Bài 25 - YOLO You Only Look Once

09 Mar 2020 - phamdinhkhanh

Menu

1. Giới thiệu chung

Có lẽ trong vài năm trở lại đây, object detection là một trong những đề tài rất hot của deep learning bởi khả năng ứng dụng cao, dữ liệu dễ chuẩn bị và kết quả ứng dụng thì cực kì nhiều. Các thuật toán mới của object detection như YOLO, SSD có tốc độ khá nhanh và độ chính xác cao nên giúp cho Object Detection có thể thực hiện được các tác vụ dường như là real time, thậm chí là nhanh hơn so với con người mà độ chính xác không giảm. Các mô hình cũng trở nên nhẹ hơn nên có thể hoạt động trên các thiết bị IoT để tạo nên các thiết bị thông minh.

Ngoài ra, một thuật toán object detection có thể tạo ra những ứng dụng rất đa dạng như: Đếm số lượng vật thể, thanh toán tiền tại quầy hàng, chấm công tự động, phát hiện vật thể nguy hiểm như súng, dao,…. và rất nhiều các ứng dụng khác. Có thể nói dường bất kì lĩnh vực nào cũng đều có thể ứng dụng được object detection.

Bên cạnh đó nguồn dữ liệu ảnh lại vô cùng đa dạng và sẵn có vì chỉ cần google là tìm được tất cả những gì bạn cần. Đó cũng là một ưu điểm để huấn luyện model object detection.

Chính vì tính ứng dụng rất cao, dễ chuẩn bị dữ liệu và huấn luyện mô hình đơn giản nên bài viết này tôi sẽ giới thiệu tới các bạn một thuật toán object detection state-of-art, đó chính là YOLO.

1.1. Mạng YOLO là gì?

Có nhiều bạn thắc mắc câu hỏi này. YOLO nghe khá giống slogan “You only live once” phải không? Nhưng YOLO trong object detection có nghĩa là “You only look once”. Tức là chúng ta chỉ cần nhìn 1 lần là có thể phát hiện ra vật thể. Rất nhanh và mạnh phải không nào?

Thật vậy, về độ chính xác thì YOLO có thể không phải là thuật toán tốt nhất nhưng nó là thuật toán nhanh nhất trong các lớp mô hình object detection. Nó có thể đạt được tốc độ gần như real time mà độ chính xác không quá giảm so với các model thuộc top đầu.

YOLO là thuật toán object detection nên mục tiêu của mô hình không chỉ là dự báo nhãn cho vật thể như các bài toán classification mà nó còn xác định location của vật thể. Do đó YOLO có thể phát hiện được nhiều vật thể có nhãn khác nhau trong một bức ảnh thay vì chỉ phân loại duy nhất

Top

một nhãn cho một bức ảnh.

Sở dĩ YOLO có thể phát hiện được nhiều vật thể trên một bức ảnh như vậy là vì thuật toán có những cơ chế rất đặc biệt mà chúng ta sẽ tìm hiểu bên dưới.

1.2. Yêu cầu

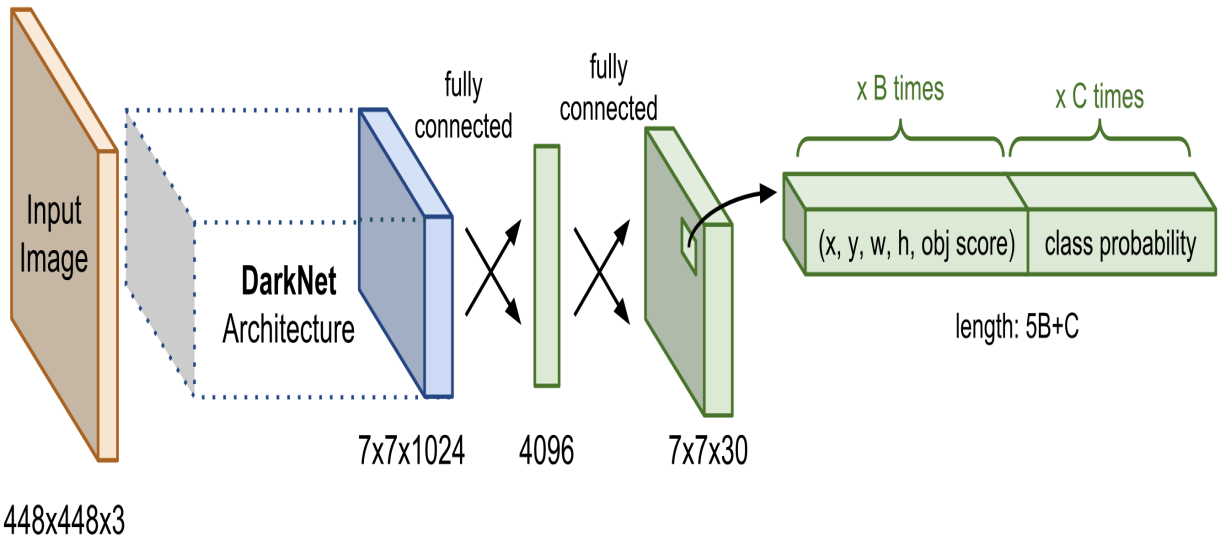
Trước khi đọc bài viết này chúng ta cần nắm vững lý thuyết:

- Nguyên lý hoạt động của mạng nơ ron tích chập (Convolutional Neural Network): Đây là mạng nơ ron áp dụng các layer Convolutional kết hợp với Maxpooling để giúp trích xuất đặc trưng của ảnh tốt hơn. Bạn đọc có thể tham khảo Lý thuyết về mạng tích chập neural (https://www.kaggle.com/phamdinhkhanh/convolutional-neural-network-p1).
- Khái niệm về bounding box, anchor box: Bounding box là khung hình bao quanh vật thể. Anchor box là những khung hình có kích thước xác định trước, có tác dụng dự đoán bounding box.
- Feature map: Là một khối output mà ta sẽ chia nó thành một lưới ô vuông và áp dụng tìm kiếm và phát hiện vật thể trên từng cell.
- Non-max suppression: Phương pháp giúp giảm thiểu nhiều bounding box overlap nhau về 1 bounding box có xác suất lớn nhất.

2. Kiến trúc mạng YOLO

Cũng tương tự như Bài 13 - thuật toán SSD (https://phamdinhkhanh.github.io/2019/10/05/SSDModelObjectDetection.html) tôi đã giới thiệu, kiến trúc YOLO bao gồm: base network là các mạng convolution làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những Extra Layers được áp dụng để phát hiện vật thể trên feature map của base network.

base network của YOLO sử dụng chủ yếu là các convolutional layer và các fully conntected layer. Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các version cho nhiều input shape khác nhau.



Hình 1: Sơ đồ kiến trúc mạng YOLO. Thành phần Darknet Architechture được gọi là base network có tác dụng trích suất đặc trưng. Output của base network là một feature map có kích thước `7x7x1024` sẽ được sử dụng làm input cho các Extra layers có tác dụng dự đoán nhãn và

Top

tọa độ bounding box của vật thể.

Trong YOLO version 3 tác giả áp dụng một mạng feature extractor là darknet-53. Mạng này gồm 53 convolutional layers kết nối liên tiếp, mỗi layer được theo sau bởi một batch normalization và một activation Leaky Relu. Để giảm kích thước của output sau mỗi convolution layer, tác giả down sample bằng các filter với kích thước là 2. Mẹo này có tác dụng giảm thiểu số lượng tham số cho mô hình.

	Type	Filters	Size	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
1x	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3 / 2	64 x 64
2x	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3 / 2	32 x 32
8x	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	16 x 16
8x	Convolutional	256	1 x 1	
	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	8 x 8
4x	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Hình 2: Các layer trong mạng darknet-53.

Các bức ảnh khi được đưa vào mô hình sẽ được scale để về chung một kích thước phù hợp với input shape của mô hình và sau đó được gom lại thành batch đưa vào huấn luyện.

Hiện tại YOLO đang hỗ trợ 2 đầu vào chính là `416x416` và `608x608`. Mỗi một đầu vào sẽ có một thiết kế các layers riêng phù hợp với shape của input. Sau khi đi qua các layer convolutional thì shape giảm dần theo cấp số nhân là 2. Cuối cùng ta thu được một feature map có kích thước tương đối nhỏ để dự báo vật thể trên từng ô của feature map.

Kích thước của feature map sẽ phụ thuộc vào đầu vào. Đối với input `416x416` thì feature map có các kích thước là `13x13`, `26x26` và `52x52`. Và khi input là `608x608` sẽ tạo ra feature map `19x19`, `38x38`, `72x72`.

Top

3. Output của YOLO

Output của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần:

$$y^T = [p_0, \langle \underbrace{t_x, t_y, t_w, t_h}_{\text{bounding box}} \rangle, \langle \underbrace{p_1, p_2, \ldots, p_c}_{\text{scores of c classes}} \rangle]$$

