(Bản nháp) Object Detection: R-CNN

June 13, 2018

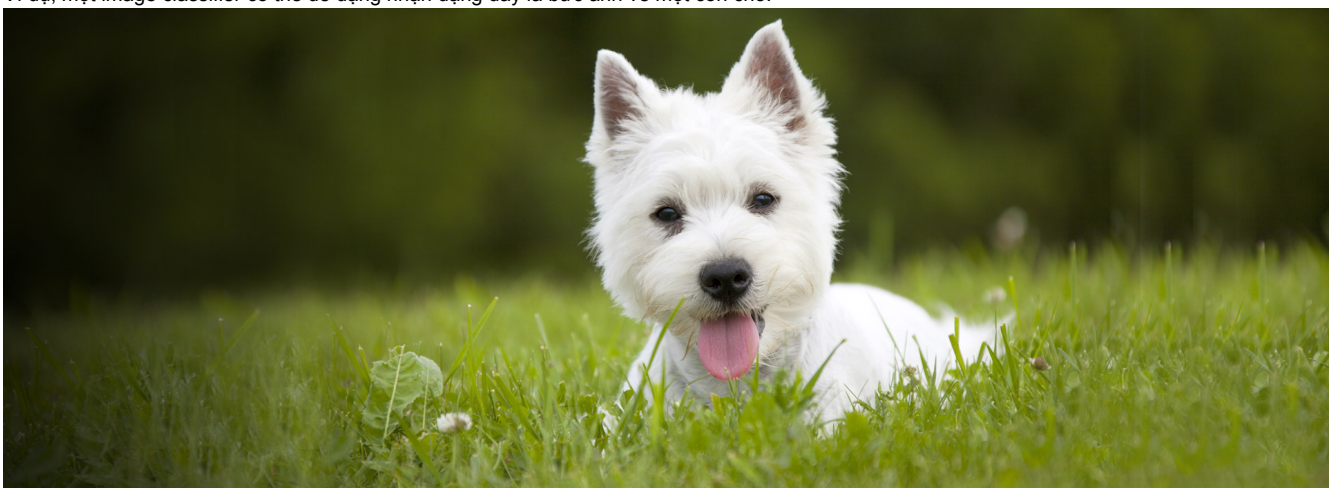
Khi viết bài này, tôi giả sử rằng bạn đọc đã biết các khái niệm về các thành phần của một Convolutional Neural Network và một vài CNN cơ bản, ví dụ AlexNet. Tôi sẽ viết các kiến thức cơ bản này trên blog Machine Learning cơ bản (https://machinelearningcoban.com) trong một ngày không xa

1. Các khái niệm cơ bản

1.1. Bài toán object detection

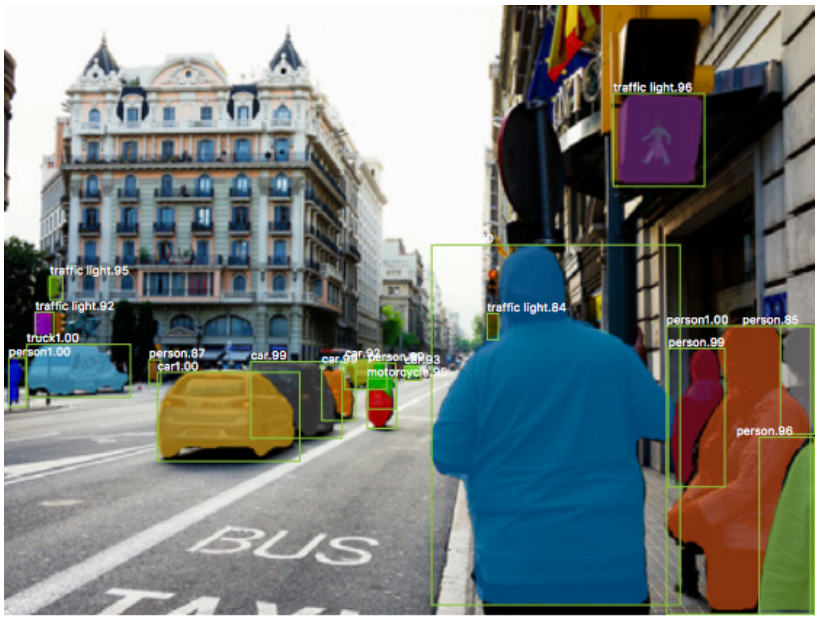
Chúng ta đã quen thuộc với bài toán image classification. Cho một bức ảnh, ta cần xác định xem bức ảnh đó nói về chủ đề gì, hoặc chứa vật thể (object) gì. Thông thường, mỗi bức ảnh thường chứa một object và *khung* của bức ảnh gần với *khung* chứa đối tượng.

