**YOLO Evolution**

**You Only Look Once**

**Thực tập sinh:** Trần Văn Trung.

**1. YOLO v1.**

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

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

Ý tưởng bài toán Object Detection:

- Chia ảnh thành nhiều box, với mỗi box sẽ detect object trong box đó. Vị trí của object chính là tọa độ của box đó.

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

Chi tiết bài toán với YOLO v1:

**Grid System**

- Ảnh được chia thành ma trận ô vuông 7x7 (Grid System), mỗi ô vuông bao gồm một tập các thông tin mà mô hình phải dự đoán.

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

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

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

- Ô vuông có chứa đối tượng nào.

- Dự đoán độ lệch 2 box chứa object so với ô vuông hiện tại.

- Lớp của đối tượng đó.

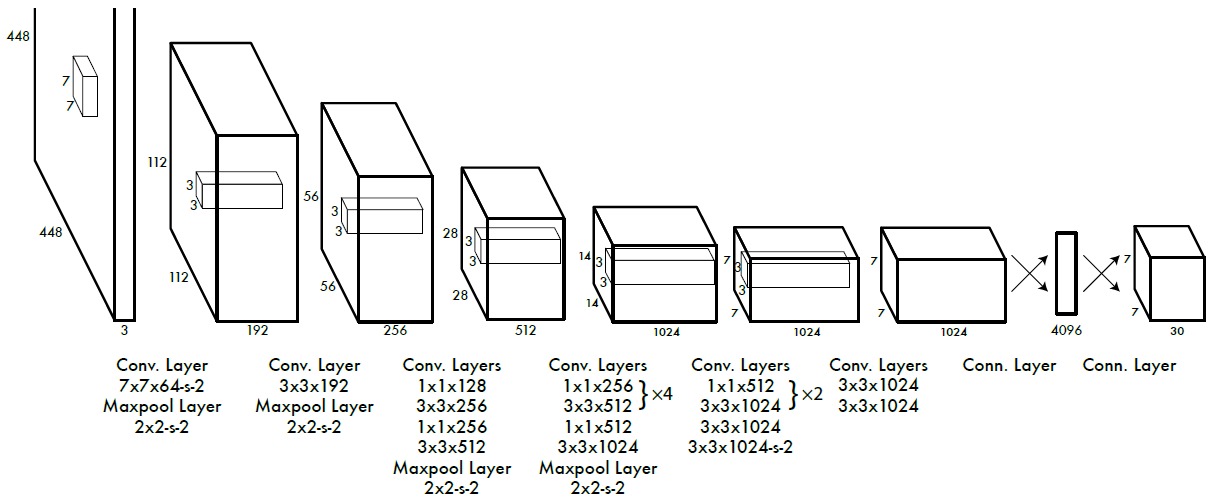
- Với mỗi ô vuông cần dự đoán một vecto có (5\*nbox + nclass)

- Ví dụ: Dự đoán 2 box, 3 lớp, với mỗi ô vuông chúng ta có một ma trận 3 chiều 7x7x13 chứa các thông tin cần thiết:

Hình 1.1: Ví dụ một vecto 13x1 cho Grid 7x7, 3 class.

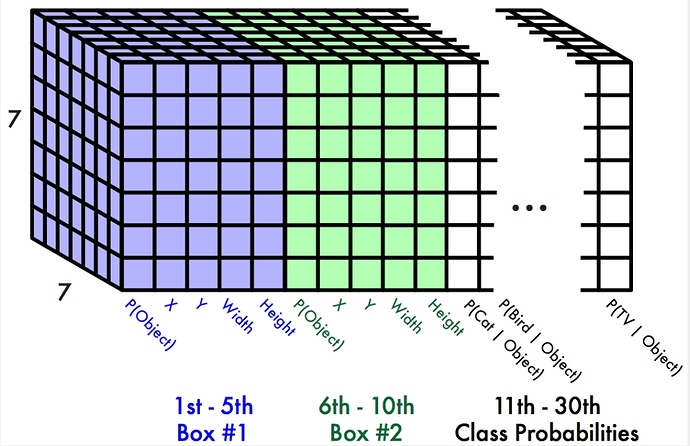
**CNN cho YOLO Object Detection**

- Chúng ta đã biết cần dự đoán thông tin nào với mỗi ô vuông, mô hình CNN phải cho ra output shape phù hợp với yêu cầu dự đoán, tức Grid\_Size x Grid\_Size x (5\*nbox + nclass). Như ví dụ trước với 2 box, 3 class là 7x7x(5\*2+3).

