Single-Stage Object Detection (YOLO)

Lesson of Content

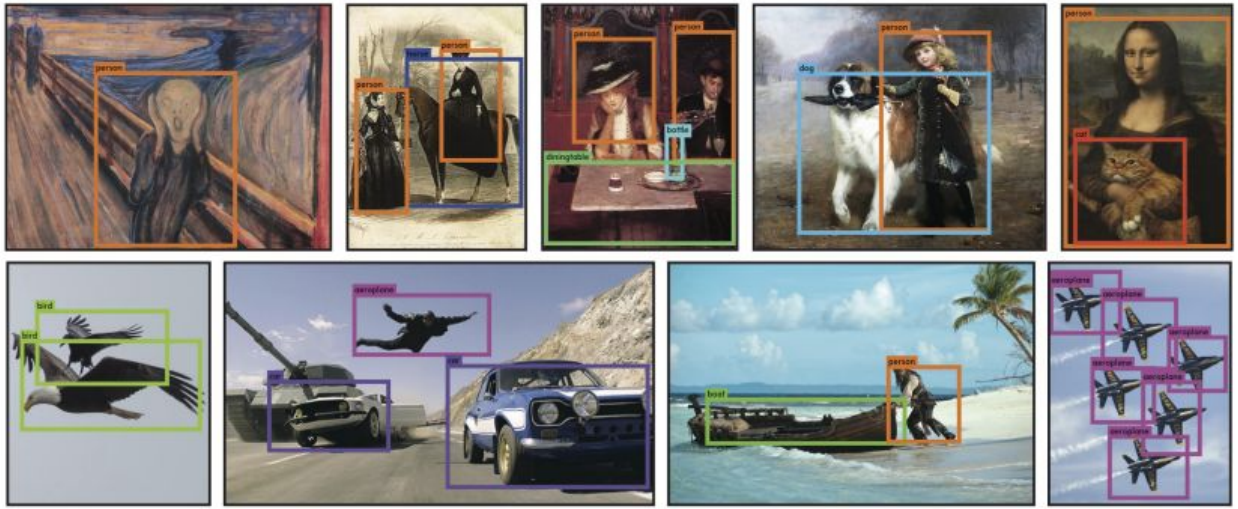
1. YOLOv1
2. YOLOv2
3. YOLOv3
4. YOLOv4
5. YOLOv5
6. Training Model with YOLOv5

Overview

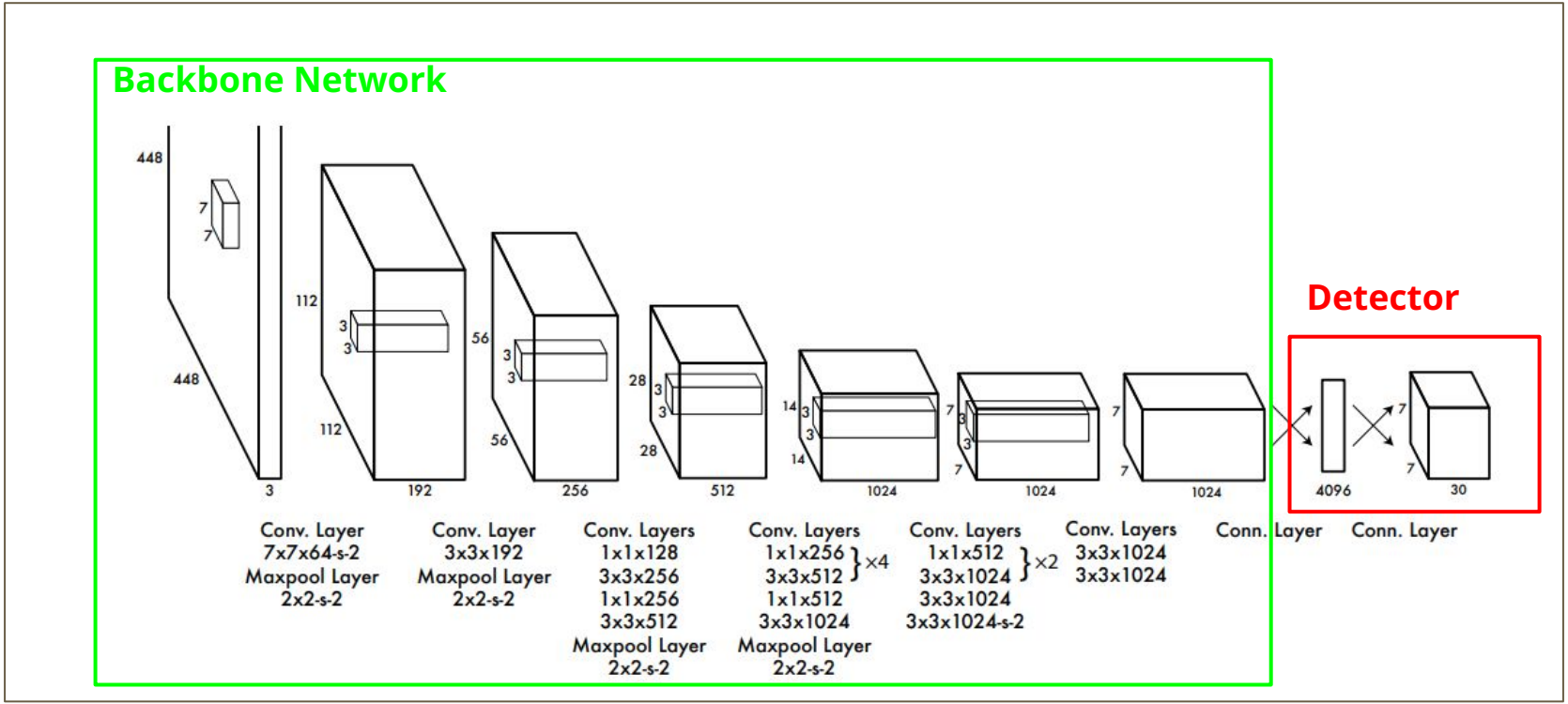
paper: https://arxiv.org/pdf/1506.02640.pdf
Year: 05.2016

YOLO: You Only Look Once

- Tạo ra để giải quyết vấn đề object detection trong real time

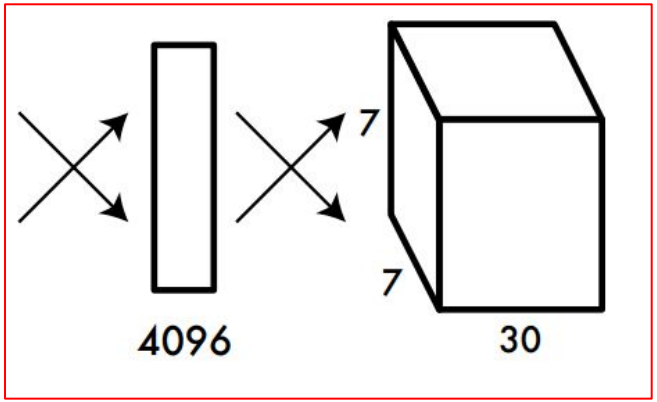


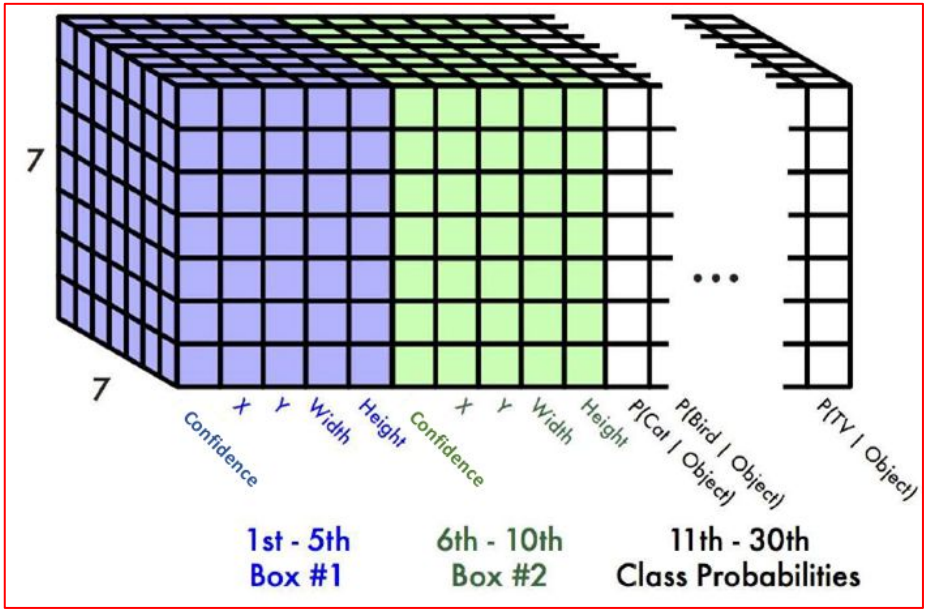
Architecture



Backbone Network: Gồm 24 lớp CNN để trích xuất đặc trưng của ảnh đầu vào

Architecture



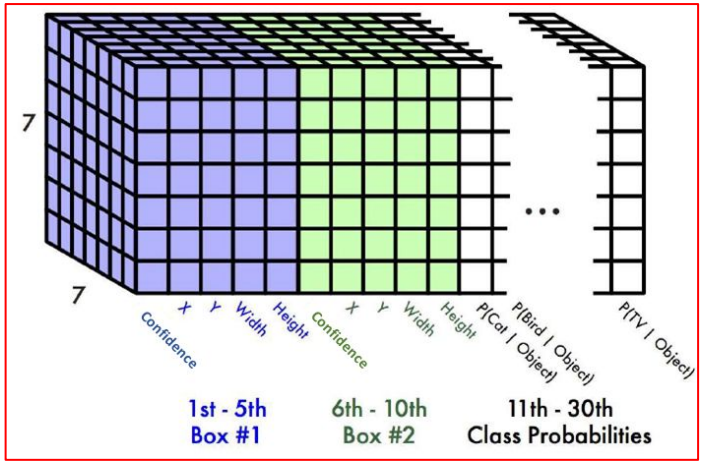


Detector: Khối này chuyển đổi các thông tin từ lớp backbone sang các thông tin về dạng box và confident.

Inference Model

- Đầu vào được chi thành S x S các grid cells **(S = 7)** . Nếu tâm của object thuộc grid cell nào thì cell đó chịu trách nhiệm phát hiện đối tượng đó.
- Mỗi grid cell sẽ dự đoán B bboxes **(B = 2)** và xác suất các classes nằm trong cell đó.
- Mỗi thông tin về bboxes bao gồm x, y, width, height và confidence

Như vậy 30 = S x S x (5B + C) = 7 x 7 x (5x2 + 20)

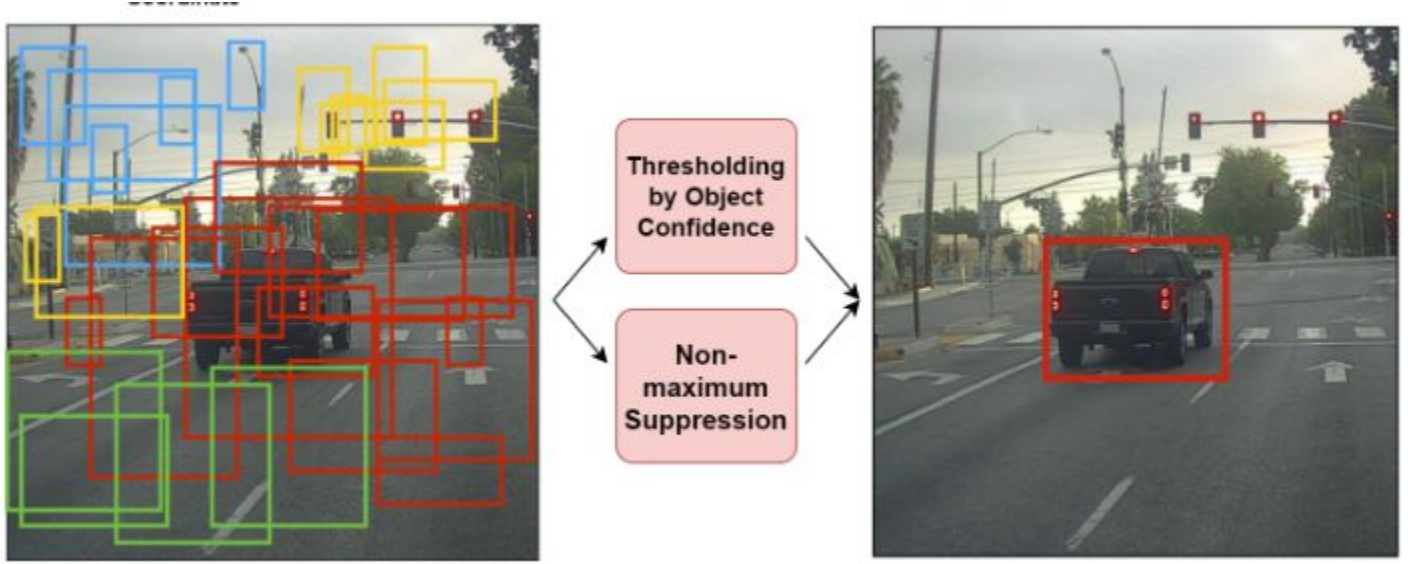


$$\text{class confidence score} = \left[Pr(object) \cdot IOU_{pred}^{truth}\right] \cdot Pr(class_i|object) = Pr(class_i) \cdot IOU_{pred}^{truth}$$

Inference Model

Problem: Nhiều cell dự đoán với xác suất bé. Cùng một cells có nhiều hơn 1 dự đoán.

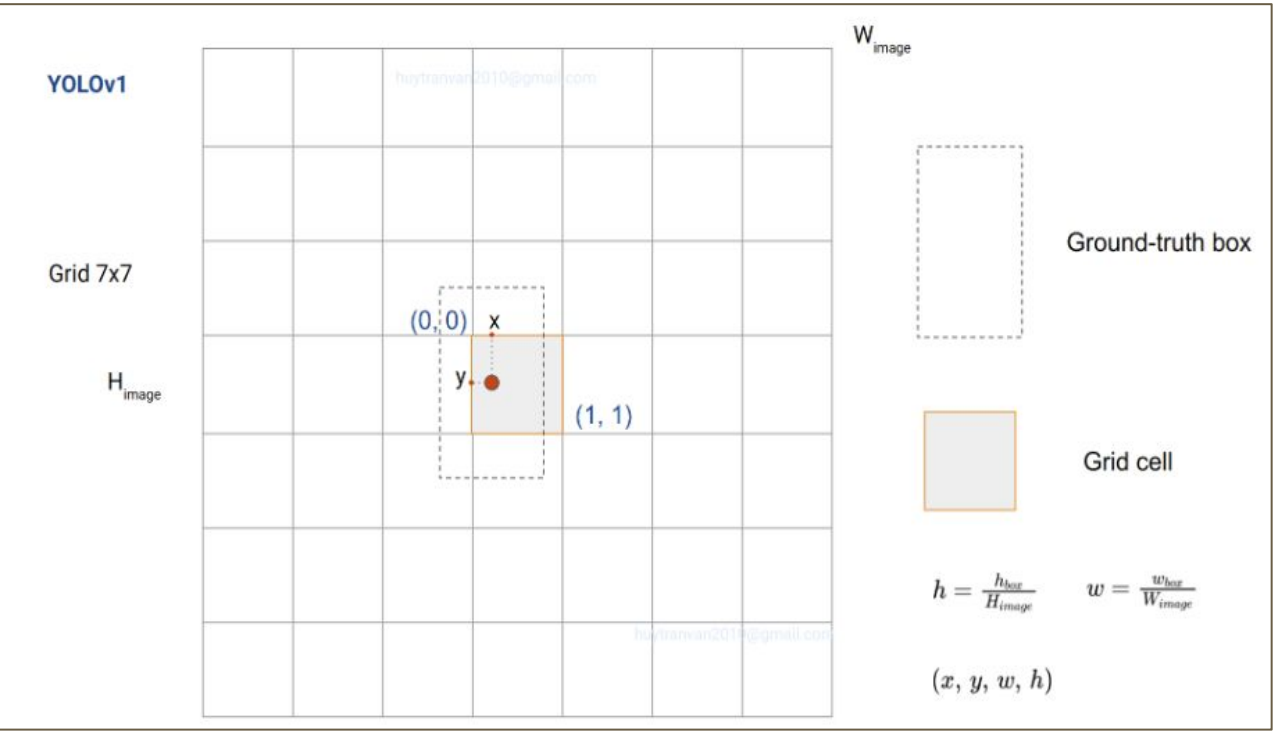
Solve: Dùng threshold để lọc bớt đối tượng và dùng NMS để lấy lại đối tượng duy nhất có độ tin cậy cao.