Ví dụ, một image classifier có thể dễ dạng nhận dạng đây là bức ảnh về một con chó:



Tuy nhiên, con người làm tốt hơn một classifier rất nhiều. Con người có thể nhận diện ra vị trí con chó ở chỗ nào, trong khi các classifier chỉ biết được rằng trong bức ảnh có con chó.

Bài toán object detection là một bài toán phức tạp hơn bài toán classification. Cho một bức ảnh, hệ thống cần phải xác định được vị trí và hình dạng của các vật thể (object) trong ảnh, đồng thời phải xác định xem đó là object nào (category nào). Dưới đây là một ví dụ:



(Nguồn: Mask R-CN paper (https://arxiv.org/abs/1703.06870))

Trong hình trên, các pixel thuộc cùng một object được cho bởi các vùng đồng màu (được gọi là *mask*). Các hình chữ nhật bao quanh các object được gọi là *bounding box* (hình bao).

Bài toán object detection trong Computer Vision là bài toán đi tìm các bounding box của các object, đồng thời xác định xem đó là object gì, hoặc cụ thể hơn là xác suất object đó rơi vào từng category là bao nhiêu.

Input: Một bức ảnh

Output: bounding box của object và label của object đó.

Bài toán đi tìm *mask* của mỗi object được gọi là bài toán *instance segmentation*. Bài toán này cũng yêu cầu xác định object nào ở đằng sau mask đó.

Trong loạt bài viết này, tôi hy vọng sẽ có cơ hội chia sẻ với các bạn về các thuật toán object detection (và Mask R-CNN cho instance segmentation) kể từ khi *kỷ nguyên deep learning* bắt đầu vào năm 2012.

Các mô hình đó bao gồm:

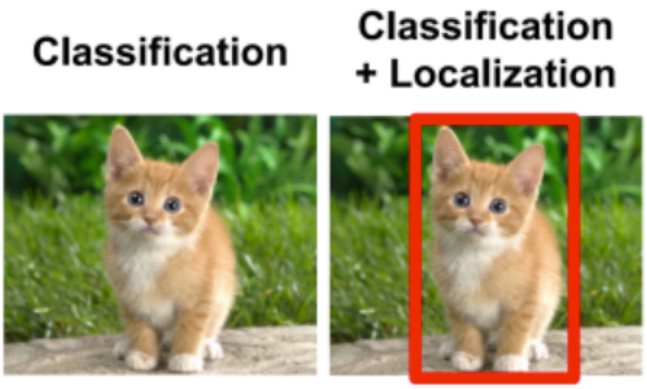
1. R-CNN (https://arxiv.org/abs/1311.2524)
2. Fast R-CNN (https://arxiv.org/abs/1504.08083)
3. Faster R-CNN (https://arxiv.org/abs/1506.01497)
4. YOLO (https://arxiv.org/abs/1506.02640)
5. SSD (https://arxiv.org/abs/1512.02325)
6. YOLOv2 (https://arxiv.org/abs/1612.08242)
7. YOLOv3 (https://arxiv.org/abs/1804.02767)
8. (có thể) [SPPNet]
9. (có thể) Mask R-CNN (https://arxiv.org/abs/1703.06870)

Trước khi đi sâu vào các mô hình, chúng ta cùng tìm hiểu một vài khái niệm quan trọng.

1.2. Bounding box

Bounding là một hình chữ nhật bao quanh object. Hình chữ nhật này càng 'khít' với object càng tốt.

Ví dụ:



Một bounding box thường được xác định bởi bốn tham số (x, y, w, h) trong đó x, y là toạ độ tâm (giao điểm hai đường chéo), w, h là chiều rộng và chiều cao của bounding box đó.

1.3. Intersection over Union (IoU)

IoU (Intersection over Union) là một phép đo sự khớp nhau của hai bounding box. Nó được tính bằng tỉ lệ giữa phần diện tích giao nhau (intersection) và hợp nhau (union) của hai boudning box đó. Ta luôn có $0 \leq \text{IoU} \leq 1$. Giá trị IoU càng cao chứng tỏ hai box càng khớp nhau. Khi hai box không giao nhau, IoU = 0. IoU = 1 khi và chỉ khi hai box trùng khít nhau.

