

BÁO CÁO ĐỒ ÁN THỊ GIÁC MÁY TÍNH

OBJECT DETECTION

Nguyễn Xuân Anh Quân - 1712135
Email: 1712135@student.hcmus.edu.vn

Lâm Đức Anh - 1712273
Email: 1712273@student.hcmus.edu.vn

Tóm tắt nội dung—Trong báo cáo về đồ án này, nhóm em sẽ trình bày tổng quan về các mô hình Object Detection chính hiện nay bao gồm các mô hình one-stage và two-stage. Sau đó chọn một mô hình để nghiên cứu sâu hơn là YOLO phiên bản 3, bao gồm trình bày về lý thuyết, kiến trúc mô hình, các nâng cấp qua từng phiên bản của YOLO và kết quả thử nghiệm trên pretrain model và custom model trên một tập dữ liệu.

I. GIỚI THIỆU

Object Detection là một bài toán quan trọng trong lĩnh vực Computer Vision, Object Detection đề cập đến khả năng của hệ thống máy tính và phần mềm để định vị các đối tượng trong một hình ảnh và xác định từng đối tượng. Object Detection đã được sử dụng rộng rãi để phát hiện khuôn mặt, phát hiện xe, đếm số người đi bộ, hệ thống bảo mật và xe không người lái. Các mô hình Object Detection hiện nay được chia thành 2 nhóm chính là one-stage và two-stage.

A. Two-Stage

Các thuật toán two-stage object detection điển hình như R-CNN, Fast R-CNN, Faster R-CNN, Mask-RCNN,... Ví với Faster R-CNN thì trong stage-1 model sẽ sử dụng RPN (Region Proposal Network) để sinh ra các box có khả năng chứa đối tượng, sau đó trong stage-2 sẽ thực hiện tiếp việc phân loại đối tượng và xác định vị trí.

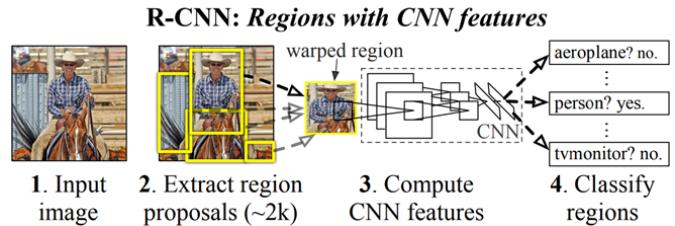
B. One-Stage

Các thuật toán one-stage object detection điển hình như YOLO v2/3/4 (YOLO v4 mới được đề xuất gần đây), SSD, Retina-Net. Gọi là one-stage vì trong kiến trúc model hoàn toàn không có phần trích chọn các vùng đặc trưng (box) có khả năng chứa đối tượng như RPN của các model two-stage. Các model one-stage coi việc phát hiện đối tượng (object localization) như một bài toán regression với 4 tọa độ offset (x, y, w, h). Các model dạng này thường nhanh hơn tuy nhiên độ chính xác của model thường kém hơn so với các model two-stage.

II. CÁC MÔ HÌNH TWO-STAGE

A. R-CNN (2014)

- R-CNN được giới thiệu lần đầu năm 2014 bởi Ross Girshick và các cộng sự ở UC Berkeley trong bài báo “Rich feature hierarchies for accurate object detection and semantic segmentation”. Đây là một trong những nghiên cứu nền móng đầu tiên của mạng CNN đối với bài toán Object Detection. Mô hình đã đạt kết quả tốt nhất trên



Hình 1: Kiến trúc mạng R-CNN lấy từ bài báo gốc.

bộ dữ liệu VOC-2012 và bộ dữ liệu phát hiện đối tượng ILSVRC-2013 gồm 200 lớp.

- Ý tưởng kiến trúc của R-CNN bao gồm các phần như sau:

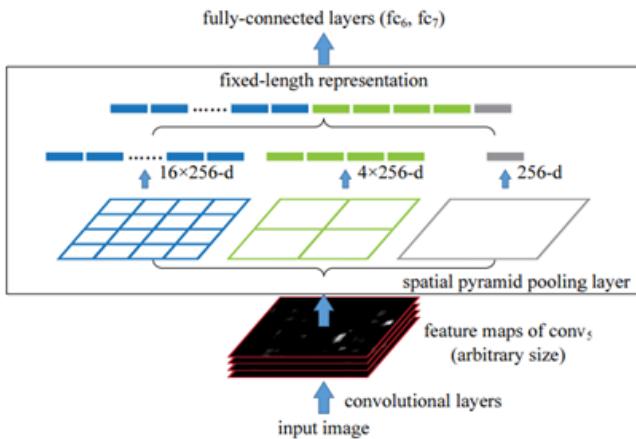
- Đề xuất vùng có khả năng chứa đối tượng (Extract Region Proposals): Dùng Selective Search Algorithm để lấy ra khoảng 2000 bounding box có khả năng chứa đối tượng.
- Trích chọn đặc trưng (Compute CNN features): Trích chọn đặc trưng để phân loại hình ảnh từ các region proposal nhờ các lớp mạng CNN
- Phân loại (Classify regions): Dựa vào input là các features ở phần trước để phân loại với mỗi bounding box xác định xem nó là đối tượng nào. (giống với task Image Classification).

- Nhược điểm của R-CNN:

- Một nhược điểm của mô hình này là chậm, đòi hỏi phải vượt qua nhiều module độc lập (như hình trên).
- Với mỗi ảnh ta cần phân loại các class cho 2000 region proposal nên thời gian huấn luyện rất lâu, do đó không thể áp dụng cho real time.

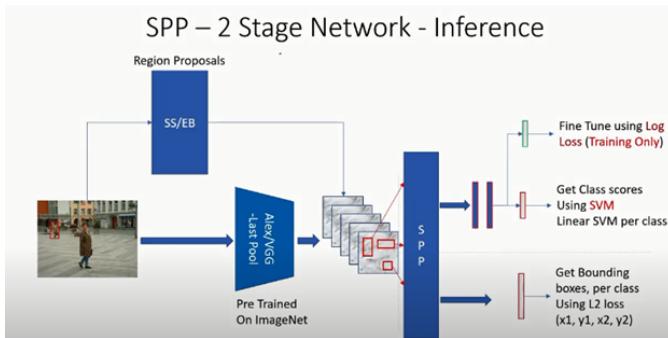
B. SPPNet (2014)

- Cùng năm 2014, một mô hình được đề xuất SPPNet trong bài báo “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”.
- Trong ILSVRC 2014, SPPNet đạt được rank 2 với Object Detection.
- Trong các mô hình CNN trước đó, trước khi chuyển qua FC layer, thường được cho qua single pooling layer. Trong SPPNet, được đề xuất nhiều pooling layer với các scales khác nhau (multiple pooling layers with different scales). Hơn nữa, nó cho phép CNN tạo ra các biểu diễn cố định bất kể kích thước region proposal khác nhau và chỉ cho

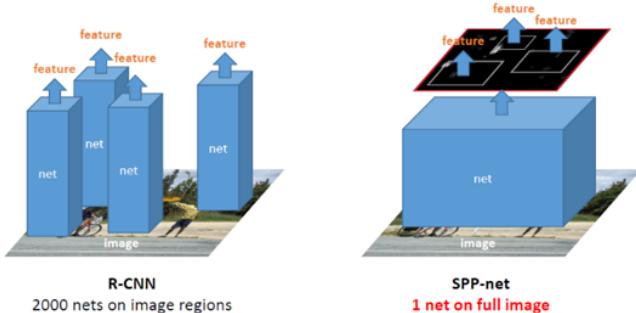


Hình 2: Kiến trúc mạng với Spatial Pyramid Pooling Layer.

qua convnet một lần. Do đó, nó nhanh hơn nhiều lần so với R-CNN.