Training Model

Data preprocessing

Input size: 448 x 448 => **Size cells:** 64x64



Raw:
x, y, w, h = 200, 220, 72, 128

Process:
c_x = 3
c_y = 3

x_ = x/W = 0.4464
y_ = y/H = 0.4911
w_ = w/W = 72/448 = 0.1607
h_ = h/H = 128/448 = 0.2857

x_offset = (x_*W - c_x*64)/64 = 0.125
y_offset = (y_*h - c_y*64)/64 = 0.4375

Training Model Loss Function

Regression loss

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

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

Confidence loss

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

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

Classification loss

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

Regression Loss: Tối ưu hóa về độ lệch của bbox.

Confidence Loss: Tối ưu hóa mức độ tin cậy với giá trị thực tế (0 nếu không chứa đối tượng và 1 khi có đối tượng)

Classification Loss: Tối ưu hóa việc dự đoán xác suất của từng lớp đối tượng.

- $\mathbb{1}_{ij}^{\text{obj}}$: Là hàm chỉ số, bằng 1 khi cell đấy chứa object hoặc bằng 0 nếu không chứa (**"Chứa"** là tâm của object nằm trong cell đó). Ngược lại hoàn toàn là $\mathbb{1}_{ij}^{\text{noobj}}$
- Có hai hệ số cân bằng trong hàm loss là λ_{coord} và λ_{noobj}, lần lượt có giá trị là 5.0 và 0.5
- $\hat{C}_i$ là confidence score prediction = **score object (p_o) * IoU (bbox_pred, bbox_grouth)**
- $\hat{p}_i(c)$ là dự đoán xác suất tại cell thứ i và class thứ c, và phải được đi qua hàm **softmax**

Good Ideas

- **Có sự đánh giá ảnh hưởng của bbox lớn và bbox nhỏ trong hàm loss**

$w_1 = 0.55, \hat{w}_1 = 0.5, w_2 = 0.3, \hat{w}_2 = 0.25$, nhận thấy $(w_1 - \hat{w}_1) = (w_2 - \hat{w}_2)$, tuy nhiên bounding boxes nhỏ hơn $w_2 = 0.3, \hat{w}_2 = 0.25$ bị lệch nhiều hơn so với bounding boxes lớn $w_1 = 0.55, \hat{w}_1 = 0.5$. Làm cách nào đó để trừng phạt bounding box nhỏ hơn, ở đây dùng square root. Thật vậy $\sqrt{0.3} - \sqrt{0.25} = 0.0477 > 0.0345 = \sqrt{0.55} - \sqrt{0.5}$.

- **Đặt nền móng cho mô hình mạng 1 stage, giúp phát hiện đối tượng nhanh và real time.**

Limitations

- Dự đoán tối đa được 49 objects (7 x 7)
- Môi cells chỉ predict được 1 vật thể duy nhất có score cao nhất, nên các vật thể gần nhau hoặc chồng lên nhau sẽ không dự đoán được
- Độ chính xác thấp
- Không có sự giới hạn cho tọa độ của các bbox (có thể vượt ngưỡng lớn hơn 1).
- Chưa có khái niệm **anchor box**, dẫn đến kém hiệu quả khi dự đoán các đối tượng có kích thước đa dạng.
- Không thể dự đoán trong trường hợp hai đối tượng có chung tọa độ tâm

Overview

paper: https://arxiv.org/pdf/1612.08242.pdf
Year: 12.2016

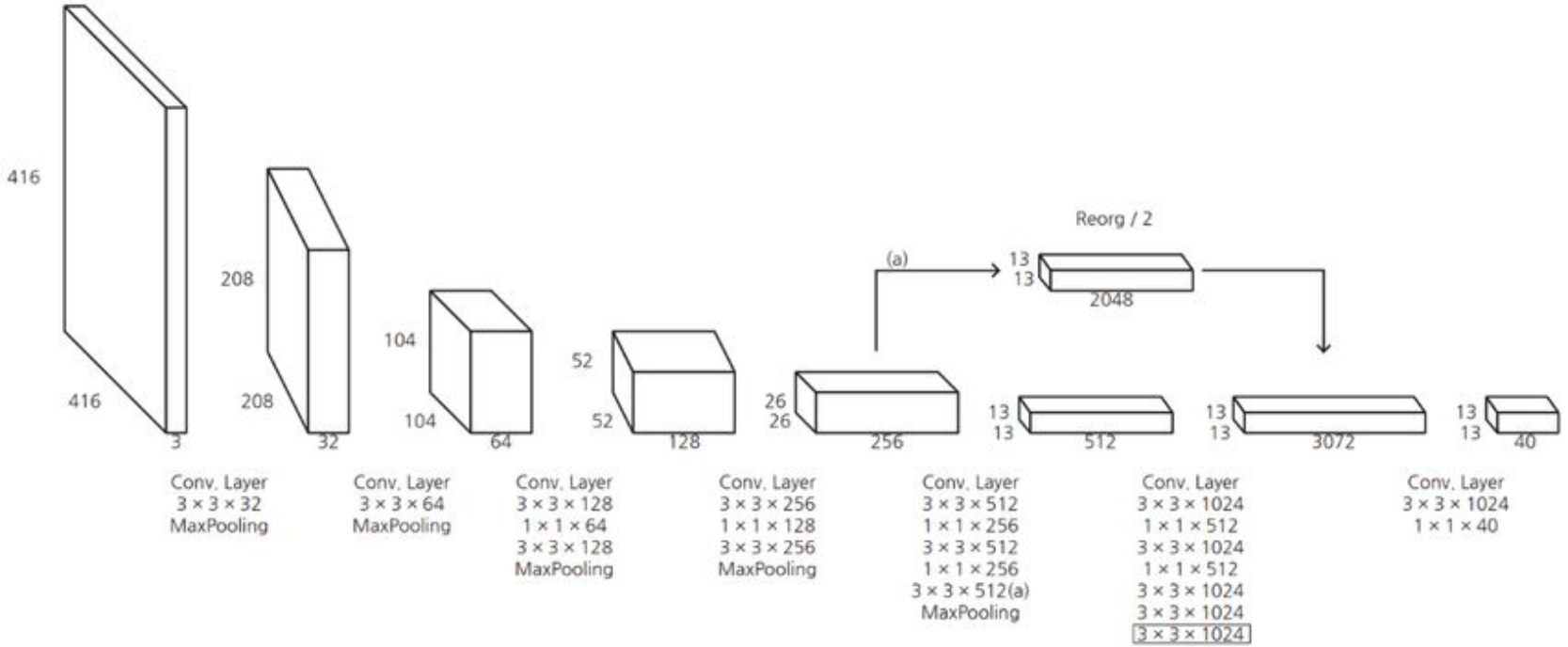
YOLO9000: Better, Faster, Stronger

- Tạo ra để giải quyết vấn đề object detection trong real time

Improvement:

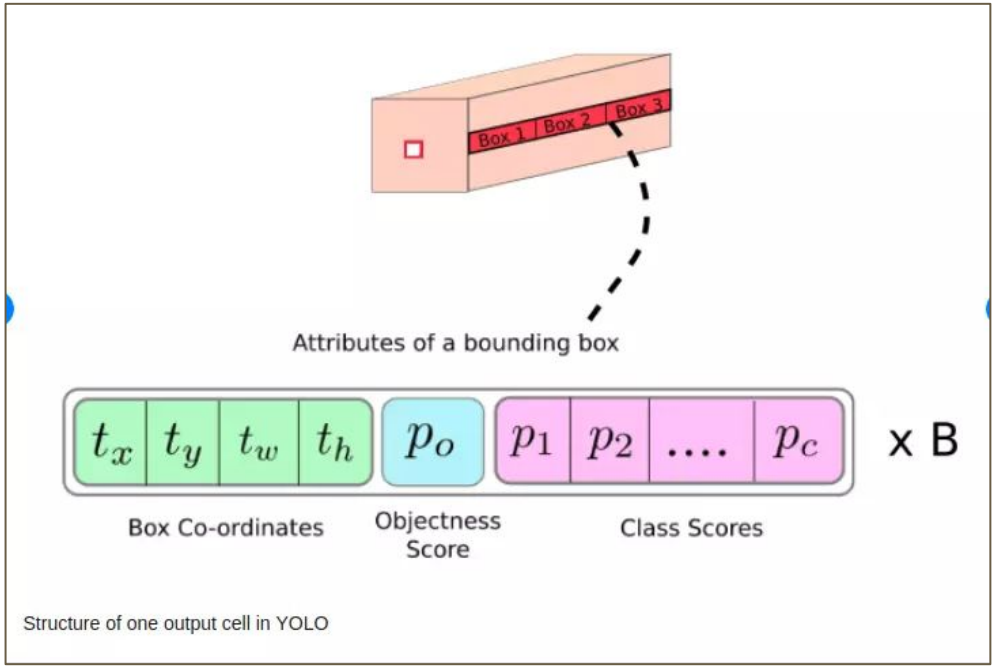
- Dùng backbone Darknet-19, nhẹ hơn nên tối ưu về tốc độ.
- Định nghĩa anchor boxes: Là một định nghĩa cho trước các tỉ lệ của bbox dựa trên tập huấn luyện.
- Định nghĩa hàm chuẩn hóa, cải thiện dự đoán về **tâm đối tượng**.
- Có dùng “Fine-grained Features": kết hợp thông tin từ các lớp đầu dữ liệu, cụ thể dùng Reorg
- Dùng nhiều cell hơn : 13x13

Architecture



Reorg: Kỹ thuật tổ chức lại từ feature map 26x26 thành 13x13, giúp việc concat ở phía sau chính xác
Output: 40 = (4 + 1 + C) *B = (5 + 3) * 5 , phân loại 3 class, và B = 5 là số lượng anchor box

Architecture



Output = S x S x (4 + 1 + C) x B

t_x, t_y, t_w, t_h : Không phải giá trị thực của bbox, mà chỉ là giá trị offset của bbox so với anchor box cho trước. Anchor box này có kích thước (p_w, p_h) được định sẵn.

p_o: Là giá trị Objectness Score (Cũng chính là Confident Score khi inference) .

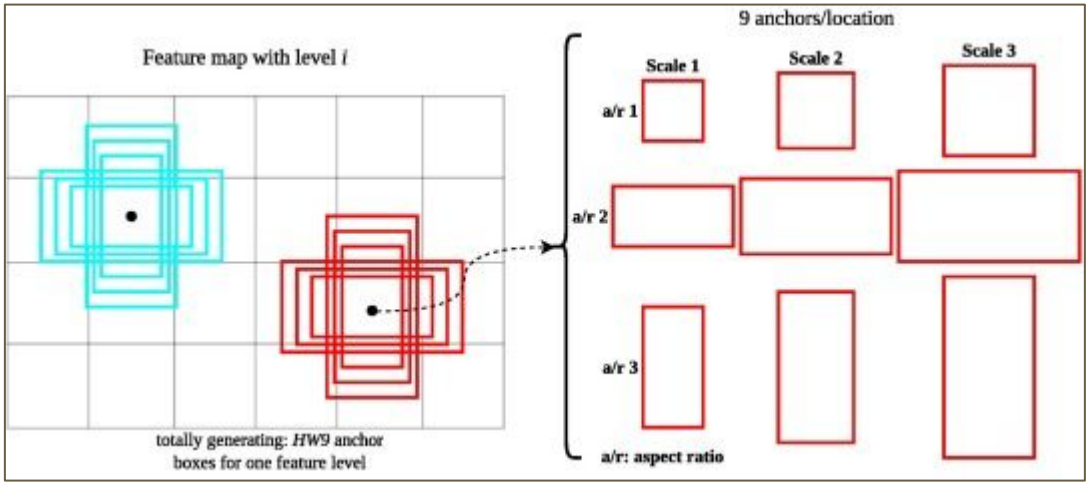
p_1, p_2, p_3, … p_c: Là tỉ lệ xác suất cho từng class.

B: Là số lượng anchor box cho 1 grid cell, (B = 5 ở yolov2).

C: Là số class định nghĩa trước, trong paper số class là 80.

Anchor Boxes

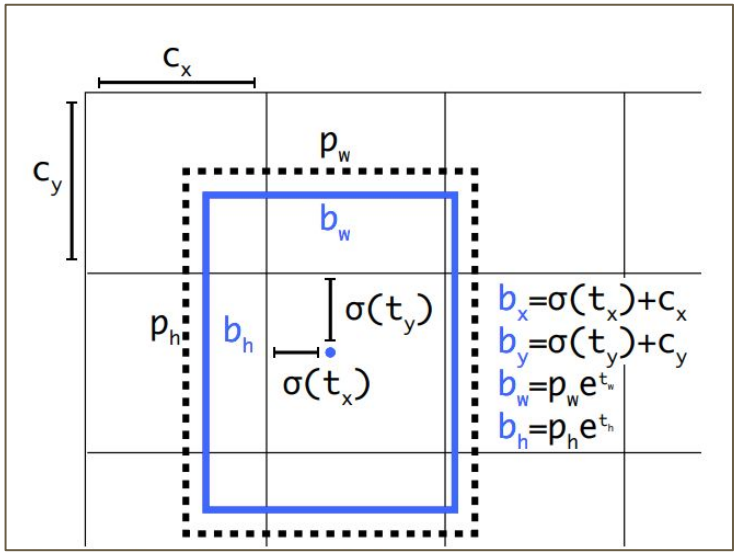
Anchor boxes là những boundingbox có **tỉ lệ xác định**. Trong quá trình huấn luyện **tỉ lệ này sẽ được canh chỉnh** sao cho phù hợp với các đối tượng thực tế



Việc khởi tạo các anchor boxes cho biết tỉ lệ, nhưng **không có kích thước chính xác.**
Thường sẽ dùng thêm **K-Means** để khởi tạo các kích thước này dựa trên dữ liệu huấn luyện. (Nó sẽ là các kích thước phổ biến).
=> Giúp có sự ước lượng ban đầu tốt.
=> Giảm mất mát trong hàm loss, giúp tăng tốc quá trình huấn luyện.

Ở YOLOV2 sử dụng K = 5

Location Prediction



$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

- ***σ(tx), σ(ty), σ(to) :*** Dùng hàm sigmoid để ép về miền giá trị khoảng (0,1)
- ***(bx, by, bw, bh) :*** Là giá trị thật của bbox, được tính từ **(*tx*, *ty*, *tw*, *th*)** và **(*pw*, *ph*).**
- ***(cx, cy)*:** là tọa độ x,y của hình vuông chưa bbox trong ô đó. (0 -> 12).
- Các giá trị ***(bx, by)*** sẽ được chuẩn hóa bằng cách chia cho S (S = 13).
- ***(bw, bh)*** phải được chuẩn hóa bằng cách chuẩn hóa **(*pw*, *ph*).** Sẽ lấy chia cho **(w,h)** của ảnh input **(448, 448).**

Lưu ý: Khi inference thì các giá trị **(*tx*, *ty*, *tw*, *th*)** cũng chính là kích thước của tâm và kích thước của bbox luôn, ta chỉ việc chuẩn hóa theo kích thước ảnh là ra kích thước thật.

