

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO MÔN HỌC
ĐỒ ÁN 2

ỨNG DỤNG DEEP LEARNING
TRONG BÀI TOÁN PHÂN ĐOẠN ĐỐI TƯỢNG
TRÊN ẢNH VỆ TINH

Giảng viên hướng dẫn: TS. Lê Hải Hà

Học viên thực hiện: Nguyễn Đức Thắng

Mã số sinh viên: 20166769

HÀ NỘI, 06/2020

Mục lục

1	Tổng quan về bài toán	2
2	Cơ sở lý thuyết	4
2.1	Convolution Neural Network (CNN)	4
2.1.1	Cấu trúc mạng CNN	4
2.1.2	Convolution layer	5
2.1.3	Pooling layer	7
2.1.4	Fully connected layer	8
2.2	Transfer learning	9
2.2.1	Feature extractor	9
2.2.2	Fine tuning	10
2.3	R-CNN	12
2.3.1	Selective search	14
2.3.2	Bounding box regression	15
2.3.3	Intersection over Union	16
2.4	Fast R-CNN	18
2.4.1	RoI pooling	19
2.4.2	Hàm mất mát của Fast R-CNN	21
2.5	Faster R-CNN	22
2.5.1	Region Proposal Network	23
2.5.2	Non-maxima suppression	25
2.6	Mask R-CNN	26
2.6.1	RoiAlign	28
3	Xây dựng mô hình và kết quả thực nghiệm	30
3.1	Xử lý dữ liệu	30
3.2	Huấn luyện mô hình	33
3.3	Kết quả thực nghiệm	35

Lời mở đầu

Trong những năm gần đây, trí tuệ nhân tạo ngày càng phát triển và đi vào cuộc sống hàng ngày. Trong đó, thị giác máy tính (computer vision) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ảnh và video. Trong trí tuệ nhân tạo thì thị giác máy tính là một trong những phần khó. Các bài toán ứng dụng của thị giác máy tính bao gồm: phân loại đối tượng trong ảnh, phát hiện đối tượng, chú thích cho ảnh...

Phân đoạn đối tượng là một trong những bài toán của thị giác máy tính, có vai trò quan trọng trong nhiều ứng dụng thực tiễn. Trong báo cáo này, với mục đích nghiên cứu về phân đoạn đối tượng trong thị giác máy tính, em thực hiện bài toán phân đoạn cảnh đồng trên ảnh vệ tinh.

Nội dung báo cáo này được chia làm 3 chương:

- **Chương 1:** Tổng quan về các bài toán trong thị giác máy tính và bài toán phân đoạn các vùng là cảnh đồng trên ảnh viễn thám.
- **Chương 2:** Các kiến thức nền tảng, cơ sở lý thuyết liên quan đến bài toán.
- **Chương 3:** Xây dựng mô hình và đánh giá kết quả thực nghiệm trên mô hình đã chọn. Và cuối cùng là kết luận và đề xuất các hướng phát triển.

Chân thành cảm ơn TS. Lê Hải Hà đã hướng dẫn em thực hiện đồ án này và cung cấp dữ liệu để em có thể hoàn thành đề tài. Mặc dù đã cố gắng rất nhiều, nhưng với kiến thức và thời gian hạn chế nên không thể tránh khỏi những thiếu sót. Rất mong được sự góp ý của thầy cô và bạn bè để báo cáo này được hoàn thiện hơn.

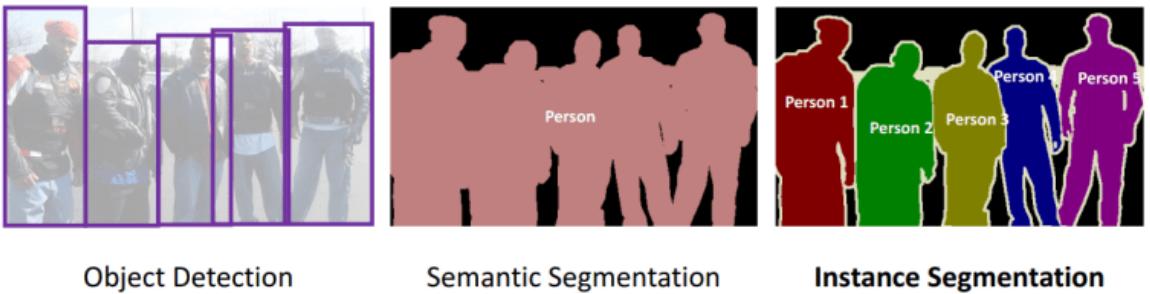
Chương 1

Tổng quan về bài toán

Thị giác máy tính (Computer vision) - một nhánh của trí tuệ nhân tạo đang ngày càng phát triển trong những năm gần đây. Thị giác máy tính đang dần đi vào cuộc sống thực tế với những bài toán thực tế có tính áp dụng cao. Trong thị giác máy tính, có những bài toán cơ bản sau:

- Phân loại hình ảnh (image classification): liên quan đến việc gán nhãn, phân loại cho một hình ảnh.
- Định vị vật thể (object localization): liên quan đến việc vẽ một hộp giới hạn (bounding box) xung quanh một hoặc nhiều đối tượng trong hình ảnh nhằm khoanh vùng đối tượng.
- Phát hiện đối tượng (object detection): Là nhiệm vụ khó khăn hơn và là sự kết hợp của cả hai nhiệm vụ trên: Vẽ một bounding box xung quanh từng đối tượng quan tâm trong ảnh và gán cho chúng một nhãn. Kết hợp cùng nhau, tất cả các vấn đề bày được gọi là object recognition hoặc object detection.
- Phân đoạn đối tượng (object segmentation): Phân loại đối tượng trên pixel ảnh, xác định những pixel nào thuộc đối tượng. Trong phân đoạn đối tượng được chia thành 2 loại là: semantic segmentation và instance segmentation. Trong đó semantic segmentation thực hiện phân đoạn với từng lớp khác nhau, ví dụ tất cả người là 1 lớp, tất cả ô tô là một lớp. Instance segmentation là thực hiện phân đoạn với từng đối tượng trong một lớp. Ví dụ có 2 người trong ảnh thì sẽ có 3 vùng phân đoạn khác nhau cho mỗi người.
- Chú thích cho ảnh (image captioning): Dưa ra lý giải về hành động và nội dung của một bức ảnh hoặc chuỗi các bức ảnh, video.

Các bài toán trong thị giác máy tính còn được tổ chức thành các cuộc thi với giải thưởng cao, thúc đẩy sự nghiên cứu và phát triển của ngành. Thị giác máy tính đã đạt được nhiều thành tựu đột phá trong những năm qua cùng với tốc độ và độ chính xác



Hình 1.1: Một số bài toán trong thị giác máy tính

ngày càng cao ở các bài toán. Hình 1.1 cho thấy một số bài toán trong thị giác máy tính.

Với tốc độ phát triển hiện nay, dữ liệu của chúng ta ngày càng nhiều và các bài toán bắt đầu khó dần lên nên những mô hình truyền thống tỏ ra kém hiệu quả. Các đặc trưng (feature) lấy ra từ thị giác máy tính truyền thống như HOG, SIFT, LBP chỉ lấy được những đặc trưng trên bề mặt nổi của ảnh. Chính vì thế, cần những mô hình lấy được những đặc trưng ẩn sâu hơn trong ảnh. Và vì thế, học sâu (Deep learning) đang ngày càng phát triển các thuật toán hơn để xử lý các bài toán của thị giác máy tính.

Việc áp dụng deep learning vào những bài toán xử lý ảnh vệ tinh là những bài toán khó. Vì ảnh vệ tinh có độ nét thường rất cao, dung lượng lớn và khó xử lý hơn các ảnh thông thường. Dữ liệu ảnh vệ tinh đang ngày càng lớn dần đặt ra những thách thức hơn trước. Trong báo cáo này, tôi áp dụng Deep learning vào bài toán trích xuất cánh đồng, instance segmentation các cánh đồng trên ảnh vệ tinh.



Hình 1.2: Bài toán trích xuất các vùng là cánh đồng trên ảnh vệ tinh

Hình 1.2 bên trái biểu thị đầu vào của bài toán và bên phải biểu thị đầu ra của bài toán này. Báo cáo tập trung vào việc xây dựng mô hình để xử lý bài toán trên hướng đến mục tiêu độ chính xác cao. Mô hình được xây dựng trong báo cáo này cũng có thể mở rộng và áp dụng cho các bài toán tương tự.

Chương 2

Cơ sở lý thuyết

2.1 Convolution Neural Network (CNN)

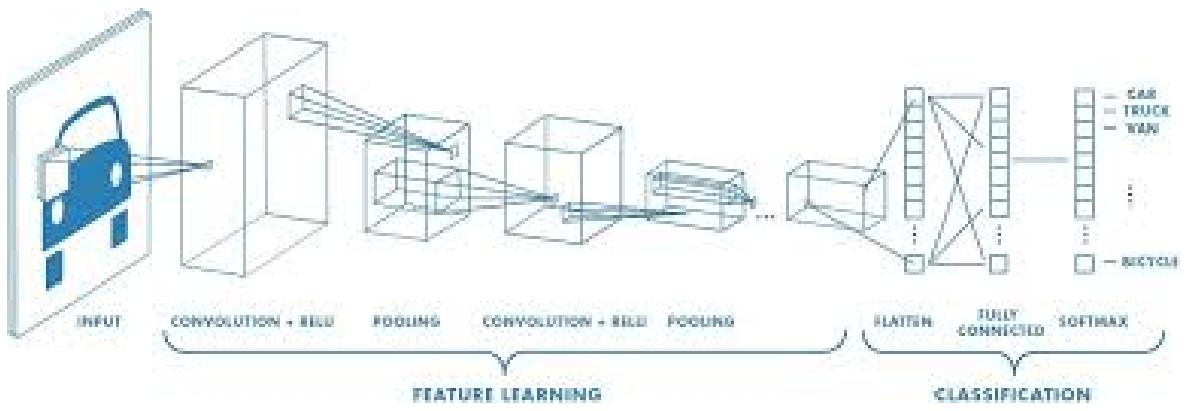
Mạng Nơ-ron Tích Chập lấy cảm hứng từ não người. Nghiên cứu trong những thập niên 1950 và 1960 của D.H Hubel và T.N Wiesel trên não của động vật đã đề xuất một mô hình mới cho việc cách mà động vật nhìn nhận thế giới. Trong báo cáo, hai ông đã diễn tả 2 loại tế bào nơ-ron trong não và cách hoạt động khác nhau: tế bào đơn giản (simple cell – S cell) và tế bào phức tạp (complex cell – C cell).

Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó 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. CNN được sử dụng nhiều trong các bài toán nhận dạng các đối tượng trong ảnh.

2.1.1 Cấu trúc mạng CNN

Mạng CNN là một tập các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation (không tuyến tính) như ReLU và tanh là hàm kích hoạt. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình CNN, các layer của mạng liên kết với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ.

Mạng CNN có kiến trúc khác với Mạng Nơ-ron thông thường. Mạng Nơ-ron bình thường chuyển đổi đầu vào thông qua hàng loạt các tầng ẩn. Mỗi tầng là một tập các nơ-ron và các tầng được liên kết đầy đủ với các nơ-ron ở tầng trước đó. Và ở tầng cuối cùng sẽ là tầng kết quả đại diện cho dự đoán của mạng.



Hình 2.1: Cấu trúc mạng CNN

CNNs gồm 2 thành phần:

- **Phản tầng ẩn hay phản rút trích đặc trưng:** trong phần này, mạng sẽ tiến hành tính toán hàng loạt phép tích chập (convolution) và phép hợp nhất (pooling) để phát hiện các đặc trưng. Ví dụ: nếu ta có hình ảnh con chó thì phần này sẽ nhận dạng mõm, lông và tai, mắt của nó.
- **Phản phân lớp:** tại phần này, một lớp với các liên kết đầy đủ sẽ đóng vai trò như một bộ phân lớp các đặc trưng đã rút trích được trước đó. Tầng này sẽ đưa ra xác suất của một đối tượng trong hình.

2.1.2 Convolution layer

Convolution layer thường là lớp đầu tiên trong mô hình CNN. Lớp này có chức năng phát hiện ra các đặc trưng về không gian một cách hiệu quả.

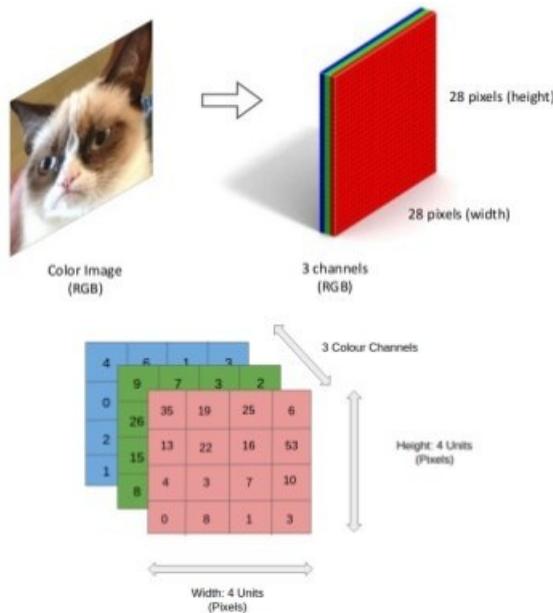
Tensor

Khi dữ liệu biểu diễn dạng 1 chiều, ta gọi là **vector**, mặc định khi viết vector sẽ viết dưới dạng cột. Khi dữ liệu dạng 2 chiều, ta gọi là **ma trận**, kích thước là số hàng \times số cột. Khi dữ liệu nhiều hơn 2 chiều thì sẽ gọi là **tensor**. Có thể hiểu tensor là sự kết hợp của các ma trận cùng kích thước, xếp k ma trận kích thước $m \times n$ lên nhau sẽ được tensor kích thước $m \times n \times k$. Ảnh màu RGB là một tensor 3 chiều (Hình 2.2).