Hình 3: SPPNet cho Object Detection.



Hình 4: So sánh R-CNN và SPP-Net.

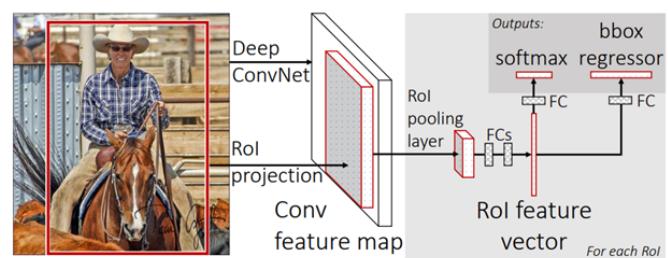
- SPPNet cho thấy đã cải thiện tốc độ so với R-CNN nhưng vẫn còn nhược điểm là mô hình trải qua nhiều giai đoạn.

C. Fast R-CNN (2015)

- Dựa trên thành công của R-CNN, Ross Girshick đề xuất một mở rộng để giải quyết các vấn đề của R-CNN (được nêu ra ở trên) để cải thiện tốc độ trong bài báo “Fast R-CNN” năm 2015.

- Điểm đặc phá của Fast R-CNN là sử dụng một single model thay vì pipeline để phát hiện region và classification cùng lúc. Vẫn giống như R-CNN thì Fast R-CNN vẫn dùng selective search algorithm để lấy ra các region proposal. Tuy nhiên nó không tách 2000 region proposal ra khỏi ảnh và thực hiện image classification cho mỗi ảnh. Fast R-CNN cho cả bức ảnh vào các lớp mạng CNN để tạo ra convolutional feature map.
- Sau đó các vùng region proposal được lấy ra tương ứng từ convolutional feature map. Tiếp đó được Flatten và cho qua 2 Fully connected layer (FC). Cuối cùng mô hình chia thành 2 đầu ra, một đầu ra cho dự đoán nhãn thông qua một softmax layer và một đầu ra khác dự đoán bounding box (bbox) dựa trên hồi quy tuyến tính. Quá trình này sau đó được lặp lại nhiều lần cho mỗi vùng ROI trong một hình ảnh.

- Về Region of Interest pooling: Để chuyển các region proposal trong feature map về cùng một kích thước rồi sau đó được flatten để cho qua các fully connected layer.

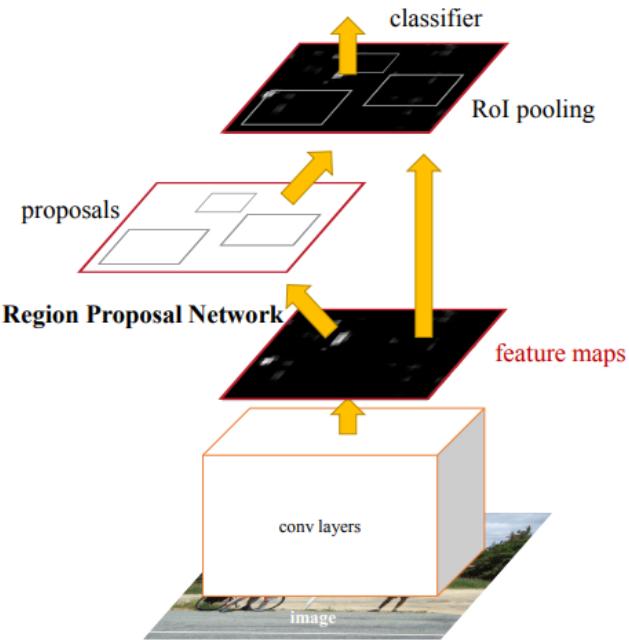


Hình 5: Kiến trúc của Fast R-CNN được lấy từ bài báo gốc.

- Fast R-CNN khác với R-CNN là nó thực hiện lấy feature map với cả ảnh sau đó lấy các region proposal ra từ feature map, còn R-CNN thực hiện tách các region proposal ra rồi mới thực hiện CNN trên từng region proposal. Do đó Fast R-CNN nhanh hơn R-CNN về huấn luyện lẫn dự đoán.
- Tuy nhiên thời gian để tính region proposal lâu và làm chậm thuật toán nên người ta nghĩ đến việc thay thế selective search và Faster R-CNN ra đời.

D. Faster R-CNN (2016)

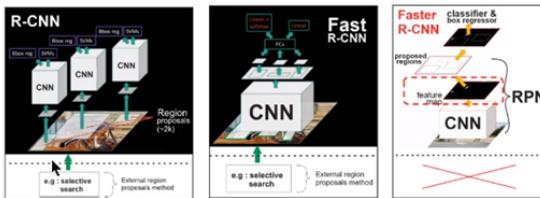
- Kiến trúc được cải thiện hơn như cái tên của nó về tốc độ huấn luyện và phát hiện được đề xuất bởi Shaoqing Ren và các cộng sự tại Microsoft Research trong bài báo năm 2016 “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”.
- Faster R-CNN không dùng thuật toán selective search để lấy ra các region proposal, mà nó thêm một mạng CNN mới gọi là Region Proposal Network (RPN) để tìm các region proposal.
- Đầu tiên cả bức ảnh được cho qua ConvNet để lấy feature map. Sau đó feature map được dùng cho Region Proposal Network (RPN) để lấy được các region proposal. Sau khi



Hình 6: Kiến trúc Faster R-CNN được lấy từ bài báo gốc.

lấy được vị trí các region proposal thì thực hiện tương tự Fast R-CNN.

- Tại thời điểm bài báo được đề xuất, kiến trúc Faster R-CNN này là đỉnh cao của họ model R-CNN và đạt được kết quả gần như tốt nhất trong các nhiệm vụ nhận dạng đối tượng, vì vậy có thể dùng cho real time object detection.



	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

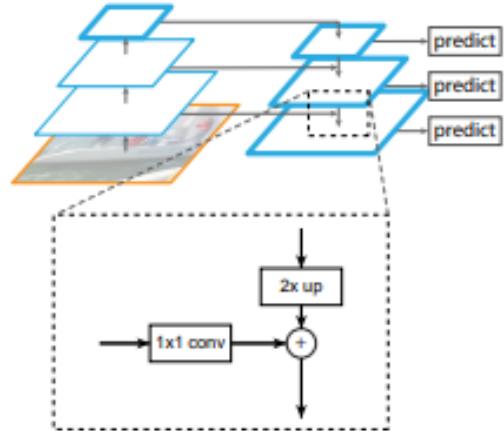
Hình 7: So sánh R-CNN, Fast R-CNN, Faster R-CNN.