Hình 1.2: Kiến trúc YOLO v1.

- YOLO sử dụng linear regression để dự đoán thông tin ở mỗi ô vuông. Ở layer cuối cùng sẽ không sử dụng hàm kích hoạt nào cả. Thay vì sử dụng FC ở cuối, có thể thay bằng các Conv 1x1 với (nbox\*5 + nclass) feature maps.

- Ví dụ: Với một bài toán nhận dạng đối tượng cho 20 lớp đối tượng, Gird chia thành 7x7, số bounding box dự đoán cho mỗi ô là 2, số lượng đối tượng là 20. Số thông tin cần tự đoán cho 1 ô vuông là 5\*2+20 = 30. Với toàn bộ 7 ô vuông là 7x7x30. Hình ảnh minh họa:

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

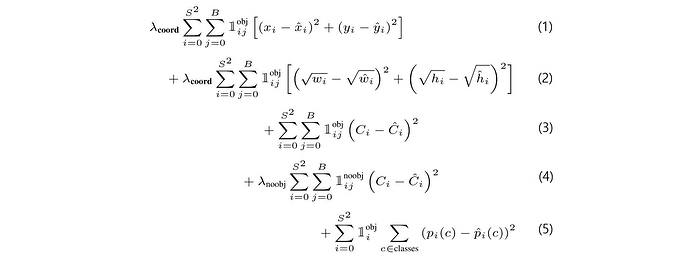
**Loss Function**

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

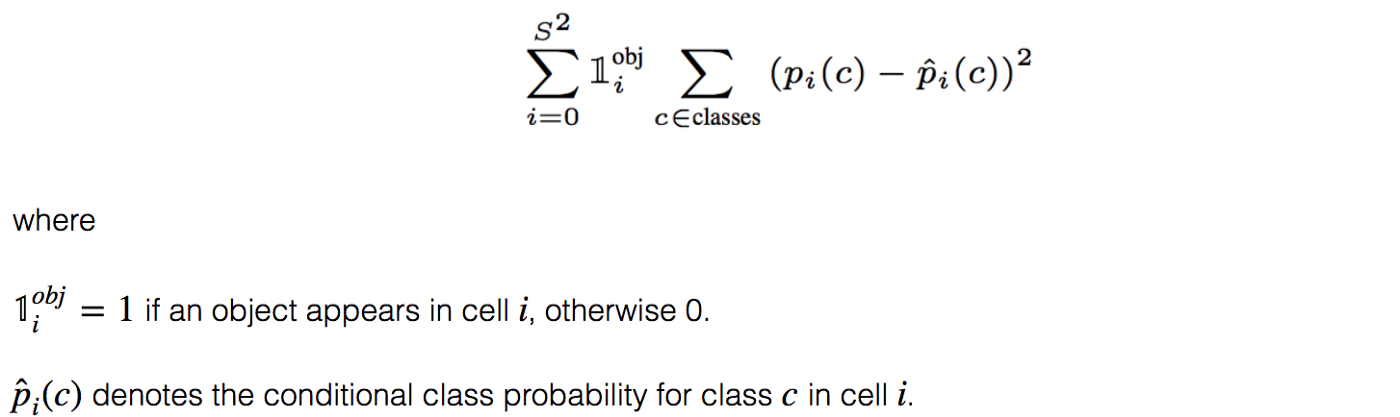
- Độ lỗi của việc dự đoán nhãn của object - Classification Loss.

- Độ lỗi của dự đoán tọa độ và dự đoán chiều dài, chiều rộng bounding box - Localization Loss.

