Object detection

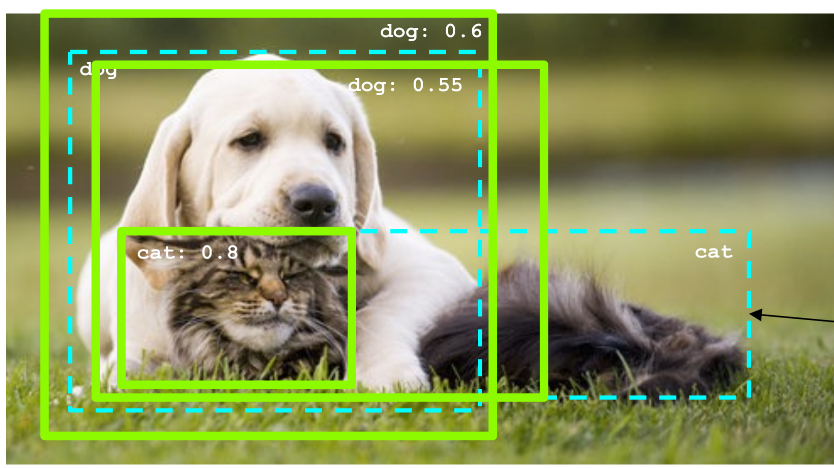
Image source

Outline

- Task definition and evaluation
- Two-stage detectors:
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single-stage and multi-resolution detectors
- Other detectors: CornerNet, DETR

Object detection evaluation

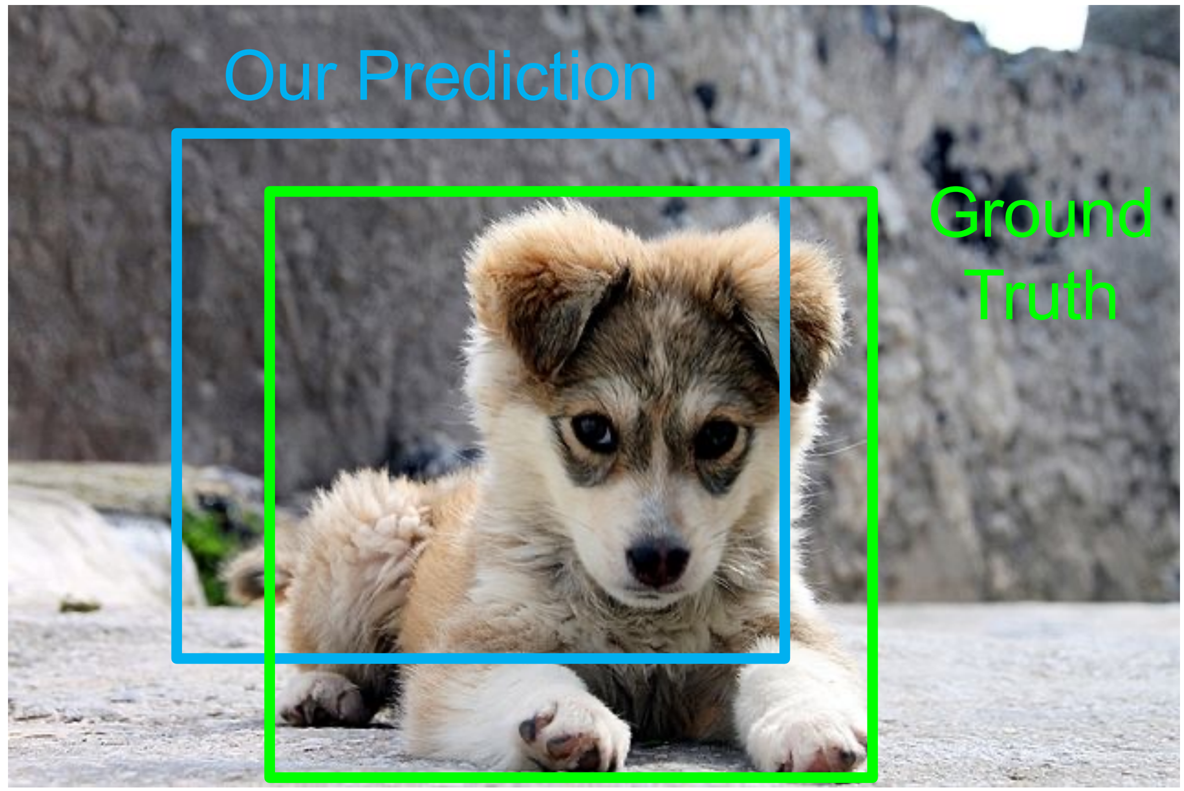
- At test time, predict bounding boxes, class labels, and confidence scores
- For each detection, determine whether it is a true or false positive
    - But how?



Ground truth (GT)

Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?



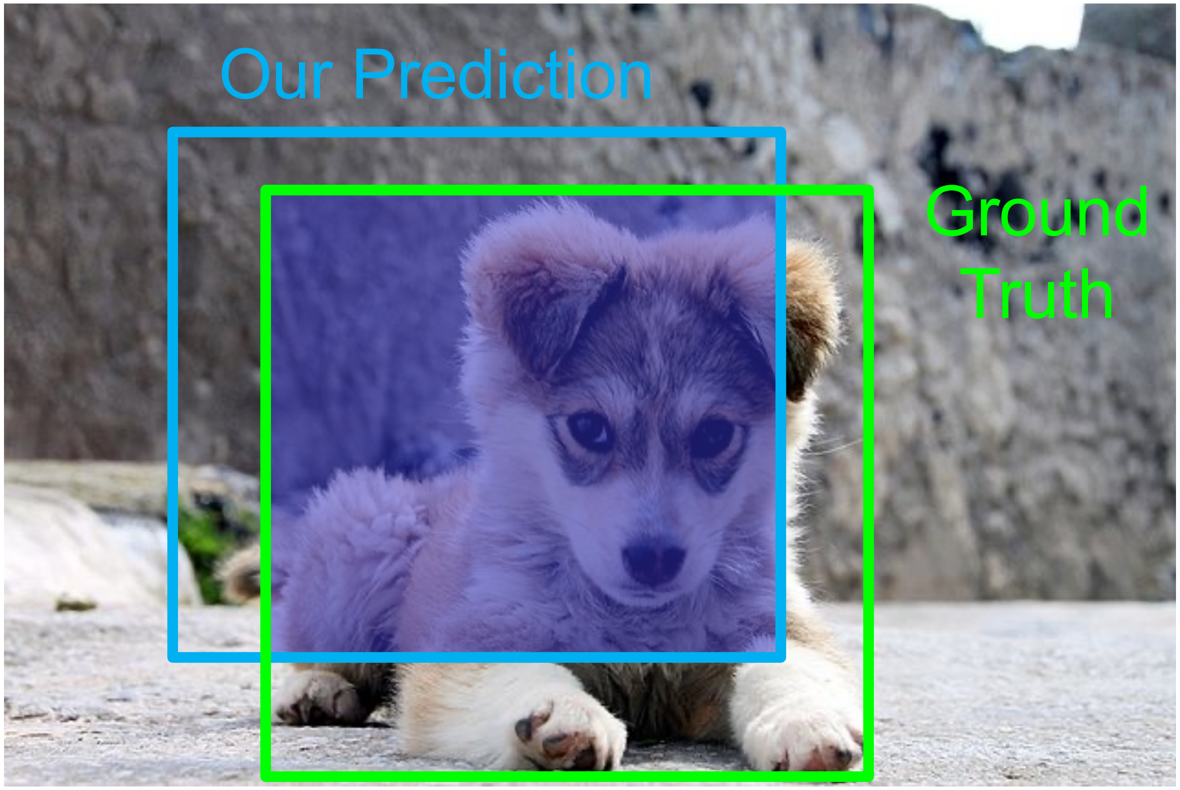
Source: J. Johnson

Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$



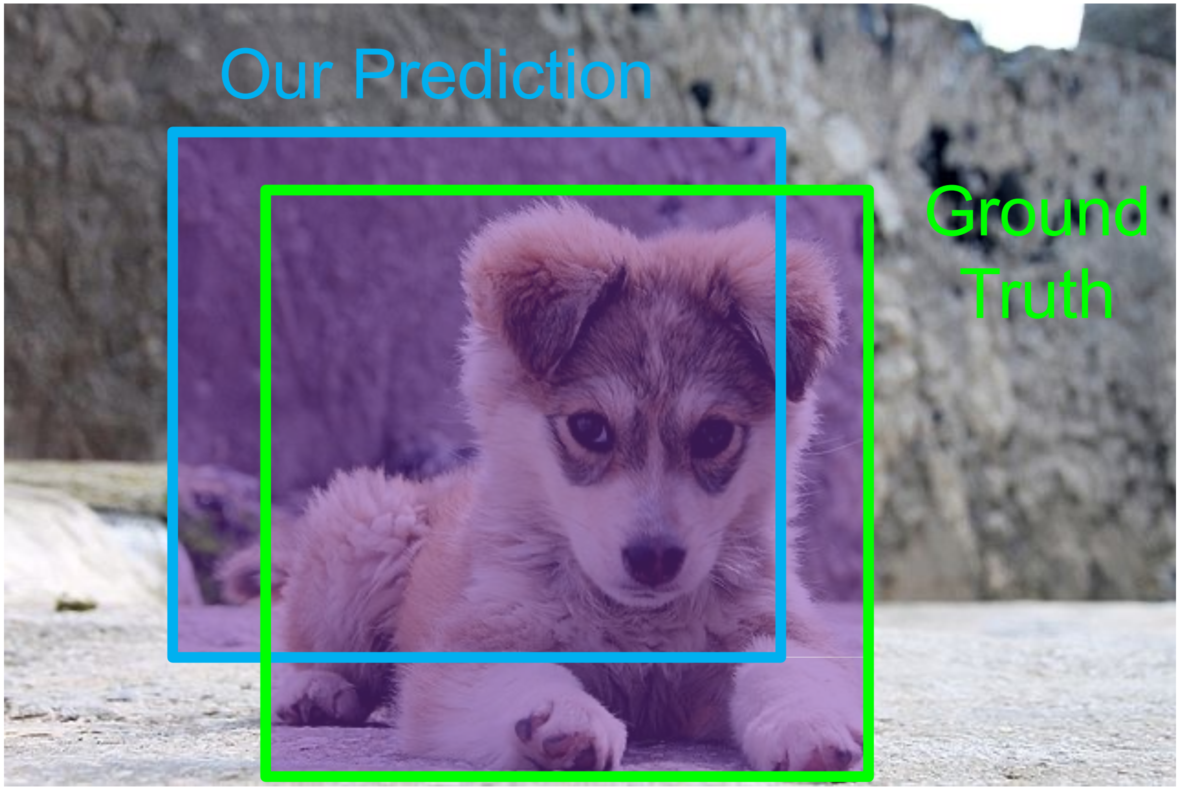
Source: [J. Johnson]

Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

$$\frac{Area\ of\ Intersection}{Area\ of\ Union}$$



Source: J. Johnson

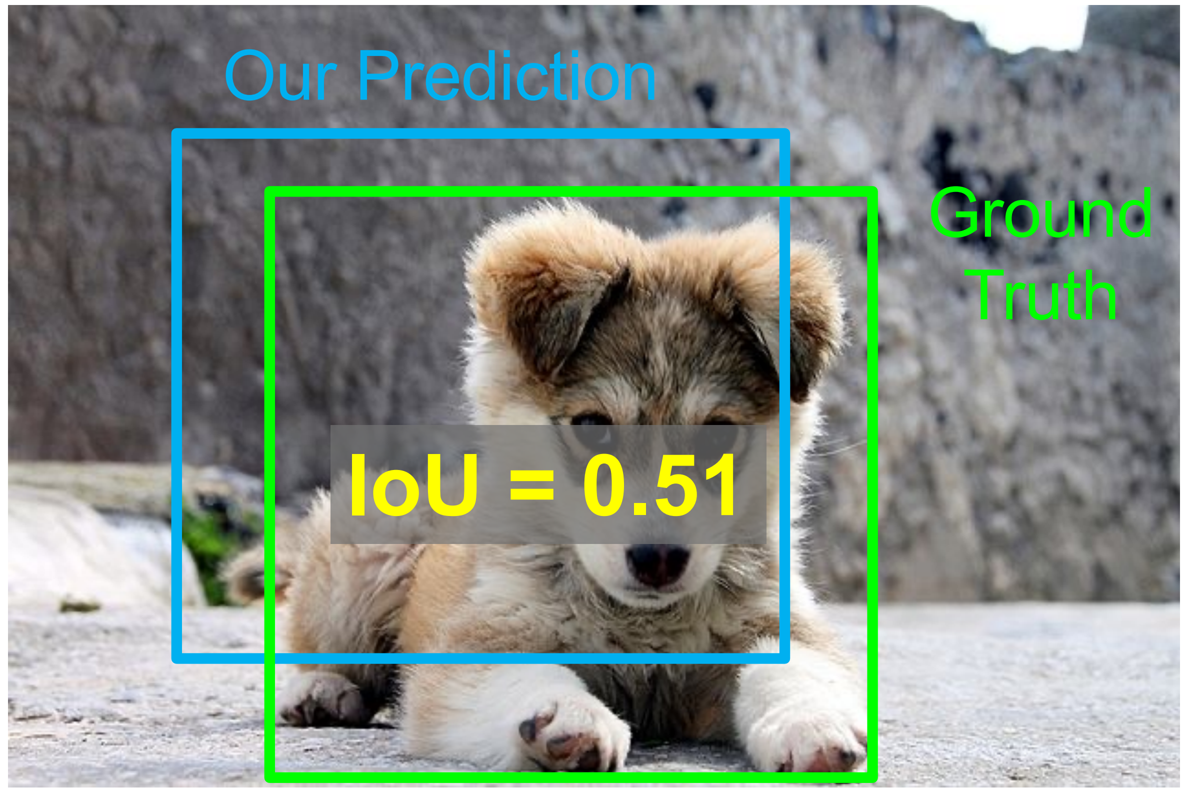
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

$$\frac{Area\ of\ Intersection}{Area\ of\ Union}$$

IoU > 0.5 is “decent”



