Convolutional networks for visual object detection

COMPUTER VISION TASKS

Image Classification

Object Detection

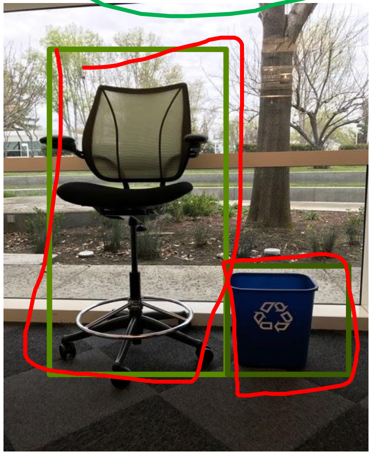
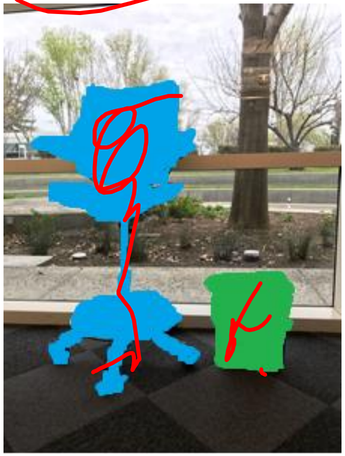
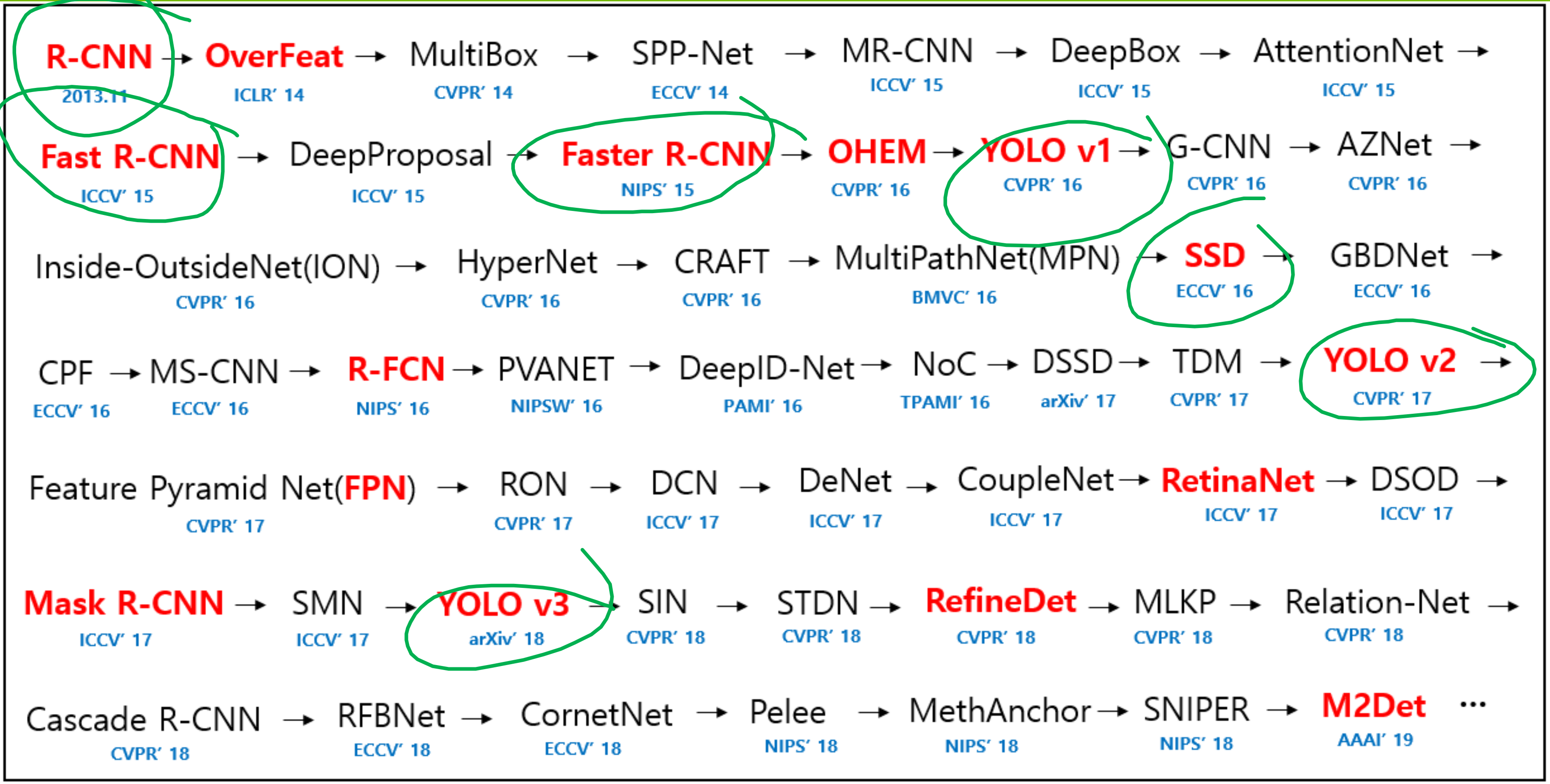


Image Segmentation



(examples are from cs231n lecture from Stanford University)



R-CNN → **OverFeat** → MultiBox → SPP-Net → MR-CNN → DeepBox → AttentionNet →
2013.11 ICLR’ 14 CVPR’ 14 ECCV’ 14 ICCV’ 15 ICCV’ 15 ICCV’ 15

Fast R-CNN → DeepProposal → **Faster R-CNN** → **OHEM** → **YOLO v1** → G-CNN → AZNet →
ICCV’ 15 ICCV’ 15 NIPS’ 15 CVPR’ 16 CVPR’ 16 CVPR’ 16 CVPR’ 16

Inside-OutsideNet(ION) → HyperNet → CRAFT → MultiPathNet(MPN) → **SSD** → GBDNet →
CVPR’ 16 CVPR’ 16 CVPR’ 16 BMVC’ 16 ECCV’ 16 ECCV’ 16

CPF → MS-CNN → **R-FCN** → PVANET → DeepID-Net → NoC → DSSD → TDM → **YOLO v2** →
ECCV’ 16 ECCV’ 16 NIPS’ 16 NIPSW’ 16 PAMI’ 16 TPAMI’ 16 arXiv’ 17 CVPR’ 17 CVPR’ 17

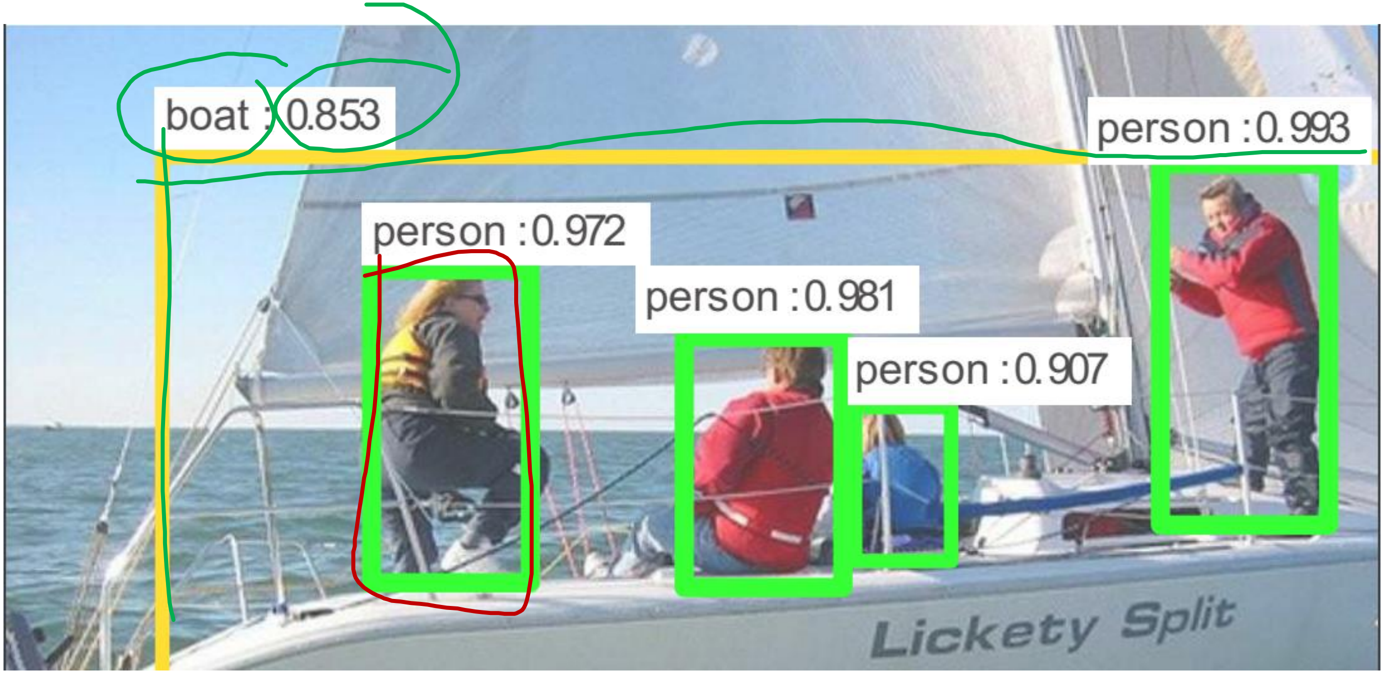
Feature Pyramid Net(**FPN**) → RON → DCN → DeNet → CoupleNet → **RetinaNet** → DSOD →
CVPR’ 17 CVPR’ 17 ICCV’ 17 ICCV’ 17 ICCV’ 17 ICCV’ 17 ICCV’ 17

Mask R-CNN → SMN → **YOLO v3** → SIN → STDN → **RefineDet** → MLKP → Relation-Net →
ICCV’ 17 ICCV’ 17 arXiv’ 18 CVPR’ 18 CVPR’ 18 CVPR’ 18 CVPR’ 18 CVPR’ 18

Cascade R-CNN → RFBNet → CornetNet → Pelee → MethAnchor → SNIPER → **M2Det** …
CVPR’ 18 ECCV’ 18 ECCV’ 18 NIPS’ 18 NIPS’ 18 NIPS’ 18 AAAI’ 19

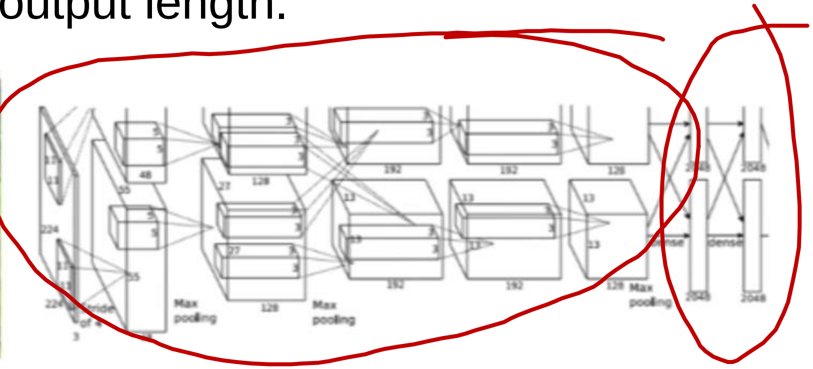
OBJECT DETECTION

- Object detection can identify and classify one or more objects in an image
- Detection is also about localizing the extent of an object in an image
  - Bounding boxes / heat maps
- Training data must have objects within images labeled
  - Can be hard to find / produce training dataset



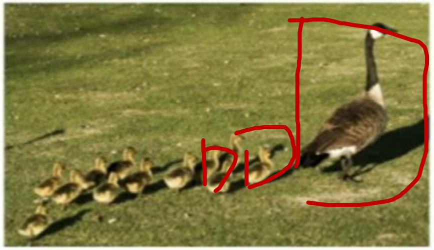
OBJECT DETECTION

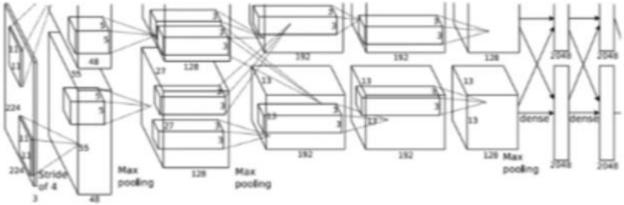
- A standard convolutional network followed by a fully connected layer has a fixed output length.



CAT: (x, y, w, h)

- On the other hand, there might be different number of objects in an image.