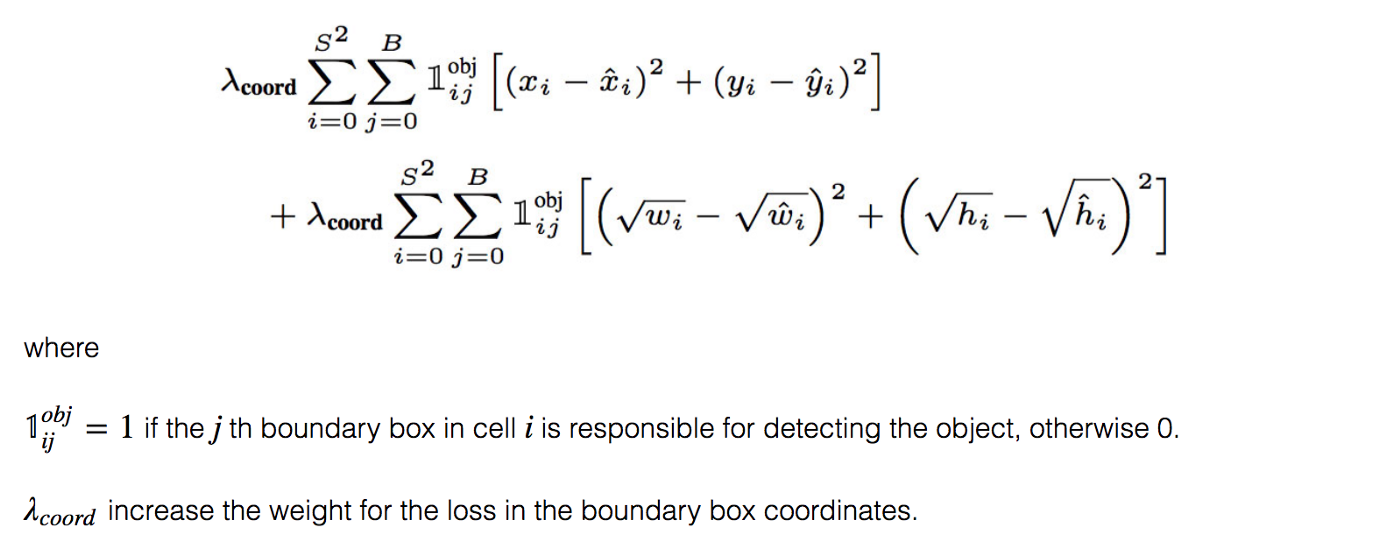
- Độ lỗi của ô vuông có chứa object nào hay không - Confidence Loss.

Hình 1.4: Công thức hàm mất mát.

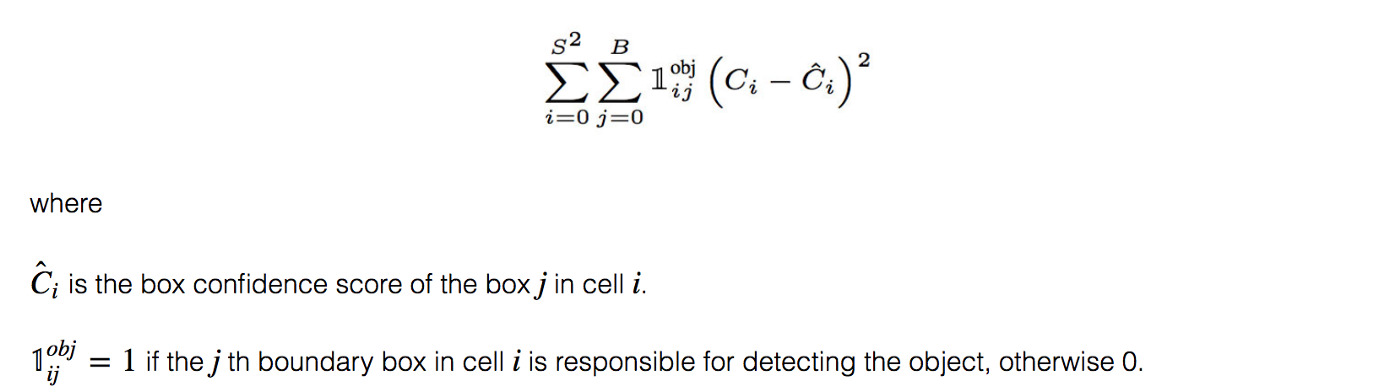
- Classification Loss:

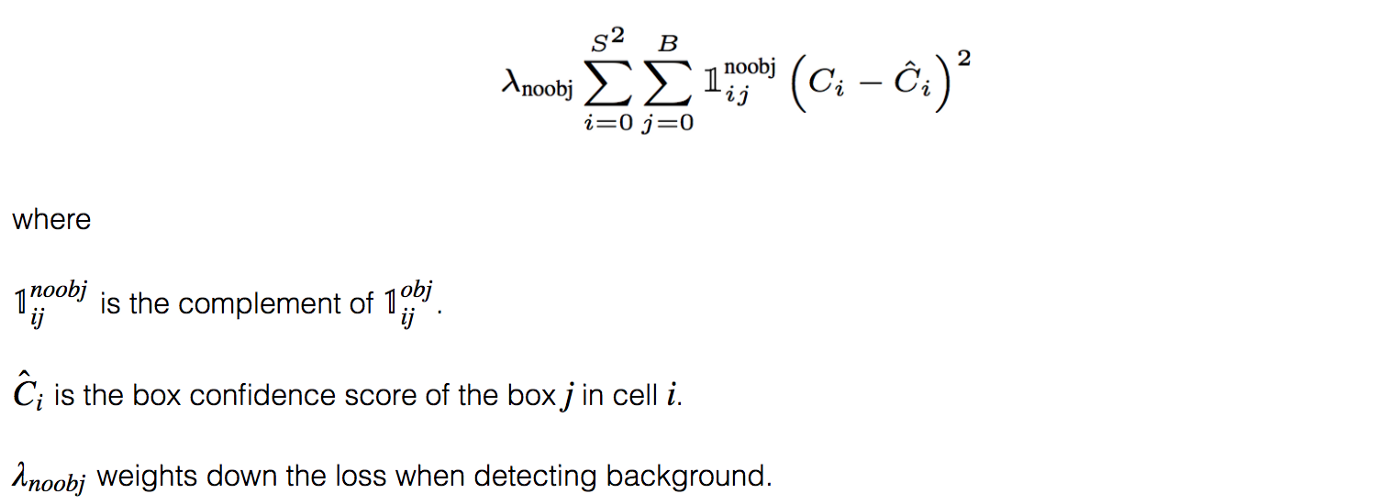
Hình 1.5: Hàm mất mát theo dự đoán nhãn của đối tượng.

- Localization Loss:

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

- Confidence Loss:

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

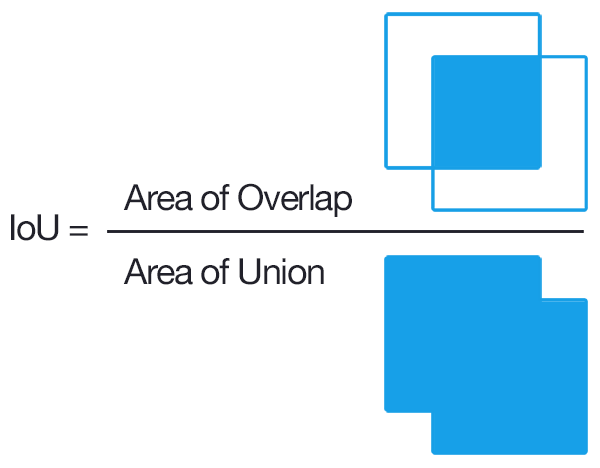
Hình 1.8: Hàm mất mát khi ô vuông không chứa đối tượng.

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

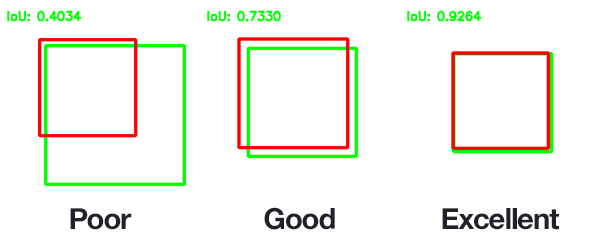
- Chỉ giữ lại những bouding box có chứa object nào đó. Tính tích xác suất có điều kiện của ô vuông thuộc về phân lớp i với xác xuất ô vuông chứa object, chỉ giữ lại những bounding box có giá trị này lớn hơn ngưỡng nhất định (threshold).