Filter

Filter là một trong những tham số quan trọng trong CNN. Kích thước filter trong tầng convolution phổ biến hiện nay là 3x3. Kích thước filter thường là số lẻ, ví dụ 3x3, 5x5, 7x7.

Kích thước của các Filter thường không quá lớn. Vì với kích thước nhỏ nó có thể trích xuất cục bộ chi tiết hơn, kích thước ảnh giảm chậm hơn; làm cho mạng sâu hơn và số lượng tham số phải học thấp hơn.



Hình 2.2: Ảnh màu RGB là một tensor 3 chiều

Ta thực hiện phép tích chập bằng cách trượt kernel/filter theo dữ liệu đầu vào. Tại mỗi vị trí, ta tiến hành phép nhân ma trận và tính tổng các giá trị để đưa vào bản đồ đặc trưng.

Với ảnh RGB có 3 channel red, green, blue thì Filter phải có cùng độ sâu với ảnh (depth). Và nó di chuyển từ trái sang phải, từ trên xuống dưới.

Padding

Khi dùng convolution, thông tin ở biên bức ảnh bị biến mất và kích thước của ảnh giảm nhanh chóng (Hình 2.3).

Ảnh 3x3		
0	4	2
1	1	0
3	4	5

*

Filter 3x3		
0	1	0
0	1	0
0	1	0

=

Feature map 1x1		
9		

Hình 2.3: Kích thước ảnh giảm nhanh chóng khi dùng convolution

Để khắc phục vấn đề này, chúng ta sử dụng **padding**. Bằng việc thêm các giá trị 0 vào biên, ta sẽ có **zero padding** (Phần màu tím hình 2.4).

Ảnh 3x3 được thêm pad	Filter 3x3	Feature map 3x3																																											
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>4</td><td>2</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>3</td><td>4</td><td>5</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	4	2	0	0	1	1	0	0	0	3	4	5	0	0	0	0	0	0	$*$ <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> </table>	0	1	0	0	1	0	0	1	0	$=$ <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>1</td><td>5</td><td>2</td></tr> <tr><td>4</td><td>9</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> </table>	1	5	2	4	9	7	4	5	6
0	0	0	0	0																																									
0	0	4	2	0																																									
0	1	1	0	0																																									
0	3	4	5	0																																									
0	0	0	0	0																																									
0	1	0																																											
0	1	0																																											
0	1	0																																											
1	5	2																																											
4	9	7																																											
4	5	6																																											

Hình 2.4: Thêm padding cho ảnh

Sau khi thêm padding chúng ta có một số lợi thế như sau:

- Không mất mát thông tin viền nên nhận diện sẽ tốt hơn, tìm được chính xác đối tượng hơn.
- Đầu ra của CNN kích thước sẽ giảm dần nên khi thêm padding sẽ giúp giảm chậm hơn.

Stride

Đối với phép convolution hay pooling thì stride (S) là độ dài bước trượt của filter. Như trong hình 2.5, độ dài bước trượt $S = 2$.



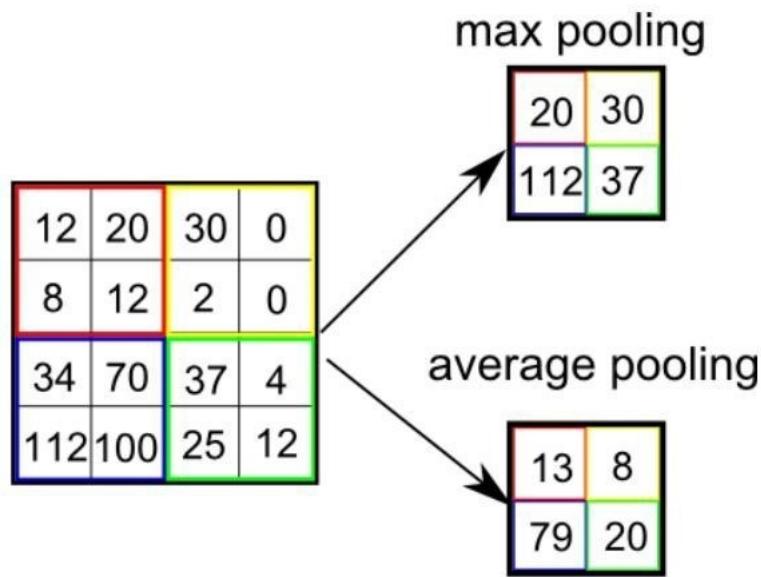
Hình 2.5: Minh họa cho stride

2.1.3 Pooling layer

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong mô hình.

Có 2 loại pooling thường được sử dụng trong CNN:

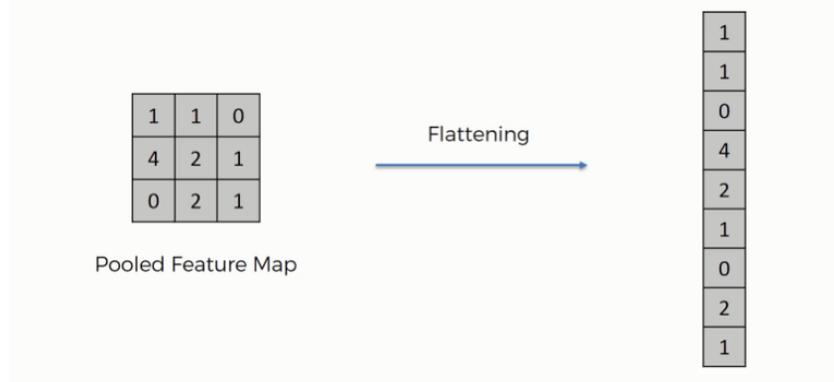
- **Max Pooling:** Thực hiện lấy giá trị lớn nhất trong kích thước filter mà ta xét.
- **Average Pooling:** Thực hiện lấy giá trị trung bình tổng trong kích thước filter ta xét.



Hình 2.6: Hoạt động của pooling

2.1.4 Fully connected layer

Trong phần phân lớp, ta sử dụng một vài tầng với kết nối đầy đủ (Fully connected layer) để xử lý kết quả của phần tích chập. Vì đầu vào của mạng liên kết đầy đủ là 1 chiều, ta cần làm phẳng đầu vào trước khi phân lớp (Hình 2.7). Tầng cuối cùng trong mạng CNN là một tầng liên kết đầy đủ, phần này hoạt động tương tự như mạng nơ-ron thông thường. Kết quả thu được cuối cùng cũng sẽ là một véc-tơ với các giá trị xác suất cho việc dự đoán như mạng nơ-ron thông thường.



Hình 2.7: Làm phẳng đầu vào ảnh

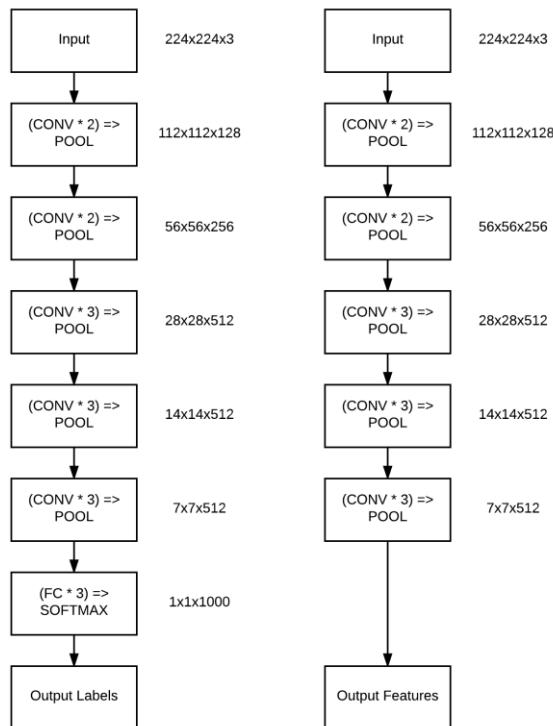
2.2 Transfer learning

Transfer learning là cách để các mô hình truyền đạt cho nhau khả năng mà mỗi mô hình có thể làm được. Trong transfer learning, một mô hình đã huấn luyện (pretrained model) trên một bài toán cụ thể nào đó thực hiện một nhiệm vụ hoặc nhiều nhiệm vụ (gọi là *source tasks*). Mô hình mới sử dụng một phần hay toàn bộ pretrained model để học một bài toán khác có thể cùng mục đích hoặc không cùng mục đích với pretrained model đã được huấn luyện (gọi là *target tasks*). Tuỳ vào nhiệm vụ của mỗi layer mà mô hình mới có thể thêm các layer khác dựa trên pretrained model sẵn có.

Có 2 loại transfer learning: Feature extractor và Fine tuning

2.2.1 Feature extractor

Feature extractor là một phần của model dùng để trích xuất ra features nói chung. Chúng ta có thể sử dụng Feature Extractor đã được huấn luyện để trích xuất ra các đặc trưng của mô hình thay vì phải tạo ra một Feature Extractor mới và huấn luyện lại từ đầu. Những đặc trưng này sau đó được dùng làm đầu vào của những thuật toán học máy hoặc logistic regression để học và dự đoán các kết quả từ những đặc trưng được tạo ra ở bước trên.



Hình 2.8: Bên trái là mô hình VGG16, bên phải là mô hình VGG16 chỉ bao gồm ConvNet với đầu ra là các đặc trưng - Feature Extractor.

2.2.2 Fine tuning

Để sử dụng pretrained model hiệu quả, chúng ta cần:

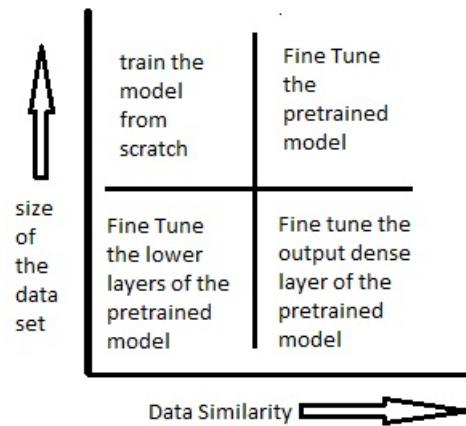
- Thêm các layer phù hợp với bài toán của chúng ta, loại bỏ các layer của pretrained model mà chúng ta không dùng đến, đây là một vấn đề khó cần phải có những nghiên cứu chuyên sâu về từng layer và mục đích của chúng.
- Có chiến lược huấn luyện tốt, vì nếu huấn luyện không tốt sẽ làm mất đi tính hiệu quả của pretrained model và do đó giảm khả năng của mô hình.

Do đó, *fine tuning* ra đời nhằm giúp chúng ta có chiến lược huấn luyện hiệu quả. Fine tuning là việc huấn luyện một mô hình transferred nhằm mục đích tối ưu độ chính xác của mô hình trên mục tiêu bài toán.

Dưới đây là một số chiến lược fine tuning thường dùng (hình 2.9):

- Khi dữ liệu cho target task *lớn và tương tự* với dữ liệu source tasks: Đây là trường hợp lý tưởng, có thể dùng các trọng số (weights) của pretrained model để khởi tạo cho phần pretrained.
- Khi dữ liệu cho target task *nhỏ và tương tự* với dữ liệu source tasks: Vì dữ liệu nhỏ, nếu huấn luyện lại phần pretrained sẽ dẫn đến overfitting, do đó chúng ta chỉ huấn luyện những layer được thêm vào với phần weights khởi tạo cho pretrained như trên.
- Khi dữ liệu cho target tasks *lớn và khác biệt* với dữ liệu source tasks: bởi vì dữ liệu của chúng ta có sự khác biệt nên khi dùng weights từ pretrained model sẽ làm giảm độ chính xác vì sự khác biệt trong tasks và dataset, nhưng cũng chính vì dataset lớn nên việc train toàn bộ transferred model từ đầu là hiệu quả nhất, giúp cho model thích nghi tốt hơn với dữ liệu này.
- Khi dữ liệu cho target tasks *nhỏ và khác biệt* với dữ liệu source tasks: đây là trường hợp khó khăn nhất, chúng ta có thể can thiệp vào pretrained model, thay thế những pretrained layer xa input để thích nghi với dữ liệu mới nhưng không được train các layer gần input của pretrained vì dữ liệu nhỏ sẽ không thể train được các layer này hiệu quả và các layer này chỉ trích xuất các features tổng quát từ dữ liệu, sẽ không ảnh hưởng đến target task.

Transfer Learning mang đến những model mới với độ chính xác cao trong thời gian ngắn, tuy nhiên, nó cũng không phải là một kỹ thuật dễ sử dụng. Nếu sai sót thì có thể gây ra kết quả tệ hơn cả ban đầu. Hầu hết các model dùng transfer learning được sử dụng trong các nghiên cứu về Computer Vision (CV), chú trọng vào việc trích xuất các features từ ảnh hoặc video một cách hiệu quả như một cách thay thế cho các phương pháp cũ (AKAZE, ORB, BRISK, ...) và kết hợp những ý tưởng mới để tận dụng các features này (Object Detection, Object Recognition, Human Pose Estimation, ...).

