CSE428: Image Processing

Lecture 16

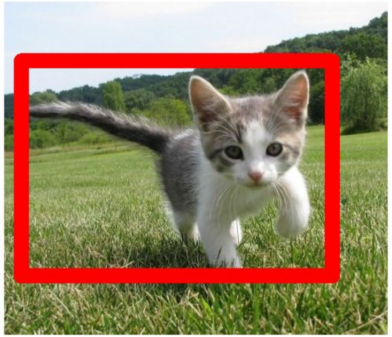
Object Detection

Image Recognition Problems

Classification

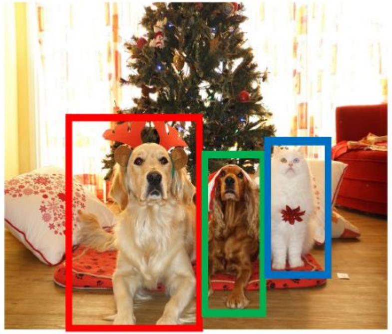
[{CAT}]

Classification +
Localization



[{CAT, (x, y, h, w)}]

Object Detection



[{DOG, (x, y, h, w)},
{DOG, (x, y, h, w)},
{CAT, (x, y, h, w)}]

Deep Learning Based Object Detection Methods

Neural network approaches:

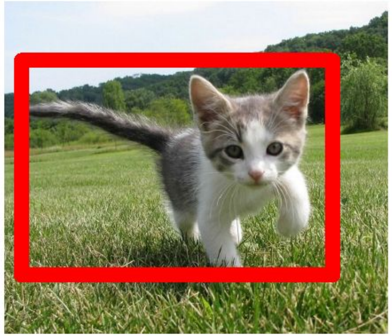
- Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
- Single Shot MultiBox Detector (SSD)
- You Only Look Once (YOLO)

neural techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

Classification + Localization

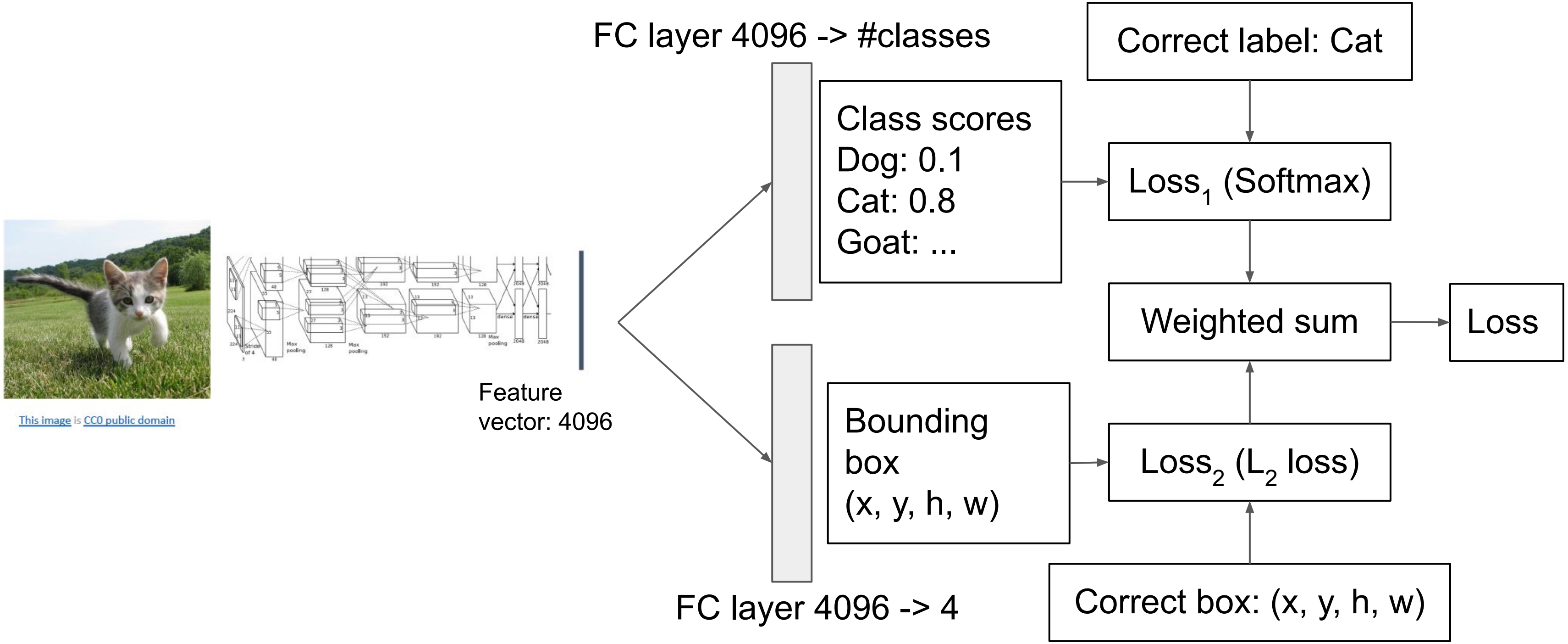
Only applicable when there is **one object/image**!

