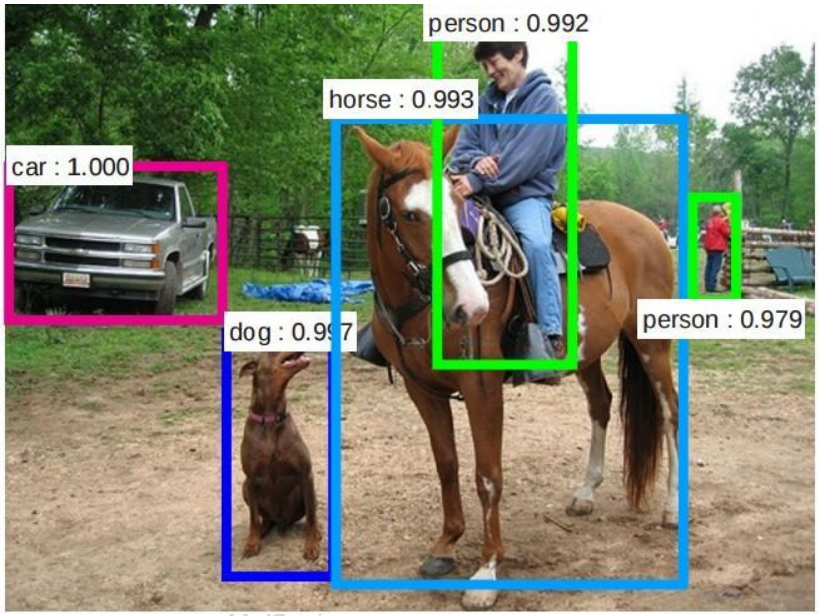
1. Bài toán object detection

Trong bài 7, sách đã giới thiệu về ứng dụng mô hình CNN cho bài toán phân loại ảnh, tuy nhiên các ảnh input của bài toán phân loại chỉ bao gồm 01 đối tượng cụ thể như chữ số hay 01 loài hoa.

Hình 13.1: Ví dụ ảnh trong bài toán phân loại ảnh

Tuy nhiên là ảnh trong cuộc sống bình thường thì không chỉ có 01 đối tượng mà thường chứa nhiều các đối tượng khác. Từ đó nảy sinh vấn đề cần tìm vị trí của từng đối tượng trong ảnh. Đó là bài toán object detection.



Hình 13.2: Ví dụ output của object detection [22]

Bài toán object detection có input là ảnh màu và output là vị trí của các đối tượng trong ảnh. Ta thấy nó bao gồm 2 bài toán nhỏ:

- Xác định các bounding box (hình chữ nhật) quanh đối tượng.
- Với mỗi bounding box thì cần phân loại xem đấy là đối tượng gì (chó, ngựa, ô tô,...) với bao nhiêu phần trăm chắc chắn.

Việc lựa chọn có bao nhiêu loại đối tượng thì phụ thuộc vào bài toán mà ta đang giải quyết.

Bạn tự hỏi liệu mô hình CNN có giải quyết được bài toán object detection không? Vấn đề chính là vì không biết trước có bao nhiêu đối tượng trong ảnh, nên không thiết kế được output layer hiệu quả => mô hình CNN truyền thống không giải quyết được => R-CNN (regional convolutional neural network) ra đời.

2. Faster R-CNN

2.1 R-CNN (Region with CNN feature)

Ý tưởng thuật toán R-CNN khá đơn giản

- Bước 1: Dùng Selective Search algorithm để lấy ra khoảng 2000 bounding box trong input mà có khả năng chứa đối tượng.
- Bước 2: Với mỗi bounding box ta xác định xem nó là đối tượng nào (người, ô tô, xe đạp,...)

2.1.1 Selective search algorithm

Input của thuật toán là ảnh màu, output là khoảng 2000 region proposal (bounding box) mà có khả năng chứa các đối tượng.

Đầu tiên ảnh được segment qua thuật toán Graph Based Image Segmentation, vì thuật toán dựa vào lý thuyết đồ thị và không áp dụng deep learning nên sách không giải thích chi tiết, bạn có thể đọc theo link ở dưới để tìm hiểu thêm.

Hình 13.3: Output sau khi thực hiện graph based image segmentation [23]

Nhận xét: Ta không thể dùng mỗi màu trong output để làm 1 region proposal được vì:

- Mỗi đối tượng trong ảnh có thể chứa nhiều hơn 1 màu.
- Các đối tượng bị che mất một phần như cái đĩa dưới cái chén không thể xác định được.