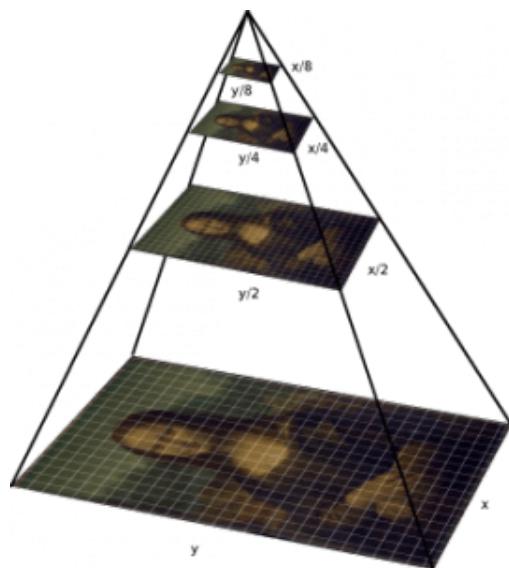


Hình 2.9: Phân loại chiến lược fine tuning

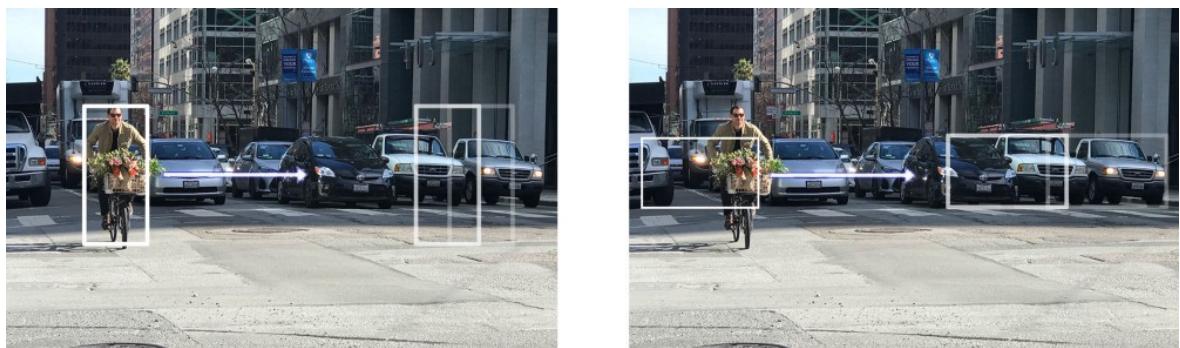
Transfer Learning cũng được sử dụng rất nhiều trong Natural Language Processing (NLP). Trên thực tế thì: nếu CV dùng Convolutional Network để trích xuất các features từ ảnh thì NLP dùng Word Embeddings như một cách để trích xuất các features từ các từ thành những vectors. Hiệu quả thực tiễn của Word Embeddings cao hơn hẳn one-hot encodings về khả năng biểu diễn thông tin.

2.3 R-CNN

Bắt đầu từ năm 2012, sau khi mạng AlexNet giành giải nhất cuộc thi 2012 ILSVRC, mọi nghiên cứu về phân lớp dữ liệu đều sử dụng mạng CNN. Kể từ đó đến năm 2014 là những khoảng trầm trong lĩnh vực thị giác máy tính, CNN khi đó được coi như là thuật toán thống trị trên mọi publish paper về các bài toán phân lớp đối tượng. Trong khi đó, để nhận dạng đối tượng trong ảnh, cách đơn giản nhất là thiết lập một cửa sổ trượt có kích thước (window size) trượt từ trái qua phải, từ trên xuống dưới, quét qua toàn bộ bức ảnh. Để phát hiện các đối tượng khác nhau có kích thước khác nhau ở các góc nhìn khác nhau, chúng ta phải sử dụng cửa sổ trượt có kích thước thay đổi và ảnh đầu vào có kích thước thay đổi. Hình 2.10 biểu diễn các cách thay đổi kích thước ảnh đầu vào, và hình 2.11 cho thấy cách cửa sổ trượt qua ảnh.



Hình 2.10: Thay đổi kích thước ảnh đầu vào



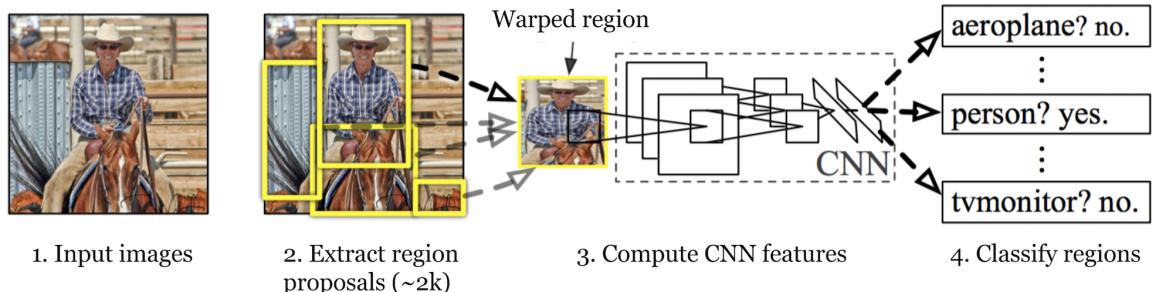
Hình 2.11: Trượt cửa sổ trên ảnh

Các phần của bức ảnh được cắt dựa vào window size sau đó được resize và đem phân lớp qua mạng CNN để trích rút đặc trưng rồi dùng một thuật toán phân lớp như

svm, logistic regression để xác định đối tượng trong bức ảnh là gì. Rõ ràng là phương pháp này hoạt động kém hiệu quả và chậm, ngoài ra thì nhiều phương pháp các thời kỳ ấy cũng chưa thực sự mạnh mẽ. Từ đó, R-CNN được nghiên cứu và ra đời và là nền móng cho nhiều thuật toán phát hiện đối tượng sau này.

R-CNN là viết tắt của "Region-based Convolutional Network". Ý tưởng chính của R-CNN trải qua 2 bước:

- 1 Dùng thuật toán Selective Search để lấy ra khoảng 2000 bounding box.
- 2 Với mỗi bounding box, ta xác định xem nó là đối tượng nào.

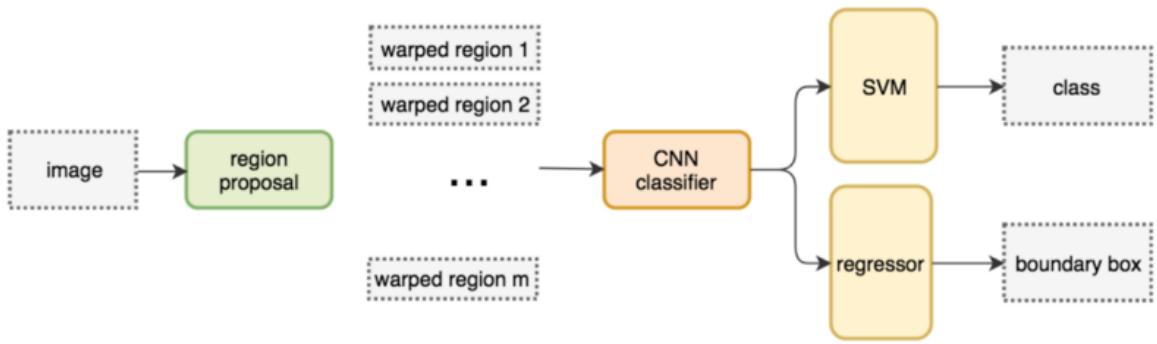


Hình 2.12: Pipeline mạng R-CNN

Luồng xử lý của R-CNN có thể được tóm tắt như sau:

- 1 Huấn luyện (hoặc sử dụng pre-train) một mô hình CNN phân loại (Thời điểm R-CNN ra đời thì AlexNet đang là lựa chọn tốt nhất)
- 2 Sử dụng Selective Search để tìm kiếm các region proposal trên hình ảnh đầu vào (khoảng 2000 vùng đề xuất trên mỗi ảnh). Các vùng này có thể có đối tượng hoặc không và chúng có kích thước khác nhau.
- 3 Các region proposal được **warped** để đưa về kích thước cố định mà mạng CNN yêu cầu.
- 4 Thực hiện việc tranfer learning mạng CNN cho $K + 1$ lớp; trong đó K là số đối tượng chúng ta cần phân lớp và bổ sung thêm 1 lớp gọi là *background*.
- 5 Với mỗi region proposal, thực hiện việc lan truyền qua mạng CNN sẽ tạo ra một vector đặc trưng. Vector đặc trưng này sau đó được SVM phân lớp.
- 6 Sử dụng **bounding box regression** để tinh chỉnh vị trí các vùng đề xuất. Đối với mỗi lớp, huấn luyện 1 mô hình hồi quy để xác định xem hộp có được tối ưu hay không.

Với việc sử dụng ít tấm ảnh nhỏ hơn, và chất lượng của mỗi tấm ảnh nhỏ tốt hơn, Mạng R-CNN chạy nhanh hơn và có độ chính xác cao hơn so với mô hình sử dụng cửa sổ trượt.



Hình 2.13: Mô phỏng mô hình R-CNN

2.3.1 Selective search

Thuật toán selective search đóng vai trò quan trọng trong việc đề xuất vùng dựa trên đầu vào là các phần phân đoạn. Đầu vào của thuật toán là ảnh màu, đầu ra là khoảng 2000 region proposal (bounding box) mà có khả năng chứa các đối tượng.

Đầu tiên ảnh được phân đoạn (segment) qua thuật toán *Graph based image segmentation*, thuật toán dựa trên đồ thị và không sử dụng deep learning.



Hình 2.14: Thuật toán graph based image segmentation

Trong thuật toán selective search, chúng ta bắt đầu bằng cách xem các pixel là mỗi nhóm, lần lượt các bước lặp tiếp theo, chúng ta sẽ tính khoảng cách ngữ nghĩa (dựa vào các tiêu chí màu sắc, kết cấu, kích thước và hình dạng) giữa các nhóm và gom các nhóm có khoảng cách gần nhất với nhau về chung một nhóm để tìm ra phân vùng có khả năng cao nhất chứa đối tượng (ưu tiên gom những nhóm nhỏ trước). Hình 2.15 mô phỏng hoạt động của thuật toán selective search qua các bước.



Hình 2.15: Thuật toán Selective search

2.3.2 Bounding box regression

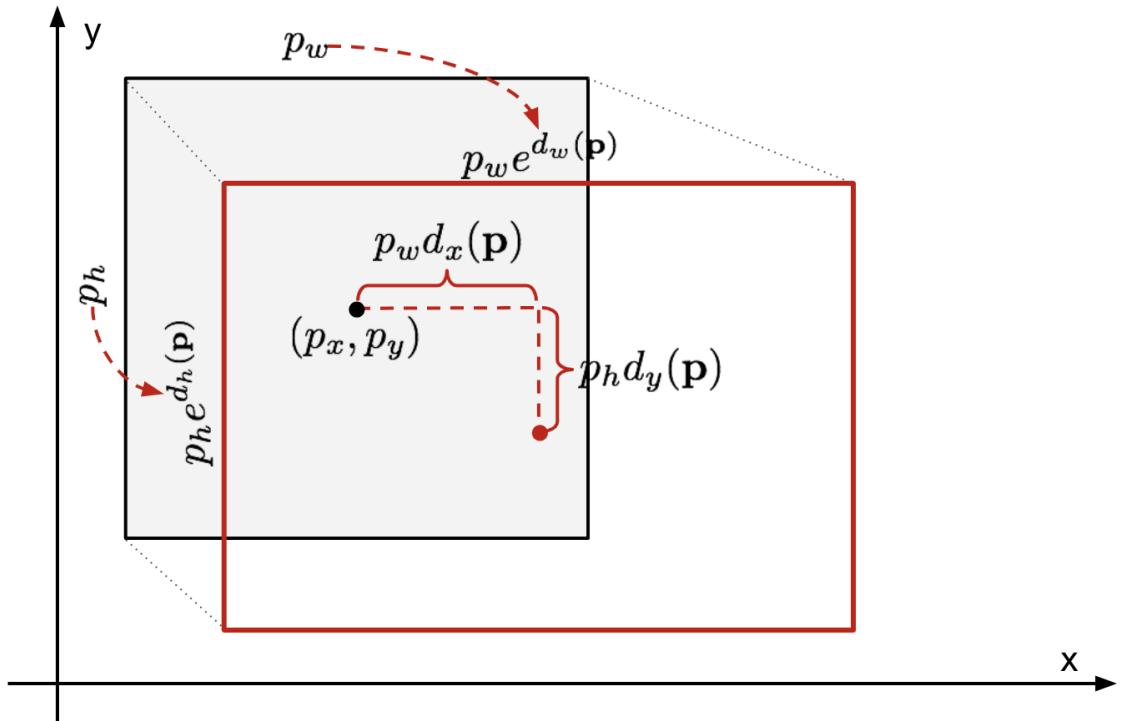
Các tác giả sử dụng bounding box regression để cải thiện hiệu suất. Sau khi dự đoán cho mỗi vùng với 1 class riêng bằng SVM, các tác giả đưa dùng thêm một tầng hồi quy để tinh chỉnh lại các bounding box dự đoán.

Dầu vào của thuật toán là một tập N cặp (p, g) với p là box dự đoán và g là box thật, box dự đoán có toạ độ $p = (p_x, p_y, p_w, p_h)$ (Toạ độ trung tâm, chiều rộng, chiều cao), box thật $g = (g_x, g_y, g_w, g_h)$. Mục tiêu của chúng ta là cần học một chuyển đổi ánh xạ từ box dự đoán p thành một box thật (ground-truth) g .

