Trên các bài toán thực tế thì ngoài việc phân loại vật thể mà còn phải nhận diện được đâu là vật thể đang cần phát hiện. Có rất nhiều thuật toán có thể giải quyết được vấn đề trên chẳng hạn như:R-CNN series (R-CNN, Faster R-CNN, Fastest R-CNN) hay YOLOv1, YOLOv2,...Tất nhiên sẽ có tính ứng dụng hơn mô hình CNN. Mô hình được áp dụng cho bài toán “…..” là mô hình YOLO.

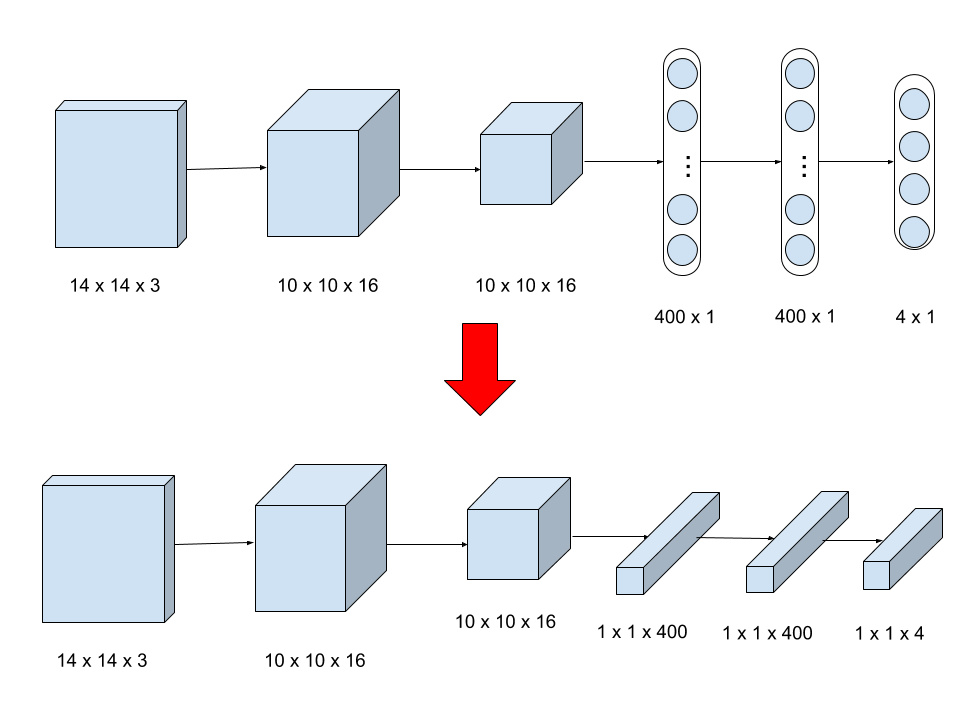
# 1. Sơ lược về thuật toán YOLO

YOLO được viết tắt từ "You Only Look Once", tạm dịch là "Bạn chỉ nhìn một lần".Thuật toán YOLO xem bài toán phát hiện vật thể là một vấn đề hồi quy duy nhất trên toàn bộ bức ảnh, trực tiếp từ các pixel của ảnh thành các ô dự đoán (bounding box) cùng với xác suất phân loại vật thể thay vì phải thực hiện nhiều bài toán hồi quy cho từng vùng vật thể như các thuật toán R-CNN series. Như vậy việc chỉ sử dụng 1 bài toán hồi quy duy nhất cho toàn bộ ảnh, thuật toán YOLO giúp giảm số lượng phép toán, tăng tốc độ xử lý khi đó có thể đáp ứng bài toán thời gian thực tốt hơn so với các thuật toán R-CNN.

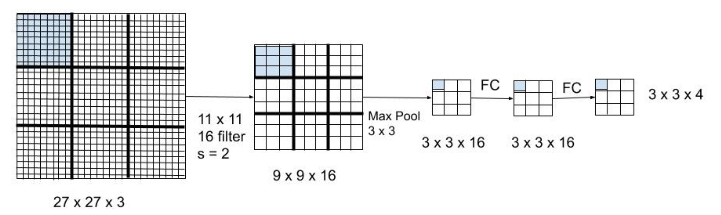
1.1 Grid System

Ý tưởng của việc thực hiện Grid System xuất phát từ việc thay vì thực hiện các lớp Fully Connected ở cuối mô hình thì các lớp này sẽ được chuyển thành các lớp tích chập như các lớp phía trước trong bài toán phân loại vật thể. Khi đó ở lớp cuối cùng thay cho lớp fully connected có kích thước (số classes, 1)  thành một lớp tích chập có kích thước 3 chiều (1,1,số classes)

Việc chuyển đổi này không làm ảnh hưởng đến kết quả dự đoán mà còn thể hiện được chúng ta hoàn toàn có thể phân loại vật thể bằng các phép toán tích chập với kết quả dự đoán nằm ở lớp tích chập cuối cùng và còn ưu việt hơn khi vẫn giữ được vị trí của vật thể. Ví dụ, hình ảnh chứa vật cần phân loại có kích thước 14 x 14 x 3, sau khi thực hiện các bước tích chập nhận được lớp tích chập cuối cùng có kích thước 1 x 1 x 4, chính lớp tích chập mang kết quả phân loại vật thể.