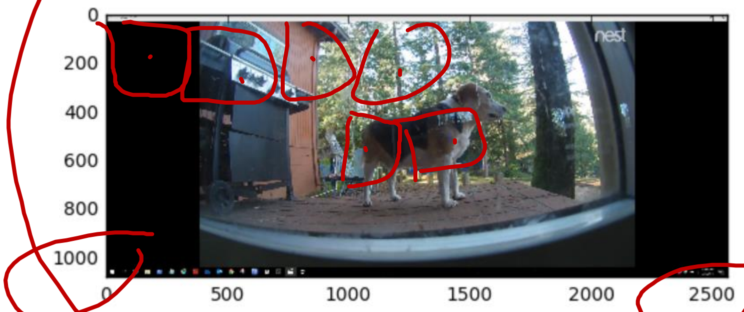


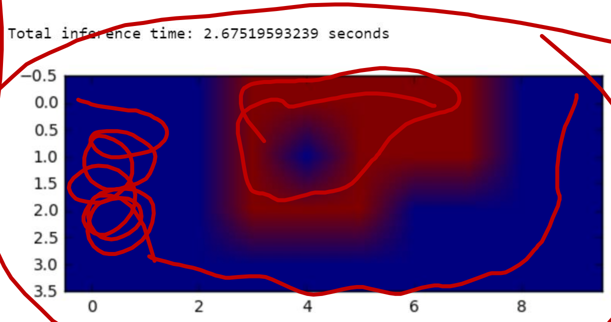
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

OBJECT DETECTION - Sliding window

- A naive approach to solve this problem would be to take different regions of interest from the image, and use a CNN to classify the presence of the object within that region.

- For example, we have a 2500x1000 image and we want a dog detector:
  - Build a “dog” / “not dog” classifier
  - Use a sliding window approach to run this classifier on each 256X256 segment





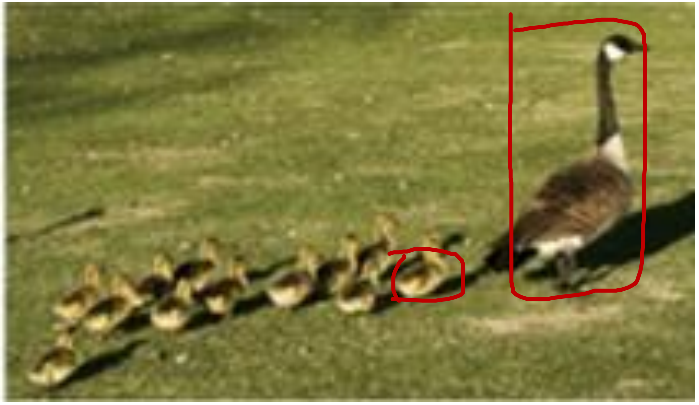
```
Total inference time: 2.67519593239 seconds
```

OBJECT DETECTION

- The objects of interest might have different spatial locations within the image and different aspect ratios.

- Hence, you would need to select a huge number of regions -> significant computational cost.

- Therefore, algorithms like R-CNN, YOLO have been developed.



FULLY CONVOLUTIONAL NETWORKS (FCN)

- Fully Convolutional Networks are a subset of CNNs

- The difference to regular CNNs is that they do not have the fully connected layer at the end

- They are mainly used for tasks such as object detection and segmentation

OBJECT DETECTION

- Two-stage Region based object detectors: the first stage takes in an input image and outputs region proposals i.e locations where an object might be present and the second stage involves classifying to which class each proposal belongs to.

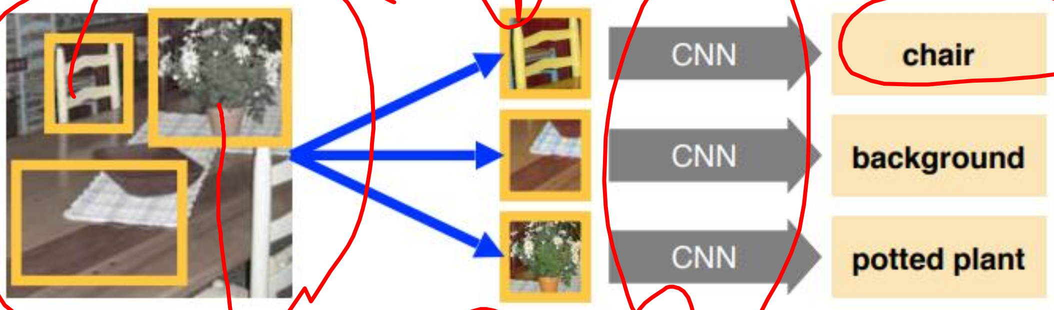
  - R-CNN (Region Based CNN), Fast R-CNN, Faster R-CNN,
  - R-FCN (Region Based Fully Convolutional NN),
  - FPN (Feature Pyramid Network)

- Single shot object detectors
  - SSD (Single Shot Detector)
  - YOLO (You Only Look Once)

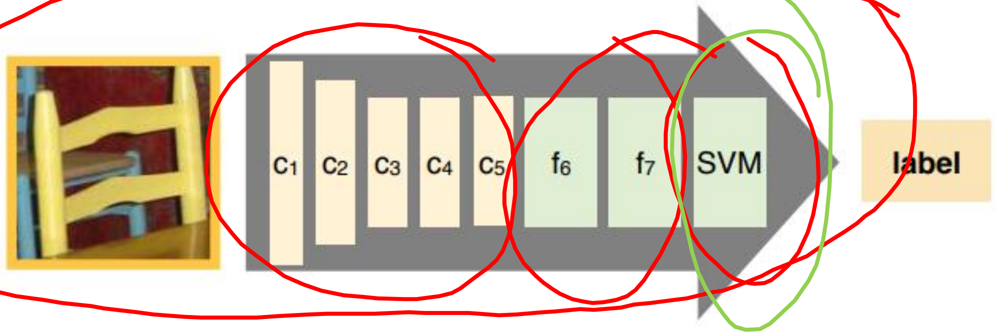
R-CNN

Region-based Convolutional Neural Network

Pros: simple and effective



Cons: slow as the CNN is re-evaluated for each tested region



Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation
R. Girshick, J. Donahue, T. Darrell, J. Malik, CVPR 2014

R-CNN

- To eliminate the problem of selecting large number of regions, selective search is used to extract just 2000 regions from the image (region proposals).

- Candidate region proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output.

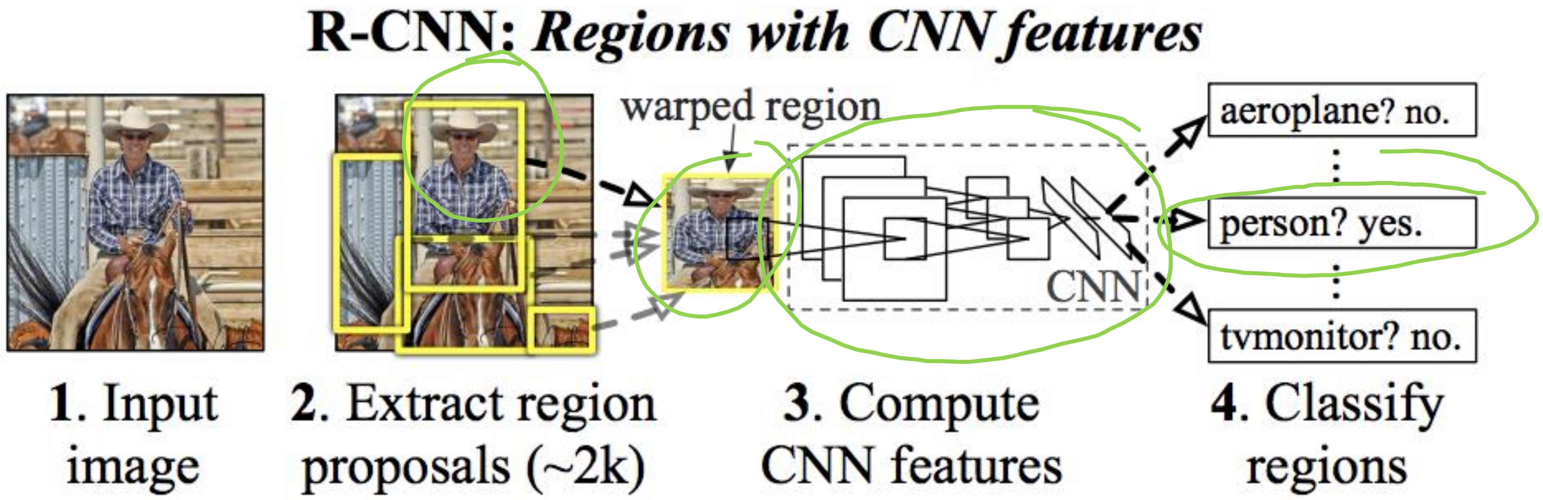
- The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM to classify the presence of the object within that candidate region proposal.

Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

R-CNN

- In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values to increase the precision of the bounding box.

- For example, given a region proposal, the algorithm would have predicted the presence of a person but the face of that person within that region proposal could've been cut in half. Therefore, the offset values help in adjusting the bounding box of the region proposal.



