

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH

~~~~~\*~~~~~



**BÁO CÁO ĐỒ ÁN CUỐI KỲ  
PHƯƠNG PHÁP TOÁN TRONG PHÂN TÍCH DỮ  
LIỆU THỊ GIÁC**

**ĐỀ TÀI:  
YOU ONLY LOOK ONCE**

Giáo viên hướng dẫn: PGS. TS Lý Quốc Ngọc  
Sinh viên: Chí Cẩm Hào - 18120353

TP. Hồ Chí Minh - 2021

|                                     |           |
|-------------------------------------|-----------|
| <b>I. Giới thiệu:</b>               | <b>3</b>  |
| <b>II. YOLO phiên bản đầu tiên:</b> | <b>3</b>  |
| 1. Động lực nghiên cứu:             | 3         |
| 2. Phát biểu bài toán:              | 3         |
| 3. Phương pháp:                     | 5         |
| 3.1. Nguyên lý:                     | 5         |
| 3.2. Intersection over Union (IoU): | 7         |
| 3.3. Class prediction:              | 7         |
| 3.4. Non-maximal suppression:       | 7         |
| 4. Loss function:                   | 8         |
| 4.1. Classification loss:           | 8         |
| 4.2. Localization loss:             | 8         |
| 4.3. Confidence loss:               | 9         |
| 4.4. Loss function:                 | 10        |
| 5. Đánh giá:                        | 11        |
| <b>III. Họ gia đình YOLO:</b>       | <b>12</b> |
| 1. YOLO v2:                         | 13        |
| 2. YOLO v3:                         | 15        |
| 3. YOLO v4:                         | 17        |
| 4. Tổng kết:                        | 20        |
| 5. Tương lai của YOLO:              | 20        |
| <b>IV. Tham khảo:</b>               | <b>22</b> |

## I. Giới thiệu:

Phương pháp được sử dụng trong phát hiện đối tượng phổ biến nhất hiện nay có thể kể đến các phương pháp Convolutional Neural Network (CNN) và YOLO. YOLO có độ chính xác thấp hơn so với họ phương pháp CNN, bù lại YOLO được ông Joseph Redmon giới thiệu là một hệ thống phát hiện vật thể thời gian thực đỉnh cao [1].

Ở bài báo cáo này, phần II chỉ giới hạn trong YOLO phiên bản đầu tiên, phần III sẽ nêu ra các cải tiến, thay đổi của họ gia đình YOLO sau thời gian hơn 5 năm qua.

## II. YOLO phiên bản đầu tiên:

### 1. Động lực nghiên cứu:

Khi họ phương pháp CNN sử dụng Region Proposal Networks (RPNs) để tạo ra các ứng cử viên để phân vùng đối tượng bằng các thuật toán như Selective Search hoặc mạng Region Proposal đạt được hiệu quả về mặt chính xác nhưng có độ phức tạp về thời gian chưa đủ để áp dụng vào các bài toán thực tế do phải duyệt ảnh lại ảnh rất nhiều lần. Trong khi đó, YOLO là nỗ lực đầu tiên trong việc xây dựng hệ thống phát hiện vật thể *real-time*. YOLO là từ viết của “You Only Look Once”, bước đột xuất ứng viên chỉ cần duyệt ảnh đầu vào một lần duy nhất.

Nếu như các phương pháp sử dụng kĩ thuật đề xuất khu vực khả dĩ trong quá trình đề xuất ứng viên thì YOLO tiếp cận với toàn bộ hình ảnh và áp dụng phương pháp “Chia để trị”, tức chia ảnh thành các grid để phát hiện đối tượng.

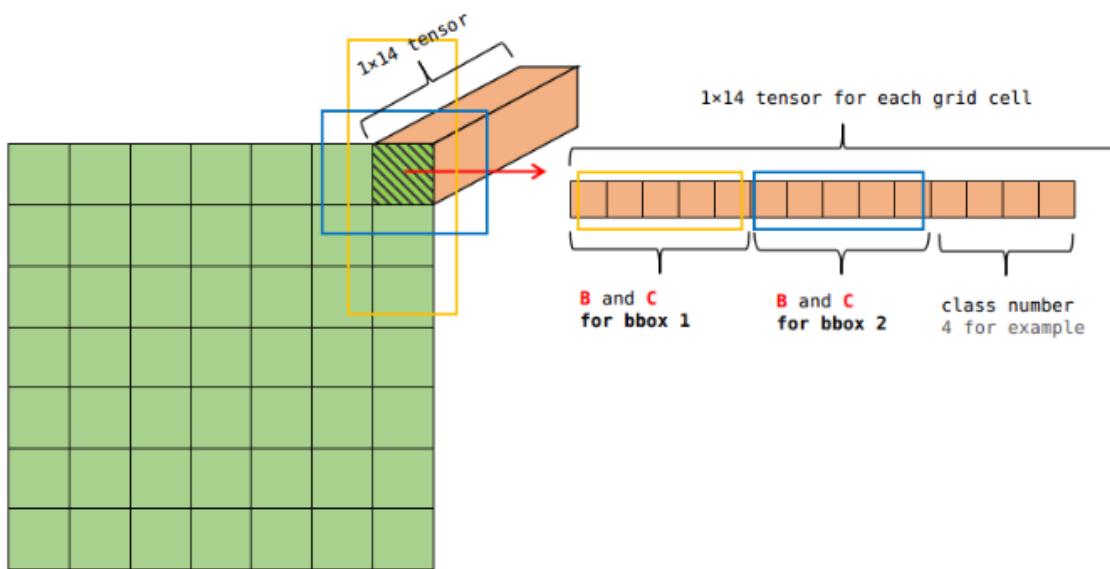
### 2. Phát biểu bài toán:

- a. Đầu vào bài toán: ảnh digital
- b. Đầu ra bài toán:  $S \times S \times (B * 5 + C)$  tensor.