  
  
*Hình 1.1: Chuyển các lớp Fully Connected cuối thành các lớp tích chập*

Để chứng minh vị trí vật thể không thay đổi và vẫn có thể xác định được qua lớp cuối ta giả sử hình ảnh đang xét có kích thước 27 x 27 x 3 được chia thành 3 x 3 grid như hình dưới, vị trí của vật thể nằm ở ô có đánh màu, sau khi thực hiện các bước tích chập, ta thu được lớp tích chập cuối cùng có kích thước 3 x 3 x 3 .



*Hình 1.2: Vị trí và thông tin của vật thể được duy trì đến lúc cuối cùng*

*(Để đơn giản, trong hình trên không vẽ số chiều của lớp tích chập)*

Ta có thể thấy sau khi thực hiện tích chập dữ liệu và vị trí của vật thể được duy trì cho đến lớp cuối cùng, và ở ô màu tương ứng mang kết quả phân loại của vật thể. Như vậy ta vừa có thể phân loại vật thể vừa xác định được vị trí của vật thể.

Qua đó, grid system sẽ chia hình gốc thành số grid tương đương với kích thước của lớp cuối(không đề cập đến số chiều), như ví dụ trên lớp cuối có kích thước 3 x 3 vậy ta sẽ chia hình gốc thành 3 x 3 grid (đường kẻ đậm). Tại đó mỗi grid sẽ mang 3 thông tin chính: Grid có đang chứa vật thể hay không, tọa độ của các bounding box (gồm tọa độ x,y của góc trên bên trái và chiều dài, chiều rộng của bounding box), xác suất phân loại vật thể. Xét hình 1.3, giả sử chúng ta cần nhận diện biển báo cấm vượt quá 30km/h, mặc dù biển báo không nằm trọn trong một grid nhưng thuật toán sẽ chỉ xác định tâm vật thể (hình tròn trong hình dưới) và tâm vật thể nằm ở grid nào thì grid đó sẽ được xác định tồn tại vật thể. Ma trận của 1 grid như sau: [ pc , bx , by , bh , bw , c1 , c2 , c3 , ... ]

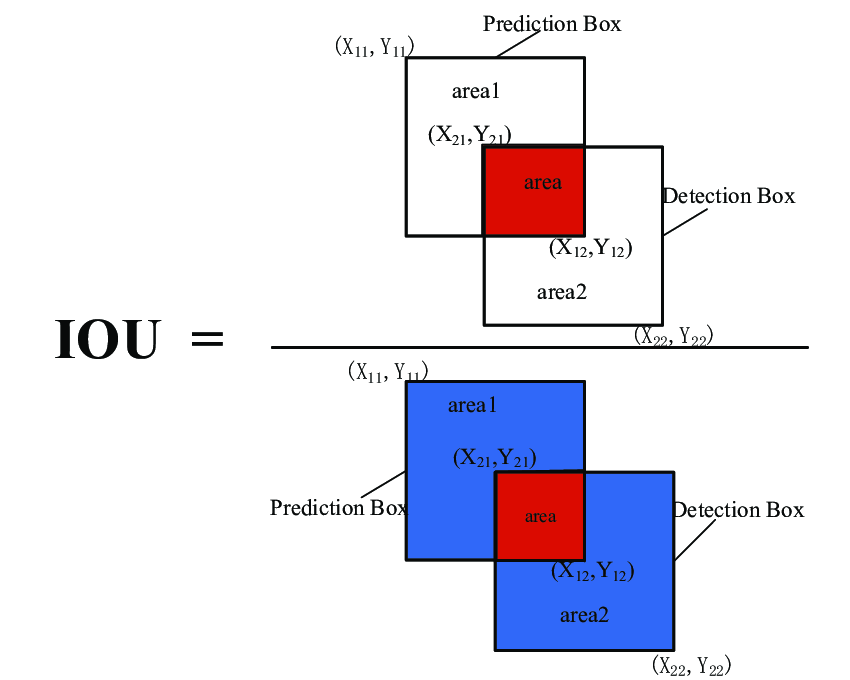


*Hình 1.3. Mô tả grid system trong bài toán nhận diện biển báo*

1.2.Khái niệm về chỉ số IoU và thuật toán Non-max suppression

Chỉ số IoU (Intersection over Union) cho ta biết tỉ lệ trùng vào nhau của 2 box.