Chúng ta tham số hoá chuyển đổi trong 4 hàm: $d_x(p)$, $d_y(p)$, $d_w(p)$, $d_h(p)$. Với mỗi đầu vào p , thực hiện chuyển đổi như sau:

$$\begin{aligned}\hat{g}_x &= p_w d_x(\mathbf{p}) + p_x \\ \hat{g}_y &= p_h d_y(\mathbf{p}) + p_y \\ \hat{g}_w &= p_w \exp(d_w(\mathbf{p})) \\ \hat{g}_h &= p_h \exp(d_h(\mathbf{p}))\end{aligned}$$

Một lợi ích rõ ràng của việc áp dụng chuyển đổi như vậy là tất cả các hàm hiệu chỉnh hộp giới hạn, $d_i(p)$ với $i \in \{x, y, w, h\}$, có thể lấy một vài giá trị giữa $[-\infty, \infty]$. Mục



Hình 2.16: Mô tả công thức chuyển đổi của bounding box

tiêu là học:

$$\begin{aligned} t_x &= (g_x - p_x)/p_w \\ t_y &= (g_y - p_y)/p_h \\ t_w &= \log(g_w/p_w) \\ t_h &= \log(g_h/p_h) \end{aligned}$$

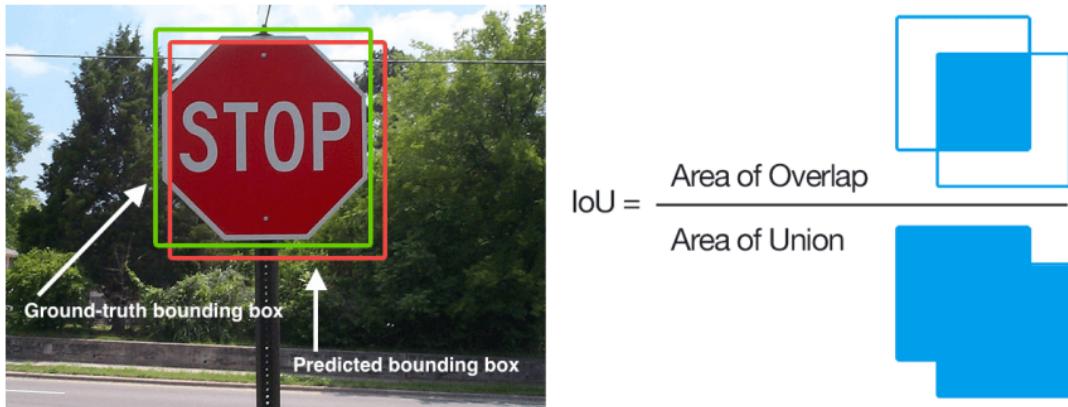
Mục tiêu của chúng ta là cần cực tiểu hoá hàm mất mát MSE với regularization:

$$\mathcal{L}_{\text{reg}} = \sum_{i \in \{x, y, w, h\}} (t_i - d_i(\mathbf{p}))^2 + \lambda \|\mathbf{w}\|^2$$

Một điều đáng chú ý là không phải tất cả các box đề xuất đều có các ground-box tương ứng. Ở đây, chỉ có các box đề xuất mà có *Intersection over Union* (IoU) với box thật ít nhất 0,6 trở lên mới được giữ lại để đào tạo mô hình.

2.3.3 Intersection over Union

Intersection over Union (IoU) được sử dụng trong bài toán object detection, để đánh giá xem bounding box dự đoán đối tượng khớp với ground truth thật của đối tượng.



Hình 2.17: Intersection over Union

Ta sẽ đánh giá model bằng tỉ lệ area overlap với area union giữa thực tế và predict. Hình 2.17 cho thấy các tính IoU.

Dễ thấy rằng, chỉ số IoU sẽ nằm trong khoảng [0, 1] và IoU càng gần 1 thì bounding box dự đoán càng gần với ground-truth box.



Hình 2.18: Dánh giá độ tốt của box trên IoU

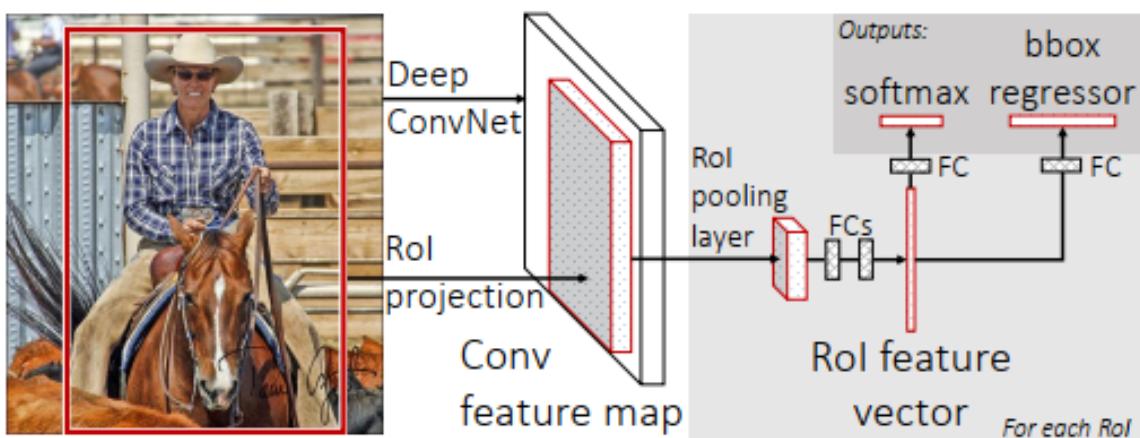
2.4 Fast R-CNN

Fast R-CNN (Fast Region-based Convolution Network) là bản cải tiến của R-CNN của cùng tác giả. So với R-CNN thì Fast R-CNN có một số cải tiến để training hiệu quả và testing tốc độ nhanh hơn đồng thời cũng tăng độ chính xác.

R-CNN xuất sắc trong bài toán phát hiện đối tượng với độ chính xác cao sử dụng deep ConvNet để phân lớp các object proposal. Tuy nhiên, nó có một số nhược điểm đáng chú ý sau:

- 1 **Training qua nhiều giai đoạn:** Đầu tiên, R-CNN sử dụng transfer learning cho một mạng CNN trên object proposal. Rồi fit SVM từ các đặc trưng của mạng CNN đó. Các SVM như các trình phát hiện đối tượng, thay thế cho các class softmax đã học ở transfer learning. Trong quá trình thứ ba, bounding box regression được học.
- 2 **Training tốn kém không gian và thời gian:** Với quá trình train SVM và bounding box regression, các đặc trưng được trích xuất từ mỗi object proposal trong mỗi hình ảnh được lưu vào ổ đĩa. Các đặc trưng này khi huấn luyện các mạng dữ liệu lớn có thể đòi hỏi hàng trăm GB lưu trữ.
- 3 **Phát hiện đối tượng chậm:** Quá trình dự đoán đối tượng trong ảnh của R-CNN chậm. Các bước chưa chia sẻ tính toán với nhau.

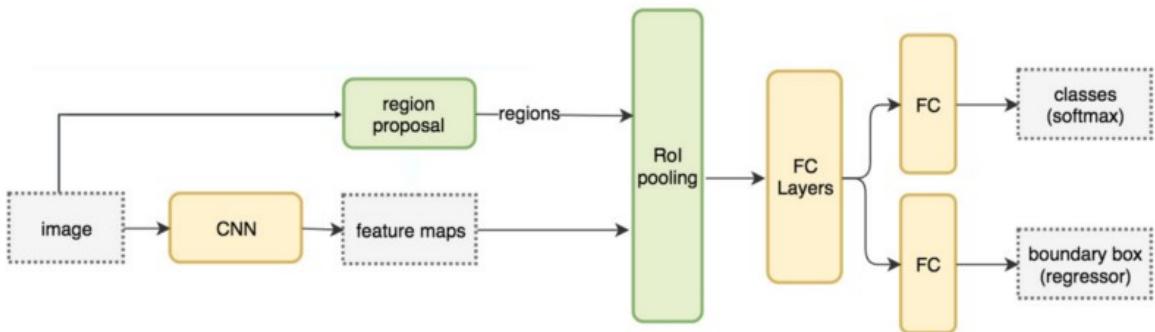
Vì những bất cập này, Grishick (2015) đã cải tiến quy trình đào tạo. Thay vì phải trích rút đặc trưng của mỗi region proposal, chúng ta có thể dùng CNN trích rút đặc trưng của toàn bộ bức ảnh trước (gọi là feature map), đồng thời trích rút các region proposal, sau đó lấy các region proposal tương ứng trên feature map, rescale và cuối cùng là phân lớp và tìm vị trí của object. Với việc không lặp lại 2000 lần trích rút đặc trưng cho mỗi ảnh, Fast R-CNN giảm thời gian xử lý một cách đáng kể.



Hình 2.19: Mô hình Fast R-CNN

Luồng xử lý của Fast R-CNN có thể tóm tắt như sau:

- 1 Pre-train cho một mạng CNN cho bài toán phân loại hình ảnh.
- 2 Đưa ra các vùng (region) bằng thuật toán selective search (khoảng 2000 vùng trên mỗi ảnh).
- 3 Sử dụng lớp ROI Pooling ở cuối mạng CNN để đầu ra là các vector đặc trưng có độ dài cố định của region proposal.
- 4 Một lớp fully connected được sử dụng để flatten các đặc trưng.
- 5 Cuối cùng, mô hình nhánh thành 2 lớp đầu ra: Một lớp softmax của K+1 lớp (giống như trong R-CNN, và thêm 1 lớp background), cho ra phân phối xác suất cho mỗi ROI (Region of Interest) ở các lớp. Một lớp đầu ra thứ 2 là dự đoán bounding box.



Hình 2.20: Luồng hoạt động của mô hình Fast R-CNN

Một khác biệt lớn nhất của Fast R-CNN là toàn bộ mạng (feature extractor, classifier, bounding box regression) có thể huấn luyện end-to-end (từ đầu đến cuối) với 2 hàm mất mát (loss function) khác nhau cùng một khác (classification loss và localization loss). Điều này làm tăng tốc độ chính xác của mô hình.

2.4.1 ROI pooling

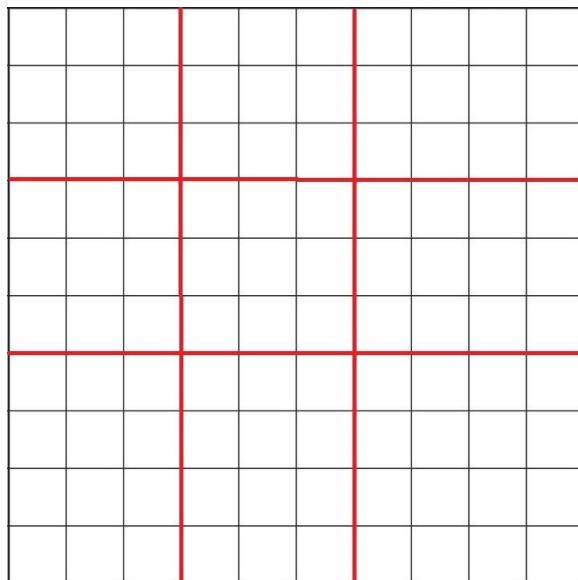
ROI pooling (Region of Interest pooling) là một dạng của **pooling layer**. Điểm khác so với max pooling hay average pooling là bất kể kích thước của đầu vào, ROI pooling luôn cho ra đầu ra kích thước cố định được định nghĩa trước.

Ta ký hiệu a/b là phần nguyên của a khi chia cho b và $a\%b$ là phần dư của a khi chia cho b . Ví dụ: $10/3=3$ và $10\%3=1$.

Gọi đầu vào của ROI pooling là ma trận kích thước $m \times n$ và đầu ra có kích thước $h \times k$:

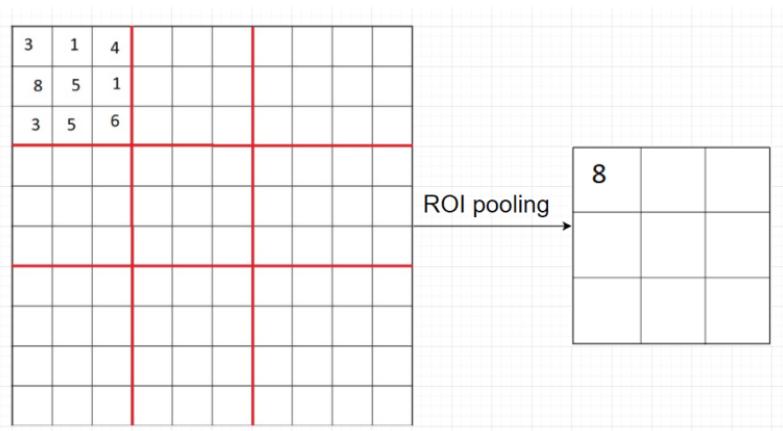
- Ta chia chiều rộng thành h phần, $h - 1$ phần có kích thước m/h , phần cuối có kích thước $m/h + m\%h$.
- Chia chiều dài thành k phần, $k - 1$ phần có kích thước n/k , phần cuối có kích thước $n/k + n\%k$.

Ví dụ $m = n = 10$, $h = k = 3$, do $m/h = 3$ và $m\%h = 1$, nên ta sẽ chia chiều rộng thành 3 phần, 2 phần có kích thước 3, và 1 phần có kích thước 4 như hình 2.21.



Hình 2.21: Ví dụ về cách chia miền của ROI pooling

Sau đó với mỗi khối được tạo ra bằng các đường đỏ và cạnh, ta thực hiện max pooling lấy ra 1 giá trị như hình 2.22.



