầng đầu ra bao gồm các giá trị đầu ra của mạng. Cụ thể, tầng này trả về các giá trị mà ta mong muốn như là kết quả cuối cùng, hoặc là các giá trị mà có thể suy ra được kết quả cuối cùng.

Trong kiến trúc này, mỗi nơ-ron trong tầng thứ j được kết nối với tất cả các nơ-ron ở tầng thứ $j + 1$. Như vậy, các mạng này còn được gọi là mạng được kết nối đầy đủ.

Các mạng nơ-ron cài đặt 2 phương thức chính: Lan truyền tiến và Lan truyền ngược.

³<https://technology.condenast.com/story/a-neural-network-primer>



Hình 2.3: Architecture of a 4-layer neural network³

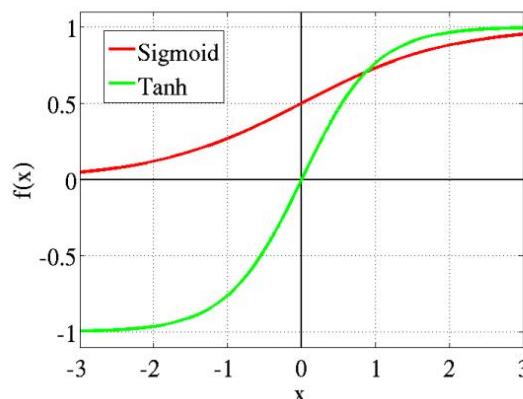
Lan truyền tiến

Lan truyền tiến được sử dụng để lấy các kết quả đầu ra từ một mạng nơ-ron. Ví dụ với đầu vào x ở tầng đầu vào, chúng ta lặp qua lần lượt từng tầng một, tính toán đầu ra ở mỗi bước lan truyền (Giải thuật 1). Đầu ra cho tầng thứ j được định nghĩa ở [8] as:

$$h^{(j)} = g^{(j)}(W^{(j)T} \cdot h^{(j-1)} + b^{(j)}) \quad (2.5)$$

Trong đó $g^{(j)}$ là hàm kích hoạt của tầng thứ j^{th} , $W^{(j)}$ là ma trận trọng số, $b^{(j)}$ là trọng số lệch, và $h^{(0)} = x$. Kích thước của $W^{(j)}$ tương ứng với số nơ-ron ở các tầng thứ j và thứ $j - 1$.

Đối với các mạng nơ-ron được trình bày phức tạp hơn, các mô hình phi tuyến tính, hàm kích hoạt tại mỗi tầng thường là phi tuyến. Các hàm được sử dụng rộng rãi nhất là *sigmoid* và *tanh* (Hình 2.4).



Hình 2.4: Các hàm kích hoạt sigmoid và tanh⁴

⁴https://www.researchgate.net/figure/The-sigmoid-and-hyperbolic-tangent-activation-functions_fig2_2654867842

Algorithm 1: Lan truyền tiến

```

Input : Vector các đặc trưng đầu vào  $x$ 
        Các tầng mạng  $L = ((W^{(1)}, b^{(1)}, g^{(1)}), \dots, (W^{(m)}, b^{(m)}, g^{(m)}))$ 
Output : Các dự đoán  $h^{(m)}$ 
        Đầu vào tại mỗi tầng  $Z = (z^{(1)}, \dots, z^{(m)})$ 

1  $h^{(0)} \leftarrow x$ 
2 for  $j \in (1 \dots m)$  do
3    $| z^{(j)} \leftarrow W^{(j)T} \cdot h^{(j-1)} + b^{(j)}$ 
4    $| h^{(j)} \leftarrow g^{(j)}(z^{(j)})$ 
5 end
6 return  $h^{(m)}, Z$ 

```

Lan truyền ngược

Các mạng nơ-ron học từ dữ liệu là dựa vào lan truyền ngược. Như tên của nó, lan truyền ngược hoạt động bằng cách truy tìm lại từ lớp đầu ra. Cho một mẫu huấn luyện x và nhãn y , đầu tiên chúng ta thu được dự đoán của mạng h sử dụng lan truyền tiến. Ý tưởng chung là "truyền" tín hiệu lỗi giữa h và y tới toàn bộ mạng, hay nói cách khác, cập nhật mạng để giảm lỗi dự đoán.

Tín hiệu lỗi được tính toán sử dụng một hàm mục tiêu, đo lường sự khác biệt giữa các dự đoán và nhãn. Đặc biệt, các bài toán hồi quy sử dụng Lỗi Trung Bình Bình Phương (Công thức 2.6), trong khi các bài toán phân loại sử dụng hàm mất mát Cross Entropy (Công thức 2.7). Các hàm đáng chú ý khác bao gồm Triplet Loss được dùng cho FaceNet [20], và Adversarial Loss được dùng bởi các mạng học cạnh tranh [9].

$$MSE = \frac{1}{n} \sum_{i=1}^n (h_i - y_i)^2 \quad (2.6)$$

$$CE = -\frac{1}{n} \sum_{i=1}^n (y_i \log(h_i) + (1 - y_i) \log(1 - h_i)) \quad (2.7)$$

trong đó n là tổng số mẫu.

Để cập nhật mạng, chúng ta cần biết các giá trị để cập nhật từng nơ-ron. Lan truyền ngược tìm những giá trị này tại mỗi tầng j sử dụng gradient w.r.t tầng thứ $j+1$. Ta bắt đầu bằng việc tính toán gradient tại tầng đầu ra m w.r.t hàm mục tiêu. Khi đó, sử dụng quy tắc chuỗi đạo hàm, tính gradient tại mỗi tầng ở trước. Mô tả chi tiết trong thuật toán 2.

Algorithm 2: Backpropagation

Input : Vector of input features x
 Ground truth labels y
 Network layers $L = ((W^{(1)}, b^{(1)}, g^{(1)}), \dots, (W^{(m)}, b^{(m)}, g^{(m)}))$

Output : Gradients at each layer $\Delta = (\delta^{(1)}, \dots, \delta^{(m)})$

```

1  $h, Z \leftarrow forwardPass(x)$ 
2  $c \leftarrow loss(h, y)$ 
3  $\delta^{(m)} = \frac{\delta c}{\delta W^{(m)}} \odot g^{(m)'}(z^{(m)})$ 
4 for  $j \in (m-1\dots 1)$  do
5   |  $\delta^{(j)} \leftarrow (W^{(j+1)T} \cdot \delta^{(j+1)}) \odot g^{(j)'}(z^{(j)})$ 
6 end
7 return  $\Delta$ 

```

Thuật toán xuồng dốc

Với gradient tại mỗi tầng, ta có thể cập nhật mạng bằng cách "di chuyển" trong các hướng đối nhau của vector đạo hàm. Với một bước đủ nhỏ, ta có thể đảm bảo rằng hàm mục tiêu giảm. Bằng việc lặp lại thủ tục này trong tập huấn luyện, mạng có thể hội tụ về mức tối thiểu của hàm mất mát. Thủ tục này được gọi là thuật toán xuồng dốc (Giải thuật 3).

Algorithm 3: Thuật toán xuồng dốc

Input : Vector đặc trưng đầu vào x
 Các nhãn thật y
 Các tầng mạng $L = ((W^{(1)}, b^{(1)}, g^{(1)}), \dots, (W^{(m)}, b^{(m)}, g^{(m)}))$
 Tỷ lệ học γ

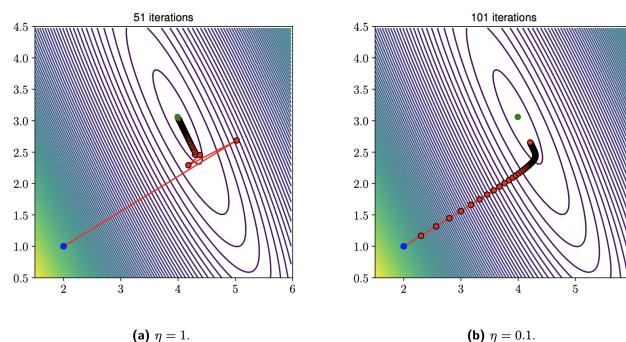
Output : Mạng được cập nhật

```

1 repeat
2   |  $\Delta \leftarrow backprop(x, y, L)$ 
3   | for  $j \in (1\dots m)$  do
4   |   |  $W^{(j)} \leftarrow W^{(j)} - \gamma \delta^{(j)}$ 
5   | end
6 until Hết tu;

```

Tốc độ học được dùng để điều chỉnh bước cho mỗi lần thuật toán xuồng dốc tại mỗi vòng lặp. Tốc độ học cao có nghĩa là mạng được cập nhật với những bước rộng, làm cho nó dễ bị phân kỳ ở các bước sau. Tốc độ học thấp thì sẽ đảm bảo hơn về sự hội tụ, nhưng sẽ tốn nhiều thời gian hơn. Thông thường thì tốc độ học nằm trong khoảng từ 10^{-2} đến 10^{-5} .



Hình 2.5: Đường đi nghiêm của linear regression với các learning rate khác nhau⁵

⁵Machine Learning cơ bản - Vũ Hữu Tiệp

Gradient descent tính đạo hàm trên toàn bộ tập dữ liệu. Đối với những tập dữ liệu rất lớn thì không khả thi cho lắm. Minibatch gradient descent là một phiên bản chỉnh sửa chỉ cập nhật mô hình trên một vài mẫu tại một thời điểm (1 batch). Phiên bản này chạy nhiều "vòng", mỗi vòng chạy qua tất cả các batch trong tập dữ liệu. Stochastic gradient descent (SGD) [2] là một biến thể với kích thước batch là 1. Stochastic và minibatch gradient descent thường đòi hỏi tốc độ học thấp hơn và nhiều vòng lặp hơn, nhưng vẫn hội tụ tại điểm cần với bộ nhớ thấp hơn.

Ngoài ra, rất nhiều nghiên cứu đã được thực hiện về việc điều chỉnh tỷ lệ học trong quá trình huấn luyện, đáng chú ý nhất một số thuật toán tối ưu như Adam [15], Adadelta [26] và RMSprop [24].

Nhìn chung, các mạng càng nhiều tầng và nơ-ron thí càng khớp hơn với dữ liệu huấn luyện. Tuy nhiên, không phải lúc nào tăng số nơ-ron cũng làm tăng độ chính xác. Các mạng nơ-ron lớn thì lại bị quá khớp rất nhanh, trong khi các mạng sâu hơn 5 - 6 lớp bị biến mất đạo hàm. Bởi vì những giới hạn này, kiến trúc mạng nơ-ron truyền thống chỉ khả thi ở một kích thước nhất định.

Các kỹ thuật Học sâu cố gắng tạo ra các mạng lớn hơn và sâu hơn để vượt qua những thách thức này. Các kỹ thuật này bao gồm các mạng, thuật toán tối ưu, các hàm kích hoạt,... Trong phần tiếp theo, chúng ta sẽ xem xét Mạng nơ-ron tích chập, một trong những mô hình có tác động mạnh nhất trong Học sâu.