Bức ảnh sẽ được chia thành  $S \times S$  grid. Mỗi grid nhận nhiệm vụ phát hiện  $B$  bounding box với  $C$  là số lượng class cần phân lớp.

Tensor là đại diện output của một grid. Trong đó, hằng số 5 đại diện 5 thông số:

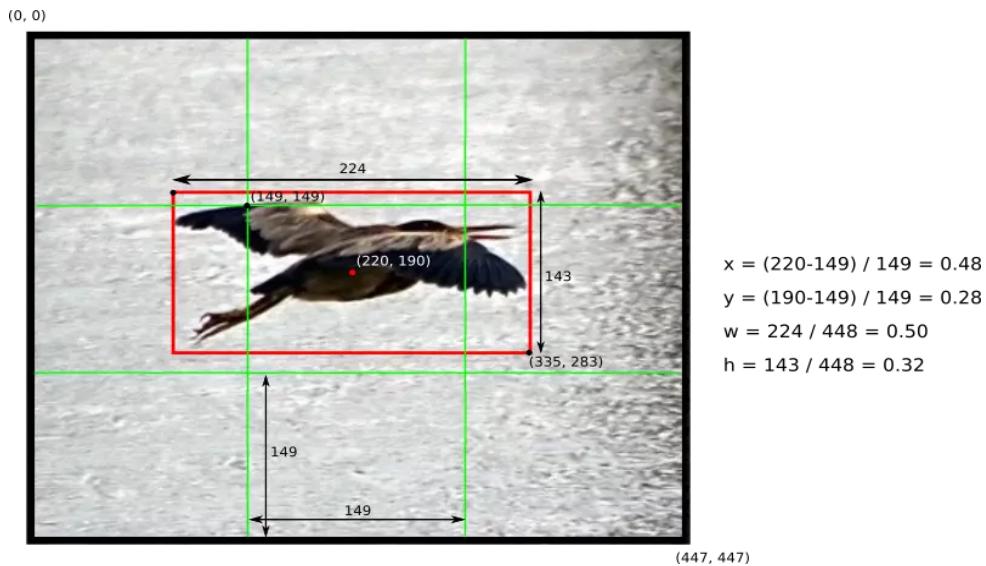
- Bounding box: x, y, w, h
- Confidence score



Hình 2: Ví dụ về một tensor với  $B = 2$  và  $C = 4$  [4]

Bounding box tượng trưng cho kích cỡ và vị trí của đối tượng trong ảnh, trong khi confidence score là thông số đại diện cho xác suất đối tượng thuộc lớp tương ứng.

Các đại lượng được chuẩn hóa trước khi lưu. Tọa độ (x, h) của bounding box được chuẩn hóa so với chiều dài và chiều cao của grid trong khi kích cỡ (w, h) được chuẩn hóa so với toàn bộ bức ảnh. Xét một ví dụ như sau:

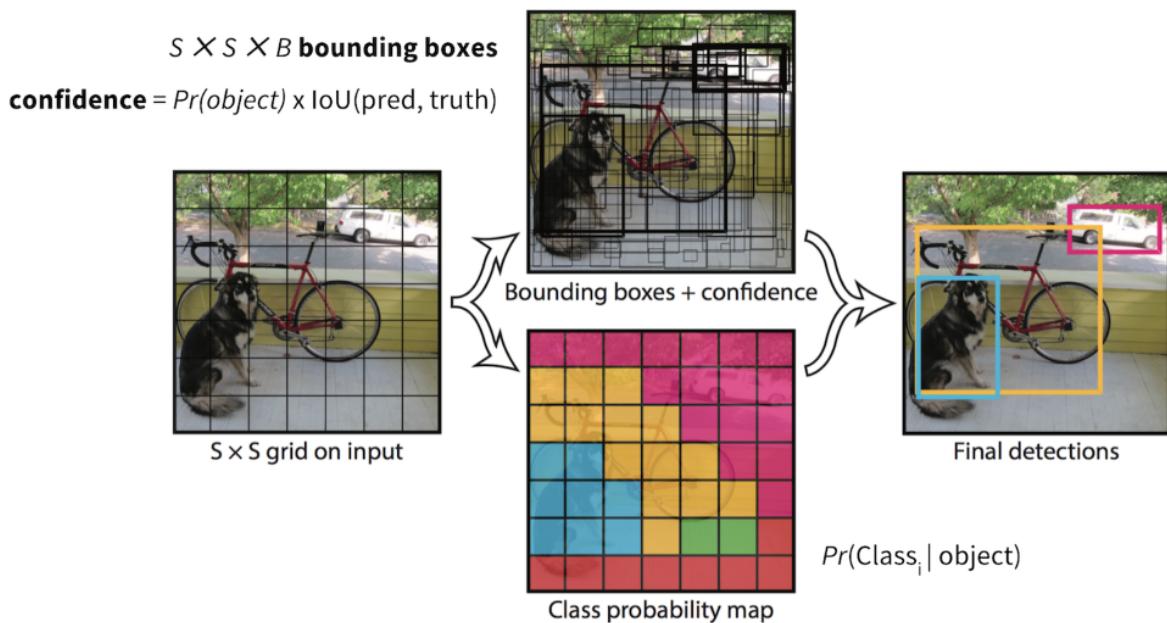


Hình 3: Ví dụ về bounding box với ảnh 448x448,  $S = 3$  [3]

### 3. Phương pháp:

#### 3.1. Nguyên lý:

YOLO chia ảnh thành các SxS grid (Hình 3). Mỗi grid cell chỉ dự đoán một số lượng cố định B các bounding box. Đầu ra của mỗi grid bao gồm 5 thông số x, y, w, h và confidence score.