Classification +
Localization



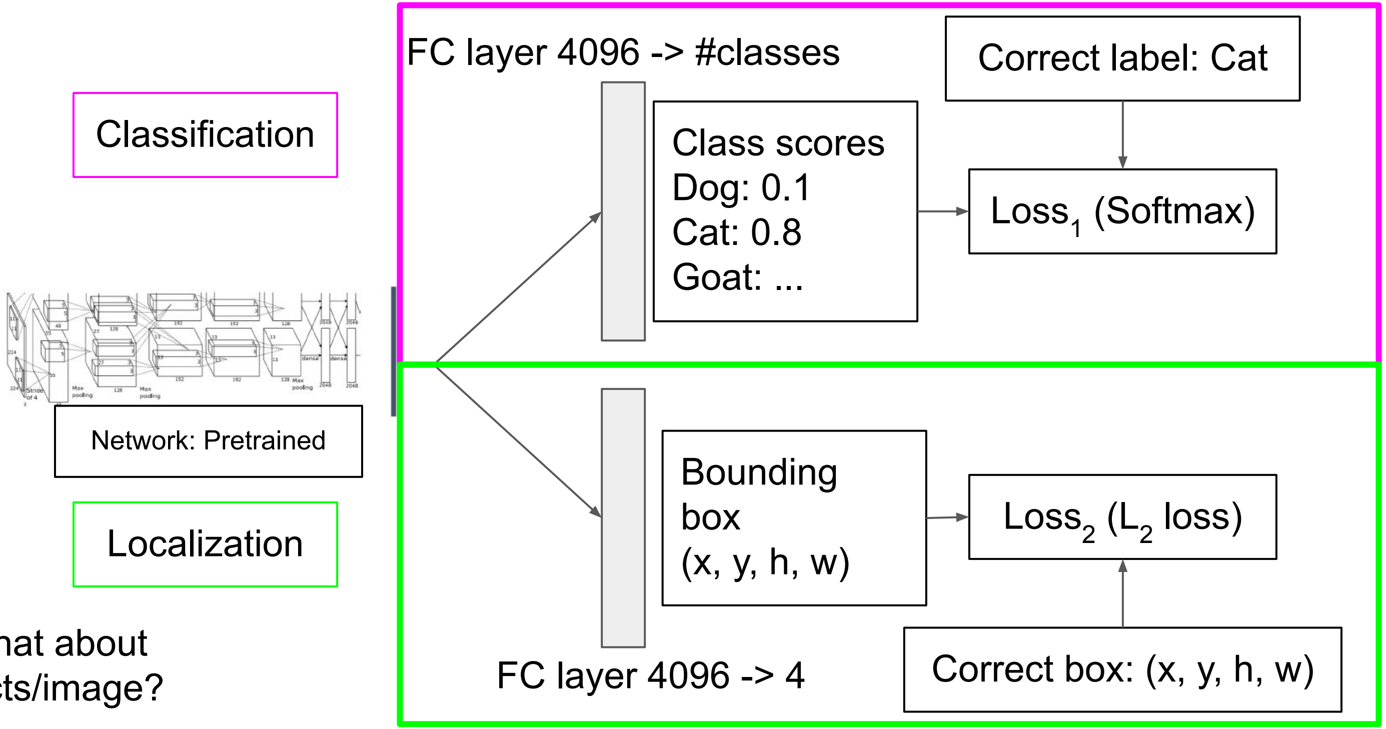
[{CAT, (x, y, h, w)}]