2.2.2 Mạng nơ-ron tích chập

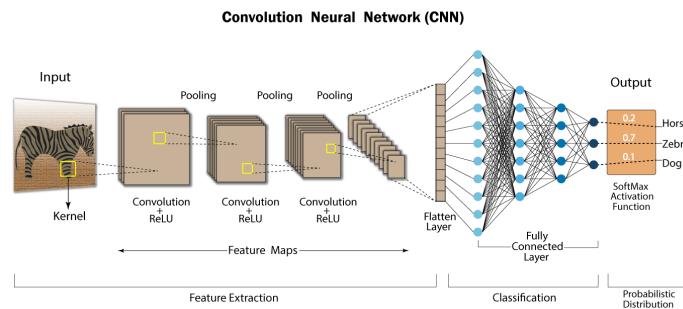
Mạng nơ-ron tích chập (Convolutional Neural Network [14] - Convolutional Neural Network (CNN)) ban đầu được thiết kế nhằm mục đích xử lý ảnh. Các kiến trúc của chúng hướng tới một vấn đề cốt lõi khi sử dụng mạng nơ-ron để xử lý ảnh đó là kích thước đầu vào có thể rất lớn (ví dụ 200 x 200 x 3) sẽ làm tăng chi phí tính toán với mạng quá đơn giản và dễ gây overfit dữ liệu.

Một mạng CNN truyền thống được cấu thành từ các thành phần sau:

Tầng Đầu vào cũng giống với mạng kết nối đầy đủ, giữ được các đặc trưng của đầu vào.

Tầng tích chập là lõi của một mạng tích chập (Hình 2.6). Các tham số của nó bao gồm một tập các bộ lọc có thể học được. Các bộ lọc nhỏ (với chiều rộng và chiều cao nhỏ), nhưng mở rộng về mặt chiều sâu của dữ liệu đầu vào. Ví dụ, một bộ lọc phổ biến trên tầng đầu tiên của một mạng tích chập có thể có kích

thước là $5 \times 5 \times 3$ (i.e. 5 điểm ảnh chiều rộng, 5 điểm ảnh chiều cao và 3 kênh màu sắc). Trong suốt giai đoạn chuyển tiếp, ta trượt bộ lọc theo chiều dọc và chiều cao của ảnh đầu vào rồi tính các tích điểm ảnh tương ứng giữa bộ lọc và ảnh đầu vào ở mỗi vị trí. Khi bộ lọc trượt qua ảnh đầu vào, nó xuất ra một đồ thị kích hoạt 2 chiều thể hiện phản hồi của nó tại mỗi vị trí không gian. Mạng sẽ tìm hiểu các bộ lọc kích hoạt khi chúng nhìn thấy một số đặc trưng trực quan như là cạnh của một số hướng hay là một vết màu, hay là các mẫu hình phức tạp trên các tầng cao hơn của mạng. Đồ thị kích hoạt từ các bộ lọc khác nhau được xếp chồng lên nhau dọc theo chiều sâu và tạo ra kết quả đầu ra.



Hình 2.6: An example convolution layer⁶

Một khác biệt chính giữa các tầng CONV và các tầng ẩn phổ biến nằm ở các kết nối địa phương của chúng. Mỗi neuron chỉ được kết nối với một vùng địa phương của ảnh được cho vào mô hình. Phạm vi không gian của kết nối này là một siêu tham số được gọi là trường tiếp nhận (hay kích thước bộ lọc). Mức độ kết nối dọc theo độ sâu luôn bằng với độ sâu của dữ liệu đầu vào. Cần nhấn mạnh sự bất đối xứng này trong cách chúng ta xử lý các chiều không gian khác nhau (chiều rộng, chiều cao và chiều sâu): Các kết nối không gian mang tính địa phương nhưng luôn phải đầy đủ dọc theo toàn bộ dữ liệu đầu vào.

Giả định được thực hiện bởi phép tích chập là nếu một đặc trưng là một đại diện tốt đối với một vùng ảnh, thì nó cũng có thể tốt cho các vùng khác. Khi các bộ lọc thực hiện phép toán tích chập, cách sử dụng các trọng số là như nhau, và gradient của chúng tích tụ trong suốt quá trình lan truyền ngược. Điều này cho phép các tầng tích chập chia sẻ trọng số giữa các vùng ảnh, làm giảm đáng kể kích thước của chúng trong khi vẫn đảm bảo toàn bộ dữ liệu đầu vào.

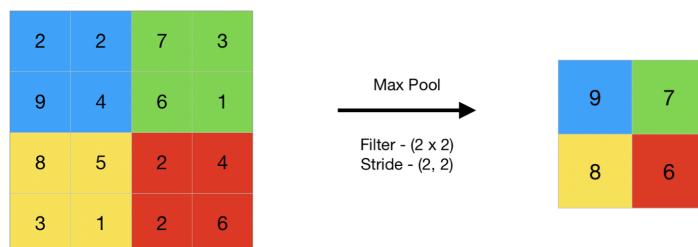
Thay cho các hàm kích hoạt như là *sigmoid* hay *tanh*, các tầng tích chập thường được kích hoạt bởi hàm ReLU, với $ReLU(x) = \max(x, 0)$. ReLU có một vài thuộc tính có thể giúp ích cho những mạng rất sâu:

⁶<http://cs231n.github.io/understanding-cnn/>

- ReLU là hàm phi tuyến tính
- Đạo hàm của ReLU bằng 1 hoặc 0, có nghĩa rằng nó không bị ảnh hưởng bởi sự mất mát đạo hàm

Một nhược điểm của hàm ReLU là sự tồn tại của "tế bào thần kinh chết" có đầu ra là âm. Những neuron này sẽ có đạo hàm bằng 0 và không thể học hoặc đóng góp vào mạng. Các hàm thay thế khác như Leaky ReLU hay Exponential ReLU đã được đề xuất để giải quyết vấn đề này.

Các lớp POOL (pooling) được sử dụng để giảm dần kích thước không gian của biểu diễn nhằm giảm lượng tham số và tính toán trong mạng, từ đó cũng giúp kiểm soát overfit. Lớp POOL hoạt động độc lập trên mọi lát cắt sâu của đầu vào và thay đổi kích thước của nó theo không gian, sử dụng toán tử MAX. Phổ biến nhất là tầng pooling với các bộ lọc 2×2 được áp dụng với sải bước bằng 2 làm giảm chiều mỗi lát cắt theo chiều sâu trong dữ liệu đầu vào theo cả chiều rộng và chiều cao, loại bỏ 75% các hàm kích hoạt. Mỗi toán tử MAX sẽ lấy giá trị lớn nhất trong 4 giá trị (nằm trong vùng 2×2 của bộ lọc). Kích thước chiều sâu vẫn không thay đổi (Hình 2.7).



Hình 2.7: Một ví dụ về maxpooling⁷

Hơn nữa để dùng max pooling, các đơn vị pooling cũng có thể thực hiện các hàm khác nhau ví dụ như pooling trung bình hoặc pooling L2-norm. Trước đây pooling trung bình thường được sử dụng nhiều nhưng gần đây không còn được ưa chuộng như toán tử max pooling đang thể hiện sự vượt trội trong thực nghiệm.

Các mạng CNN được cấu thành từ các "khối" CONV, ReLU, POOL với các kích thước khác nhau, và cuối cùng được gắn thêm một vài tầng kết nối đầy đủ để sinh ra các kết quả phân loại cần thiết. VGGNet [22] là một mạng rất phổ biến trong các kiến trúc như thế này.

Các kiến trúc được giới thiệu sau thường phức tạp hơn, các thứ tự phi tuyến tính của các khối tích chập để tăng khả năng học hỏi của mạng. Ví dụ như

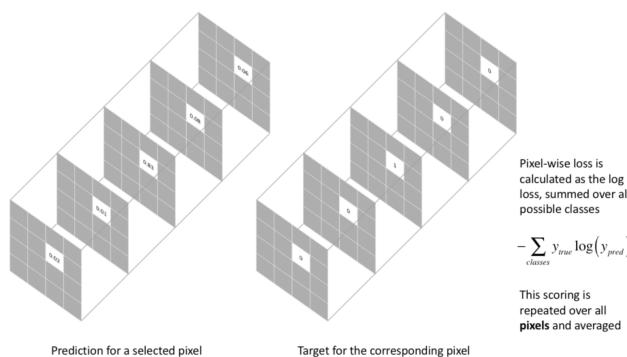
⁷<http://cs231n.github.io/understanding-cnn/>

ResNet [10] sử dụng các kết nối bỏ qua (thường được gọi là các "khối dư") giữa các khối tích chập.

2.2.3 Hàm mục tiêu và giải thuật huấn luyện

Cross Entropy Loss

Cross Entropy Loss [27] là hàm mục tiêu được sử dụng phổ biến nhất cho các bài toán phân loại ảnh pixel-wise cross entropy. Hàm mục tiêu này kiểm tra từng pixel riêng lẻ, so sánh các dự đoán của lớp (vector pixel theo chiều sâu) với vector mục tiêu được mã dưới dạng one-hot (vector chỉ có các giá trị 0 và 1).



Hình 2.8: Minh họa cho cách dùng cross entropy⁸

Bởi vì hàm cross entropy đánh giá các dự đoán lớp đối với mỗi điểm ảnh một cách độc lập và sau đó tính trung bình tất cả các điểm ảnh. Vậy nên về cơ bản thì việc học là bình đẳng giữa các điểm ảnh trong hình. Đây có thể là một vấn đề nếu các lớp khác nhau của bài toán có biểu diễn không cân bằng trong hình ảnh, khi huấn luyện có thể bị chi phối bởi lớp xuất hiện nhiều nhất. Hệ số xác suất ban đầu vốn được phát triển cho dữ liệu nhị phân và có thể tính được bởi công thức sau:

$$\text{Cross Entropy} = -\frac{1}{N} \sum_j y_j * \log(\hat{y}_j)$$

Trong đó, y_j là nhãn thật, \hat{y}_j là nhãn dự đoán

Dice Loss

Dice Loss [23] là một hàm mục tiêu phổ biến khác cho các bài toán phân vùng ảnh là dự trên hệ số xác, vốn là một phép đo cơ bản cho mức độ chồng chéo

⁸<https://www.jeremyjordan.me/semantic-segmentation/>

giữa hai mẫu với nhau. Số đo này nằm trong khoảng từ 0 đến 1 trong đó hệ số Xúc xác thể hiện cho sự chồng chéo hoàn hảo và hoàn chỉnh.

$$Dice = \frac{2|A \cap B|}{|A| + |B|} \quad (2.8)$$

Trong đó $A \cap B$ thể hiện tập các phần tử chung giữa A và B, $|A|$ thể hiện các phần tử thuộc A (tương tự với B).

