Yolo

Yolo divides the image into grid cells. Each grid cell predicts only one object. we will assign center of that object to that grid cell. So this grid cell is responsible for predicting the object

(only convolutional layers) no pooling

For each grid cell we predict $\underbrace{n \times n \times 8}$ $n \to$ no. of classes

$P_c \to$ confidence value

$\left.\begin{matrix} b_x \\ b_y \\ b_h \\ b_w \end{matrix}\right\} \to$ bounding box of object if there is.

$\left.\begin{matrix} C_1 \\ C_2 \\ C_3 \end{matrix}\right\}$ Say we have 3 classes.

$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \to$ if object not present

$3 \times 3 \times 8 \to$ feature map prediction

(0,0) $\to x \to$ (0-1)

$y \to$ (0-1)

What if more than object's centre lies in same grid cell?

Here comes Anchor Boces as there is limitation with only having grid cells.

To solve problem we will introduce the concept of anchor box which makes yolo to predict multiple objects centered in one cell.

Different objects have different shapes so we use anchor box of different shapes.

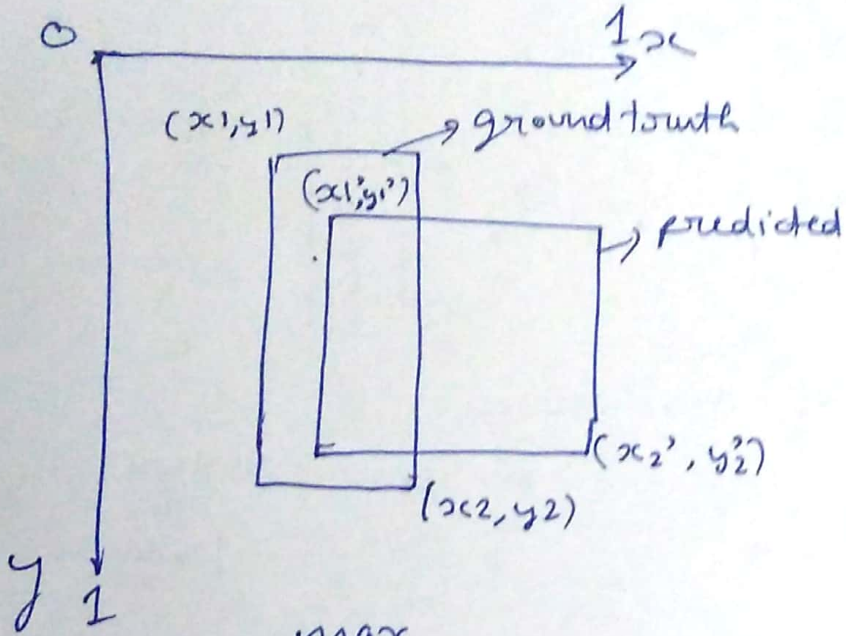
$(N \times N) \times$ [num ~~class~~ anchors $\times$ (5 + no. of classes)]

$N \times N$ grid cells   say we have 3X3 grid cells.

IOU for evaluation
& for inference.

Instead of defining box by center, width & height we define it using two corners (upper left & bottom right)

$$IOU = \frac{\text{Intersection}}{\text{Union}}.$$



max

Non max suppression

used for cleaning up when multiple grid cells are predicted the same object.

Steps!

- Discard all boxes with pc ~~less~~ ≤ 0.6
- Pick the box with largest pc output as prediction
- Discard any remaining box with IOU >= 0.5

~~So~~

Note that given an input image with 3x3 grid cells 2 anchor boxes. each grid cell will have 2 predictions even for those grid cells that don't have any object inside.

In this case we will filter by class scores. threshold

pc bx by bw bh c1 c2 c3

$$\begin{bmatrix} pc \times c1 \\ pc \times c2 \\ pc \times c3 \end{bmatrix}$$

☆ fully convolutional

- no pooling to downsample convolution layer with stride 2 is used
  → to prevent low-level features

1

In yolo v3 we don't predict one feature map. We have 3 of these.

different scales

13x13 26x26 52x52

In yolo v1 & v2 they only have 1

416x416

(32, 16, 8) → stride

☆ Its very helpful for predicting small objects.

3 bounding boxes / anchor for each scale

80 classes

→ coco dataset { object detection, image captioning.

Prediction done by convolution layer 1x1 conv. reduce feature map depth to

output → feature map

Depth wise entries in feature map

$(B \times (5 + C))$

1x1 conv

$(N \times N) \times (\text{num anchors} \times (5 + \text{classes}))$

e.g. $\underbrace{13 \times 13}_{\text{size}} \times (3 \times (5 + 80))$

$26 \times 26 \times (3 \times (5 + 80))$

Prediction of Anchor Boxes

→ predict width & height of bounding box but it leads to unstable gradients.

