Hyeonwoo Yoo Tuesday, 30 October 2018

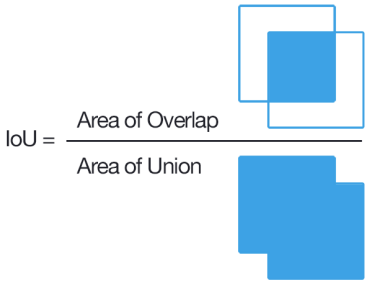
YOLO(You Only Look Once)

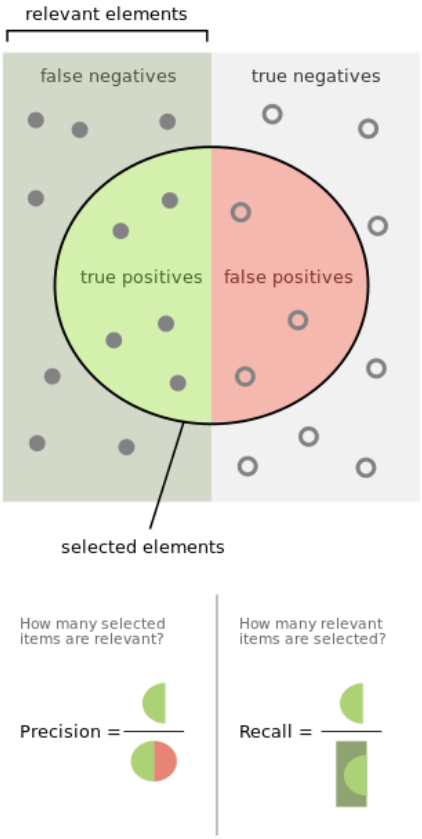
PASCAL VOC DATASET

- Dataset of more than 10,000 images which contains several objects in the image for testing classification, detection, segmentation performances
- It requires to submit multiple confidence score, which is possibility of existence of specific object class in the image -> for further analysis

Classification, Detection, Segmentation

- Classification : Whether required image exists in the image
- Detection : When image contains specific object, detect where it is as a bounding box
- Segmentation : Similar with detection, but more detail. It returns every pixel coordinates of object
- GT(Ground Truth) : It is given as a label of training data, which is the information of bounding box
- IOU(Intersection over union) : Metrics will be used to determine how accurate it detects the bounding box





- Precision : # of successfully classified image as class A / # of images predicted to be in class A
- Recall : # of successfully classified images as class A / entire number of images in class A

Short history of previous models

- DPM : using sliding window approach where the classifier is run at evenly spaced locations over the entire images -> slow, requires lot of computation
- R-CNN : first generate potential bounding boxes -> run a classifier on these proposed boxes -> slow and hard to optimize because each individual component must be trained separately

Differentiated points from previous models

- This is a single regression problem. Straight from image pixels to bounding box coordinates and class probabilities -> This is why it's called 'You Only Look Once'

Pros they claim

- extremely fast : because it doesn't require a complex pipeline
- it doesn't lose the context : since it sees entire image during the training and test time, contextual information is expected to be encoded in the model unlike R-CNN.
- it learns generalizable representations of objects : it even works with artwork.

