

YOLO Evolution

You Only Look Once

Thực tập sinh: Trần Văn Trung.

1. YOLO v1.

Trong bài toán Object Detection, cần nhận dạng được loại đối tượng trên bức ảnh mà còn cần định vị được vị trí của đối tượng.

YOLO là một mô hình CNN để detect đối tượng, một trong những phương pháp tốt nhất và nhanh nhất (real-time) hiện nay.

Ý tưởng bài toán Object Detection:

- Chia ảnh thành nhiều box, với mỗi box sẽ detect object trong box đó. Vị trí của object chính là tọa độ của box đó.
- Thay vì chia ảnh thành nhiều box, có thể sử dụng một thuật toán để lựa chọn các region ứng viên (thuật toán Selective Search), các vùng ứng viên này được hiểu như các vùng liên thông với nhau trên kênh màu RGB, sau đó dùng model để phân loại object trên từng vùng ứng viên này. Các thuật toán tiêu biểu cho ý tưởng này: R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN.

Chi tiết bài toán với YOLO v1:

Grid System

- Ảnh được chia thành ma trận ô vuông 7×7 (Grid System), mỗi ô vuông bao gồm một tập các thông tin mà mô hình phải dự đoán.

- Đối tượng duy nhất mà ô vuông đó chứa. Tâm của đối tượng nằm trong ô vuông nào thì ô vuông đó chứa đối tượng, dù cho phần ảnh thực tế của đối tượng nằm trên những ô vuông khác, nhưng tâm của đối tượng ko nằm trong ô vuông đó thì cũng ko được tính là ô vuông chứa đối tượng. Nếu có nhiều tâm nằm trong một ô vuông thì vẫn chỉ có một đối tượng được gán cho mỗi ô vuông. Đây là một nhược điểm của mô hình, không thể detect nhiều object trong cùng một ô vuông. Giải pháp là chia Grid System thành nhiều ô vuông hơn để có thể detect được nhiều ô vuông hơn. Kích thước của Grid System phải là ước số của kích thước ảnh đầu vào.

- Mỗi ô vuông chịu trách nhiệm dự đoán 2 boundary box của đối tượng. Mỗi boundary box dự đoán có chứa object hay không và thông tin vị trí của boundary box gồm trung tâm boundary box của đối tượng và chiều dài, rộng của boundary box đó. Một điều cần lưu ý, lúc cài đặt chúng ta không dự đoán giá trị pixel mà cần phải chuẩn hóa kích thước ảnh về đoạn từ $[0-1]$ và dự đoán độ lệch của tâm đối tượng đến box chứa đối tượng đó. Ví dụ, chúng ta thay vì dự đoán vị trí pixel của điểm màu đỏ, thì cần dự đoán độ lệch a, b trong ô vuông chứa tâm object.

- Tổng hợp lại, với mỗi ô vuông chúng ta cần dự đoán những thông tin sau:

- Ô vuông có chứa đối tượng nào.
- Dự đoán độ lệch 2 box chứa object so với ô vuông hiện tại.
- Lớp của đối tượng đó.
- Với mỗi ô vuông cần dự đoán một vecto có $(5 * n_{box} + n_{class})$

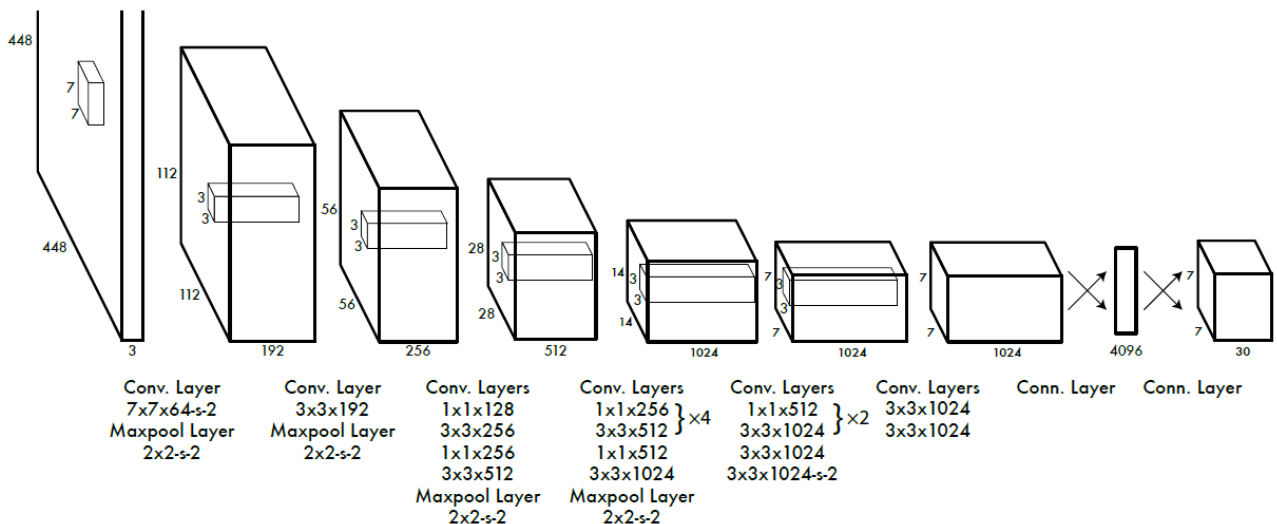
- Ví dụ: Dự đoán 2 box, 3 lớp, với mỗi ô vuông chúng ta có một ma trận 3 chiều $7 \times 7 \times 13$ chứa các thông tin cần thiết:

object 1?	object 2?	offset x1	offset y1	width 1	height 1	offset x2	offset y2	width 2	height 2	0	0	1
-----------	-----------	-----------	-----------	---------	----------	-----------	-----------	---------	----------	---	---	---

Hình 1.1: Ví dụ một vecto 13×1 cho Grid 7×7 , 3 class.

CNN cho YOLO Object Detection

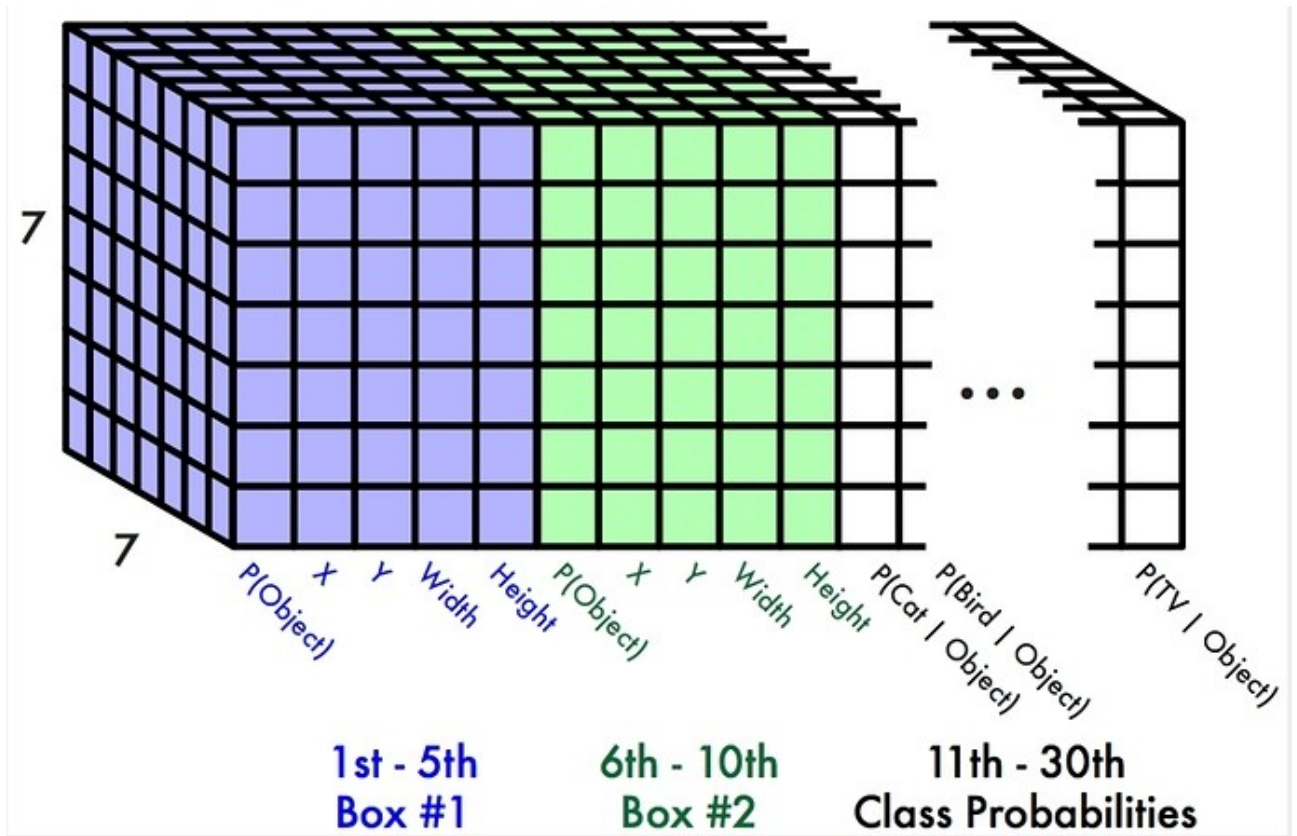
- Chúng ta đã biết cần dự đoán thông tin nào với mỗi ô vuông, mô hình CNN phải cho ra output shape phù hợp với yêu cầu dự đoán, tức $Grid_Size \times Grid_Size \times (5 * n_{box} + n_{class})$. Như ví dụ trước với 2 box, 3 class là $7 \times 7 \times (5 * 2 + 3)$.