Hình 3: Mô hình Yolo [2]

Mục đích của việc chia grid để đảm bảo các vật thể đối tượng trên ảnh không may bị bỏ sót thay vì chỉ tập chung vào một vùng của bức ảnh như các phương pháp RCNN thông thường.

Số lượng B tăng đồng nghĩa việc có thể phát hiện nhiều đối tượng khác nhau trong một grid.

Mỗi phiên bản YOLO có các thông số khác nhau, nhưng đều có thuật toán tương đồng. Do là mạng phức hợp (united). Thuật toán phát hiện đối tượng và phân lớp của YOLO có thể được gộp chung và thể hiện qua 4 bước như sau:

1. Chia bức ảnh thành  $S \times S$  grid.
2. Mỗi grid tính các giá trị IoU (3.2). Tính confidence score để lọc ra

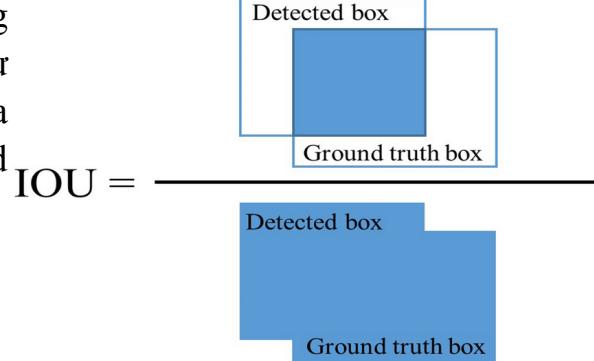
*những bounding box không chứa object.*

3. *Tính xác suất có điều kiện với từng class (3.3).*
4. *Non-maximal suppression để loại bỏ các bounding box trùng lắp (3.4).*

IoU là thông số đánh giá độ chính xác của bounding box được dự đoán so với thực tế, đây là đầu vào để khai thác các giá trị thông số khác như confidence score, xác suất có điều kiện.

### 3.2. Intersection over Union (IoU):

IoU thể hiện bằng thương của vùng diện tích của bounding box sau khi dự đoán (Detected box) và vùng diện tích của bounding box được tô thủ công(Ground truth box).



Hình 4: IoU [4]

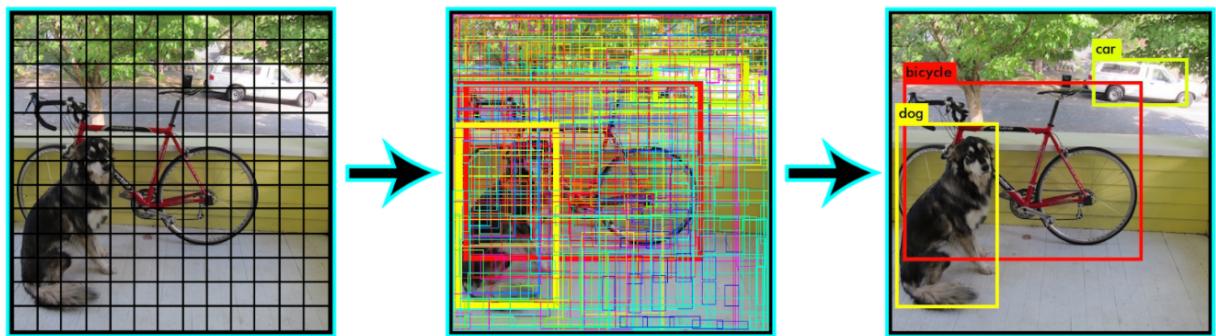
### 3.3. Class prediction:

Sử dụng xác suất có điều kiện và hai tham số là confidence score và IoU đã được tính từ trước. Class prediction được xác định dựa vào phương trình sau:

$$\Pr(\text{Class}_i | \text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

### 3.4. Non-maximal suppression:

Ở giai đoạn hậu xử lý, việc chia ảnh thành các grid dẫn đến có nhiều bounding box cho một đối tượng khi tỉ lệ kích cỡ của đối tượng lớn hơn nhiều so với kích cỡ của grid. YOLO giải quyết vấn đề này bằng thuật toán Non-maximum suppression.



Hình 4: Sử dụng Non-maximal suppression [2]

Với đầu vào là confidence score và IoU của từng bounding box, thuật toán của hậu xử lý sau quá trình training như sau:

1. Quét qua tất cả các bounding box đầu ra từ mạng và chỉ giữ lại những bounding box có confidence score cao (trên ngưỡng  $confThreshold$ ).
2. Gán nhãn bounding box có confidence score cao nhất.
3. So sánh overlap (Intersection over Union) của box này với các box khác.
4. Loại bỏ các bounding box có overlap bằng Non-maximal-suppression (thông thường xét với  $Intersection over Union > 50\%$ )
5. Xét tiếp box có confidence score cao nhất tiếp theo.
6. Lặp lại bước 2-5 cho đến khi tất cả các box được duyệt.

## 4. Loss function:

### 4.1. Classification loss:

Nếu vật thể được phát hiện, thì sai số phân lớp được tính bằng tổng bình phương lỗi của các lớp xác suất có điều kiện:

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

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  $i$ .