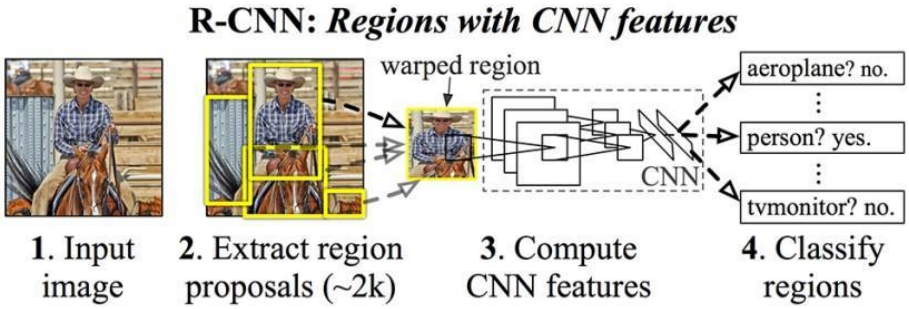
=> Cần nhóm các vùng màu với nhau để làm region proposal.

Tiếp theo, các vùng màu được nhóm với nhau dựa trên độ tương đồng về màu sắc, hướng gradient, kích thước,...

Cuối cùng các region proposal được xác định dựa trên các nhóm vùng màu.

2.1.2 Phân loại region proposal

Bài toán trở thành phân loại ảnh cho các region proposal. Do thuật toán selective search cho tới 2000 region proposal nên có rất nhiều region proposal không chứa đối tượng nào. Vậy nên ta cần thêm 1 lớp background (không chứa đối tượng nào). Ví dụ như hình dưới ta có 4 region proposal, ta sẽ phân loại mỗi bounding box là người, ngựa hay background.



Hình 13.4: Các bước trong RCNN [8]

Sau đó các region proposal được resize lại về cùng kích thước và thực hiện transfer learning với feature extractor, sau đó các extracted feature được cho vào thuật toán SVM để phân loại ảnh.

Bên cạnh đó thì extracted feature cũng được dùng để dự đoán 4 offset values cho mỗi cạnh. Ví dụ như khi region proposal chứa người nhưng chỉ có phần thân và nửa mặt, nửa mặt còn lại không có trong region proposal đó thì offset value có thể giúp mở rộng region proposal để lấy được toàn bộ người.

2.1.3 Vấn đề với R-CNN

Hồi mới xuất hiện thì thuật toán hoạt động khá tốt cho với các thuật toán về computer vision trước đó nhờ vào CNN, tuy nhiên nó vẫn có khá nhiều hạn chế:

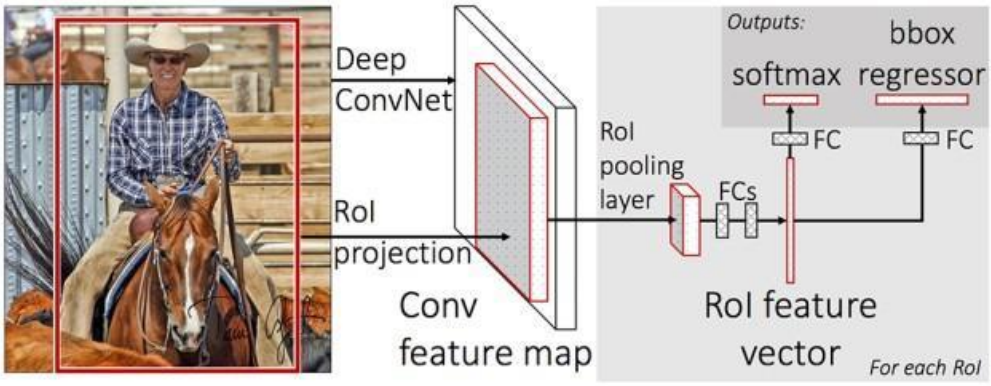
- Vì với mỗi ảnh ta cần phân loại các class cho 2000 region proposal nên thời gian train rất lâu.
- Không thể áp dụng cho real-time vì mỗi ảnh trong test set mất tới 47s để xử lý.

2.2 Fast R-CNN

Khoảng 1.5 năm sau đó, Fast R-CNN được giới thiệu bởi cùng tác giả của R-CNN, nó giải quyết được một số hạn chế của R-CNN để cải thiện tốc độ.

Tương tự như R-CNN thì Fast R-CNN vẫn dùng selective search để lấy ra các region proposal. Tuy nhiên là nó không tách 2000 region proposal ra khỏi ảnh và thực hiện bài toán image classification cho mỗi ảnh. Fast R-CNN cho cả bức ảnh vào ConvNet (một vài convolutional layer + max pooling layer) để tạo ra convolutional feature map.