Hình 1.2: Kiến trúc YOLO v1.

- YOLO sử dụng linear regression để dự đoán thông tin ở mỗi ô vuông. Ở layer cuối cùng sẽ không sử dụng hàm kích hoạt nào cả. Thay vì sử dụng FC ở cuối, có thể thay bằng các Conv 1×1 với $(n_{box} * 5 + n_{class})$ feature maps.

- Ví dụ: Với một bài toán nhận dạng đối tượng cho 20 lớp đối tượng, Grid chia thành 7×7 , số bounding box dự đoán cho mỗi ô là 2, số lượng đối tượng là 20. Số thông tin cần dự đoán cho 1 ô vuông là $5 * 2 + 20 = 30$. Với toàn bộ 7 ô vuông là $7 \times 7 \times 30$. Hình ảnh minh họa:



Hình 1.3: Ví dụ đầu ra của mạng CNN với Grid Size = 7, nbox = 2, nclass = 20.

Loss Function

- YOLO v1 sử dụng hàm độ lỗi bình phương giữa dự đoán và nhãn để tính độ lỗi cho mô hình. Độ lỗi tổng của mô hình sẽ dựa trên 3 độ lỗi con:

- Độ lỗi của việc dự đoán nhãn của object - Classification Loss.
- Độ lỗi của dự đoán tọa độ và dự đoán chiều dài, chiều rộng bounding box - Localization Loss.
- Độ lỗi của ô vuông có chứa object nào hay không - Confidence Loss.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \quad (1)$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \quad (2)$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \quad (3)$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \quad (4)$$

$$+ \sum_{i=0}^{S^2} \mathbb{I}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (5)$$

Hình 1.4: Công thức hàm mất mát.

- Classification Loss:

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

where

$\mathbb{1}_i^{\text{obj}} = 1$ if an object appears in cell i , otherwise 0.

$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i .

Hình 1.5: Hàm mất mát theo dự đoán nhãn của đối tượng.

- Localization Loss:

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

where

$\mathbb{1}_{ij}^{\text{obj}} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

Hình 1.6: Hàm mất mát theo tọa độ và kích thước của bounding box.

- Confidence Loss:

$$\sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2$$

where

\hat{C}_i is the box confidence score of the box j in cell i .

$\mathbb{1}_{ij}^{\text{obj}} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

Hình 1.7: Hàm mất mát khi ô vuông có chứa đối tượng.

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

where

$\mathbb{1}_{ij}^{noobj}$ is the complement of $\mathbb{1}_{ij}^{obj}$.


\hat{C}_i is the box confidence score of the box j in cell i .

λ_{noobj} weights down the loss when detecting background.

Hình 1.8: Hàm mất mát khi ô vuông không chứa đối tượng.

Dự đoán lớp và tọa độ bounding box sau quá trình huấn luyện.

- Chỉ giữ lại những bounding box có chứa object nào đó. Tính tích xác suất có điều kiện của ô vuông thuộc về phân lớp i với xác suất ô vuông chứa object, chỉ giữ lại những bounding box có giá trị này lớn hơn ngưỡng nhất định (threshold).
- Mỗi object lại có thể có nhiều boundary box khác nhau do mô hình dự đoán. Để tìm boundary box tốt nhất các object, chúng ta có thể dùng thuật toán non-maximal suppression để loại những boundary box giao nhau nhiều, tức là có IOU giữa 2 boundary box lớn.
- Để tính IOU giữa 2 box chúng ta cần tính diện tích giao nhau giữa 2 box chia cho tổng diện tích của 2 box đó.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Hình 1.9: Công thức tính IoU.