→ so we use offset (how much we should move in order to predict desired bounding box) to pre-defined default bounding box.

→ yolo v3 has 3 anchors which results in predicting 3 bounding boxes per cell.

t_x, t_y, t_w, t_h (offset of centre)

(c_x, c_y) → top left co-ordinate of grid.

(to get value b/w 0 & 1) ← σ → sigmoid

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

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

basically adding offset to ~~left~~ top left co-ordinate of grid.

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

$$b_h = p_h e^{t_h}$$

p_w, p_h → anchor dimension of box

t_w, t_h → offsets for width & height

Scanned by CamScanner

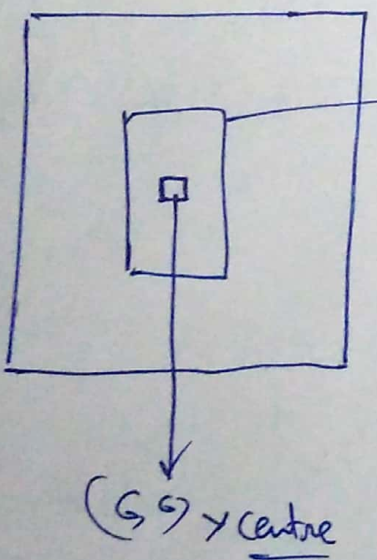
The resultant predictions b_w & b_h are normalised by height & width of image.

To get them back to original feature map shape multiply by grid that we are ~~predicting~~ using.

Say we are using 13 X 13 feature map then we need to multiply by 13

Denormalize the prediction

We are passing the predict centre offsets with Sigmoid to get value b/w 0 & 1.



$\sigma(t_x)$ $\sigma(t_y)$

0.4 0.7

0.4 + 6 = 6.4

0.7 + 6 = 6.7

if not passed

1.2 & 0.7

6 + 1.7 → 7.7 ✗ out of bound

0.7 + 6 = 6.7

to keep the center in the grid which is predicting.

(2)

yolov1 & v2 uses softmax → as they assume that classes are mutually exclusive.

i.e. if object belongs to one class then it can't belong to another class.

(sum up to 1)

yolov3 → sigmoid.

So each class score is predicted using logistic regression & a threshold is used to predict multiple labels for an object.

106 layers fully convolution.

residual to restore spatial info that's lost

79	91	106
13×13	26×26	52×52

$\frac{416}{32}$ → stride

to get predicted feature map size.

$\frac{416}{16}$ → stride ← $\frac{416}{8}$

Total no. of bounding boxes yolo predicts.

$$(13 \times 13 \times 3) + (26 \times 26 \times 3) + (52 \times 52 \times 3)$$
$$= 10647$$

yolov3 → 9 anchor boxes.
3 for each scale

Loss

loss { classification, localization, confidence

v2

Center co-ordinates

width/height of bounding box

class confidence (object) → 1 $\sum\sum 1_{ij}^{obj} (C_i - \hat{C}_i)^2$

no object → 0

classes → classification loss ← only penalizes when there is an object

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

To compute loss for true positive we only one of them to be responsible $\sum$ for the object. For this we consider one with highest IOU.

Penalizes the objectness score prediction for bounding box responsible for predicting obj. ideally 1

penalizes for bounding box having no object (ideally 0)

~~[illegible]~~

~~[illegible]~~

Classification loss → If object detected classification loss at each cell is → Squared error of the class conditional probabilities for each class.

$$\sum_{i=0}^{S^2} \underbrace{1_i^{obj}}_{\text{= 1 if object appears in cell } i \text{ otherwise 0.}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Localization loss

measures errors in the predicted boundary box locations and sizes. We only take box responsible for detecting the object.

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

$1_{ij}^{obj} = 1$ if the jth boundary box in cell i is responsible for detecting object. otherwise 0

$$+$$

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

We don't want absolute errors in large boxes & small boxes. So yolo predicts square root of bounding box width & height.

To put more emphasis on boundary box accuracy we multiply the loss by λ_{coord} (default: 5)

Confidence loss

If object is detected in the box

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

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

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

If object is not detected:

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

\# Most boxes don't contain any objects. This causes class imbalance problem i.e. we train the model to detect background more frequently than detecting objects. So we weight this loss down by a factor λ_{noobj} (default: 0.5)

The last 3 terms in Yolo v2 are the squared errors, whereas in yolo v3, they have been replaced by cross entropy ~~loss~~ error terms.
In classification loss, instead of using mean squared error yolov3 uses bce loss for each label. Also reducing the computation complexity by avoiding the softmax function