Source: [J. Johnson]

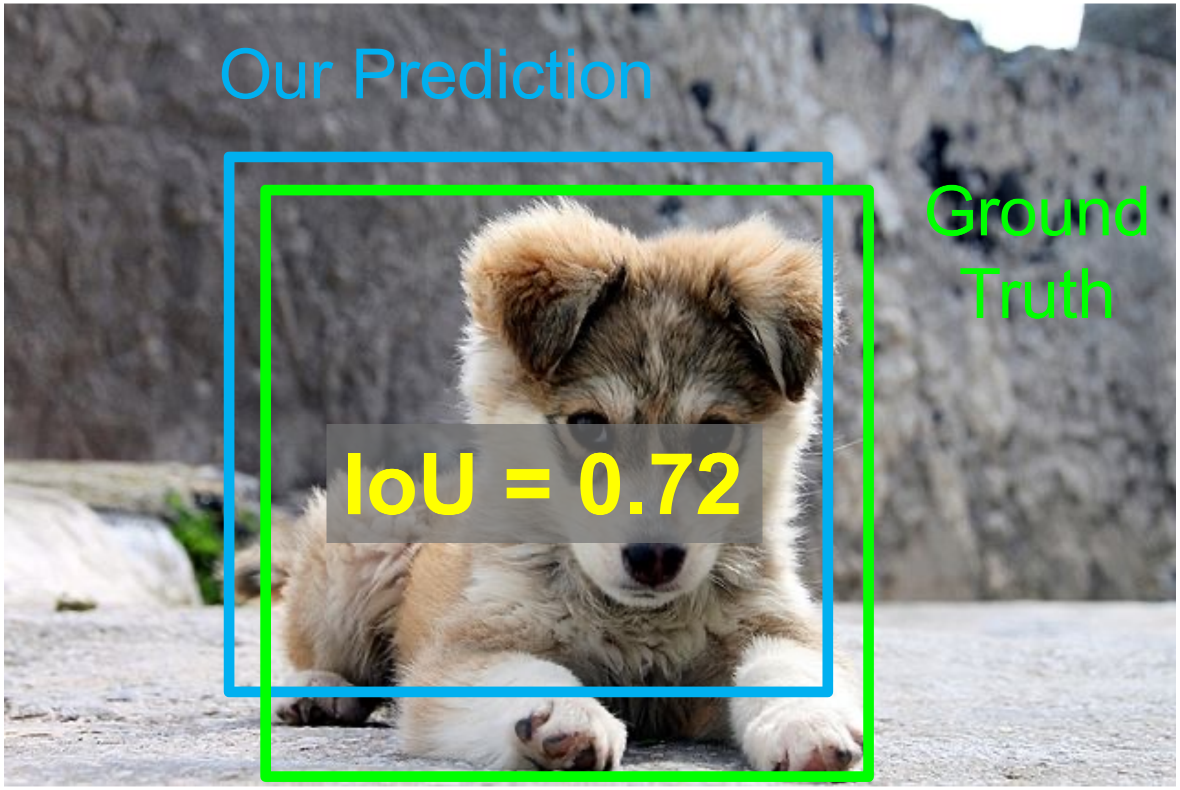
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is “decent”
IoU > 0.7 is “pretty good”



Source: [J. Johnson]

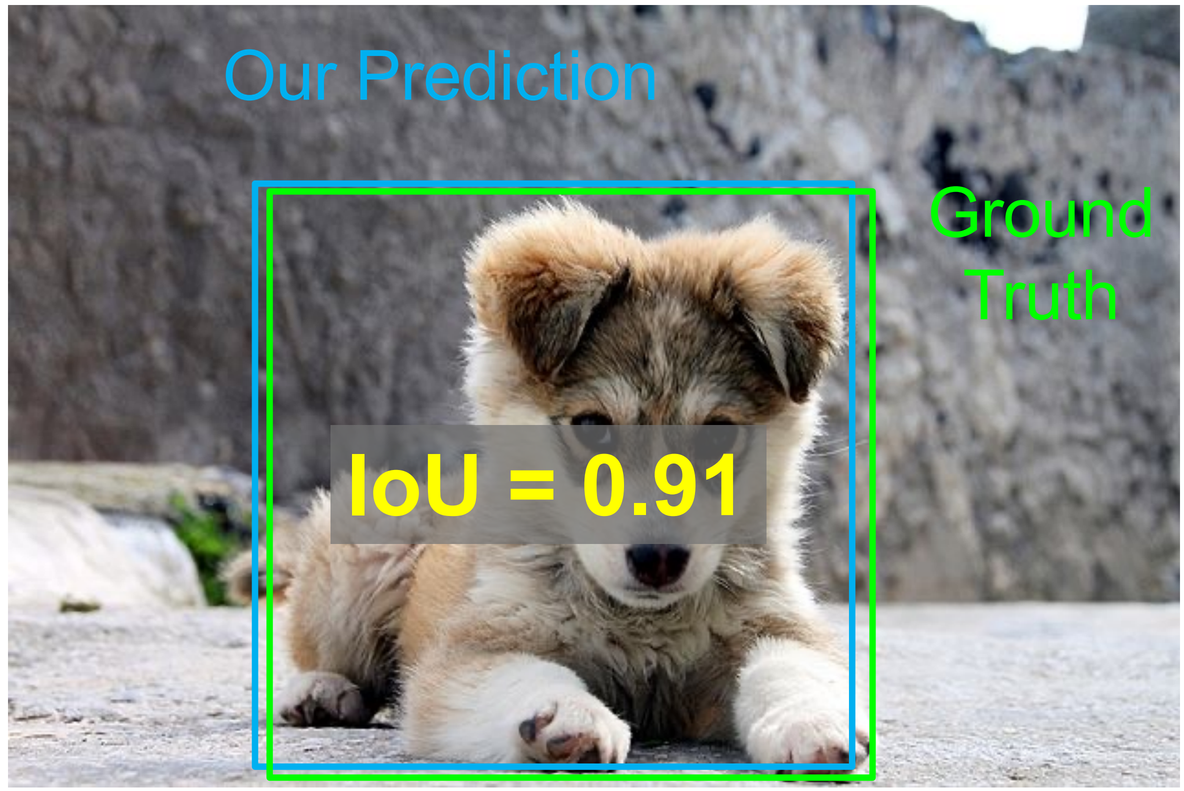
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

$$\frac{Area\ of\ Intersection}{Area\ of\ Union}$$

IoU > 0.5 is "decent"
IoU > 0.7 is "pretty good"
IoU > 0.9 is "almost perfect"

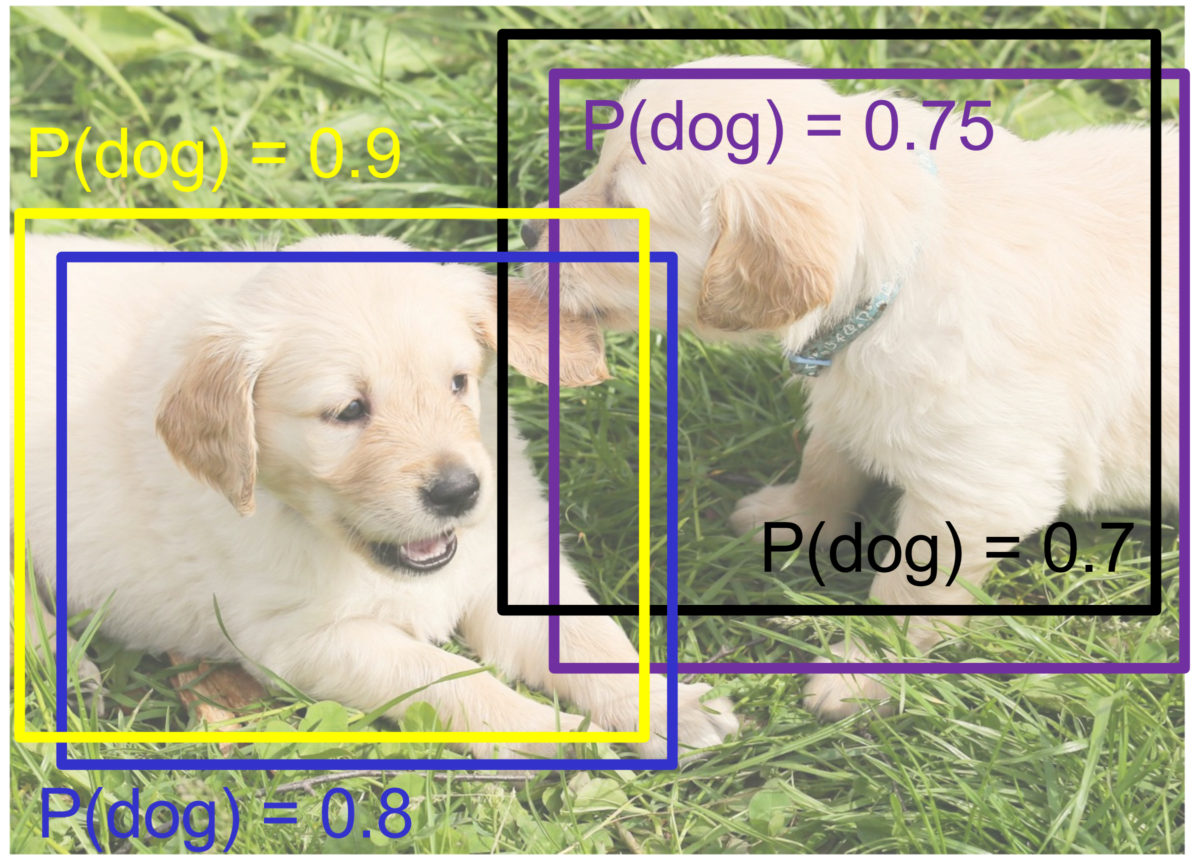


Source: [J. Johnson]

Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)