Hình 2.22: Hoạt động của Roi Pooling

2.4.2 Hàm mất mát của Fast R-CNN

Mô hình sử dụng một hàm mất mát cho cả hai tác vụ phân loại và xác định vị trí vật thể:

- u : True class label, $u \in 0, 1, \dots, K$; theo quy ước, lớp background $u = 0$
- p : Phân phối xác suất rời rạc (per ROI) qua $K+1$ lớp: $p = (p_0, \dots, p_K)$, tính toán một softmax qua $K+1$ đầu ra của Fully connected layer
- v : True bounding box $v = (v_x, v_y, v_w, v_h)$
- t^u : Predicted bounding box correction, $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$

Hàm mất mát bằng tổng của chi phí phân lớp và bounding box regression:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}}$$

Với ROI background, \mathcal{L}_{box} bỏ qua với hàm $[u \geq 1]$ được định nghĩa như sau:

$$[u \geq 1] = \begin{cases} 1 & \text{if } u \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

Vậy hàm mất mát cuối cùng có công thức như sau:

$$\begin{aligned} \mathcal{L}(p, u, t^u, v) &= \mathcal{L}_{\text{cls}}(p, u) + [u \geq 1]\mathcal{L}_{\text{box}}(t^u, v) \\ \mathcal{L}_{\text{cls}}(p, u) &= -\log p_u \\ \mathcal{L}_{\text{box}}(t^u, v) &= \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L_1}(t_i^u - v_i) \end{aligned}$$

trong đó smooth_{L_1} được cho thấy là ít nhạy cảm hơn với các ngoại lệ và có công thức như sau:

$$\text{smooth}_{L_1} = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

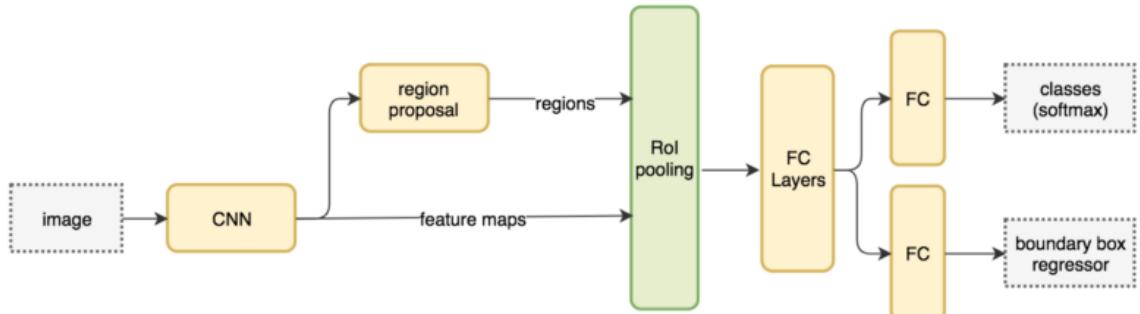
2.5 Faster R-CNN

Có thể thấy, mô hình Fast R-CNN đã cải thiện hiệu suất và tốc độ đáng kể so với R-CNN. Tuy nhiên, một nhược điểm dễ nhận thấy đó là Fast R-CNN vẫn sử dụng Selective Search để trích xuất ra các region proposal cho ảnh và đây cũng chính là nút thắt cổ chai của Fast R-CNN. Từ đó, năm 2016, Shaoqing Ren và các cộng sự tại Microsoft Research đã nghiên cứu và công bố thuật toán Faster R-CNN, kiến trúc này mang lại độ chính xác cao nhất đạt được trên cả hai nhiệm vụ phát hiện và nhận dạng đối tượng tại các cuộc thi ILSVRC-2015 và MS COCO-2015.

Kiến trúc được thiết kế để đề xuất và tinh chỉnh các region proposals như là một phần của quá trình huấn luyện, được gọi là mạng đề xuất khu vực (Region Proposal Network), hoặc RPN. Các vùng này sau đó được sử dụng cùng với mô hình Fast R-CNN trong một thiết kế mô hình duy nhất. Những cải tiến này vừa làm giảm số lượng region proposal vừa tăng tốc hoạt động trong thời gian thử nghiệm mô hình lên gần thời gian thực với hiệu suất tốt nhất.

Mặc dù là một mô hình đơn lẻ duy nhất, kiến trúc này là kết hợp của hai modules:

- Mạng đề xuất khu vực (Region Proposal Network, viết tắt là RPN). Mạng CNN để đề xuất các vùng và loại đối tượng cần xem xét trong vùng.
- Fast R-CNN: Mạng CNN để trích xuất các features từ các region proposal và trả ra các bounding box và nhãn.



Hình 2.23: Mô hình Faster R-CNN

Luồng xử lý của Faster R-CNN có thể tóm tắt như sau:

- 1 Đầu vào là một ảnh được đưa qua mạng CNN để thu được feature map.
- 2 Mạng RPN (gồm các lớp convolution) dùng để trích xuất ra các Region of Interest (RoI) - các vùng có khả năng chứa đối tượng của ảnh. Đầu vào của mạng RPN là feature map, đầu ra gồm 2 thành phần là binary object classification (phân biệt RoI là background - không chứa đối tượng hay foreground - chứa đối tượng), và bounding box regression để dự đoán các giá trị offset cho bounding box.
- 3 Sau khi đi qua RPN, sẽ thu được các RoI có kích thước khác nhau, mô hình sử dụng Roi pooling để thu về cùng một kích thước cố định.
- 4 Một lớp fully connected được sử dụng để flatten các đặc trưng.
- 5 Cuối cùng, mô hình nhánh thành hai nhánh: classification cho phân loại đối tượng và bounding box regression.

2.5.1 Region Proposal Network

Mạng Faster R-CNN sử dụng **Region Proposal Network** (RPN) để trích rút các vùng có khả năng chứa đối tượng của ảnh. Nhờ vào cải tiến này mà Faster R-CNN như một mạng end-to-end giúp cải thiện tốc độ.

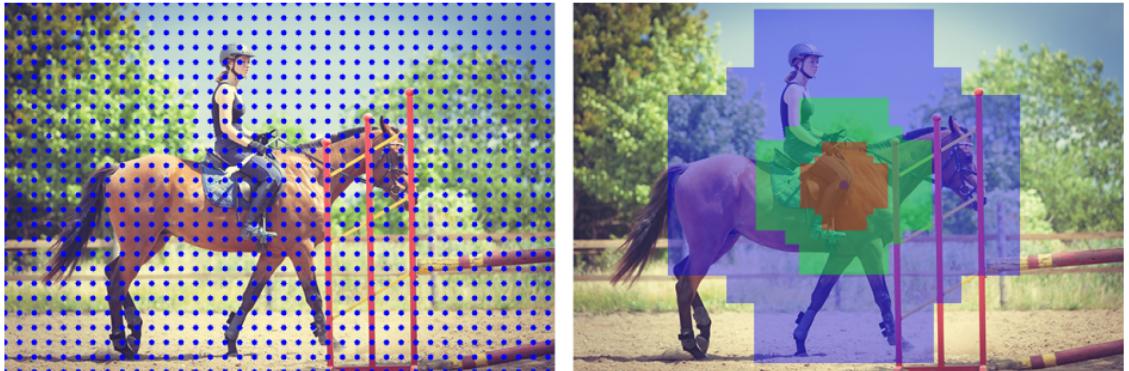
RPN sử dụng 1 convolution layer với 512 channels, kernel size là (3,3) lên feature map. Sau đó được chia làm hai nhánh, một cho binary object classification, một cho bounding box regression. Cả hai sử dụng một convolution layer với kernel size là (1,1) nhưng với số channel đầu ra khác nhau. Với binary object classification thì có $2k$ channel đầu ra, với k là tổng số lượng **anchors** nhằm xác định xem một anchor là background (không chứa đối tượng) hoặc foreground (chứa đối tượng). Thực chất, ở đây số channel có thể chuyển về chỉ còn k channel với việc sử dụng sigmoid activation. Với bounding box regression thì có $4k$ channel output, với 4 là biểu trưng cho 4 tọa độ offset (x, y, w, h).

Anchors

Với một bounding box được xác định bằng 2 điểm ở góc, ví dụ $A(x_{min}, y_{min})$ và $B(x_{max}, y_{max})$. Tuy nhiên, có một số vấn đề phát sinh như:

- Khi RPN dự đoán ra phải có ràng buộc $x_{min} < x_{max}$ và $y_{min} < y_{max}$.
- Các giá trị x, y khi dự đoán có thể ra ngoài khôi bức ảnh.

Từ đó, **Anchor** ra đời như một kỹ thuật mới để biểu diễn region proposal. Chúng ta sẽ dự đoán điểm của trung tâm box (x_{center}, y_{center}) và chiều rộng (width), chiều cao (height) của box. Mỗi anchor sẽ được xác định bởi 4 tham số ($x_{center}, y_{center}, width, height$).



Hình 2.24: Anchor

Hình 2.24 bên trái kích thước $400 * 600$ pixel, các tâm của anchor box màu xanh, cách nhau 16 pixel \Rightarrow có khoảng $(400*600)/(16*16) = 938$ tâm. Do các object trong ảnh có thể có kích thước và tỉ lệ khác nhau nên với mỗi tâm ta định nghĩa 9 anchors với kích thước $64 \times 64, 128 \times 128, 256 \times 256$, mỗi kích thước có 3 tỉ lệ tương ứng: 1 : 1, 1 : 2 và 2 : 1.

Các anchor được gán mác là positive/negative (foreground/background) dựa vào IoU với ground-truth bounding box theo quy tắc sau:

- Các anchor có tỉ lệ IoU lớn nhất với ground-truth box sẽ được gọi là positive.
- Các anchor có tỉ lệ IoU ≥ 0.7 sẽ được coi là positive.
- Các anchor có tỉ lệ IoU < 0.3 sẽ được coi là negative.
- Các anchor có IoU nằm trong khoảng $0.3 \leq x < 0.7$ sẽ được coi là trung tính (neutral) và sẽ không được sử dụng trong quá trình huấn luyện mô hình.

Hàm mất mát

Với mỗi anchor, thực hiện việc phân loại có hay không có đối tượng và hồi quy giá trị độ bù trừ (offsets) giữa ground-truth và anchor bằng cách xây dựng hàm mất mát:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Với i là chỉ số của anchor trong mini-batch và p_i là xác suất dự đoán của anchor i là một đối tượng. Giá trị nhãn ground-truth p_i^* bằng 1 nếu anchor là positive, bằng 0 nếu anchor là negative.

- t_i là một vector 4 chiều biểu diễn giá trị toạ độ của bounding box dự đoán.

- t_i^* là vector 4 chiều biểu diễn giá trị tọa độ của ground-truth box tương ứng với positive anchor.
- L_{cls} là log loss của 2 class (background và foreground).
- L_{reg} là hàm loss tính cho bounding box regression. Trong bài báo, các tác giả đề cập sử dụng hàm Smooth L1.

Về cơ bản, hàm mất mát của RPN có tư tưởng tương tự như hàm mất mát mà Fast R-CNN đã sử dụng.

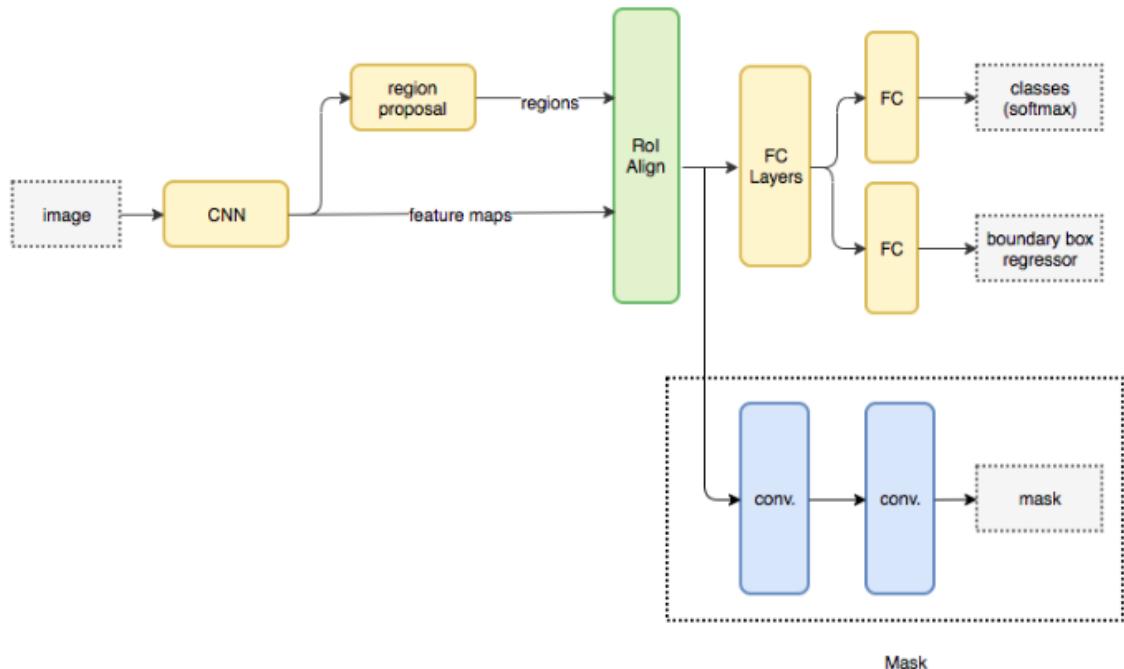
2.5.2 Non-maxima suppression

Các region proposal thu được sau RPN sẽ bị chồng chéo lên nhau rất nhiều nên cần một cơ chế để lọc các vùng đó, và đồng thời cũng tăng tốc độ xử lý của mô hình. Một phương pháp được đề xuất đó là Non-maxima suppression. Các bước thực hiện như sau:

- **Bước 1:** Chọn ra anchor box (A) có xác suất là foreground lớn nhất trong tập Input.
- **Bước 2:** Thêm A vào tập Output.
- **Bước 3:** Loại bỏ A và các anchor box trong tập Input mà có hệ số IoU với A lớn hơn 0.5 ra khỏi tập Input.
- **Bước 4:** Kiểm tra nếu tập Input rỗng hoặc tập Output đủ anchor mong muốn thì dừng lại, nếu không quay lại bước 1.

2.6 Mask R-CNN

Mask R-CNN là mô hình cho phân đoạn đối tượng. Kế thừa từ kiến trúc Faster R-CNN và thêm nhánh làm nhiệm vụ phân đoạn đối tượng trên mỗi RoI song song với nhánh dự đoán lớp đối tượng và hộp bao (Hình 2.25).