Classification + Localization



Classification + Localization

This image is CC0 public domain



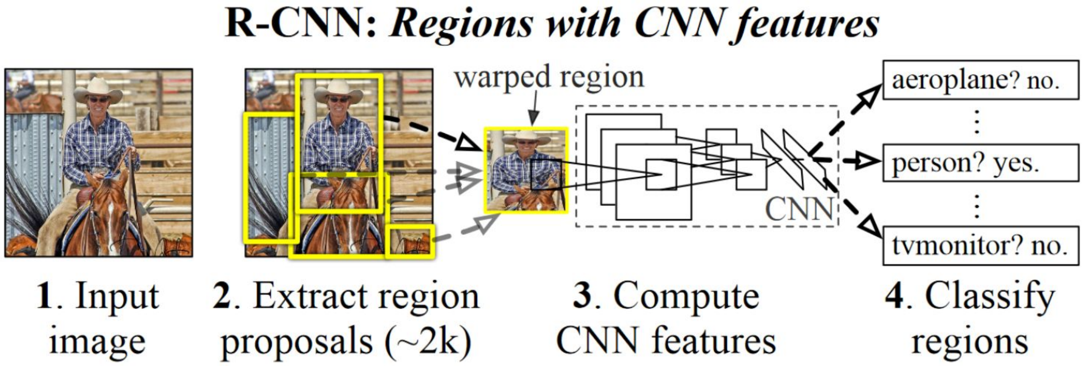
Question: What about multiple objects/image?

For multiple objects/image this algorithm would be highly inefficient!

Region-based CNN (R-CNN)

Region-based CNN

1. takes an input image
2. extracts around 2000 bottom-up region proposals
3. computes features for each proposal using a large CNN
4. classifies each region using class-specific linear SVMs

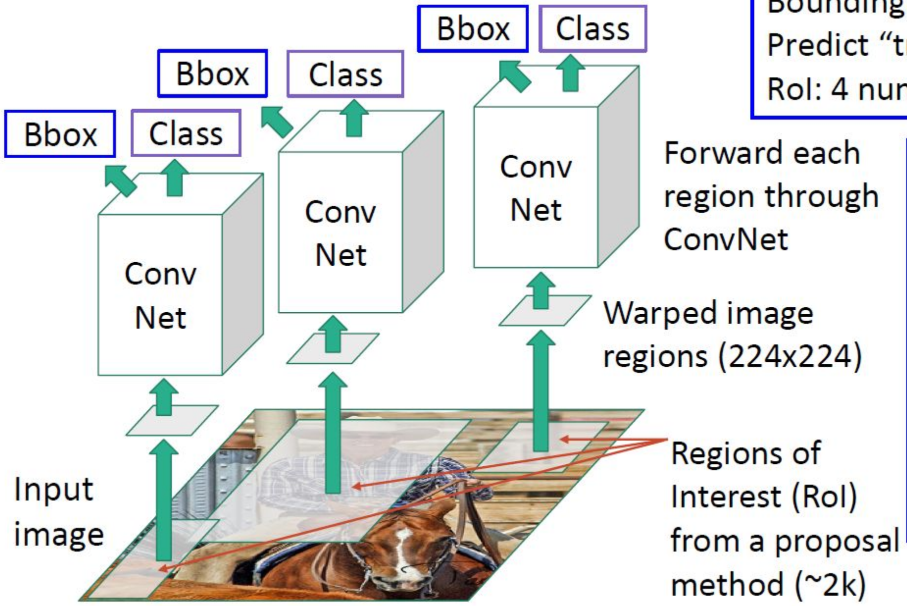


- R-CNN achieves a mean average precision (mAP) of 53.7% on PASCAL VOC 2010.
- On the 200-class ILSVRC 2013 detection dataset, R-CNN's mAP is 31.4%

https://arxiv.org/abs/1311.2524

R-CNN

Bounding box regression



Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Region proposal: (p_x, p_y, p_h, p_w)
Transform: (t_x, t_y, t_h, t_w)
Output box: (b_x, b_y, b_h, b_w)