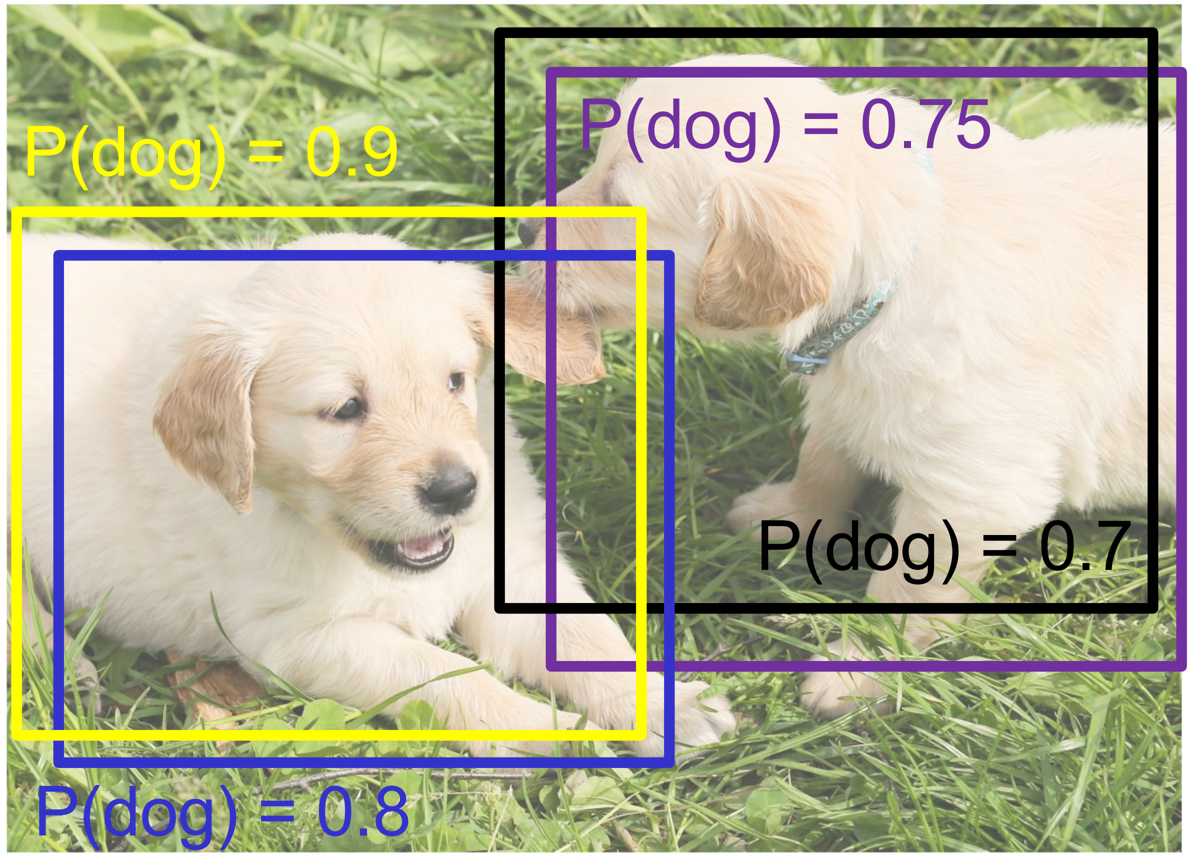
Source: J. Johnson

Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



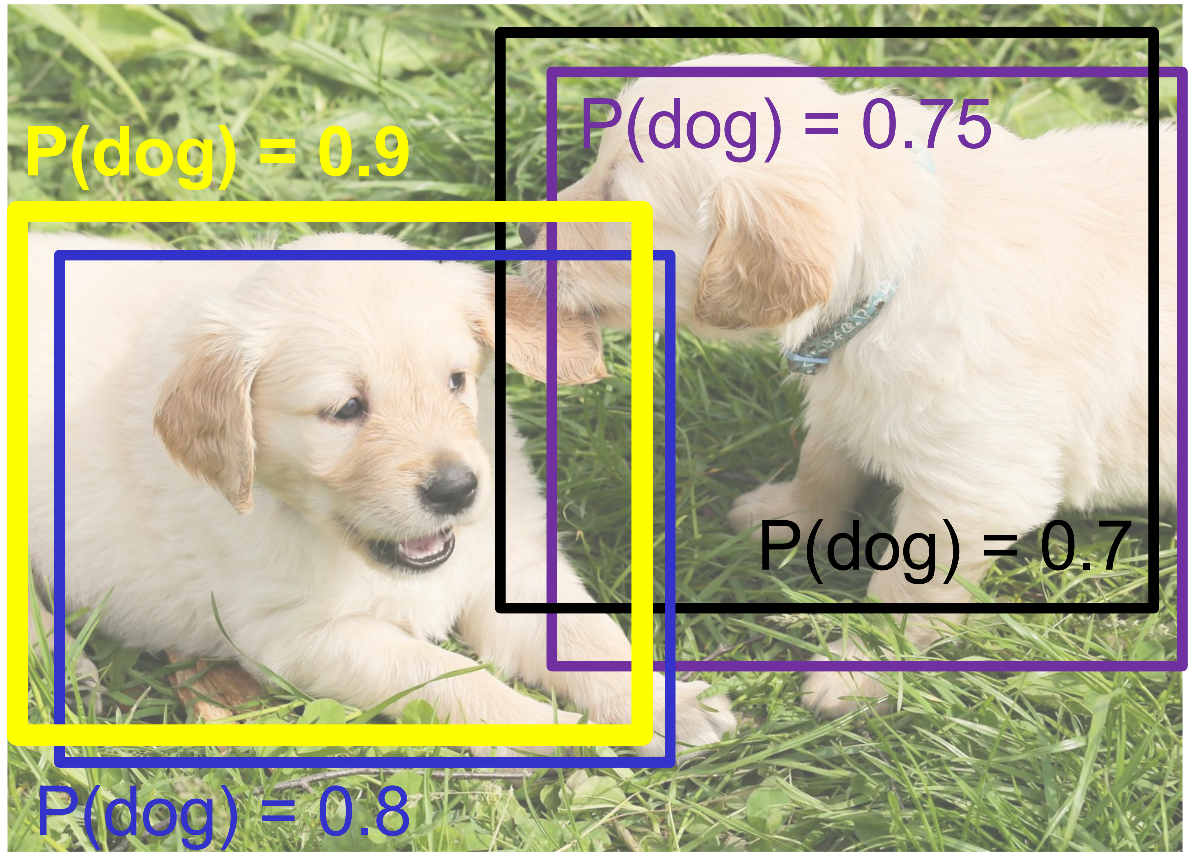
Source: [J. Johnson]

Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



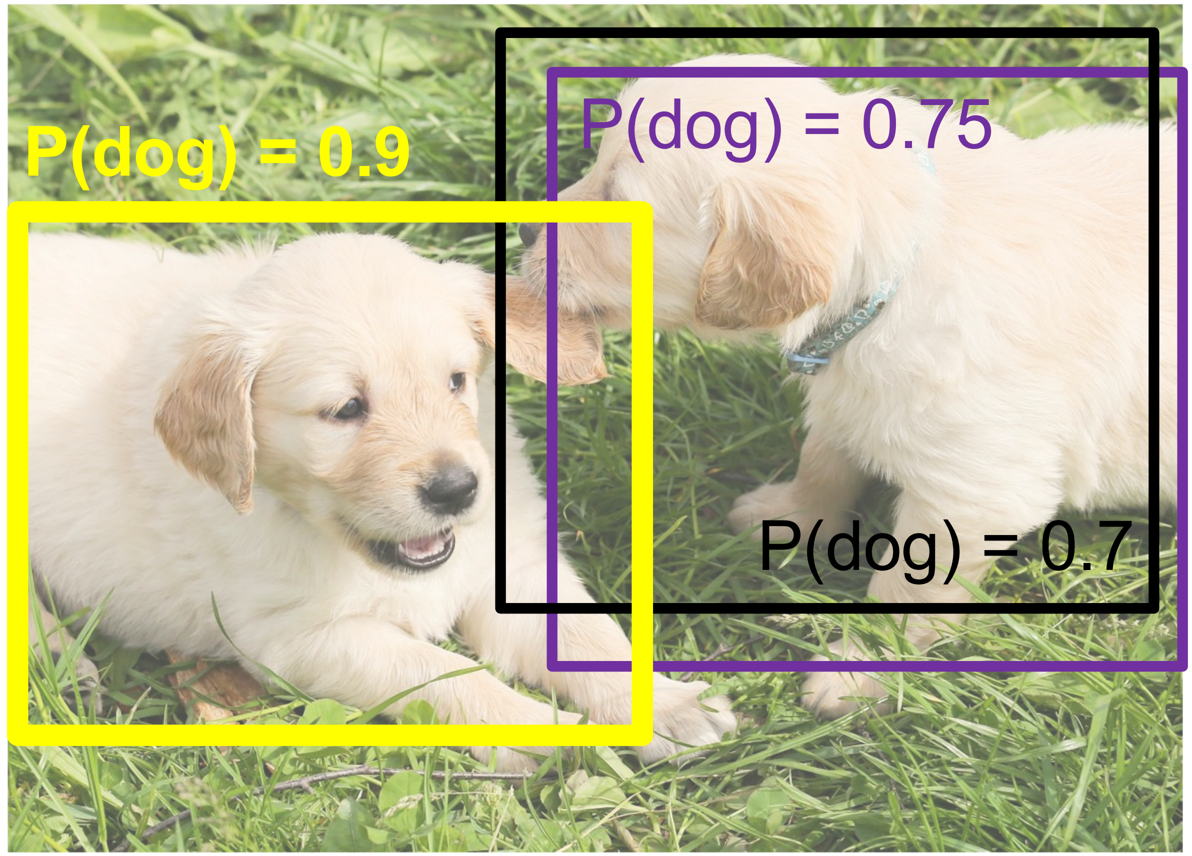
Source: [J. Johnson]

Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



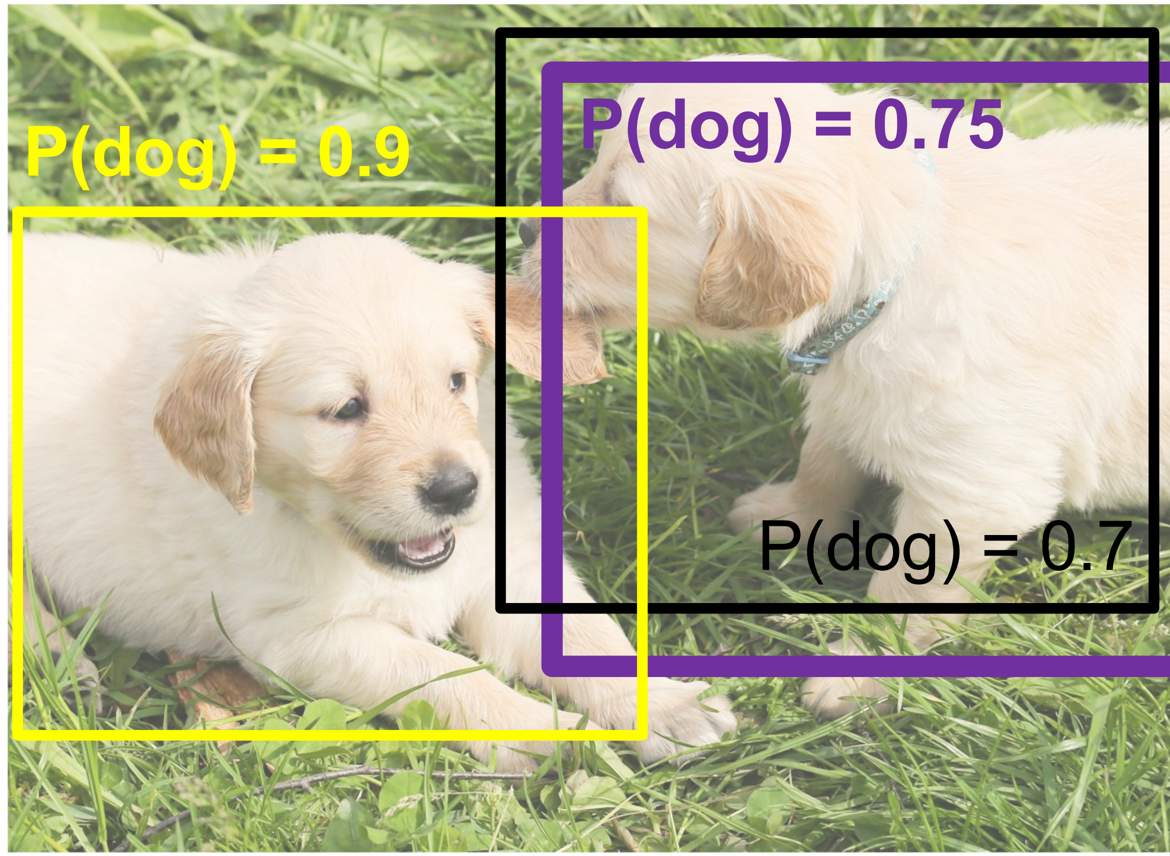
Source: [J. Johnson]

Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



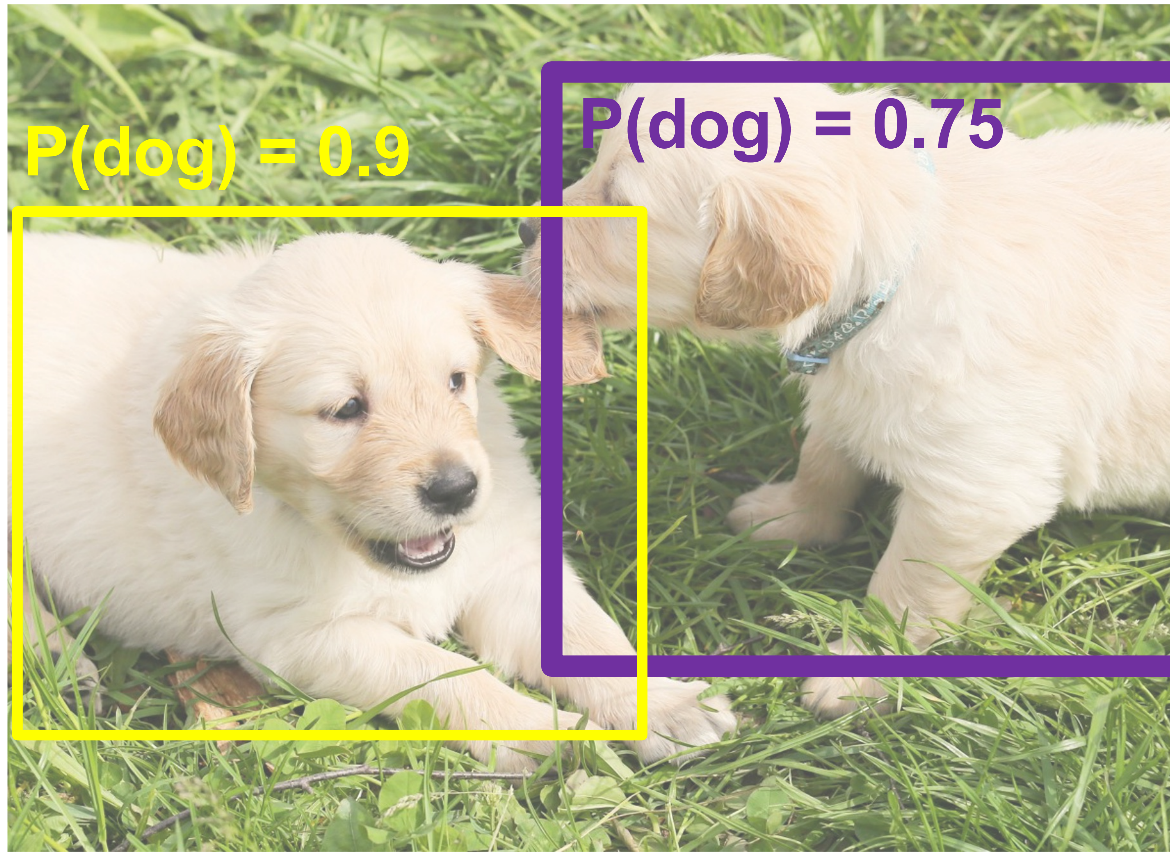
Source: [J. Johnson]

Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



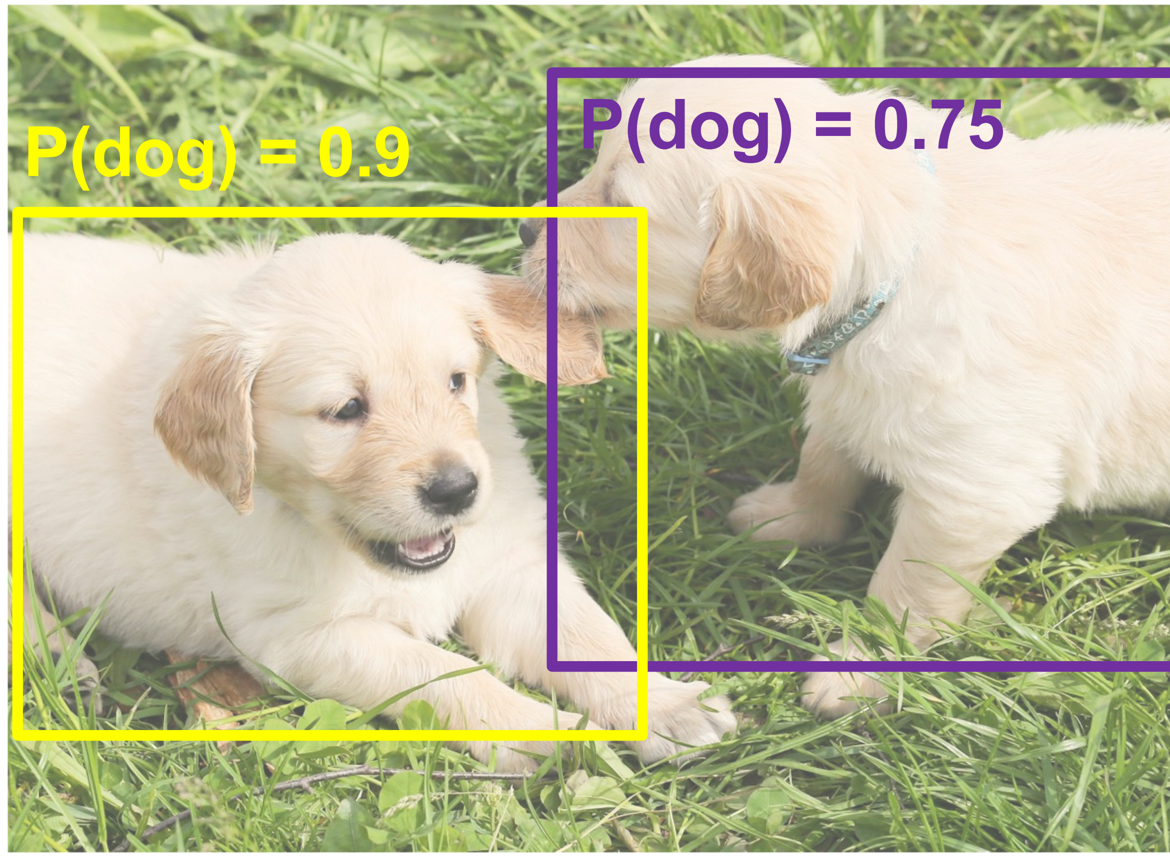
Source: J. Johnson

Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Source: J. Johnson

Non-maximum suppression

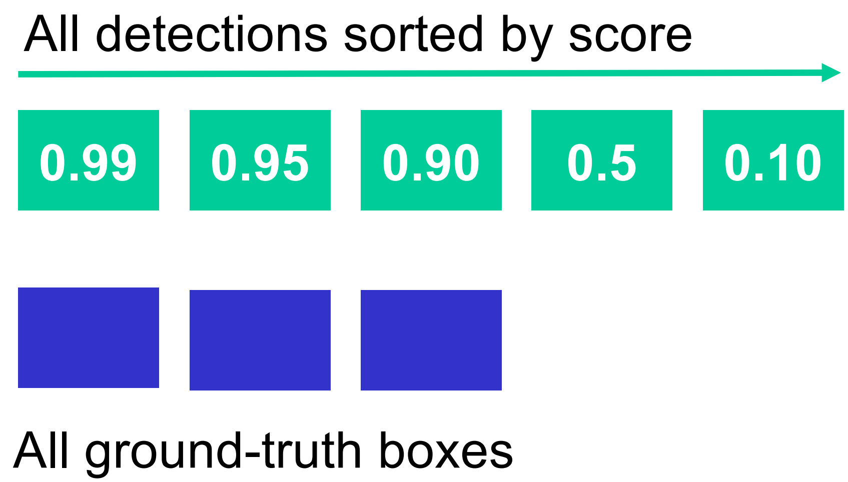
How would NMS do on an image like this?

- It will eliminate "good" boxes when objects are highly overlapping

Source: [J. Johnson]

Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**



Source: [J. Johnson]

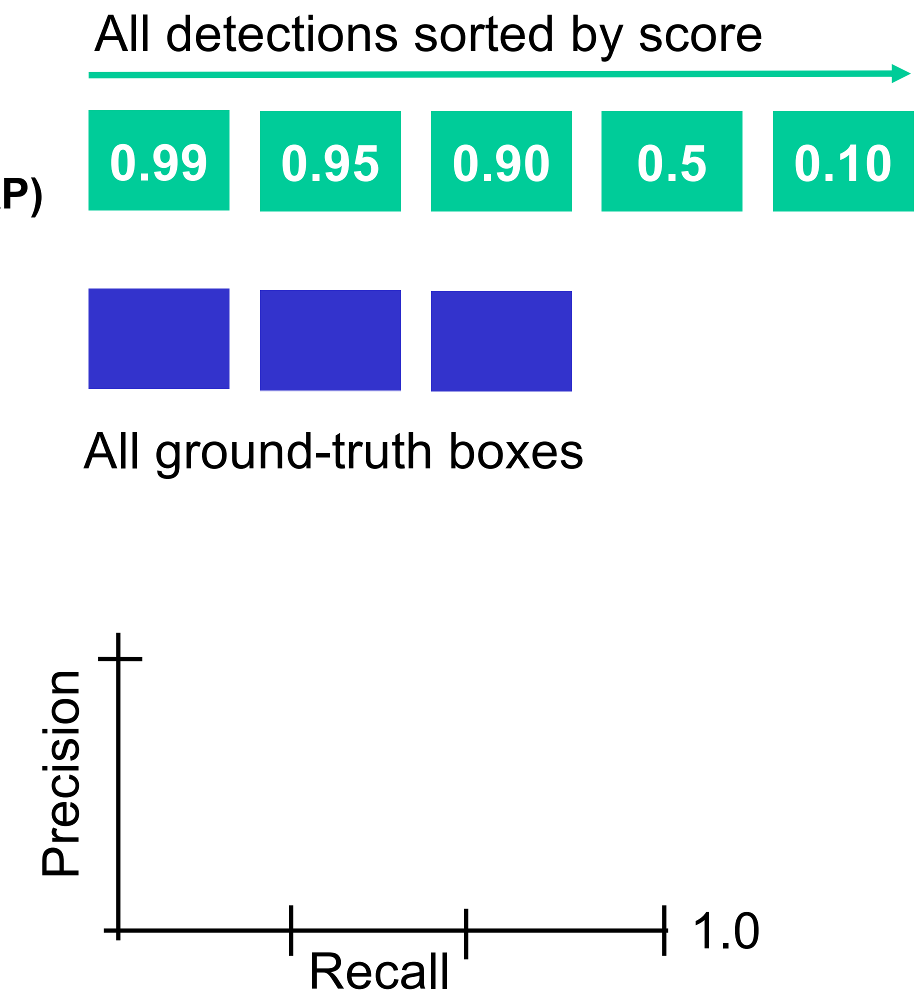
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