Sau đó các vùng region proposal được lấy ra tương ứng từ convolutional feature map. Tiếp đó được Flatten và thêm 2 Fully connected layer (FCs) để dự đoán lớp của region proposal và giá trị offset values của bounding box.



Hình 13.5: Các bước trong Fast RCNN [7]

Tuy nhiên là kích thước của các region proposal khác nhau nên khi Flatten sẽ ra các vector có kích thước khác nhau nên không thể áp dụng neural network được. Thử nhìn lại xem ở trên R-CNN đã xử lý như thế nào? Nó đã resize các region proposal về cùng kích thước trước khi dùng transfer learning. Tuy nhiên ở feature map ta không thể resize được, nên ta phải có cách gì đấy để chuyển các region proposal trong feature map về cùng kích thước => Region of Interest (ROI) pooling ra đời.

2.2.1 Region of Interest (ROI) pooling

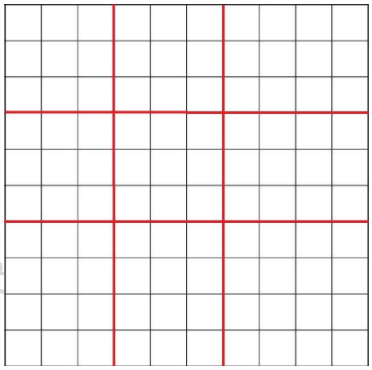
ROI pooling là một dạng của pooling layer. Điểm khác so với max pooling hay average pooling là bất kể kích thước của tensor input, ROI pooling luôn cho ra output có kích thước cố định được định nghĩa trước.

Ta kí 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.

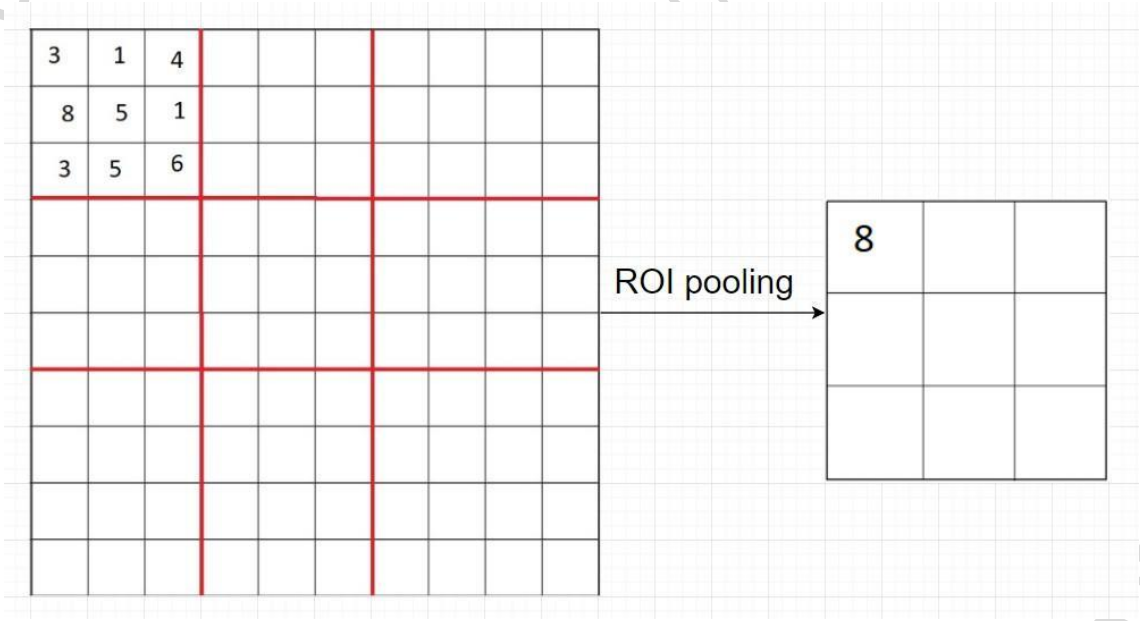
Gọi input của ROI pooling kích thước m*n và output có kích thước h*k (thông thường h, k nhỏ ví dụ 7*7).

- Ta chia chiều rộng thành h phần, (h-1) phần có kích thước m/h, phần cuối có kích thước m/h + m%h.
- Tương tự ta 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.