From proposals to CNN features

Dilate, crop, reshape

Propose

Dilate

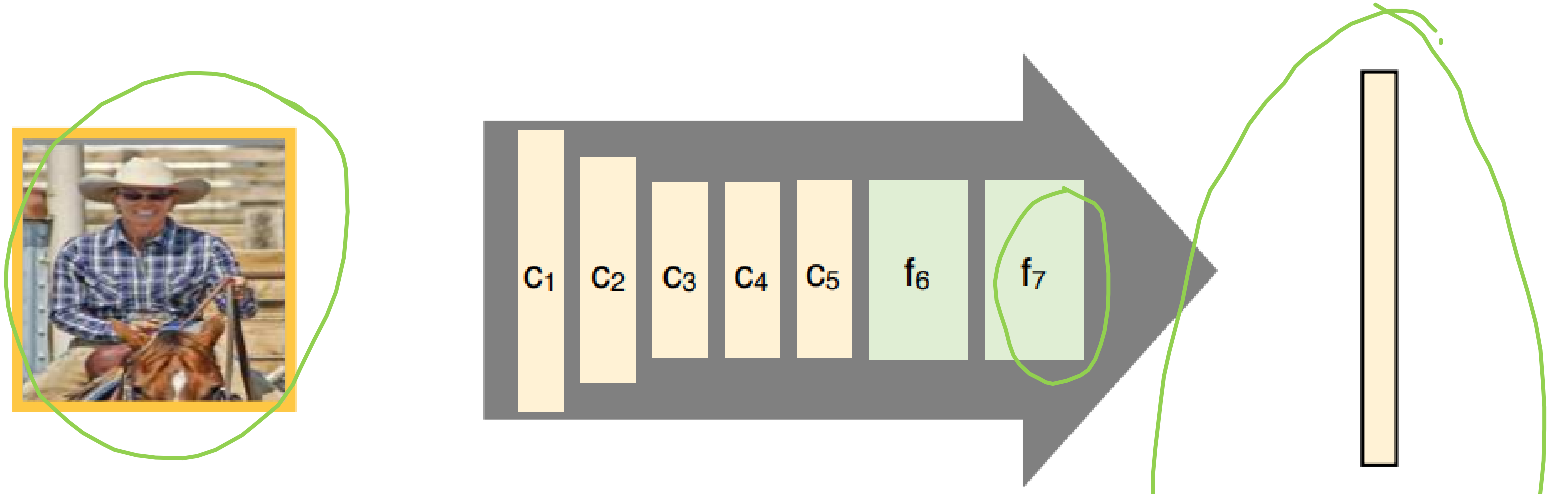
Crop & scale

Anisotropic

227 x 227

From proposal to CNN features

Evaluate CNN



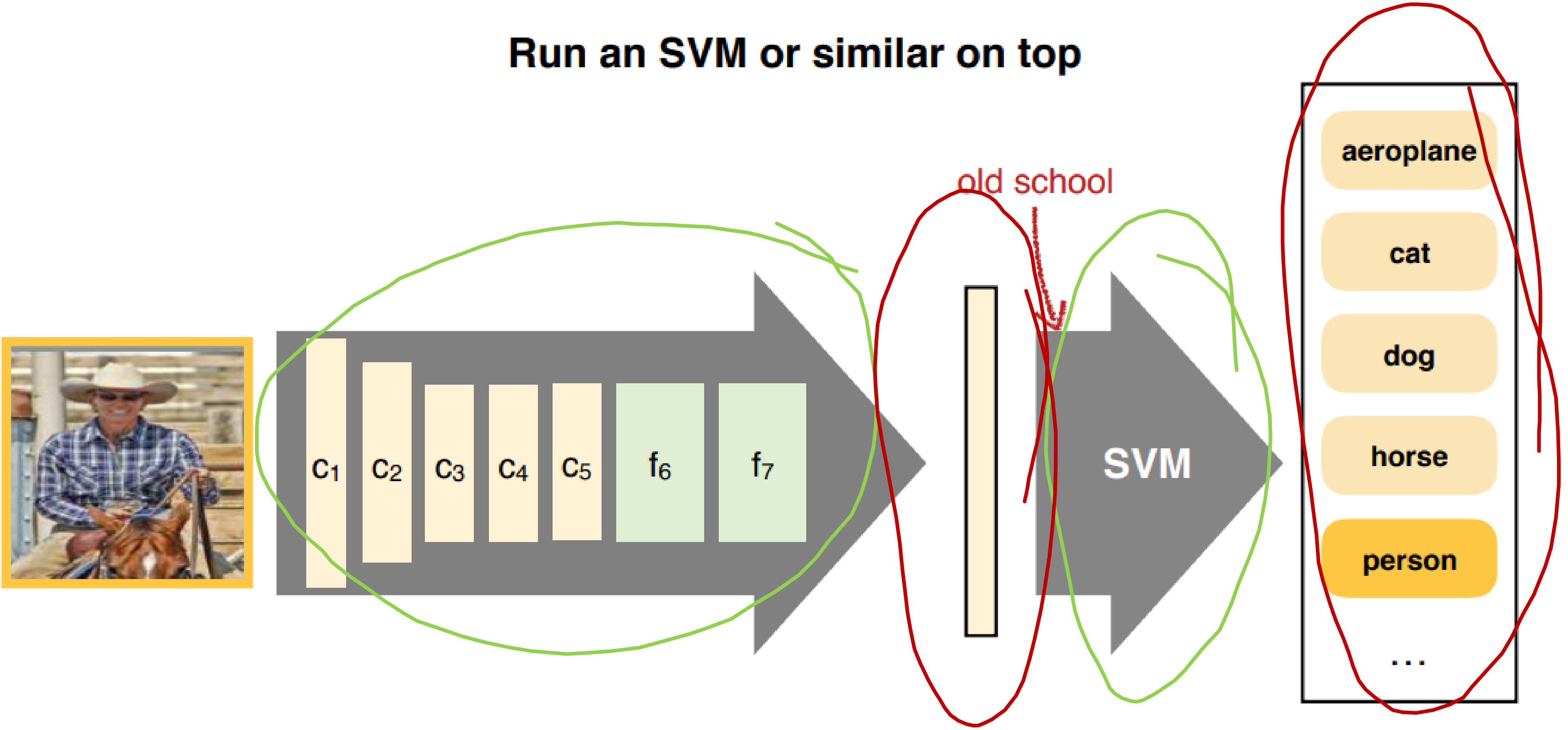
Scale
Anisotropic
227 x 227

CNN features
Up to FC-7
AlexNet

Feature vector
4096 D

Classification of a region

Run an SVM or similar on top



Scale
Anisotropic
227 x 227

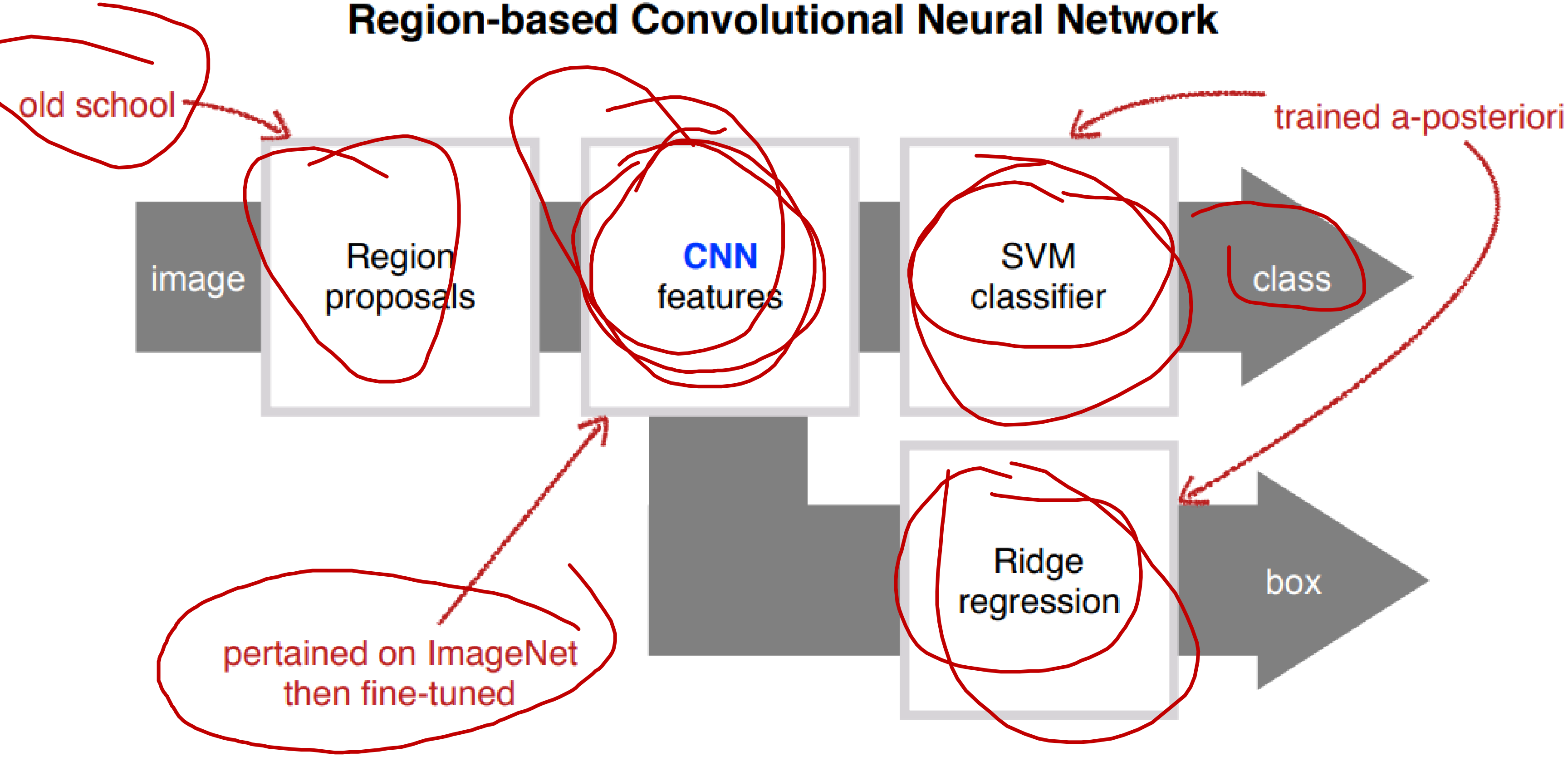
CNN features
Up to FC-7
AlexNet

Feature vector
4096 D

Label
One out of *N*

R-CNN summary

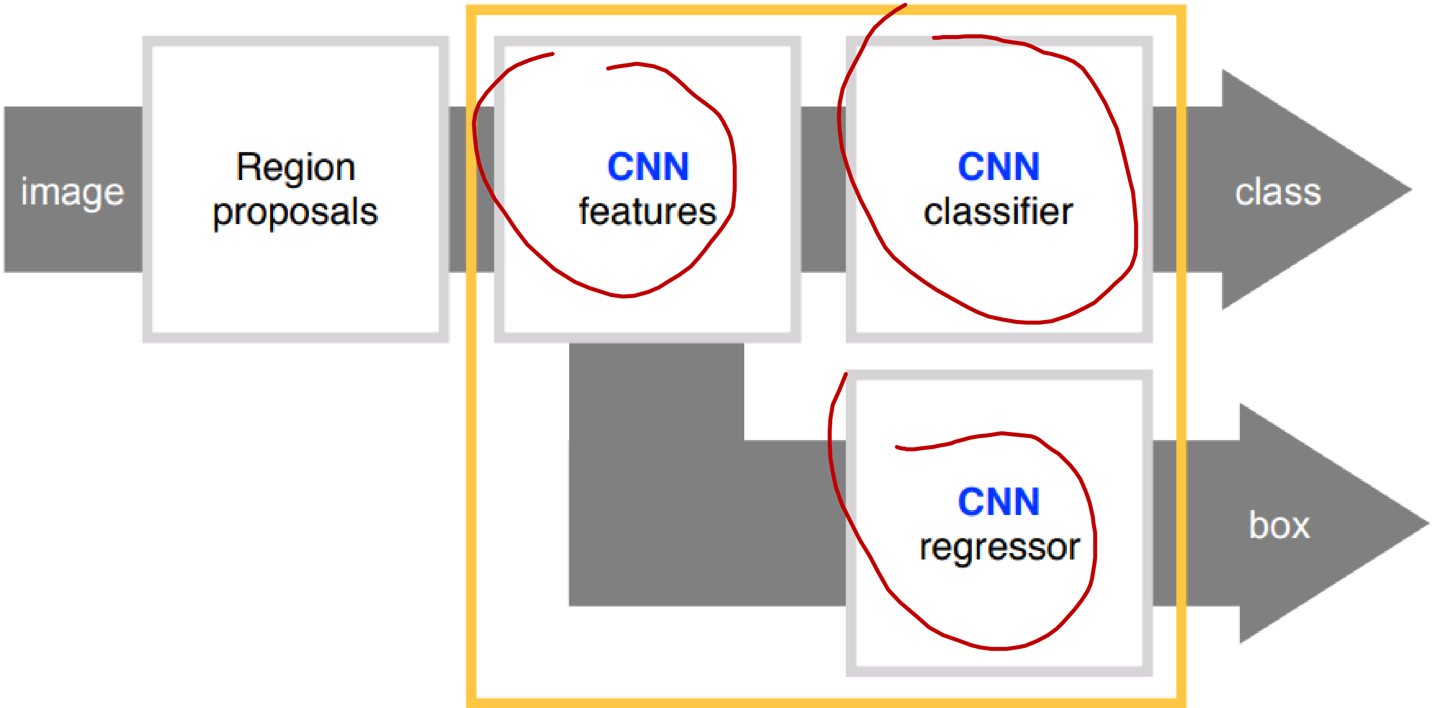
Region-based Convolutional Neural Network



Can we achieve end-to-end training?

Towards better R-CNNs

Region-based Convolutional Neural Network



End-to-end training

Except for region proposals

Problem: this is still pretty slow!

Fast R-CNN

- Instead of feeding the region proposals to the CNN, feed the input image to the CNN to generate a convolutional feature map.

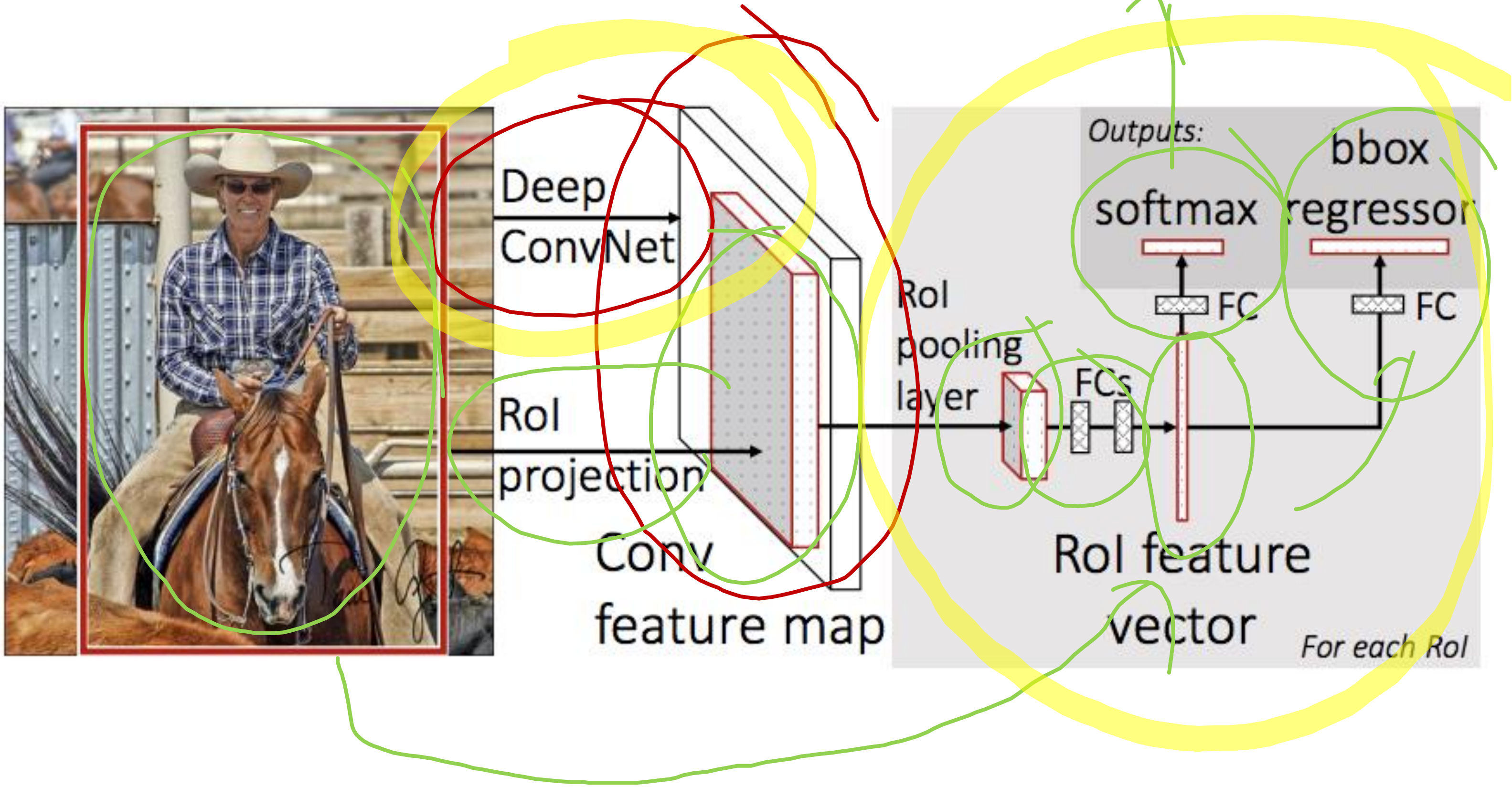
- From the convolutional feature map, identify the region of proposals and warp them into squares and by using a RoI pooling layer reshape them into a fixed size so that it can be fed into a fully connected layer.