- Mỗi object lại có thế có nhiều boundary box khác nhau do mô hình dự đoán. Để tìm boundary box tốt nhất các object, chúng ta có thể dùng thuật toán non-maximal suppression để loại những boundary box giao nhau nhiều, tức là có IOU giữ 2 boundary box lớn.

- Để tính IOU giữ 2 box chúng ta cần tính diện tích giao nhau giữa 2 box chia cho tổng diện tích của 2 box đó.



Hình 1.9: Công thức tính IoU.

Hình 1.10: Phân cấp các mức độ IoU.

**2. YOLO v2.**

- YOLO v2 cho phép một mô hình có thể huấn luyện trên cả nhận dạng và phân lớp đối tượng. điểm chính trong mô hình là cải thiện Recall và Localization trong khi duy trì accuracy của classification. Mô hình YOLO v2 có những cải tiến giúp tiệm cận hơn Faster R-CNN và SSD.

- Kiến trúc mô hình:

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

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

- Những thay đổi chính:

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

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

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

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

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

- **Finer Features**: YOLO v2 sử dụng Grid System 13x13. Thêm một lớp passthrough mang lại các features từ lớp trước đó ở độ phân giải 26 × 26. Nó kết hợp các features độ phân giải cao với các features độ phân giải thấp bằng cách xếp chồng vào các kênh khác nhau thay vì các vị trí không gian. Điều này thay đổi features map từ 26x26x512 thành 13x13x2048. YOLO v2 tránh được nhược điểm của YOLO v1 do sử dụng Gird System 13x13, có thể detect được những đối tượng nhỏ hơn những vẫn hiệu quả với những đối tượng lớn.

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

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

**3. YOLO v3.**

- Các thuật toán Object Detection hiện nay chủ yếu cạnh tranh nhau về tốc độ và độ chính xác trong việc nhận dạng và phân loại các đối tượng trong ảnh. YOLO v3 là một sự lựa chọn hoàn hảo cho các hệ thống nhận dạng Real-time.

- **Bounding Box Prediction**: YOLO v3 tính điểm khách quan cho mọi Bounding Box sử dụng Logistic Regression. Thay đổi cách tính điểm. Nếu anchor trùng với khung của đối tượng thật sự lớn hơn hẳn các anchors khác, điểm số của nó sẽ là 1, với các anchor trùng với khung của đối tượng lớn hơn một ngưỡng xác định (thường là 0,5) thì sẽ không có giá trị điểm. Mỗi đối tượng thật sự chỉ có một sự liên kết với 1 anchor.

- **Class Prediction**: YOLO sử dụng hàm softmax để tính xác suất output của mạng về một đối tượng. YOLO v3 cải tiến có thể phân loại đa nhãn (multi-label classification). YOLO v3 sử dụng phân lớp logistic để xác định đầu ra mang một nhãn cụ thể. Sử dụng binary cross-erntropy để tính loss cho mỗi nhãn thay vì sử dụng mean sqare error. Giảm độ phức tạp của thuật toán bằng cách tránh sử dụng hàm softmax.

**- Feature Pyramid Networks (FPN) like Feature Pyramid:** YOLO v3 thực hiện 3 sự dự đoán với mỗi location. Mỗi dự đoán bao gồm một bounding box, một đối tượng và điểm số của 80 lớp. YOLO v3 thực hiện dự đoán với 3 tỉ lệ khác nhau (như FPN).

- Ở features map layer cuối cùng.

- Quay lại trước 2 layers và thực hiện upsamples với mỗi lớp (kernel = 2). Lấy features map với độ phân giải cao hơn kết hợp với features map đã upsample. Thực hiện dự đoán.

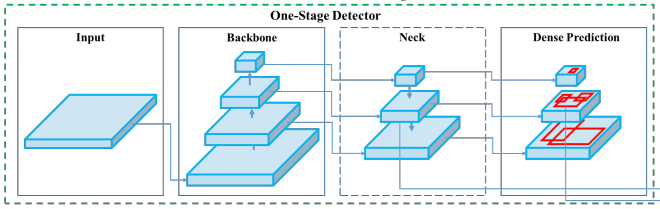
- Lặp lại 2 lần nữa để có features map với kết quả semantic (ngữ nghĩa) tốt và có độ phân giải tốt trên các vị trí đối tượng.