 Trong đó  A B là phần giao nhau (Intersection) của box A và box B, A B là phần chung của 2 box (Union) bằng tổng diện tích của 2 box trừ đi phần giao nhau. Việc xác định IoU giúp tính toán khả năng phát hiện chính xác vật thể, trong đó box A thường là các anchor box ( hay groundtruth bounding box) được gán nhãn ở pha huấn luyện và box B là bounding box của hệ thống xác định ở pha kiểm tra. Tính toán IoU để đánh giá mô hình đã phát hiện vật thể đúng hay chưa.



*Hình 1.4: Phần giao nhau giữa 2 box A và B*

Trong pha kiểm tra, hệ thống sẽ đưa ra nhiều bounding box khác nhau với các xác suất dự đoán khác nhau và tỉ số IoU khác nhau, vì vậy thuật toán Non-max suppression giúp loại bỏ các bounding box có tỷ lệ dự đoán thấp và chỉ giữ lại 1 bounding box cuối cùng có tỷ lệ dự đoán cao nhất. Thuật toán Non-max suppression diễn ra như sau:

Bước 1: Loại bỏ tất cả các bounding box có xác suất xuất hiện của vật thể pc thấp hơn ngưỡng. Việc loại bỏ như vậy để các grid không chứa vật thể có xác suất xuất hiện của vật thể thấp sẽ không hiển thị bounding box.

Bước 2: Chọn các bounding box có xác suất xuất hiện vật thể cao nhất.

Bước 3: Nếu có nhiều bounding box có cùng xác suất xuất hiện vật thể cao nhất thì ta sẽ loại bỏ bằng IoU, bounding box nào có chỉ số IoU thấp hơn ngưỡng sẽ bị loại bỏ. Kết thúc bước ba, ta sẽ nhận được bounding box có tỉ lệ nhận diện vật thể tốt nhất.

1.3 Gán nhãn cho các mẫu

Tương tự các bài toán "Máy học"/"Học sâu", bài toán nhận diện vật thể cũng gồm có 2 pha, pha kiểm tra và pha huấn luyện. Trong pha học của thuật toán YOLO, chúng ta sẽ thực hiện label vị trí và phân loại cho vật thể, khi đó y có dạng như sau: y = [pc , bx , by , bh , bw , c1 , c2 , c3,...]

Ví dụ đối với hình 1.3 giả sử bài toán cần tìm vị trí vật thể và phân loại với 3 nhãn, trong đó biển báo của hình thuộc nhãn thứ nhất. Khi đó, ở grid phát hiện được biển báo giao thông, y sẽ được gán như sau y = [1 , 230 , 120 , 20 , 30 , 1 , 0 , 0], tức là grid đó có vật thể, tọa độ (x,y) ở phía trên bên trái của anchor box là (230,120), anchor box có chiều dài là 20 chiều rộng là 30, biển thuộc nhãn 1 và không thuộc hai nhãn còn lại. Nếu xét grid không chứa vật thể y sẽ được gán y = [0 , x , x , x , x , x , x , x ] tức là không có vật thể ở grid đó và các giá trị còn lại không cần quan tâm.

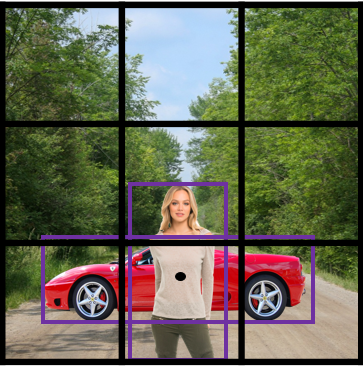
Tuy nhiên, nếu có cùng 2 vật thể cùng xuất hiện trong 1 grid thì không thể thực hiện gán y như trên. Với vấn đề đó, chúng ta có thể xử lý theo 2 cách:

Cách 1: Chia nhỏ grid ra đến khi nào 2 vật thể nằm ở 2 grid khác nhau. Tuy nhiên nếu chia càng nhỏ grid, việc học không thể diễn ra sâu khiến việc phân loại vật thể khó chính xác do các lớp sau không học được các đặc trưng cao.Và nếu tâm của 2 vật thể gần như trùng nhau cũng không thể giải quyết được như trên. Khi đó, ta phải thực hiện theo cách thứ 2.

Cách 2: Thay vì y chỉ được gán cho 1 vật thể, y sẽ được mở rộng ra với nhiều vật thể như sau:

y =[pc1 , bx1 , by1 , bh1 , bw1 , c1 , c2 , c3,...,pc2 , bx2 , by2 , bh2 , bw2 , c1 , c2 , c3,...]