- From the RoI feature vector, use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.

- The reason “Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

Fast R-CNN

Person



Girshick, R., 2015. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

Faster R-CNN

- Both of the above algorithms(R-CNN & Fast R-CNN) uses selective search to find out the region proposals.

- Selective search is a slow and time-consuming process affecting the performance of the network. Faster R-CNN eliminates the selective search algorithm and lets the network learn the region proposals.

Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

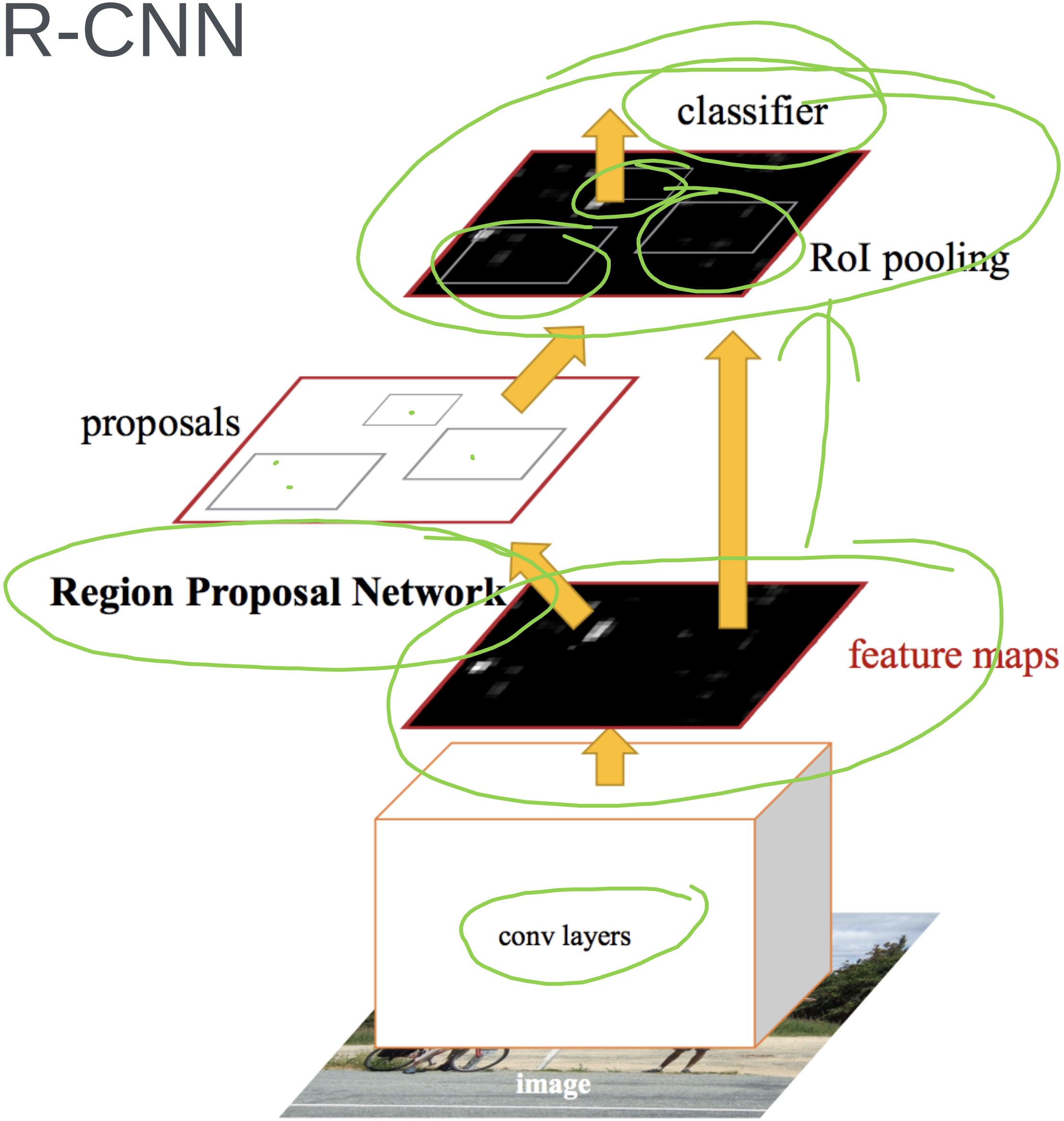
Faster R-CNN

- Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map.

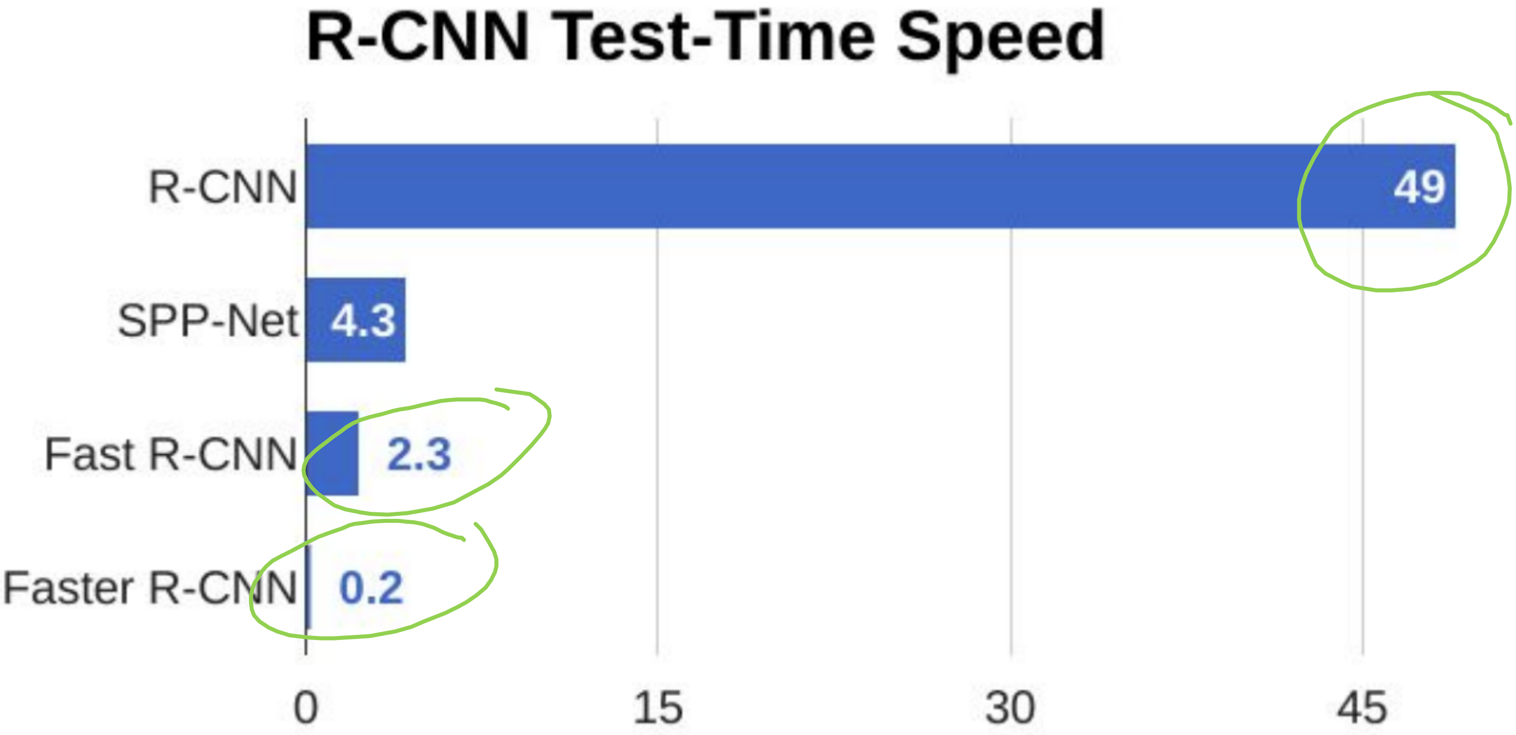
- Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals.

- The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

Faster R-CNN


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

Faster R-CNN


R-CNN Test-Time Speed
R-CNN
SPP-Net
Fast R-CNN
Faster R-CNN
49
4.3
2.3
0.2
0
15
30
45

OBJECT DETECTION

- Instead of having another neural network/algorithm to generate region proposals, an entire grid of feature map is considered as region proposals which in turn is classified by the same neural network to produce class scores and bounding box offsets.

- Single shot object detectors use the concept of anchor boxes. Anchors are logical boxes assigned to every cell in a feature map, relative to which the bounding box regression and classification takes place. They can be of different aspect ratios

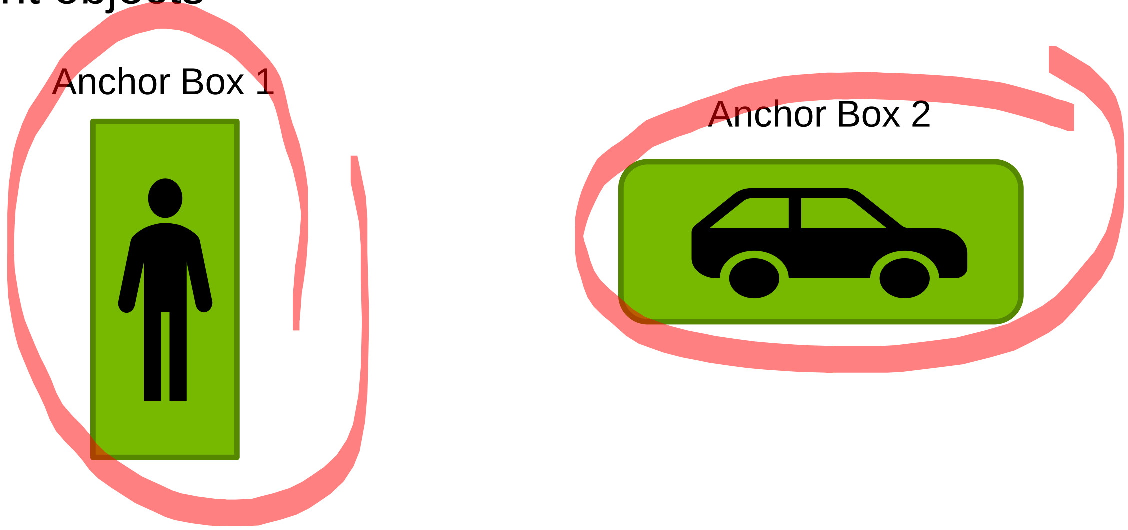
  - SSD (Single Shot Detector)
  - YOLO (You Only Look Once)

Single Shot Detectors - Anchor Boxes

- These networks make high number of (thousands) predictions.

- Then they only output the ones that it is highly likely to contain an object.

- Anchor boxes could be of different shapes to allow detection of different objects



YOLO—You Only Look Once

- All of the previous object detection algorithms use regions to localize the object within the image.

- The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object.

- In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

YOLO—You Only Look Once

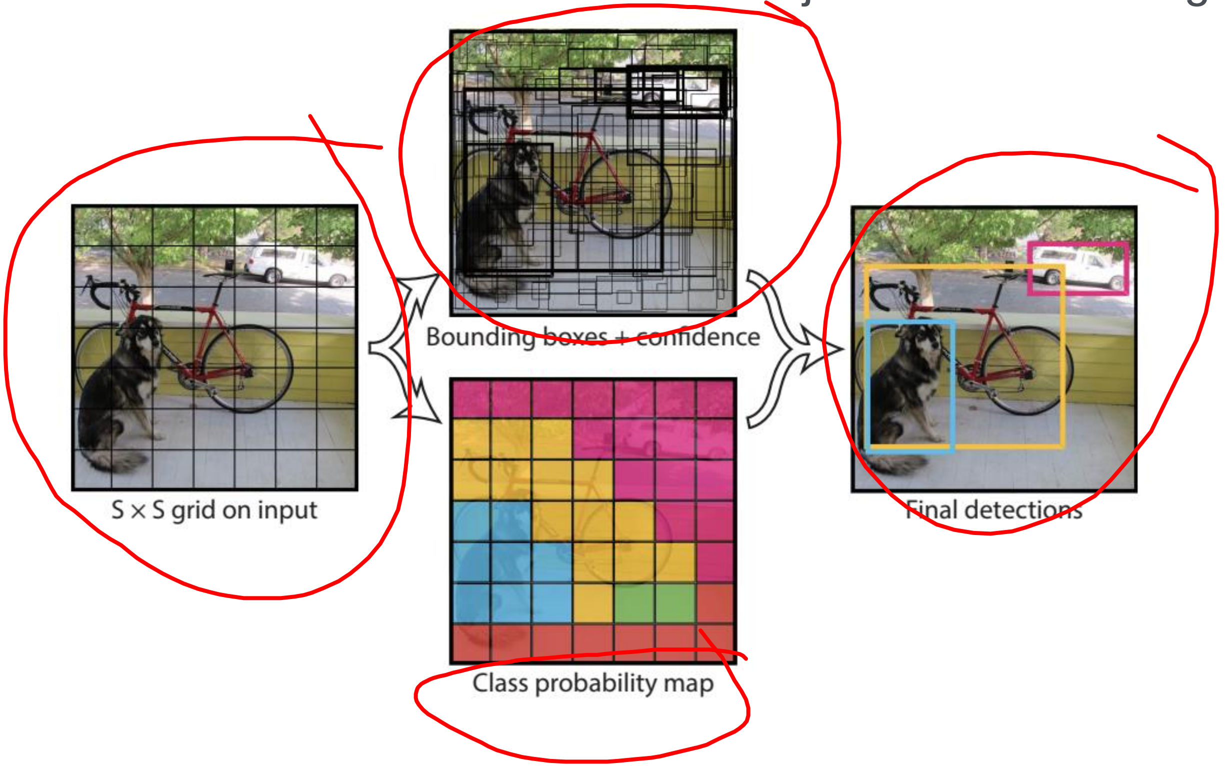
- YOLO is orders of magnitude faster than other object detection algorithms.