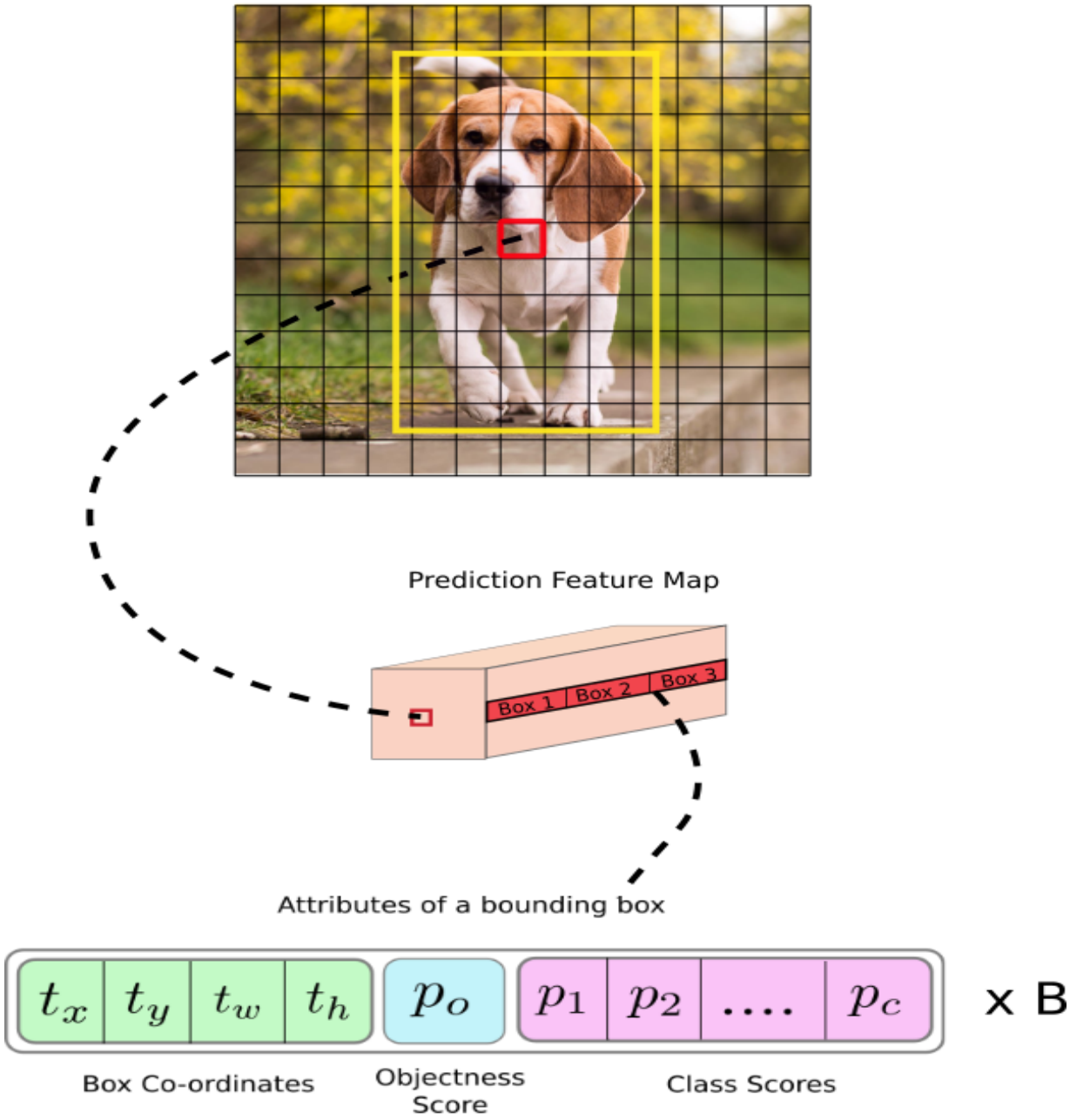
Trong đó

- p_0 là xác suất dự báo vật thể xuất hiện trong bounding box.
- $\langle \underbrace{t_x, t_y, t_w, t_h}_{\text{bounding box}} \rangle$ giúp xác định bounding box. Trong đó t_x, t_y là tọa độ tâm và t_w, t_h là kích thước rộng, dài của bounding box.
- $\langle \underbrace{p_1, p_2, \ldots, p_c}_{\text{scores of c classes}} \rangle$ là véc tơ phân phối xác suất dự báo của các classes.

Việc hiểu output khá là quan trọng để chúng ta cấu hình tham số chuẩn xác khi huấn luyện model qua các open source như darknet. Như vậy output sẽ được xác định theo số lượng classes theo công thức $(\text{n\_class} + 5)$. Nếu huấn luyện 80 classes thì bạn sẽ có output là 85. Trường hợp bạn áp dụng 3 anchors/cell thì số lượng tham số output sẽ là:

$$(\text{n\_class} + 5) \times 3 = 85 \times 3 = 255$$

Top

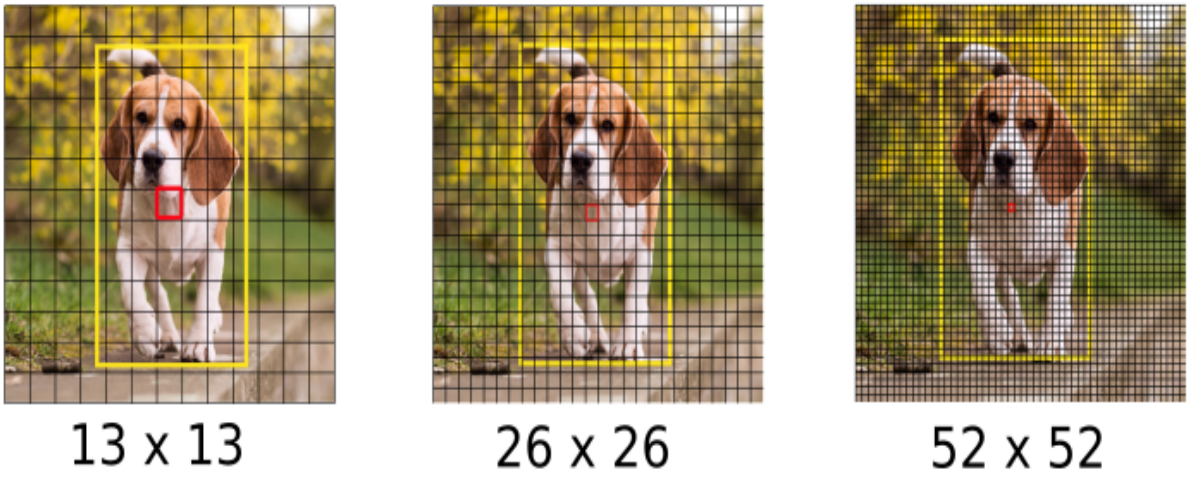


Hình 3: Kiến trúc một output của model YOLO. Hình ảnh gốc là một feature map kích thước `13x13`. Trên mỗi một cell của feature map chúng ta lựa chọn ra 3 anchor boxes với kích thước khác nhau lần lượt là `Box 1`, `Box 2`, `Box 3` sao cho tâm của các anchor boxes trùng với cell. Khi đó output của YOLO là một véc tơ concatenate của 3 bounding boxes. Các attributes của một bounding box được mô tả như dòng cuối cùng trong hình.

4. Dự báo trên nhiều feature map

Cũng tương tự như SSD, YOLOv3 dự báo trên nhiều feature map. Những feature map ban đầu có kích thước nhỏ giúp dự báo được các object kích thước lớn. Những feature map sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo các vật thể kích thước nhỏ.

Top



Hình 4: Các feature maps của mạng YOLOv3 với input shape là `416x416` , output là 3 feature maps có kích thước lần lượt là `13x13` , `26x26` và `52x52` .

Trên mỗi một cell của các feature map chúng ta sẽ áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 featue map x 3 anchor box).

Đồng thời trên một feature map hình vuông `S x S` , mô hình YOLOv3 sinh ra một số lượng anchor box là: `S x S x 3` . Như vậy số lượng anchor boxes trên một bức ảnh sẽ là:

$$(13 \times 13 + 26 \times 26 + 52 \times 52) \times 3 = 10647(\text{anchor boxes})$$

Đây là một số lượng rất lớn và là nguyên nhân khiến quá trình huấn luyện mô hình YOLO vô cùng chậm bởi chúng ta cần dự báo đồng thời nhãn và bounding box trên đồng thời 10647 bounding boxes.

Một số lưu ý khi huấn luyện YOLO:

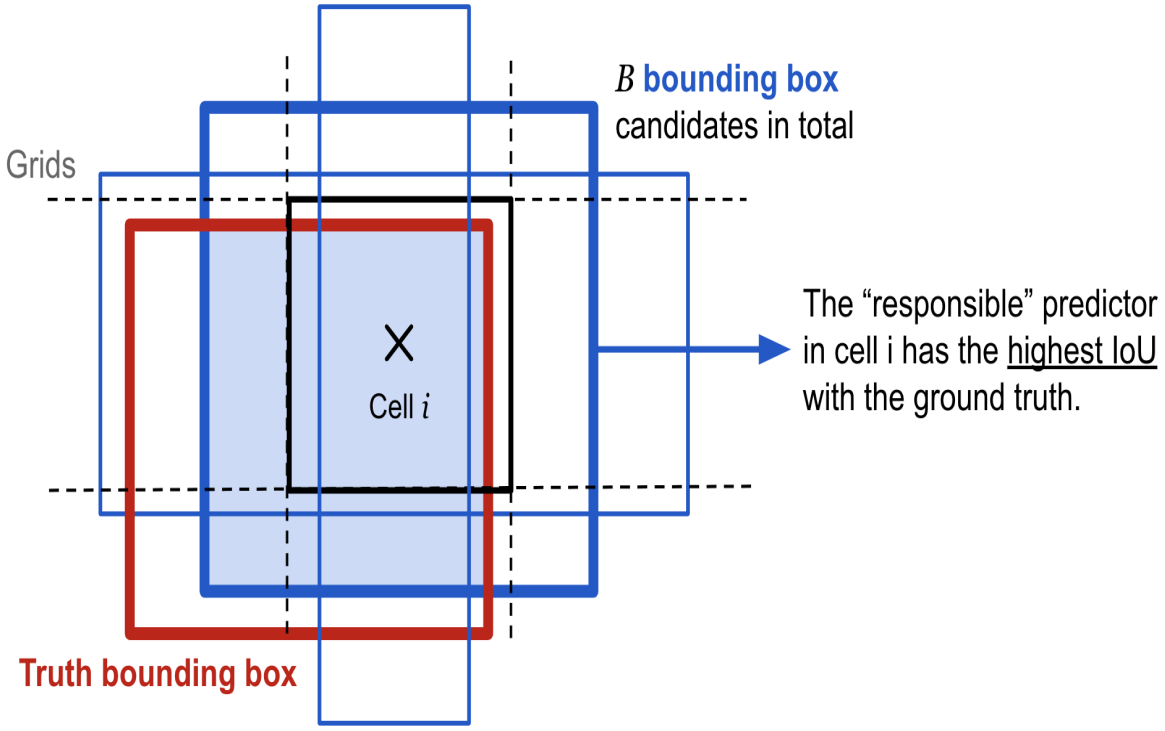
- Khi huấn luyện YOLO sẽ cần phải có RAM dung lượng lớn hơn để save được 10647 bounding boxes như trong kiến trúc này.
- Không thể thiết lập các batch_size quá lớn như trong các mô hình classification vì rất dễ Out of memory. Package darknet của YOLO đã chia nhỏ một batch thành các subdivisions cho vừa với RAM.
- Thời gian xử lý của một step trên YOLO lâu hơn rất rất nhiều lần so với các mô hình classification. Do đó nên thiết lập steps giới hạn huấn luyện cho YOLO nhỏ. Đối với các tác vụ nhận diện dưới 5 classes, dưới 5000 steps là có thể thu được nghiệm tạm chấp nhận được. Các mô hình có nhiều classes hơn có thể tăng số lượng steps theo cấp số nhân tùy bạn.

5. Anchor box

Để tìm được bounding box cho vật thể, YOLO sẽ cần các anchor box làm cơ sở ước lượng. Những anchor box này sẽ được xác định trước và sẽ bao quanh vật thể một cách tương đối chính xác. Sau này thuật toán regression bounding box sẽ tinh chỉnh lại anchor box để tạo ra bounding box dự đoán cho vật thể. Trong một mô hình YOLO:

- Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một anchor box. Trong trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchor box mà có IoU với ground truth bounding box là cao nhất.

Top



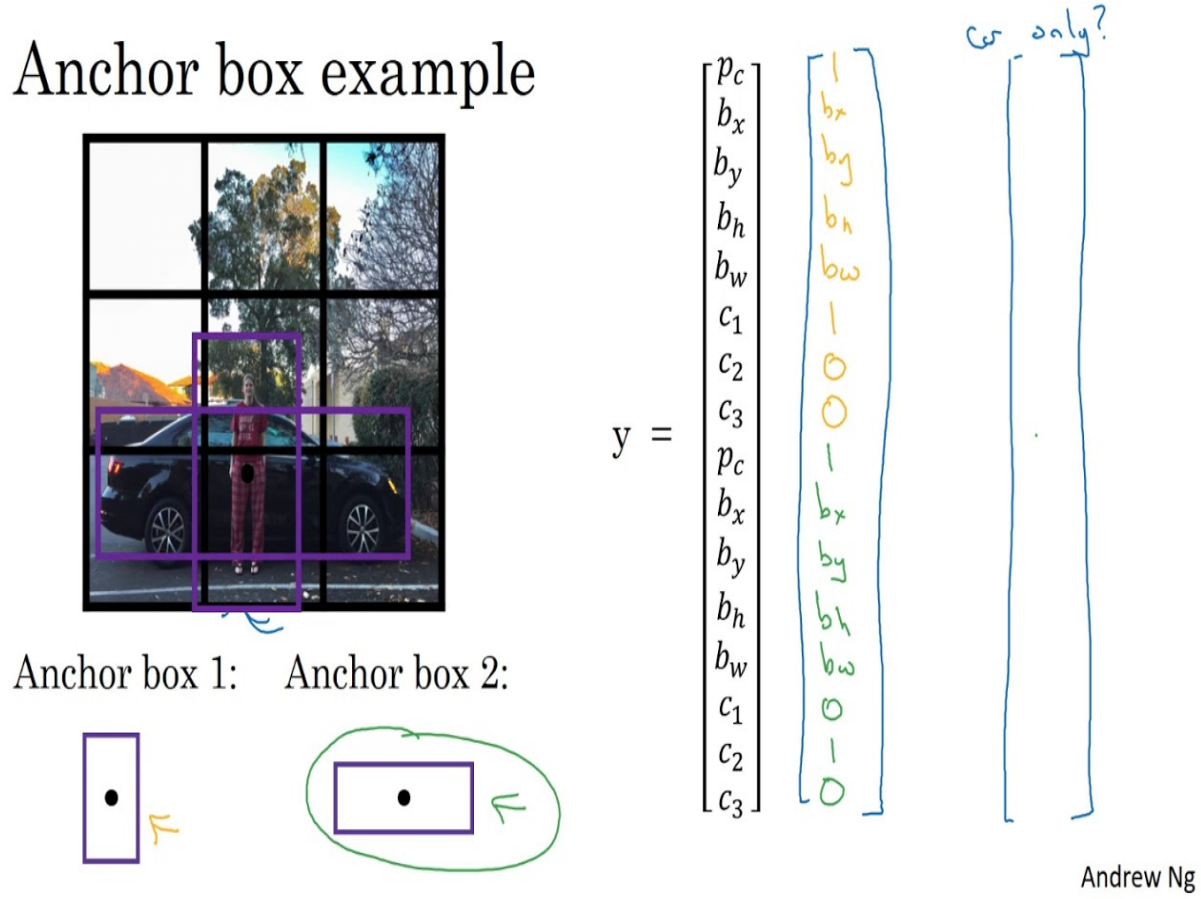
Hình 5: Xác định anchor box cho một vật thể. Từ `Cell i` ta xác định được 3 anchor boxes viền xanh như trong hình. Cả 3 anchor boxes này đều giao nhau với bounding box của vật thể. Tuy nhiên chỉ anchor box có đường viền dày nhất màu xanh được lựa chọn làm anchor box cho vật thể bởi nó có IoU so với ground truth bounding box là cao nhất.

- Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một cell trên feature map mà chứa điểm mid point của vật thể. Chẳng hạn như hình chú chó trong hình 3 sẽ được phân về cho cell màu đỏ vì điểm mid point của ảnh chú chó rơi vào đúng cell này. Từ cell ta sẽ xác định các anchor boxes bao quanh hình ảnh chú chó.

Như vậy khi xác định một vật thể ta sẽ cần xác định 2 thành phần gắn liền với nó là `(cell, anchor box)`. Không chỉ riêng mình `cell` hoặc chỉ mình `anchor box`.

Một số trường hợp 2 vật thể bị trùng mid point, mặc dù rất hiếm khi xảy ra, thuật toán sẽ rất khó xác định được class cho chúng.

Top



Hình 6: Khi 2 vật thể người và xe trùng mid point và cùng thuộc một cell. Thuật toán sẽ cần thêm những lượt tiebreak để quyết định đâu là class cho cell.

6. Hàm loss function

Cũng tương tự như SSD, hàm loss function của YOLO chia thành 2 phần: $\mathcal{L}_{\text{loc}}$ (localization loss) đo lường sai số của bounding box và $\mathcal{L}_{\text{cls}}$ (confidence loss) đo lường sai số của phân phối xác suất các classes.

$$\mathcal{L}_{\text{loc}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

$$\mathcal{L}_{\text{cls}} = \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^{B} \left(\mathbb{1}_{ij}^{\text{obj}} + \lambda_{\text{noobj}}(1 - \mathbb{1}_{ij}^{\text{obj}})\right)(C_{ij} - \hat{C}_{ij})^2}_{\text{cell contain object}} + \underbrace{\sum_{i=0}^{S^2} \sum_{c \in \mathcal{C}} \mathbb{1}_{i}^{\text{obj}} (p_i(c) - \hat{p}_i(c))^2}_{\text{probability distribution classes}}$$

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

- $\mathbb{1}_{i}^{\text{obj}}$: Hàm indicator có giá trị $0, 1$ nhằm xác định xem cell i có chứa vật thể hay không. Bằng 1 nếu chứa vật thể và 0 nếu không chứa.
- $\mathbb{1}_{ij}^{\text{obj}}$: Cho biết bounding box thứ j của cell i có phải là bouding box của vật thể được dự đoán hay không? (xem hình 5).
- C_{ij}: Điểm tin cậy của ô i, `P(contain object) * IoU (predict bbox, ground truth bbox)` .
- $\hat{C}_{ij}$: Điểm tự tin dự đoán.
- $\mathcal{C}$: Tập hợp tất cả các lớp.
- $p_i(c)$: Xác suất có điều kiện, có hay không ô i có chứa một đối tượng của lớp $c \in$
- $\hat{p}_i(c)$: Xác suất có điều kiện dự đoán.

Có thể ban đầu công thức trên khá khó hiểu với người bắt đầu. Chúng ta hãy hiểu đơn giản hóa mục đích của chúng:

- $\mathcal{L}_{\text{loc}}$ là hàm mất mát của bounding box dự báo so với thực tế.
- $\mathcal{L}_{\text{cls}}$ là hàm mất mát của phân phối xác suất. Trong đó tổng đầu tiên là mất mát của dự đoán có vật thể trong cell hay không? Và tổng thứ 2 là mất mát của phân phối xác suất nếu có vật thể trong cell.