Limitations

- Dự đoán kém với các object nhỏ
- Không có khả năng xử lý với các đối tượng ở nhiều tỉ lệ khác nhau
- Độ chính xác vẫn chưa thực sự cao
- Không thể dự đoán hai đối tượng cùng chung tọa độ tâm

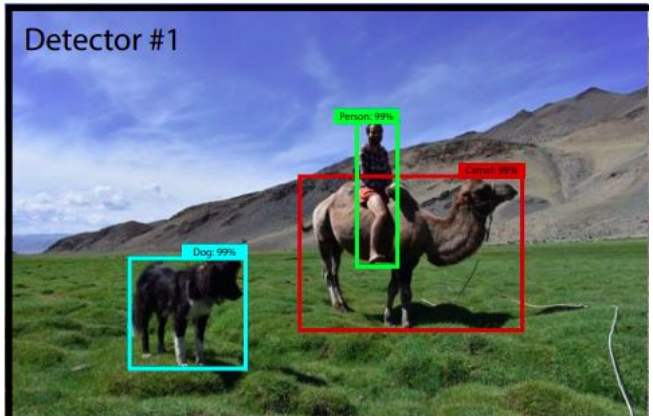
Overview

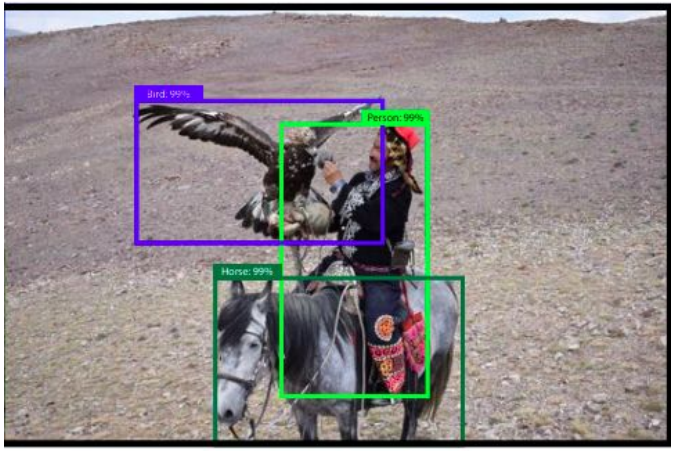
paper: https://arxiv.org/pdf/1804.02767.pdf
Year: 4.2018

YOLOv3: An Incremental Improvement

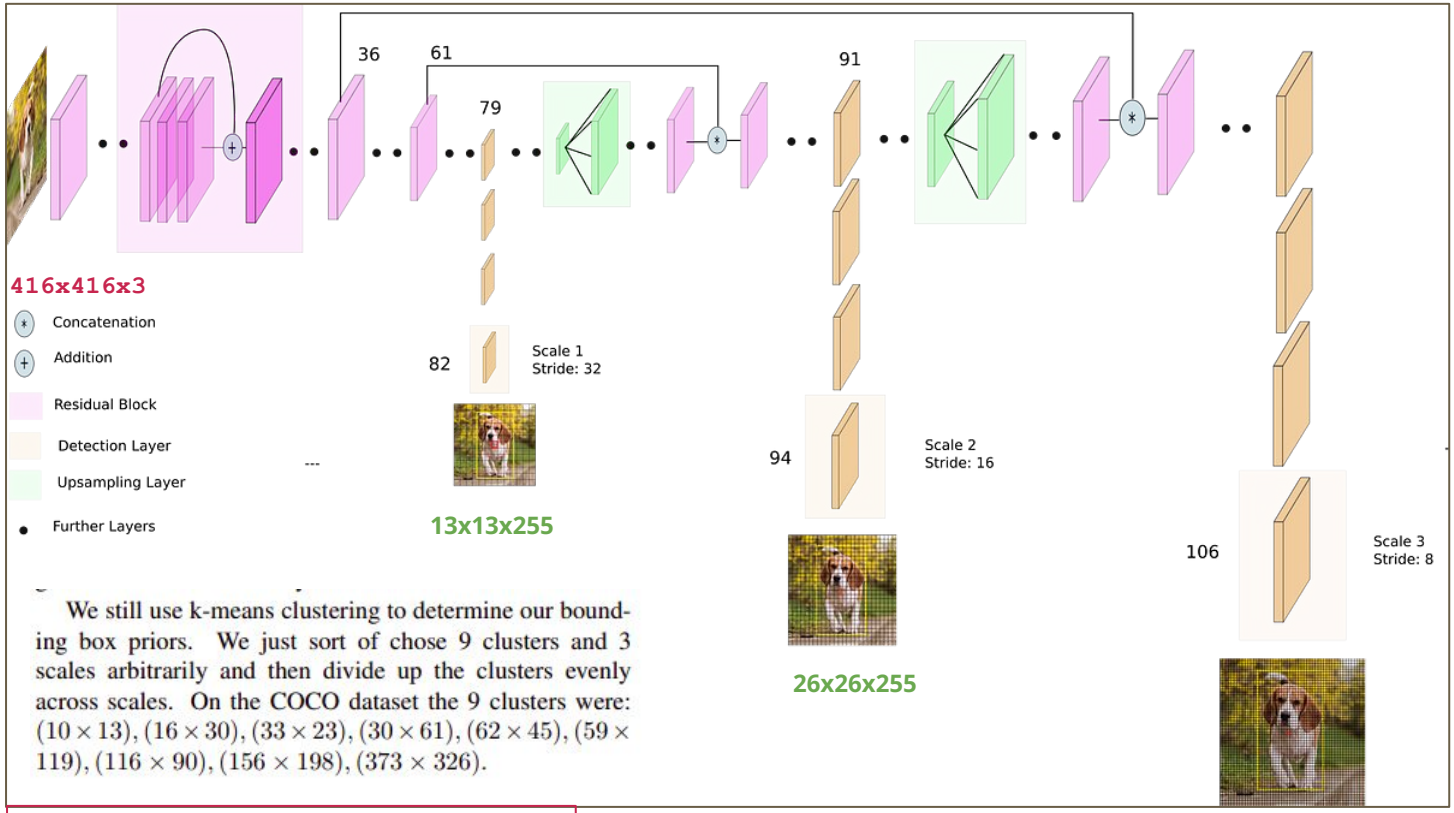
Improvement:

- Phát hiện đối tượng nhỏ tốt hơn yolov2
- Đối tượng được chia ở nhiều tỉ lệ khác nhau
- Độ chính xác cải thiện hơn yolov2
- Sử dụng Residual block
- Tận dụng tốt **Feature Pyramid Network**
- Thay đổi lại hàm loss cho tối ưu hơn
- Bắt được hai đối tượng có cùng tâm





Architecture



We still use k-means clustering to determine our bounding box priors. We just sort of chose 9 clusters and 3 scales arbitrarily and then divide up the clusters evenly across scales. On the COCO dataset the 9 clusters were: $(10 \times 13), (16 \times 30), (33 \times 23), (30 \times 61), (62 \times 45), (59 \times 119), (116 \times 90), (156 \times 198), (373 \times 326)$.

255 = (4 + 1 + 80)*3 = (5 + C)*B

Đối với lưới 13×13:

- Anchor box 1: (116×90)
- Anchor box 2: (156×198)
- Anchor box 3: (373×326)

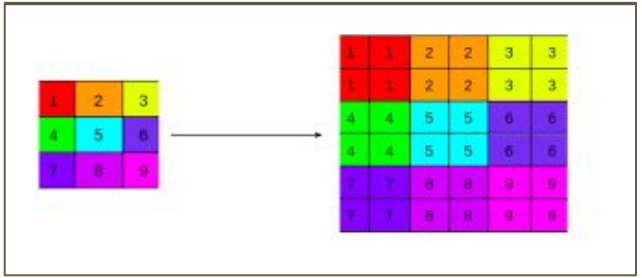
Đối với lưới 26×26:

- Anchor box 1: (30×61)
- Anchor box 2: (62×45)
- Anchor box 3: (59×119)

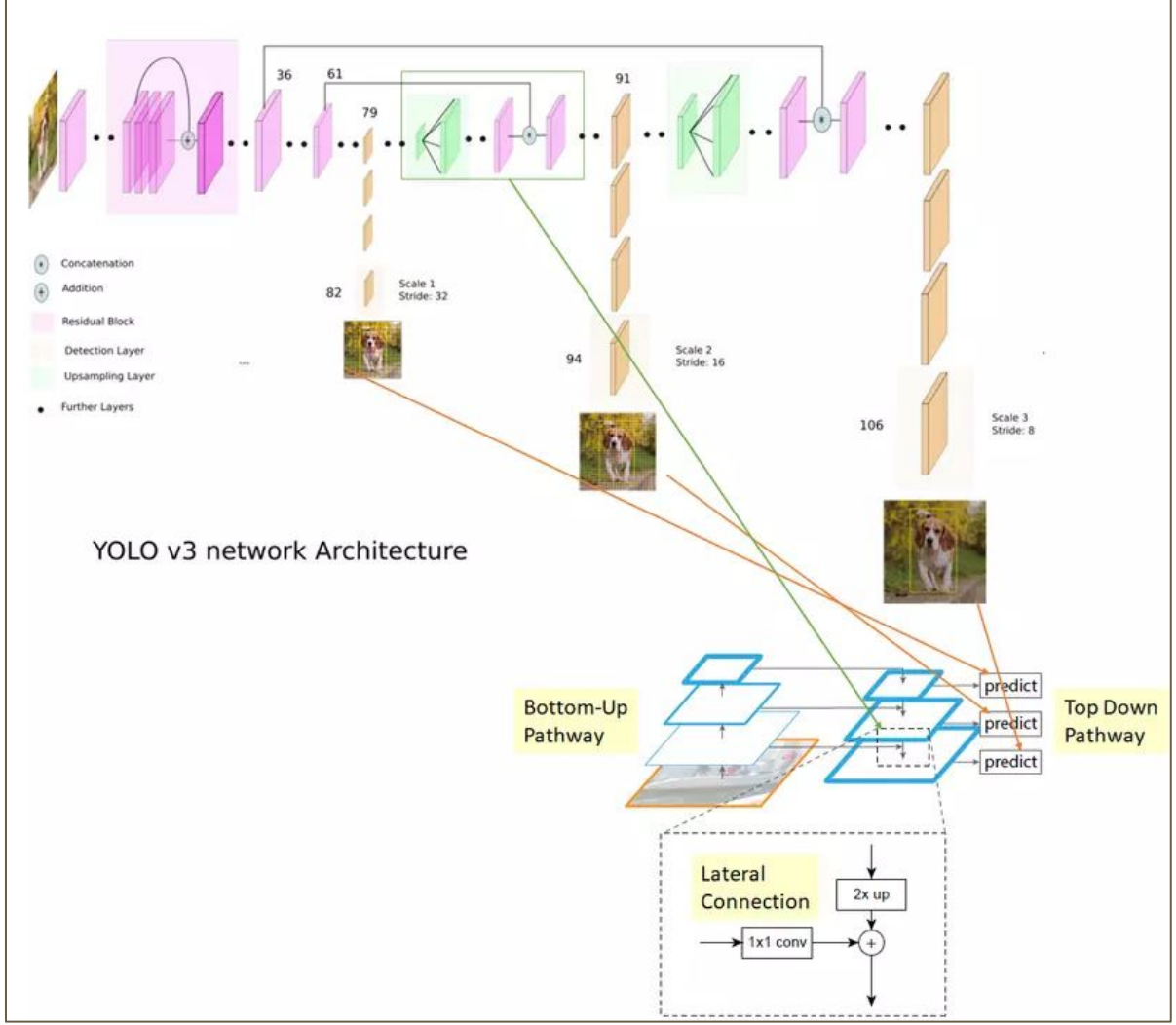
Đối với lưới 52×52:

- Anchor box 1: (10×13)
- Anchor box 2: (16×30)
- Anchor box 3: (33×23)

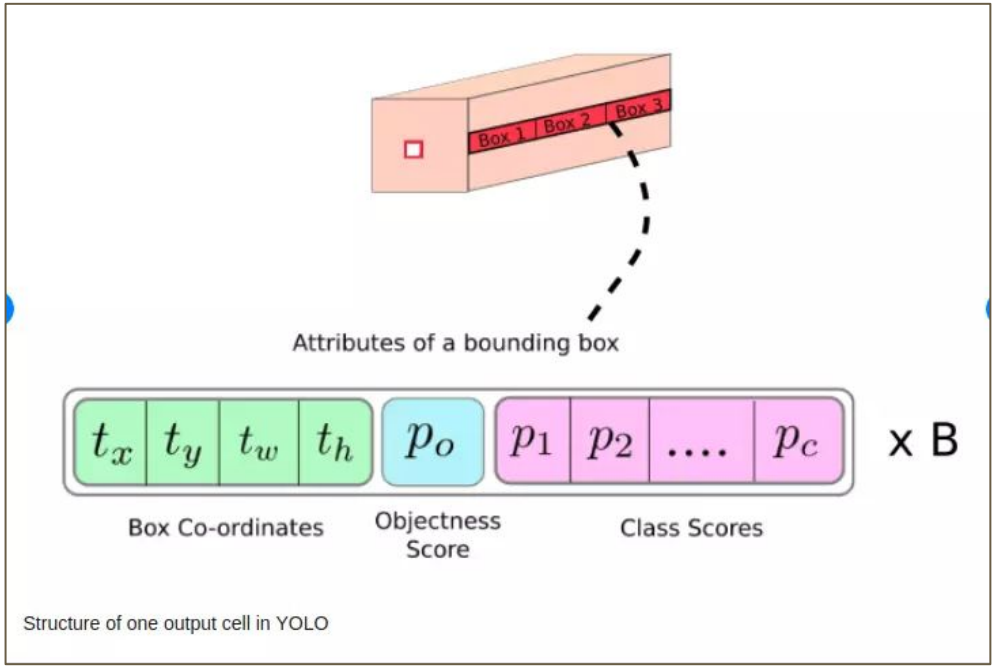
3. YOLOv3



Upsampling Block



Architecture



t_x, t_y, t_w, t_h : Không phải giá trị thực của bbox, mà chỉ là giá trị offset của bbox so với anchor box cho trước. Anchor box này có kích thước (p_w, p_h) được định sẵn.

p_o: Là giá trị Objectness Score (Cũng chính là Confident Score khi inference) .

$p_1, p_2, p_3, \ldots p_c$: Là tỉ lệ xác suất cho từng class.

B: Là số lượng anchor box cho 1 grid cell, thường là 3.

C: Là số class định nghĩa trước, trong paper số class là 80.

Lý giải: Ngõ ra của Yolov3: 13x13x255, 26x26x255, 52x52x255
Output sau cùng là: (13x13 + 26x26 + 52x52) x 255 = 3549 x 255

Loss Function

Regression loss

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

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

Confidence loss

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

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

Classification loss

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

Loss function của Yolov1 và Yolov2

Regression Loss

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{i,j}^{obj} \left((t_x - \hat{t}_x)^2 + (t_y - \hat{t}_y)^2 + (t_w - \hat{t}_w)^2 + (t_h - \hat{t}_h)^2 \right)$$

Confidence Loss

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{i,j}^{obj} \left(-\log(\sigma(t_o)) \right) + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{i,j}^{noobj} \left(-\log(1 - \sigma(t_o)) \right)$$

Classification Loss

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

$\hat{p}_i(c)$ là dự đoán xác suất tại cell thứ i và class c, và phải được đi qua hàm **sigmoid**

$\hat{t}_x$ là dự đoán của offset tâm x và được đi qua làm **log**

Limitations

- **Độ chính xác:** Cao hơn các phiên bản trước, nhưng vẫn còn thấp hơn so với mô hình mạng two stage.

- **Tốc độ:** Tốc độ vẫn chưa ngon (mặc dù có yolov3-tiny nhưng độ chính xác không ngon) do dùng mạng Darknet-53 khá nặng về lưu trữ và tính toán.

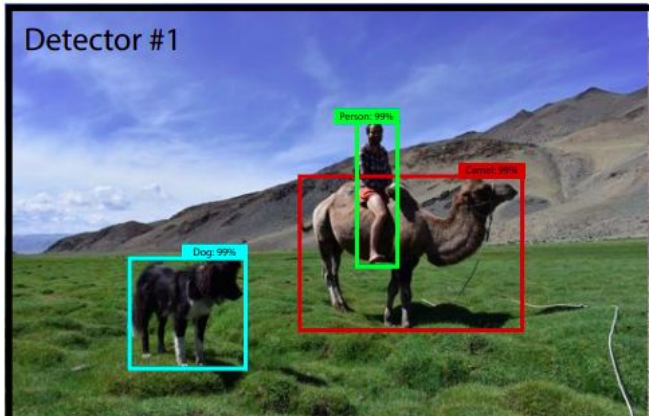
Overview

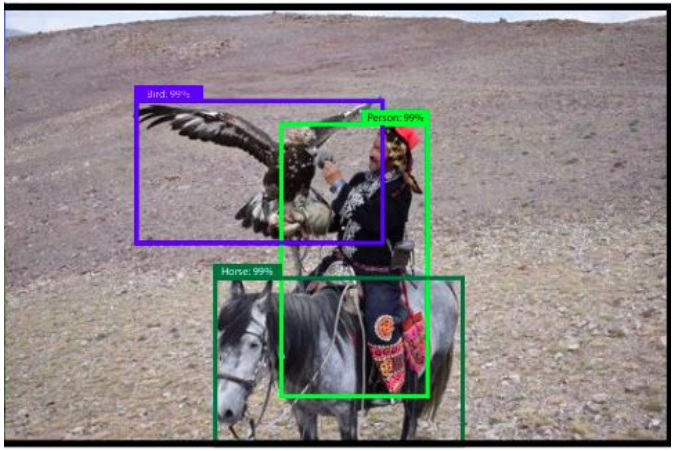
paper: https://arxiv.org/pdf/2004.10934.pdf
Year: 4.2020

YOLOv4: Optimal Speed and Accuracy of Object Detection

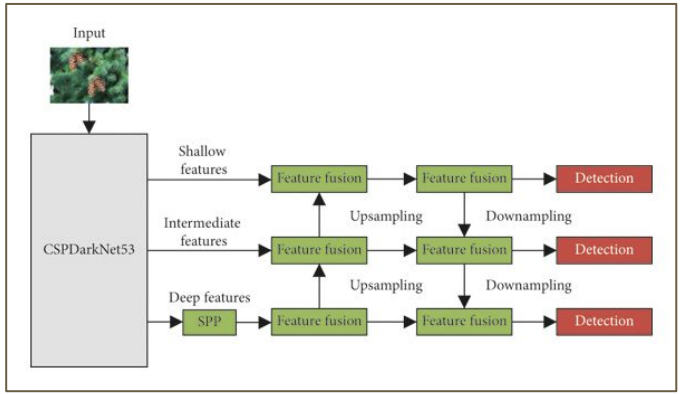
Improvement:

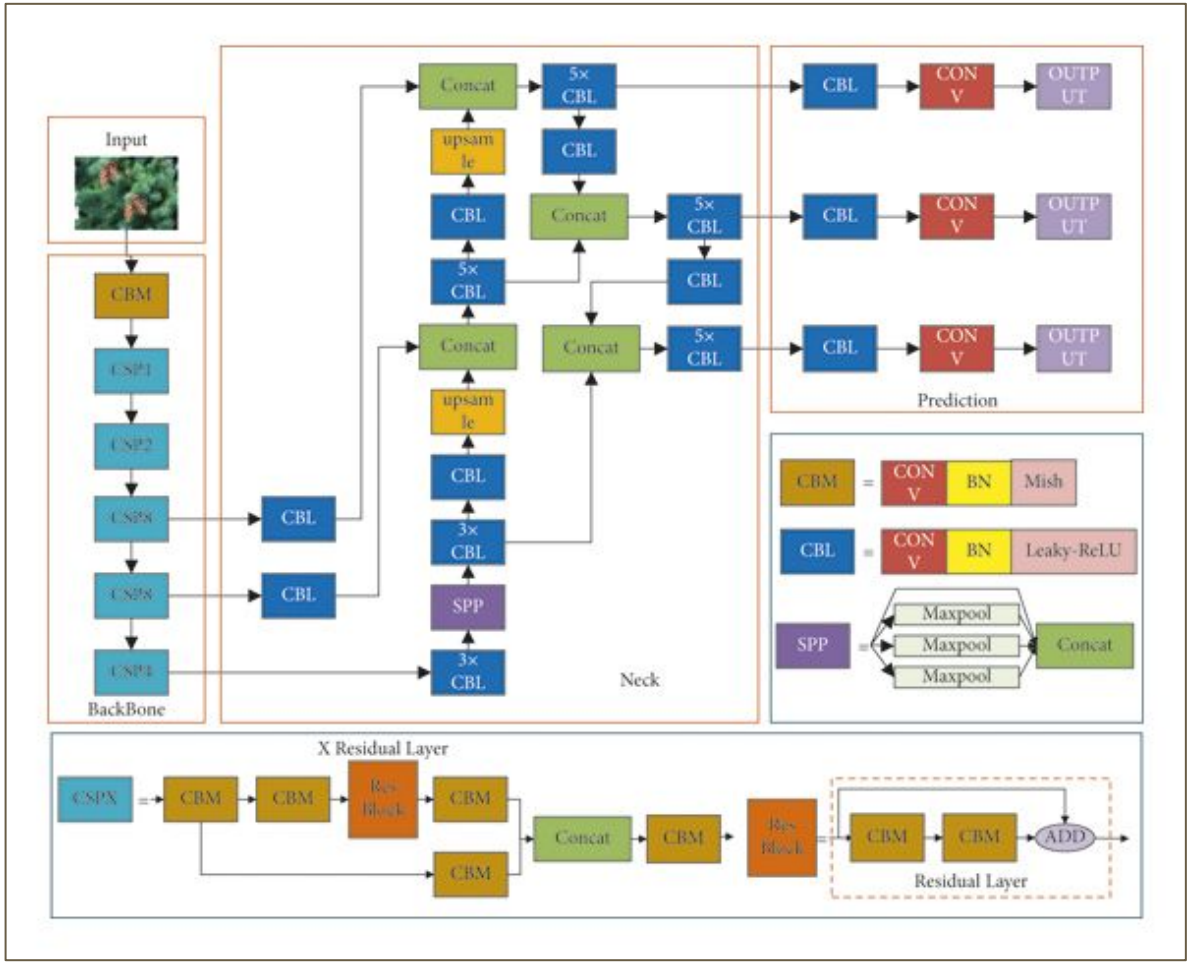
- Tăng độ chính xác
- Tăng tốc độ inference mô hình





Architecture





Bag of Freebies (BoF)

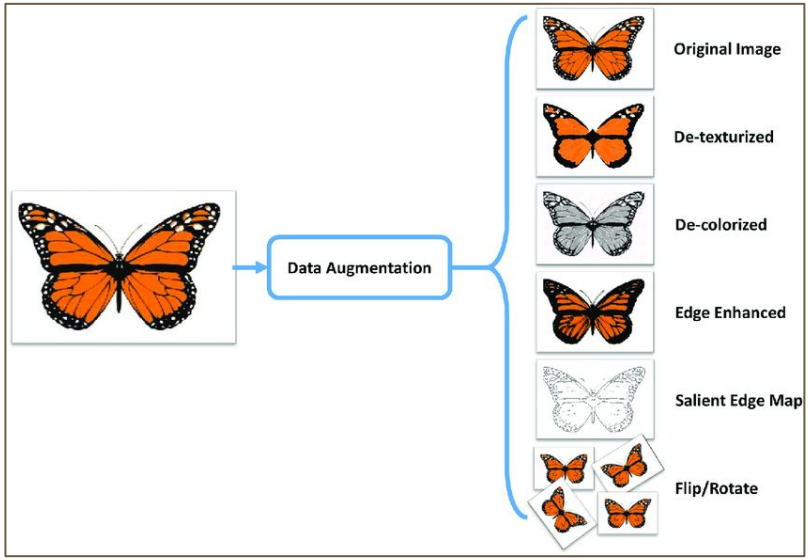
Là một tập hợp các kỹ thuật thay đổi **chiến thuật training** hoặc **chi phí training** để làm **tăng hiệu suất mô hình** mà **không làm tăng thời gian inference**.

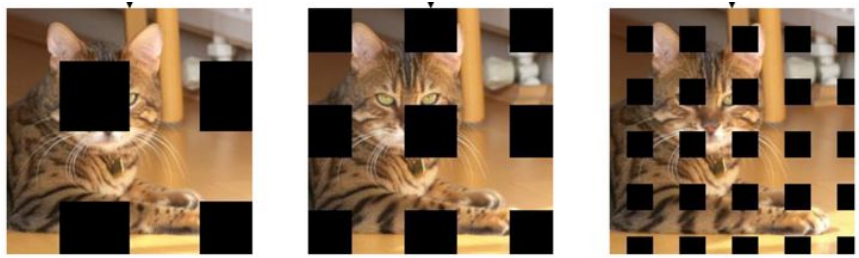
- Data Augmentation:
  - Thay đổi về màu sắc của ảnh: Độ sáng, tương phản, HUE, Saturation,..
  - Thay đổi hình học của ảnh: random scaling, cropping, rotating, flipping,..
  - Phương pháp khác: Grid Mask, Cutout, MixUp, CutMix,..

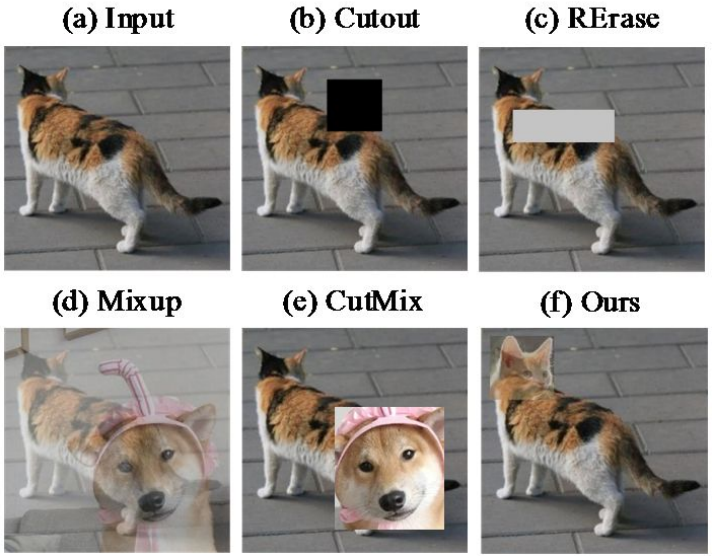
- Semantic Distribution Bias in Datasets:
  - Data imbalance giữa các classes
  - Không thể diễn giải mối quan hệ giữa các lớp khác nhau với biểu diễn one-hot

- Objective Function of BBox Regression:
  - Hàm loss trong yolo dùng MSE để tìm độ sai khác giữa các tọa độ tâm và chiều cao, chiều rộng. Và nó chưa thực sự work well.
  - Đề xuất phương pháp **Anchor based approaches**.

Data Augmentation

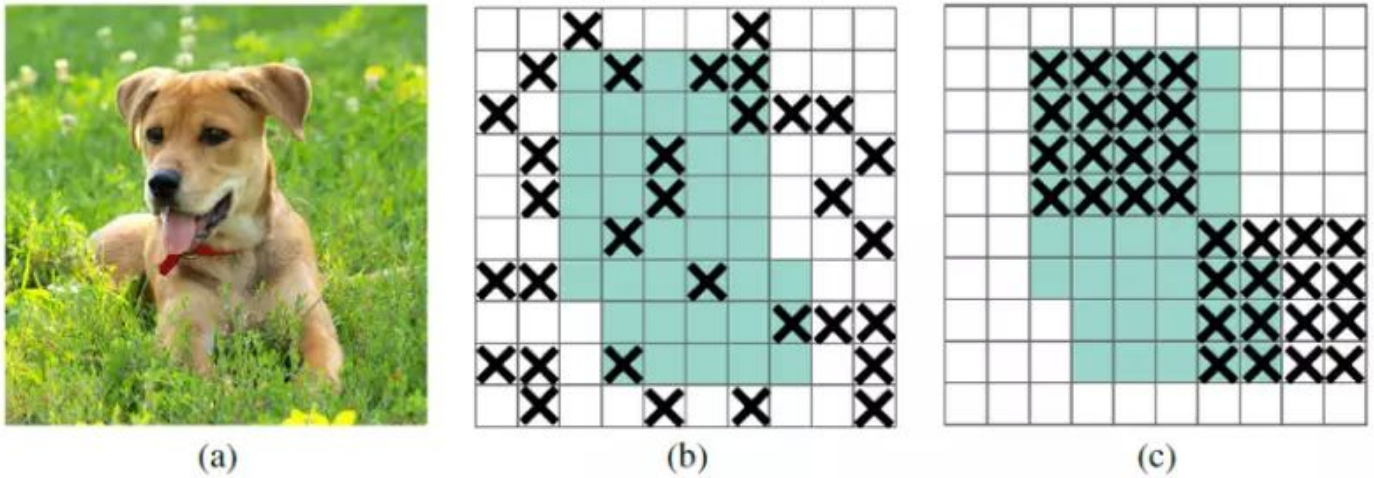






DropBlock regularization

Chúng sẽ tìm cách bỏ đi một nhóm vị trí gần nhau trong feature map. Do các feature map cạnh nhau thường có tính tương quan cao với nhau, nên việc bỏ ngẫu nhiên thì sẽ không hiệu quả.



(a) Input Image, (b) Dropout Randomly at Feature Maps, (c) DropBlock at Feature Maps

Semantic Distribution Bias in Datasets

Là một vấn đề hay gặp trong các bài toán deep learning về việc phân phối dữ liệu ảnh hưởng đến tính generation của mô hình. Và việc dùng one-hot không thể hiện mối quan hệ giữa các classes.

Giải pháp trước đó:

- Dùng dữ liệu negative khó hoặc dữ liệu online khó
- Dùng focal loss để giải quyết vấn đề **imbalance data**

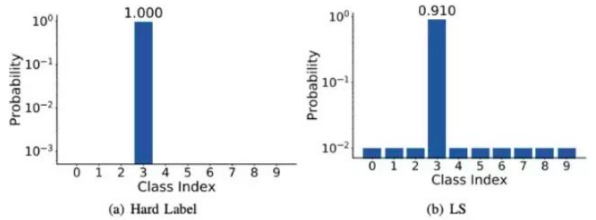
$$\mathrm{CE}(p_t) = -\log(p_t)$$
$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t)$$

Yolov4: Đề xuất dùng Smooth label

- Tinh chỉnh được e.

$$q_i = \begin{cases} 1-\varepsilon & \text{if } i = y, \\ \varepsilon/(K-1) & \text{otherwise,} \end{cases}$$

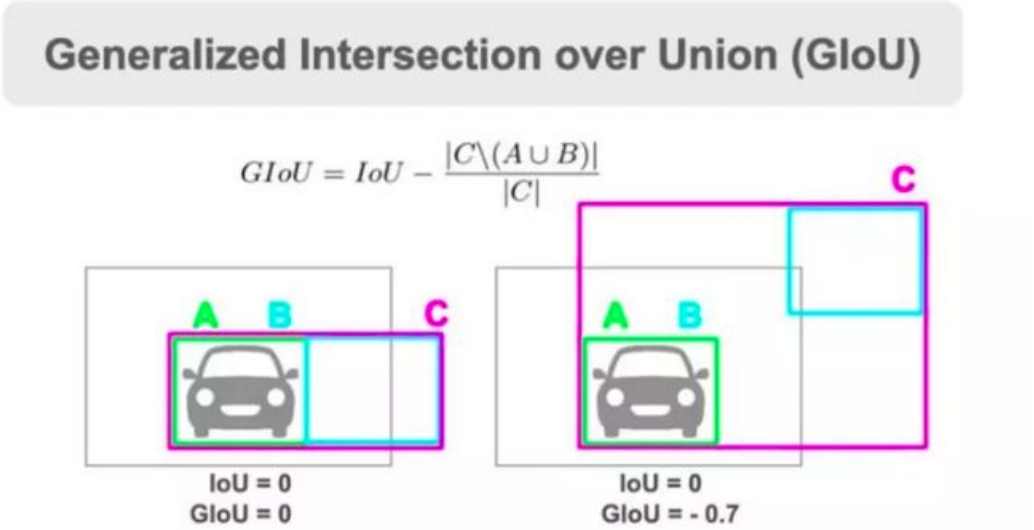
Điều này sẽ làm cho mô hình được khái quát hóa tốt hơn. Phân phối class sẽ trở thành như sau:



Objective Function of BBox Regression

- Đề xuất dùng **Anchor based approaches,** sử dụng IOU như một hàm loss, có các biến thể như : GIoU, DIoU, CIoU

GIoU (Generalized Intersection over Union): Nếu IoU của hai bbox không trùng nhau thì loss = 0 và mô hình không học được gì. Hàm này càng lớn càng tốt.



Objective Function of BBox Regression

DIoU (Distance IoU): Là một biến thể của GIoU nhưng hội tụ nhanh hơn và có đưa thêm độ lệch tâm dự đoán và thực tế để phạt thêm hàm loss này.

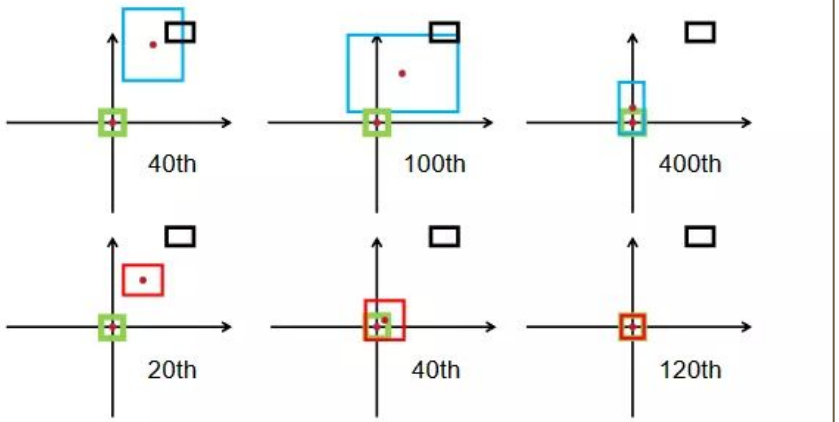


Figure 1: Bounding box regression steps by GIoU loss (first row) and DIoU loss (second row). Green and **black** denote target box and **anchor** box, respectively. Blue and red denote predicted boxes for GIoU loss and DIoU loss, respectively. GIoU loss generally increases the size of predicted box to overlap with target box, while DIoU loss directly minimizes normalized distance of central points.

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2}.$$

b, b_gt: Lần lượt là tâm của bbox dự đoán và bbox thực tế
c: Diện tích tối thiểu chứa hai bbox dự đoán và thực tế.

Objective Function of BBox Regression

CIoU (Complete IoU): Loss này quan tâm đến: Diện tích trùng lặp, Khoảng cách tâm và tỉ lệ khung .

Nhìn chung giống của DIoU, nhưng có thêm hai đại lượng alpha và v:

- **alpha**: Đánh độ thay đổi tích cực về mặt trùng lặp. Nếu IoU càng nhỏ thì alpha càng nhỏ, do đó khi IoU lớn nó sẽ đánh trọng số lớn hơn, giúp mô hình hội tụ tốt theo chiều hướng tích cực.
- **v**: Đánh giá tính nhất quán của tỉ lệ khung bbox.