Lấy ví dụ về hệ số xúc xác trên mặt nạ phân vùng được dự đoán, ta xấp xỉ $|A \cap B|$ như là phép nhân từng phần tử tương ứng giữa dự đoán của mô hình với mặt nạ mục tiêu rồi sau đó cộng tổng ma trận kết quả.

$$|A \cap B| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix}_{\text{prediction}} * \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}_{\text{target}} \xrightarrow{\text{element-wise multiply}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.41$$

Hình 2.9: Phép nhân tương ứng từng điểm ảnh rồi cộng tổng kết quả thu được ⁹

Bởi vì các mặt nạ của ta là nhị phân, nên ta có thể loại bỏ một cách hiệu quả bất kỳ điểm ảnh nào mà không "hoạt động" trong mặt nạ mục tiêu khỏi dự đoán của mình. Đối với những điểm ảnh còn lại, về căn bản thì ta sẽ trừng phạt các dự đoán có độ tin cậy thấp; với một giá trị cao hơn thì sẽ dẫn tới hệ số xúc xác tốt hơn.

Để định lượng $|A|$ và $|B|$, đơn giản ta chỉ cộng tổng bình phương các điểm ảnh. Ta có hình 2.10

Có 2 ở tử số khi ta tính hệ số xúc xác vì mẫu số của chúng ta "đếm gấp đôi" các phần tử chung giữa hai tập. Để tính một hàm mục tiêu mà có thể được tối ưu, ta sử dụng $1 - Dice$. Hàm mục tiêu này được biết đến như là "Hàm xúc xác mềm" bởi vì ta sử dụng các xác suất được dự đoán thay vì dùng ngưỡng và biến chúng thành mặt nạ nhị phân.

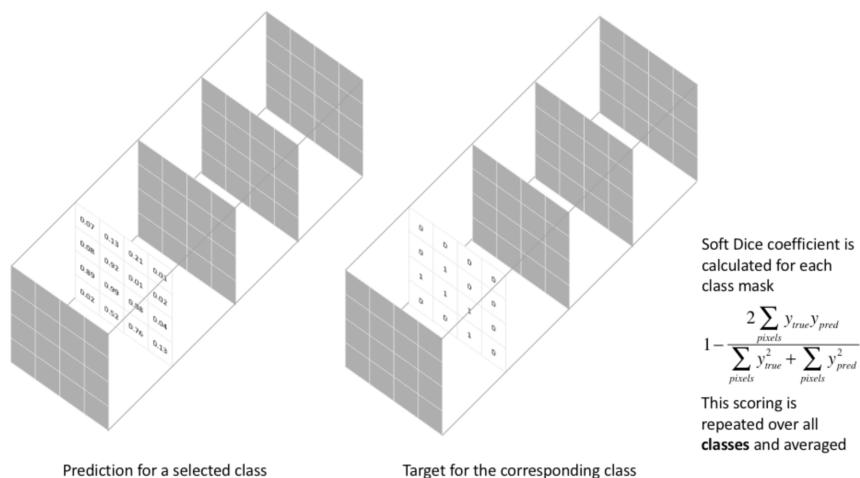
Đối với đầu ra của hàm nơ-ron, tử số liên quan đến các kích hoạt chung giữa dự đoán của mô hình và mặt nạ mục tiêu, trong đó mẫu số liên quan đến số lượng kích hoạt trong mỗi mặt nạ riêng biệt. Điều này có tác dụng chuẩn hóa măt măt theo kích thước của mặt nạ mục tiêu để "hàm mục tiêu xúc xác mềm" không gặp khó khăn trong việc học từ các lớp có ít biểu diễn không gian hơn trong một hình ảnh.

⁹<https://www.jeremyjordan.me/semantic-segmentation/>

$$|A| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix}^2 \text{ (optional)} \xrightarrow{\text{sum}} 7.82$$

$$|B| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}^2 \text{ (optional)} \xrightarrow{\text{sum}} 8$$

Hình 2.10: Tổng các bình phương điểm ảnh định lượng cho tập A và tập B ¹⁰



Hình 2.11: Một hàm mất mát xác mềm được tính cho từng lớp riêng biệt và sau đó lấy trung bình cộng để thu được kết quả cuối cùng ¹¹

Chương 3

Phương pháp sử dụng

3.1 Kiến trúc tổng thể

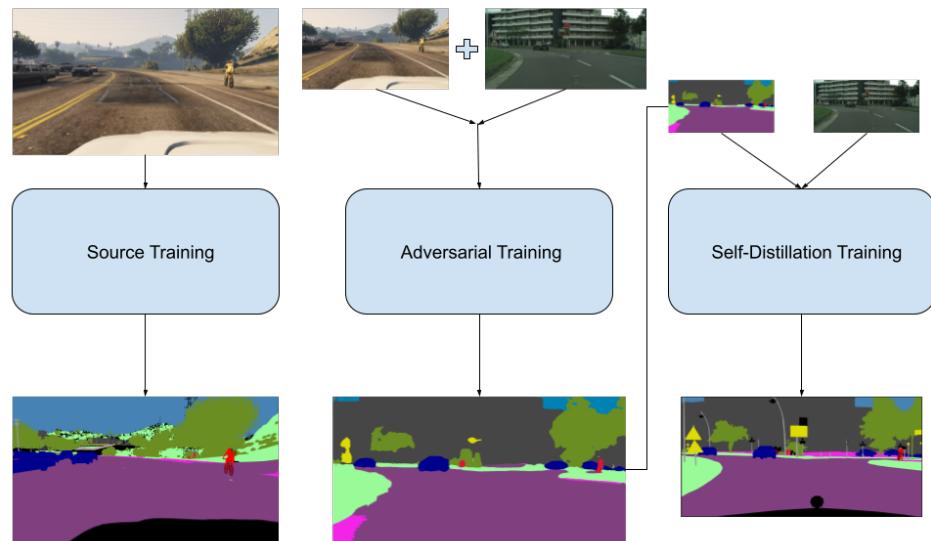
Để giải quyết bài toán đặt ra ở phía trên, chúng ta cần phải thực hiện ba giai đoạn khác nhau:

- Huấn luyện mô hình phân vùng ảnh ngữ nghĩa trên tập nguồn
- Huấn luyện cạnh tranh mô hình trên cả tập nguồn và tập đích (tập đích sử dụng nhãn giả được sinh ra từ mô hình có được từ giai đoạn 1)
- Huấn luyện mô hình tự cô đặc bằng nhãn mà nó đã sinh ra ở giai đoạn 2

Mô hình tổng quan được xây dựng một cách mềm dẻo với 3 bộ phận khác nhau có thể thay thế và thử nghiệm một cách linh hoạt, bao gồm:

- Encoder để trích xuất các đặc trưng của ảnh
- Decoder có tác dụng phân loại các điểm ảnh dựa vào thông tin mà Encoder mang lại
- Discriminator phục vụ cho giai đoạn huấn luyện cạnh tranh nhằm học các thông tin từ ảnh không nhãn (miền đích)

Luồng kết hợp của bài toán được thể hiện như hình 3.8.



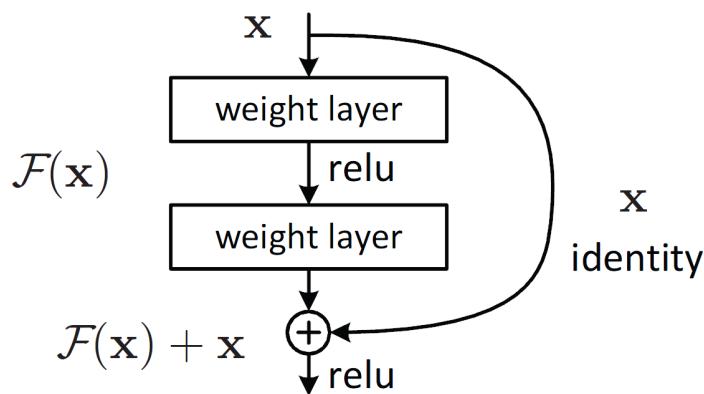
Hình 3.1: Kiến trúc tổng thể của mô hình¹

3.2 Bộ trích xuất đặc trưng

Mô hình thử nghiệm với 2 kiến trúc Encoder khác nhau là Resnet101 và Hardnet68

Resnet

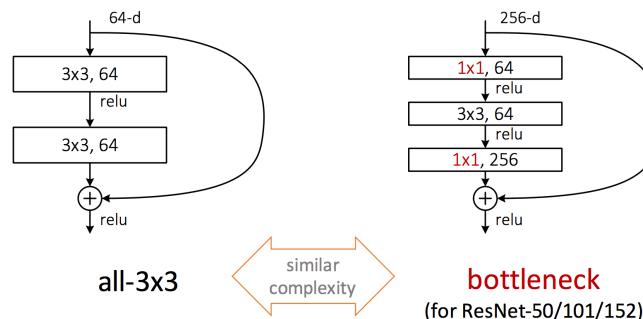
Mạng Resnet [11] được sinh ra nhằm mục đích giúp cho mô hình học sâu hơn nhưng không bị biến mất đạo hàm (vanishing gradient). Đây là hiện tượng Gradient thường sẽ có giá trị nhỏ dần khi đi xuống các layer thấp hơn. Dẫn đến kết quả là các cập nhật thực hiện bởi Gradients Descent không làm thay đổi nhiều trọng số của các tầng đó và làm chúng không thể hội tụ và mạng sẽ không thu được kết quả tốt. Mạng Resnet có thể làm được điều này là nhờ vào khối phần dư. Trong đó, đầu vào x được thêm trực tiếp vào khối phần dư của mạng trở thành $F(x) + x$ được gọi là kết nối bỏ qua (skip connection) hoặc kết nối tắt.



Hình 3.2: Kiến trúc kết nối bù qua của Resnet²

Có 2 kiến trúc khối trong Resnet đó là **Khối cơ bản** và **Khối cổ chai** thể hiện trong Hình 3.3:

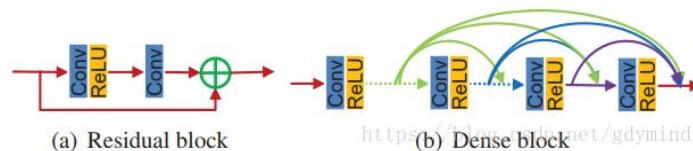
- **Khối cơ bản** thường được sử dụng trong các mạng không quá sâu. Sử dụng 2 lớp tích chập 3×3 .
- **Khối cổ chai** thường được sử dụng cho các mạng có kiến trúc sâu hơn. Đầu tiên sử dụng tích chập 1×1 để giảm kích thước, sau đó tích chập 3×3 và cuối cùng sử dụng kích thước 1×1 để khôi phục kích thước ban đầu.