- Một mô hình mở rộng hỗ trợ cho phân đoạn ảnh (Image Segmentation), được mô tả trong bài báo năm 2017 là “Mask R-CNN”.

E. Feature Pyramid Networks (2017)

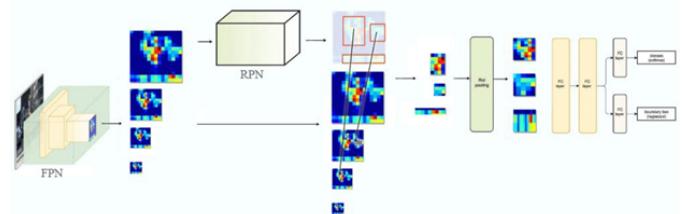
- Năm 2017, Tsung-Yi Lin cùng các cộng sự ở Facebook AI Research đã đề xuất Feature Pyramid Networks (FPN) dựa trên cơ sở Faster R-CNN trong bài báo “Feature Pyramid Networks for Object Detection”. Trước FPN, các mô hình Deep Learning đều dự đoán ở lớp trên cùng của

mạng. Ở FPN, mỗi layer của kiến trúc mạng sẽ đưa ra một predict. Phương pháp này thể hiện sự tiến bộ trong Object Detection với độ chính xác cao hơn. Gần đây, FPN là một trong những building block cơ bản nhất của các phương pháp detectors.



Hình 8: Kiến trúc Feature Pyramid Networks được lấy từ bài báo gốc.

- Ví dụ đối với tầng cao nhất của kiến trúc mạng nó sẽ 2x upsampling và kết hợp với tầng trước đó đã cho qua 1x1 conv để cho ra layer predict sau đó như hình trên.

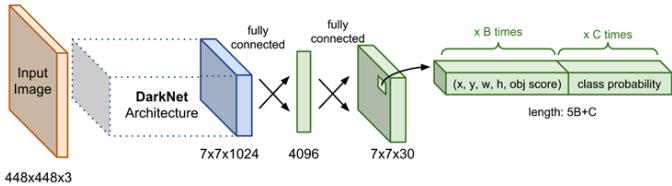


Hình 9: FPN cho Faster R-CNN.

III. CÁC MÔ HÌNH ONE-STAGE

A. You only look once (YOLO)

- Mô hình YOLO được đề nghị bởi R. Joseph vào năm 2015, là mô hình phát hiện đối tượng one-state đầu tiên.
- YOLO đã thay đổi hoàn toàn các mô hình phát hiện đối tượng trước đó với các giai đoạn: phát hiện vùng vật thể và nhận diện. Thay vào đó áp dụng một mạng neural duy nhất, mạng này chia ảnh đầu vào thành các vùng, dự đoán vùng chứa vật thể và xác suất một cách đồng thời.
- Về độ chính xác, YOLO không bằng các thuật toán two-stage nhưng tốc độ nhận diện rất nhanh (nhanh nhất trong các thuật toán phát hiện đối tượng), độ chính xác cũng không quá giảm so với các mô hình top đầu.
- Kiến trúc YOLO bao gồm: base network là các mạng convolution làm nhiệm vụ trích xuất đặc trưng, phía sau là những Extra Layers được áp dụng để phát hiện vật thể trên feature map của base network.

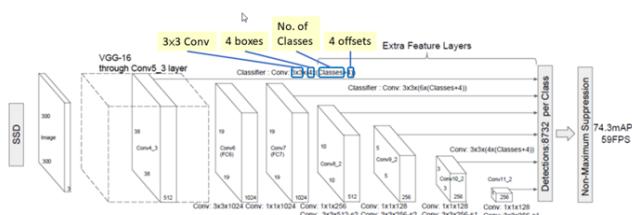


Hình 10: Sơ đồ kiến trúc mạng YOLO.

- Các phiên bản của YOLO: Yolo-v4 (13-05-2020), Yolo-v3 (7-06-2018), Yolo-v2 (28-03-2018)

B. Single Shot MultiBox Detector (SSD)

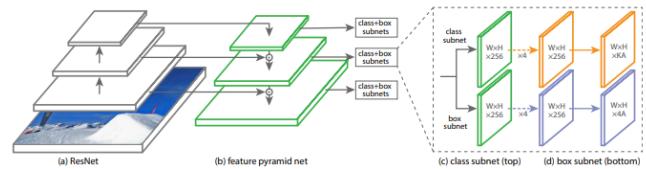
- Mô hình SSD được đề xuất bởi W. Liu vào năm 2015. SSD là mô hình phát hiện đối tượng thứ 2 mang tính chất one-stage.
- Đóng góp chính của mô hình SSD là giới thiệu các kĩ thuật nhận diện “multi-reference” và “multi-resolution”, các kĩ thuật giúp tăng độ chính xác của mô hình one-stage, đặc biệt với các đối tượng nhỏ.
 - Multi-references: ý tưởng chính là định nghĩa trước tập các khung tham chiếu – reference boxes (hay gọi là anchor boxes) với kích thước khác nhau và tỷ lệ cạnh (aspect-ratios) tại các vị trí khác nhau toàn bộ ảnh, và dự đoán các khung dự đoán (detection box) dựa trên tập khung định trước.
 - Multi-resolution: nhận diện đối tượng trên nhiều độ phân giải của ảnh (tỉ lệ khác nhau) ở các lớp (layer) khác nhau trong mạng.
- Điểm khác biệt chính giữa SSD và các mô hình ra đời sớm hơn là SSD nhận diện vật thể trên nhiều tỉ lệ (scale) trên nhiều layer khác nhau của mạng, trong khi các mô hình trước SSD chỉ chạy phát hiện đối tượng trên lớp cao nhất.



Hình 11: Sơ đồ kiến trúc mạng SSD.

C. RetinaNet

- Mặc dù các mô hình one-stage object detection có tốc độ cao và đơn giản, nhưng chúng có độ chính xác vẫn sau các mô hình two-stage. T.-Y. Lin tìm ra nguyên nhân đằng sau đó và đề xuất mô hình RetinaNet vào năm 2017.
- Nguyên chính là sự mất cân bằng giữa lớp đối tượng nền và lớp đối tượng không phải nền gấp phai trong quá trình huấn luyện ở các giai đoạn huấn luyện lớp dày đặc.



Hình 12: Sơ đồ kiến trúc mạng RetinaNet sử dụng Feature Pyramid Network.

- RetinaNet đã đưa ra một hàm lối mới mang tên “focal loss” bằng cách tinh chỉnh lại hàm Cross Entropy Loss nguyên thủy.