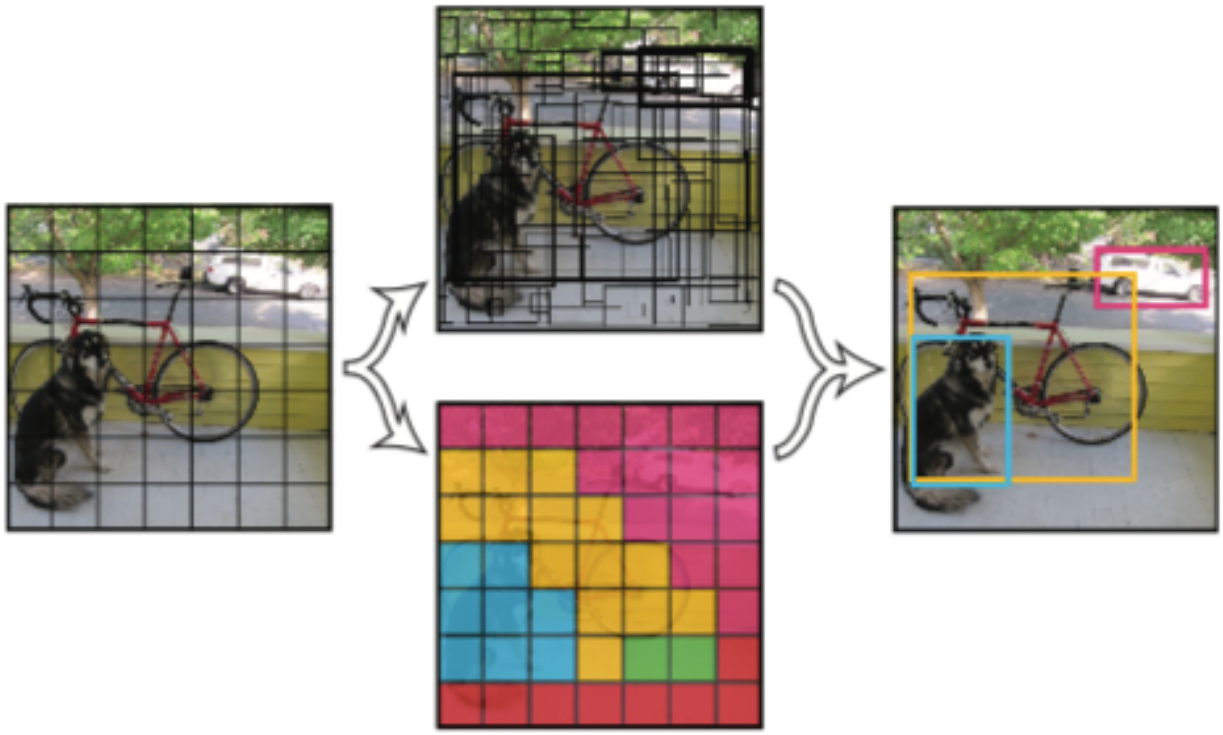
Model Design

Based on GoogLeNet for image classification + More

We're gonna see more detail later

How they did train it

- pre-train convolutional layers on the ImageNet 1000-class competition dataset, first 20 convolutional layers (trained for a week)
- add 4 convolutional layers and 2 fully connected layers with randomly initialized weights, resolution is increased in this level due to detection (it requires higher resolution)
- AND THEN FINAL LAYER(THIS IS THE CORE OF YOLO) : normalize the bounding box width and height by the image width and height so that they fall between 0 and 1
- (x,y) should also bounded between 0 and 1
- linear activation function will be used for final layer, all other layers use leaky rectified linear activation function



<OUTCOME OF FINAL LAYER>

This is about unified detection

1) system divides the input images into a S x S grid
2) each grid cell predicts B bounding boxes and confidence scores for those boxes, these confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts
3) they define confidence as Pr(Object) * IOU(truth over prediction). If no object exist in that cell, confidence score would be 0.
4) each bounding box predicts 5 values : x, y (coordinates), w, h (size of box), and confidence
5) each grid also predicts C (conditional class probabilities), Pr(Class i | Object). It only predicts one set of class probabilities per grid cell, regardless of the number of boxes B
6) It outputs : S x S x (B * 5 + C) tensor.

S : # of grid

B : # of bounding boxes to detect

- B boxes are assigned for each grid cell
- C (conditional class probabilities) determine which class is most likely to match with the grid

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

first term at left side is conditional class probability

first term at right side is class probability

For Instance : S=7, B=2, C=20

As a outcome of final layer, each grid produces two(# of boxes) 20 * 1(# of classes) vector.

The whole outcome would be (2 * 1 * 20) * (7 * 7)

(what each grid produces) * (# of all grids)

WITH BUNCH OF DATAS AS OUTCOME OF FINAL LAYER,

SO HOW DO WE CONCLUDE? WHERE IS OBJECT? AT WHICH CLASS IT BELONGS?

-> ON SLIDES

Details : epochs, variable learning rate, dropout, data augmentation

---

Error

- sum-squared error is used
- pros : easy to optimize
- cons : it doesn't perfectly align with our goal of maximizing average precision
- It weights localization error equally with classification error (It seems they want to weight those values different)

- <ISSUE 1>
- Many grid cells may not contain any object -> confidence value would be 0 -> gradient training problem (guess it means gradient vanishing)
- As a solution for it : INCREASE LOSS FROM BOUNDING BOX, DECREASE LOSS FROM BOX DON'T CONTAIN OBJECTS
- λ_{coord} and λ_{noobj} will be used for this. We set $\lambda_{coord} = 5$ and $\lambda_{noobj} = .5$.
- <ISSUE 2>
- same weight issue (previously mentioned) : Our error metric should reflect that small deviations in large boxes matter less than in small boxes
- Solution : we predict the square root of the bounding box width and height instead of the width and height directly

---

Loss function

- predictor : we want only one predicted bounding box to be responsible for each object as it predicts multiple bounding boxes. We want to focus on 'highest-likely-to-contain-object-box' and train weights to it to be more accurate.
- Among the multiple predicted bounding boxes, how do we determine PREDICTOR?
- one has highest IOU with the ground truth will be it.

THIS IS THE KEY POINT OF IT

<Notation>

$\mathbb{1}_{ij}^{\text{obj}}$: jth bounding box is predictor, it's on cell i, it contains object

$\mathbb{1}_{ij}^{\text{noobj}}$: jth bounding box, it's on cell i, which doesn't contain any object

$\mathbb{1}_{i}^{\text{obj}}$: if object appears in cell i

<Multi-part loss function>

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

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

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

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

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

- the loss function only penalizes classification error if an object is present in that grid cell

1 : (penalized) sum-squared error for coordinates

2 : (penalized) due to <ISSUE 2>, use square-root sum-squared error for box size

3 : loss of confidence score for box j, cell i

4 : (amplified) when it doesn't exist, amplify the error

5 : error of conditional class probability (multiplication of conditional class probability and object probability) (when it's correct = 1, otherwise = 0)

Limitation of YOLO

1) due to non-maximal suppression : it limits the number of nearby objects YOLO can predict
2) since it learns predicting bounding boxes from data, it struggles to generalize to object in new our unusual aspect ratios or configurations
3) small error in a small box has a much greater effect on IOU -> ??

Comparison to Other Detection Systems

Experiments they've done