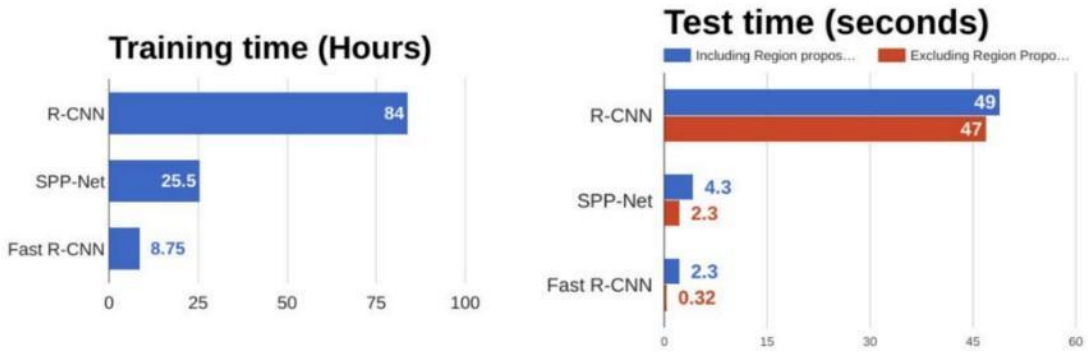
Sau đó với mỗi khối được tạo ra bằng các đường đỏ và cạnh, ta thực hiện max pooling lấy ra 1 giá trị.



Hình 13.6: Thực hiện ROI pooling

Ta có thể thấy là kích thước sau khi thực hiện ROI pooling về đúng h*k như ta mong muốn.

2.2.2 Đánh giá Fast R-CNN



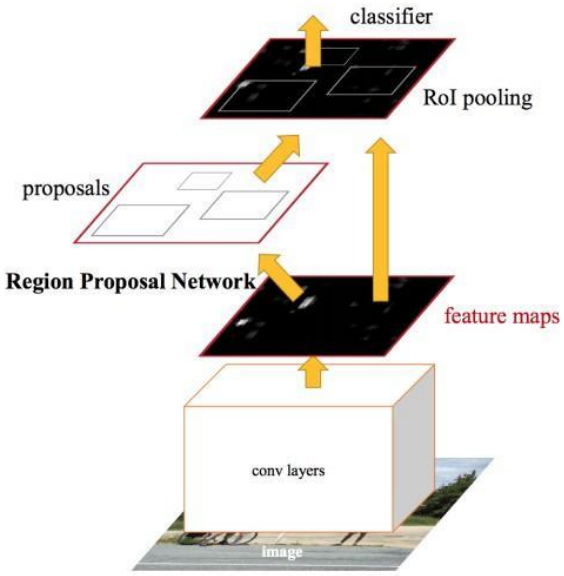
Hình 13.7: So sánh thời train train và test giữa R-CNN và Fast R-CNN [17]

Fast R-CNN khác với R-CNN là nó thực hiện feature map với cả ảnh sau đó với lấy các region proposal ra từ feature map, còn R-CNN thực hiện tách các region proposal ra rồi mới thực hiện CNN trên từng region proposal. Do đó Fast R-CNN nhanh hơn đáng kể nhờ tối ưu việc tính toán bằng Vectorization.

Tuy nhiên nhìn hình trên ở phần test time với mục Fast R-CNN thì thời gian tính region proposal rất lâu và làm chậm thuật toán => Cần thay thế thuật toán selective search. Giờ người ta nghĩ đến việc dùng deep learning để tạo ra region proposal => Faster R-CNN ra đời.

2.3 Faster R-CNN

Faster R-CNN không dùng thuật toán selective search để lấy ra các region proposal, mà nó thêm một mạng CNN mới gọi là Region Proposal Network (RPN) để tìm các region proposal.



Hình 13.8: Kiến trúc mới Faster R-CNN [18]

Đầu tiên cả bức ảnh được cho qua pre-trained model để lấy feature map. Sau đó feature map được dùng cho Region Proposal Network để lấy được các region proposal. Sau khi lấy được vị trí các region proposal thì thực hiện tương tự Fast R-CNN.

2.3.1 Region Proposal Network (RPN)

Input của RPN là feature map và output là các region proposal. Ta thấy các region proposal là hình chữ nhật.