Xét ví dụ với hình 2.16 bên dưới, cô gái và chiếc xe đều nằm cùng chung 1 grid. Khi đó, y sẽ được gán như sau y = [1, 120, 20, 20 , 120 , 1 , 0 , 1 , 90, 50 ,  90 , 20 , 0 , 1] . Tức  là ở grid này có xuất hiện ô tô (y[0] = 1), (120,20,20,120) là 4 thông số anchor box của xe ô tô, (y[5]=1,y[6]=0) để phân loại cho anchor box này là xe ô tô chứ không phải cô gái, tương tự y[7]=1 có nghĩa grid này cũng có cô gái, 4 thông số tiếp theo để xác định tọa độ anchor box cho cô gái cà (y[12]=0,y[13]=1) để phân loại anchor box này là cô gái chứ không phải xe ô tô. Như vậy, với cách 2 ta sẽ ghép y của 2 hay nhiều vật thể nằm trong cùng 1 grid lại thành một, y[0:6] để xác định cho việc phát hiện chiếc xe, và y[7:13] để xác định cho cô gái. Tuy nhiên với cách thứ 2, nếu ta ghép càng nhiều tốc độ xử lý càng lâu do càng có nhiều phép toán phép thực hiện, vì vậy không nên quá lạm dụng cách này mà nên phối hợp hài hòa với cách thứ 1, tăng số grid phải chia lên.



*Hình 1.5 Tâm của 2 vật thể trùng nhau và cùng nằm trong 1 grid.*

*Nguồn : reddit.com*

1.4.Hàm mất mát (Loss function)

Sau khi gán nhãn cho toàn bộ tập dữ liệu, các mẫu dữ liệu sẽ được đưa qua mạng CNN (toàn bộ ảnh sẽ đưa vào mạng CNN) để thực hiện việc học các tham số. Trong quá trình này, các hàm mất mát sẽ được tính toán. Đối với bài toán nhận diện vật thể sẽ cần tính toán 3 hàm mất mát. Hàm lỗi trong YOLO được tính trên việc dự đoán và mô hình để tính . Cụ thể là nó tính tổng sai số của 3 thành phần con sau: Độ lỗi của việc dự đoán nhãn của Object- *Classification loss*; Độ lỗi của dự đoán tọa độ tâm, chiều dài, chiều rộng của boundary box (x,y,w,h) – *Localization loss*; Độ lỗi của việc dữ đoán bounding box đó chứa object so với nhãn thực tế tại ô vuông đó – *Confindent Loss*.

### Hàm mất mát phân loại (Classification Loss Function):

Classification loss - độ lỗi của việc dự đoán loại nhãn cuả object, hàm lỗi này chỉ tính trên những ô vuông có xuất hiện object, còn những ô vuông khác ta không quan tâm. Classification loss được tính bằng công thức sau:

 (1)

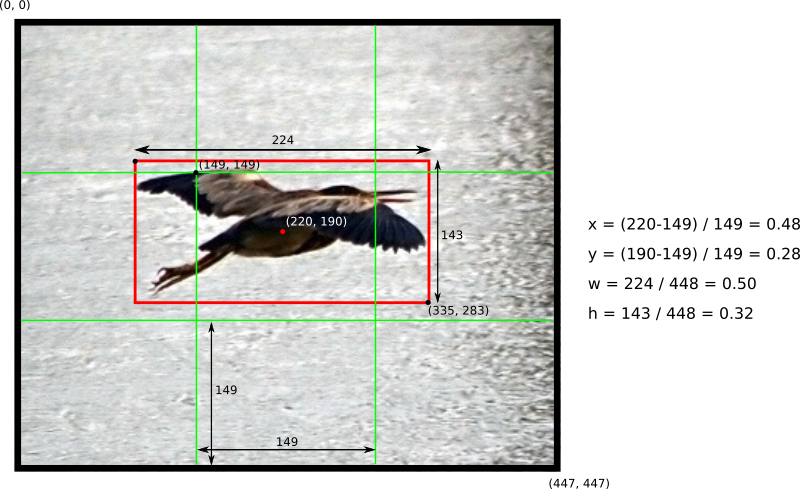
Trong đó:

: bằng 1 nếu ô vuông đang xét có object ngược lại bằng 0.

: là xác xuất có điều kiện của lớp c tại ô vuông tương ứng mà mô hình dự đoán.

### Hàm mất mát vị trí (Localization Loss Function)

Localization loss là hàm lỗi dùng để tính giá trị lỗi cho boundary box được dự đoán bao gồm tọa độ tâm, chiều rộng, chiều cao của so với vị trí thực tế từ dữ liệu huấn luyện của mô hình. Lưu ý rằng chúng ta không nên tính giá trị hàm lỗi này trực tiếp từ kích thức ảnh thực tế mà cần phải chuẩn hóa về [0, 1] so với tâm của bounding box. Việc chuẩn hóa này kích thước này giúp cho mô hình dự đoán nhanh hơn và chính xác hơn so với để giá trị mặc định của ảnh. Hãy cùng xem một ví dụ:



*Hình 1.6. Hàm Localization loss*

Giá trị hàm Localization loss được tính trên tổng giá trị lỗi dự đoán toạ độ tâm (x, y) và (w, h) của predicted bounding box với grouth-truth bounding box. Tại mỗi ô có chưa object, ta chọn 1 boundary box có IOU (Intersect over union) tốt nhất, rồi sau đó tính độ lỗi theo các boundary box này.