- On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev.

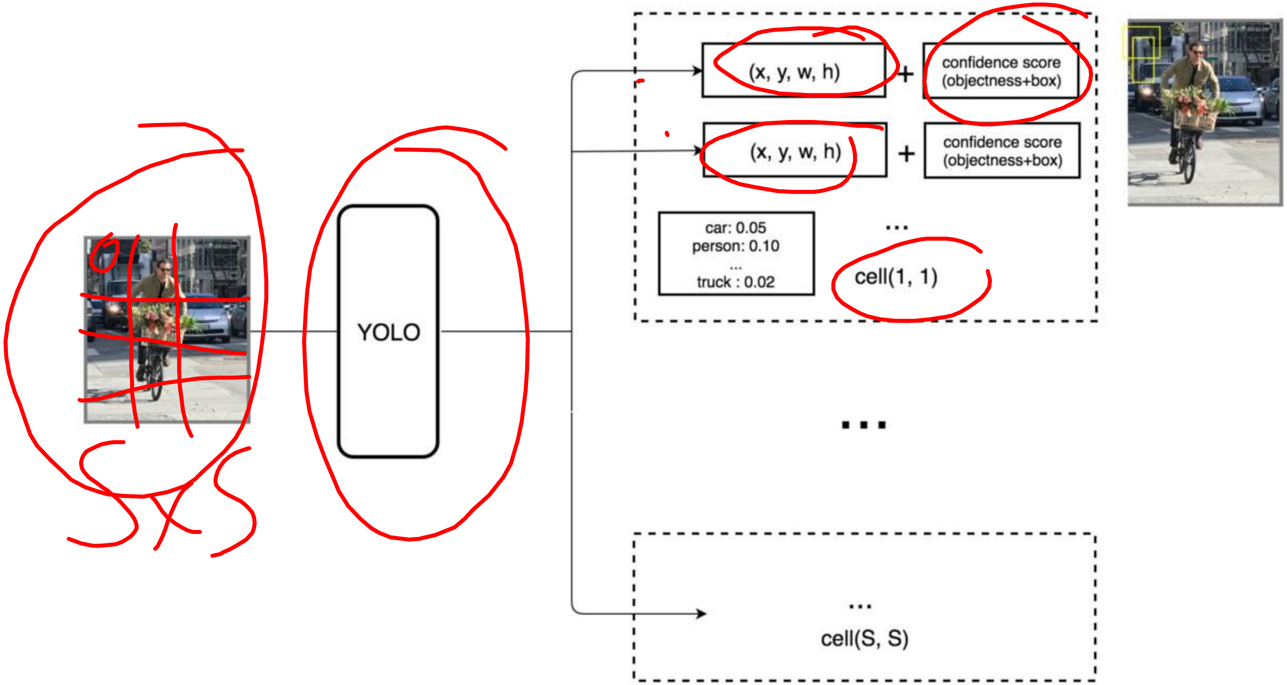
- The limitation of YOLO algorithm is that it struggles with small objects within the image due to the spatial constraints of the algorithm.

YOLO—You Only Look Once

- YOLO first splits an image into *SxS* grid.
- For each grid, the network outputs *B* bounding boxes.
- Each bounding box has a class probability and offset values for the bounding box.
- The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.



YOLO—You Only Look Once



- Each bounding box contains 5 elements: (*x*, *y*, *w*, *h*) and a **box confidence score.**
- The box confidence score reflects how likely the box contains an object (**objectness**) and how accurate is the boundary box.

YOLO—You Only Look Once

- Bounding box width w and height h is normalized with respect to the image width and height.
- x and y are offsets to the corresponding cell. Hence, x, y, w and h are all between 0 and 1.

- To evaluate PASCAL VOC, YOLO uses 7×7 grids (S×S), 2 Bounding boxes (B) and 20 classes (C).

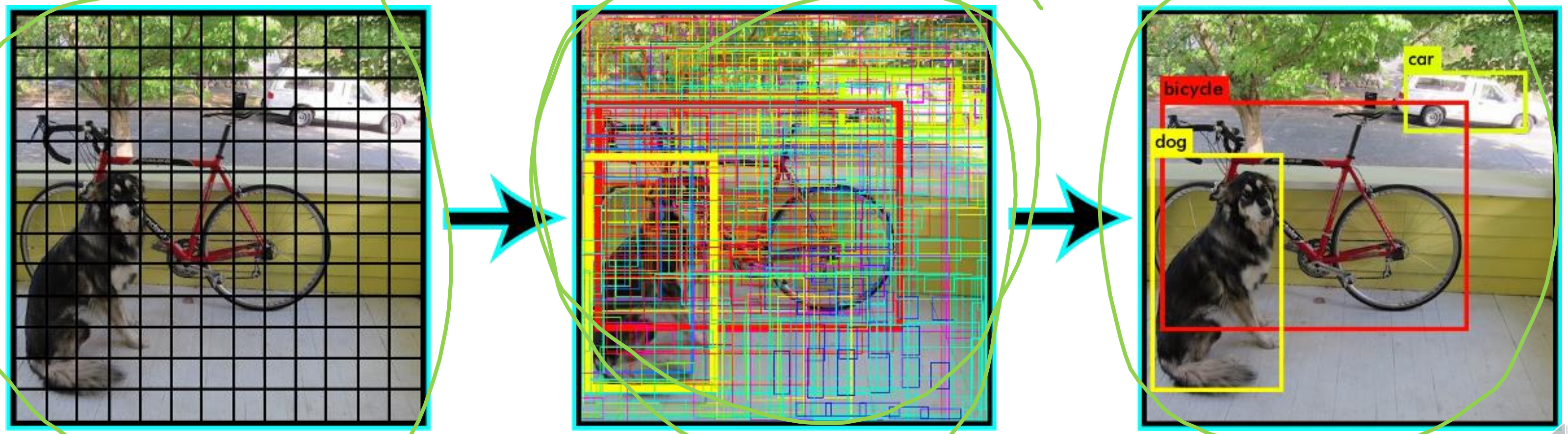
- Each cell has 20 conditional class probabilities. The **conditional class probability** is the probability that the detected object belongs to a particular class (one probability per category for each cell).

- So, YOLO's prediction has a shape of

$$(S, S, B\times 5 + C) = (7, 7, 2\times 5 + 20) = (7, 7, 30).$$

YOLO—You Only Look Once

- The main concept of YOLO is to build a CNN network to predict a (7, 7, 30) tensor.
- It uses a CNN network to reduce the spatial dimension to 7×7 with 1024 output channels at each location.
- YOLO performs a linear regression using two fully connected layers to make 7×7×2 boundary box predictions (the middle picture below).
- To make a final prediction, we keep those with high box confidence scores (greater than 0.25) as our final predictions (the right picture).

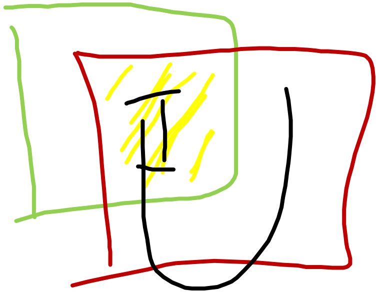


YOLO—You Only Look Once

- There are many predictions and they need to be pruned. This is done using **class confidence scores**:

$$\text{class confidence score} = \text{box confidence score} \times \text{conditional class probability}$$

- It measures the confidence on both the classification and the **localization** (where an object is located).



$$\begin{aligned}
\text{box confidence score} &\equiv P_r(object) \cdot IoU \\
\text{conditional class probability} &\equiv P_r(class_i | object) \\
\text{class confidence score} &\equiv P_r(class_i) \cdot IoU \\
&= \text{box confidence score} \times \text{conditional class probability}
\end{aligned}$$

where

$P_r(object)$ is the probability the box contains an object.
IoU is the IoU (intersection over union) between the predicted box and the ground truth.
$P_r(class_i | object)$ is the probability the object belongs to $class_i$ given an object is presence.
$P_r(class_i)$ is the probability the object belongs to $class_i$

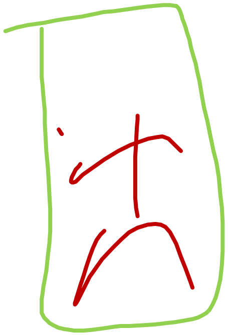
YOLO—You Only Look Once

- Then these scores are sorted and thresholded to remove the predictions with lower confidence scores.

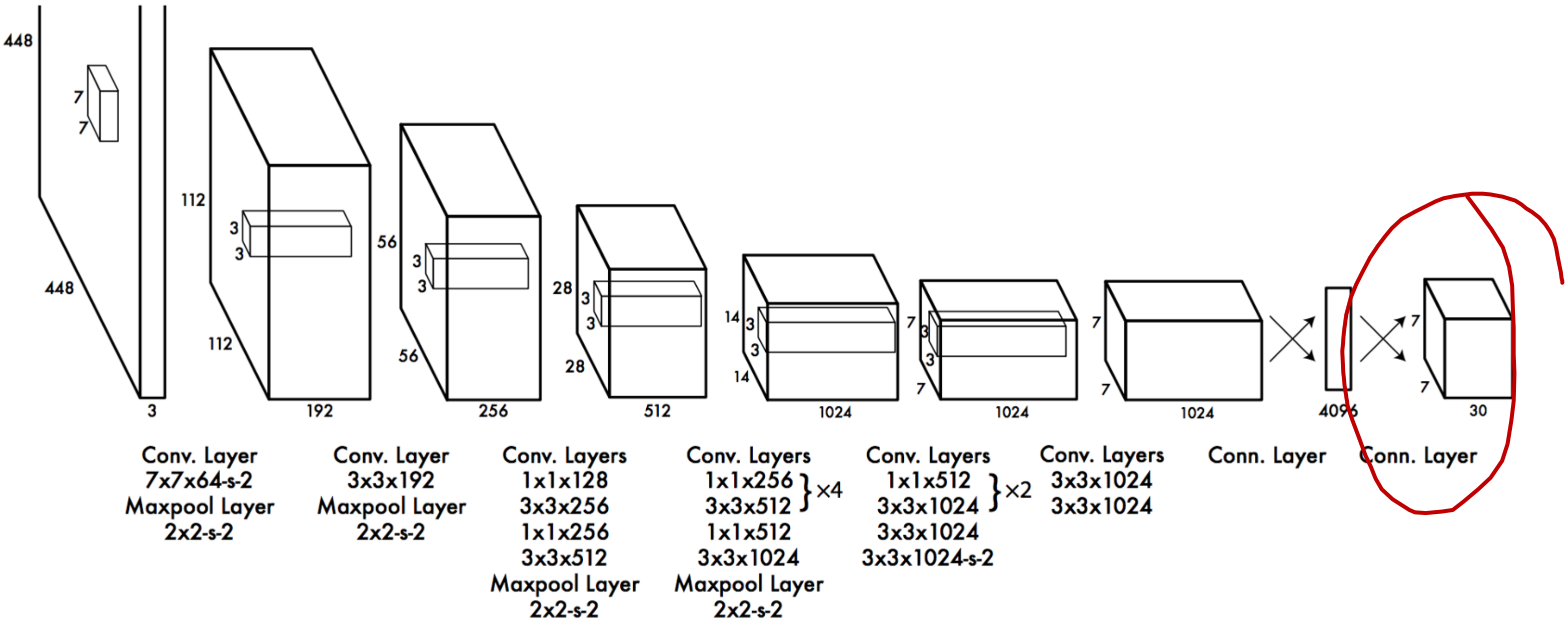
- Remaining predictions are fed into non-maximum suppression algorithm.

- Non-max suppression merges the detections belonging to the same object.

NMS



YOLO—You Only Look Once



- YOLO has 24 convolutional layers followed by 2 fully connected layers (FC).
- A faster but less accurate version of YOLO, called Fast YOLO, uses only 9 convolutional layers with shallower feature maps.

YOLO Loss Function

Bounding Box Location

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

Bounding Box Size

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

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

Confidence

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

Predicted Class Probability

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

YOLO Loss Function

- *λ* are constants.

- As most grids do not have objects λ_{obj} reduce the contribution of no object grids.

- $\mathbb{1}_i^{\text{obj}}$ is 1when there is an object in cell i and 0 elsewhere

- $\mathbb{1}_{ij}^{\text{obj}}$ is 1 if there is an object in cell *i* and confidence of the *j*th predictors of this cell is the highest among all the predictors of this cell.