Giải thích công thức: Duyệt tất cả các cell có vật thể xuất hiện, tính tổng bình phương sai số của xác suất có điều kiện.

#### 4.2. Localization loss:

Sai số giữa bounding box được dự đoán và ground truth:

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

where

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

$\lambda_{\text{coord}}$  increase the weight for the loss in the boundary box coordinates.

Giải thích công thức: Duyệt bounding box trong các grid chịu trách nhiệm phát hiện vật thể, tính tổng sai số bình phương của bốn tham số ( $x, y, w, h$ ). Lý do biểu thức chứa  $w$  và  $h$  có căn bậc hai vì một phần muốn giải quyết vấn đề là chúng ta không muốn trọng lượng sai số tuyệt đối của box có kích thước lớn bằng box có kích thước nhỏ. Ta muốn với cùng một lượng error of pixels khi dự đoán, box kích thước lớn sẽ chịu error ít hơn box kích thước nhỏ. Hơn nữa, để nhấn mạnh vào độ chính xác của bounding box, chúng ta nhân hàm mất mát này với thông số lamda. Ở phiên bản đầu tiên, mặc định tham số lamda này là 5.

### 4.3. Confidence loss:

Tính toán khả năng bounding box có chứa đối tượng. Nếu đối tượng vật thể được phát hiện trong box, confidence loss được tính như sau:

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

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.

Giải thích công thức: Duyệt bounding box trong các grid mà bounding box đó chịu trách nhiệm cho phát hiện vật thể, tính sai số bình phương của tham số confidence score.

Nếu đối tượng vật thể không được phát hiện trong box, confidence loss được tính như sau:

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

where

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

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

$\lambda_{\text{noobj}}$  weights down the loss when detecting background.

Giải thích công thức: Chỉ khác một chút là ta chỉ duyệt các bounding box không tham gia biểu thức ở trên và thêm trọng số lamda để giảm trọng lượng của biểu thức. Điều này nhằm mục đích tránh việc chúng ta phát hiện background

nhiều hơn là đối tượng vật thể vì đa số các bounding box ở đây không chứa vật thể. Ở YOLO phiên bản đầu tiên, mặc định tham số lamda này là 0.5.

#### 4.4. Loss function:

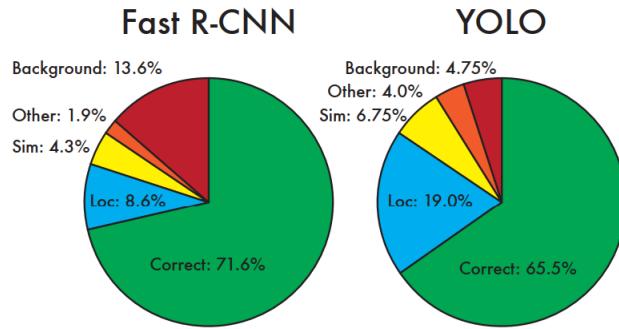
Được thể hiện bằng tổng các hàm mất mát trên. Để tối ưu hóa YOLO, ta phải tính hàm mất mát và tính toán sao cho hàm mất mát là nhỏ nhất.:

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

#### 5. Đánh giá:

Tuy mạng phức hợp YOLO có thể phát hiện và phân lớp đối tượng trong thời gian thực nhưng đánh đổi ta cũng có các hạn chế sau:

1. Đánh đổi giữa inference time và độ chính xác nên độ chính xác thấp.
2. Khó phát hiện các vật thể nhỏ xuất hiện theo nhóm, như là một đàn chim chǎng hạn.
3. Vấn đề Localization errors lớn. Ta xem biểu đồ dưới với Localization Errors 19%, việc xác định bounding box bao vật thể với độ chính xác thấp.

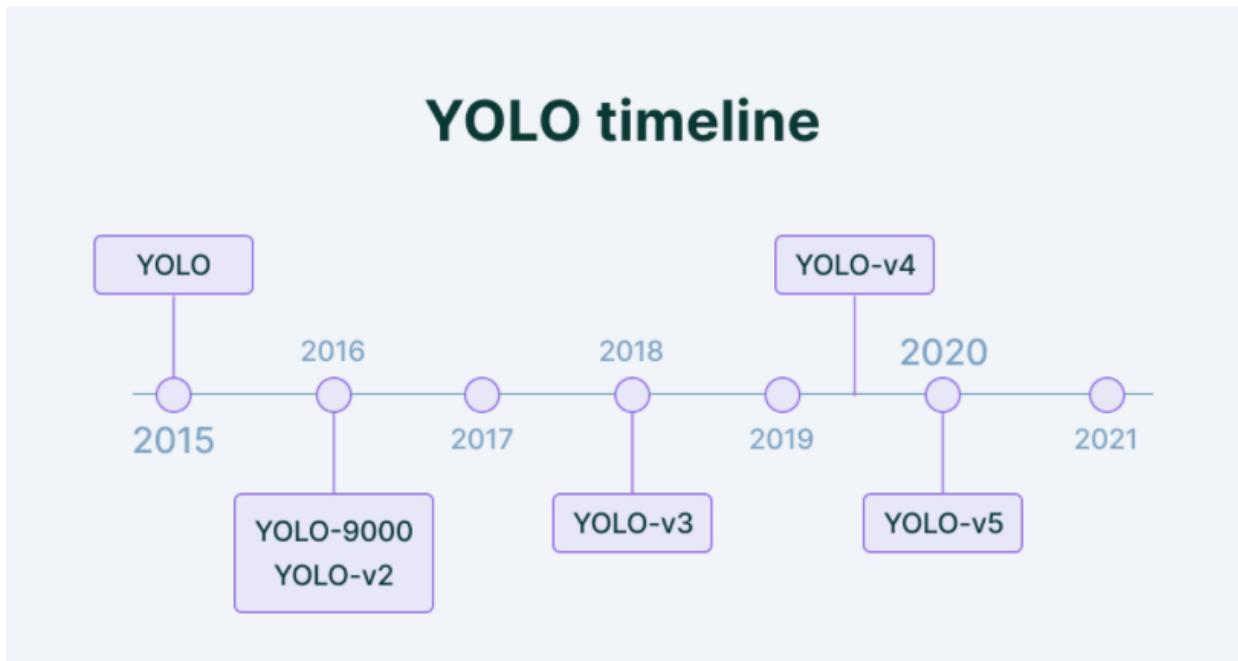


**Figure 4: Error Analysis: Fast R-CNN vs. YOLO** These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

*Hình 5: Localization errors của YOLO khá lớn so với Fast R-CNN[1].*