Hình 3.3: Kiến trúc khối cơ bản và khối cổ chai trong Resnet³

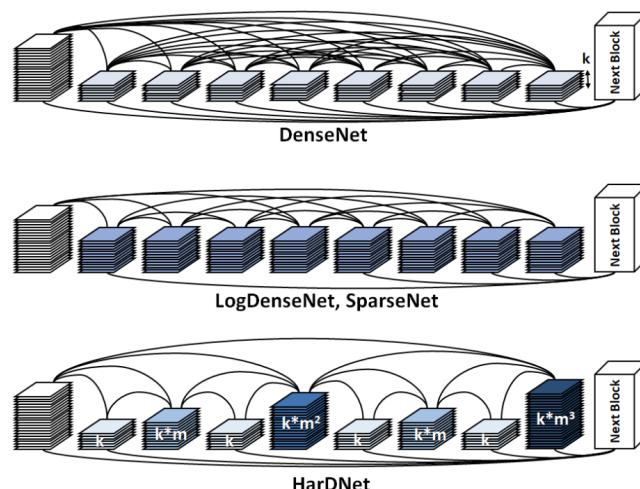
Hardnet

Hardnet [3] được lấy cảm hứng từ Densenet[13]. Densenet khá tương đồng với Resnet nhưng kiến trúc gồm các lớp chuyển tiếp và các khối dày đặc. Khối dày đặc này tương tự như khối phần dư tuy nhiên kiến trúc không sử dụng phép cộng mà sử dụng phép ghép nối. Điều này khiến cho mô hình trở nên nặng hơn. Lớp chuyển tiếp được cấu tạo bằng cách kết hợp một lớp tích chập làm giảm độ sâu và một lớp maxpooling làm giảm kích thước nhằm giảm chiều dữ liệu. So sánh Khối phần dư và Khối dày đặc trong Hình 3.4



Hình 3.4: Kiến trúc khđi phđn đđu và khđi dđy đđc⁴

Không giống như sự thđa thđt đđc đđe xuđt trong LogDenseNet[12], chúng tôi đđe tđng k kết nối với tđng $k - 2^n$ nếu 2^n chia hết cho k , với n là số nguyên khđng âm và $k - 2^n \geq 0$; cụ thđ, lớp 0 là lớp đđu vào. Dđo lđt đđo kết nối này (Hình 3.5), mđt khi lớp 2^n đđc xử lý, lớp 1 đđn 2ⁿ⁻¹ có thđ đđc xđa khđi bđ nhđ. Các kết nối làm cho mđng xuđt hiện dđo lđt đđu dạng chđng chđo của sđng hđi bđc hai. Do đđ nđc có tên là Harmonic Density Connected (HarDNet). Dđe án phđn biệt hđa đđc đđe xuđt giđm chi phđ nđi tốt hđn đđng kđ so với LogDenseNet thuđn. Kiđu kết nối này cđng giđng như mđt FractalNet[16] , ngoại trừ cái sau sử dụng các phđm tđt tđnh trung bđnh thay vđ nđi.



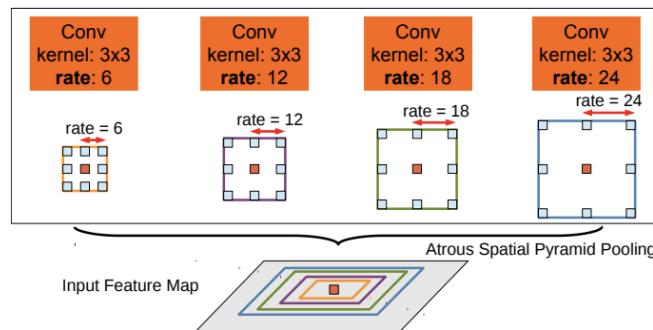
Hình 3.5: Hình minh họa cho DenseNet, LogDenseNet, SparseNet và Harmonic DenseNet đđc đđe xuđt (HarDNet), trong đđ mỗi lớp là một tích chđp 3x3⁵

3.3 Bđ giải mđ

Atrous Spatial Pyramid Pooling

Atrous Spatial Pyramid Pooling [4] hay ASPP là một loại tích chđp đđe phđn vđng mđnh mđ các đđi tđng ở nhiđu kích thđc bằng các bộ lọc ở nhiđu tđy lđt lđy mđu và trđrđg xđm hiđu quđ khđc nhau. Ưđ đđm của ASPP là khđng lđm giđm chiđu của bđn đđ đặc trưng quá sâu nhưng vđn giđu sđ lượng thđm sđ và chi phđ tđnh tođn tđng đđu đđng mđng nđ-ron tích chđp.

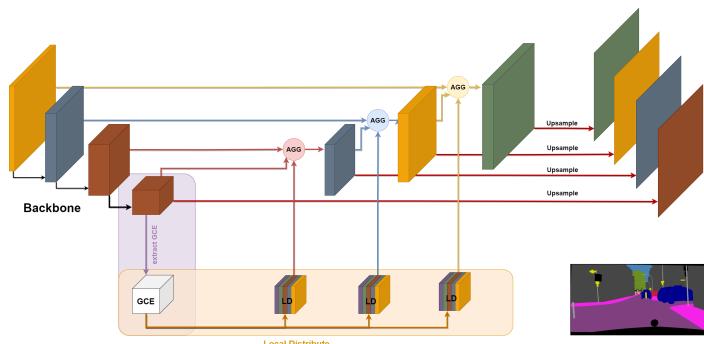
Dữ liệu đầu vào (đầu ra của bộ trích xuất đặc trưng) sẽ được đưa qua đồng thời 4 mạng nơ-ron tích chập với bộ lọc 3×3 và các tham số khác như padding và dilation. Điều này giúp một Atrous convolution với tỷ lệ R thêm các số không R-1 giữa giá trị liên tục để nó phóng to bộ lọc k-kernel thành $K + (K-1)(R-1)$ mà không tăng số lượng tham số. Kết quả được tổng hợp bằng cách cộng các kết quả từ cả 4 mạng lại với nhau, như trong hình 3.6



Hình 3.6: Để phân loại các điểm ảnh trung tâm (màu cam), ASPP khai thác đặc trưng đa kích thước bằng cách sử dụng nhiều bộ lọc song song với các tỷ lệ khác nhau. Trưởng nhìn hiệu quả được hiển thị bằng các màu khác nhau⁶

Global Aggregation Local Distribution (GALD)

Tác giả thử cải tiến bằng cách sử dụng một kiến trúc cơ sở do nhóm đề xuất cho giai đoạn 1 nhằm khớp dữ liệu nguồn một cách tốt hơn.



Hình 3.7: Kiến trúc Hardnet68 được kết hợp với GALD để tăng khả năng khớp dữ liệu của mô hình ⁷

Mô hình tổng quan (Hình 3.7) được xây dựng như một biến thể của mạng Unet [18]. Ảnh đường phố đầu tiên sẽ được đưa qua các mô hình cơ sở để trích xuất đặc trưng và thu được các thông tin bậc cao. Qua mỗi lớp gồm một tập các khối convolution và pooling, độ phân giải của ảnh sẽ bị giảm một nửa tuy nhiên số kênh sẽ tăng lên gấp đôi do đó đặc trưng sẽ có ít thông tin về không gian (góc, cạnh,...) thay vào đó là thông tin về ngữ nghĩa. Đặc trưng ở lớp cuối cùng sẽ được đi qua lớp global context encoder(GCE) để lấy được thông tin ngữ cảnh và sau đó phân phối lại các đối tượng cụ thể bằng local distribution(LD). Cuối

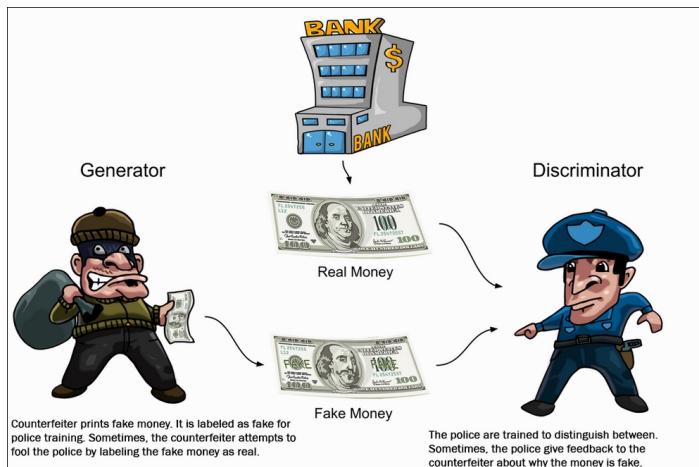
cùng, mô hình sử dụng một lớp Aggregator(AGG) để kết hợp các thông tin bậc cao, bậc thấp và toàn cục để đưa dần ảnh về độ phân giải ban đầu để đưa qua lớp phân loại.

3.4 Học đối nghịch

3.4.1 Tổng quan về mạng GAN

Khi nhắc đến học đối nghịch, ta không thể không đề cập đến kiến trúc mạng GAN (Generative Adversarial Network) [6]. GAN thuộc họ các mô hình sinh dữ liệu. Các mô hình sinh có thể tạo ra các đầu ra có ý nghĩa mới được cung cấp các mã hóa tùy ý. Ví dụ: GAN có thể tạo khuôn mặt người nổi tiếng mới không phải là người thật bằng cách thực hiện phép nội suy không gian tiềm ẩn. Ta có thể áp dụng hiệu quả ý tưởng này cho những bộ dữ liệu không gán nhãn hay nói đúng hơn là việc gán nhãn rất tốn kém chi phí và thời gian.

Mô hình tổng quát của mạng GAN bao gồm 2 bộ phận là bộ sinh (generator) và bộ phân biệt (discriminator) có vai trò tương tự như một kẻ làm giả và một cảnh sát. Mục tiêu của kẻ làm giả là tạo ra những tờ tiền giả nhằm đánh lừa cảnh sát. Ngược lại, nhiệm vụ của cảnh sát là phân biệt được đâu là tiền giả và đâu là tiền thật.



Hình 3.8: Kiến trúc tổng quát của mạng GAN được minh họa như sự cạnh tranh giữa kẻ làm tiền giả và cảnh sát⁸

Áp dụng ý tưởng đó vào bài toán thích ứng miền cho ảnh đường phố, ta có thể sử dụng thêm bộ phân biệt có nhiệm vụ phân biệt ảnh thuộc miền nguồn hay miền đích. Trong khi đó, bộ trích xuất đặc trưng (Encoder) được coi như một generator có nhiệm vụ trích xuất ra các thông tin từ ảnh của cả 2 miền sao cho có thể đánh lừa được bộ phân biệt. Để làm tốt nhiệm vụ này, việc lựa chọn hàm

mất mát phù hợp là rất quan trọng.