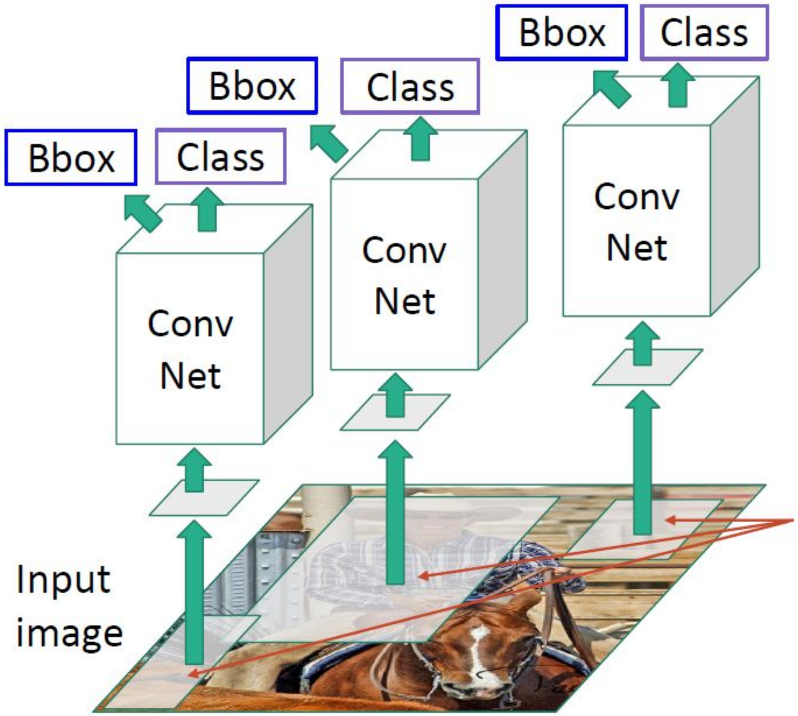
Translate relative to box size:
$b_x = p_x + p_w t_x$ $\quad$ $b_y = p_y + p_h t_y$

Log-space scale transform:
$b_w = p_w \exp(t_w)$ $\quad$ $b_h = p_h \exp(t_h)$

R-CNN

Inference time: for an RGB input image

1. Run region proposal method to compute ~2000 region proposals
2. Resize each region to 224x224 and run independently through CNN to predict **class scores** and **bbox transform**
3. Use scores to **filter** a subset of region proposals to output
4. <u>*Compare* with ground-truth boxes</u>

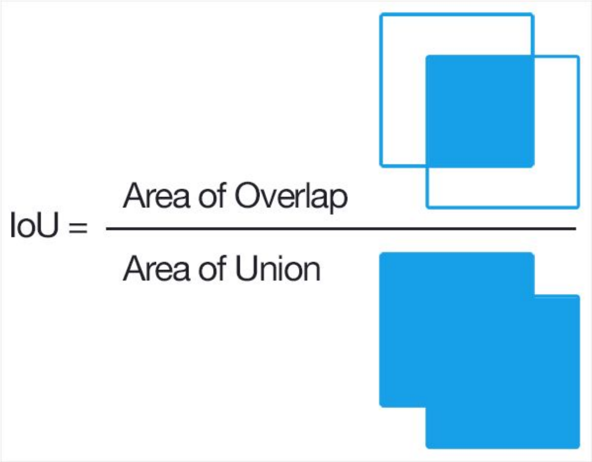


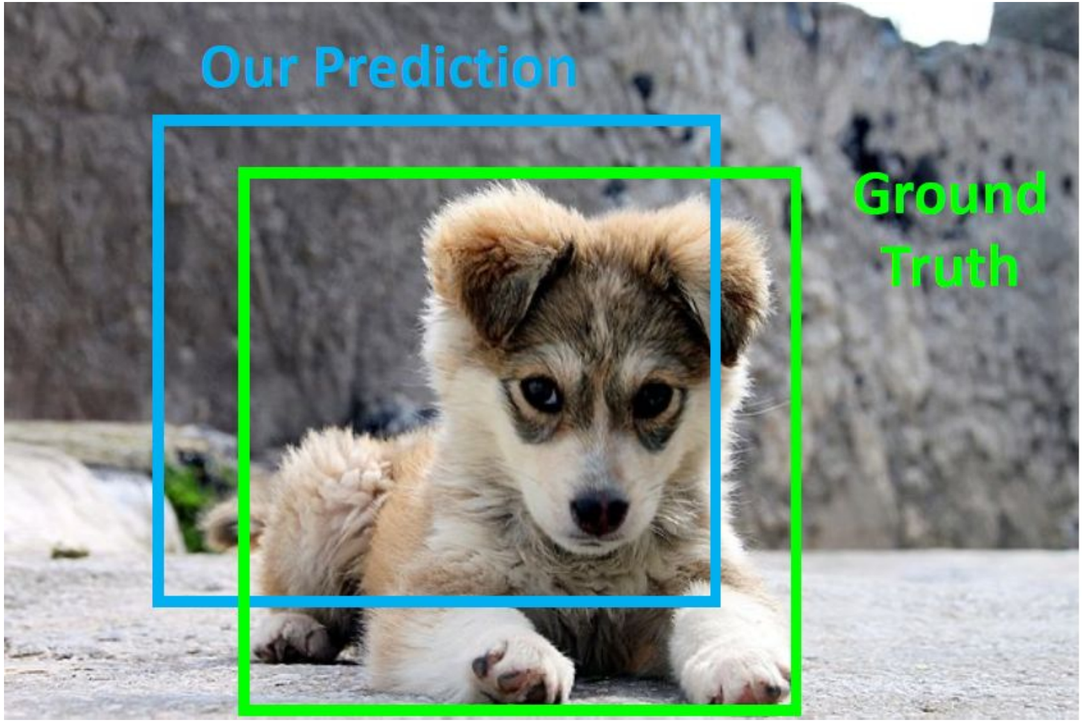
How to evaluate predictions for object detection?