Giá trị hàm lỗi dự đoán tọa độ tâm (x, y) của predicted bounding box và (x̂, ŷ) là tọa độ tâm của truth bounding box được tính như sau:

 (2)

Giá trị hàm lỗi dự đoán (w, h) của predicted bounding box so với truth bounding box được tính như sau:

 (3)

Với ví dụ trên thì S =7, B =2, còn λcoord là trọng số thành phần trong paper gốc tác giả lấy giá trị là 5.

### Hàm mất mát dự đoán (Confidence Loss Function)

Confidence loss là độ lỗi giữa dự đoán boundary box đó chứa object so với nhãn thực tế tại ô vuông đó. Độ lỗi này tính trên cả những ô vuông chứa object và không chứa object.

 (4)

Với ví dụ trên thì S =7, B =2, còn λnoobject là trọng số thành phần trong paper gốc tác giả lấy giá trị là 0.5. Đối với các hộp j của ô thứ i nếu xuất hiệu object thì Ci =1 và ngược lại.

### Total loss

Tổng lại chúng ta có hàm lỗi là tổng của 3 hàm lỗi trên

 (5)

1.5.Ngõ ra

Ở ngõ ra, mỗi grid sẽ thực hiện dự đoán 2 bounding box có pc cao nhất. Loại bỏ tất cả bounding box có pc thấp trên toàn bộ bức ảnh. Và cuối cùng thuật toán Non-max suppression sẽ được thực hiện để giữ lại bounding box chính xác nhất cho từng vật thể.



(a) (b) (c)

*Hình 1.7: Thuật toán được áp dụng ở ngõ ra*

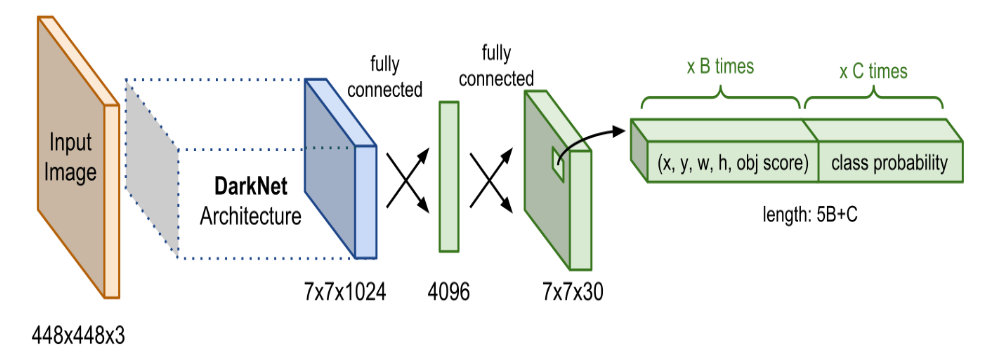
*(a): Với mỗi grid, lấy 2 bounding box có pc cao nhất.*

*(b): Loại bỏ tất cả các bounding box có pc thấp trên toàn bộ bức ảnh, bước này giúp loại bỏ các grid không chứa vật thể.*

*(c): Áp dụng non-max suppression để chỉ chọn ra bounding-box cuối cùng có xác suất dự đoán cao nhất.*

# 2. Kiến trúc YOLO v3

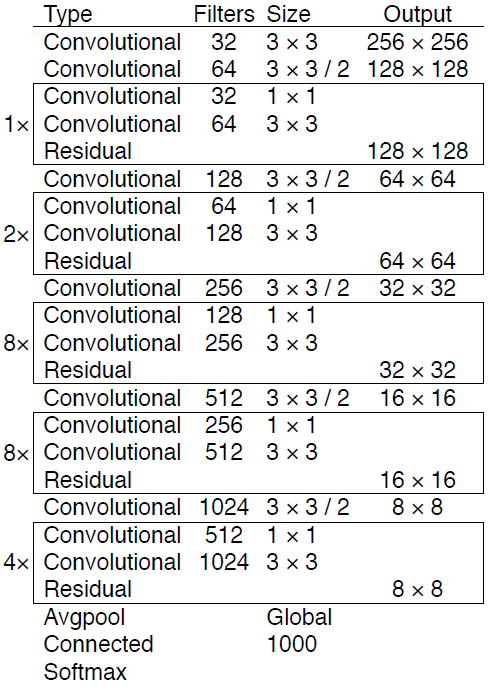
Kiến trúc YOLO bao gồm: base network là các mạng convolution làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những Extra Layers được áp dụng để phát hiện vật thể trên feature map của base network.