- Để xác định các anchor (priors), YOLO v3 sử dụng phân cụm k-means. Sau đó lựa chọn ra 9 cụm. Chia 9 cụm thành 3 nhóm theo tỉ lệ của từng anchor. Mỗi nhóm thực hiện detect objects cho từng features map trong 3 lần detect (FPN).

**- Features Extractor**.

**4. YOLO v4.**

**-** Tối ưu tốc độ và độ chính xác trong bài toán nhận dạng đối tượng.

Hình 4.1: Kiến trucsYOLO v4.

**- Backbone**: Các mô hình như ResNet, DenseNet, VGG, … được sử dụng để trích xuất đặc trưng. Các mô hình này được pre-trained bằng các tập ảnh phân lớp, và được tinh chinrg lại bằng các tập ảnh nhận dạng. Các mạng này cung cấp các level khác nhau của các đặc trưng, với các mức ngữ nghĩa khác nhau khi số lớp của mạng thay đổi, mạng càng sâu thì các mức ngữ nghĩa càng cao, tốt hơn cho việc nhận dạng các đối tương sau này.

- **Neck**: Là một lớp mở rộng ở giữa Backbone và lớp FC, dùng để extract các features map khác nhau của các state khác nhau trong backbone. Tương tư như YOLO v 3 dùng FPN để extract features map.

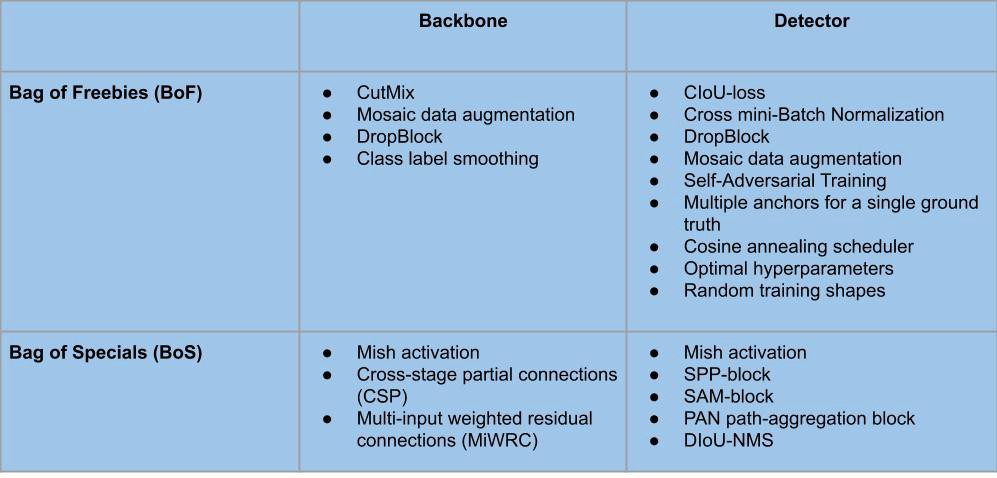
- **Head:** là lớp chính thực hiện nhận dạng (phân lớp và hồi quy) bounding box. Dự đoán về bounding box bao gồm các thông tin về chiều dài chiều rộng, và tọa độ so với ảnh. Thêm vào đó là dự đoán nhãn của k+1 lớp (tính cả lớp ko background).

- **Bag of freebies & Bag of Specials:**

- Đây là hai phương pháp được sử dụng để tăng độ chính xác của mô hình dự đoán.

- **bag of freebies (BoF):** là các phương pháp làm cho đối tươngj tốt hơn mà không cần phải suy luận, tính toán, các phương pháp này chỉ làm thay đổi cách học và tăng cường training. Một ví dụ của nó là Data, Augmentation: bằng cách thay đổi độ sáng, độ bão hòa, độ tương phản và nhiễu, hoặc có thể làm biến dạng hình học bằng cách cắt, xoay… Một cách khác đó là CutOut (che một phần đầu vào của ảnh), tương tự như Random Erasing, bằng cách chọn một vùng ảnh và xóa các giá trị một cách ngẫu nhiên. Hoặc sử dụng các kỹ thuật Regulaziration để tránh overfitting (DropOut, DropConnect, DropBlock).