Concept: IoU

How to compare the quality of bounding box prediction?

Intersection over Union (IoU)

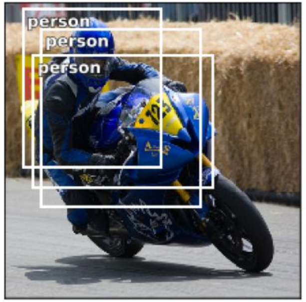




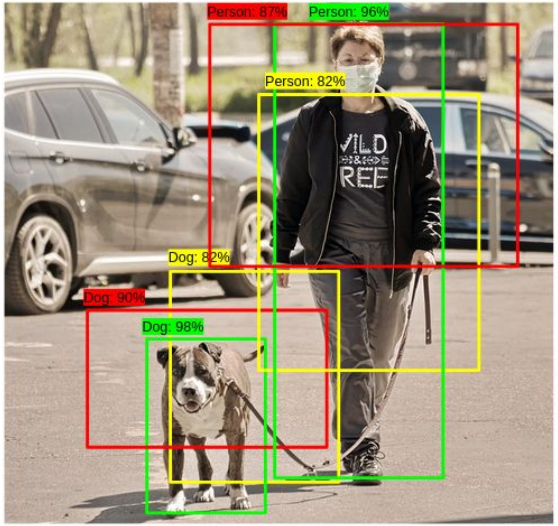
Concept: Non Max Suppression (NMS)

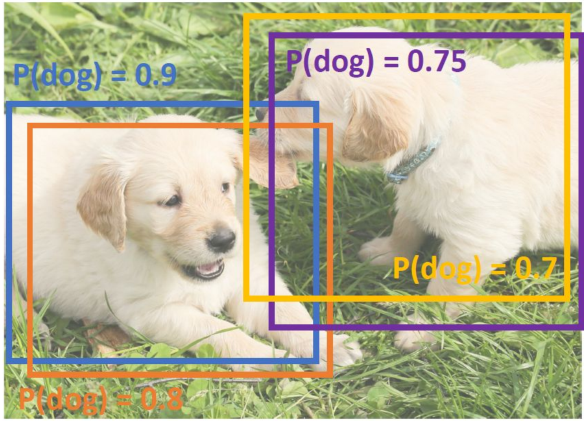
Model usually outputs multiple redundant boxes per object

For each class...



After filtering out low confidence predictions, we may still be left with ***redundant*** **detections**



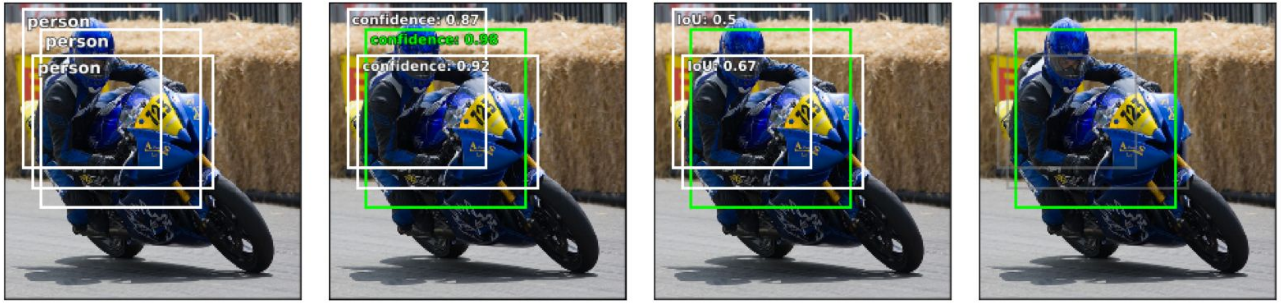


Concept: Non Max Suppression (NMS)

Apply NMS

Repeat with next highest confidence prediction until no more boxes are being suppressed

For each class...