IV. MÔ HÌNH ĐỀ XUẤT

- Mô hình nhóm chọn: You Only Look Once (YOLO).
- Lý do chọn mô hình:
 - Là mô hình one-stage đầu tiên được đề xuất nên nó là nền tảng cho các mô hình sau.
 - Được áp dụng nhiều trong các bài toán Real Time.
 - Nhiều tài liệu hướng dẫn, tham khảo từ đó có thể tìm hiểu sâu và nhanh về mô hình.

V. TÌM HIỂU VỀ MÔ HÌNH YOU ONLY LOOK ONCE (YOLO)

A. Tổng quan

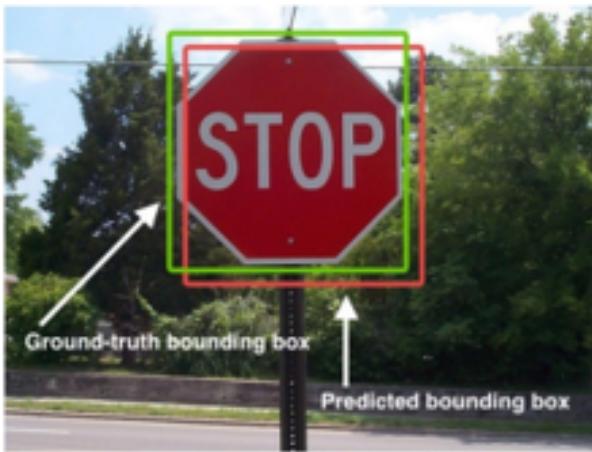
- YOLO – YOU ONLY LOOK ONCE là một mô hình mạng CNN cho phát hiện, nhận dạng, phân loại đối tượng. YOLO là một trong những phương pháp tốt nhất và nhanh nhất (real time) hiện nay. Mặc dù không phải là phương pháp có độ chính xác cao nhất tuy nhiên YOLO vẫn được ứng dụng rất nhiều trong các dự án thực tế mà khi độ chính xác không phải ưu tiên hàng đầu.
- Đến thời điểm hiện nay YOLO đã có 4 phiên bản v1, v2, v3, v4.
 - Paper – YOLOv1: You Only Look Once: Unified, Real-Time Object Detection (2016)
 - Paper – YOLOv2: YOLO9000: Better, Faster, Stronger (2016)
 - Paper – YOLOv3: YOLOv3: An Incremental Improvement (2018)
 - Paper – YOLOv4: YOLOv4: Optimal Speed and Accuracy of Object Detection (2020)

B. Kiến trúc của mô hình YOLO

1) Yolo v1:

- Trước hết ta nói về Intersection Over Union (IoU) trong Object Detection.
- Intersection Over Union (IoU)
 - IoU được sử dụng trong bài toán Object Detection, để đánh giá xem bounding box dự đoán đối tượng khớp với ground truth thật của đối tượng.
 - IoU đơn giản chỉ là một chỉ số đánh giá. Mọi thuật toán có khả năng predict ra các bounding box làm output đều có thể được đánh giá thông qua IoU.

- Để áp dụng được IoU để đánh giá một object detector bắt kè ta cần:
 - * Những ground truth bounding box (bounding box đúng của đối tượng, ví dụ như bounding box của đối tượng được khoanh vùng và đánh nhãn bằng tay sử dụng trong tập test.)
 - * Những predicted bounding box được model sinh ra.
- Minh họa IoU.



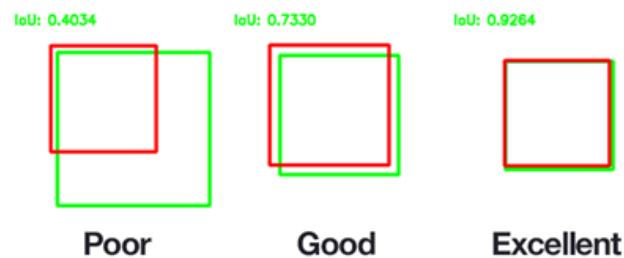
Hình 13: Minh họa IoU.

- Công thức tính IoU

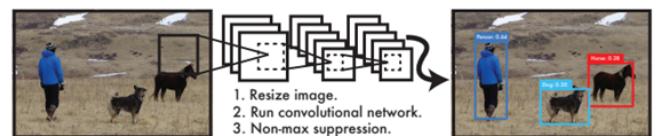
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Hình 14: Cách tính IoU.

- Area of Overlap: diện tích phần chồng lênh nhau giữa predicted bounding box và ground truth bounding box.
- Area of Union: diện tích hợp giữa predicted và ground truth bounding box.
- Chỉ số IoU nằm trong khoảng [0,1].
- IoU càng gần 1 thì bounding box dự đoán càng gần ground truth.
- Cách Yolo v1 hoạt động
 - Đầu vào của mô hình là một ảnh, mô hình sẽ nhận dạng ảnh đó có đối tượng nào hay không, sau đó sẽ



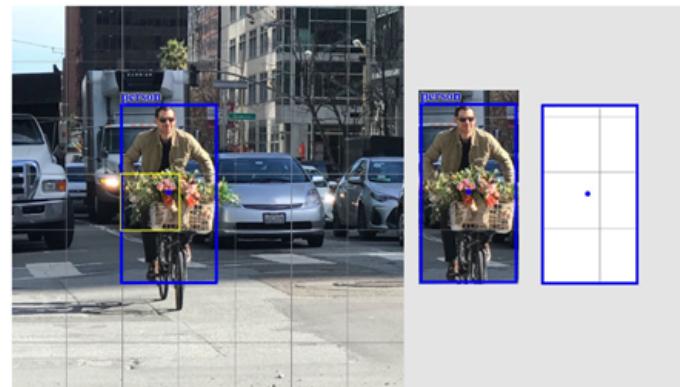
Hình 15: Ví dụ về IoU.



Hình 16: Tổng quan hoạt động của YOLOv1, hình được lấy từ bài báo gốc.

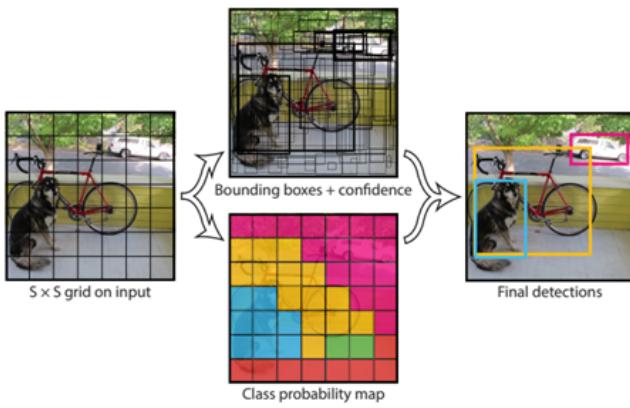
xác định tọa độ của đối tượng trong bức ảnh. Ảnh đầu vào được chia thành $S \times S$ ô, mỗi ô dự đoán B bounding box để phát hiện một đối tượng, độ tin cậy của mỗi box và xác suất của C class (giả sử trong dữ liệu huấn luyện có C class). Đầu ra của mô hình là một tensor có kích thước $(S \times S \times (5 * B + C))$.