4. Chỉ có thể dự đoán 49 vật thể do 49 grid, mỗi grid chỉ có thể dự đoán 1 vật thể.

### III. Họ gia đình YOLO:



## 1. YOLO v2:

### 1.1. Batch normalization (tăng 2% mAP):

Cách thức hoạt động: normalize output của activation của lớp trước trong mạng Neural Network bằng cách trừ đi giá trị trung bình của batch và sau đó chia cho độ lệch chuẩn của batch.

Điều này làm tăng tốc thời gian huấn luyện và giảm overfitting và covariance shift.

Để giải thích về covariance shift, ví dụ khi ta huấn luyện trên tập ảnh những con mèo màu đen rồi dự đoán trên tập ảnh mèo có màu thì mô hình sẽ không cho kết quả tốt dẫn đến covariance shift cao.

### 1.2. High Resolution Classifier (tăng 4% mAP):

Tăng resolution tại bước phân lớp từ 224x244 lên 448x448.

Huấn luyện cho bước phân lớp tốt hơn nếu trong ảnh có các chi tiết nhỏ do độ phân giải ảnh, điều này cũng tồn tại hạn chế sẽ giảm tốc độ phát hiện đối tượng của YOLO tiềm nhiệm.

### 1.3. Anchor Box và Dimension Cluster (tăng 7% recall):

Sử dụng Anchor Box – tức k box phổ biến nhất trong training set của bounding box với các hình dạng khác nhau. Sử dụng phân cụm K-means và IoU để phân loại k hình dạng khác nhau. Chọn k sao cho cân bằng trade-off giữa độ phức tạp và độ recall cao.

Việc này sẽ giúp mô hình chỉ xét những box phổ biến, đẩy tốc độ dự đoán nhanh hơn.

### 1.4. Fine-Grained Features:

| Yolo v1                        | Yolo v2                          |
|--------------------------------|----------------------------------|
| Chia ảnh thành 7x7 grid cells. | Chia ảnh thành 13x13 grid cells. |

Phát hiện và localize những vật thể nhỏ và tăng số lượng vật thể có thể được phát hiện từ 49 lên 169 mỗi ảnh.

### 1.5. Darknet 19:

Thay đổi trong cấu trúc mạng gồm 19 lớp convolution và 5 lớp max-pooling và 1 lớp softmax cho phân lớp:

| Type          | Filters | Size/Stride    | Output           |
|---------------|---------|----------------|------------------|
| Convolutional | 32      | $3 \times 3$   | $224 \times 224$ |
| Maxpool       |         | $2 \times 2/2$ | $112 \times 112$ |
| Convolutional | 64      | $3 \times 3$   | $112 \times 112$ |
| Maxpool       |         | $2 \times 2/2$ | $56 \times 56$   |
| Convolutional | 128     | $3 \times 3$   | $56 \times 56$   |
| Convolutional | 64      | $1 \times 1$   | $56 \times 56$   |
| Convolutional | 128     | $3 \times 3$   | $56 \times 56$   |
| Maxpool       |         | $2 \times 2/2$ | $28 \times 28$   |
| Convolutional | 256     | $3 \times 3$   | $28 \times 28$   |
| Convolutional | 128     | $1 \times 1$   | $28 \times 28$   |
| Convolutional | 256     | $3 \times 3$   | $28 \times 28$   |
| Maxpool       |         | $2 \times 2/2$ | $14 \times 14$   |
| Convolutional | 512     | $3 \times 3$   | $14 \times 14$   |
| Convolutional | 256     | $1 \times 1$   | $14 \times 14$   |
| Convolutional | 512     | $3 \times 3$   | $14 \times 14$   |
| Convolutional | 256     | $1 \times 1$   | $14 \times 14$   |
| Convolutional | 512     | $3 \times 3$   | $14 \times 14$   |
| Maxpool       |         | $2 \times 2/2$ | $7 \times 7$     |