1.4. mAp

Tôi sẽ đề cập mAP (mean of average precision) trong bản chính thức của bài viết này. Bạn đọc có thể đoc thêm mAP (mean Average Precision) for Object Detection (https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173)

1.5. Các dataset thông dụng

- Pascal VOC (http://host.robots.ox.ac.uk/pascal/VOC/)
- COCO (http://cocodataset.org/#home)

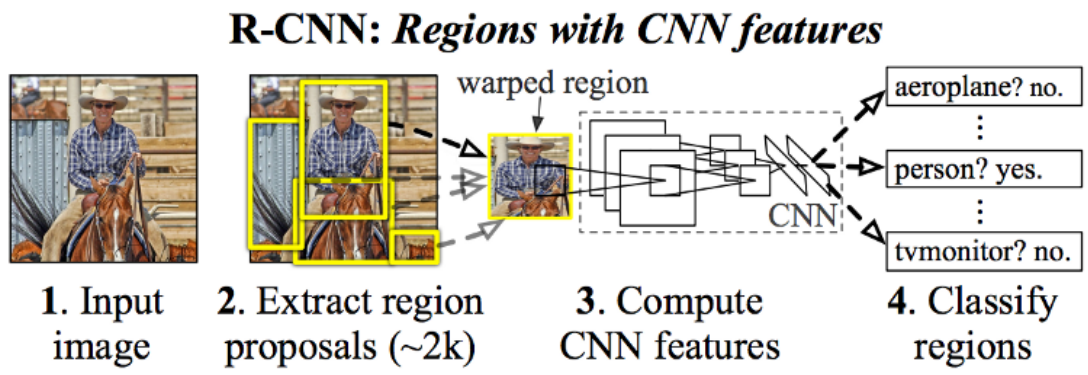
Các dataset này chứa các boudning box của từng object trong ảnh và category của object đó.

2. R-CNN (Regional CNN)

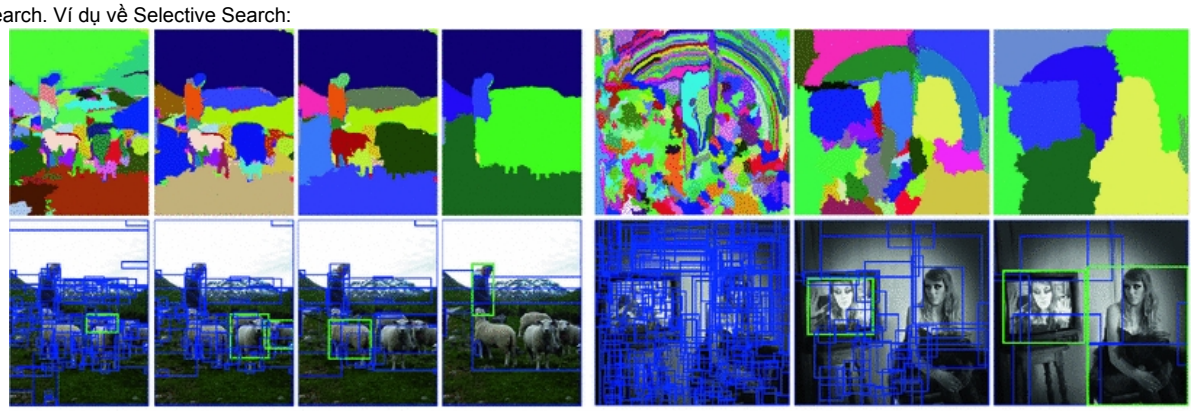
Ý tưởng cơ bản của R-CNN là tìm các khung có thể chứa đối tượng, được gọi là các region proposal. Sau đó phân lớp từng region proposal đó (xem nó thuộc vào category nào hay chỉ là một vùng không mang thông tin gì -- background -- trong ảnh) và tinh chỉnh vị trí và kích thước của các

region proposal đó. Các region proposal đã được tinh chỉnh có score cao ứng với category được giữ lại.

2.1. Kiến trúc



1. Từ bức ảnh ban đầu, các vùng tiềm năng hình chữ nhật (proposals) được xác định (khoảng ~2000 vùng) bằng phương pháp Selective Search. Ví dụ về Selective Search:



   Selective Search là một thuật toán đi tìm các vùng có khả năng chứa vật thể trong ảnh. Về cơ bản, bức ảnh trước hết được *over segmented* thành các *mảnh* nhỏ chứa các bộ phận nhỏ của một object, sau đó các mảnh gần nhau được *gom* lại dần dần tạo thành các vùng *có thể là* các object lớn hơn. Để không bị *sót* object nào, ban đầu rất nhiều region proposals được chọn ra (khoảng ~2k cho mỗi bức ảnh)
2. Mỗi Proposal này được resize về chung một kích thước (ví dụ 227x227, 224x224) để đưa qua một CNN, tạo ra một feature vector.
3. Mỗi feature vector này được đưa qua C linear SVM (mỗi cho một category) để tạo ra các *score* cho từng category. Score càng cao thể hiện khả năng proposal đó rơi vào category tương ứng càng lớn. Chú ý: Khác với logistic regression, bản thân SVM không sinh ra score (hay xác suất để object rơi vào một category), tác giả không giải thích cụ thể score được sinh ra như thế nào. Tôi đoán rằng tác giả dùng cách tính score bằng SVM (http://scikit-learn.org/stable/modules/svm.html#scores-probabilities) như trong tài liệu của scikit-learn.)
4. Các proposal mà tất cả các score nhỏ được bỏ đi vì khả năng cao đó là background. Các proposal còn lại được loại bỏ dần khi xét tới từng category. Với mỗi category, xét tất cả các proposal mà score của nó theo category đủ lớn. Nếu hai proposal chồng lấn nhau quá nhiều (IoU lớn hơn một đại lượng nào đó) thì proposal có score nhỏ hơn được bỏ đi.
5. Sau bước 4, các bounding box tìm được sẽ được đưa qua một *bounding box regressor* (của category tương ứng, mỗi category có một BB regressor khác nhau). **Input**: bounding box x, y, w, h, đầu ra của pooling layer thứ 4, và category tương ứng. **Output**: bouding box mới (khít hơn với object) x_1, y_1, w_1, h_1.

2.2. Training

Có ba khối cần được *trained* trong R-CNN: CNN, các category-dependent SVM, và các category-dependent BB regressor.

2.2.1. CNN

1. Sử dụng pre-trained AlexNet (R-CNN được viết năm 2013, lúc đó chưa có VGG, ResNet, GoogLeNet, ... Ta có thể thay AlexNet bằng các mạng về sau để có kết quả tốt hơn). Fully connected layer ở cuối của AlexNet và softmax layer tương ứng được bỏ đi và thay bằng một fully connected layer mới và softmax layer mới với $C + 1$ đầu ra (C object + 1 background).
2. Các region proposals với IoU với ground truth box ≥ 0.5 được coi là *positive sample* của class đó. Các region proposals không giao với ground truth box nào (tất cả IoU ≤ 0.5) được coi như background. Các region proposals này được *warped* về cùng kích thước với đầu vào của AlexNet (227x227). Chú ý, trước khi được *warped*, các region proposals được lấy thêm 16 pixel về mỗi phía để đảm bảo rằng chúng chứa trọn vẹn object).
3. Stochastic Gradient Descent được dùng để huấn luyện CNN này. Mỗi mini-batch gồm 128 proposals, trong đó có 96 background proposals và 32 foreground proposals (các foreground proposals có thể thuộc các category khác nhau).

Sau khi train xong CNN này, output của pooling layer thứ 5 (pooling layer cuối cùng của R-CNN) được coi như feature vector của warped region proposal ở đầu vào. Các feature vectors này được lưu lại để huấn luyện các SVM và các BB regressor sau này. (Ta có thể thấy rằng việc này

proposal ở đầu vào. Các feature vectors này được lưu lại để huấn luyện các SVM và các BB regressor sau này. (Ta có thể thấy rằng việc này tương đối tốn bộ nhớ. Việc huấn luyện riêng từng mô hình buộc chúng ta phải lưu đầu ra của từng mô hình trước khi coi nó là đầu vào để huấn luyện các mô hình tiếp theo.)

2.2.2. Linear SVMs

Sau khi có các feature vectors cho các region proposals, một linear SVM được huấn luyện **cho mỗi category**. Cách định nghĩa *positive* và *negative* cũng khác với CNN.

Trong SVM của một category, chỉ các (feature vector của) ground truth box của category đó được coi là *positive*. Các proposal với it hơn 0.3 IoU với tất cả các ground truth box của category đó được coi là *negative*. (Bằng thực nghiệm, tác giả cho rằng cách định nghĩa positive và negative này mang lại kết quả cao hơn so với việc định nghĩa positive/negative như khi fine tune CNN).

Việc định nghĩa này làm giảm đáng kể training set cho mỗi SVM. Khi fine tune CNN, chúng ta cần lượng training data lớn hơn nhiều nên cần giữ lại các box với IoU từ 0.3 đến 1.

Câu hỏi đặt ra: Tại sao không dùng đầu ra của softmax layer sau khi fine-tuning để xác định category của mỗi proposal? Tác giả đã thử các này nhưng mAP trên VOC 2007 giảm đi đáng kể, từ 54.2% xuống 50.9%). Việc giảm đáng kể này được cho là đến từ việc cách định nghĩa các positives/negatives quá lỏng (ngưỡng IoU bằng 0.5).