*Hình 2.1. Kiến trúc mạng YOLO*

 Thành phần Darknet Architechture được gọi là base network có tác dụng trích suất đặc trưng. Output của base network là một feature map có kích thước 7x7x1024 sẽ được sử dụng làm input cho các Extra layers có tác dụng dự đoán nhãn và tọa độ bounding box của vật thể.

YOLOv3 có một vài cải tiến gia tăng trên YOLOv2. Trong YOLO version 3 tác giả áp dụng một mạng feature extractor là darknet-53. Mạng này gồm 53 convolutional layers kết nối liên tiếp, mỗi layer được theo sau bởi một batch normalization và một activation Leaky Relu. Để giảm kích thước của output sau mỗi convolution layer, tác giả down sample bằng các filter với kích thước là 2. Mẹo này có tác dụng giảm thiểu số lượng tham số cho mô hình.



***Hình 2.2.*** *Các layer trong mạng darknet-53*

Các bức ảnh khi được đưa vào mô hình sẽ được scale để về chung một kích thước phù hợp với input shape của mô hình và sau đó được gom lại thành batch đưa vào huấn luyện. Hiện tại YOLO đang hỗ trợ 2 đầu vào chính là 416x416 và 608x608. Mỗi một đầu vào sẽ có một thiết kế các layers riêng phù hợp với shape của input. Sau khi đi qua các layer convolutional thì shape giảm dần theo cấp số nhân là 2. Cuối cùng ta thu được một feature map có kích thước tương đối nhỏ để dự báo vật thể trên từng ô của feature map. Kích thước của feature map sẽ phụ thuộc vào đầu vào. Đối với input 416x416 thì feature map có các kích thước là 13x13, 26x26 và 52x52. Và khi input là 608x608 sẽ tạo ra feature map 19x19, 38x38, 72x72.

Output của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần:

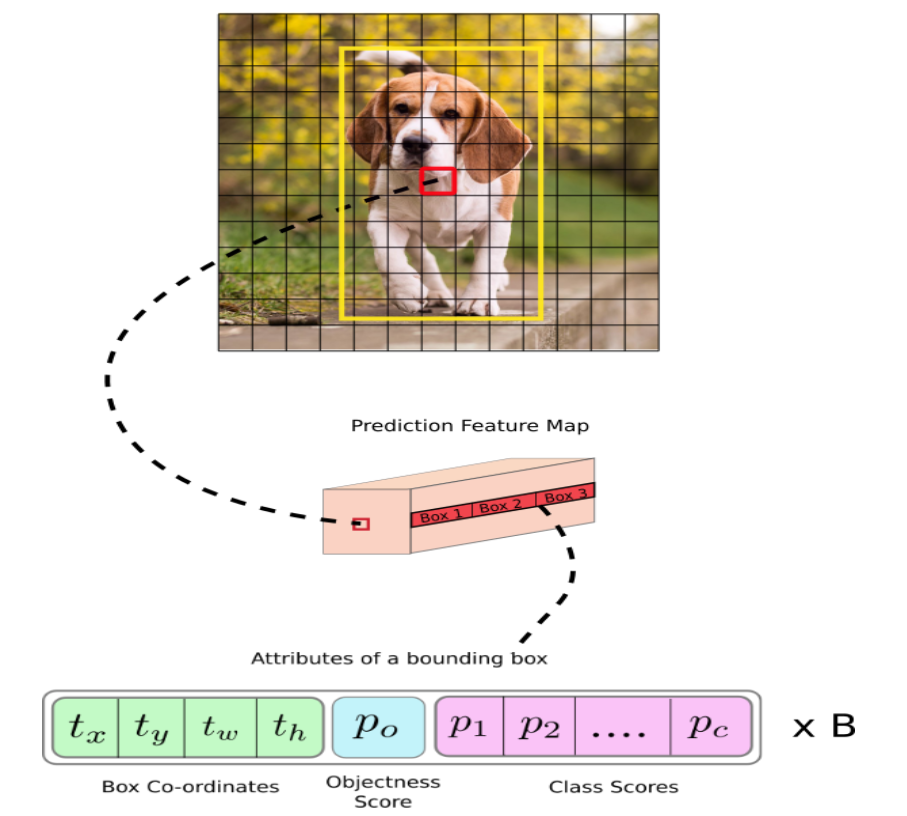
 (6)

Trong đó:

* p0 là xác suất dự báo vật thể xuất hiện trong bounding box.
* giúp xác định bounding box. Trong đó là tọa độ tâm và  là kích thước rộng, dài của bounding box.
*  là véc tơ phân phối xác suất dự báo của các classes.