| Convolutional | 1024    | $3 \times 3$   | $7 \times 7$     |
| Convolutional | 512     | $1 \times 1$   | $7 \times 7$     |
| Convolutional | 1024    | $3 \times 3$   | $7 \times 7$     |
| Convolutional | 512     | $1 \times 1$   | $7 \times 7$     |
| Convolutional | 1024    | $3 \times 3$   | $7 \times 7$     |
| Convolutional | 1000    | $1 \times 1$   | $7 \times 7$     |
| Avgpool       |         | Global         | 1000             |
| Softmax       |         |                |                  |

**Table 6: Darknet-19.**

*Hình 7: Darknet-19 [5]*

## 1.6. Tổng kết:

|                      | YOLO |      |      |      |      |      |      |      |             | YOLOv2 |
|----------------------|------|------|------|------|------|------|------|------|-------------|--------|
| batch norm?          |      | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓           | ✓      |
| hi-res classifier?   |      | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓           | ✓      |
| convolutional?       |      | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓           | ✓      |
| anchor boxes?        |      | ✓    | ✓    |      |      |      |      |      |             |        |
| new network?         |      |      | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓           | ✓      |
| dimension priors?    |      |      |      | ✓    | ✓    | ✓    | ✓    | ✓    | ✓           | ✓      |
| location prediction? |      |      |      |      | ✓    | ✓    | ✓    | ✓    | ✓           | ✓      |
| passthrough?         |      |      |      |      |      | ✓    | ✓    | ✓    | ✓           | ✓      |
| multi-scale?         |      |      |      |      |      |      | ✓    | ✓    | ✓           |        |
| hi-res detector?     |      |      |      |      |      |      |      | ✓    |             | ✓      |
| VOC2007 mAP          | 63.4 | 65.8 | 69.5 | 69.2 | 69.6 | 74.4 | 75.4 | 76.8 | <b>78.6</b> |        |

**Table 2: The path from YOLO to YOLOv2.** Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

*Hình 8: Hiệu năng YOLO v2 [5]*

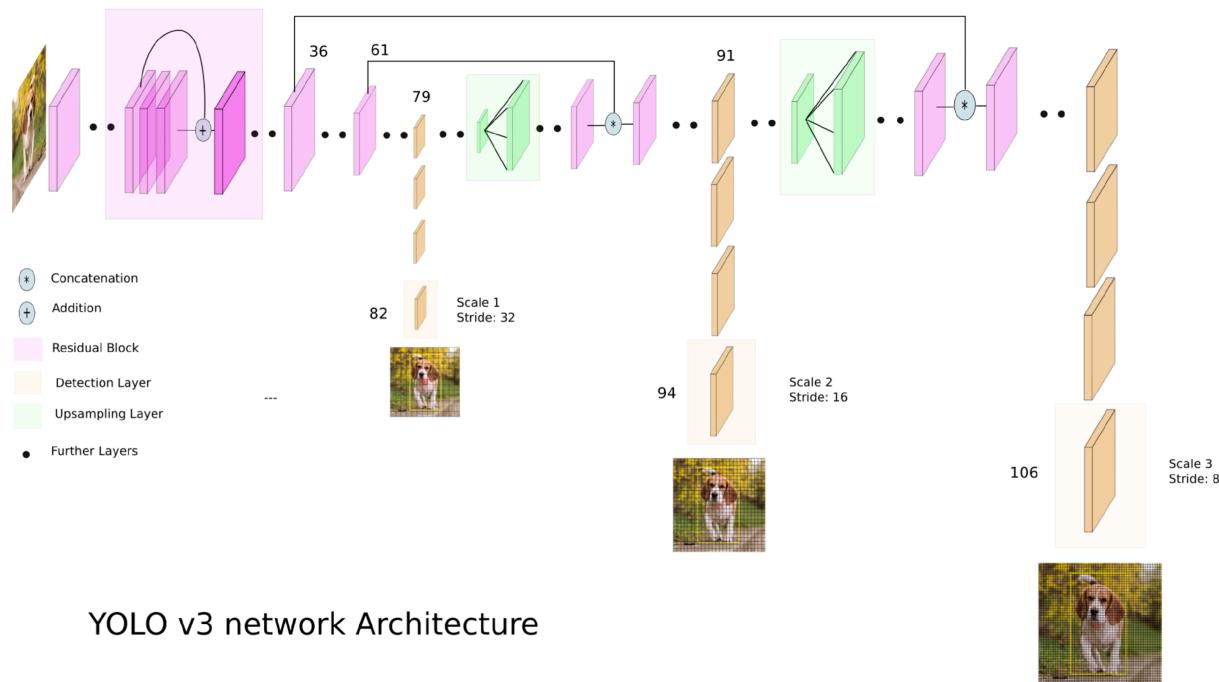
Từ bảng ta thấy với những cải tiến ở trên, mAP của YOLO v2 so với YOLO tiềm nghiệm đã tăng được 24%.

## 2. YOLO v3:

YOLO v2 không áp dụng cho đối tượng cùng thuộc 2 hay nhiều class khác nhau và khả năng phát hiện đối tượng nhỏ còn thấp. Đã được khắc phục ở YOLO v3 bằng phương pháp logistic classifier thay vì softmax cho việc classify đối tượng.

### 2.1. Darknet -53:

Sự thay đổi nhiều nhất so với phiên bản trước là mạng Darknet-53, cũng là nguyên nhân khiến YOLO v3 giảm FPS. Phát hiện các đối tượng nhỏ hơn và độ chính xác của vùng đối tượng bằng sử dụng Multiscale prediction ( $S = 13, 26, 52$ ) và skip-layer concatenation vào kiến trúc. Cải thiện rút trích đặc trưng.



Hình 9: Kiến trúc của mạng YOLO v3 [1]

Residual blocks nằm trong mạng cho phép đầu ra của các lớp nhảy tới các lớp tiếp theo cách 2 - 3 lớp. Nhờ đó khả năng tính gradient ngược trong deep learning (Backpropagation) sẽ nhanh hơn. Đặc biệt là khi mạng có rất nhiều lớp, cho phép chúng ta xây dựng những mạng sâu hơn nữa.