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v.$$

$$\alpha = \frac{v}{(1 - IoU) + v},$$

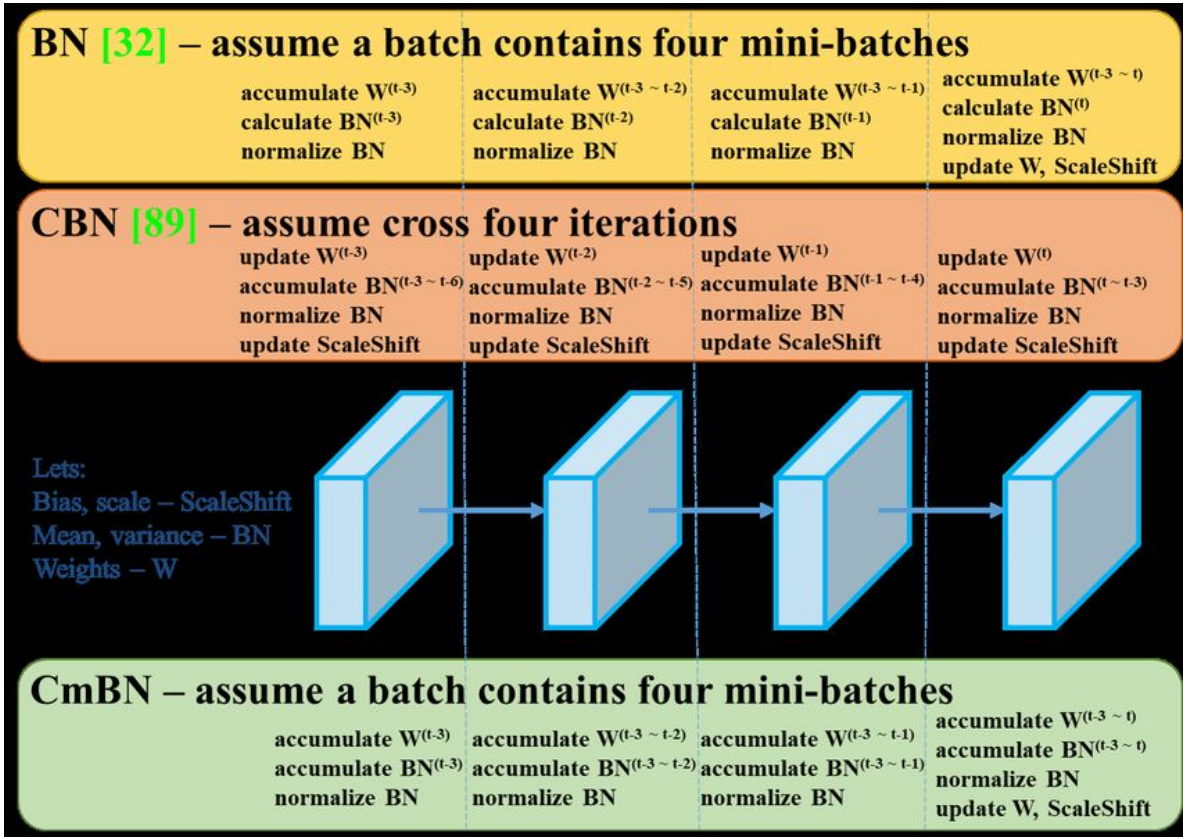
$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2.$$

Bag of Freebies (BoF)

Một số phương pháp nữa:

- Cross mini Batch Normalization
- Seft-Adversarial Training
- Multiple anchors
- Cosine annealing scheduler
- Optimal hyperparameters with Genetic Algorithm

Cross mini Batch Normalization



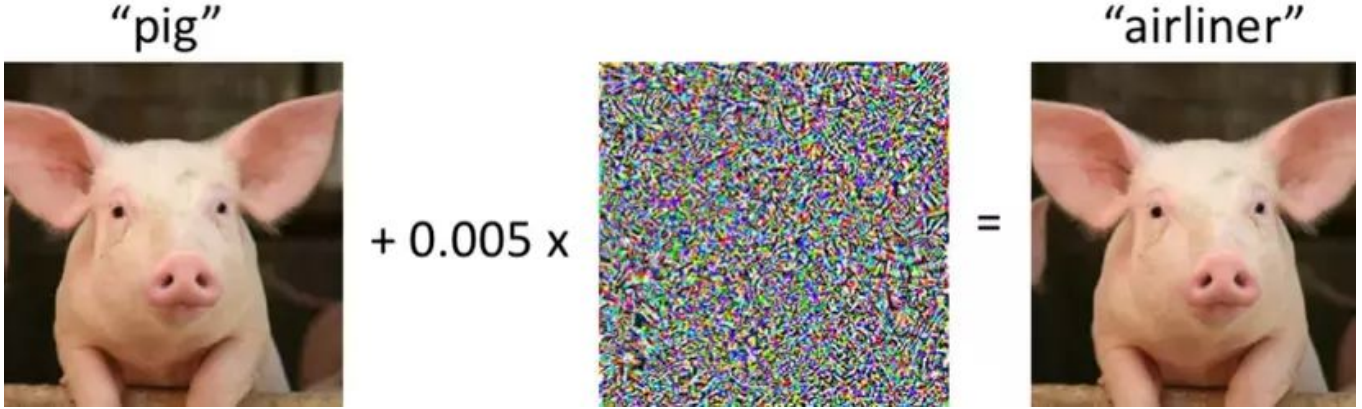
BN: Cứ mỗi batch dữ liệu vào sẽ phải tính mean và variance cho riêng từng batch. Sau khi xong 1 batch dữ liệu thì cập nhật W và ScaleShift.

CmBN: Cứ mỗi batch dữ liệu bình thường, cho đến khi đạt số K mini-batch thì nó sẽ tính mean và variance của cả K mini-batch đấy. Sau khi xong 1 batch dữ liệu thì cập nhật W và ScaleShift

Seft-Adversarial Training (SAT)

Là một phương pháp Data Augmentation mới. Thông thường ta tạo ảnh từ ảnh gốc thành các ảnh mới gây khó khăn cho network để network thực sự hiểu và phân biệt được các object khác nhau.
Nó bao gồm hai bước:

- Bước 1: Mô hình thực hiện forward bình thường, sau đó thay vì điều chỉnh weights của model, ta thay đổi inputs images để làm cho model có kết quả tồi đi theo hướng network không thể phát hiện đối tượng trong ảnh.
- Bước 2: Sử dụng các ảnh mới được tạo ra để training model như bình thường, sử dụng ground truth bbox và label của ảnh gốc.



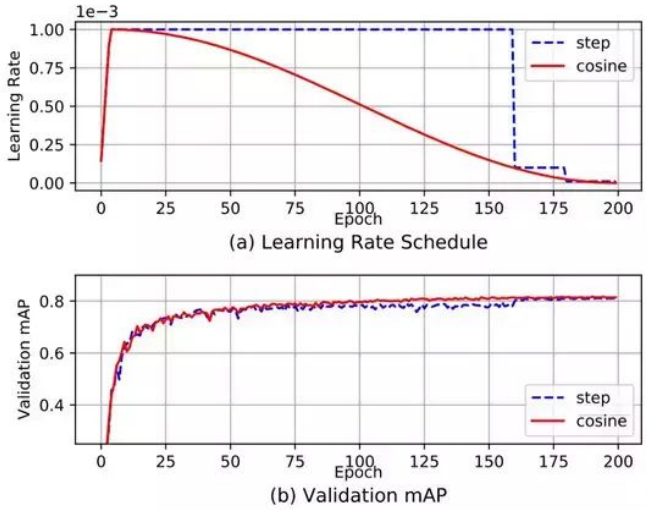
Multiple anchors

- Ở Yolov3, chỉ lấy những anchor box nào có IoU với ảnh ground truth lớn nhất . Còn Yolov4 thì giữ lại hết tất cả các anchor box nào có IOU với GT lớn hơn 0.5 là được.
  => Hiệu quả thì chưa được chứng minh

Consine anneling scheduler

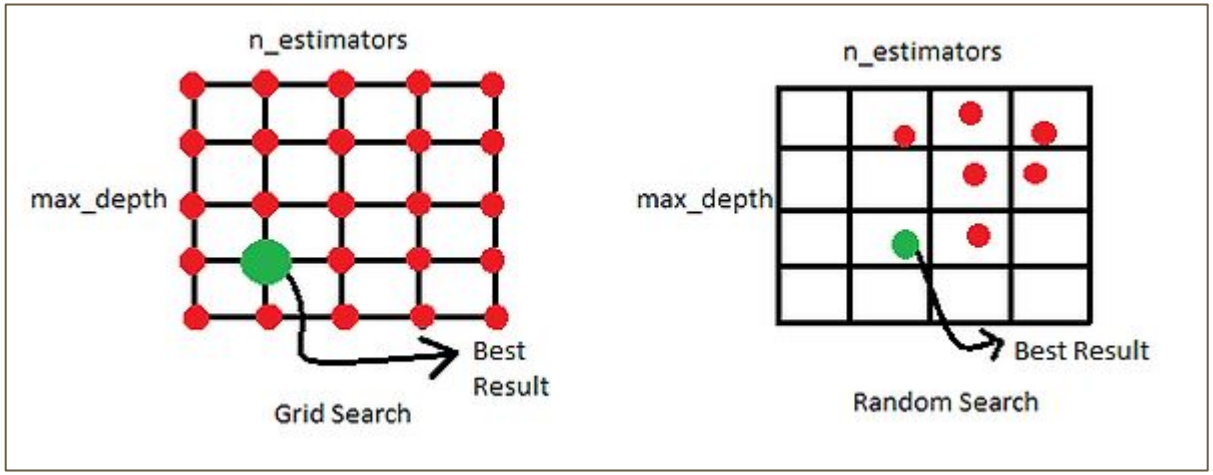
$$\eta_t = \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)\eta,$$

Yolov3 dùng Step Scheduler.
còn Yolov4 dùng Consine.
=> Hiệu quả hơn



Optimal hyperparameters with Genetic Algorithm

Thông thường sẽ sử dụng Grid Search hoặc Random Search để tìm bộ hyperparameters tuyệt vời, nhưng không gian parameter lớn, kiếm lâu và khó hiệu quả.



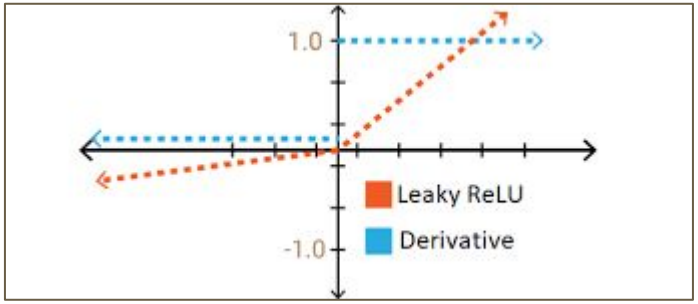
Trong yolov4: Dùng thuật toán di truyền để mã hóa các hyperparameter thành các đoạn gen, để từ đó dự đoán các xu hướng kết hợp để tạo ra hyperparameter có **xu hướng tích cực**.

Bag of Specials (BoS)

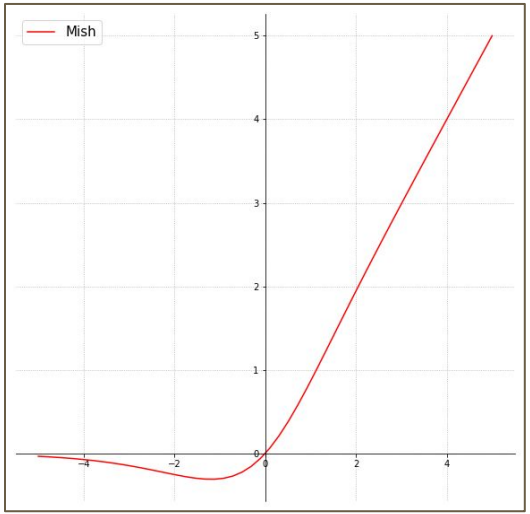
Là một tập hợp các kỹ thuật **thêm module vào mô hình** hoặc **Post-process** để **tăng thêm chi phí** một chút nhưng cải thiện độ chính xác của mô hình một cách đáng kể

- Mish activation
- CSP block
- Multi-input weighted residual connection (SAM-block, SPP-block, PAN)
- DIoU-NMS

Activation function



Yolov3 using Leaky ReLU



Yolov4 using Mish

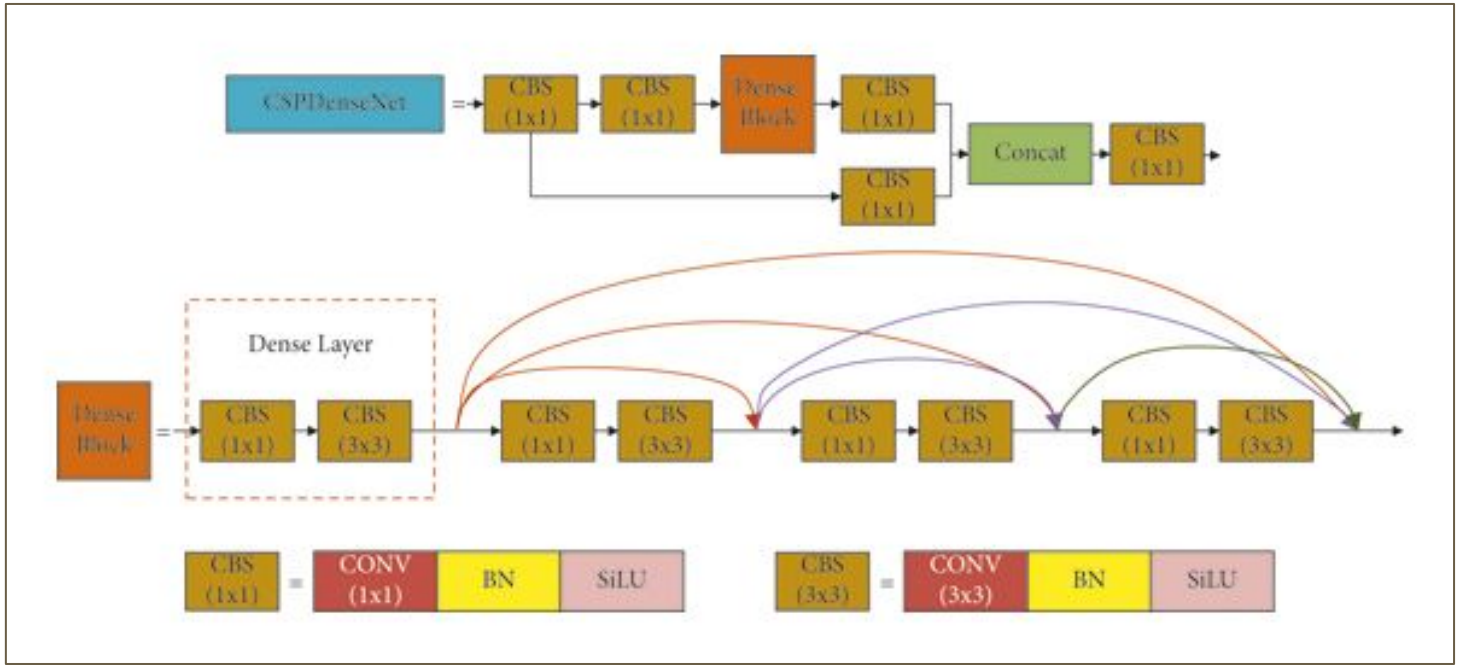
$$f(x) = \begin{cases} 0.01x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

$$z\ tanh(ln(1 + e^z))$$

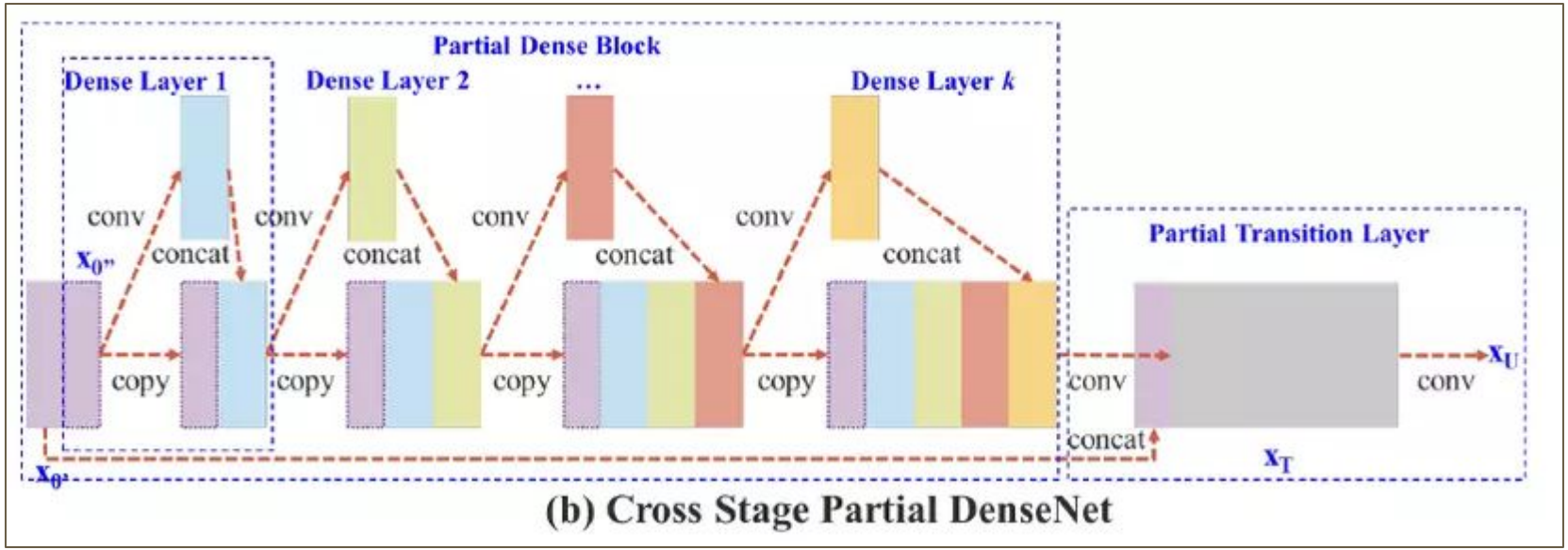
Mặc dù độ phức tạp tính toán của Mish hơn Leaky ReLu nhiêu, nhưng bù lại độ hiệu quả đã được chứng minh tốt cho bài toán Deep Learning

CSP block

Chức năng: Tạo ra hai luồng gradient và feature map độc lập, có ý nghĩa về việc sử dụng lại các feature và ngăn chặn lượng thông tin trùng lặp. Do đó tăng tốc quá trình training và giảm số lượng parameter => Model inference nhanh hơn.