Việc hiểu output khá là quan trọng để chúng ta cấu hình tham số chuẩn xác khi huấn luyện model qua các open source như darknet. Như vậy output sẽ được xác định theo số lượng classes theo công thức (n\_class+5). Nếu huấn luyện 80 classes thì sẽ có output là 85. Trường hợp áp dụng 3 anchors/cell thì số lượng tham số output sẽ là:



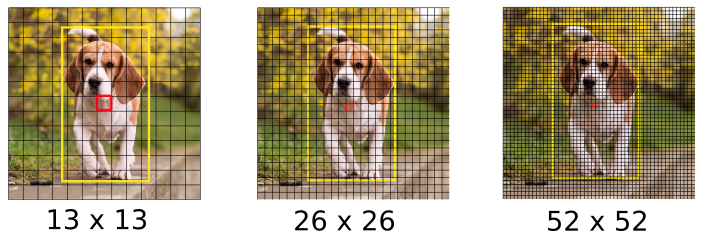


*Hình 2.3 Kiến trúc một output của model YOLO*

Hình ảnh gốc là một feature map kích thước 13x13. Trên mỗi một cell của feature map chúng ta lựa chọn ra 3 anchor boxes với kích thước khác nhau lần lượt là Box 1, Box 2, Box 3 sao cho tâm của các anchor boxes trùng với cell. Khi đó output của YOLO là một véc tơ concatenate của 3 bounding boxes. Các attributes của một bounding box được mô tả như dòng cuối cùng trong hình 3.8.

**Dự báo trên nhiều feature map**

YOLOv3 dự báo trên nhiều feature map. Những feature map ban đầu có kích thước nhỏ giúp dự báo được các object kích thước lớn. Những feature map sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo các vật thể kích thước nhỏ.



*Hình 2.4. Các feature maps của mạng YOLOv3*

*(với input shape là 416x416, output là 3 feature maps có kích thước lần lượt là 13x13, 26x26 và 52x52)*

Trên mỗi một cell của các feature map áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 featue map x 3 anchor box). Đồng thời trên một feature map hình vuông S x S, mô hình YOLOv3 sinh ra một số lượng anchor box là: S x S x 3. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là:

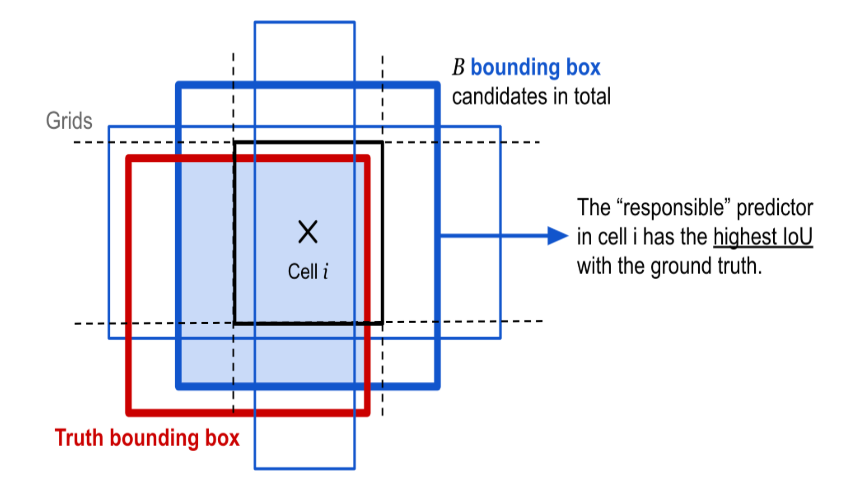
(13×13+26×26+52×52) × 3=10647(anchor boxes)

Đây là một số lượng rất lớn và là nguyên nhân khiến quá trình huấn luyện mô hình YOLO vô cùng chậm bởi cần dự báo đồng thời nhãn và bounding box trên đồng thời 10647 bounding boxes.

**Anchor box**

Để tìm được bounding box cho vật thể, YOLO sẽ cần các anchor box làm cơ sở ước lượng. Những anchor box này sẽ được xác định trước và sẽ bao quanh vật thể một cách tương đối chính xác. Thuật toán regression bounding box sẽ tinh chỉnh lại anchor box để tạo ra bounding box dự đoán cho vật thể. Trong một mô hình YOLO, 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.

Từ Cell ta xác định được 3 anchor boxes viền xanh như trong hình bên dưới. Cả 3 anchor boxes này đều giao nhau với bounding box của vật thể. Tuy nhiên chỉ anchor box có đường viền dày nhất màu xanh được lựa chọn làm anchor box cho vật thể bởi nó có IoU so với ground truth bounding box là cao nhất. Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một cell trên feature map mà chứa điểm mid point của vật thể. Chẳng hạn như hình chú chó trong hình 3.9 sẽ được phân về cho cell màu đỏ vì điểm mid point của ảnh chú chó rơi vào đúng cell này. Từ cell ta sẽ xác định các anchor boxes bao quanh hình ảnh chú chó.