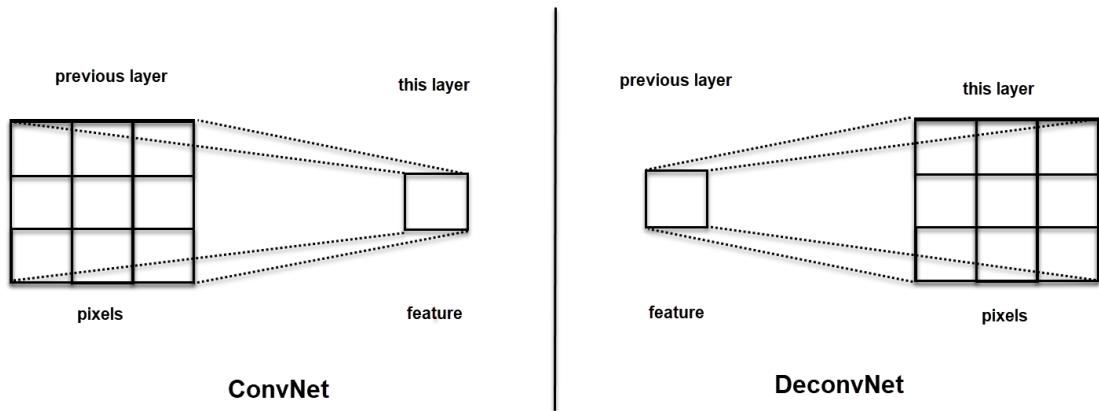


Hình 2.25: Mô phỏng mô hình Mask R-CNN

Mask R-CNN thực hiện cả 2 nhiệm vụ là phát hiện đối tượng và phân đoạn đối tượng. Về cơ bản, luồng xử lý của Mask R-CNN vẫn như Faster R-CNN, chỉ bổ sung thêm một nhánh để dự đoán mặt nạ (mask) cho đối tượng. Do đó, hàm mất mát tương ứng với nhiều nhiệm vụ sẽ là $\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{mask}$. Phần phân đoạn đối tượng **deconvolution** và **unpooling** để thu được mặt nạ trên ảnh gốc. Cụ thể cách thức hoạt động hai bước này như sau:

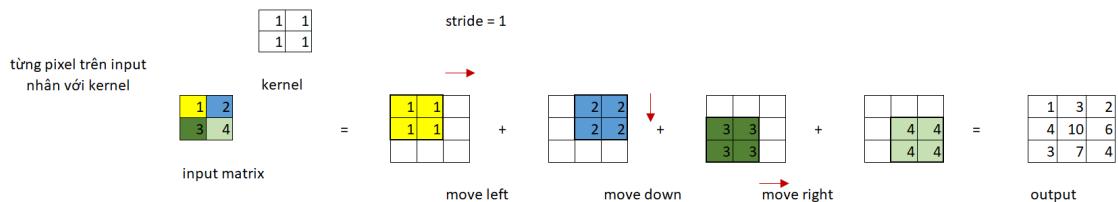
Deconvolution

Decovolution (Mạng giải chập) là một mạng để giải mã ngược trở lại từ đặc trưng sang ảnh, giúp gia tăng kích thước của mạng CNN thông qua tích chập chuyển vị (Transposed Convolution). Vai trò của tích chập chuyển vị xuất phát từ nhu cầu biến đổi theo quá trình ngược lại của mạng tích chập thông thường hay còn gọi là giải chập (Deconvolutional Neural Network). Trên thực tế thì có thể coi tích chập chuyển vị là một quá trình ngược của tích chập thông thường khi mỗi một đặc trưng (feature) được mapping sang các pixels ảnh thay vì ngược lại từ các pixels sang đặc trưng (feature). Ta di chuyển các pixel của ma trận đầu vào từ trái qua phải và từ trên xuống dưới. Sau đó lấy giá trị của pixel nhân với ma trận bộ lọc sẽ thu được ma trận output có kích



Hình 2.26: ConvNet và DeconvNet

thước tương đương. Tùy vào stride qui định là bao nhiêu mà ta sẽ di chuyển kết quả của mỗi lần nhân pixel với bộ lọc sang bấy nhiêu đơn vị. Sau cùng ta tính tổng các vị trí tương ứng của các ma trận kết quả để thu được ma trận chuyển vị. Trong trường hợp stride không bằng kích thước kernel thì ma trận kết quả tích chập sẽ overlapping lên nhau. Khi đó ta sẽ cộng dồn chúng. Hình 2.27 cho ta một ví dụ về cách tính tích chập chuyển vị.

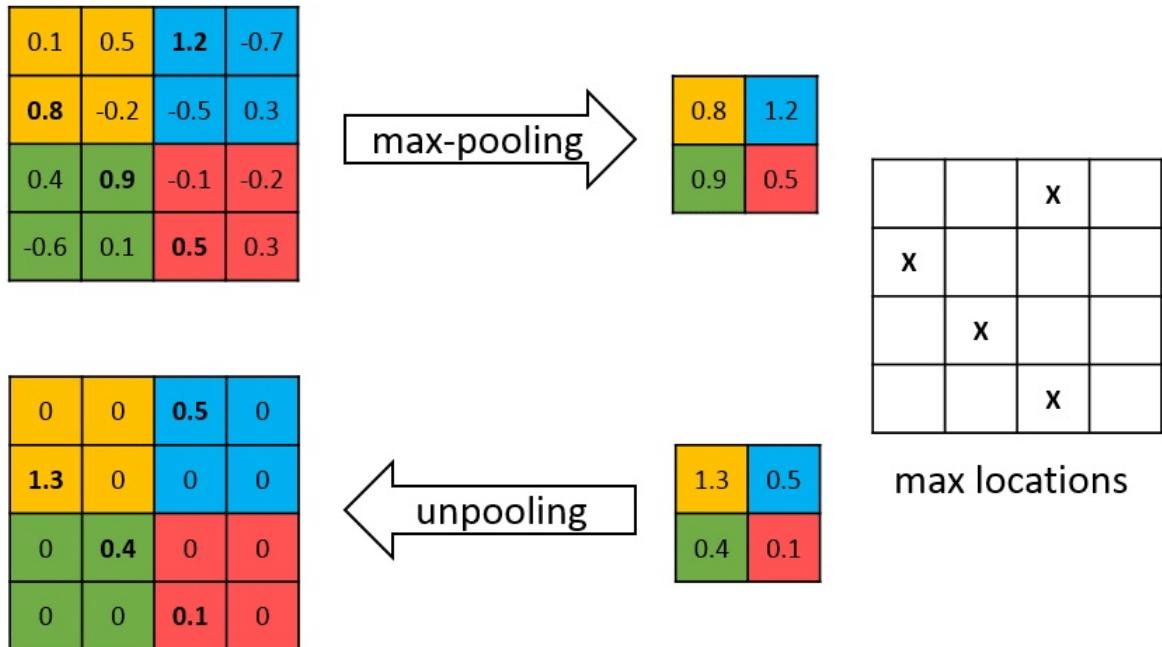


Hình 2.27: Ví dụ về cách tính tích chập chuyển vị

Ngoài việc sử dụng tích chập chuyển vị, ta có thể thay thế bằng các phương pháp như UnSampling hoặc tích chập giãn nở (Dilation Convolution).

Unpooling

Với max pooling, ta lấy giá trị max của các khối, vì vậy thông tin sẽ bị mất mát đi qua quá trình pooling. Unpooling là quá trình ta xây dựng lại ma trận bằng cách nhớ toạ độ của điểm max và điền chính xác điểm max, còn lại các điểm khác trong khối sẽ được xấp xỉ từ điểm đã có giá trị. Hình 2.28 mô phỏng ví dụ về cách hoạt động của unpooling.



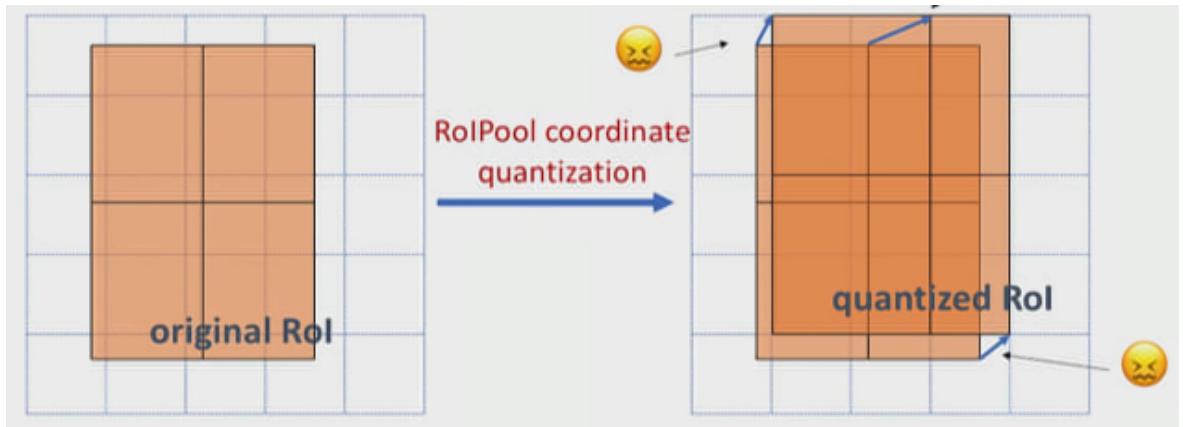
Hình 2.28: Ví dụ về hoạt động của Unpooling

Qua việc deconvolution và unpooling, chúng ta có thể xây dựng một phân vùng dự đoán trên ảnh gốc cho tất cả các lớp đối tượng. Đây cũng chính là đầu ra cho khối phân vùng đối tượng.

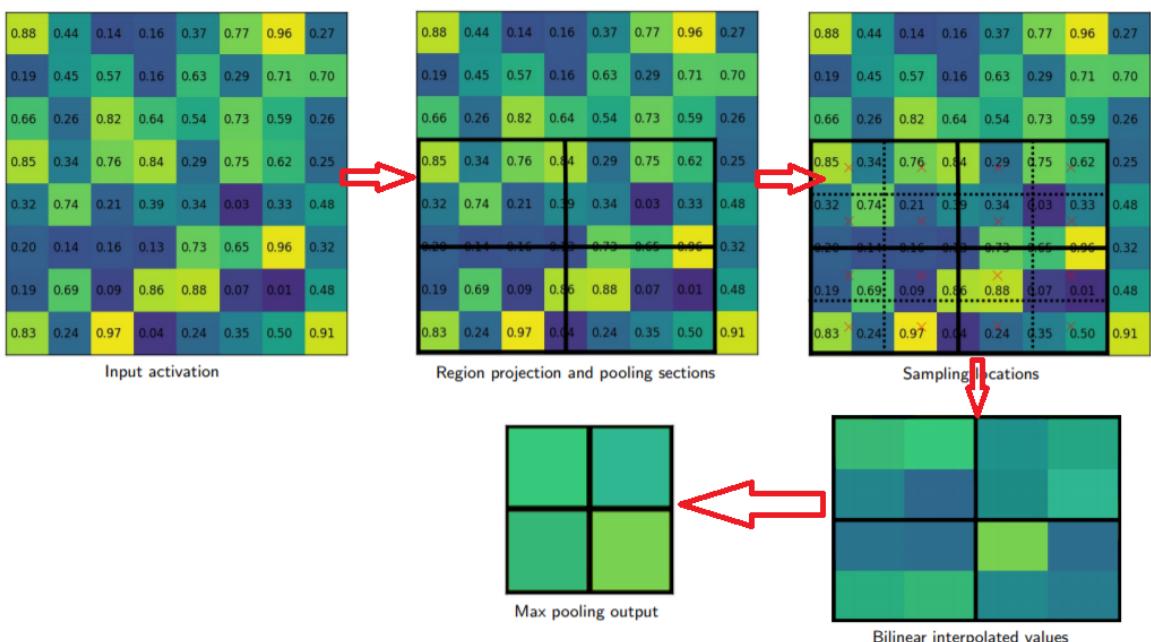
2.6.1 RoiAlign

Mask R-CNN cải tiến Faster R-CNN bằng cách thay thế việc sử dụng Roi Pooling bằng RoiAlign. Điều này giúp cải thiện độ chính xác cho mô hình. Lớp RoiPool trích xuất feature map từ mỗi ROI. Vấn đề với RoiPooling là lượng tử hoá (quantization). Nếu ROI không hoàn toàn phù hợp với lưới trong feature map như hình 2.29, lượng tử hoá sẽ phá vỡ căn chỉnh pixel-to-pixel. Vấn đề này không quan trọng trong phát hiện đối tượng, nhưng với bài toán phân đoạn đối tượng thì nó đòi hỏi chính xác cao và quan trọng hơn.

RoiAlign thay vì việc tính mỗi điểm trên đặc trưng thông qua việc lượng tử hóa từng khối nhỏ trên ma trận để thu được kích cỡ cố định thì RoiAlign thực hiện một phép nội suy song tuyến để tính ra từ đặc trưng của mỗi vùng như hình 2.30.



Hình 2.29: Vấn đề lượng tử hóa của RoiPooling



Hình 2.30: Mô tả phương pháp RoiAlign

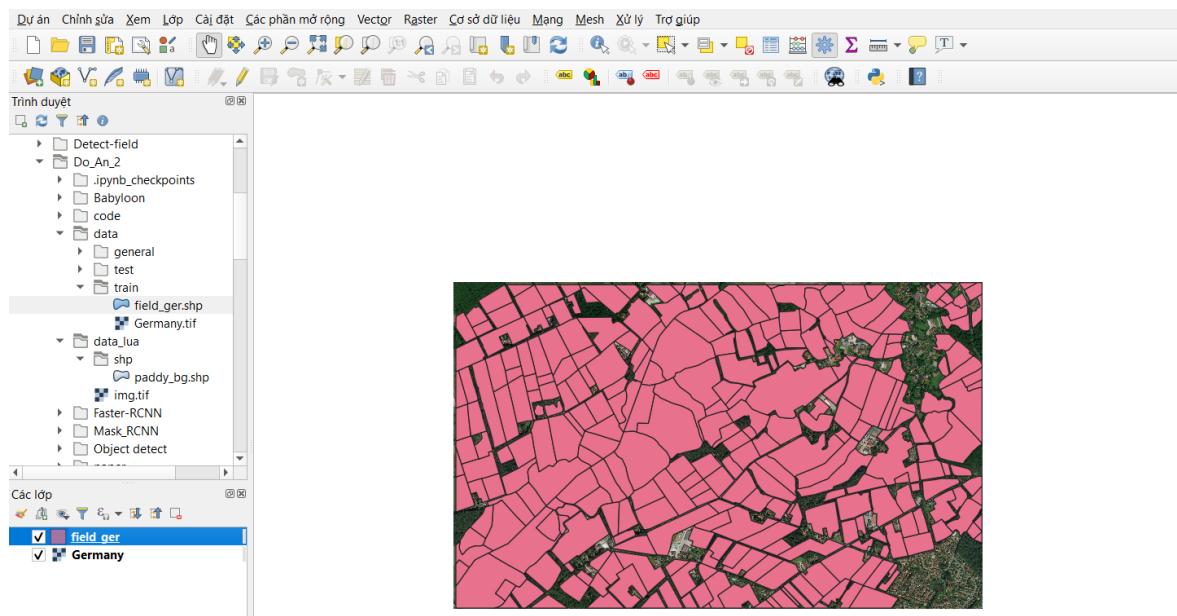
Chương 3

Xây dựng mô hình và kết quả thực nghiệm

Toàn bộ mã nguồn cho báo cáo được lưu trữ tại địa chỉ: <https://github.com/nducthang/Detect-field>

3.1 Xử lý dữ liệu

Dữ liệu đầu vào ảnh ở dạng *.tif*, các tệp định dạng nhãn cho ảnh được lưu dưới dạng file *.shp* với thông tin của 300 polygon. Để có thể mở và nhìn trực quan các file này, em sử dụng phần mềm **QGIS**.