### **3.2. Dự đoán multiscale:**

Thuật toán như sau:

1. Lần đầu detection là tại lớp thứ 82. 81 lớp đầu downsample ảnh.  
Ví dụ với ảnh đầu vào 416x416 thì tại lớp thứ 81 và stride = 32, tại lớp thứ 82 sử dụng 1x1 detection kernel nên sẽ cho chúng ta một detection feature map có dạng:  $(416 / 32) \times (416 / 32) \times (B^*(5+C)) = 13 \times 13 \times 255$  với  $B=3$ ,  $C=80$ .
2. Sau đó feature map sẽ được upsample 2x thành 26x26 và depth concatenated với feature map ở lớp 61 nhờ vào residual blocks. Việc này giúp giữ lại các đặc trưng do downsample tạo ra.
3. Lần thứ 2 detection ở lớp thứ 94, tạo ra feature map có dạng 26x26x255.
4. Tương tự lần 3 detection sẽ ở lớp cuối cùng với feature map có dạng 52x52x255.

### 3.3. Phân lớp và Loss Function:

|               | YOLO v2                                                                                                                                                                                | YOLO v3                                                                                                                                                                                                                                      |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Phân lớp      | Áp dụng softmax function khi các nhãn của vật thể là không liên quan tới nhau (exclusive).                                                                                             | Áp dụng independent logistic classifiers khi các nhãn của vật thể có liên quan tới nhau ‘non-exclusive’.                                                                                                                                     |
|               | Do đó tối thiểu hóa hàm MSE như ở YOLO v1 cho phân lớp.                                                                                                                                | Do đó tối thiểu hóa hàm binary cross-entropy loss cho phân lớp.                                                                                                                                                                              |
| Loss Function | $+ \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2$ $+ \lambda_{\text{noobj}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$ | Anchor box với IoU cao nhất sẽ có confidence(objectness) = 1. Các anchor boxes khác có ngưỡng > 0.5 sẽ không sinh ra cost. Với các anchor box chưa được gán, không sinh ra Classification loss và Localization loss, chỉ có confidence loss. |

### 3.4. Tổng kết:

Đánh đổi FPS, inference time để tăng độ chính xác. (YOLOv2 chạy tốt với 45 FPS, YOLO v3 chạy tốt hơn với FPS 30):

|                           | backbone                 | AP          | AP <sub>50</sub> | AP <sub>75</sub> | AP <sub>S</sub> | AP <sub>M</sub> | AP <sub>L</sub> |
|---------------------------|--------------------------|-------------|------------------|------------------|-----------------|-----------------|-----------------|
| <i>Two-stage methods</i>  |                          |             |                  |                  |                 |                 |                 |
| Faster R-CNN+++ [3]       | ResNet-101-C4            | 34.9        | 55.7             | 37.4             | 15.6            | 38.7            | 50.9            |
| Faster R-CNN w FPN [6]    | ResNet-101-FPN           | 36.2        | 59.1             | 39.0             | 18.2            | 39.0            | 48.2            |
| Faster R-CNN by G-RMI [4] | Inception-ResNet-v2 [19] | 34.7        | 55.5             | 36.7             | 13.5            | 38.1            | 52.0            |
| Faster R-CNN w TDM [18]   | Inception-ResNet-v2-TDM  | 36.8        | 57.7             | 39.2             | 16.2            | 39.8            | <b>52.1</b>     |
| <i>One-stage methods</i>  |                          |             |                  |                  |                 |                 |                 |
| YOLOv2 [13]               | DarkNet-19 [13]          | 21.6        | 44.0             | 19.2             | 5.0             | 22.4            | 35.5            |
| SSD513 [9, 2]             | ResNet-101-SSD           | 31.2        | 50.4             | 33.3             | 10.2            | 34.5            | 49.8            |
| DSSD513 [2]               | ResNet-101-DSSD          | 33.2        | 53.3             | 35.2             | 13.0            | 35.4            | 51.1            |
| RetinaNet [7]             | ResNet-101-FPN           | 39.1        | 59.1             | 42.3             | 21.8            | 42.7            | 50.2            |
| RetinaNet [7]             | ResNeXt-101-FPN          | <b>40.8</b> | <b>61.1</b>      | <b>44.1</b>      | <b>24.1</b>     | <b>44.2</b>     | 51.2            |
| YOLOv3 608 × 608          | Darknet-53               | 33.0        | 57.9             | 34.4             | 18.3            | 35.4            | 41.9            |

Hình 10: Hiệu năng của YOLO v3 [6]

YOLO v3 hơn hẳn so với YOLO v2. Việc phát hiện các vật thể có kích thước nhỏ được cải thiện rất nhiều (5% mAP trong YOLO v2 so với 18.3% mAP trong YOLO v3), nhưng vẫn chưa bắt kịp với mô hình Retina-Net.

### **3. YOLO v4:**

#### **3.1. Back bone:**

Một thành phần trong mạng học sâu, được cấu thành chủ yếu là các lớp convolution nhằm trích xuất các features. Bao gồm Bag of freebies và bag of specials:

##### *a. Bag of freebies:*

Một tập các phương pháp chỉ tăng chi phí huấn luyện hoặc thay đổi chiến lược huấn luyện trong khi giữ inference time mức thấp.

Data augmentation tạo ra các bức ảnh mới từ các bức ảnh trong dữ liệu huấn luyện để giúp cho mô hình học được đa dạng các trường hợp có thể có trong thực tế. Các phương pháp được sử dụng gồm Label Smoothing và DropBlock regularization.