CSP block


Partial Dense Block
Dense Layer 1
Dense Layer 2
...
Dense Layer k
conv
concat
copy
Partial Transition Layer
(b) Cross Stage Partial DenseNet

Multi-input weighted residual connection

Là kỹ thuật kết hợp thông tin từ nhiều tầng khác nhau của mạng và tạo ra một đầu ra kết hợp có khả năng tổng quát hóa cao. Giúp cải thiện khả năng trích rút của mô hình.
Có các khối hay dùng:

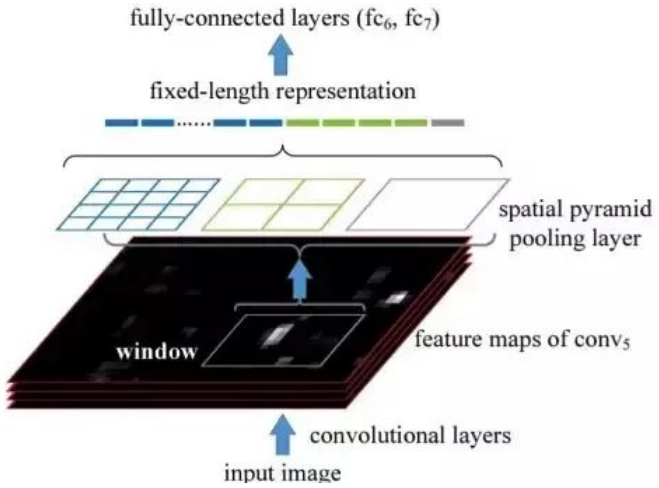
- SPP (Spatial Pyramid Pooling) Block
- PAN (Path Aggregation Network) Block
- SAM (Spatial Attention Module) Block

SPP Block

Thường thì các lớp cuối của classification, thì output feature map sẽ được flatten để làm đầu vào cho các lớp fully connected ở cuối. Tuy nhiên việc này phải làm cố định size của hidden layer gây khó khăn khi trong việc detect objects ở các kích thước khác nhau.

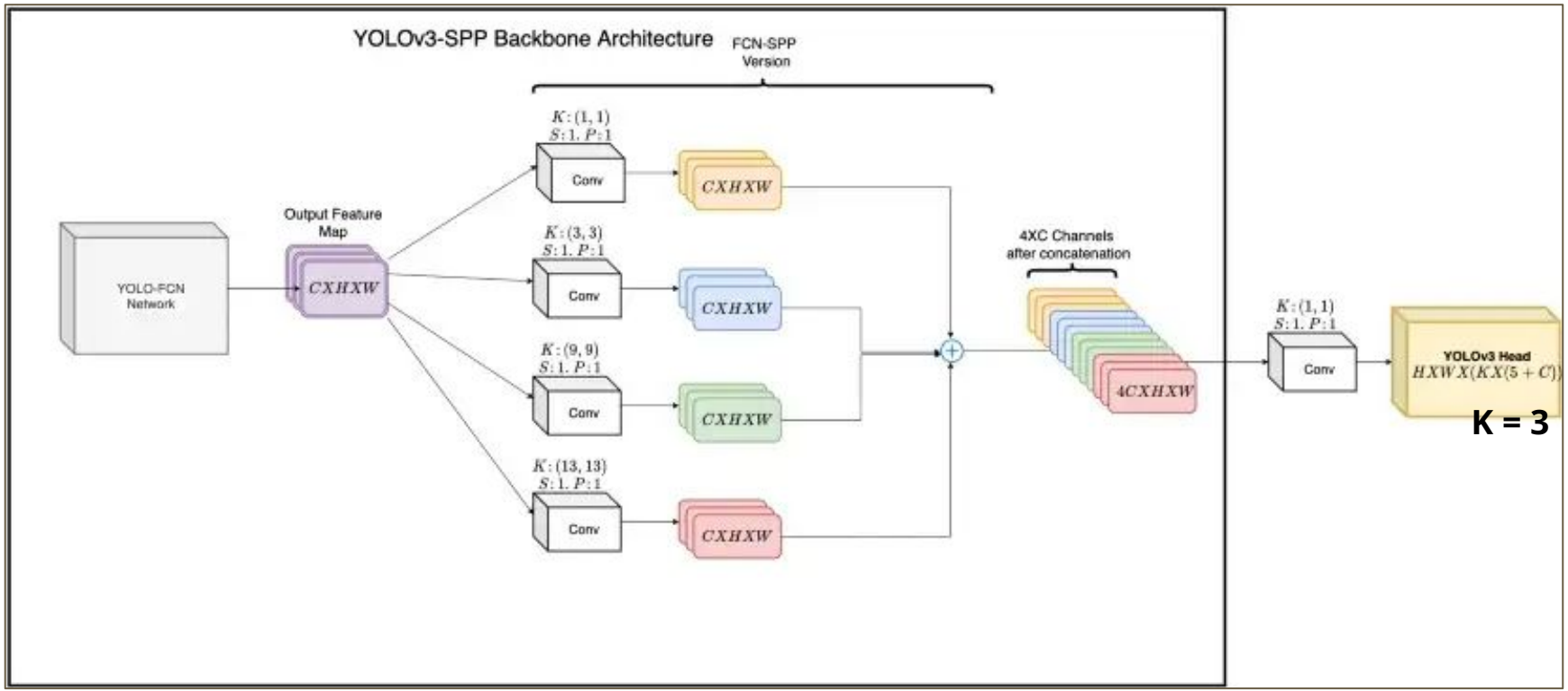
Để khắc phục điều này. Cho feature map đi qua các 3 lớp pooling có kích thước khác nhau: 1x1, 2x2, 4x4.

Ví dụ: Feature map là WxHxC = 80x80x512
Sau khi ba lớp trên thu được lần lượt: 512, 512x4, 512x16 những vector 1D, tiến hành concatenation lại.



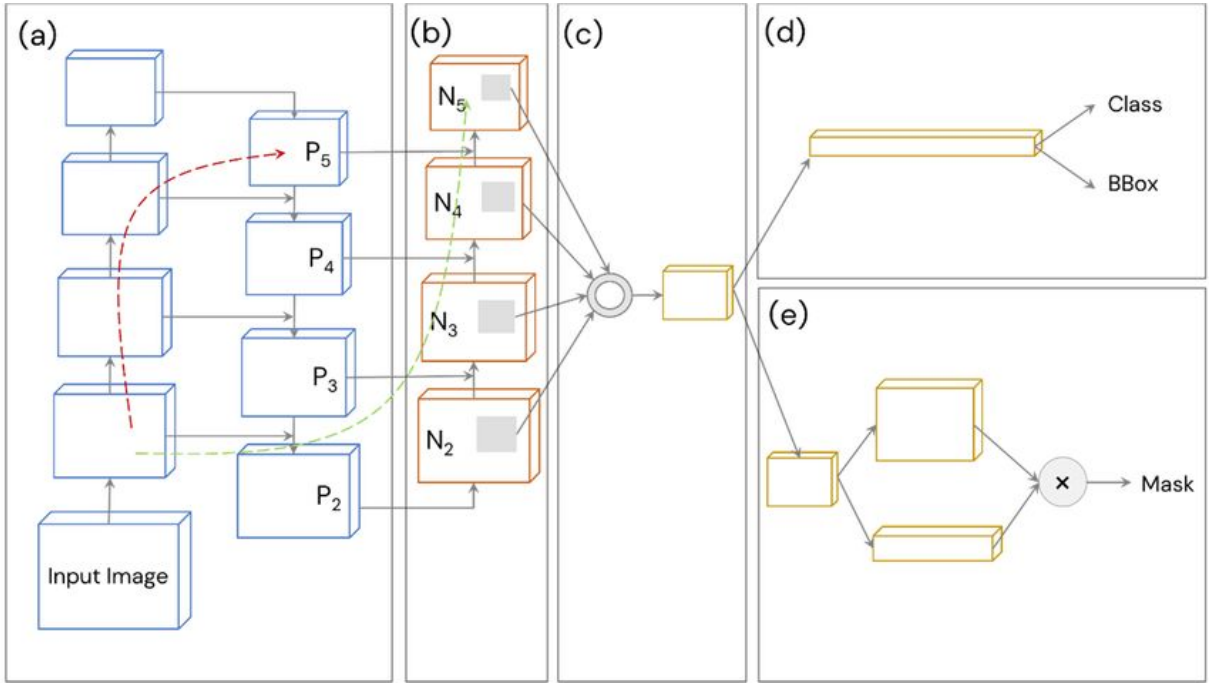
SPP Block

Giúp mô hình nắm bắt thông tin từ các vùng ảnh có kích thước và vị trí khác nhau, cải thiện hiệu suất phát hiện đối tượng trong ảnh.



PAN Block

Là phiên bản cải tiến của FPN. Nhìn chung giống FPN, tuy nhiên PAN có thêm một bottom-up path để những localization features từ những lớp thấp có thể kết hợp với những features từ các lớp cao của P5 (Dùng shortcut path) để tạo ra N5.



Do đó toàn bộ những sự thay đổi của feature máp có thể được sử dụng trong quá trình predict.

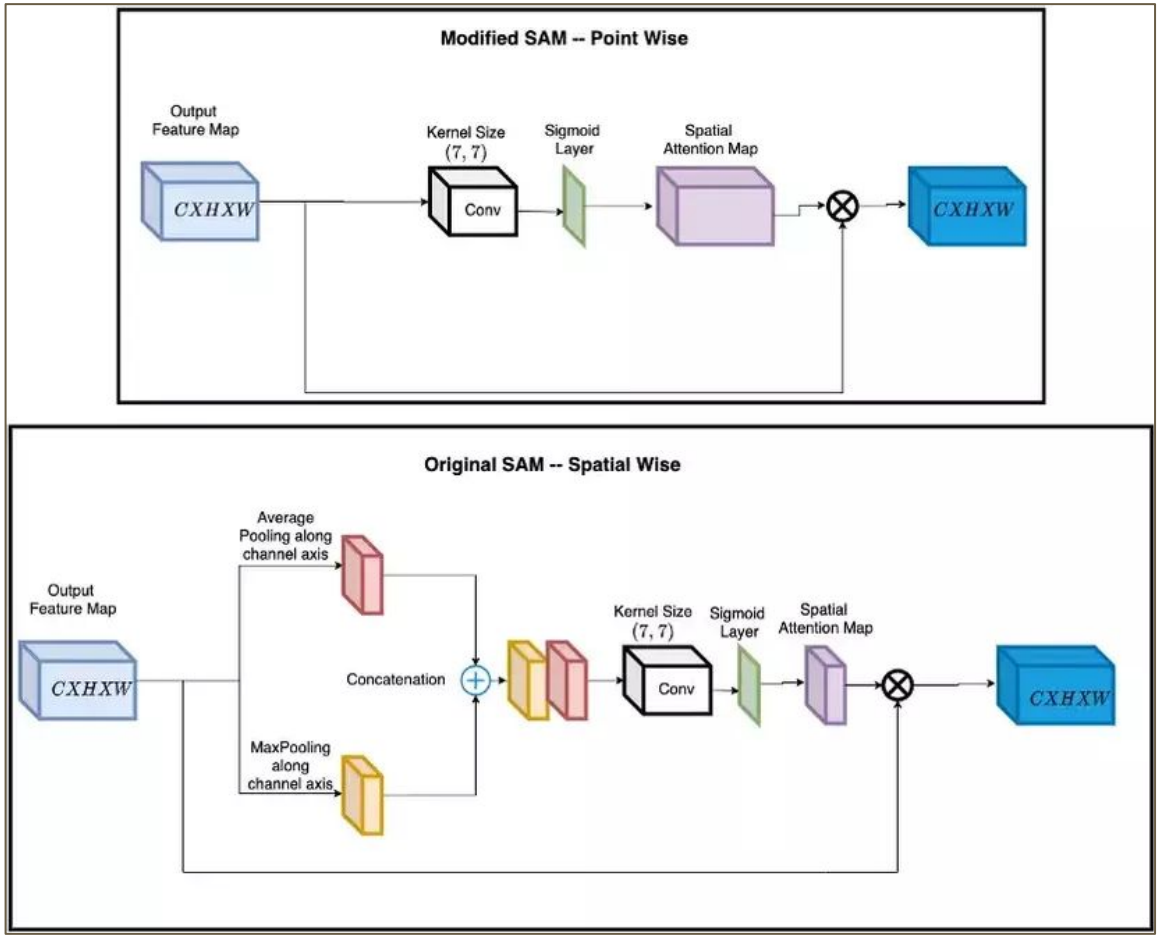
Chỗ vị trí N5 có hai hướng:

- Cộng lại
- Concatenation

SAM Block

- SAM thường được dùng trong mạng CNN để làm mô hình tập trung vào những objects có trong ảnh hơn là tập trung tổng thể.

- Trong mô hình YOLOv4, tác giả đã tinh chỉnh lại để những thông tin nào trong không có giá trị hoặc vai trò trong việc classification/detection sẽ bị loại bỏ bằng phép element-wise.



DIoU NMS

Có hai thay đổi so với NMS thông thường:

- NMS cho từng class có độ chồng lấn cao nhất.