- $\mathbb{1}_{ij}^{\text{noobj}}$ is the same except it values 1 when there are NO objects in cell *i*

YOLOv3

- Most classifiers assume output labels are mutually exclusive. It is true if the output are mutually exclusive object classes. Therefore, earlier YOLO versions used softmax to convert scores into probabilities that sum up to one.

- YOLOv3 uses multi-label classification. For example, the output labels may be “pedestrian” and “child” which are not non-exclusive. (the sum of output can be greater than 1 now.)

- YOLOv3 replaces the softmax function with independent logistic classifiers to calculate the likeliness of the input belongs to a specific label.

- Instead of using mean square error in calculating the classification loss, YOLOv3 uses binary cross-entropy loss for each label. This also reduces the computation complexity by avoiding the softmax function.

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLOv3

- YOLOv3 predicts an objectness score for each bounding box using logistic regression.

- YOLOv3 changes the way in calculating the cost function. If the bounding box prior (anchor) overlaps a ground truth object more than others, the corresponding objectness score should be 1.

- For other priors with overlap greater than a predefined threshold (default 0.5), they incur no cost.

- Each ground truth object is associated with one boundary box prior only. If a bounding box prior is not assigned, it incurs no classification and localization lost, just confidence loss on objectness. We use tx and ty (instead of bx and by) to compute the loss.

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLOv3

– We use t_x and t_y (instead of b_x and b_y) to compute the loss.

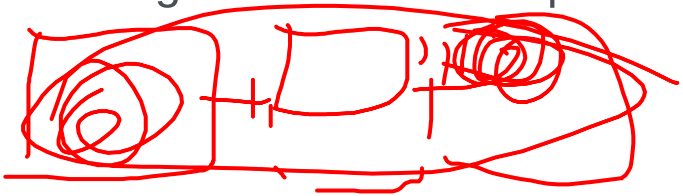
$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLOv3

- YOLOv3 makes 3 predictions per location. Each prediction composes of a boundary box, a objectness and 80 class scores, i.e. $N \times N \times [3 \times (4 + 1 + 80)]$ predictions.

- YOLOv3 makes predictions at 3 different scales (similar to the FPN):
- In the last feature map layer.
- Then it goes back 2 layers back and upsamples it by 2. YOLOv3 then takes a feature map with higher resolution and merge it with the upsampled feature map using element-wise addition. YOLOv3 apply convolutional filters on the merged map to make the second set of predictions.
- Repeat 2 again so the resulted feature map layer has good high-level structure (semantic) information and good resolution spatial information on object locations.



https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLOv3

- To determine the priors, YOLOv3 applies k-means cluster. Then it pre-selects 9 clusters.

- For COCO, the width and height of the anchors are (10×13),(16×30),(33×23),(30×61),(62×45),(59× 119),(116 × 90),(156 × 198),(373 × 326). These 9 priors are grouped into 3 different groups according to their scale. Each group is assigned to a specific feature map above in detecting objects.

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLOv3

- A new 53-layer Darknet-53 is used to replace the Darknet-19 as the feature extractor.
- Darknet-53 is mainly composed of 3 × 3 and 1× 1 filters with skip connections like the residual network in ResNet.
- Darknet-53 has fewer BFLOP (billion floating point operations) than ResNet-152, but achieves the same classification accuracy at 2x faster.

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLOv3

– Most popular YOLOv3 implementation is Darknet.

– Darknet is an open source neural network framework written in C and CUDA.

– It supports CPU and GPU computation.

https://github.com/pjreddie/darknet

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

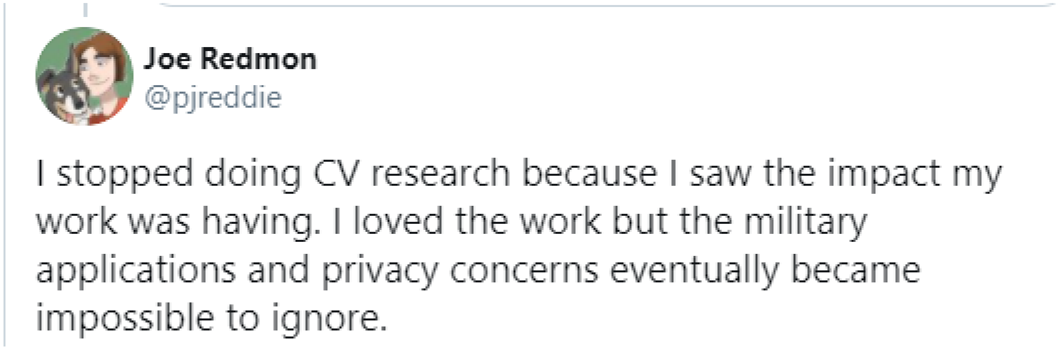
PyTorch Translation of YOLOv3 by Glenn Jocher

https://github.com/ultralytics/yolov3

Newer YOLO versions

In February 2020, Joseph Redmon, the creator of YOLO **announced that he has stopped his research in computer vision**!

However, three major versions of YOLO have been released in 2020: YOLO v4, YOLO v5 and PP-YOLO



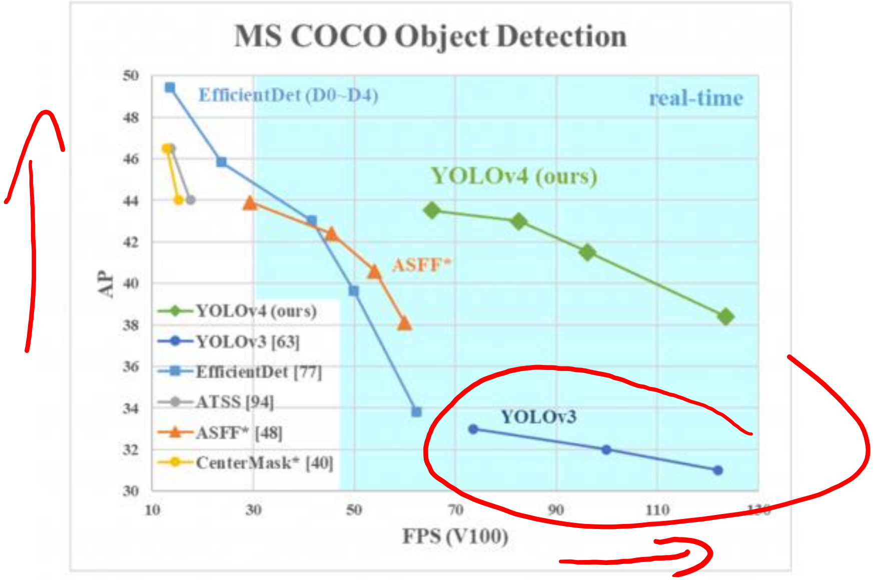
Joe Redmon
@pjreddie

I stopped doing CV research because I saw the impact my work was having. I loved the work but the military applications and privacy concerns eventually became impossible to ignore.

YOLOv4

YOLO v4 makes use of

- BoF (bag of freebies) improve the accuracy of the detector, without increasing the inference time. They only increase the training cost.
- BoS (bag of specials) which increase the inference cost by a small amount however they significantly improve the accuracy of object detection.



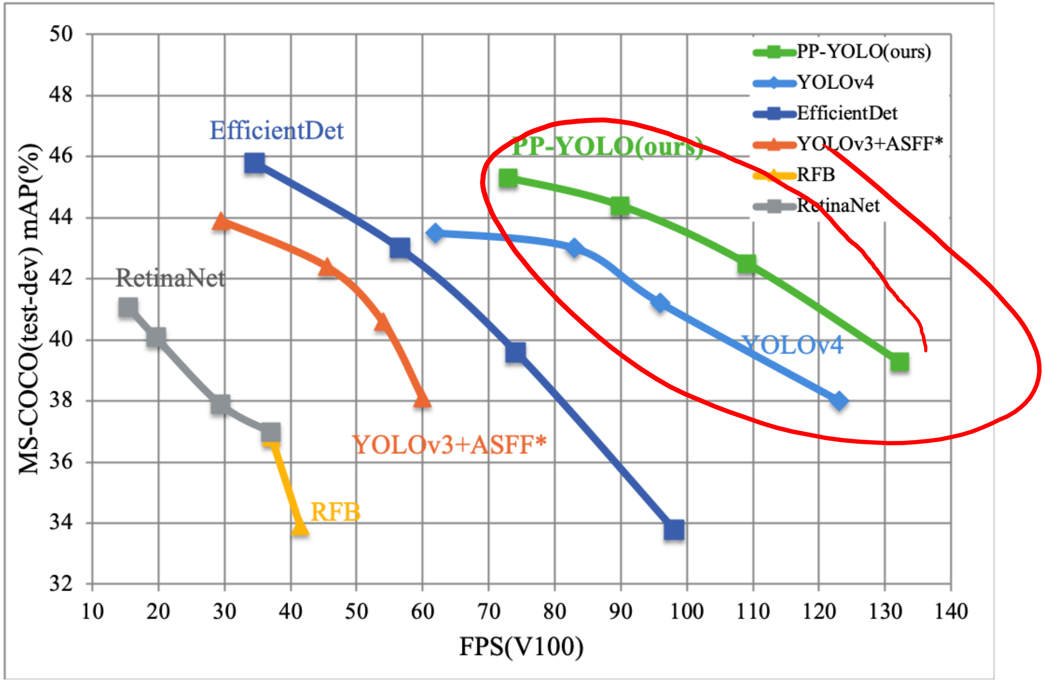
Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*.

YOLOv5

- Created by Glenn Jocher, implentor of the popular PyTorch implementation of YOLO v3
- There is no paper accompanying the code!
- The major improvements includes mosaic data augmentation and auto learning bounding box anchors.
- It is reported to be faster and lighter-weight than YOLOv4

PP-YOLO

- It is based on PaddlePaddle (Parallel Distributed Deep Learning), an open source deep learning platform originally developed by Baidu.
- Based on YOLOv3, replaced Darknet53 backbone with a ResNet backbone and increase of training batch size from 64 to 192 (as mini-batch size of 24 on 8 GPUs).



Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., ... & Wen, S. (2020). PP-YOLO: An Effective and Efficient Implementation of Object Detector. *arXiv preprint arXiv:2007.12099*

RetinaNet

- There is extreme foreground-background class imbalance problem in one-stage detectors. This is the central cause which makes the performance of one-stage detectors inferior to two-stage detectors.

- In RetinaNet, an one-stage detector, by using **focal loss,** lower loss is contributed by “easy” negative samples so that the loss is focusing on “hard” samples, which improves the prediction accuracy.

BG ↑

object ↓

Lin, Tsung-Yi, et al. "Focal loss for dense object detection." *Proceedings of the IEEE international conference on computer vision*. 2017.

https://towardsdatascience.com/review-retinanet-focal-loss-object-detection-38fba6afabe4

Object Detection Performance
Precision & recall

Precision: Accuracy of predictions, i.e. the percentage of correct predictions
Recall: How good are the detection of positives.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$$TP = \text{True positive}$$

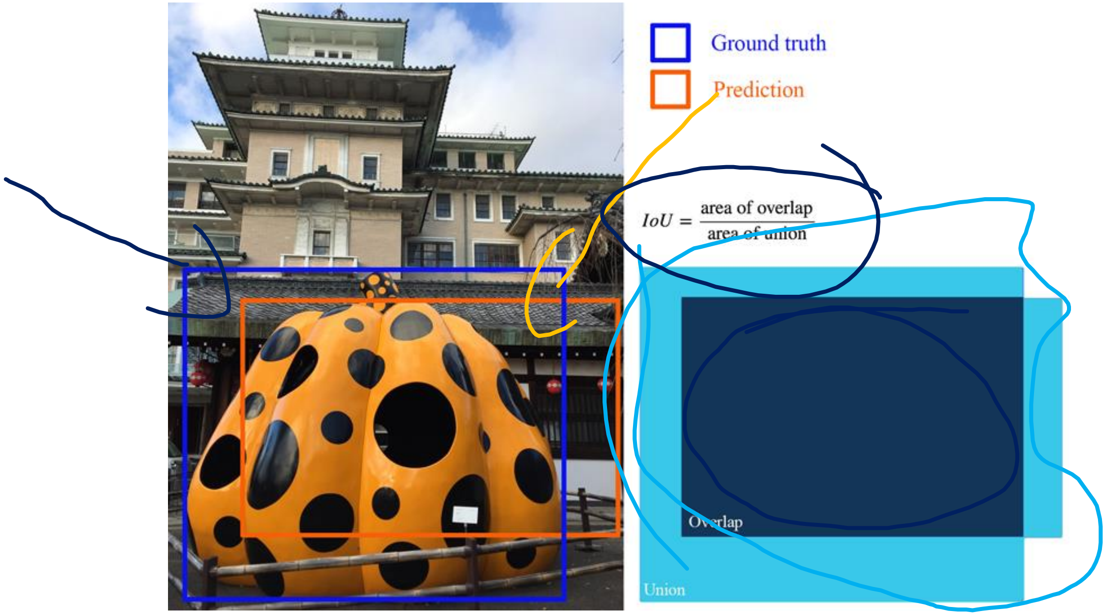
$$TN = \text{True negative}$$

$$FP = \text{False positive}$$

$$FN = \text{False negative}$$

Object Detection Performance
Intersection over Union (IoU)

- IoU measures the overlap between 2 bounding boxes.
- Measures how much the predicted bounding box overlaps with the ground truth bounding box.
- Also known as Jaccard index
- In segmentation/detection tasks the IoU is preferred over accuracy as it is not as affected by the class imbalances that are inherent in foreground/background based tasks.



https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

Object Detection Performance Intersection over Union (IoU)

An interactive tool to understand IoU:

https://calebrob.com/ml/2018/09/11/understanding-iou.html

https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

Object Detection Performance
Intersection over Union (IoU)

FP

A threshold is used to detect whether a prediction is a TP or FP.