Hình 1.10: Phân cấp các mức độ IoU.

2. YOLO v2.

- YOLO v2 cho phép một mô hình có thể huấn luyện trên cả nhận dạng và phân lớp đối tượng. điểm chính trong mô hình là cải thiện Recall và Localization trong khi duy trì accuracy của classification.

- Kiến trúc mô hình:

- Có 23 ConV thay vì 24 so với bản YOLO v1.

- Activation function: Leaky ReLU (tất cả các lớp trừ lớp cuối cùng), Linear Layer (lớp cuối cùng).

- Những thay đổi chính:

- **Batch Normalization:** bằng cách thêm batch normalization vào toàn bộ các ConV của YOLO v1, kết quả cải thiện 2% đối với mAP. Bằng cách này cũng có thể giảm overfitting đối với model.

- **High Resolution Classifier:** YOLO v1 huấn luyện mạng trên ảnh 224x224, thực hiện tăng độ phân giải lên 448x448 để detect. YOLO v2 thực hiện huấn luyện mạng phân lớp trên ảnh 448x448 và sau đó tinh chỉnh mạng để thực hiện detect. Điều này làm kết quả tăng 4% của mAP.

- **Anchor box:** Trong YOLO v1, BBox được dự đoán bằng lớp FC, trong YOLO v2, lớp FC này đã bị loại bỏ và các Anchor Box được sử dụng để dự đoán BBox. Và các Anchor Box cho một tập dữ liệu đã cho được chọn bằng cách sử dụng phân cụm k-means.

- Mô hình được co lại để xử lý ảnh đầu vào 416x416, thay vì 448x448. Điều này giúp tạo một ô grid ở chính giữa ảnh, bởi vì các đối tượng lớn thường có xu hướng ở trung tâm ảnh. Sử dụng Anchor Box làm giảm mAP (0.3 mAP so với YOLO v1) nhưng làm tăng Recall (7% so với YOLO v1). Vấn đề của việc sử dụng Anchor Box, là các box được xác định bằng tay và mô hình khi sử dụng Anchor Box thiếu tính ổn định.

- **Dimension Clusters:** Giải quyết vấn đề thứ nhất khi sử dụng Anchor Box, thay vì các box được xác định bằng tay, một thuật toán phân cụm k-means được chạy trên toàn bộ tập huấn luyện để xác định ra box tốt nhất.

- **Finer Features:** YOLO v2 sử dụng Grid System 13x13. Thêm một lớp passthrough mang lại các tính năng từ lớp trước đó ở độ phân giải 26×26 . Nó kết hợp các tính năng độ phân giải cao với các tính năng độ phân giải thấp bằng cách xếp chồng vào các kênh khác nhau thay vì các vị trí không gian. Điều này thay đổi bản đồ tính năng $26 \times 26 \times 512$ thành $13 \times 13 \times 2048$. YOLO v2 tránh được nhược điểm của YOLO v1 do sử dụng Grid System 13x13, có thể detect được những đối tượng nhỏ hơn những vẫn hiệu quả với những đối tượng lớn.

- **Multi-scale Training:** YOLO v1 có điểm yếu là nếu các đối tượng đầu vào có kích thước khác nhau thì kết quả sẽ có vấn đề. Nếu YOLO v1 được huấn luyện trên các ảnh nhỏ thì sẽ khó phát hiện được cùng các đối tượng đó trên ảnh lớn. Với YOLO v2, thay vì cố định kích thước hình ảnh đầu vào, mô hình sẽ thay đổi mạng mỗi vài lần lặp. Ở mỗi epoch 10, mạng chọn ngẫu nhiên các kích thước hình ảnh mới. Các hình ảnh ngẫu nhiên với các kích thước khác nhau trong khoảng từ $320 * 320$ đến $608 * 608$ [5]. Điều này cho phép mạng tìm hiểu và dự đoán các đối tượng từ các kích thước đầu vào khác nhau với độ chính xác.

- Các cải tiến của YOLO v2 so với v1, cho phép mô hình nhanh hơn, mạnh hơn và tốt hơn. Có thể detect được các đối tượng với kích thước ảnh khác nhau. Có thể detect được các đối tượng nhỏ trong ảnh.

3. YOLO v3.

- Em sẽ cập nhật tiếp ạ