- Thay vì dùng IOU thì dùng DIoU là thang đo

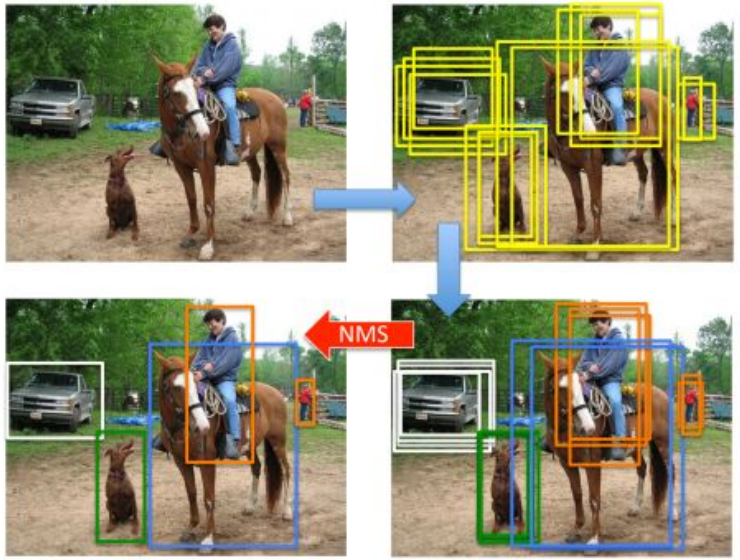
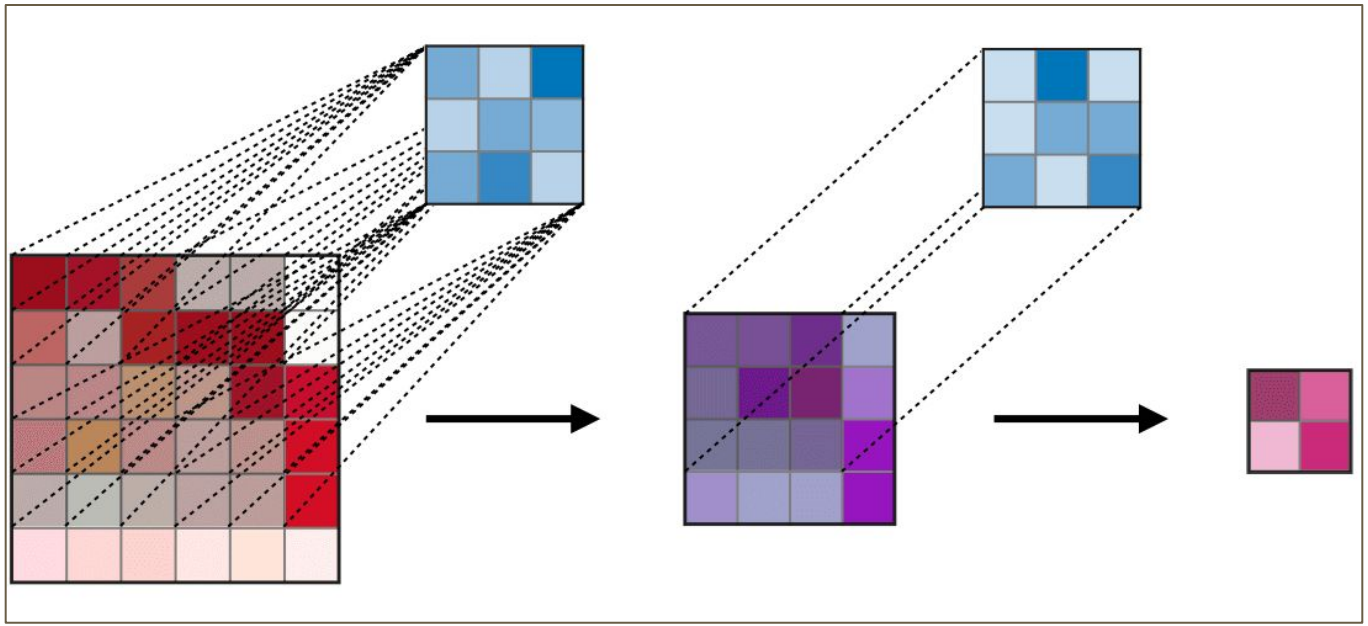


Figure 3. In object detection, first category independent region proposals are generated. These region proposals are then assigned a score for each class label using a classification network and their positions are updated slightly using a regression network. Finally, non-maximum-suppression is applied to obtain detections.

Receptive field

Receptive Field không phải là một lớp cụ thể nào mà là một khái niệm.
Là khái niệm giúp hiểu về phạm vi và thông tin ngữ cảnh mà một neuron có thể nhận biết từ ảnh đầu vào.



Receptive field

Ta có receptive field tại layer k được ký hiệu $R_k \times R_k$. Gọi F_j là kích thước của kernel của layer thứ *j* và S_i là giá trị của stride của layer thứ *I*, ta sẽ tạm để $S_0 = 1$. Từ đó kích thước receptive field tại layer *k* được tính bằng công thức:

$$R_k = 1 + \sum_{j=1}^{k} (F_j - 1) \prod_{i=0}^{j-1} S_i$$

Ở ví dụ phía dưới chúng ta có, $F_1 = F_2 = 3$ và $S_1 = S_2 = 1$, nên cho ra được $R_2 = 1 + 2.1 + 2.1 = 5$.

=> Khi tăng số lượng layer lên thì kích thước receptive field sẽ tăng lên.

Overview

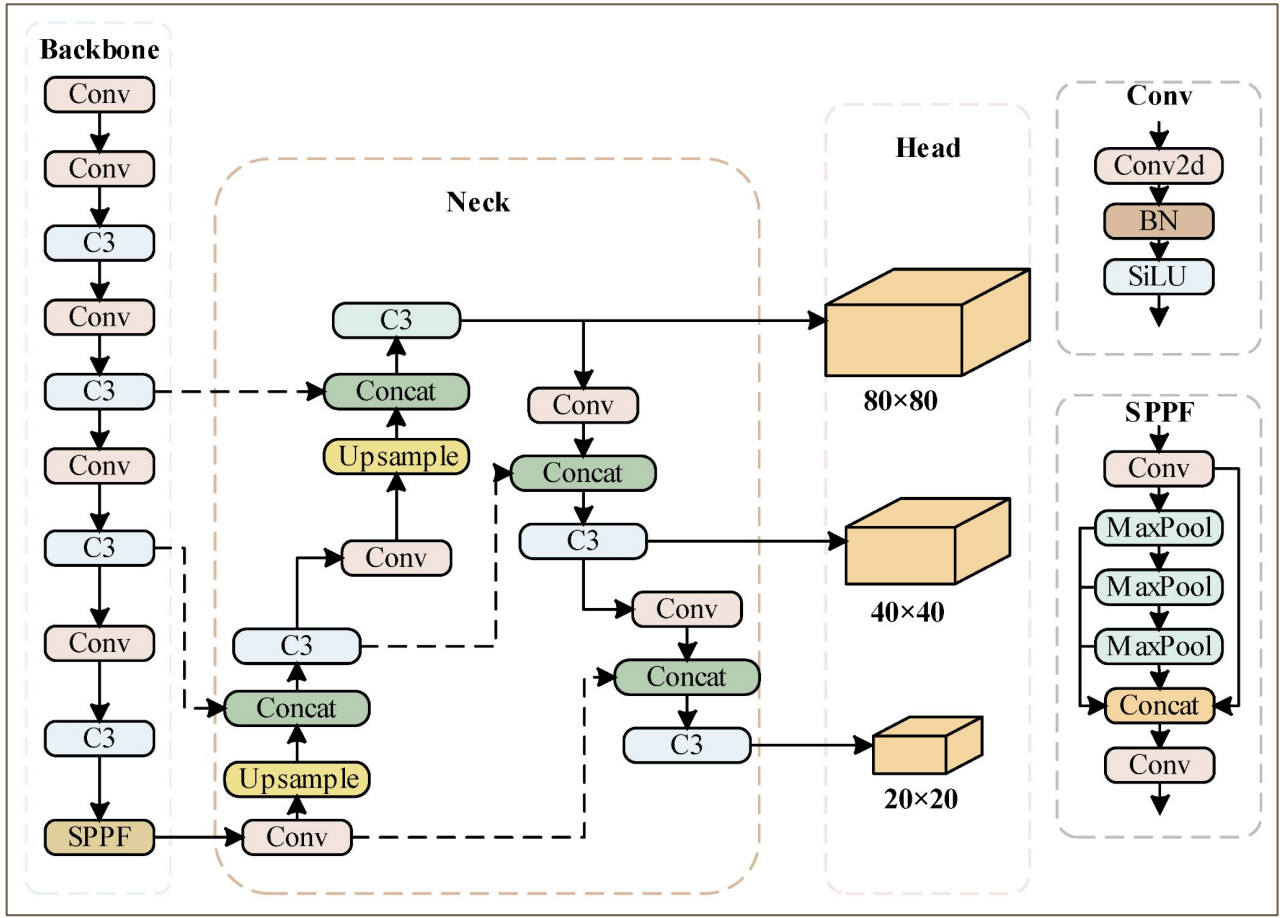
paper: https://arxiv.org/pdf/2104.13634.pdf
or https://arxiv.org/pdf/2203.16506.pdf
Year: 06.2021

YOLOv5:

Improvement:

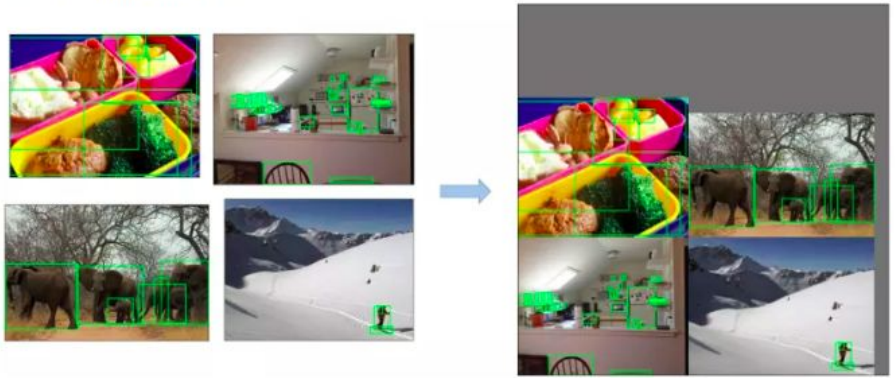
- Tập trung vào tối ưu thời gian model

Architecture



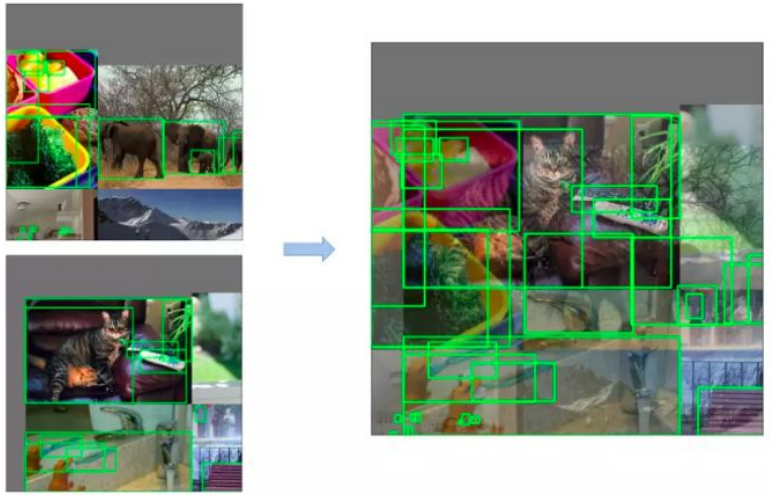
Data Augmentation

- Mosaic Augmentation

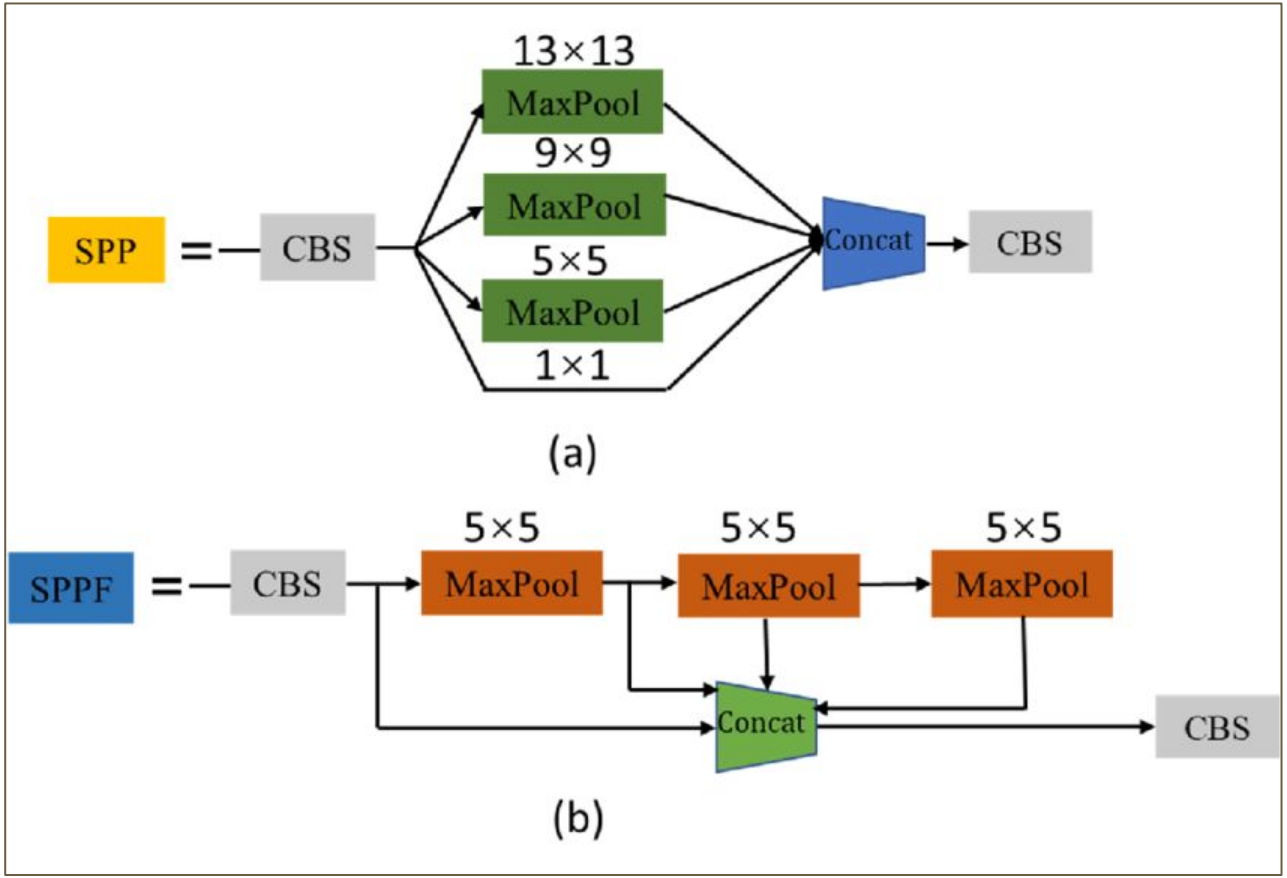


- Copy-paste Augmentation

- MixUp Augmentation



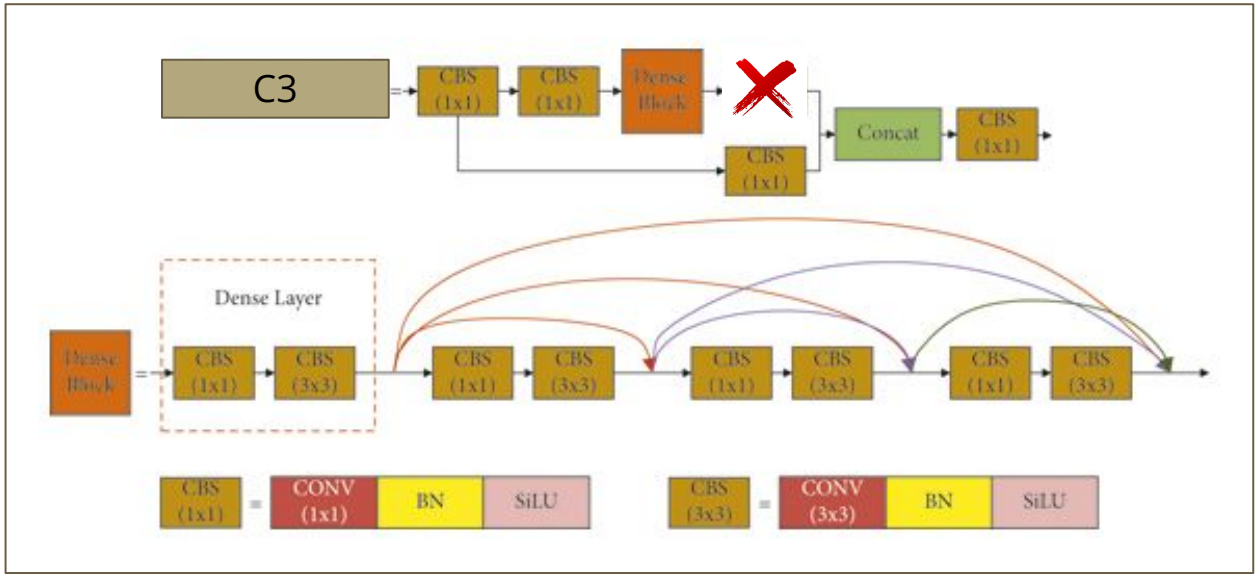
SPPF (SPP - Fast)