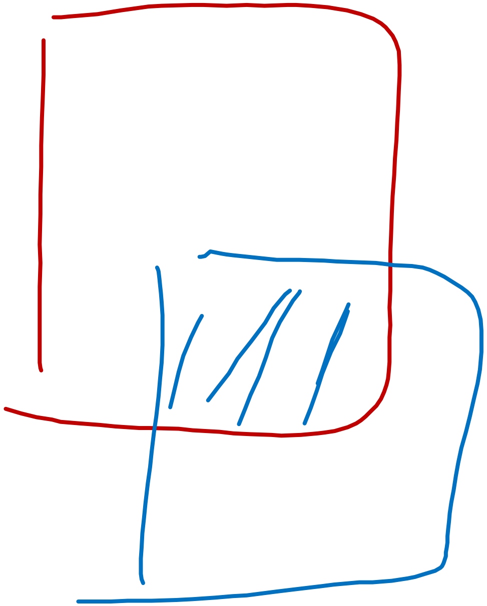
With a threshold of *T*

IoU > *T* ➔ a true positive TP

IoU < *T* ➔ a false positive FP

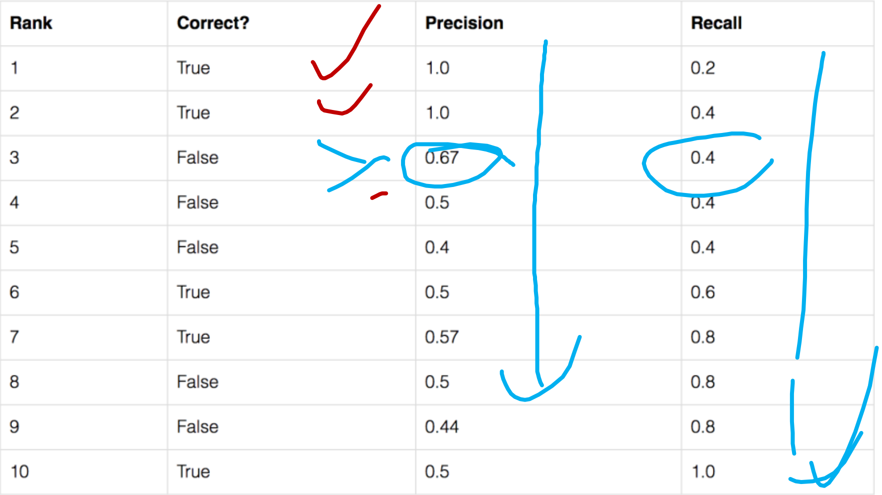
IoU > *T* but object is miss classified ➔ a false negative.

FP



https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

Object Detection Performance
Average Precision

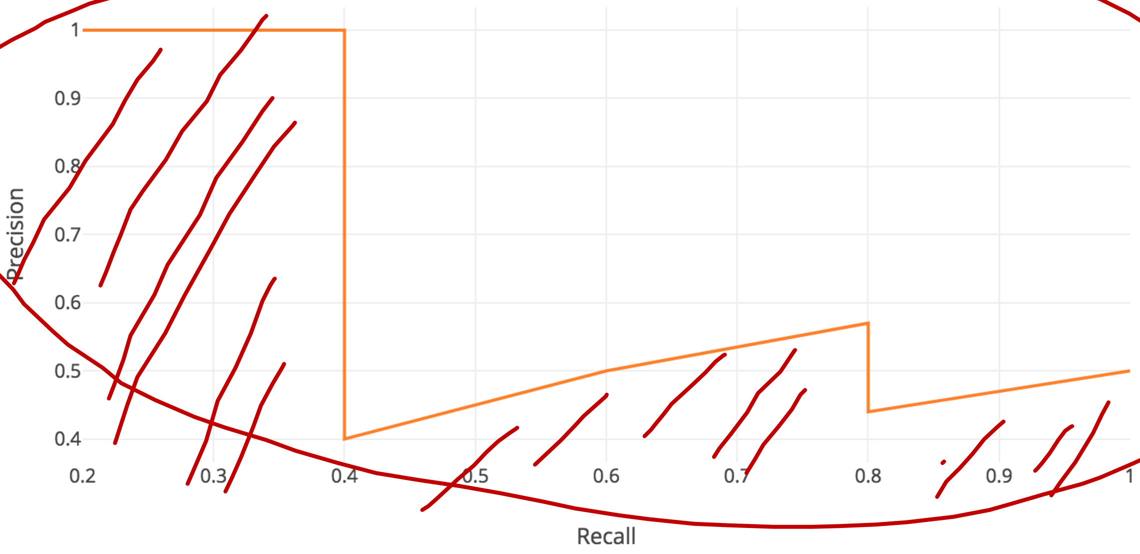


Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

We have 5 TRUE classes
We collect all the predictions made for apples in all the images and rank it in descending order according to the predicted confidence level.

At Rank#3:
Precision: TP/TP+FP = 2/3 = 0.67
Recall: TP/TP+FN = 2/5 = 0.4

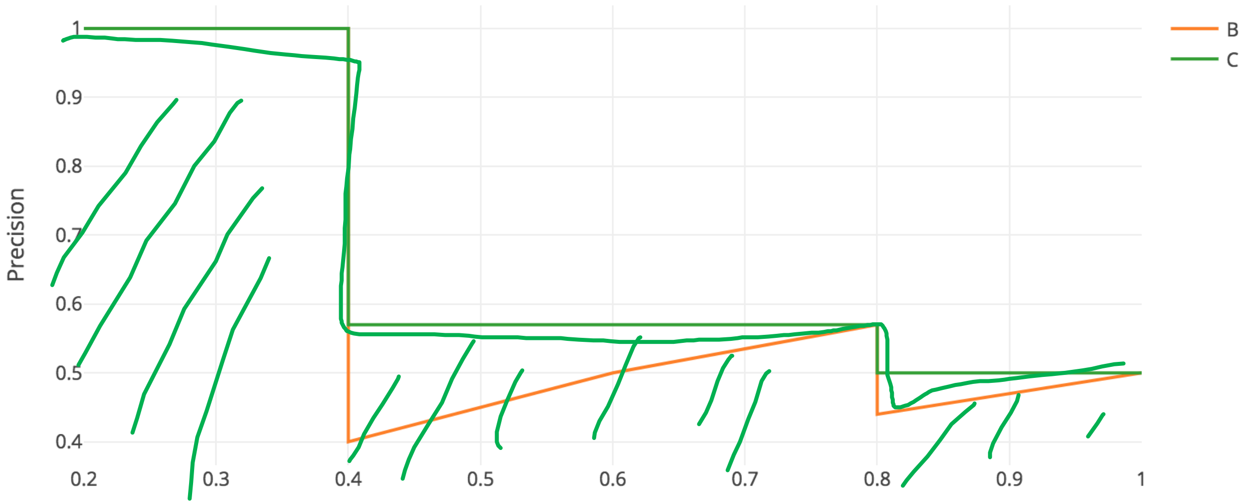
Object Detection Performance
Average Precision



Average Precision (AP) is the area under the precision-recall curve.
Precision and recall are always between 0 and 1. Therefore, AP falls within 0 and 1 also.

$$\text{AP} = \int_0^1 p(r)dr$$

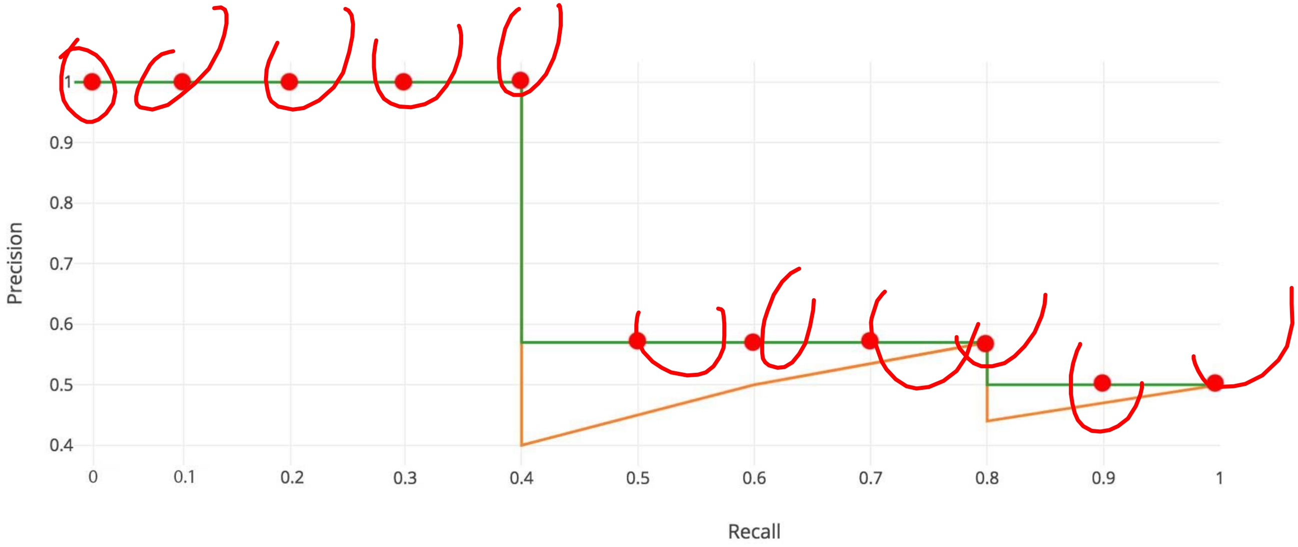
Object Detection Performance
Average Precision



Before calculating AP for the object detection, we often smooth out the zigzag pattern first.

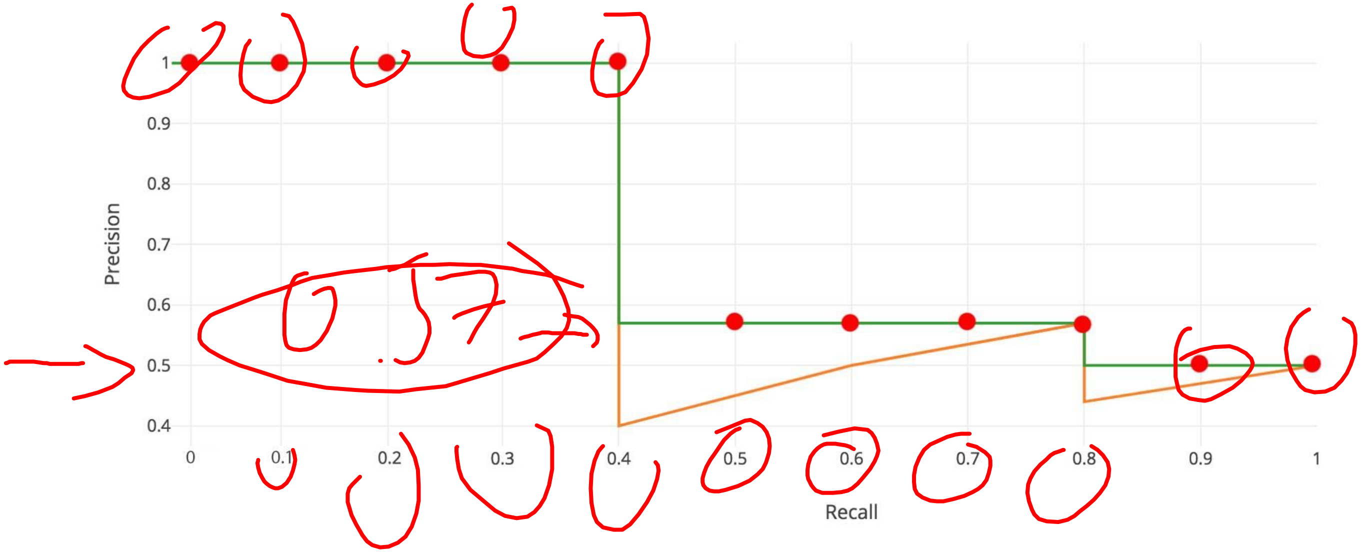
The curve will decrease monotonically instead of the zigzag pattern. The calculated AP value will be less susceptible to small variations in the ranking.

Object Detection Performance
PASCAL VOC2008



A prediction is positive if IoU ≥ 0.5.
If multiple detections of the same object are detected, it counts the first one as a positive while the rest as negatives.
an average for the 11-point interpolated AP is calculated.