3.4.2 Hàm mất mát sử dụng

Trong suốt quá trình huấn luyện, bộ phân biệt không chỉ có nhiệm vụ phân biệt xem ảnh nào thuộc miền nào, mà còn có vai trò học các cấu trúc lớp của mô hình, hay nói cách khác là phân biệt ảnh đầu vào đến mức lớp, xem ảnh nào thuộc miền nào và điểm ảnh thuộc lớp nào.

$$\mathcal{L}_D = - \sum_{i=1}^{n_s} \sum_{k=1}^K a_{ik}^{(s)} \log P(d=0, c=k | f_i) - \sum_{j=1}^{n_t} \sum_{k=1}^K a_{jk}^{(t)} \log P(d=1, c=k | f_j)$$

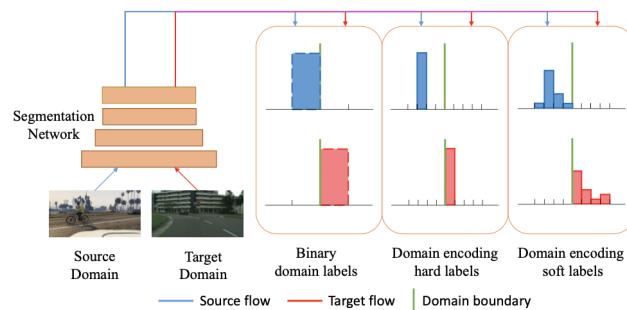
Đặc trưng của hàm GAN chính là ở hàm mất mát cạnh tranh. Hàm mất mát cạnh tranh \mathcal{L}_{adv} được dùng nhằm đánh lừa bộ phân biệt. \mathcal{L}_{adv} được thiết kế để tối đa hóa xác suất của các đặc trưng từ miền đích được coi như là các đặc trưng nguồn mà không phá hỏng đi mối quan hệ giữa các đặc trưng và các lớp.

$$\mathcal{L}_{adv} = - \sum_{j=1}^{n_t} \sum_{k=1}^K a_{jk}^{(t)} \log P(d=0, c=k | f_j)$$

3.4.3 Kiến trúc bộ phân biệt

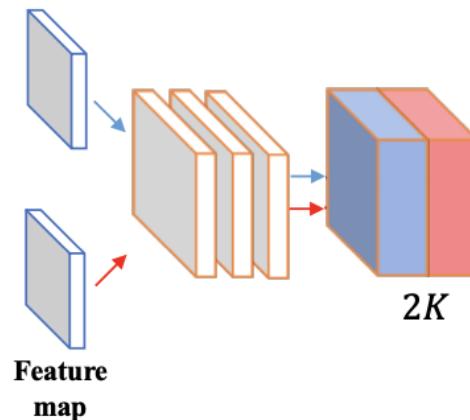
Để làm cho bộ phân biệt không chỉ tập trung vào việc phân biệt các miền, ta chia mỗi kênh trong số hai kênh đầu ra của bộ phân biệt nhị phân thành K (số lớp cần phân loại) kênh và khuyến khích việc học cạnh tranh ở mức độ chi tiết.

Để minh họa các chiến lược khác nhau tạo mã hóa cho các miền. Ở đây ta so sánh ba chiến lược khác nhau để trích xuất kiến thức từ mạng phân đoạn để xây dựng mã hóa miền: nhãn miền nhị phân, nhãn cứng 0 1 và nhãn mềm đa kênh. Hình 3.9



Hình 3.9: Từ trái sang phải lần lượt là nhãn nhị phân, nhãn cứng với chỉ số 0 1, nhãn mềm⁹

Ta sử dụng bộ phân biệt với kiến trúc như sau: Đối với bộ phân biệt chi tiết, ta áp dụng một cấu trúc đơn giản bao gồm 3 lớp tích chập với số kênh 256, 128, $2K$, bộ lọc 3×3 và độ dịch chuyển là 1. Mỗi lớp tích chập được theo sau bởi một hàm Leaky-ReLU được tham số hóa bởi 0,2 ngoại trừ lớp cuối cùng. Hình 3.10



Hình 3.10: Bộ phân biệt có tác dụng nhằm giúp cho bộ mã hoá căn chỉnh các đặc trưng của 2 miền ảnh nhằm tìm ra những điểm chung nhất giữa 2 miền¹⁰

3.5 Học chắt lọc

3.5.1 Chắt lọc kiến thức

Để giải quyết độ trễ, độ chính xác và nhu cầu tính toán tại thời điểm sử dụng mô hình phân loại, chúng ta có thể sử dụng kỹ thuật nén mô hình:

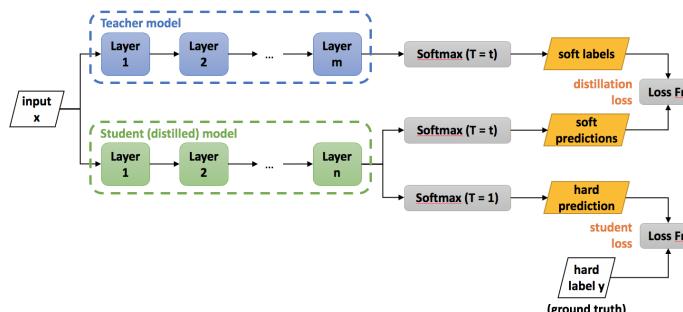
- Lượng tử hóa mô hình, số học có độ chính xác thấp để suy luận, ví dụ như chuyển đổi một số thực sang một số nguyên không dấu
- Cắt tia, loại bỏ trọng số hoặc các hàm kích hoạt gần bằng 0 để thu nhỏ mô hình hơn

- Chắt lọc kiến thức: một mô hình phức hợp lớn chắt lọc kiến thức của nó và chuyển hoá nó để đào tạo một mạng lưới nhỏ hơn. Mạng nhỏ hơn có gắng khớp với các dự đoán và phân phối của mạng lớn hơn

Mạng "giáo viên" và "học sinh" thường có các đặc điểm sau đây:

- Giáo viên: Mô hình lớn hoặc là tập hợp của các mô hình (khá nhiều tham số cồng kềnh)
- Học sinh: thường nhỏ và gọn nhẹ

Chắt lọc kiến thức nghĩa là đào tạo mạng học sinh bằng cách sử dụng kiến thức chắt lọc được ngoại suy từ mạng giáo viên, học sinh cố gắng hết sức để tiếp thu mọi thứ do giáo viên giảng dạy (mô hình học sinh được đào tạo để khái quát hóa theo cùng một cách với giáo viên). Một mô hình nhỏ được đào tạo để tổng quát hóa theo cùng một cách thường sẽ làm tốt hơn nhiều trên dữ liệu thử nghiệm so với một mô hình nhỏ được đào tạo theo cách thông thường trên cùng một tập hợp đào tạo đã được sử dụng để đào tạo nhóm. Lưu ý là mô hình lớn nên đạt được hiệu suất State of the art (SOTA). Ta có thể hình dung các hoạt động của phương pháp này như hình 3.8



Hình 3.11: Mô hình tổng quát của học cõi đọng kiến thức¹¹

3.5.2 Tự chắt lọc

Trong bài toán này ta sẽ áp dụng phép tự chắt lọc. Ở đó, mô hình giáo viên và mô hình học sinh có cấu trúc không khác gì nhau. Ta sẽ sử dụng các xác suất lớp được suy ra từ mô hình lớn như là các "nhân mềm" cho việc huấn luyện mô hình nhỏ. Công thức cụ thể như sau:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

Trong đó T là nhiệt độ, sử dụng giá trị T lớn có thể xuất ra một phân phối xác suất mềm hơn trên tất cả các lớp. Để giải thích cho điều này, ta nhận thấy rằng với mỗi điểm ảnh đã được dự đoán, nếu giữ nguyên xác suất được tính ra từ mô hình hoặc sử dụng "nhân cứng" 0 1 thì khi đó thông tin về các nhãn sai sẽ bị giảm đi vai trò hay thậm chí là không có vai trò gì.



Hình 3.12: "Nhân mềm" linh hoạt và cung cấp nhiều thông tin hơn "nhân cứng"¹²

"Nhân mềm" với giá trị T sẽ chứa thông tin có giá trị dữ liệu phong phú hơn. Cung cấp nhiều thông tin hơn và ít chênh lệch hơn về gradient trong quá trình huấn luyện. Cho phép mô hình học sinh nhỏ hơn được huấn luyện trên dữ liệu nhỏ hơn nhiều so với mô hình cồng kềnh ban đầu và với tốc độ học tập cao hơn nhiều.

Chương 4

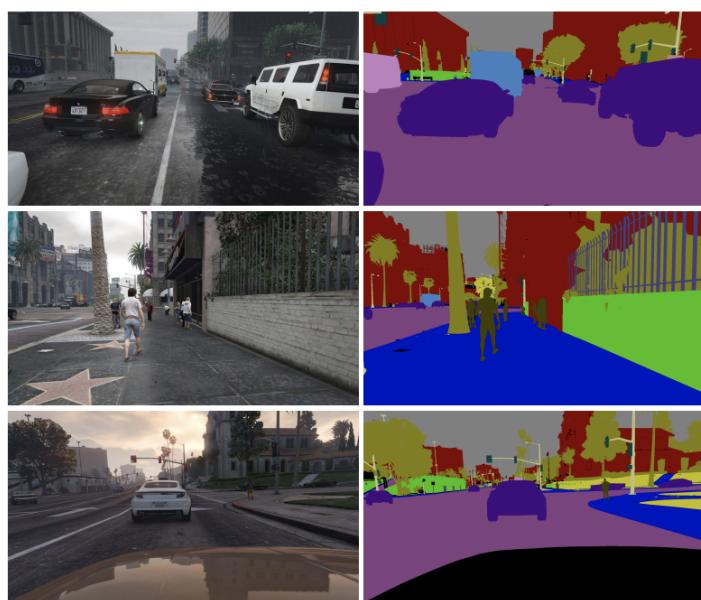
Thử nghiệm và đánh giá

4.1 Dữ liệu

Dữ liệu nguồn và dữ liệu đích

Dữ liệu chính được sử dụng trong đồ án này là GTA5 [17] và Cityscapes [5].

Trong đó, bộ dữ liệu GTA5 chứa 24966 hình ảnh tổng hợp với chủ thích ngữ nghĩa tới từng điểm ảnh được chia thành 10 tập nhỏ, mỗi tập 2500 ảnh ngoại trừ tập cuối có 2466 ảnh. Các hình ảnh đã được kết xuất bằng cách sử dụng trò chơi điện tử thế giới mở Grand Theft Auto 5 và tất cả đều từ góc nhìn xe hơi trên đường phố của các thành phố ảo theo phong cách Mỹ. Có 19 lớp ngữ nghĩa tương thích với các lớp trong bộ dữ liệu Cityscapes (Hình 4.1).



Hình 4.1: Dữ liệu trong GTA5 được trích xuất từ trò chơi điện tử

Bên cạnh đó, bộ dữ liệu Cityscapes tập trung vào cảnh đường phố đô thị. Tập

Bảng 4.1: Định nghĩa các lớp trong tập dữ liệu Cityscapes

Nhóm	Các lớp tương ứng
flat	road · sidewalk · parking+ · rail track+
human	person* · rider*
vehicle	car* · truck* · bus* · on rails* · motorcycle* · bicycle* · caravan*+ · trailer*+
construction	building · wall · fence · guard rail+ · bridge+ · tunnel+
object	pole · pole group+ · traffic sign · traffic light
nature	vegetation · terrain
sky	sky
void	ground+ · dynamic+ · static+

dữ liệu bao gồm 30 lớp ngữ nghĩa được chụp từ góc nhìn của xe hơi tại 50 thành phố trên khắp thế giới trải dài qua các tháng trong năm. Cityscapes bao gồm 5000 ảnh với nhãn tinh và 20000 ảnh với nhãn thô. Đồ án này chỉ sử dụng 5000 ảnh với nhãn tinh và kiểm thử trên tập dữ liệu với 500 ảnh.

Các đối tượng cần nhận diện được gắn nhãn không được có những mảng đen, tức là nếu có một số hậu cảnh có thể nhìn thấy ‘xuyên qua’ đối tượng nào đó, nó được coi là một phần của đối tượng đó luôn. Điều này cũng áp dụng cho các vùng có nhiều hỗn hợp với hai hoặc nhiều lớp: chúng được gắn nhãn với lớp nằm phía trước. Ví dụ: cây lá trước cửa nhà hoặc bầu trời (mọi thứ thuộc về cây), cửa sổ ô tô trong suốt (mọi thứ thuộc về ô tô) (Hình 4.2).