Ngoài ra để điều chỉnh phạt loss function trong trường hợp dự đoán sai bounding box ta thông qua hệ số điều chỉnh λ_{coord} và ta muốn giảm nhẹ hàm loss function trong trường hợp cell không chứa vật thể bằng hệ số điều chỉnh λ_{noobj}.

7. Dự báo bounding box

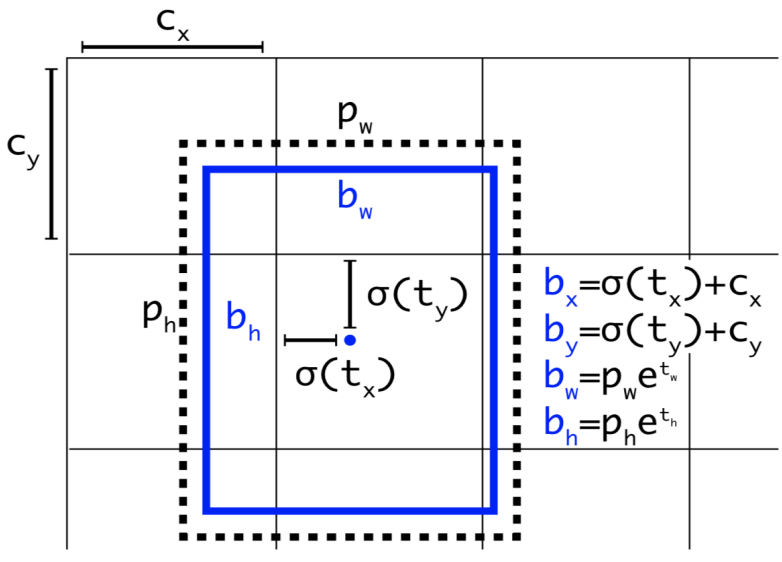
Để dự báo bounding box cho một vật thể chúng ta dựa trên một phép biến đổi từ anchor box và cell.

YOLOv2 vả YOLOv3 dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều. Nếu bounding box dự đoán có thể đặt vào bất kỳ phần nào của hình ảnh, như trong mạng regional proposal network, việc huấn luyện mô hình có thể trở nên không ổn định.

Cho một anchor box có kích thước (p_w, p_h) tại cell nằm trên feature map với góc trên cùng bên trái của nó là (c_x, c_y), mô hình dự đoán 4 tham số (t_x, t_y, t_w, t_h) trong đó 2 tham số đầu là độ lệch (offset) so với góc trên cùng bên trái của cell và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định bounding box dự đoán b có tâm (b_x, b_y) và kích thước (b_w, b_h) thông qua hàm sigmoid và hàm exponential như các công thức bên dưới:

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

Ngoài ra do các tọa độ đã được hiệu chỉnh theo width và height của bức ảnh nên luôn có giá trị nằm trong ngưỡng [0, 1]. Do đó khi áp dụng hàm sigmoid giúp ta giới hạn được tọa độ không vượt quá xa các ngưỡng này.