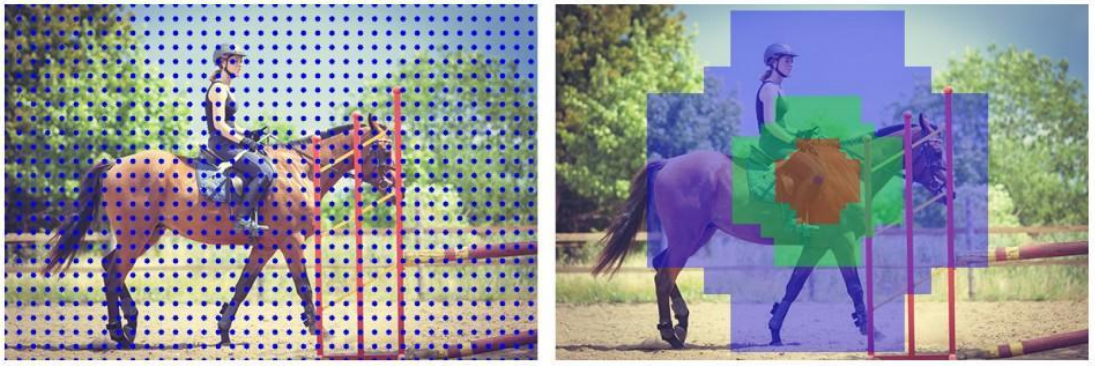
Mà một hình chữ nhật được xác định bằng 2 điểm ở 2 góc, ví dụ A(x_min, y_min) và B(x_max, y_max). Nhận xét:

- Khi RPN dự đoán ta phải rằng buộc x_min < x_max và y_min < y_max.
- Hơn nữa các giá trị x,y khi dự đoán có thể ra ngoài khỏi bức ảnh

=> Cần một kĩ thuật mới để biểu diễn region propsal => Anchor ra đời.

Ý tưởng là thay vì dự đoán 2 góc ta sẽ dự đoán điểm trung tâm (x_center, y_center) và width, height của hình chữ nhật. Như vậy mỗi anchor được xác định bằng 4 tham số (x_center, y_center, width, height).

Vì không sử dụng Selective search nên RPN ban đầu cần xác định các anchor box có thể là region proposal, sau đó qua RPN thì chỉ output những anchor box chắc chắn chứa đối tượng.



Hình 13.9: Ví dụ về anchor [21]

Ảnh bên trái kích thước 400 * 600 pixel, các tâm của anchor box màu xanh, cách nhau 16 pixel => 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 × 64, 128 × 128, 256×256, mỗi kích thước có 3 tỉ lệ tương ứng: 1 : 1, 1 : 2 và 2 : 1.

Giống như hình bên phải với tâm ở giữa 3 kích thước ứng với màu da cam, xanh lam, xanh lục và với mỗi kích thước có 3 tỉ lệ.

=> Số lượng anchor box giờ là 938 * 9 = 8442 anchors. Tuy nhiên sau RPN ta chỉ giữ lại khoảng 1000 anchors box để thực hiện như trong Fast R-CNN.

Việc của RPN là lấy ra các region proposal giống như selective search thôi chứ không phải là phân loại ảnh.

Mô hình RPN khá đơn giản, feature map được cho qua Conv layer 3*3, 512 kernels. Sau đó với mỗi anchor lấy được ở trên, RPN thực hiện 2 bước:

1. Dự đoán xem anchor đấy là foreground (chứa object) hay background (không chứa object)

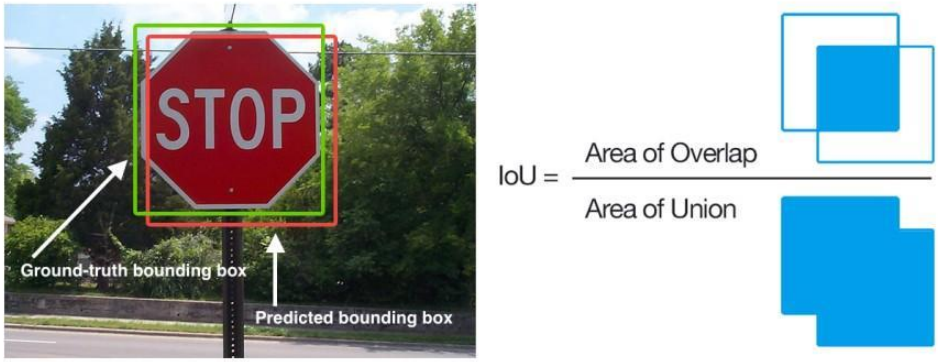
2. Dự đoán 4 offset value cho x_center, y_center, width, height cho các anchor.

Nhận xét: có rất nhiều anchor bị chồng lên nhau nên non-maxima suppression được dùng để loại bỏ các anchor chồng lên nhau.

Sau cùng dựa vào phần trăm dự đoán background RPN sẽ lấy N anchor (N có thể 2000, 1000, thậm chí 100 vẫn chạy tốt) để làm region proposal.

2.3.2 Intersection over Union (IoU)

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 13.10: Ví dụ về IoU [11]

Ví dụ về hệ số IoU, nhận xét:

- Chỉ số IoU trong khoảng [0,1]
- IoU càng gần 1 thì bounding box dự đoán càng gần ground truth