Hình 3.1: Sử dụng phần mềm QGIS để mở file

File ảnh dữ liệu này có kích thước 20833×12833 pixel. Chúng ta cần phải thực hiện việc cắt file ảnh lớn này ra thành các ảnh nhỏ, đồng thời giữ được các nhãn ban đầu

phục vụ cho bước huấn luyện mô hình sau này. Ngoài ra, chúng ta cần chuyển định dạng *.tif* về các ảnh định dạng thông thường để dễ xử lý trên các pixel ảnh thay vì xử lý trên toạ độ địa lý.

Để có thể đọc được thông tin của ảnh *.tif* và *.shp* trên Python, em sử dụng thư viện **osgeo** của Python để đọc dữ liệu đầu vào, sau đó thực hiện việc chuyển ảnh *.tif* sang định dạng *.jpg* bình thường, đồng thời chuyển các thông tin về toạ độ địa lý ở trong file *.shp* tương ứng về toạ độ pixel.

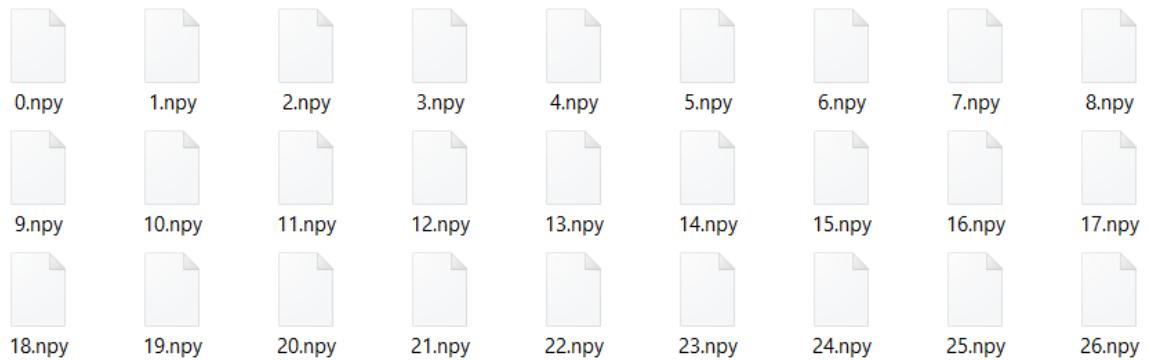
Tiếp theo, em tiến hành việc cắt ảnh và tạo dữ liệu phục vụ cho bước huấn luyện mô hình. Việc cắt dữ liệu phải đảm bảo cắt được cả các nhãn polygon ở ảnh to ban đầu sao cho không mất mát các thông tin về phân đoạn đối tượng các cánh đồng trong ảnh.



Hình 3.2: Ảnh được cắt sử dụng để huấn luyện

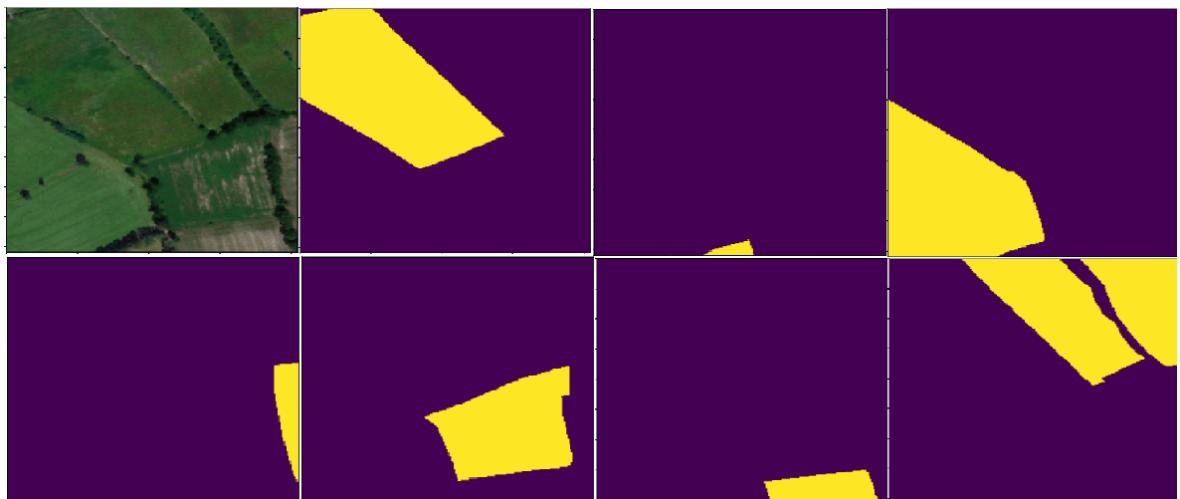
Em tiến hành việc cắt ảnh lớn thành các ảnh nhỏ có kích thước 1024×1024 . Sở dĩ chọn kích thước này vì nó phù hợp với bài toán chúng ta đang làm, đối tượng đang xét có kích thước khá lớn và là bài toán phân đoạn nên ta cần để kích thước ảnh to đồng thời đảm bảo việc load dữ liệu vào mô hình ổn định, không gây tràn RAM. Khi cắt ảnh, việc xác định được các toạ độ đỉnh mới của polygon sau khi cắt là *khó*, một polygon ban đầu có thể bị cắt thành nhiều phần, khó để xác định toạ độ pixel các đỉnh của từng phần con này. Vì thế, thay vì lưu thông tin nhãn dưới dạng các toạ độ polygon, em tiến hành lưu thông tin mặt nạ của ảnh dưới dạng *array* của *numpy*. Giả

sử một ảnh có 3 đối tượng là cánh đồng, thì sẽ có tương ứng có 3 ma trận cùng kích thước với ảnh đó, trong đó, mỗi ma trận sẽ chứa thông tin về 1 đối tượng trong ảnh, các pixel chứa đối tượng được đánh số 1, các pixel không chứa đối tượng được đánh số 0. Các thông tin ma trận này có thể được lưu lại trong ổ cứng dưới dạng file *.npy*. Khi lưu dữ liệu dưới dạng này, các thông tin về bounding box của đối tượng hoàn toàn có thể được lấy ra từ thông tin các pixel mặt nạ. Hình 3.2 cho thấy kết quả các ảnh con



Hình 3.3: Mặt nạ các ảnh được lưu tương ứng dưới dạng file *.npy*

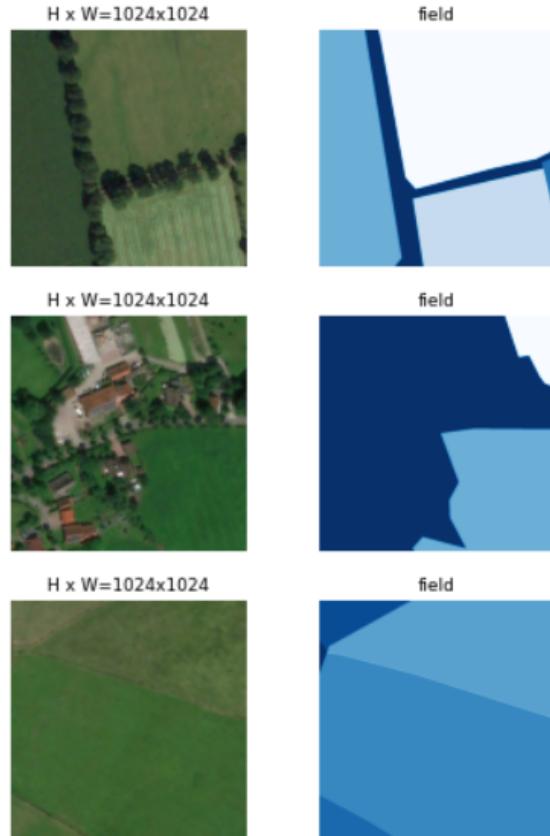
thu được sau khi tiến hành cắt ảnh to, hình 3.3 cho thấy các file mặt nạ tương ứng với các ảnh con này, các file này là các *array* của *numpy* chứa thông tin phân đoạn của các đối tượng trong ảnh, được lưu dưới định dạng *.npy*. Các ảnh sau khi được cắt và các mặt nạ này được sử dụng cho bước huấn luyện mô hình sau này. Hình 3.4 cho thấy biểu diễn của một ảnh bất kỳ trong tập dữ liệu huấn luyện cùng với các mặt nạ của nó.



Hình 3.4: Mô phỏng một ảnh bất kỳ trong dữ liệu huấn luyện

3.2 Huấn luyện mô hình

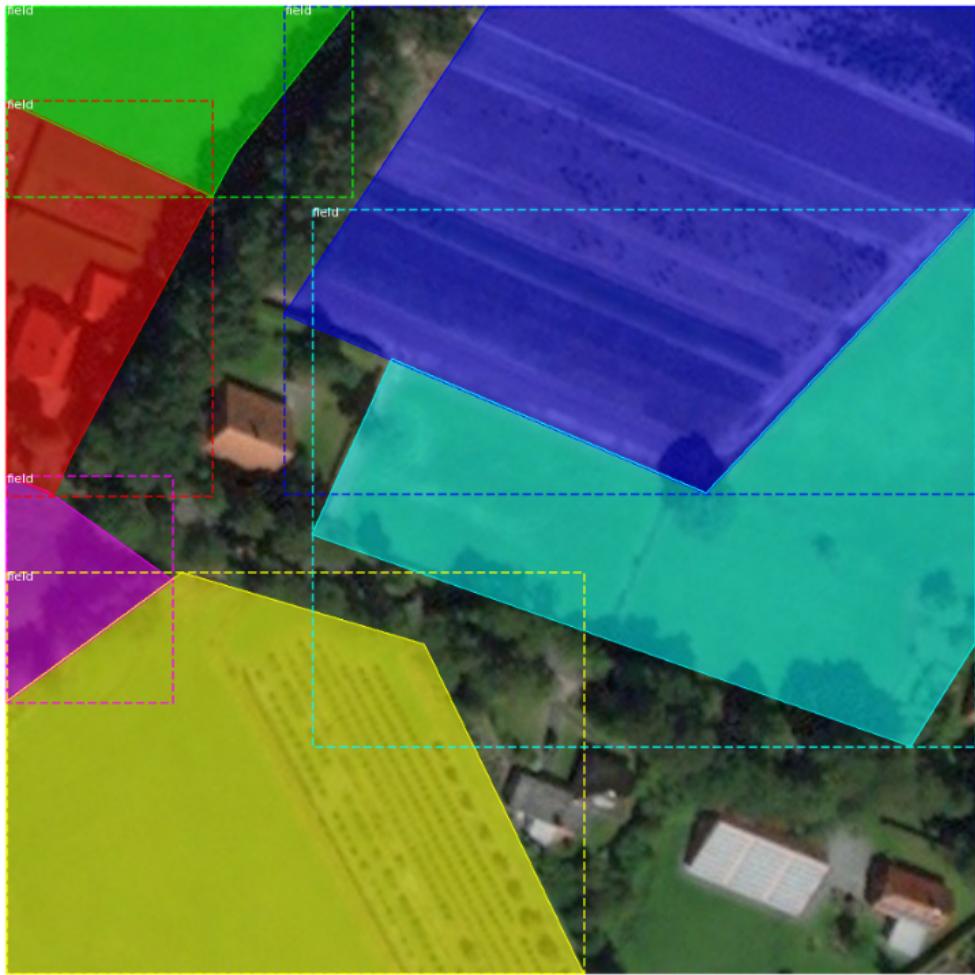
Mô hình được huấn luyện trên bộ dữ liệu gồm 936 ảnh. Hình 3.5 cho ví dụ về mô phỏng một vài dữ liệu của tập huấn luyện.



Hình 3.5: Mô phỏng dữ liệu

Các box của đối tượng sẽ được trích xuất ra từ các mặt nạ của dữ liệu để phục vụ cho quá trình huấn luyện như hình 3.6. Em sử dụng pre-train của COCO để khởi tạo các tham số cho mô hình mạng. Theo thực nghiệm của em thì pre-train của COCO hội tụ nhanh và tốt hơn so với pre-train ImageNet trong bài toán này. Các bước huấn luyện như sau:

- 1 Chuẩn bị dữ liệu ảnh và mặt nạ của ảnh đó.
- 2 Tải bộ trọng số pre-train COCO và cấu hình cho mô hình mạng.
- 3 Tiến hành huấn luyện.
- 4 Lưu lại checkpoint để có thể thực hiện tiếp tục huấn luyện hoặc tải trọng số mô hình cho việc dự đoán nếu xảy ra lỗi.