    1. For each detection (highest to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

$$Precision = \frac{true\ positive\ detections}{total\ detections\ so\ far}$$

$$Recall = \frac{true\ positive\ detections}{true\ positive\ test\ instances}$$

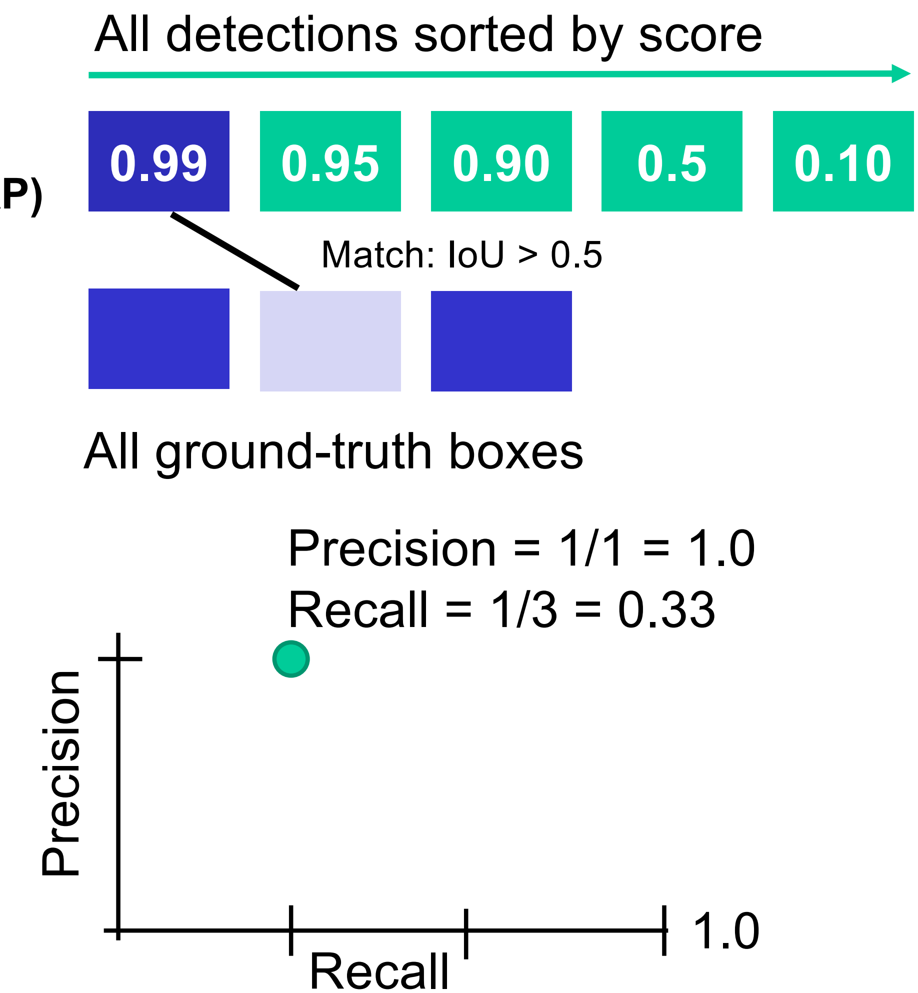


Source: J. Johnson

Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

    1. For each detection (highest to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

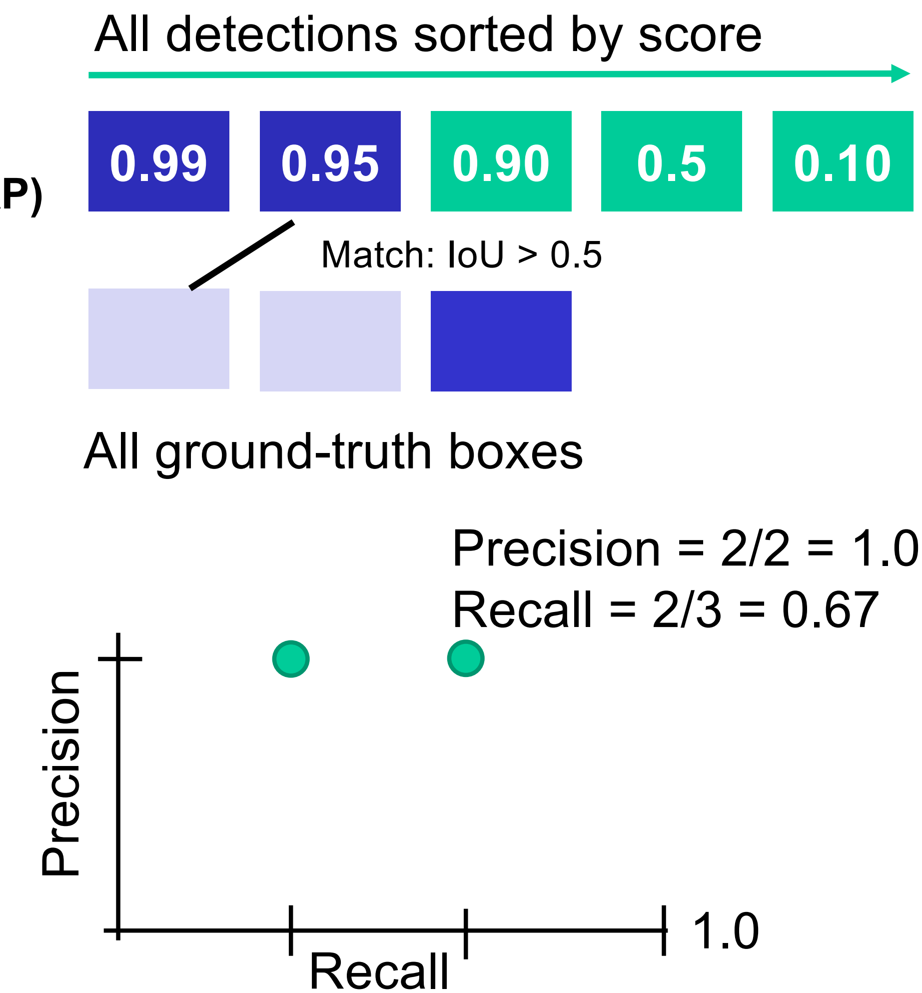


Source: [J. Johnson]

Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

    1. For each detection (highest to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

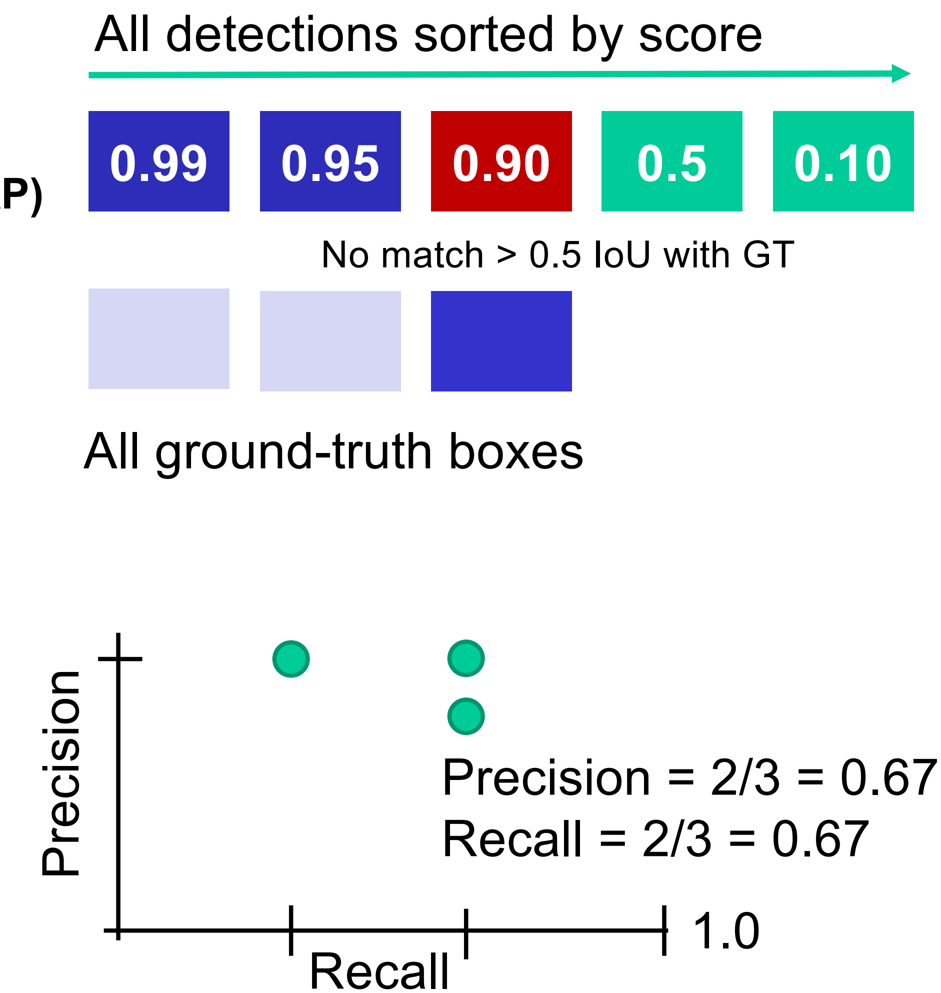


Source: [J. Johnson]

Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

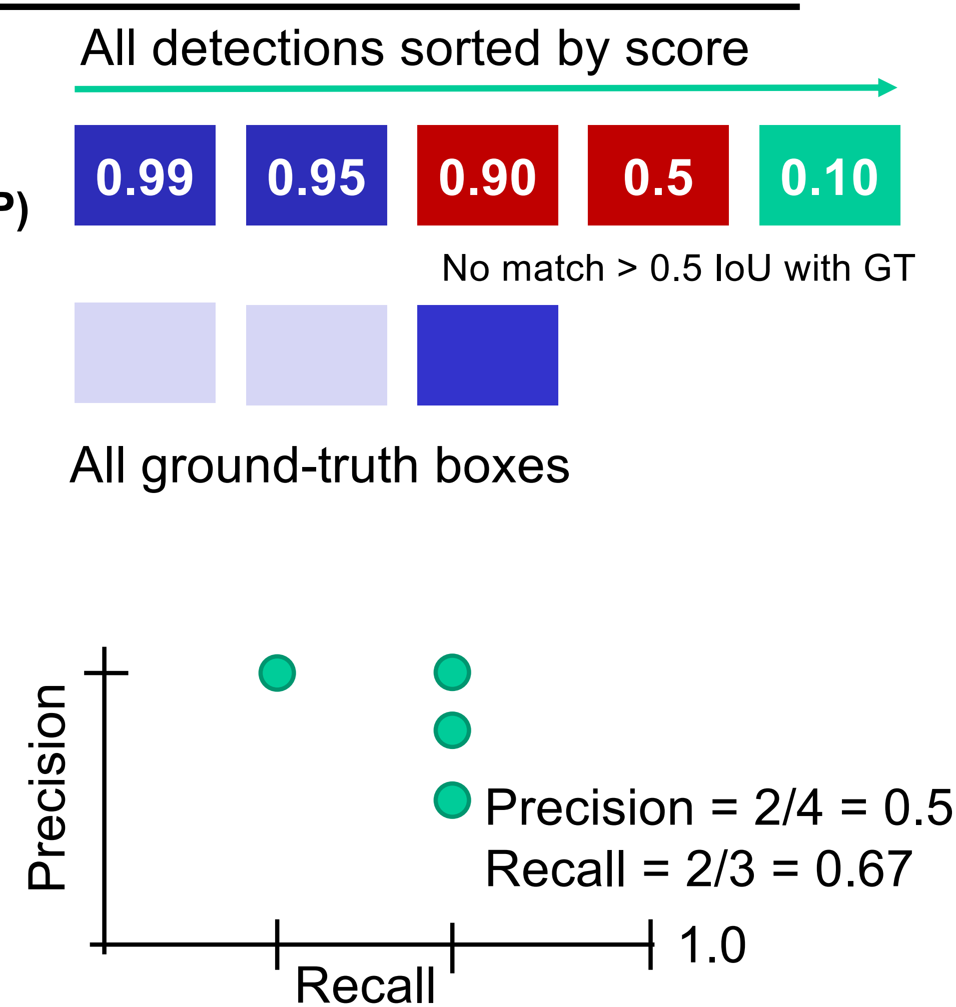
    1. For each detection (highest to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



Source: [J. Johnson]

Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**
    1. For each detection (highest to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

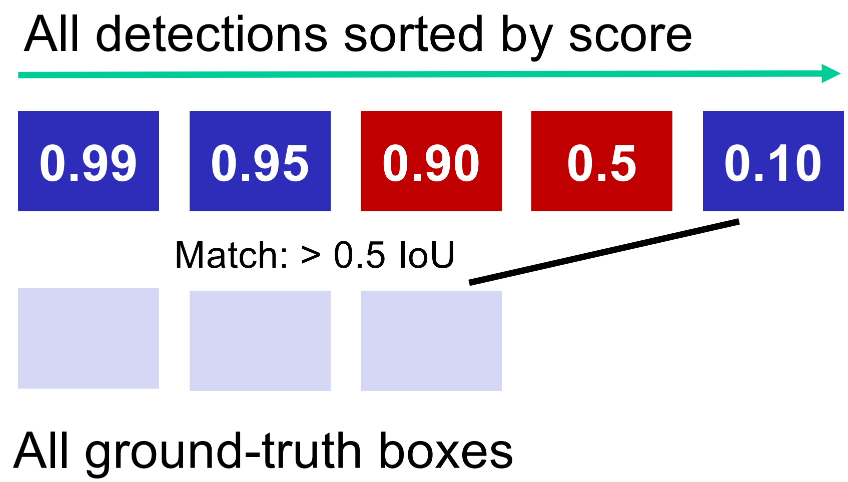


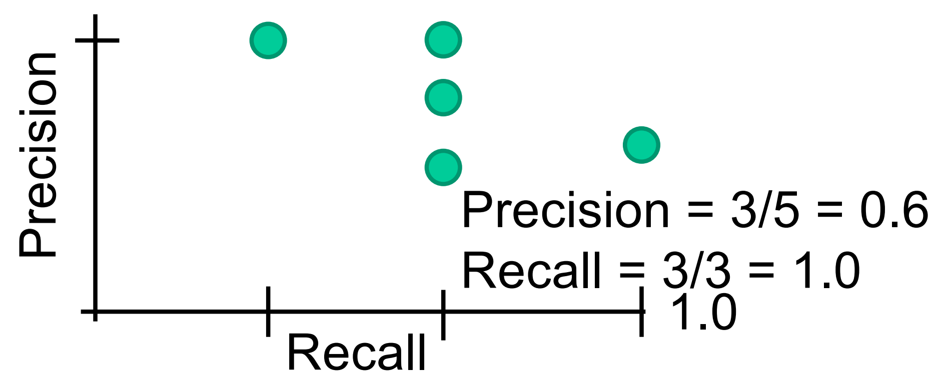
Source: [J. Johnson]()

Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

    1. For each detection (highest to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve





Source: [J. Johnson]