Hình 4.2: Một kiểu gán nhãn trong Cityscapes

Bộ mô phỏng Carla

Một hướng cải tiến của bài toán là ta áp dụng bộ mô phỏng Carla [7] thêm dữ liệu cho tập nguồn, làm cho thông tin trở nên đa dạng và phong phú hơn. Thêm vào đó, dữ liệu được thêm vào không chỉ là cảnh đường phố ban ngày với đường khô ráo nữa mà còn được bổ sung thêm các kiểu thời tiết và thời gian khác như: trong cơn mưa, sau cơn mưa, bình minh, hoàng hôn, ban đêm, ... Đồ án thực hiện sinh ảnh từ bộ mô phỏng với 5 cấu hình được thể hiện trong Bảng 4.2

⁰<https://github.com/taintpro98/rnd-semantic-segmentation>

Bảng 4.2: Cấu hình trong bộ sinh ảnh Carla

Cấu hình	Thời tiết	Số phương tiện	Số người	Bản đồ
1	1 (ClearNoon)	100	300	Thị trấn 1
2	8 (ClearSunset)	100	50	Thị trấn 1
3	2 (CloudyNoon)	100	50	Thị trấn 1
4	0 (Default)	100	100	Thị trấn 2
5	7 (SoftRainNoon)	80	50	Thị trấn 2

Bảng 4.3: Định nghĩa các lớp trong tập dữ liệu được sinh từ Bộ mô phỏng Carla

Tên lớp	Giá trị RGB
None	0, 0, 0
building	0, 127, 180
fence	127, 40, 127
other	80, 180, 80
pedestrians	255, 127, 127
pole	180, 180, 180
roadline	255, 180, 0
road	255, 255, 0
sidewalk	255, 0, 255
vegetation	0, 255, 0
vehicle	0, 0, 255
wall	127, 40, 127
traffic sight	255, 0, 0

Trong bộ mô phỏng Carla chỉ có 13 lớp. Trong đó nhãn "other" ta buộc phải bỏ qua vì thông tin của nó không cụ thể. Hoặc ví dụ như nhãn "vehicle" trong Carla quá chung chung bao gồm nhiều nhãn khác trong Cityscapes. Vậy nên, bộ mô phỏng Carla chỉ đóng vai trò làm giàu thêm cho một vài lớp sẵn có trong GTA5 chứ không thể hoàn toàn thay thế. Bù lại, nó đảm bảo cho sự đa dạng, tùy biến dữ liệu ảnh đường phố theo nhu cầu huấn luyện mà ta mong muốn. Cụ thể các lớp của Carla được thể hiện trong Bảng 4.3

Tương thích nhãn giữa các tập dữ liệu

Một vấn đề cần lưu ý là sự tương thích về nhãn giữa các tập dữ liệu với nhau. Định nghĩa các lớp trong tập dữ liệu Cityscapes đã được thể hiện trong Bảng 4.1. Nhưng trong tập GTA5 và Bộ mô phỏng Carla không thể có hết. Coi dữ liệu được sinh ra từ bộ mô phỏng Carla như dữ liệu tập nguồn bổ sung, ta có quy tắc tương thích nhãn các tập dữ liệu như sau:

- Ta chỉ quan tâm đến 19 nhãn chung giữa GTA5 và Cityscapes. Chỉ huấn luyện và đánh giá mô hình trên 19 nhãn này. Gọi đây là tập "nhãn đích"
- Trên tập dữ liệu GTA5 và Cityscapes, nhãn nào không thuộc tập "nhãn đích" thì ta bỏ qua bằng cách gán cho nó giá trị là 255. Các nhãn còn lại, chuyển chúng về khoảng các số nguyên từ 0 đến 18 cho đồng bộ.

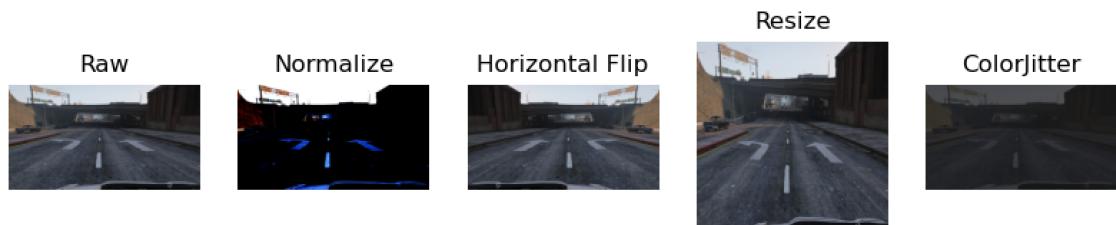
- Trên tập dữ liệu Carla, nhãn nào thuộc tập "nhãn đích" hoặc được bao hàm trong một nhãn thuộc "nhãn đích" thì sẽ được chuyển đổi tương ứng. Ví dụ như pedestrians được chuyển đổi thành person. Nếu nhãn nào bao hàm một nhãn trong tập "nhãn đích" thì ta có thể bỏ qua luôn bằng cách chuyển đổi nó thành 255.

4.2 Tiền xử lý dữ liệu

Kích thước ảnh Cityscapes cần xử lý là 2048 x 1024. Tập dữ liệu của ta được chia làm 2 loại là tập nguồn (GTA5) và tập đích (Cityscapes). Ta sử dụng các phép biến đổi sau cho việc xử lý dữ liệu đầu vào:

- Normalize: Chuẩn hóa một ảnh với giá trị trung bình và độ lệch chuẩn
- Horizontal Flip: Lật ngược ảnh theo chiều ngang
- Resize: Thay đổi kích thước của ảnh
- Color Jitter: Thay đổi ngẫu nhiên độ sáng, độ tương phản, độ bão hòa và màu sắc của hình ảnh.

Trước khi bắt đầu huấn luyện, ta thực hiện xử lý dữ liệu như trong hình 4.3



Hình 4.3: Tiền xử lý dữ liệu giúp cho mô hình học được những thông tin đa dạng hơn

4.3 Thiết lập thử nghiệm

Trong bài toán này ta sử dụng mô hình Fine-grained Discriminator [25]. Việc huấn luyện mô hình này tương đối mất nhiều thời gian và đòi hỏi lượng dữ liệu lớn. Quá trình huấn luyện bao gồm 3 giai đoạn như đã nêu ở chương trước. Cùng với đó, chúng ta cần phải sử dụng các kỹ thuật tiền xử lý dữ liệu và lựa chọn các tham số phù hợp trong quá trình huấn luyện mô hình. Một vài hạn chế của mô hình hiện tại đó là:

- Ở giai đoạn 1, mô hình chỉ được học từ dữ liệu nguồn nhằm tạo một mô hình cơ bản. Với những thông tin được cho từ tập nguồn, mô hình có thể học tốt nhưng chưa chắc đã khớp với thông tin trong tập đích.
- Mô hình gấp hiện tượng quá khớp khi mà càng về các vòng cuối thì hàm mất mát tuy giảm nhưng hiệu năng lại không tăng tương ứng.

Những hạn chế này có thể ảnh hưởng đến kết quả cuối cùng của bài toán phân vùng ảnh. Một số kết quả thu được được minh họa trong (Hình 4.4).



Hình 4.4: Theo thứ tự lần lượt là: Ảnh đầu vào, nhãn thật, kết quả sau giai đoạn 1, kết quả sau giai đoạn 2, kết quả sau giai đoạn 3

Tất cả các mô hình đều được thực hiện bằng:

- Pytorch 1.6.0 và CUDA 11.1
- Chạy trên 1 GPU Nvidia RTX3090 duy nhất

4.3.1 Huấn luyện mô hình cơ sở trên tập dữ liệu nguồn

Giai đoạn này mô hình được huấn luyện trên tập dữ liệu nguồn GTA5 bao gồm 22500 ảnh. Bao gồm 9 fold, mỗi fold là 2500 ảnh. Để lại fold thứ 10 để đánh giá khả năng học của mô hình trên tập dữ liệu nguồn. Ta sử dụng Cross Entropy làm hàm mất mát như đã trình bày ở chương trước (lưu ý là ta dùng ignore label bằng 255 để bỏ qua những nhãn không muốn huấn luyện). Bên cạnh đó, ta còn cần một chiến thuật cho tốc độ học hiệu quả. Ở đây, cả 2 mô hình đều sử dụng hàm điều tiết tốc độ học là hàm mũ có công thức:

$$lr = base * \left(1 - \frac{i}{m}\right)^{power} \quad (4.1)$$

Trong đó,

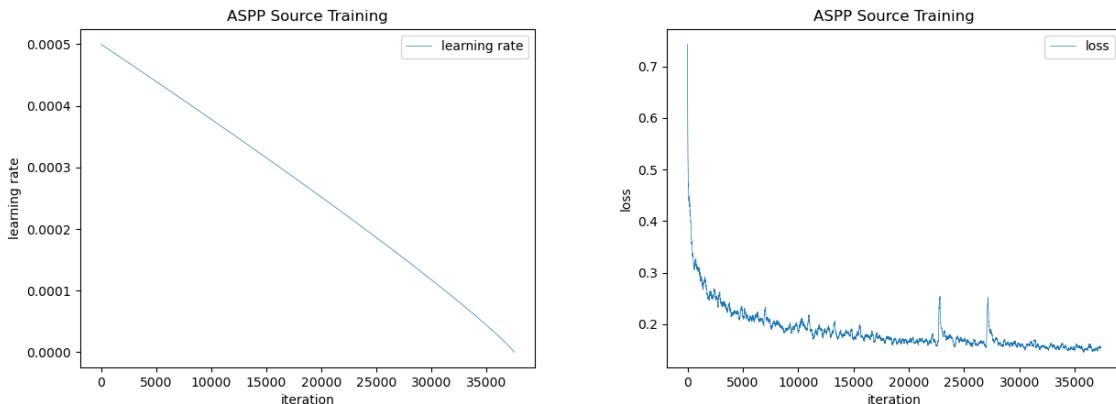
- lr là tốc độ học
- $base$ là tốc độ học khởi tạo

- i là iteration hiện tại
- m là iteration kết thúc
- $power$ là số mũ, ở đây ta chọn $power$ bằng 0.9

Huấn luyện mô hình Resnet101 + DeeplabV2

Hàm mất mát được tối ưu sử dụng thuật toán Stochastic gradient descent(SGD) với tốc độ học (learning rate) khởi tạo cho khối Encoder là 5e-4 và cho khối Decoder là 5e-3 được điều tiết sử dụng hàm mũ 4.1. Thuật toán huấn luyện trên 10 vòng và kích thước một batch là 6 (do kích thước ảnh khá lớn).