- YOLO sẽ dự đoán xem trong mỗi ô liệu có object mà điểm trung tâm object rơi vào ô đó và ô đó sẽ “chịu trách nhiệm” phát hiện đối tượng đó.



Hình 17: Ví dụ ô khung màu vàng sẽ có găng phát hiện đối tượng người mà điểm trung tâm của đối tượng nằm trong ô này.

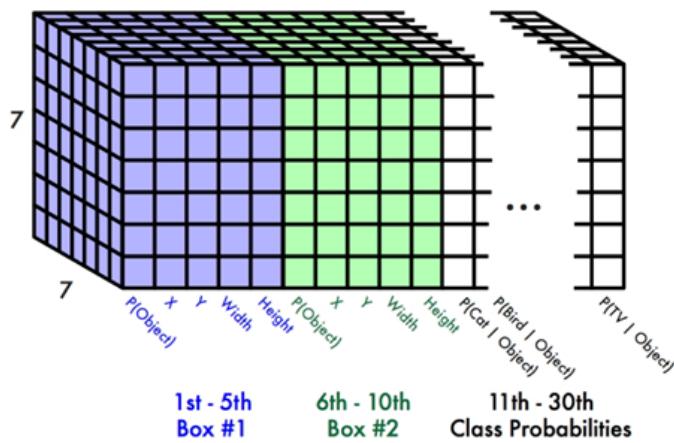
- Trong bài báo với dữ liệu huấn luyện là PASCAL VOC, tác giả sử dụng $S = 7, B = 2$. PASCAL VOC có 20 nhãn dữ liệu nên $C = 20$. Do đó đầu ra của mô hình là $7 \times 7 \times 30$.
- Ta lấy ví dụ đơn giản với hình ảnh lấy từ bài báo gốc ở trên, ta thấy chia thành 7×7 ô, mỗi ô cần dự đoán 2 bounding box và 3 object: con chó, ô tô, xe đạp ($B = 2, C = 3$) thì output là $7 \times 7 \times 13$, mỗi ô



Hình 18: Được lấy từ bài báo gốc.

sẽ có 13 tham số, kết quả trả về có $7 \times 7 \times 2 = 98$ bounding box.

- Tổng quát hơn, với dữ liệu PASCAL VOC có 20 class, đầu ra của mô hình là tensor $7 \times 7 \times 30$. Dự đoán mỗi bounding box bao gồm 5 thành phần: (x, y, w, h, prediction) với (x, y) là tọa độ tâm của bounding box, (w, h) lần lượt là chiều rộng và chiều cao của bounding box, prediction được định nghĩa bởi $\text{Pr}(\text{Object}) * \text{IoU}(\text{pred}, \text{truth})$. Trong đó:
 - * IoU(pred, truth) là chỉ số được định nghĩa ở phần trên.
 - * $\text{P}(\text{Object})$: Xác suất box đó có chứa object hay không.
- Với dữ liệu huấn luyện PASCAL VOC, ta có thể hiểu mỗi ô có 30 tham số, tham số thứ nhất sẽ chỉ ra ô đó có chứa đối tượng nào hay không $\text{P}(\text{Object})$, tham số 2, 3, 4, 5 sẽ trả về x, y, w, h của box thứ nhất. Tham số 6, 7, 8, 9, 10 tương tự cho box thứ 2. Tham số thứ 11 là xác suất object trong ô là object 1 (trong 20 object cần nhận dạng), tương tự tham số 12 là xác suất object trong ô là object 2,... cho đến tham số thứ 30 là xác suất object trong ô là object 20.



Hình 19: Minh họa tensor đầu ra $7 \times 7 \times 30$.

- Khi đó, class confidence score cho mỗi prediction box được tính bởi:

$$\text{class confidence score} = \text{box confidence score} \times \text{conditional class probability}$$

Trong đó:

- * box confidence score là $\text{P}(\text{Object}) * \text{IoU}$ được định nghĩa ở trên.
- * conditional class probability là $P(\text{class}_i | \text{object})$: xác suất object đó thuộc về class_i khi đã biết box chứa object.
- * $\text{Pr}(\text{class}_i | \text{Object}) * \text{Pr}(\text{Object}) * \text{IoU}_{\text{pred}}^{\text{truth}} = \text{Pr}(\text{class}_i) * \text{IoU}_{\text{pred}}^{\text{truth}}$

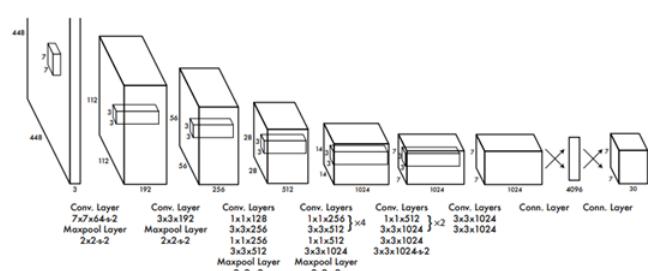
- Về kiến trúc mạng của YOLO v1:

* Kiến trúc mạng được lấy cảm hứng từ mô hình GoogleNet dùng trong phân loại ảnh. Gồm có 24 convolutional layers, bao gồm các 1×1 reduction layers và 3×3 convolutional layers, theo sau là 2 fully connected layers. Hàm kích hoạt (activation function) cho layer cuối cùng là một linear activation function và tất cả các layers khác sử dụng leaky ReLU.

Hàm Leaky ReLU:

$$\phi(x) = \begin{cases} x, & \text{if } x > 0. \\ 0.1x & \text{otherwise.} \end{cases} \quad (1)$$

- * Image input được resize thành image $448 \times 448 \times 3$ (image-dimension=448 \times 448 với số channels=3, YOLO sử dụng hệ màu HSV).
- * Qua các layers, biến đổi image $448 \times 448 \times 3$ thành 1 grid có kích thước 7×7 với số tham số cho mỗi ô trong grid là 30 ($30 = 5 \times \text{boxes-number} + \text{number-of-objects}$).
- * Neural net có nhiệm vụ huấn luyện các trọng số của các layers để có được mô hình tốt cuối cùng.
- * Dựa vào các tham số trong grid $7 \times 7 \times 30$ ta sẽ xác định được các box chứa object với xác suất cao. (các box đè lên nhau sẽ được loại bằng phương pháp NMS (Non-max suppression), chỉ giữ lại box có xác suất cao nhất).



Hình 20: Kiến trúc mạng của YOLOv1 được lấy từ bài báo gốc.