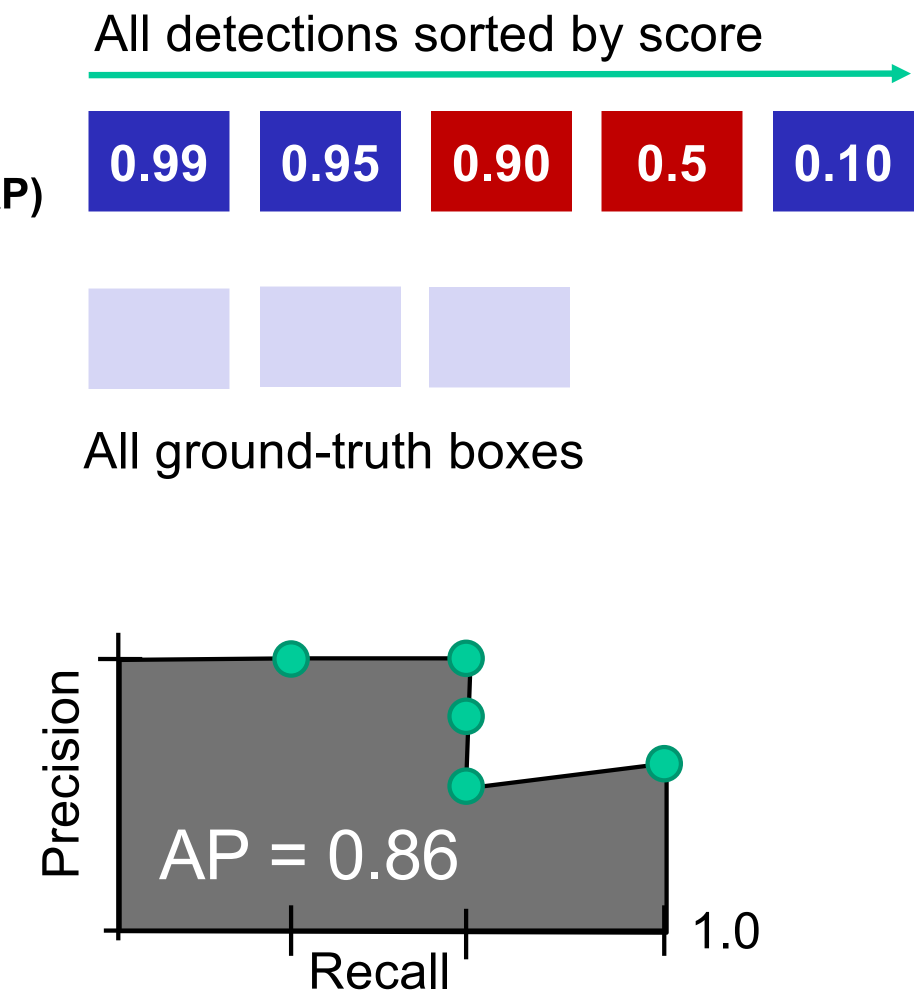
Object detection: Outline

- Task definition and evaluation
- Two-stage detectors:
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single-stage and multi-resolution detectors
- Other detectors: CornerNet, DETR

Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

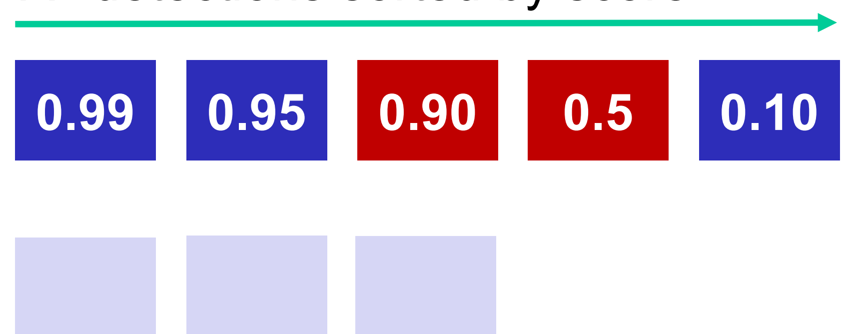
    1. For each detection (highest to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve



Source: [J. Johnson]

Evaluating object detectors

All detections sorted by score

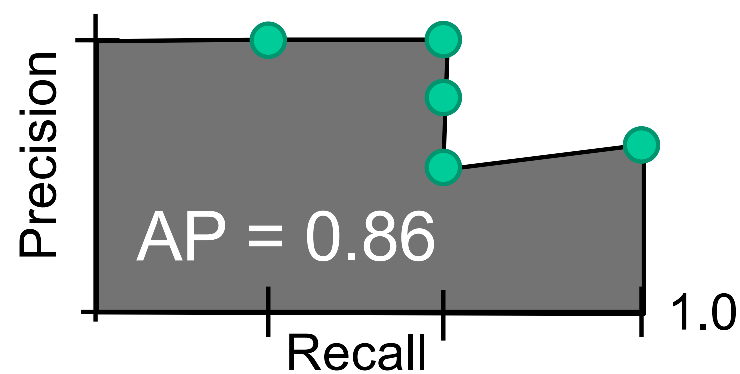


All ground-truth boxes

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

How to get AP = 1.0?

- Hit all GT boxes with IoU > 0.5, and have no “false positive” detections ranked above any “true positives”



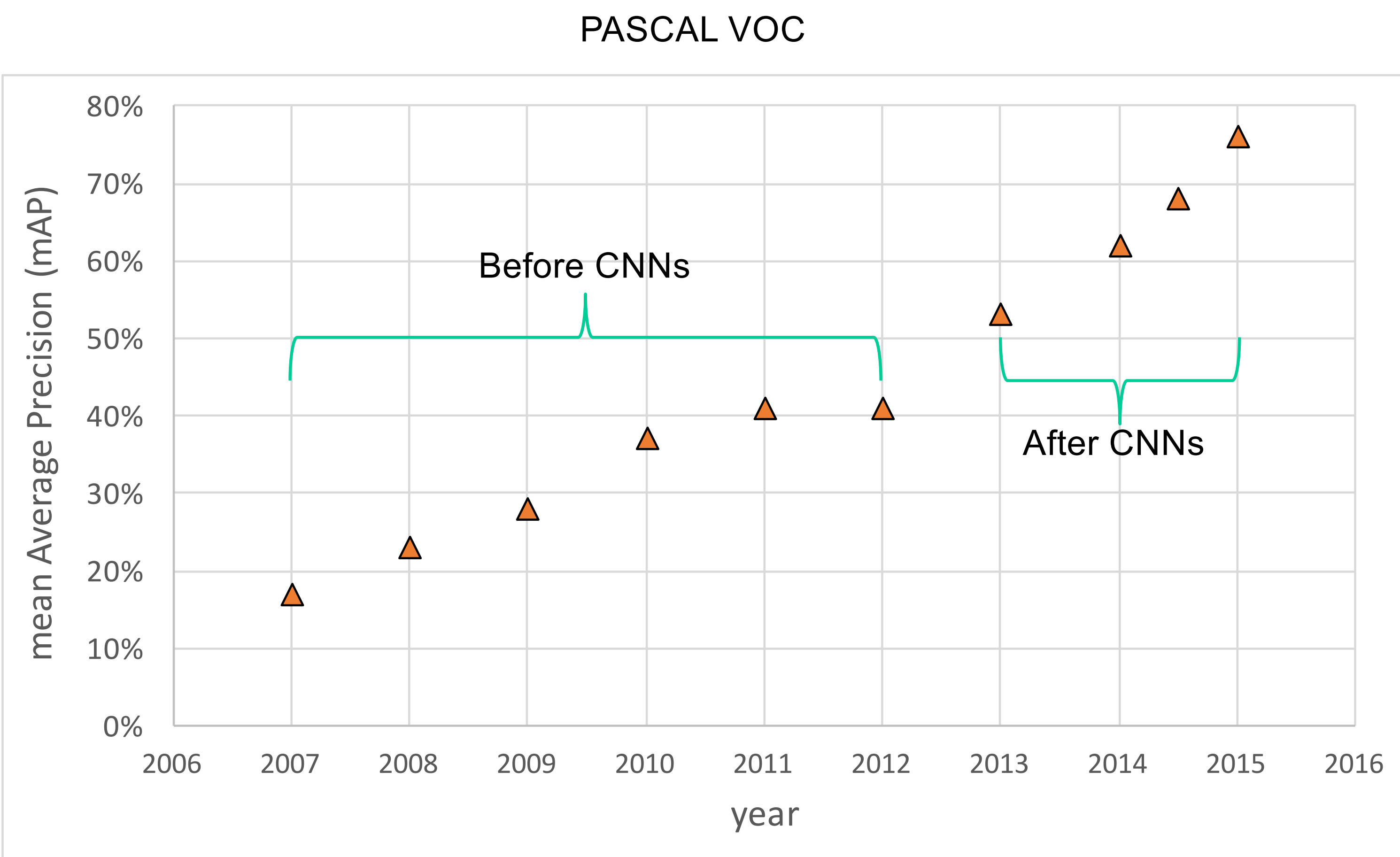
Source: [J. Johnson]

PASCAL VOC Challenge (2005-2012)

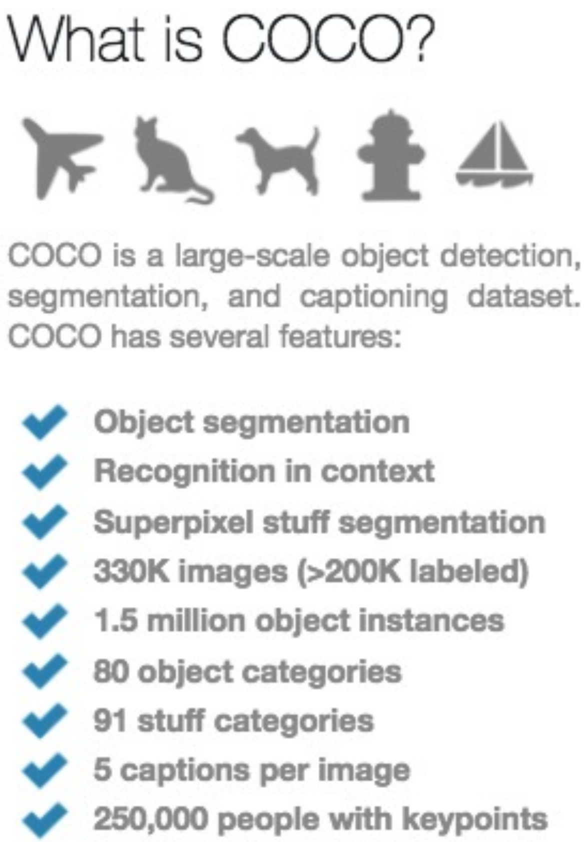
- 20 challenge classes:
  - *Person*
  - *Animals:* bird, cat, cow, dog, horse, sheep
  - *Vehicles:* airplane, bicycle, boat, bus, car, motorbike, train
  - *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor
- Dataset size (by 2012): 11.5K training/validation images, 27K bounding boxes, 7K segmentations

http://host.robots.ox.ac.uk/pascal/VOC/

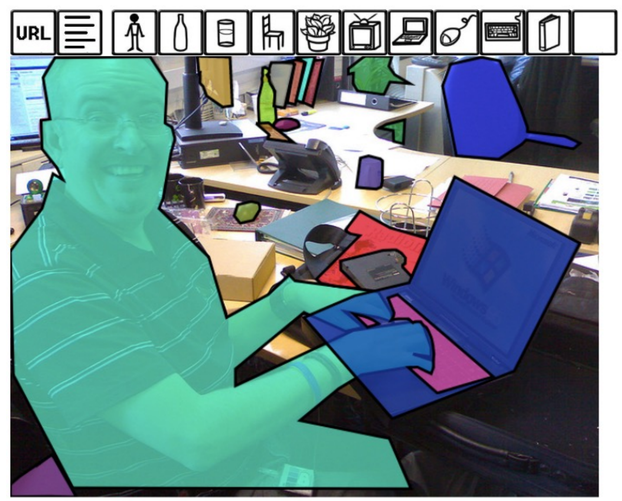
Progress on PASCAL detection



More recent benchmark: COCO



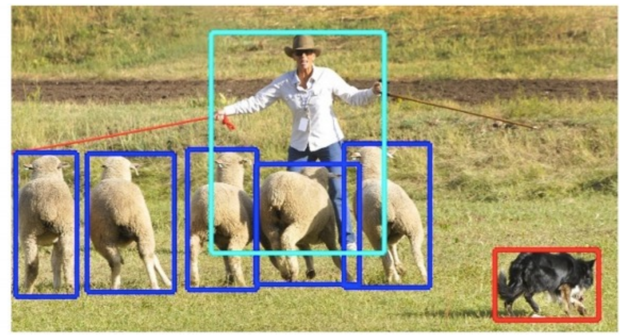
COCO
Common Objects in Context



http://cocodataset.org/#home

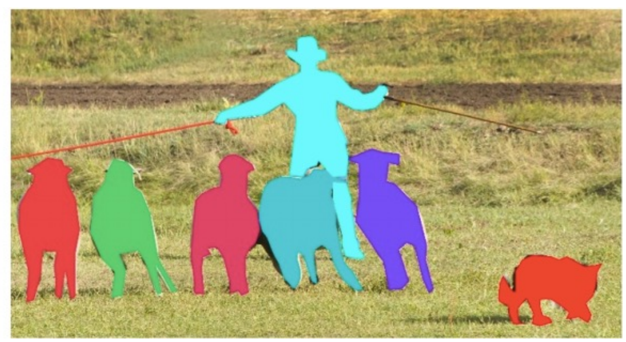
COCO dataset: Tasks

image classification



object detection

semantic segmentation



instance segmentation

- Also: keypoint prediction, captioning, question answering…

COCO detection metrics

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

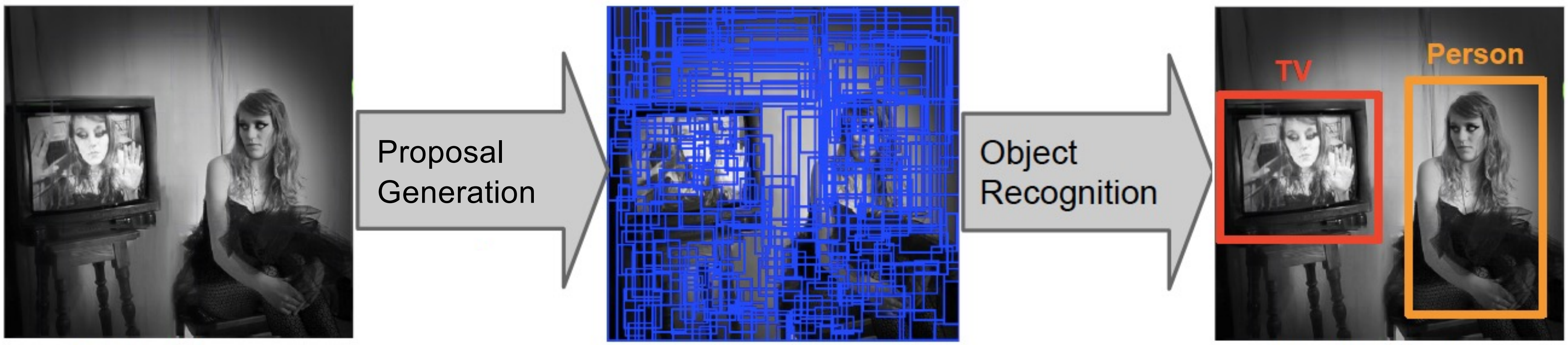
AR Across Scales:

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

- Leaderboard: http://cocodataset.org/#detection-leaderboard
  - Not updated since 2020

Object detection: Outline

- Task definition and evaluation
- Two-stage detectors

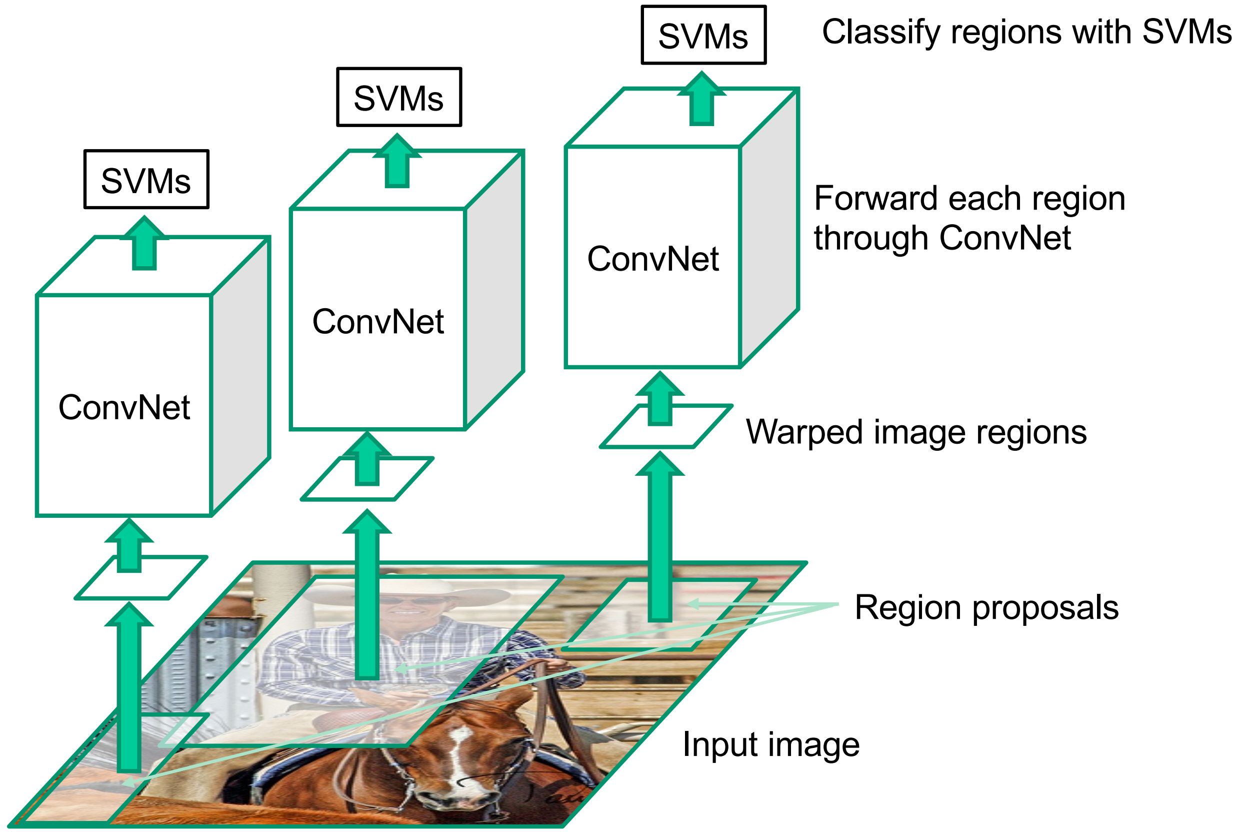


Original Image
Region Proposals
Final Detections

Image source

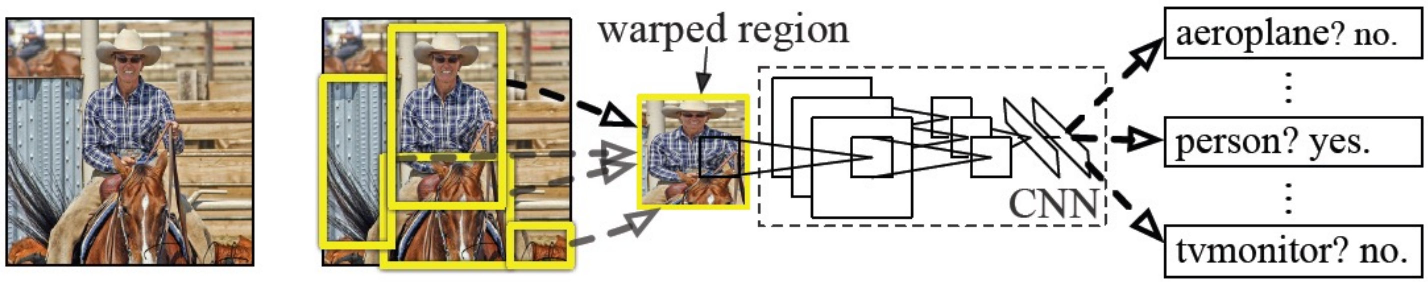
R-CNN: Region proposals + CNN features

Source: R. Girshick



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

R-CNN details

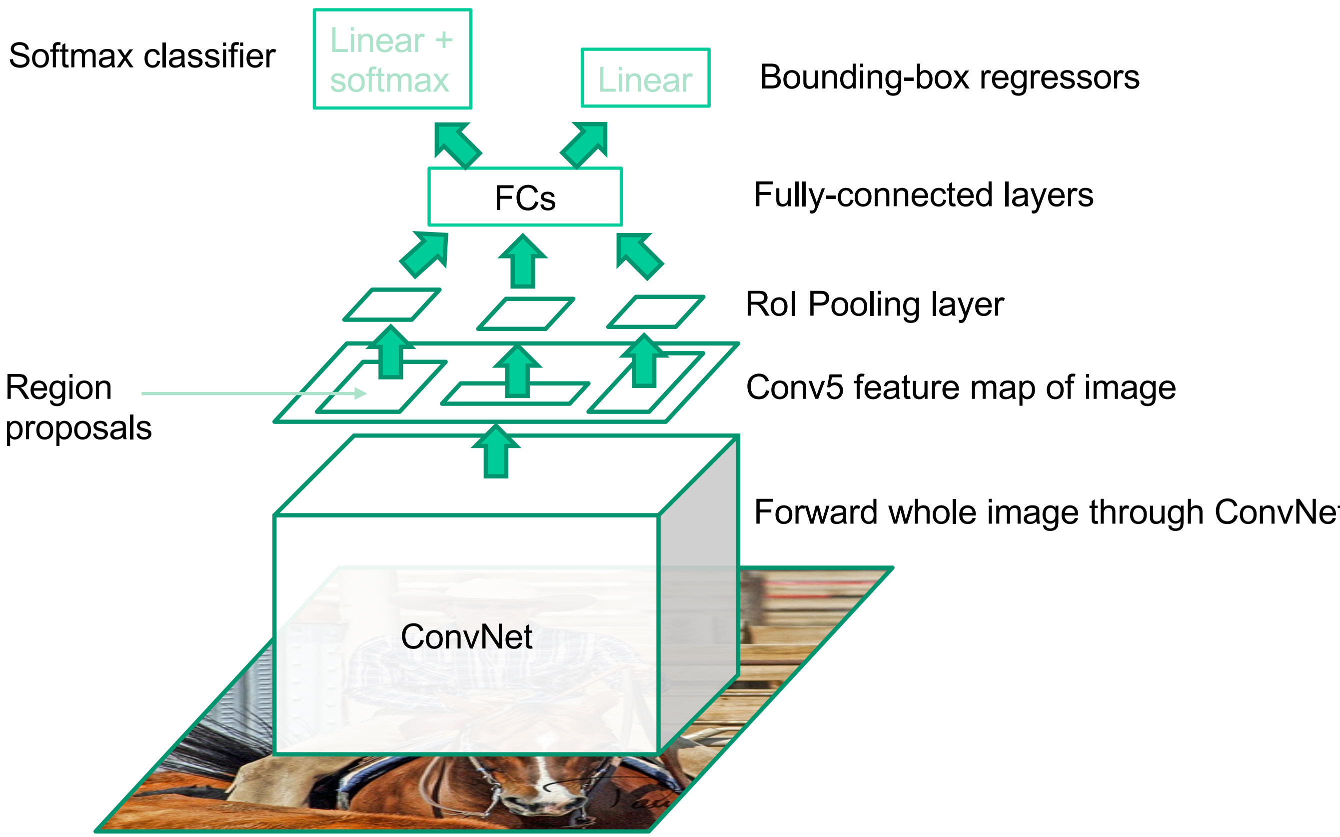


- **Regions**: ~2000 Selective Search proposals
- **Network**: AlexNet *pre-trained* on ImageNet (1000 classes), *fine-tuned* on PASCAL (21 classes)
- **Final detector**: warp proposal regions, extract fc7 network activations (4096 dimensions), classify with linear SVM
- **Bounding box regression** to refine box locations
- **Performance:** mAP of **53.7%** on PASCAL 2010
(vs. **35.1%** for Selective Search and **33.4%** for Deformable Part Models)

R-CNN pros and cons

- Pros
  - Much more accurate than previous approaches!
  - Any deep architecture can immediately be “plugged in”
- Cons
  - Not a single end-to-end system
    - Fine-tune network with softmax classifier (log loss)
    - Train post-hoc linear SVMs (hinge loss)
    - Train post-hoc bounding-box regressions (least squares)
  - Training was slow (84h), took up a lot of storage
    - 2000 CNN passes per image
  - Inference (detection) was slow (47s / image with VGG16)

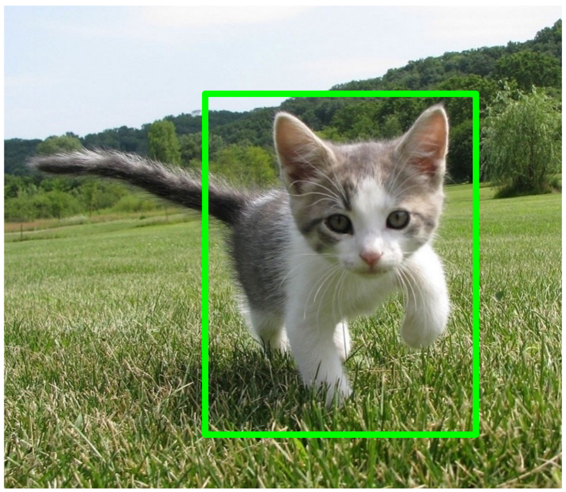
Fast R-CNN



Source: R. Girshick

R. Girshick, [Fast R-CNN](), ICCV 2015

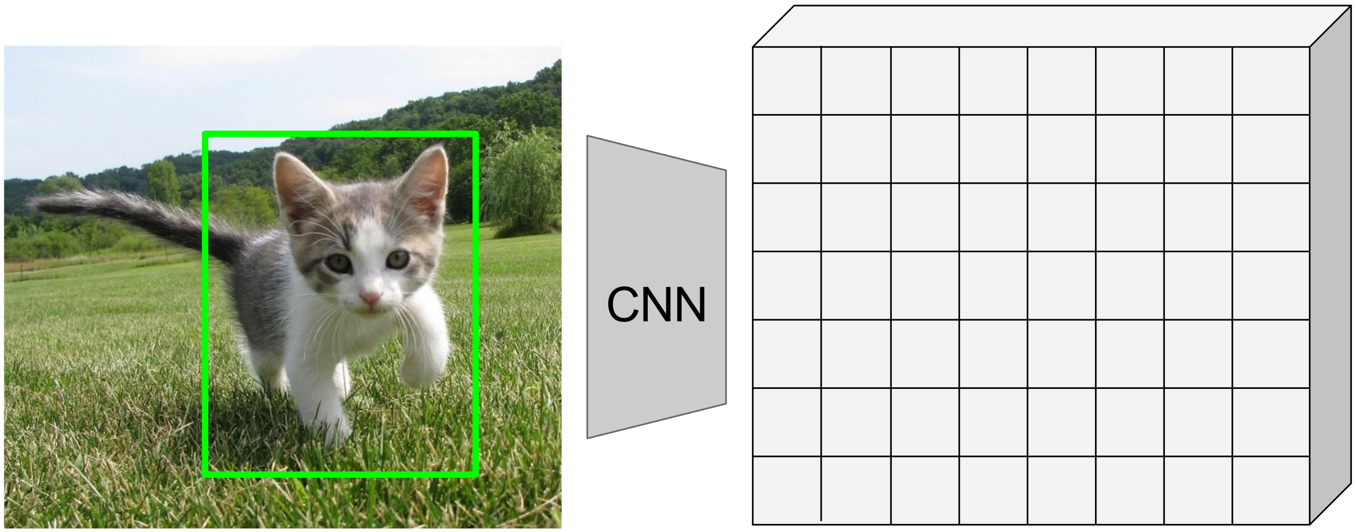
RoI pooling



Input Image
(e.g., 3 x 640 x 480)