Tốc độ nhanh nhơn SPP gấp đôi

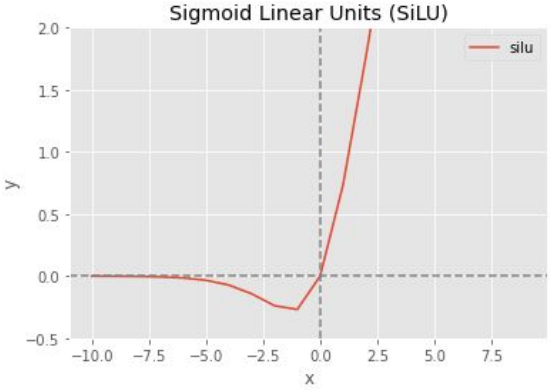
Chức năng như SPP của yolov4

C3 Block and Activation Function



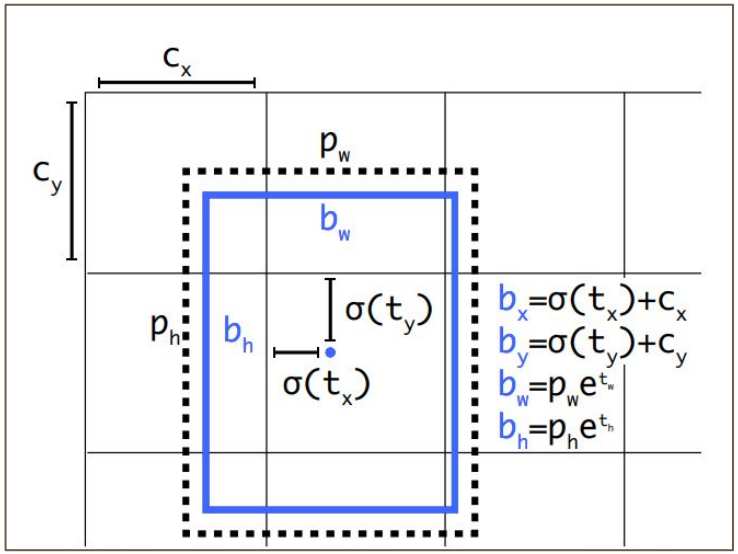
Đơn giản là nó bỏ đi một khối **CBS 1x1.**

$$SiLU(x) = x\left(\frac{1}{1+e^{-x}}\right)$$



Yolov5: chỉ dùng activation function là **SiLU,** thay vì là Mish hoặc LeakyReLu

Location Prediction



Yolov3

$$b_x = 2\sigma(t_x) - 0.5 + c_x$$
$$b_y = 2\sigma(t_y) - 0.5 + c_y$$
$$b_w = p_w(2\sigma(t_w))^2$$
$$b_h = p_h(2\sigma(t_h))^2$$

Yolov5

Loss function

Yolov5 có 3 ngõ ra, với độ phân giải tương ứng : 80x80, 40x40, 20x20.

$$L_{obj} = 4.0 \cdot L_{obj}^{small} + 1.0 \cdot L_{obj}^{medium} + 0.4 \cdot L_{obj}^{large}$$

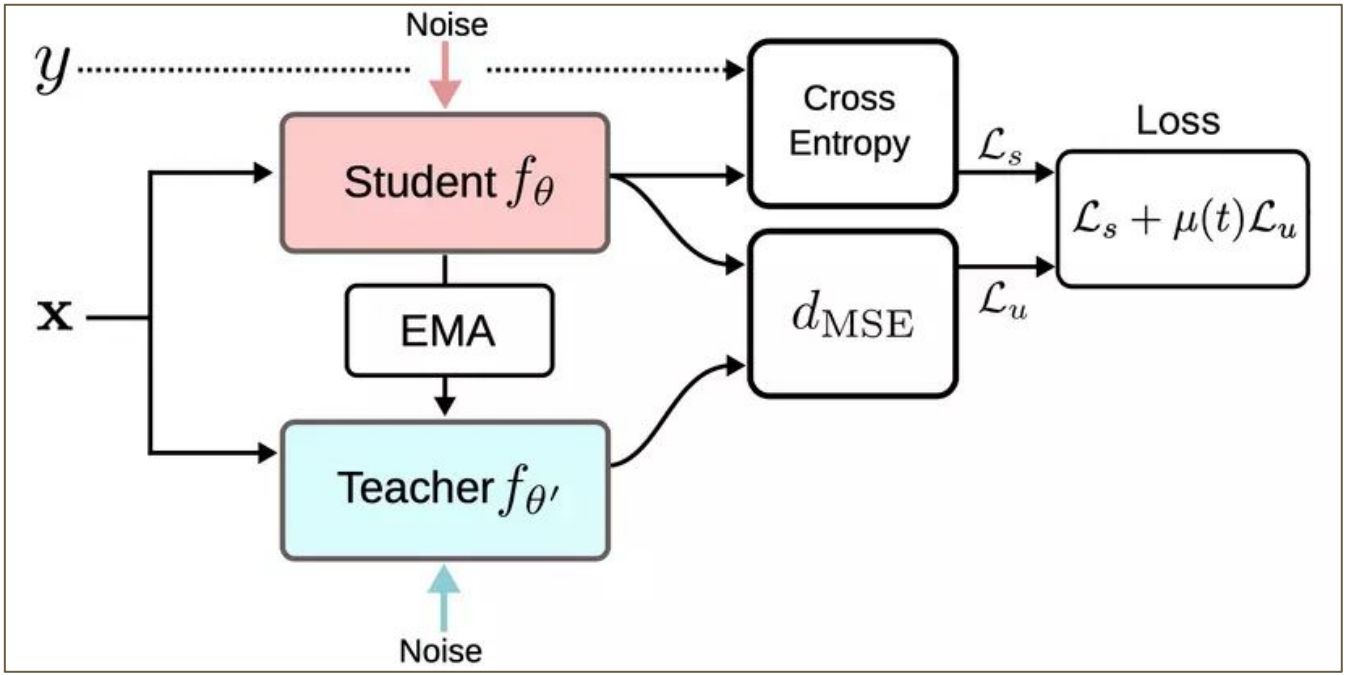
Anchor box

Thay vì chỉ dùng **K-mean** như Yolov2, thì yolov5 còn kết hợp sau đó là **giải thuật di truyền** vào **Anchor Box** để **Anchor Box** hoạt động tốt hơn với những custom dataset của người dùng.

Giải thuật di truyền:
https://viblo.asia/p/giai-thuat-di-truyen-va-ung-dung-trong-yolov5-6J3Zgyrg5mB#_tien-hoa-anchors-9

EMA Weight

- Khi dùng EMA thì sẽ tạo ra một model giống y hệt model đang dùng training, nhưng cách cập nhật weight sẽ tuân theo công thức **Exponential Moving Average (EMA)**.
- Một kiểu của Semi-supervised Learning



L_s: Loss Supervised
L_u: Loss Unsupervised

What changes ?

Backbone: CSPResidualBlock -> C3 Module
Neck: SPP + PAN – > SPPF + PAN
Head: Giữ nguyên của Yolov3

Augmentation: Thêm Mosaic, Copy-paste Augmentation, Mixup Augmentation.

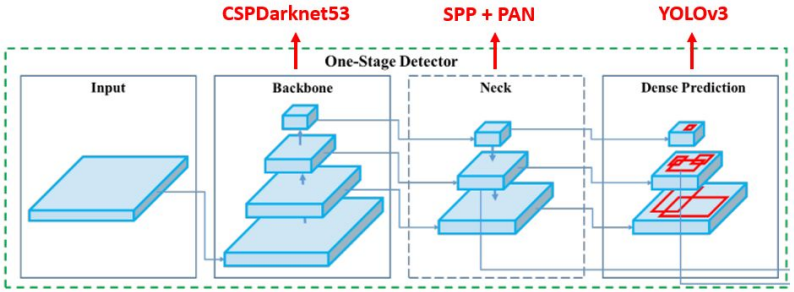
Loss Function: Thêm hệ số scale cho objectness Loss

Anchor Box: Auto Anchor Sử dụng GA

Sử dụng công thức Location Prediction khác

EMA Weight

Summarize



YOLOv4 = YOLOv3 + CSPDarknet53 + SPP + PAN + BoF + BoS

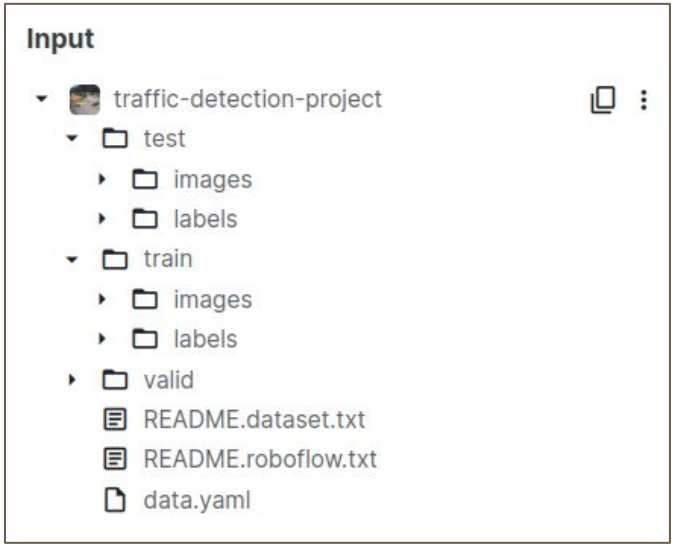
Path Aggregation Network

Backbone: Thường dùng để trích xuất đặc trưng từ ảnh gốc, Các đặc trưng này có khả năng phân biệt được đối tượng này với đối tượng khác.

Neck: Neck là phần tiếp theo trong mạng là một tập hợp các convolutional và pooling để kết hợp các thông tin của Backbone và là nó phức tạp hơn. Giúp phát hiện được các đối tượng có kích thước và hình dạng khác nhau.

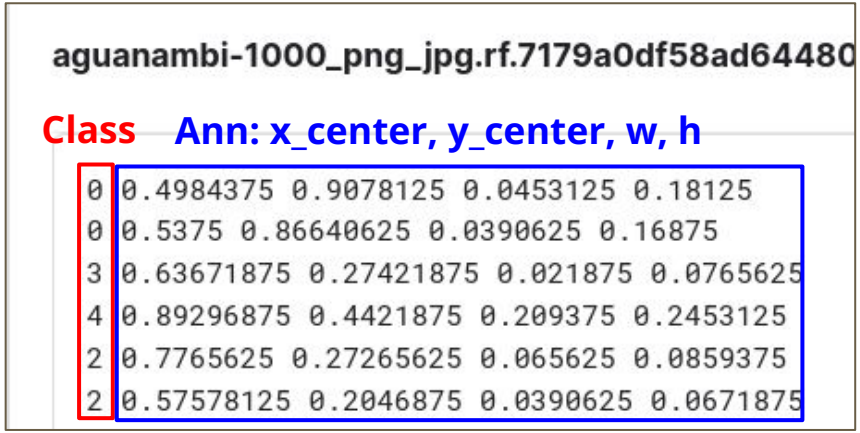
Head: Là phần cuối cùng của mô hình one-stage, dùng để thực hiện việc dự đoán vị trí và xác định đối tượng trong ảnh. Dùng tạo ra bbox và phân loại class của đối tượng

Annotator

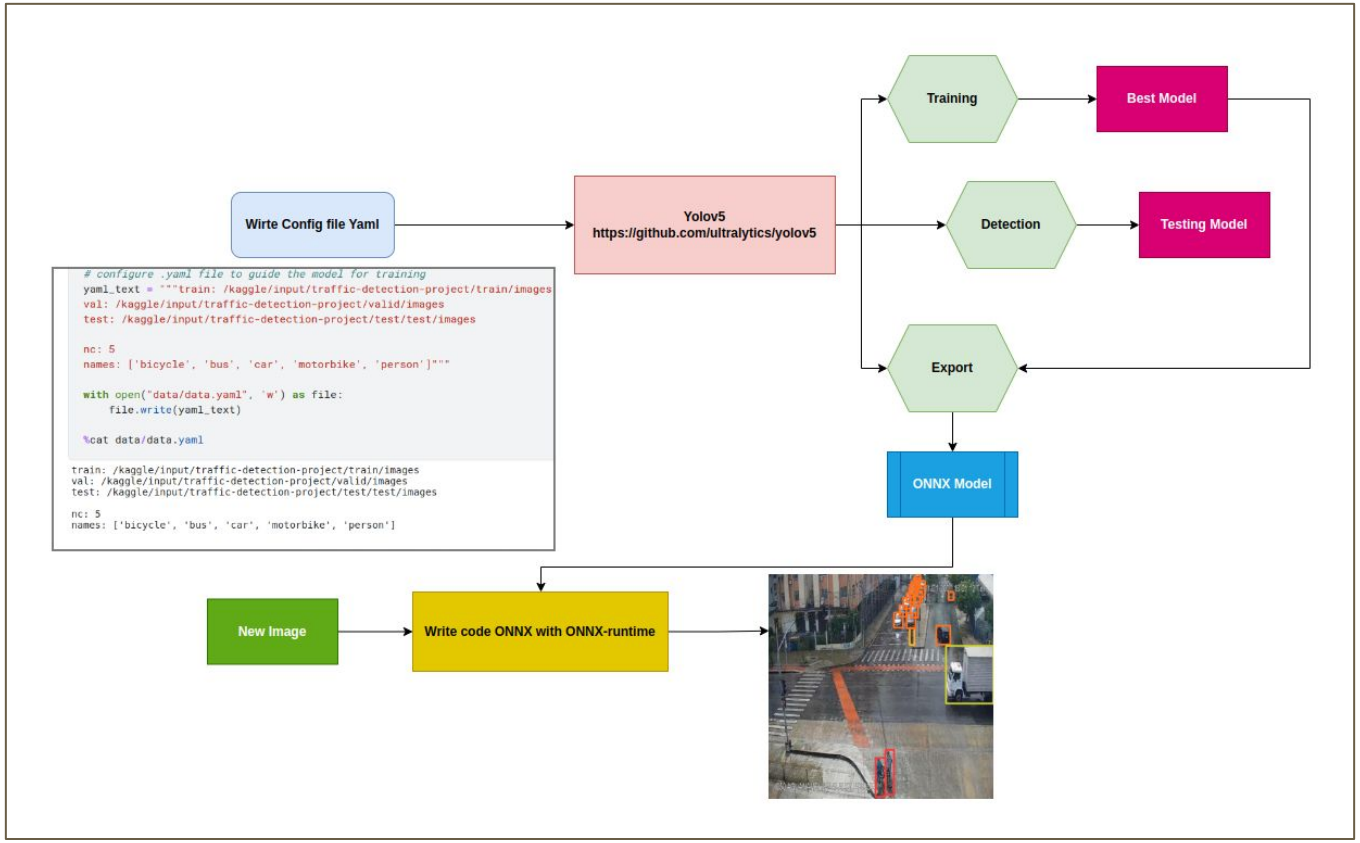


Cấu trúc thư mục data

Trong thư mục 'images' nhiều ảnh thì trong thư mục 'labels' cũng phải có tên giống vậy.



Pipeline with Yolov5

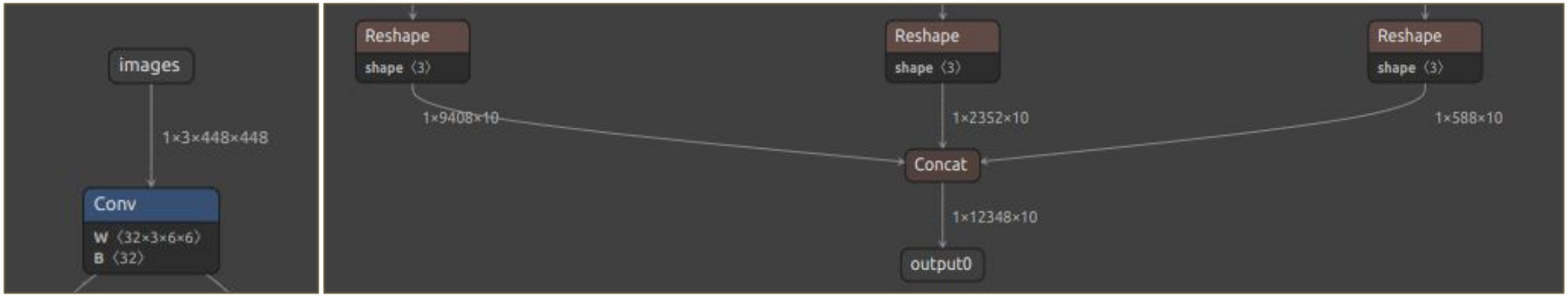



```
# configure .yaml file to guide the model for training
yaml_text = """train: /kaggle/input/traffic-detection-project/train/images
val: /kaggle/input/traffic-detection-project/valid/images
test: /kaggle/input/traffic-detection-project/test/test/images

nc: 5
names: ['bicycle', 'bus', 'car', 'motorbike', 'person']"""

with open("data/data.yaml", 'w') as file:
    file.write(yaml_text)

%cat data/data.yaml
```

```
train: /kaggle/input/traffic-detection-project/train/images
val: /kaggle/input/traffic-detection-project/valid/images
test: /kaggle/input/traffic-detection-project/test/test/images

nc: 5
names: ['bicycle', 'bus', 'car', 'motorbike', 'person']
```

ONNX with Yolov5

Check Model: https://netron.app/



Kiểm tra input và output của mô hình