- **bag of specials (BoS):** là các phương pháp và modules làm tăng mức độ suy luận lên một mức nhỏ nhưng làm cải thiện accuracy của mô hình. Các phương pháp được sử dụng: Cơ chế Attention (Squeeze and Excitation và Spatial Attention Module), các phương pháp carit hiện receptive field như SPP, ASPP, RFD, sử dụng Mish là một hàm kích hoạt (activation function).

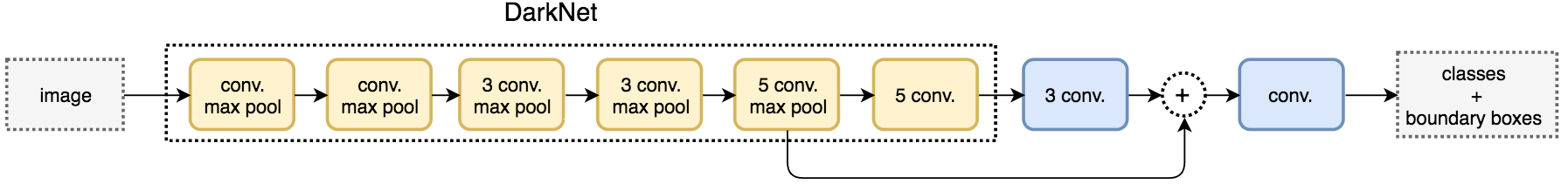
Hình 4.2: Tổng hợp các phương pháp, modules, BoF, BoS được sử dụng trong Backbone.

- Một phương pháp Data Augmentation mới được áp dụng là Mosaic, bằng cách gộp 4 images thành 1 images và đưa vào mỗi lớp.

**Câu hỏi 1:** Có thể sử dụng một mạng CNN khác (mô hình khác, hoặc custom), thay thế cho mạng CNN của YOLO để extract features hay không? Có thể sử dụng một backbone của mô hình khác Darknet cho YOLO hay không? Ví dụ: VGG, Resnes, GoogleNet,…

- Câu trả lời: **có thể** dùng một mạng CNN khác làm features exteaction layers cho YOLO.

- Kiến trúc của YOLO:

Kiến trúc YOLO dử dụng Darknet.

- Sử dụng DarkNet làm features extraction layers, các FC dùng để thực hiện phân lớp đối tượng và dự đoán box.

- Nguyên tắc chung khi thiết kế mạng CNN:

- Sử dụng nhiều Conv chồng lên nhau.

- Giảm dần kích thước output mỗi tầng.

- Tâng dần số lượng features map.

- Điểm khác biệt của YOLO là sử dụng kết hợp quá trình features extraction và, detection và localization trong một mạng xuyên suốt. Khác với SSD, sau mỗi lần extract features sẽ thực hiện phân lớp. Khác với các phương pháp học R-CNN là thực hiện dựa trên phương pháp Selective Search.

- Việc sử dụng các mạng CNN khác làm features extraction là hoàn toàn có thể. Tuy nhiên DarkNet được xây dựng riêng để huấn luyện cho YOLO. Đã có các so sánh về tốc độ và hiệu quả của các mạng khác so với DarkNet, bao gồm GoogleNet, chứng mình DarkNet là hiệu quả nhất. Do đó, khi sử dụng một CNN khác, hoặc custom một CNN mới là điều hoàn toàn có thể, nhưng hiệu quả thì cần so sánh voiwsDarkNet.

Tham khảo:

- <https://github.com/pjreddie/darknet/issues/323>

- <https://github.com/qqwweee/keras-yolo3/issues/449>

-<https://medium.com/@m.khan/implementing-yolo-using-resnet-as-feature-extractor-5857f9da5014>

-<https://github.com/schwittlick/ofxDarknet>

-<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6630625/>

**Câu hỏi 2:** Cơ chết hoạt động của down/up sampling trong huấn luyện mạng CNN.

**Câu hỏi 3:** Non-maximal supperession và cách giải quyết.