Hình 3.6: Trích xuất box đối tượng từ mặt nã

Trọng số pre-train COCO để khởi tạo mô hình được tải tại: https://github.com/matterport/Mask_RCNN/releases

Các tham số cấu hình cho mạng được mô tả ở hình 3.7. Sau đó thực hiện việc huấn luyện mô hình trên TPU của *Google Colab* với 200 epochs, thời gian huấn luyện khoảng 15 tiếng. Đồng thời, lưu lại checkpoint cho mỗi epochs để có thể tải lại trọng số của mô hình để huấn luyện tiếp nếu lỗi xảy ra.

```

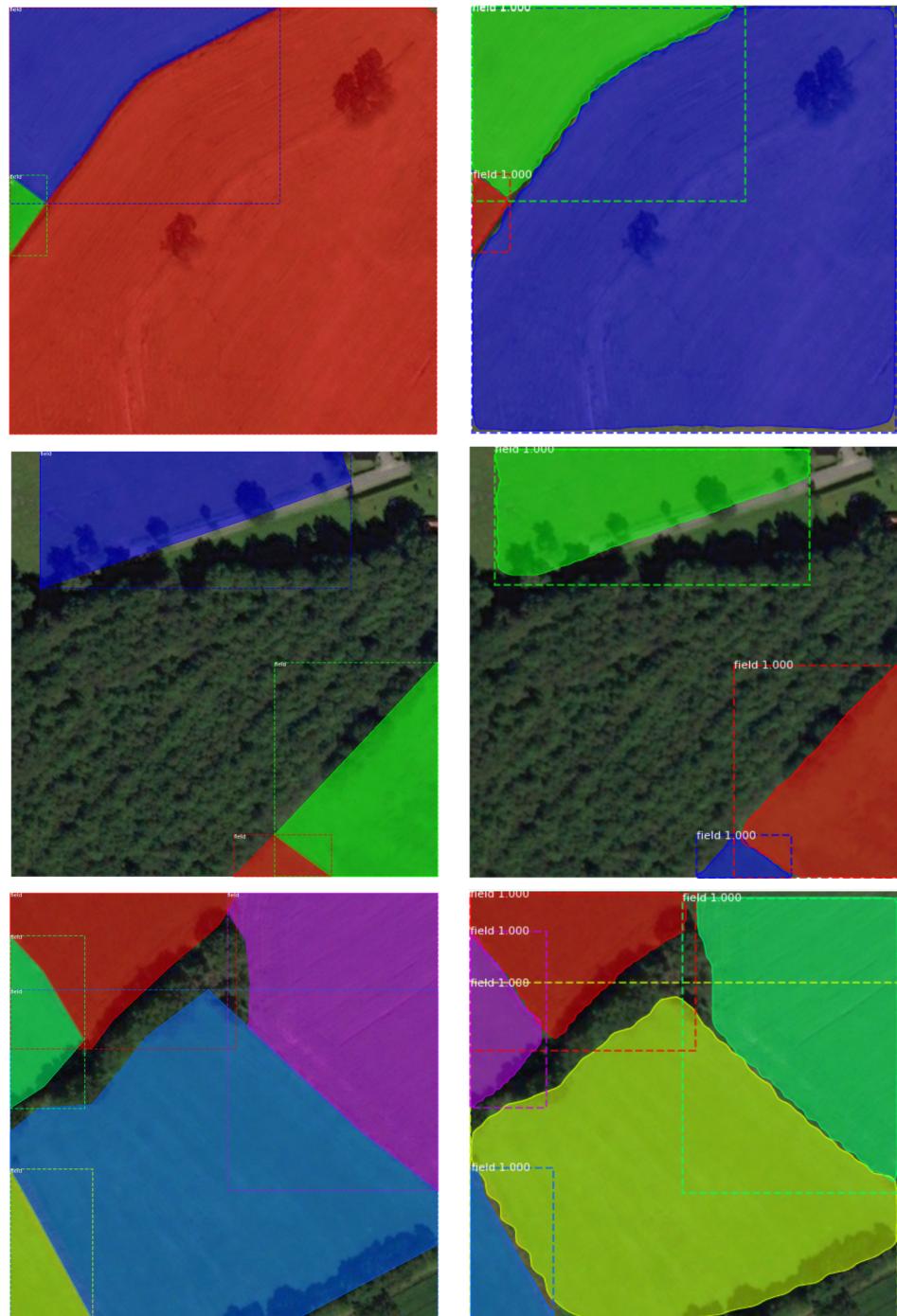
BACKBONE           resnet101
BACKBONE_STRIDES [4, 8, 16, 32, 64]
BATCH_SIZE         1
BBOX_STD_DEV      [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE None
DETECTION_MAX_INSTANCES 100
DETECTION_MIN_CONFIDENCE 0.6
DETECTION_NMS_THRESHOLD 0.6
FPN_CLASSIF_FC_LAYERS_SIZE 1024
GPU_COUNT          1
GRADIENT_CLIP_NORM 5.0
IMAGES_PER_GPU     1
IMAGE_CHANNEL_COUNT 3
IMAGE_MAX_DIM      1024
IMAGE_META_SIZE    14
IMAGE_MIN_DIM      800
IMAGE_MIN_SCALE    0
IMAGE_RESIZE_MODE square
IMAGE_SHAPE        [1024 1024 3]
LEARNING_MOMENTUM 0.9
LEARNING_RATE      0.001
LOSS_WEIGHTS       {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_SHAPE          [28, 28]
MAX_GT_INSTANCES   100
MEAN_PIXEL          [123.7 116.8 103.9]
MINI_MASK_SHAPE    (56, 56)
NAME                field
NUM_CLASSES         2
POOL_SIZE           7
POST_NMS_ROIS_INFERENCE 1000
POST_NMS_ROIS_TRAINING 2000
PRE_NMS_LIMIT       6000
ROI_POSITIVE_RATIO 0.33
RPN_ANCHOR RATIOS [0.5, 1, 2]
RPN_ANCHOR_SCALES (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE 1
RPN_BBOX_STD_DEV   [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD  0.7
RPN_TRAIN_ANCHORS_PER_IMAGE 256
STEPS_PER_EPOCH     936
TOP_DOWN_PYRAMID_SIZE 256
TRAIN_BN            False
TRAIN_ROIS_PER_IMAGE 200
USE_MINI_MASK       False
USE_RPN_ROIS        True
VALIDATION_STEPS    50
WEIGHT_DECAY        0.0001

```

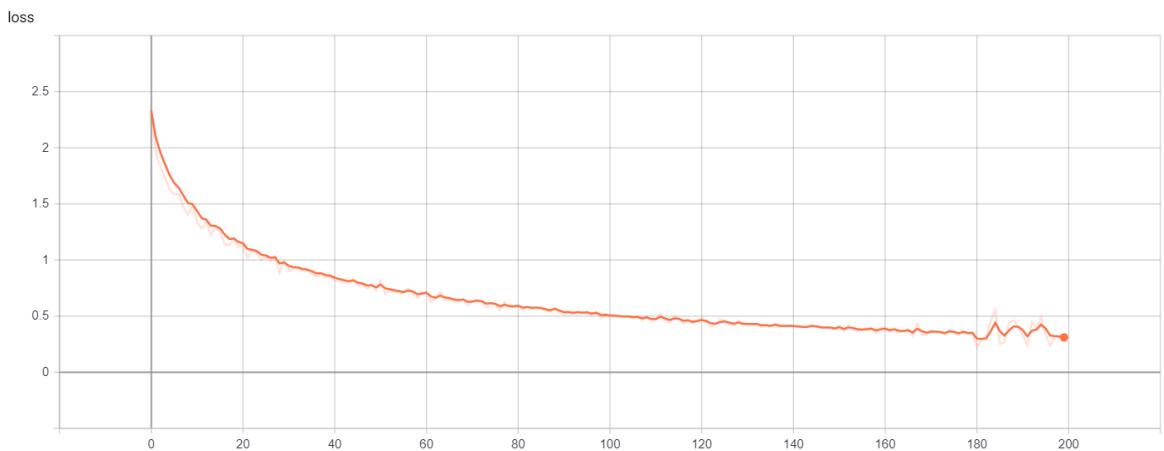
Hình 3.7: Các thông số cấu hình cho mạng

3.3 Kết quả thực nghiệm

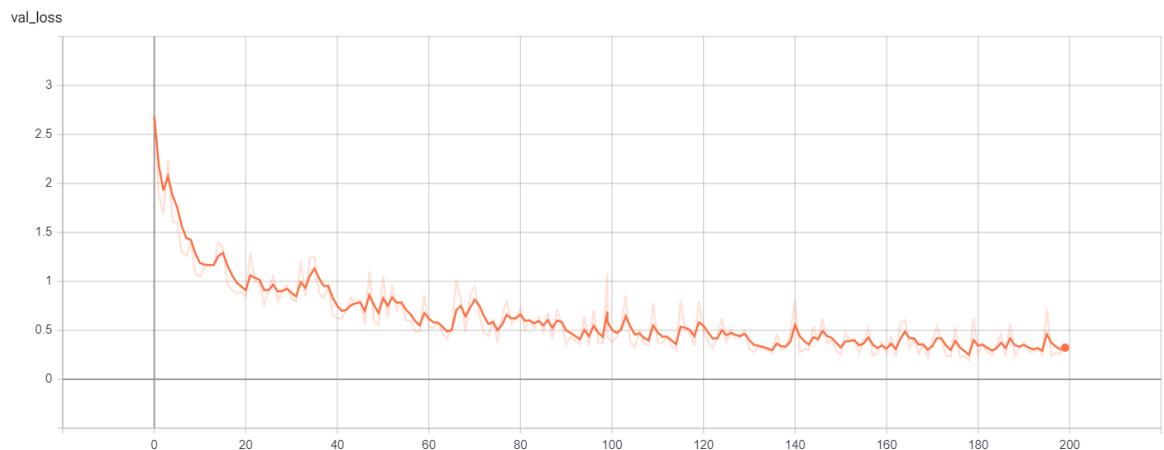
Mô hình đạt hiệu quả tương đối tốt so với yêu cầu. Hình 3.8 cho thấy kết quả dự đoán của mô hình, các hình 3.9 và 3.10 mô tả biểu đồ của hàm mất mát trên tập huấn luyện và tập validation qua 200 epoch. Hàm mất mát ở đây là kết hợp của các hàm mất mát của nhiều tác vụ khác nhau trên mô hình. Từ đây, có thể thấy rằng mô hình của chúng ta có hàm mất mát giảm trên cả tập huấn luyện và tập validation và thu được một kết quả tốt.



Hình 3.8: Kết quả thực nghiệm: bên trái là hình đúng, bên phải là dự đoán của mô hình.



Hình 3.9: Biểu đồ hàm mất mát của tập huấn luyện qua các epoch



Hình 3.10: Biểu đồ hàm mất mát của tập validation qua các epoch

KẾT LUẬN

Trong quá trình làm đồ án 2, em đã tìm hiểu được nhiều về các mô hình trong lĩnh vực thị giác máy tính và học máy. Em đã triển khai một số mô hình nhỏ và triển khai mô hình Mask R-CNN cho bài toán trong báo cáo này. Kết quả tương đối so với yêu cầu của bài toán.

Tuy nhiên cũng có thể thấy rõ mô hình vẫn còn nhiều nhược điểm chưa được giải quyết như:

- 1 Thời gian chạy chương trình chậm, chưa thể đạt đến thời gian thực, vẫn đề này chưa thực sự quan trọng với bài toán của chúng ta nhưng sẽ cần thiết trong nhiều bài toán phân đoạn đối tượng khác.
- 2 Đầu ra của mô hình chưa phải dạng Polygon, một pixel có thể bị thuộc về nhiều đối tượng hay phân đoạn bị chồng lấn với nhau.

Với nhược điểm đầu tiên, em đề xuất một số mô hình có tốc độ nhanh hơn như là: Yolact, CenterMask, InstaBoost, SOLOv2, ... VỚI nhược điểm thứ hai, em đề xuất hướng cải tiến là thay lớp Mask của Mask R-CNN bằng lớp RNN của mô hình Polygon-RNN++. Mặc dù đã có những ý tưởng cải tiến nhưng vì mới tìm hiểu về lĩnh vực thị giác máy tính và với thời gian eo hẹp nên em chưa thực hiện được những mong muốn của mình trong báo cáo này và đó cũng là hướng nghiên cứu tiếp theo của bài toán. Hy vọng sẽ nhận được thêm sự góp ý của thầy cô và các bạn để có thể có những hướng phát triển tiếp theo tốt hơn.

Tài liệu tham khảo

- [1] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation", In CVPR, 2014.
- [2] R. Girshick. "Fast R-CNN". arXiv:1504.08083, 2015.
- [3] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks", In NIPS, 2015.
- [4] K. He, G. Gkioxari, P. Dollar, and R. Girshick. "Mask r-cnn", arXiv:1703.06870, 2017.
- [5] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. "Selective search for object recognition", IJCV, 2013.
- [6] K. He, X. Zhang, S. Ren, and J. Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition", In ECCV. 2014.
- [7] Aston Zhang, Zack C. Lipton, Mu Li. Alex J. Smola, *Dive into Deep Learning*, d2l.ai, 2019.
- [8] Adrian Rosebrock, *Deep learning for Computer vision with Python*, pyimagesearch, 2017.
- [9] Zhongling Huang, Zongxu Pan, Bin Lei. "Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data", MDPI, 2017.
- [10] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. "Yolact: Real-time instance segmentation", In ICCV, 2019.
- [11] Youngwan Lee and Jongyoul Park. "CenterMask: Real-time anchor-free instance segmentation". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
- [12] Yuqing Wang, Zhaoliang Xu, Hao Shen, Baoshan Cheng, Lirong Yang. "Center-Mask: Single Shot Instance Segmentation With Point Representation", In CVPR, 2020.

- [13] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yonglu Li, and Cewu Lu. "Instaboost: Boosting instance segmentation via probability map guided copy-pasting", In The IEEE International Conference on Computer Vision (ICCV), Oct 2019.
- [14] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. "SOLO: Segmenting objects by locations", arXiv preprint arXiv:1912.04488, 2019.
- [15] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. "Solov2: Dynamic, faster and stronger", arXiv preprint arXiv:2003.10152, 2020.
- [16] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. "Annotating object instances with a Polygon-RNN", In CVPR, 2017.