2.2.3. Bounding box regression

Các region proposal có khả năng cao chứa object (một score tương ứng với một category nào đó cao) chưa hẳn đã là một bouding box tốt. Bouding box regression làm nhiệm vụ *tinh chỉnh* proposal tìm được trở thành một bouding box đáng tin cậy.

Input: proposal, được cho bởi các tham số của bounding box (P_x, P_y, P_w, P_h); category tương ứng; và output của pooling layer thứ 5 đã được vector hoá, ký hiệu là $\Phi_5(P)$.

Output: $(\hat{G}_x, \hat{G}_y, \hat{G}_w, \hat{G}_h)$ là các tham số của bounding box cuối cùng.

Quan hệ giữa chúng như sau:

$$\hat{G}_x = P_x + P_w \mathbf{w}_x \Phi_5(P)$$

$$\hat{G}_x = P_y + P_h \mathbf{w}_y \Phi_5(P)$$

$$\hat{G}_w = P_w \exp(\mathbf{w}_w \Phi_5(P))$$

$$\hat{G}_h = P_h \exp(\mathbf{w}_h \Phi_5(P))$$

Trong đó $\mathbf{w}_*(* \in \{x, y, w, h\})$ là các hệ số được huấn luyện thông qua các linear regression **cho từng category**. Để ý rằng khi các hệ số $\mathbf{w}_*$ đều bằng vector 0 thì $\hat{G} \equiv P$.

Với **từng category**, các hệ số $\mathbf{w}_*$ là nghiệm của bài toán linear regression:

$$\mathbf{w}_* = \arg\min_{\mathbf{w}_*} \sum_{i=1}^{N} (t_*^i - \mathbf{w}_*^T \Phi_5(P^i))^2 + \lambda \parallel \mathbf{w}_* \parallel_2^2$$

Trong đó:

1. Trong đó, N là số cặp training (proposal, ground truth) bounding box của category tương ứng. Mỗi proposal P ban đầu sẽ tương ứng với **nhiều nhất** một ground truth bounding box G. Với mỗi một proposal P trong một bức ảnh ở training set, ta tìm ground truth bounding box trong bức ảnh đó sao cho IoU với P là lớn nhất. Nếu IoU này nhỏ hơn 0.6, ta bỏ qua P, ngược lại cặp (P, G) sẽ được dùng làm training data để tìm các hệ số $\mathbf{w}_*$ cho category tương ứng với ground truth bounding box đó.
2. Với mỗi cặp (P, G), các *target* t_* được xác định như sau (ngược với hệ (1)):

$$t_x = (G_x - P_x) / P_w$$

$$t_y = (G_y - P_y) / P_h$$

$$t_w = \log(G_w / P_w)$$

$$t_h = \log(G_h / P_h)$$

(Bạn có thể muốn đọc thêm về Linear Regression (https://machinelearningcoban.com/))

Trong bài báo gốc, tác giả có đề cập tới việc lặp lại quá trình *bouding box fine tune* này nhiều lần. Tuy nhiên, họ tìm thấy rằng việc này không làm tăng kết quả cuối cùng.

3. Thảo luận

- R-CNN phải thực hiện việc tính feature vector cho từng proposal (tổng khoảng ~2k proposals cho một bức ảnh), vì vậy tốc độ của R-CNN rất chậm. Để xử lý một bức ảnh, R-CNN tốn khoảng 50s trên GPU.
- Ngoài ra, R-CNN cần lưu các feature vector trung gian để huấn luyện các SVM và bounding box regressor sau này.
- Ba khối chính: CNN, SVMs, BB regressor được huấn luyện riêng biệt. Việc này làm giảm đáng kể độ chính xác, vì chất lượng của các

SVM và BB regressor phụ thuộc vào các feature vectors tạo ra bởi CNN. Một năm sau R-CNN, tác giả chính viết bài Fast R-CNN **gộp** cả ba khối này lại huấn luyện đồng thời, giúp tăng đáng kể tốc độ tính toán.

- Trong bài báo, tác giả dùng SVM để tính score cho mỗi proposal thuộc vào từng category. Câu hỏi đặt ra là tại sao không dùng trực tiếp logistic regression. Thực tế, các mô hình object detection về sau không sử dụng SVM mà trực tiếp sử dụng đầu ra của softmax layer làm các score cho từng category.

4. Tài liệu tham khảo

[1] A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN (https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4)

[2] Object Localization and Detection (https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html)