After filtering out low confidence predictions, we may still be left with ***redundant* detections**

Select the bounding box prediction with the **highest confidence**

Calculate the IoU between the **selected box** and all remaining predictions

Remove any boxes which have an IoU score above some defined threshold

Concept: Non Max Suppression (NMS)

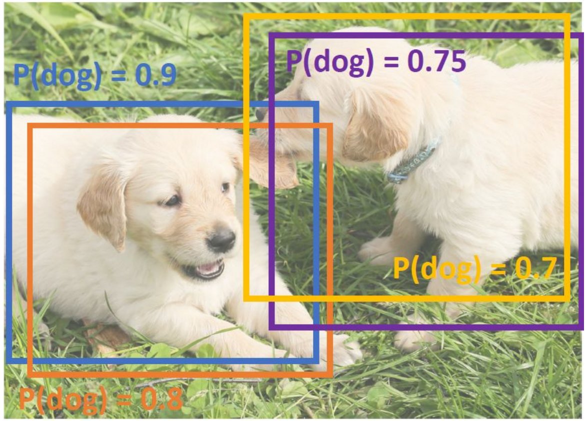
NMS Algorithm

Pseudocode

```
sorted_boxes = sort_boxes_by_confidence(boxes)
ids_to_suppress = []

for maximum_box in sorted_boxes:
    for idx, box in enumerate(boxes):
        iou = compute_iou(maximum_box, box)
        if iou > iou_threshold:
            ids_to_suppress.append(idx)

processed_boxes = np.delete(boxes, ids_to_suppress)
```



Concept: TP, FP, FN, P, R

True positive (TP): A correct detection. Detection with IoU ≥ threshold

False positive (FP): An incorrect detection of a nonexistent object or a misplaced detection of an existing object. Detection with IoU < threshold

False negative (FN): An undetected ground-truth bounding box

Precision (P): TP / (TP + FP) = TP / All detections

$$P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} = \frac{\mathrm{TP}}{\text{all detections}},$$

Recall (R): TP / (TP + TN) = TP / All ground truths

$$R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = \frac{\mathrm{TP}}{\text{all ground truths}}.$$

Concept: AP

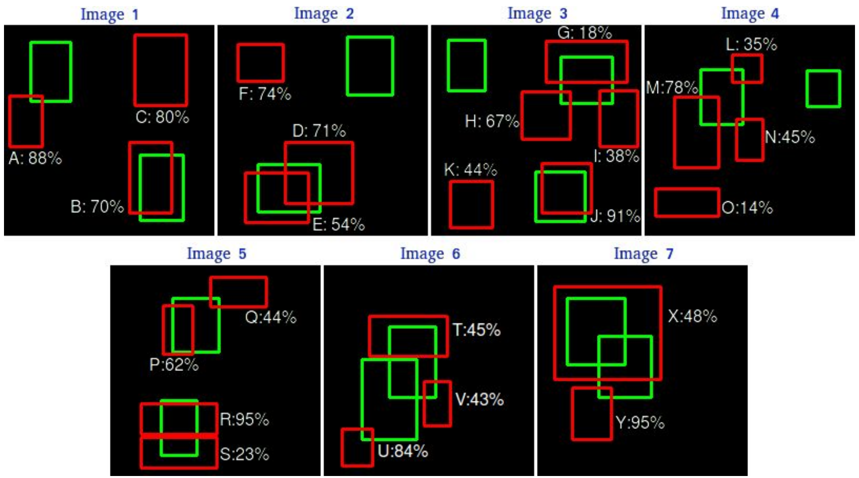
Average Precision (AP):

For *each* detection of a **single category** (highest score to lowest score)

1. If it matches some GT box with IoU > **0.5**, mark it as true positive (TP) and eliminate the GT
2. Otherwise mark it as negative (FP)
3. Plot a point on PR Curve
4. Average Precision (AP) = area under PR curve

Concept: AP

There are 7 images with 15 ground truth objects represented by the green bounding boxes and 24 detected objects represented by the red bounding boxes. Each detected object has a confidence level and is identified by a letter (A,B,...,Y).