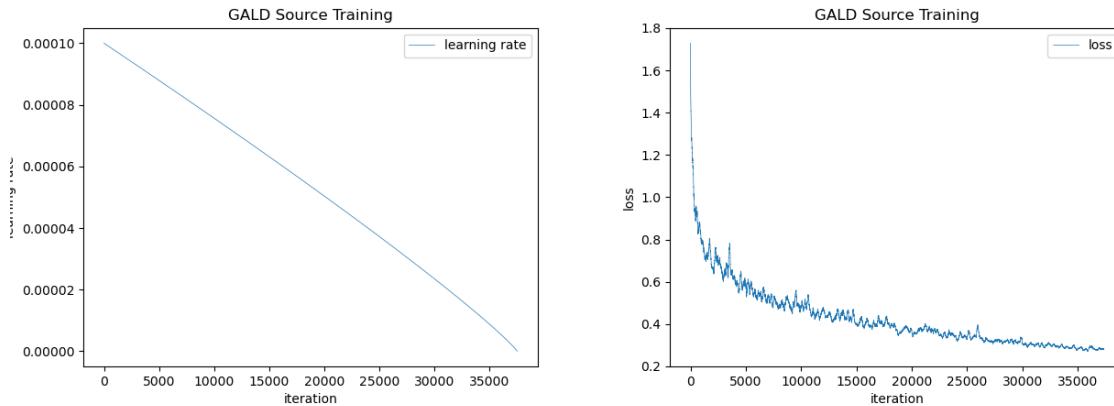
Sau 10 vòng huấn luyện, hàm mất mát từ 1.7370 dần hội tụ đến 0.1498. Hình 4.5



Hình 4.5: Bên trái là kết quả tốc độ học qua các iteration. Bên phải là kết quả hàm mất mát trong quá trình huấn luyện Resnet101 + DeeplabV2 trên tập nguồn. Được đánh giá trên từng iteration, trong đó ta sử dụng trung bình động với kích thước cửa sổ là 150

Huấn luyện mô hình Hardnet68 + GALT

Hàm mất mát được tối ưu sử dụng thuật toán Adam với tốc độ học (learning rate) khởi tạo cho khối Encoder là 0.0001 và cho khối Decoder là 0.001 được điều tiết sử dụng hàm mũ 4.1. Thuật toán huấn luyện trên 10 vòng và kích thước một batch là 6 (do kích thước ảnh khá lớn). Sau 10 vòng huấn luyện, hàm mất mát từ 1.8543 dần hội tụ đến 0.2757. Điều đó cho thấy mô hình này đã học rất tốt với các thông tin từ tập nguồn. Hình 4.6



Hình 4.6: Bên trái là kết quả tốc độ học qua các iteration. Bên phải là kết quả hàm mất mát trong quá trình huấn luyện Hardnet68 + GALD trên tập nguồn. Được đánh giá trên từng iteration, trong đó ta sử dụng trung bình động với kích thước cửa sổ là 150

Huấn luyện mô hình Resnet101 + DeeplabV2 với tập dữ liệu nguồn bổ sung thêm Carla

Ta giữ nguyên cấu hình các tham số như khi huấn luyện mô hình Resnet101 + DeeplabV2 bình thường nhưng trộn thêm dữ liệu được Carla sinh thêm với 22500 ảnh GTA5.

4.3.2 Huấn luyện cạnh tranh trên tập dữ liệu trộn giữa tập nguồn và tập đích

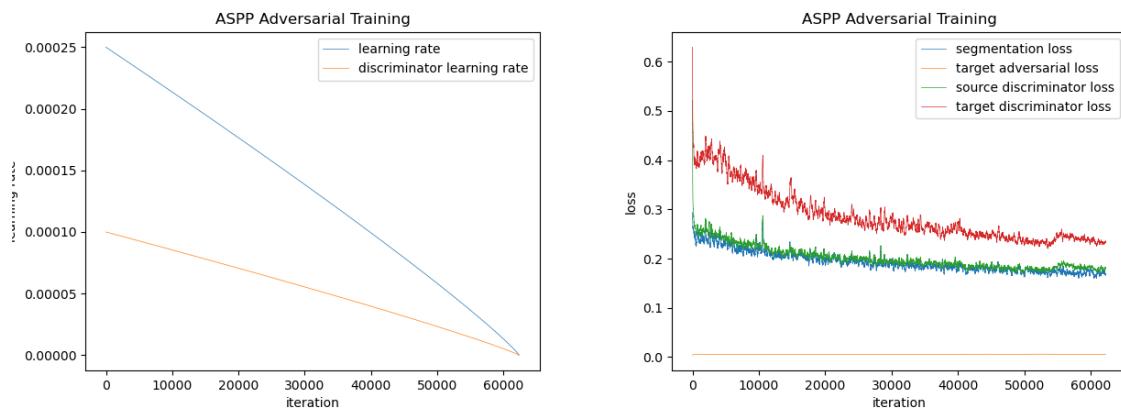
Ta đồng thời khởi tạo cả mô hình cơ sở và Bộ phân loại, cho chúng huấn luyện cùng lúc để cạnh tranh lẫn nhau. Trong đó, mô hình cơ sở vẫn giữ nguyên cấu hình như ở giai đoạn 1. Đối với Bộ phân loại, hàm mất mát được tối ưu sử dụng thuật toán Adam với tốc độ học là 0.0001 vẫn sử dụng hàm mũ để điều tiết. Thuật toán được huấn luyện trên 10 vòng với kích thước batch là 8 (4 ảnh nguồn và 4 ảnh đích). Ta nhắc lại vai trò của 4 hàm mất mát cùng được sử dụng trong giai đoạn này:

- segmentation loss: tính toán sai số giữa kết quả do mô hình cơ sở dự đoán trên tập nguồn.
- target adversarial loss: tính toán mức độ tương đồng giữa "tập đích" với "tập nguồn". Nghĩa là nếu như kết quả Bộ phân biệt dự đoán một ảnh thuộc "tập đích" càng giống với "tập nguồn" thì hàm mất mát này càng thấp.
- source discriminator loss và target discriminator loss: đo khả năng phân biệt một ảnh có thuộc đúng tập hợp của nó hay không. Tức là nếu như Bộ phân biệt dự đoán một ảnh trong đúng tập của nó thì hàm mất mát càng thấp.

Huấn luyện cạnh tranh mô hình Resnet101 + DeeplabV2

Sau 10 vòng huấn luyện:

- segmentation loss từ 0.3263 dần hội tụ về 0.1568
- target adversarial loss không thay đổi nhiều, từ 0.0020 hội tụ về 0.0056.
Nên ta thấy trong Hình 4.7 thì hàm mất mát này gần như bằng 0 vì ta lấy trung bình động.
- source disciminator loss từ 1.3027 hội tụ về 0.1658
- target discriminator loss từ 1.1807 hội tụ về 0.2205



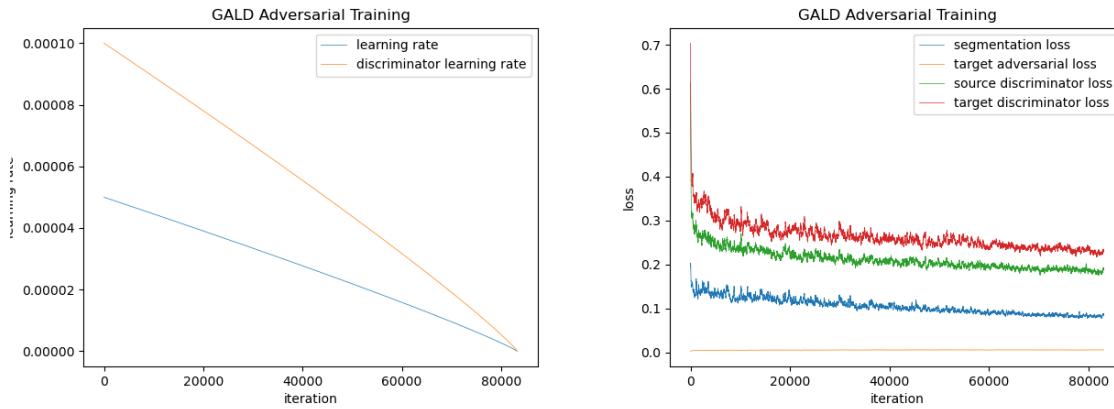
Hình 4.7: Bên trái là kết quả tốc độ học của mô hình cơ sở và bộ phân biệt qua các iteration. Bên phải là kết quả hàm mất mát trong quá trình huấn luyện cạnh tranh Resnet101 + DeeplabV2 trên tập nguồn trộn tập đích. Được đánh giá trên từng iteration, trong đó ta sử dụng trung bình động với kích thước cửa sổ là 150

Sở dĩ giá trị của target adversarial loss lại nhỏ như vậy là do trọng số cho hàm mất mát này tác giả để rất nhỏ, chỉ 0.001

Huấn luyện cạnh tranh mô hình Hardnet68 + GALT

Sau 10 vòng huấn luyện:

- segmentation loss từ 0.1737 dần hội tụ về 0.0796
- target adversarial loss không thay đổi nhiều, từ 0.0023 hội tụ về 0.0054.
Cũng tương tự như mô hình thứ nhất (như Hình 4.8) thì hàm mất mát này gần như bằng 0
- source disciminator loss từ 1.0428 hội tụ về 0.1615
- target discriminator loss từ 0.9669 hội tụ về 0.2058



Hình 4.8: Bên trái là kết quả tốc độ học của mô hình cơ sở và bộ phân biệt qua các iteration. Bên phải là kết quả hàm mất mát trong quá trình huấn luyện cạnh tranh Global Aggregate and Local Distribution trên tập nguồn trộn tập đích. Được đánh giá trên từng iteration, trong đó ta sử dụng trung bình động với kích thước cửa sổ là 150

So với mô hình thứ nhất, ta thấy rằng segmentation loss của mô hình Hardnet68 + GALT thấp hơn. Điều này thể hiện khả năng khớp dữ liệu nguồn của mô hình này rất tốt.