Object Detection Performance
PASCAL VOC2008



First, we divide the recall value from 0 to 1.0 into 11 points — 0, 0.1, 0.2, …, 0.9 and 1.0. Next, we compute the average of maximum precision value for these 11 recall values

For 20 different classes in PASCAL VOC, we compute an AP for every class and also provide an average for those 20 AP results.

$$AP = \frac{1}{11} \times \left(AP_r(0) + AP_r(0.1) + \ldots + AP_r(1.0)\right)$$

$$AP = (\underline{5 \times 1.0} + \underline{4 \times 0.57} + \underline{2 \times 0.5})/11$$

Object Detection Performance
PASCAL VOC2008

According to the original paper the intention in interpolating the precision/recall curve is:
“to reduce the impact of the “wiggles” in the precision/recall curve, caused by small variations in the ranking of examples.”

However, this interpolation method is an approximation which suffers two issues.

- It is less precise.
- it does not have the capability in measuring the difference for methods with low AP. Therefore, a different AP calculation is adopted after 2008 for PASCAL VOC.

Object Detection Performance
PASCAL VOC2010-2012

the curve at all unique recall values (*r_1, r_2, …*), whenever the maximum precision value drops. With this change, we are measuring the exact area under the precision-recall curve after the zigzags are removed.

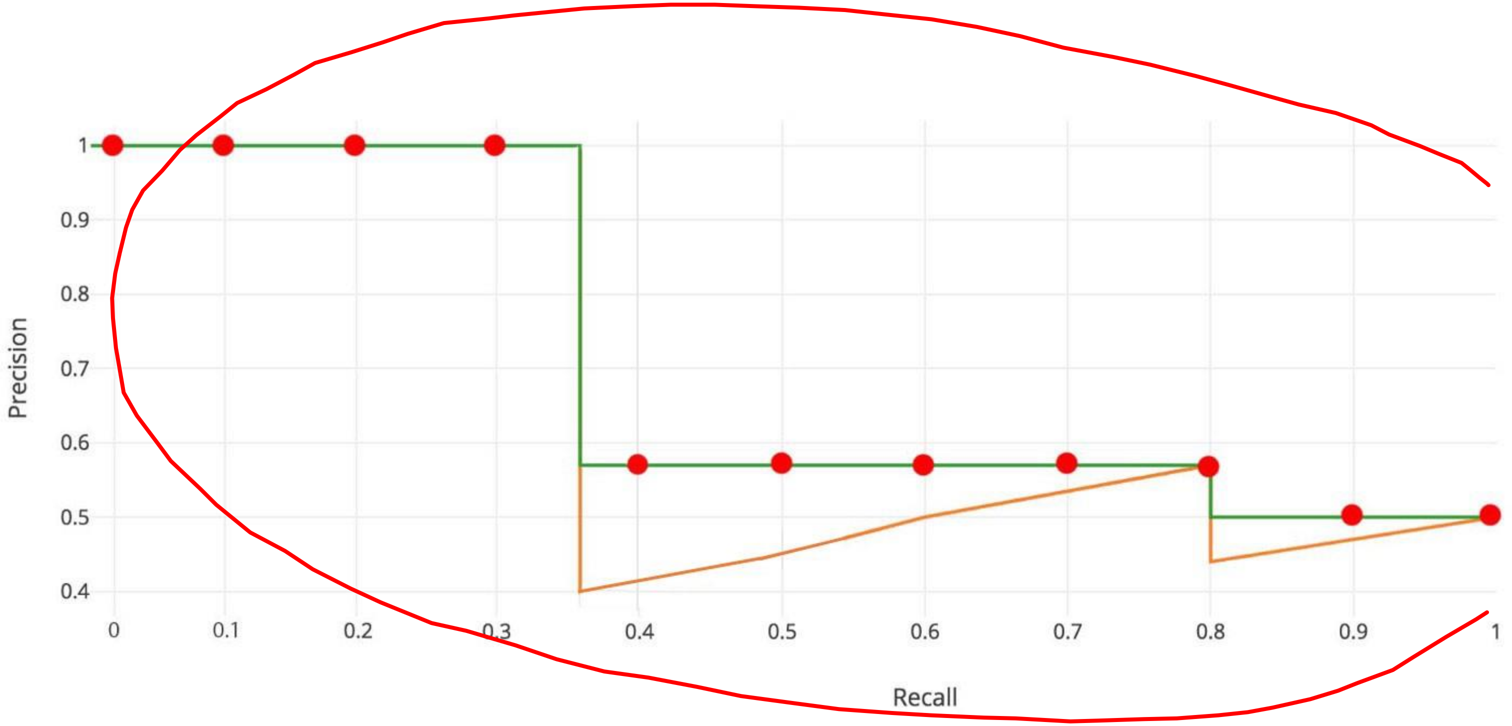
No approximation or interpolation is needed. Instead of sampling 11 points, we sample $p(r_i)$ whenever it drops and computes AP as the sum of the rectangular blocks.

$$\mathrm{AP} = \sum \left(r_{n+1} - r_n \right) p_{interp}(r_{n+1})$$

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

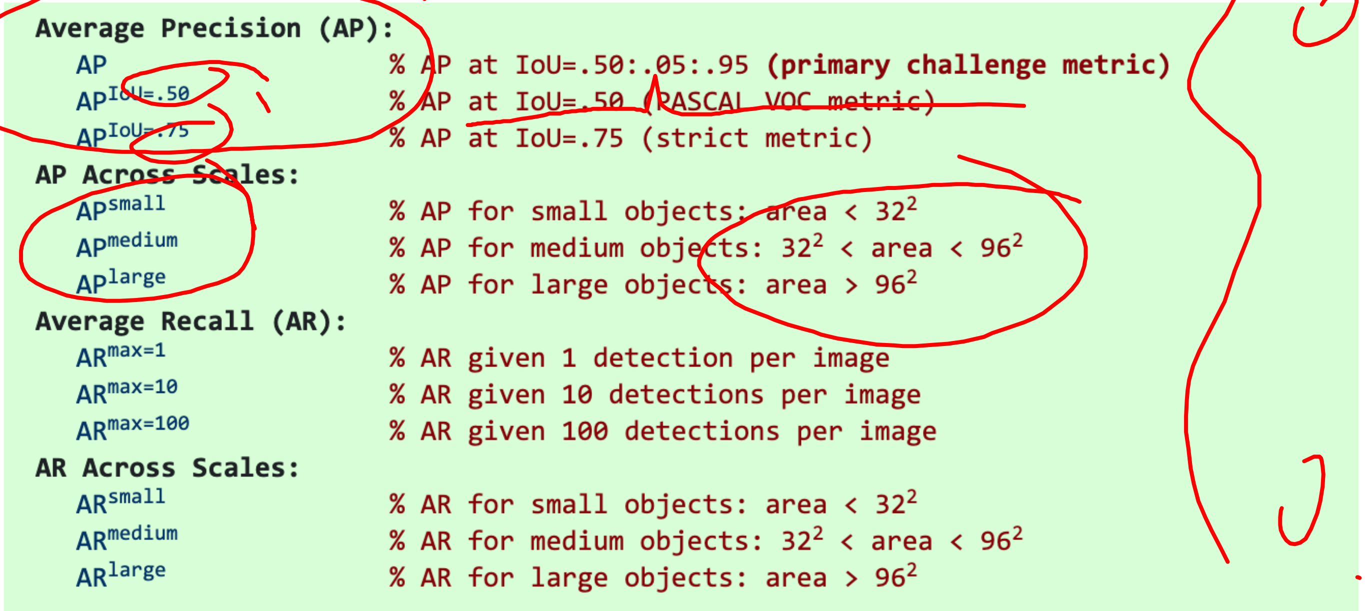
Object Detection Performance PASCAL VOC2010-2012

This definition is called the Area Under Curve (AUC). As shown below, as the interpolated points do not cover where the precision drops, both methods will diverge.



Object Detection Performance
COCO mAP

Latest research papers tend to give results for the COCO dataset only.
In COCO mAP, a 101-point interpolated AP definition is used in the calculation.
For COCO, AP is the average over multiple IoU (the minimum IoU to consider a positive match).
AP@[.5:.95] corresponds to the average AP for IoU from 0.5 to 0.95 with a step size of 0.05.
For the COCO competition, AP is the average over 10 IoU levels on 80 categories (AP@[.50:.05:.95]: start from 0.5 to 0.95 with a step size of 0.05).



```
Average Precision (AP):
   AP                 % AP at IoU=.50:.05:.95 (primary challenge metric)
   AP^IoU=.50         % AP at IoU=.50 (PASCAL VOC metric)
   AP^IoU=.75         % AP at IoU=.75 (strict metric)
AP Across Scales:
   AP^small           % AP for small objects: area < 32^2
   AP^medium          % AP for medium objects: 32^2 < area < 96^2
   AP^large           % AP for large objects: area > 96^2
Average Recall (AR):
   AR^max=1           % AR given 1 detection per image
   AR^max=10          % AR given 10 detections per image
   AR^max=100         % AR given 100 detections per image
AR Across Scales:
   AR^small           % AR for small objects: area < 32^2
   AR^medium          % AR for medium objects: 32^2 < area < 96^2
   AR^large           % AR for large objects: area > 96^2
```

0 50
55
.
.
0 95

Object Detection Performance
Mean Average Precision (mAP)

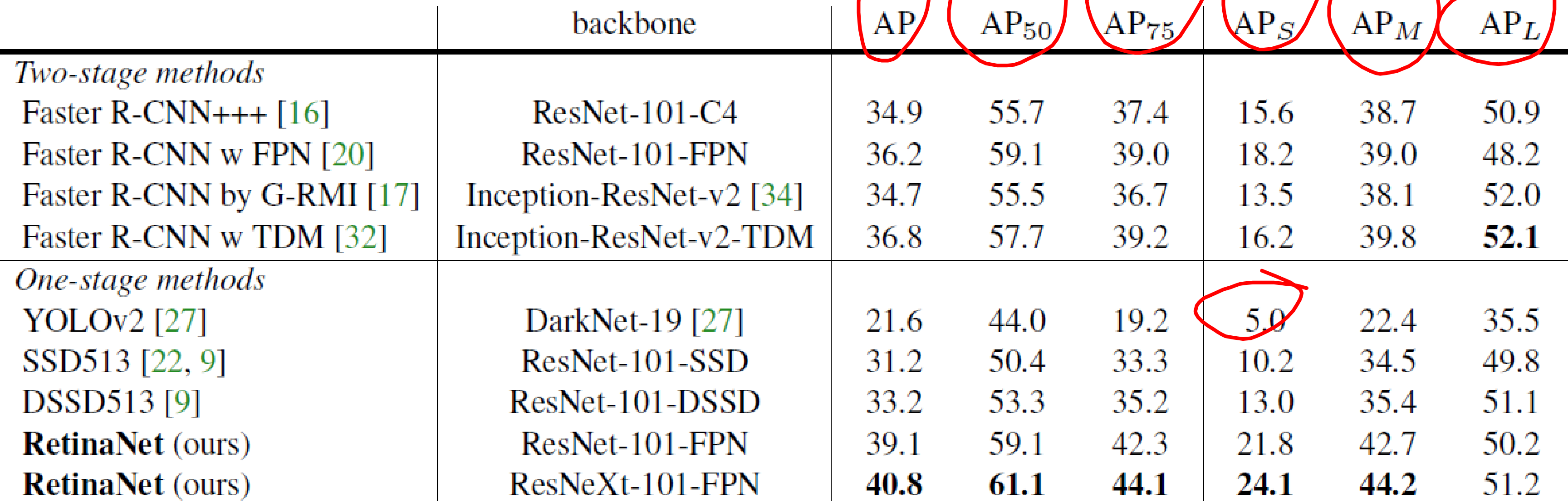
When we have multiple classes:

We can calculate average precision (AP) for each class
This is calculated by taking the average of the maximum precisions at different recall values

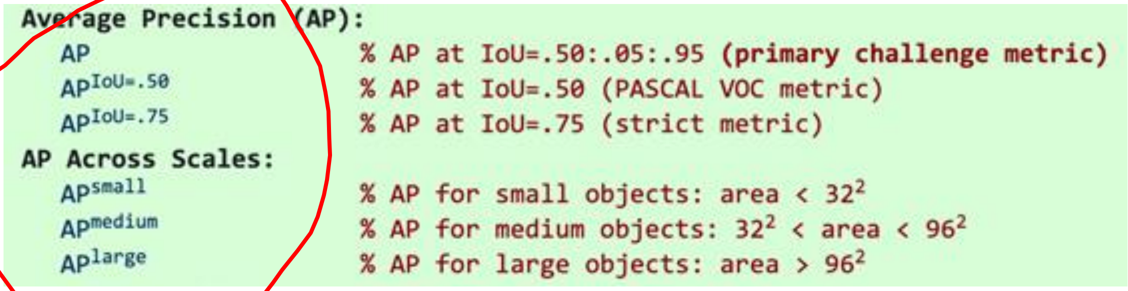
Then we calculate the mean of AP over all classes

https://calebrob.com/ml/2018/09/11/understanding-iou.html

RetinaNet



	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2



Average Precision (AP):

- AP % AP at IoU=.50:.05:.95 (primary challenge metric)
- $AP^{IoU=.50}$ % AP at IoU=.50 (PASCAL VOC metric)
- $AP^{IoU=.75}$ % AP at IoU=.75 (strict metric)

AP Across Scales:

- AP^{small} % AP for small objects: area < 32^2
- AP^{medium} % AP for medium objects: 32^2 < area < 96^2
- AP^{large} % AP for large objects: area > 96^2