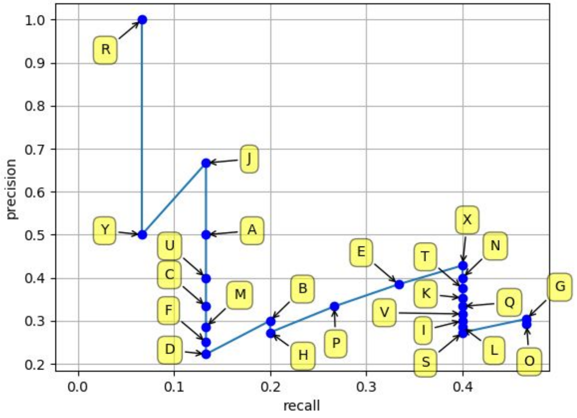
Images	Detections	Confidences	TP or FP
Image 1	A	88%	FP
Image 1	B	70%	TP
Image 1	C	80%	FP
Image 2	D	71%	FP
Image 2	E	54%	TP
Image 2	F	74%	FP
Image 3	G	18%	TP
Image 3	H	67%	FP
Image 3	I	38%	FP
Image 3	J	91%	TP
Image 3	K	44%	FP
Image 4	L	35%	FP
Image 4	M	78%	FP
Image 4	N	45%	FP
Image 4	O	14%	FP
Image 5	P	62%	TP
Image 5	Q	44%	FP
Image 5	R	95%	TP
Image 5	S	23%	FP
Image 6	T	45%	FP
Image 6	U	84%	FP
Image 6	V	43%	FP
Image 7	X	48%	TP
Image 7	Y	95%	FP

The following table shows the bounding boxes with their corresponding confidences. The last column identifies the detections as TP or FP. In this example a TP is considered if IoU 30%, otherwise it is a FP.

https://github.com/rafaelpadilla/Object-Detection-Metrics

Concept: AP

- First we need to order the detections by their confidences, then we calculate the precision and recall for each accumulated detection.
- Plotting the precision and recall values we have the following Precision vs. Recall curve



Images	Detections	Confidences	TP	FP	Acc TP	Acc FP	Precision	Recall
Image 5	R	95%	1	0	1	0	1	0.0666
Image 7	Y	95%	0	1	1	1	0.5	0.0666
Image 3	J	91%	1	0	2	1	0.6666	0.1333
Image 1	A	88%	0	1	2	2	0.5	0.1333
Image 6	U	84%	0	1	2	3	0.4	0.1333
Image 1	C	80%	0	1	2	4	0.3333	0.1333
Image 4	M	78%	0	1	2	5	0.2857	0.1333
Image 2	F	74%	0	1	2	6	0.25	0.1333
Image 2	D	71%	0	1	2	7	0.2222	0.1333
Image 1	B	70%	1	0	3	7	0.3	0.2
Image 3	H	67%	0	1	3	8	0.2727	0.2
Image 5	P	62%	1	0	4	8	0.3333	0.2666
Image 2	E	54%	1	0	5	8	0.3846	0.3333
Image 7	X	48%	1	0	6	8	0.4285	0.4
Image 4	N	45%	0	1	6	9	0.4	0.4
Image 6	T	45%	0	1	6	10	0.375	0.4
Image 3	K	44%	0	1	6	11	0.3529	0.4
Image 5	Q	44%	0	1	6	12	0.3333	0.4
Image 6	V	43%	0	1	6	13	0.3157	0.4
Image 3	I	38%	0	1	6	14	0.3	0.4
Image 4	L	35%	0	1	6	15	0.2857	0.4
Image 5	S	23%	0	1	6	16	0.2727	0.4
Image 3	G	18%	1	0	7	16	0.3043	0.4666
Image 4	O	14%	0	1	7	17	0.2916	0.4666

https://github.com/rafaelpadilla/Object-Detection-Metrics

Resources

1. https://cs231n.github.io/convolutional-networks/
2. https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/
3. https://www.tensorflow.org/api_docs/python/tf/keras
4. Deep Learning with Python Book by François Chollet
5. https://www.deeplearningbook.org/
6. **Hands-on Computer Vision with TensorFlow 2 by Eliot Andres & Benjamin Planche (Packt Pub.)**