4.3.3 Huấn luyện tự chắt lọc trên tập dữ liệu đích với nhãn giả được sinh từ giai đoạn 2

Trước tiên, từ mô hình thu được ở giai đoạn 2, ta phải sinh ra tập các nhãn giả của bộ dữ liệu huấn luyện Cityscapes (vì mặc định coi như ta không có nhãn thật của Cityscapes). Nhãn giả được sinh ra tương tự minh họa trong Hình 4.4 (chính là kết quả sau giai đoạn 2). Các bước cấu hình huấn luyện và tham số không khác gì ở giai đoạn 1. Chỉ là thay đổi tập dữ liệu huấn luyện từ GTA5 thành tập Cityscapes với nhãn giả. Phương pháp này không làm tăng hiệu năng cho mô hình với kích thước huấn luyện là [1024, 512]. Nhưng khi kiểm thử với kích thước gốc của ảnh trong tập kiểm thử là [2048, 1024] thì kết quả cải thiện lên rất tốt. Sự thay đổi được thể hiện trong Bảng 4.6

4.4 Kết quả thực nghiệm

4.4.1 Sau khi học trên tập nguồn

Ta có thể thấy rằng mô hình Hardnet68 + GALT khớp dữ liệu tốt hơn mô hình Resnet101 + DeeplabV2 do kết quả đánh giá trên tập huấn luyện là GTA5 tốt hơn tới 10 %. Nhưng khi đánh giá trên tập Cityscapes thì mô hình Resnet101 + DeeplabV2 lại tốt hơn 4 %. Điều này cho thấy rằng việc quá khớp với dữ liệu

Bảng 4.4: So sánh kết quả trên mô hình Resnet101 + DeeplabV2 và mô hình Hardnet68 + GALD đánh giá trên fold thứ 10 của tập GTA5 và tập kiểm thử của Cityscapes với kích thước đầu vào là [1024, 512]

Mô hình	Tập dữ liệu GTA5	Tập dữ liệu Cityscapes
Resnet101 + DeeplabV2	67.88	36.93
Hardnet68 + GALD	77.54	32.17
Resnet101 + DeeplabV2 (+Carla)	60.87	37.76

Bảng 4.5: So sánh kết quả mô hình Resnet101 + DeeplabV2 và mô hình Hardnet68 + GALD đánh giá trên tập kiểm thử của Cityscapes với kích thước đầu vào là [1024, 512]

Mô hình	Sau khi huấn luyện nguồn	Sau khi huấn luyện cạnh tranh
Resnet101 + DeeplabV2	36.93	45.39
Hardnet68 + GALD	32.17	42.88
Resnet101 + DeeplabV2 (+Carla)	37.76	44.14

nguồn đã làm cho mô hình Hardnet68 + GALD trở nên kém tổng quát hơn. Trong khi đó, việc thêm dữ liệu Carla vào tập nguồn mặc dù làm giảm đi khả năng học của mô hình Resnet101 + DeeplabV2 nhưng đã giúp đa dạng hóa dữ liệu hơn và đạt được hiệu năng tốt nhất trong 3 hướng tiếp cận. Bảng 4.4

4.4.2 Sau khi học cạnh tranh

Sau khi huấn luyện cạnh tranh thì kết quả trên tập kiểm thử của Cityscapes đều tăng gần 10 % với mỗi mô hình. Ta thấy rằng kết quả ở giai đoạn 1 của mô hình rất quan trọng đối với việc huấn luyện ở giai đoạn 2. Mặc dù mô hình Hardnet68 + GALD đã khớp dữ liệu GTA5 tốt hơn hẳn Resnet101 + DeeplabV2 nhưng lại thua kém khi đánh giá trên tập kiểm thử Cityscapes. Điều đó dẫn đến khi huấn luyện giai đoạn 2, mô hình Hardnet68 + GALD đã không đạt hiệu năng tốt như mô hình Resnet101 + DeeplabV2. Mặc dù, mô hình Resnet101 + DeeplabV2 chỉ tăng 8.46 % còn mô hình Hardnet68 + GALD tăng những 10.71 %. Ta cũng thấy rằng khi thêm dữ liệu Carla vào tập nguồn để huấn luyện thì hiệu năng sau giai đoạn 2 lại không tăng nhiều, chỉ tăng 6.38 %. Bảng 4.5

4.4.3 Sau khi học tự chắt lọc

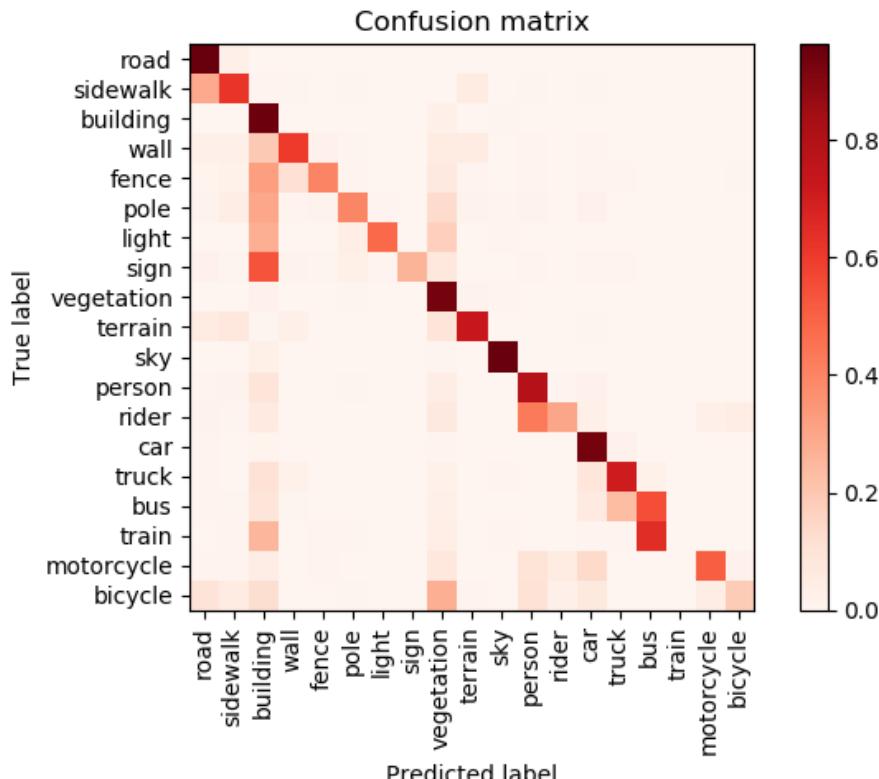
Ta nhận thấy rằng các phương pháp tiếp cận mới mặc dù làm tăng hiệu năng của mô hình trên tập GTA5 và trên tập kiểm thử của Cityscapes khi ở giai đoạn 1. Nhưng ở giai đoạn 2, thì mô hình Resnet101 + DeeplabV2 vẫn đạt hiệu quả cao nhất. Do đó, ta dùng mô hình này như một "giáo viên" để sinh "nhân giả" cho các mô hình "sinh viên" học tập. Khi đó, mô hình "sinh viên" Hardnet68 + GALD đạt được hiệu năng cao nhất là 48.9 %. Sau khi áp dụng các hướng tiếp

cận trên, ta thu được kết quả như trong Bảng 4.6

Bảng 4.6: Bảng so sánh kết quả các mô hình trên 2 tập dữ liệu GTA5 và Cityscapes

	GTA5		Cityscapes	
	Kích thước (1024, 512)	Kích thước (2048, 1024)	Kích thước (1024, 512)	Kích thước (2048, 1024)
Resnet101 + DeeplabV2	0.6788	0.6980	0.3693	0.3201
Resnet101 + DeeplabV2 (+Carla)	0.6087	0.6176	0.3776	0.3515
Hardnet68 + GALD	0.7754	0.7859	0.3217	0.2703
Resnet101 + DeeplabV2 + Adv	0.6839	0.7129	0.4539	0.4305
Resnet101 + DeeplabV2 + Adv(+Carla)	0.6566	0.6312	0.4414	0.4323
Hardnet68 + GALD + Adv	0.6723	0.6428	0.4288	0.4112
Resnet101 + DeeplabV2 + Distill	0.6639	0.6514	0.4535	0.4842
Hardnet68 + GALD + Distill	0.7428	0.7219	0.4546	0.4890

mIoU cho mô hình Hardnet68 + GALD với kích thước đầu vào [2048, 1024] là 48.9 %. Ta minh họa kết quả bằng một confusion matrix như Hình 4.9



Hình 4.9: Minh họa Confusion Matrix cho kết quả của mô hình Hardnet68 + GALD

Ta có một số nhận xét:

- (Hàng 2) sidewalk thường bị nhầm thành road
- (Cột 3) nhiều lớp bị nhầm với lớp building, do mô hình không có khả năng trích xuất các vật thể nhỏ (light, pole, sign,...) khỏi nền có building
- Giao điểm của hàng 12-13 (person - rider) với cột 12-13 (person - rider):

phân loại sai của mô hình giữa hai lớp này là dễ hiểu vì chúng không khác nhau trong thực tế.

Chương 5

Kết luận và hướng phát triển

Sự phát triển nghiên cứu về công nghệ cho xe tự lái là một xu hướng trong tương lai gần. Các lĩnh vực nghiên cứu thị giác máy tính ngày càng đóng vai trò quan trọng hơn. Do đó, chúng nhận được sự quan tâm lớn từ cộng đồng khoa học nói chung và nghiên cứu trí tuệ nhân tạo nói riêng.

Docket này đã trình bày tương đối đầy đủ về bài toán thích ứng miền trong phân vùng ảnh ngữ nghĩa. Tác giả đã trình bày các khái niệm chính cũng như cách thực hiện các bài toán con, bao gồm:

- Bài toán phân vùng ảnh ngữ nghĩa
- Bài toán thích ứng miền
- Bài toán học tự chất lọc

Tác giả cũng đã trình bày cách giải quyết đối với từng bài toán, cũng như thử nghiệm các phương pháp cải tiến, tối ưu. Ngoài ra, tác giả cũng đưa ra ưu nhược điểm của từng thuật toán. Do thời gian nghiên cứu có hạn, bài báo cáo có thể còn có sai sót, tác giả mong nhận được các góp ý để bài báo cáo được tốt hơn.

Trong thực tế, ta cần cân bằng giữa cả độ chính xác và mức độ đa dạng mà mô hình có thể nhận biết được. Cho nên, do án đã xây dựng một mô hình cơ sở, từ đó có thể dễ dàng thử nghiệm thêm các mô đun phân vùng ảnh ngữ nghĩa, các kỹ thuật thích ứng miền khác nhau và các tổ hợp của chúng. Trong tương lai, tác giả sẽ thử nghiệm thêm các kỹ thuật thích ứng miền mạnh mẽ hơn như IAST và các kiến trúc nổi bật như Transformer trên bộ dữ liệu được mô phỏng từ đồ họa máy tính với các kiểu môi trường đa dạng.

Tài liệu tham khảo

- [1] Yoshua Bengio and Daniel Faggella. The rise of neural networks and deep learning in our everyday lives – a conversation with yoshua bengio, 2019.
- [2] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.
- [3] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. Hardnet: A low memory traffic network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3552–3561, 2019.
- [4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.
- [5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset. In *CVPR Workshop on the Future of Datasets in Vision*, volume 2, 2015.
- [6] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.
- [7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative ad-

- versarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [12] Hanzhang Hu, Debadatta Dey, Allison Del Giorno, Martial Hebert, and J Andrew Bagnell. Log-densenet: How to sparsify a densenet. *arXiv preprint arXiv:1711.00002*, 2017.
- [13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.
- [14] Phil Kim. Convolutional neural network. In *MATLAB deep learning*, pages 121–147. Springer, 2017.
- [15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [16] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.
- [17] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4500–4509, 2018.
- [18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [19] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [20] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the*

- IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [21] David Silver and Demis Hassabis. Alphago: Mastering the ancient game of go with machine learning. *Research Blog*, 9, 2016.
 - [22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
 - [23] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sébastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 240–248. Springer, 2017.
 - [24] Tijmen Tieleman and Geoffrey Hinton. Rmsprop gradient optimization. 2014.
 - [25] Haoran Wang, Tong Shen, Wei Zhang, Ling-Yu Duan, and Tao Mei. Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In *European Conference on Computer Vision*, pages 642–659. Springer, 2020.
 - [26] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
 - [27] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018.