##### *b. Bag of specials:*

Các phương pháp tăng inference time nhưng đổi lại cải thiện đáng kể độ chính xác:

- Sử dụng Mish activation cho các neurons trong mạng.
- Cross-stage partial connection (CSP).
- Multi-input weighted residual connections (MiWRC).

#### **3.2. Detector:**

##### *a. Bag of Freebies:*

- CIoU
- DropBlock regularization
- Mosaic data augmentation
- Self-Adversarial Training

##### *b. Bag of Specials:*

- Mish activation

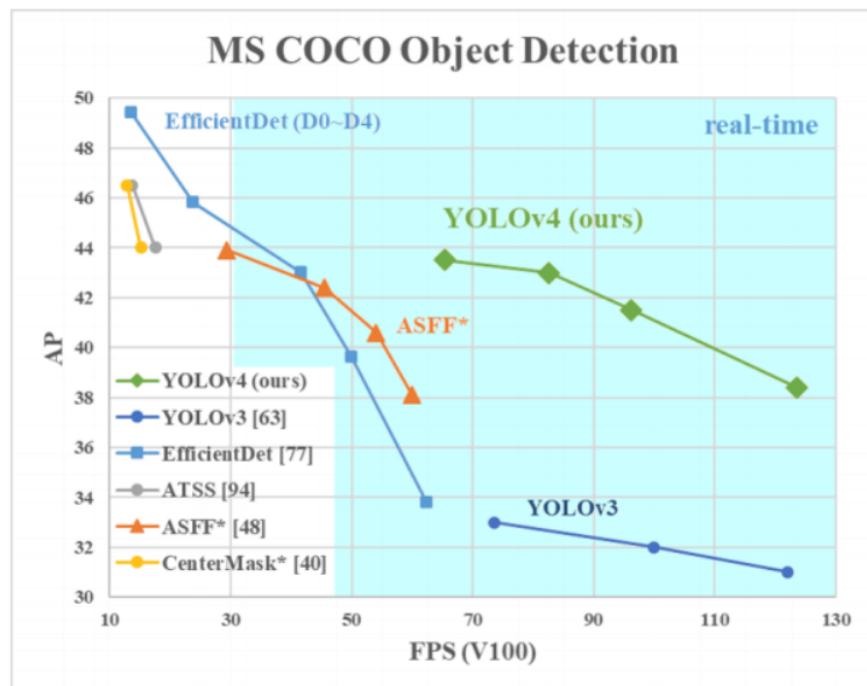
- SPP-block
- SAM-block
- PAN path aggregation block

### 3.3. Kết luận:

Khi chạy trên các kiến trúc Maxwell, Volta, Pascal GPUs thì từ biểu đồ trên ta có thể thấy với cùng một khoảng FPS thì YOLO v4 có độ chính xác cao nhất. So với phiên bản YOLO v3 thì với cùng FPS thì độ chính xác đã cải thiện 10-12 % mAP.

YOLO v4 hoạt động ổn định trên GPU 1080ti hoặc 2080ti. Điều mà các tiềm nhiệm trước không làm được.

YOLO v4 đạt 43.5% AP trên tập dữ liệu MS COCO ở tốc độ 65 FPS, trên GPU Tesla V100.



Hình 11. Hiệu năng của YOLO v4 [7]

## 4. Tổng kết:

| YOLO 1 | - S = 13, B = 2                                                               | Darknet: Image-net1000<br>Fully connected                                                   | Softmax             |
|--------|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|---------------------|
| YOLO 2 | - S = 13, B = 2, C*5<br>- Direct location prediction<br>- Batch normalization | Darknet19<br>Fully connected + classifier<br>(High resolution classifier)                   | Softmax             |
| YOLO 3 | - S = 52, B = 3<br>- Logistic regression cho confidence score                 | Darknet-53<br>Residual block (resNet)<br>Multi-scale prediction<br>Skip-layer concatenation | Logistic classifier |
| YOLO 4 |                                                                               | Backbone: CSP<br>Neck: aggregation net<br>Head: YOLO v3                                     |                     |

## 5. Tương lai của YOLO:

YOLO đã trở nên rất nổi tiếng cho bài toán phát hiện vật thể thời gian thực. Tuy nhiên kể từ phiên bản YOLO v3, tác giả đầu tiên của YOLO là Joseph Redmon đã không còn nghiên cứu và cải thiện kiến trúc này nữa. Anh còn tuyên bố đã ngừng nghiên cứu về thị giác máy tính do các lo ngại công nghệ được sử dụng sai mục đích (sử dụng cho quân sự, các lo ngại về quyền riêng tư).

Tuy thế, chúng ta vẫn sẽ nhận được các "bản nâng cấp" YOLO từ các tác giả khác, như YOLO v4 vừa mới ra mắt gần đây. Và trong tương lai YOLO v5 có thể sẽ được đăng báo chính thống.

## IV. Tham khảo:

- [1] Deep Learning for Vision Systems
- [2] You only look once, unified, real-time object detection | by Joseph Redmon
- [3] Object Detection Yolo | by Army Intelligence
- [4] Yolo Object Detection | v7Labs
- [5] YOLO9000: Better, Faster, Stronger | by Joseph Redmon Ali Farhadi
- [6] YOLOv3: An Incremental Improvement | by Joseph Redmon Ali Farhadi
- [7] Explanation of YOLO V4 a one stage detector | by Pierrick RUGERY | Becoming Human: Artificial Intelligence Magazine
- [8] What's new in YOLOv4?. YOLO is a real-time object recognition... | by Roman Orac | Towards Data Science
- [9] Cloth Detection | by 8-bit | University of Science - HCMUS