*Hình 2.5 Xác định anchor box cho một vật thể.*

Như vậy khi xác định một vật thể ta sẽ cần xác định 2 thành phần gắn liền với nó là (cell, anchor box). Không chỉ riêng mình cell hoặc chỉ mình anchor box. Một số trường hợp 2 vật thể bị trùng mid point, mặc dù rất hiếm khi xảy ra, thuật toán sẽ rất khó xác định được class cho chúng.

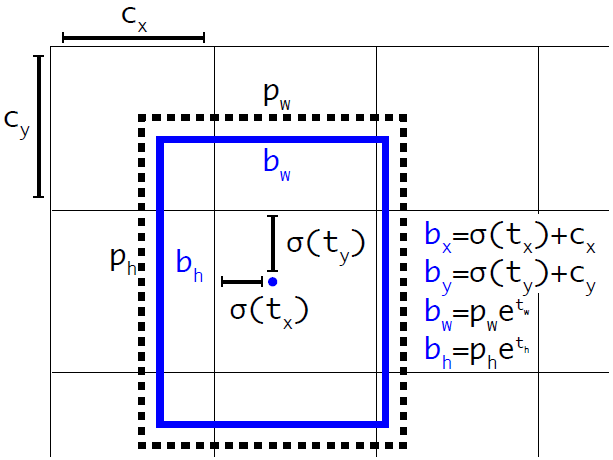
**Hàm loss function**

Như đã trình bày trong phần tổng quan về YOLO các hàm loss function. YOLOv3 nhìn chung không thay đổi gì về cách tính và xác định hàm loss. Bao gồm: (localization loss) đo lường sai số của bounding box (là hàm mất mát của bounding box dự báo so với thực tế.) và (confidence loss) đo lường sai số của phân phối xác suất các classes (là hàm mất mát của phân phối xác suất. Trong đó bao gồm 2 tổng: Tổng đầu tiên là mất mát của dự đoán có vật thể trong cell hay không? Và tổng thứ 2 là mất mát của phân phối xác suất nếu có vật thể trong cell.)

**Dự báo bounding box**

Để dự báo bounding box cho một vật thể chúng ta dựa trên một phép biến đổi từ anchor box và cell. YOLOv3 dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều. Nếu bounding box dự đoán có thể đặt vào bất kỳ phần nào của hình ảnh, như trong mạng regional proposal network, việc huấn luyện mô hình có thể trở nên không ổn định.

Cho một anchor box có kích thước (pw,ph) tại cell nằm trên feature map với góc trên cùng bên trái của nó là (cx,cy), mô hình dự đoán 4 tham số (tx,ty,tw,th) trong đó 2 tham số đầu là độ lệch (offset) so với góc trên cùng bên trái của cell và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định bounding box dự đoán b có tâm (bx,by) và kích thước (bw,bh) thông qua hàm sigmoid và hàm exponential như các công thức hình bên dưới:



*Hình 2.6 Công thức ước lượng bounding box từ anchor box*

Ngoài ra do các tọa độ đã được hiệu chỉnh theo width và height của bức ảnh nên luôn có giá trị nằm trong ngưỡng [0, 1]. Do đó khi áp dụng hàm sigmoid giúp ta giới hạn được tọa độ không vượt quá xa các ngưỡng này.

Hình chữ nhật nét đứt bên ngoài là anchor box có kích thước là (pw,ph) .Tọa độ của một bounding box sẽ được xác định dựa trên đồng thời cả anchor box và cell mà nó thuộc về. Điều này giúp kiểm soát vị trí của bounding box dự đoán đâu đó quanh vị trí của cell và bounding box mà không vượt quá xa ra bên ngoài giới hạn này. Do đó quá trình huấn luyện sẽ ổn định hơn rất nhiều so với YOLO version 1.

# 3. Một số lưu ý khi huấn luyện YOLO

Khi huấn luyện YOLO sẽ cần phải có RAM dung lượng lớn hơn để save được 10647 bounding boxes như trong kiến trúc này.

Không thể thiết lập các batch\_size quá lớn như trong các mô hình classification vì rất dễ Out of memory. Package darknet của YOLO đã chia nhỏ một batch thành các subdivisions cho vừa với RAM.

Thời gian xử lý của một step trên YOLO lâu hơn rất rất nhiều lần so với các mô hình classification. Do đó nên thiết lập steps giới hạn huấn luyện cho YOLO nhỏ. Đối với các tác vụ nhận diện dưới 5 classes, dưới 5000 steps là có thể thu được nghiệm tạm chấp nhận được. Các mô hình có nhiều classes hơn có thể tăng số lượng steps theo cấp số nhân.