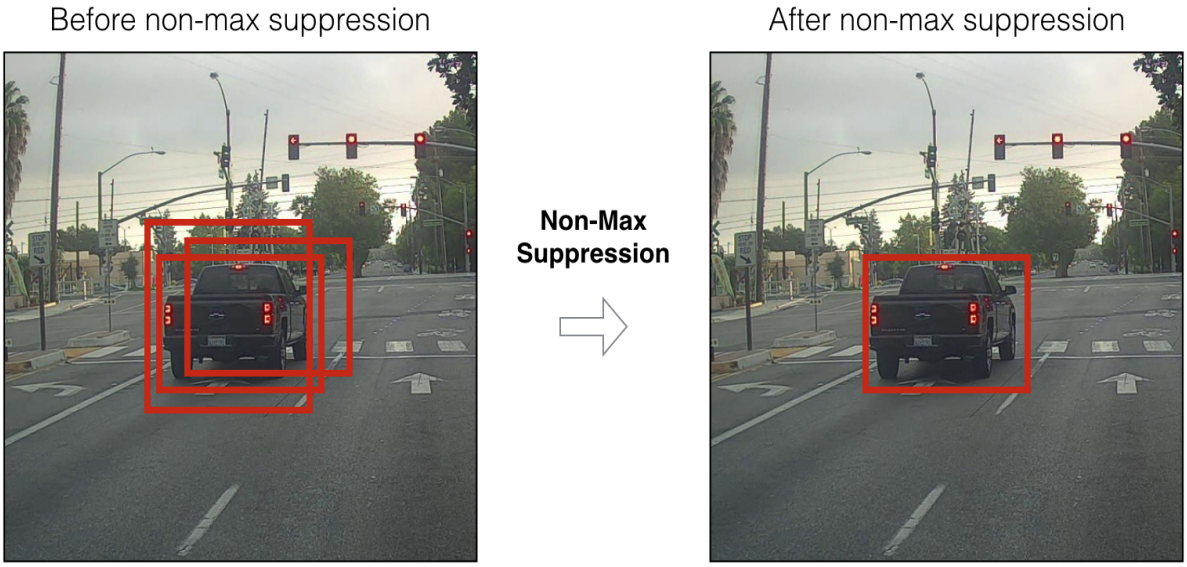


Top

Hình 7: Công thức ước lượng bounding box từ anchor box. Hình chữ nhật nét đứt bên ngoài là anchor box có kích thước là (p_w, p_h). Tọa độ của một bounding box sẽ được xác định dựa trên đồng thời cả anchor box và cell mà nó thuộc về. Điều này giúp kiểm soát vị trí của bounding box dự đoán đâu đó quanh vị trí của cell và bounding box mà không vượt quá xa ra bên ngoài giới hạn này. Do đó quá trình huấn luyện sẽ ổn định hơn rất nhiều so với YOLO version 1.

8. Non-max suppression

Do thuật toán YOLO dự báo ra rất nhiều bounding box trên một bức ảnh nên đối với những cell có vị trí gần nhau, khả năng các khung hình bị overlap là rất cao. Trong trường hợp đó YOLO sẽ cần đến non-max suppression để giảm bớt số lượng các khung hình được sinh ra một cách đáng kể.



Hình 8: non-max suppression. Từ 3 bounding box ban đầu cùng bao quanh chiếc xe đã giảm xuống còn một bounding box cuối cùng.

Các bước của non-max suppression:

- Step 1: Đầu tiên chúng ta sẽ tìm cách giảm bớt số lượng các bounding box bằng cách lọc bỏ toàn bộ những bounding box có xác suất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó, thường là 0.5.
- Step 2: Đối với các bouding box giao nhau, non-max suppression sẽ lựa chọn ra một bounding box có xác xuất chứa vật thể là lớn nhất. Sau đó tính toán chỉ số giao thoa IoU với các bounding box còn lại. Về chỉ số IoU, các bạn xem lại mục Một số định nghĩa - Bài 13 - Model SSD trong Object Detection (https://phamdinhkhanh.github.io/2019/10/05/SSDModelObjectDetection.html)

Nếu chỉ số này lớn hơn ngưỡng threshold thì điều đó chứng tỏ 2 bounding boxes đang overlap nhau rất cao. Ta sẽ xóa các bounding có có xác xuất thấp hơn và giữ lại bouding box có xác xuất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

Code của thuật toán non-max suppression, tôi lại một lần nữa khuyến nghị các bạn tham khảo Mục 2.4.6 - Bài 21 - Tiền xử lý ảnh (https://phamdinhkhanh.github.io/2020/01/06/ImagePreprocessing.html).

9. Tổng kết:

Top

Như vậy qua bài viết này tôi đã lý giải cho các bạn nguyên lý hoạt động mạng YOLO theo một cách khái quát nhất. Đây là một thuật toán rất phức tạp và bên trong nó có rất nhiều các xử lý tính toán mà không đơn giản để chúng ta hiểu hết được toàn bộ chúng. Nếu bạn đọc muốn tìm hiểu thật sâu, tôi khuyến nghị các bạn đọc lại các bài báo gốc được liệt kê ở mục tổng kết. Trong bài viết tôi cố gắng trình bày mọi kiến thức theo cách dễ hiểu nhất. Tôi hi vọng rằng sẽ giúp ích cho bạn trong việc hiểu được mô hình YOLO hoạt động như thế nào.

Phần thực hành xây dựng thuật toán YOLO là một vấn đề khá thú vị. Sẽ có rất nhiều tip và trick cần lưu ý. Do đó tôi dự định sẽ dành nguyên một bài hướng dẫn cho nó ở buổi sau.

10. Tài liệu tham khảo

1. YOLO you only look once real time object detection explained - Manish Chablani (https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006)
2. YOLO object detection YOLO - forum machine learning cơ bản (https://forum.machinelearningcoban.com/t/object-detection-yolo/503)
3. YOLO, YOLOv2 - jonathan hui (https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088)
4. You Only Look Once: Unified, Real-Time Object Detection - Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi (https://arxiv.org/pdf/1506.02640.pdf)
5. YOLO9000: Better, Faster, Stronger - Joseph Redmon, Ali Farhadi (https://arxiv.org/pdf/1612.08242.pdf)
6. C4W3L09 YOLO Algorithm - Andrew Ng - Youtube (https://www.youtube.com/watch?v=9s_FpMpdYW8&t=1s)

Top