Name	Filters	Output Dimension
Conv 1	7 x 7 x 64, stride=2	224 x 224 x 64
Max Pool 1	2 x 2, stride=2	112 x 112 x 64
Conv 2	3 x 3 x 192	112 x 112 x 192
Max Pool 2	2 x 2, stride=2	56 x 56 x 192
Conv 3	1 x 1 x 128	56 x 56 x 128
Conv 4	3 x 3 x 256	56 x 56 x 256
Conv 5	1 x 1 x 256	56 x 56 x 256
Conv 6	1 x 1 x 512	56 x 56 x 512
Max Pool 3	2 x 2, stride=2	28 x 28 x 512
Conv 7	1 x 1 x 256	28 x 28 x 256
Conv 8	3 x 3 x 512	28 x 28 x 512
Conv 9	1 x 1 x 256	28 x 28 x 256
Conv 10	3 x 3 x 512	28 x 28 x 512
Conv 11	1 x 1 x 256	28 x 28 x 256
Conv 12	3 x 3 x 512	28 x 28 x 512
Conv 13	1 x 1 x 256	28 x 28 x 256
Conv 14	3 x 3 x 512	28 x 28 x 512
Conv 15	1 x 1 x 512	28 x 28 x 512
Conv 16	3 x 3 x 1024	28 x 28 x 1024
Max Pool 4	2 x 2, stride=2	14 x 14 x 1024
Conv 17	1 x 1 x 512	14 x 14 x 512
Conv 18	3 x 3 x 1024	14 x 14 x 1024
Conv 19	1 x 1 x 512	14 x 14 x 512
Conv 20	3 x 3 x 1024	14 x 14 x 1024
Conv 21	3 x 3 x 1024	14 x 14 x 1024
Conv 22	3 x 3 x 1024, stride=2	7 x 7 x 1024
Conv 23	3 x 3 x 1024	7 x 7 x 1024
Conv 24	3 x 3 x 1024	7 x 7 x 1024
FC 1	-	4096
FC 2	-	7 x 7 x 30 (1470)

Hình 21: Tham số cù thể của từng layers.

• Hàm Loss Function

- Các khái niệm

- * 1_i^{obj} : Hàm có giá trị 0, 1 nhằm xác định xem ô i có chứa vật thể hay không. Bằng 1 nếu chứa vật thể và bằng 0 nếu không chứa.
- * 1_{ij}^{obj} : Cho biết bounding box thứ j của ô i có phải là bounding box của vật thể được dự đoán hay không, bằng 1 nếu box thứ j của ô thứ i có chứa object và ngược lại.
- * Đối với box j của ô i nơi object tồn tại, tính tổn thất của xác suất object tồn tại. ($C_i = 1$)
- * Đối với box j của ô i nơi không chứa object, tính tổn thất của xác suất này. ($C_i = 0$)
- * C : tập hợp tất cả các nhãn.
- * $p_i(c)$: Xác suất có điều kiện ô i chứa một đối tượng của lớp c thuộc C .
- * $\hat{p}_i(c)$: Xác suất có điều kiện của mô hình dự đoán.
- * Chú ý $p_i(c) = 1$ nếu đúng lớp c với ground-truth, ngược lại thì $p_i(c) = 0$.

- Loss Function của Yolo

- * Hàm lỗi trong YOLO được tính trên việc dự đoán và nhãn mô hình để tính. Cụ thể hơn nó là tổng độ lỗi của 3 thành phần sau:

- Lỗi dự đoán nhãn của object – Classification loss.
- Lỗi dự đoán tâm, chiều dài, chiều cao của bounding box – Localization loss (độ lỗi giữa predict bounding box và ground truth).

- Lỗi phát hiện bounding box có chứa object hay không – Confidence loss.

* Classification Loss

- Lỗi dự đoán loại nhãn của object, hàm lỗi này chỉ tính trên những ô xuất hiện object.

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

* Localization Loss

- Localization loss là hàm lỗi dùng để tính giá trị lỗi cho bounding box được dự đoán bao gồm tọa độ tâm, chiều rộng, chiều cao của bounding box so với vị trí thực tế từ dữ liệu huấn luyện của mô hình

- Ta không nên tính giá trị hàm lỗi này trực tiếp từ kích thước ảnh thực tế mà cần chuẩn hóa về đoạn [0,1] so với tâm của bounding box. Việc chuẩn hóa kích thước này giúp cho giá trị hàm loss nhỏ và mô hình dự đoán nhanh hơn và chính xác hơn so với để giá trị mặc định của ảnh.

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

* Confidence Loss

- Lỗi dự đoán bounding box có chứa object hay không. Độ lỗi này tính trên cả những ô vuông chứa object và không chứa object.

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

- 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 ô không chứa vật thể bằng hệ số điều chỉnh λ_{noobj} .

- * Trong bài báo gốc tác giả chọn $\lambda_{\text{coord}} = 5$ và $\lambda_{\text{noobj}} = 0.5$.

* Total Loss

- Tổng lại ta có Loss Function của YOLO là tổng của 3 hàm lỗi trên.

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

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

- Non-max suppression

- Một số object lớn có thể được phát hiện bởi nhiều grid cùng một lúc. Trong trường hợp đó YOLO sẽ cần đến non-max suppression để giảm bớt số lượng bounding box sinh ra.



Hình 22: Non-max suppression, từ 3 bounding box bao quanh chiếc xe đã giảm thành 1 bounding box cuối cùng.

- Các bước của non-max suppression

- Lọc bỏ toàn bộ những bounding box có xác suất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó, thường là 0.5.
- Đối với các bounding box giao nhau, non-max suppression sẽ chọn ra một bounding box có xác suất chứa vật thể lớn nhất. Sau đó tính toán chỉ số IoU với các bounding box còn lại.
- 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 box có xác suất thấp hơn và giữ lại bounding box có xác suất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

- Hạn chế của YOLOv1

- Như cách hoạt động của YOLOv1 như trên, ta thấy với mỗi ô được chia nó giới hạn số lượng bounding box dự đoán (=2) và mỗi ô chỉ phát hiện một object. Do vậy nên nó không thể tìm thấy các object nhỏ và xuất hiện dưới dạng một cụm (chẳng hạn như một đàn chim). Phiên bản này gặp khó khăn trong việc phát hiện các đối tượng nếu hình ảnh có kích thước khác với hình ảnh được train .



Hình 23: Ví dụ thể hiện hạn chế của YOLOv1 với các đối tượng xuất hiện dưới dạng một cụm. Như chúng ta thấy thì YOLOv1 chỉ phát hiện 5 ông già Noel từ góc bên trái, nhưng thực tế có 9 ông già Noel.

2) Yolo v2:

- Nhanh hơn, tiên tiến hơn để đáp ứng Faster R-CNN.
- Những thay đổi của YOLOv2 so với YOLOv1:

- Batch Normalization:

- * Mục tiêu của phương pháp: chuẩn hóa các feature (đầu ra của mỗi layer sau khi đi qua các activation) về trạng thái zero-mean với độ lệch chuẩn 1.
- * Tránh được hiện tượng giá trị tham số rơi vào khoảng bão hòa sau khi đi qua các hàm kích hoạt phi tuyến, đảm bảo rằng không có sự kích hoạt nào bị vượt quá cao hoặc quá thấp, các weights mà khi không dùng BN có thể sẽ không bao giờ được học thì nay lại được học bình thường nên làm giảm đi sự phụ thuộc vào giá trị khởi tạo của các tham số.
- * Vai trò như một dạng của regularization giúp cho việc giảm thiểu overfitting. Sử dụng batch normalization, không cần phải sử dụng quá nhiều dropout và điều này rất có ý nghĩa vì không cần phải lo lắng vì bị mất quá nhiều thông tin khi dropout weights của mạng. Tuy nhiên vẫn nên sử dụng kết hợp cả hai kĩ thuật này.
- * Bằng cách thêm chuẩn hóa hàng loạt vào các lớp chập trong kiến trúc, mAP (mean average precision) đã được cải thiện 2%.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;	
Parameters to be learned: γ, β	
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$	
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$	// mini-batch mean
$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2$	// mini-batch variance
$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$	// normalize
$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$	// scale and shift

Hình 24: Thuật toán Batch Normalization.

- Higher Resolution Classifier: Kích thước đầu vào trong YOLOv2 đã được tăng từ 224×224 lên 448×448 , từ đó cải thiện mAP (mean average precision) tăng lên 4%.
- Anchor boxes: YOLO v2 thực hiện phân loại và dự đoán trong một khung duy nhất.
 - * Là một bounding box cơ sở để xác định bounding box bao quanh vật thể dựa trên các phép dịch tâm và scale kích thước chiều dài, rộng.
 - * 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.
 - * 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.
 - * Mỗi loại anchor box sẽ phù hợp để tìm ra bounding box cho 1 loại vật thể đặc trưng. (Chẳng hạn vật thể là con người thường có chiều cao > chiều rộng trong khi đoàn tàu sẽ có chiều rộng lớn hơn nhiều lần chiều cao).
- Fine-Grained Features: Một vấn đề của YOLOv1 là phát hiện vật thể nhỏ trong ảnh đầu vào. Ở YOLOv2, ảnh đầu vào được chia thành lưới 13×13 (nhỏ hơn so với YOLOv1) từ đó xác định, định vị được cho cả những vật thể nhỏ lẫn vật thể lớn.
- Multi-Scale Training: YOLOv1 có điểm yếu không nhận dạng được vật thể ở ảnh đầu vào khi nó có kích thước khác với kích thước đã học (cùng 1 một vật thể) và YOLOv2 đã khắc phục được điều này. Khi trong YOLOv2 được học với các hình ảnh ngẫu nhiên với kích thước khác nhau trong khoảng từ 320×320 đến 608×608 từ đó có thể học và dự đoán các đối tượng từ các kích thước đầu vào khác nhau với độ

chính xác cao.

- Darknet 19: YOLOv2 sử dụng kiến trúc Darknet19 với 19 lớp chập (convolutional layers) và 5 lớp gộp tối đa (max pooling layers) và một lớp softmax cho các đối tượng phân loại.

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

Hình 25: Darknet 19.

- Dự đoán 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 dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều.
- * 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) . 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 sau:

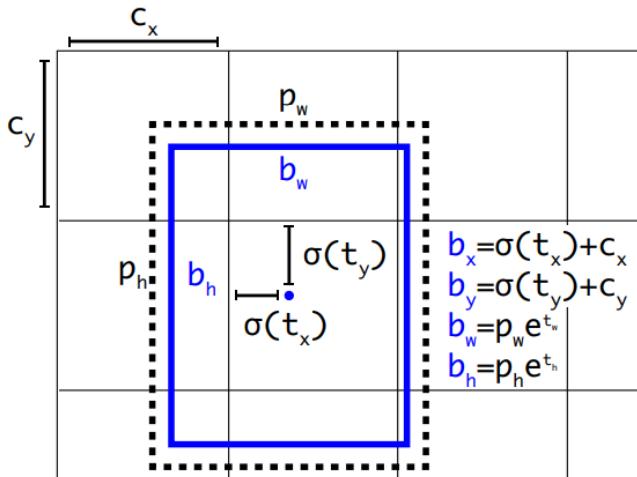
$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$
- * Bounding box dự đoán sẽ đậu đó quanh vị trí của cell và anchor box mà không vượt quá xa ra ngoài giới hạn này. Do đó quá trình huấn luyện sẽ ổn

định hơn rất nhiều so với YOLOv1.



Hình 26: 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 .)

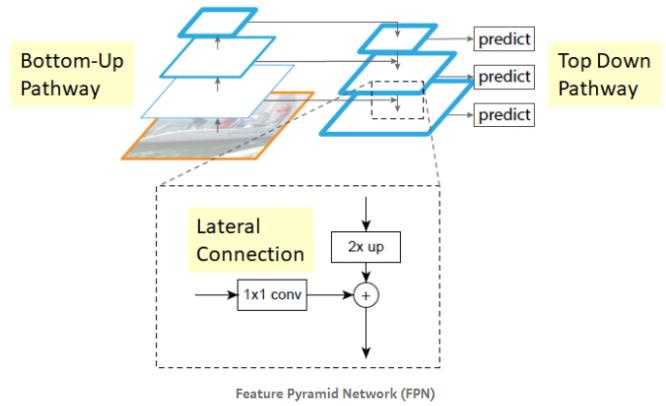
- Nhờ Multi-Scale, mạng có thể phát hiện và phân loại các đối tượng có cấu hình và kích thước khác nhau.
- Khi phát hiện các vật thể nhỏ, có độ chính xác cao hơn so với phiên bản trước.

3) Yolo v3:

- YOLOv3 có thể giúp phát hiện đối tượng trong thời gian thực với việc phân loại chính xác.
- Class Predictions: Không sử dụng softmax layers như trong YOLOv2, YOLOv3 sử dụng logistic classifiers cho mỗi lớp và binary cross-entropy loss, do vậy có thể phân loại đa nhãn (multi-label classification) :
 - Softmax layer: nếu train người và đàn ông thì xác suất giữa người và người đàn ông là 0.4 và 0.47.
 - Phân loại từng lớp (independent classifier) đưa ra xác suất cho từng lớp đối tượng. Ví dụ nếu train cho cả người và người đàn ông thì sẽ đưa ra xác suất cho người là 0.85 và người đàn ông là 0.8 và gắn nhãn cho đối tượng trong ảnh là người và người đàn ông.

• Feature Pyramid Networks (FPN)

- YOLOv3 đưa ra các dự đoán tương tự như FPN trong đó 3 dự đoán được thực hiện cho mọi vị trí hình ảnh đầu vào và các tính năng được trích xuất từ mỗi dự đoán nên có khả năng tốt hơn ở các quy mô khác nhau.
- Việc upsampling từ các layer trước cho phép nhận được thông tin ngữ nghĩa đầy đủ và thông tin chi tiết hơn từ feature map trước đó và thêm vài lớp chập nữa để xử lý cải thiện đầu ra .



Hình 27: Feature Pyramid Network (FPN)

• Darknet 53

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

Hình 28: Darknet-53

- YOLOv3 sâu hơn nhiều so với YOLOv2 (53 > 19) và có những shortcut connections.
- Darknet-53 bao gồm chủ yếu các bộ lọc 3x3 và 1x1 với các shortcut connections
- Có rất nhiều phạm vi cho những cải tiến trong các thuật toán phát hiện đối tượng như YOLO v3, faster R-CNN, SSD,...

- Những cải tiến nhỏ nhất trong các thuật toán này có thể thay đổi toàn bộ nhận thức trong thế giới thực.
- Phát hiện đối tượng giúp con người đỡ tốn công sức trong nhiều lĩnh vực.
- Phát hiện đối tượng trong thời gian thực với độ chính xác cao là một trong những tiêu chí chính trên thế giới nơi xe tự lái đang trở thành hiện thực.
- Nhận dạng với hơn 80 objects trên một tấm hình.

VI. THỰC NGHIỆM, PHÂN TÍCH VÀ ĐÁNH GIÁ MÔ HÌNH

A. Thực nghiệm với mô hình đã huấn luyện sẵn (pretrained model)

- YOLO được huấn luyện trên rất nhiều các model pretrained. Những mô hình này được xây dựng trên các bộ dữ liệu ảnh mẫu lớn như: COCO, Pascal VOC, Imagenet, CIFAR.
- Chi tiết về cách chạy dữ liệu đưa vào được trình bày ở 2 file: [RunYoloV3_image.ipynb](#) đối với đầu vào là ảnh và [RunYoloV3_video.ipynb](#) đối với đầu vào là video.



Hình 29: Ví dụ ảnh thử nghiệm.



Hình 30: Ví dụ kết quả thử nghiệm trên pretrained model.

- Mô hình chạy khá nhanh và dự đoán chính xác. Tuy nhiên, nếu mục đích của chúng ta là chỉ dự đoán với một số loại đối tượng mà ta quan tâm thì phải huấn luyện lại mô hình trên tập dữ liệu chứa các đối tượng đó.

B. Thực nghiệm trên một tập dữ liệu mới

1) Mô tả tập dữ liệu:

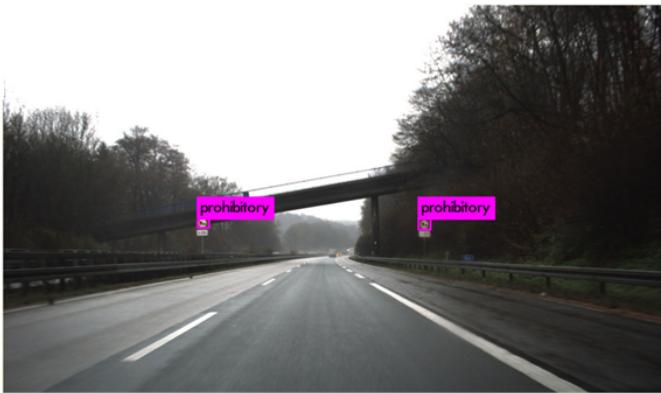
- Nguồn dữ liệu: [Traffic_Sign_Dataset](#)
- Bộ dữ liệu về biển báo giao thông được đánh nhãn theo format YOLO cho tác vụ nhận diện.
- Bộ dữ liệu bao gồm ảnh định dạng *.jpg và các file đánh nhãn các bounding box cho từng ảnh định dạng *.txt.
- Phần thực hành dùng: 640 ảnh để huấn luyện, 101 ảnh để kiểm thử.
- Các biển báo trong bộ dữ liệu được nhóm thành 4 nhóm:
 - Prohibitory (biển báo cấm): biển giới hạn tốc độ, biển cấm vượt xe, biển báo giao thông 1 chiều, cấm xe tải.
 - Danger (biển báo nguy hiểm): giao nhau với đường ưu tiên, biển nguy hiểm khác, biển chố ngoặt bên trái, biển chố ngoặt bên phải, điểm ngoặt, đường không đồng đều, đường trơn trượt, công trình xây dựng, giao nhau có tín hiệu giao thông, người đi bộ sang đường, điểm có trường học, người đi xe đạp cắt ngang, tuyết, động vật.
 - Mandatory (biển hiệu lệnh): hướng đi phải phải theo, hướng đi trái phải theo, hướng đi thẳng phải theo, hướng đi thẳng hoặc queo phải, hướng đi thẳng hoặc queo trái, các xe chỉ đường queo phải, các xe chỉ đường queo trái, nơi giao nhau chạy theo vòng xuyến.
 - Other (biển báo khác): hướng cùt, đường ưu tiên, dừng lại, không có đường, được phép đi.

2) Minh họa kết quả:

- Ảnh thử thứ nhất



Hình 31: Ảnh thử thứ nhất.

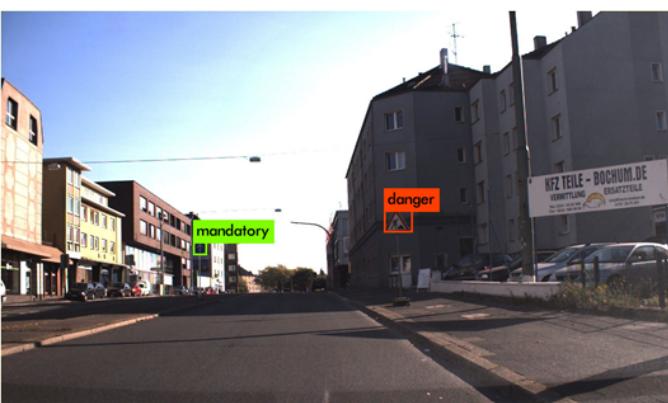


Hình 32: Minh họa kết quả thử nghiệm ảnh thứ nhất.

- Ảnh thử thứ hai



Hình 33: Ảnh thử thứ hai.



Hình 34: Minh họa kết quả thử nghiệm ảnh thứ hai.

3) Nhận xét và đánh giá:

- Thông số huấn luyện: huấn luyện 1000 batches, mỗi batch số lượng 64 ảnh, thời gian huấn luyện trên mỗi batch dao động 2-6 giây tùy thuộc vào kích thước ảnh đã resize.
- Thời gian huấn luyện mất hơn 2 giờ đồng hồ, nên việc huấn luyện số lượng batches lớn sẽ không khả thi trên google colab.

- Tuy nhiên, độ lỗi trung bình sau khi huấn luyện là ~ 0.5 . Việc nhận dạng trên tập thử khá chính xác.

4) Liên kết:

- File Weight: [weights](#).
- File Config: [config](#).
- Chi tiết về cách chạy mô hình được trình bày ở file [RoadSignDetection.ipynb](#).

VII. TÀI LIỆU THAM KHẢO

- Object Detection in 20 Years: A Survey
- Rich feature hierarchies for accurate object detection and semantic segmentation
- Fast R-CNN
- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
- Feature Pyramid Networks for Object Detection
- Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition
- Focal Loss for Dense Object Detection
- You Only Look Once: Unified, Real-Time Object Detection
- YOLO9000: Better, Faster, Stronger
- YOLOv3: An Incremental Improvement
- Khoa học dữ liệu - Khanh's blog
- <https://medium.com/@venkatakrishna.jonnalagadda/object-detection-yolo-v1-v2-v3-c3d5eca2312a>
- https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088
- <https://forum.machinelearningcobel.com/t/object-detection-yolo/503>
- <https://www.phamduytung.com/blog/2018-12-06-what-do-we-learn-from-single-shot-object-detection/>
- <https://phamduinhkhanh.github.io/2019/09/29/OverviewObjectDetection.html>