Source: J. Johnson

RoI pooling

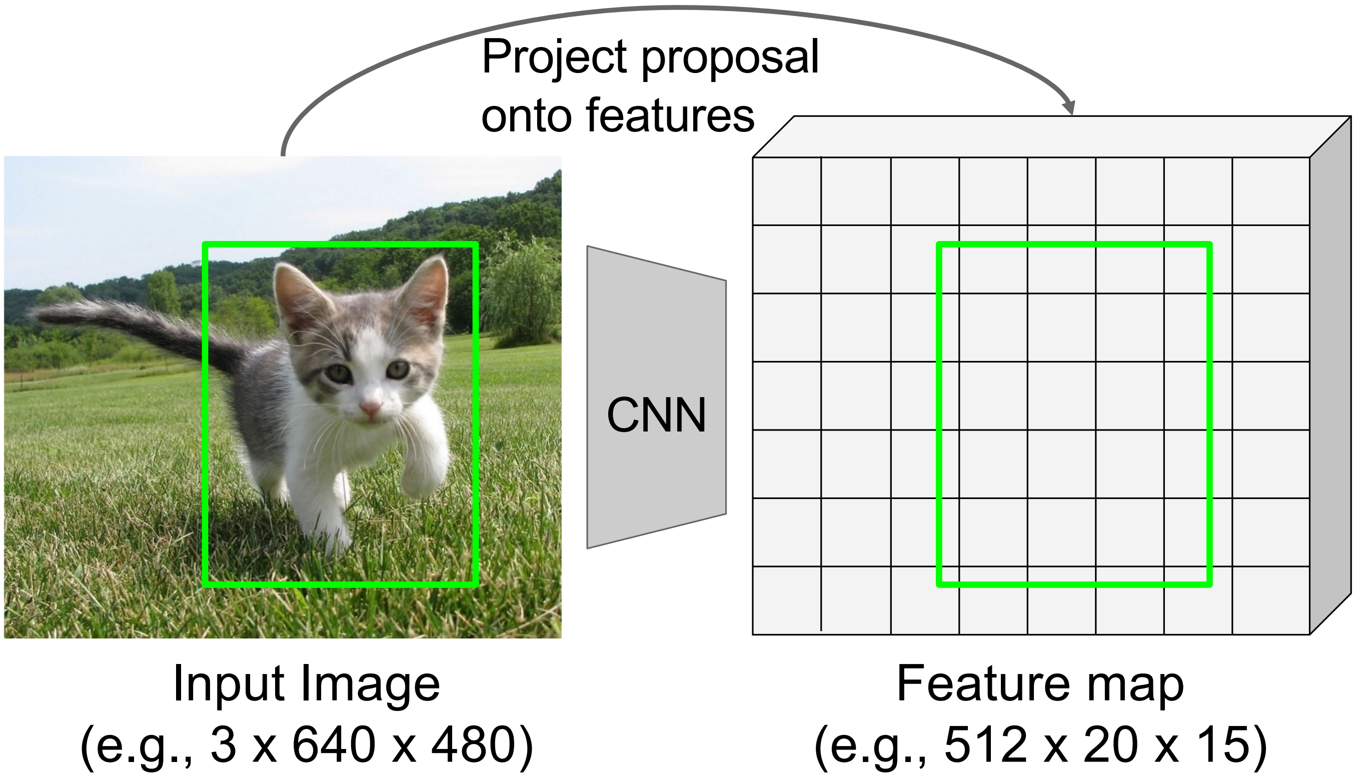


Input Image
(e.g., 3 x 640 x 480)

Feature map
(e.g., 512 x 20 x 15)

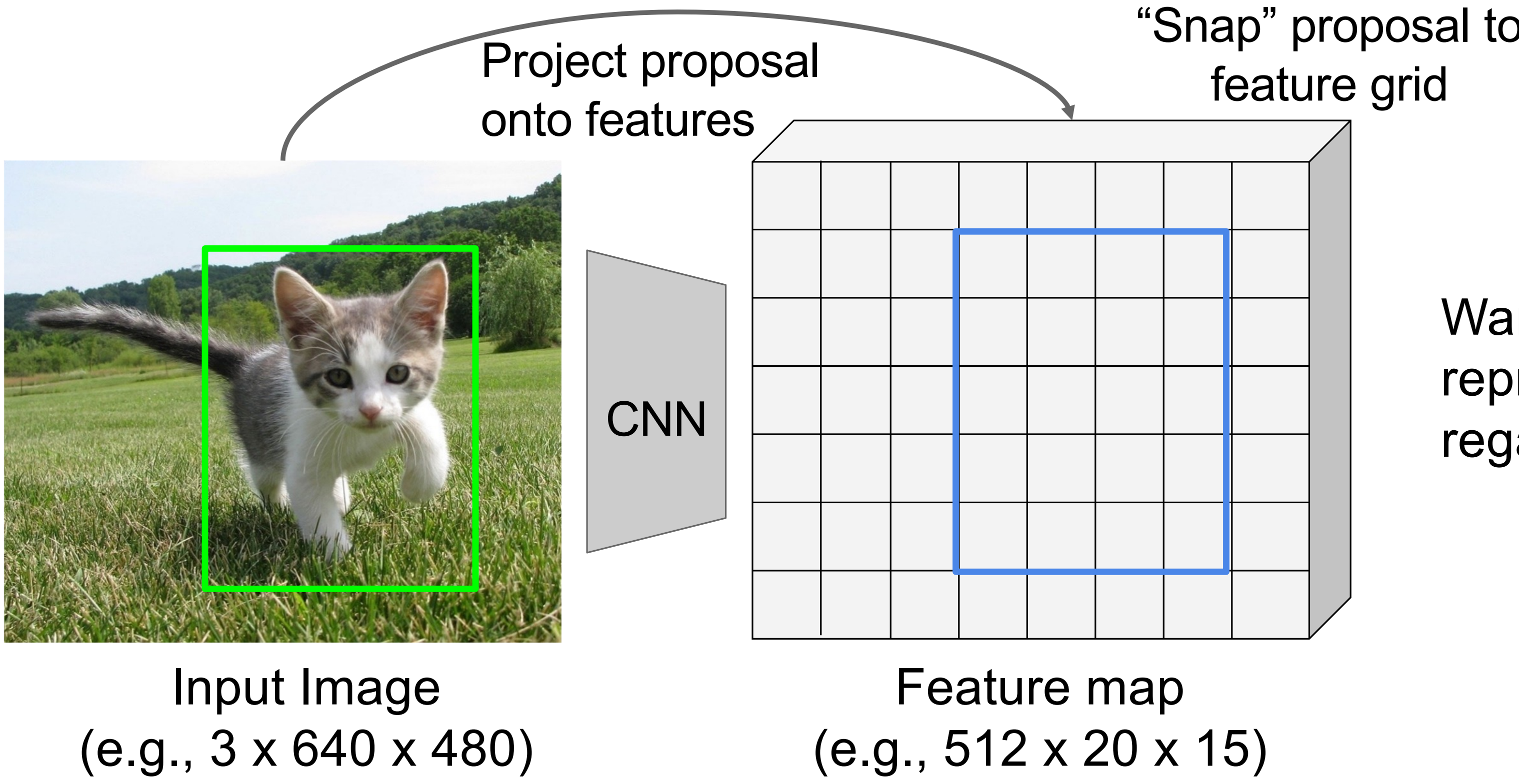
Source: [J. Johnson]

RoI pooling



Source: [J. Johnson]

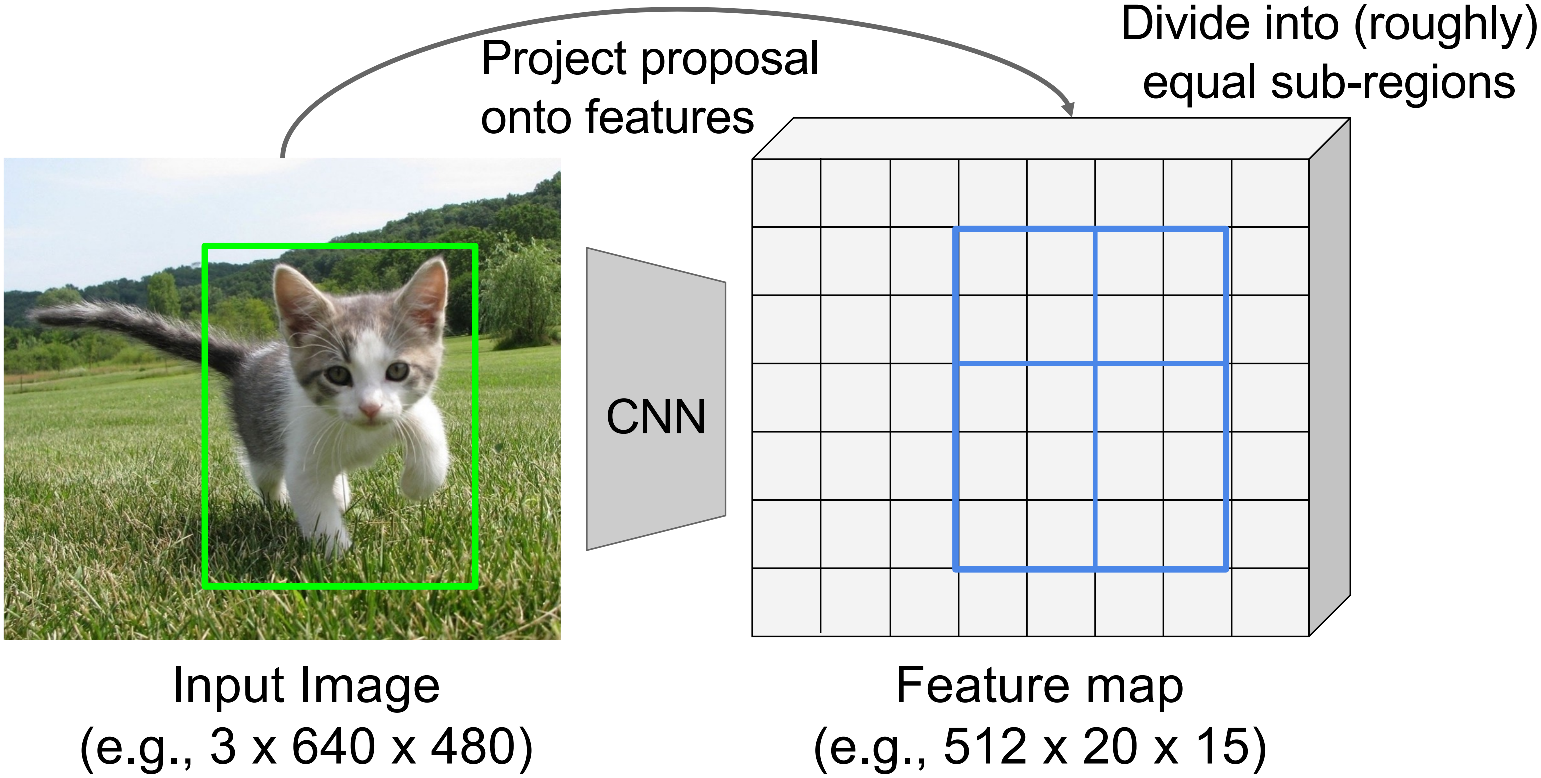
RoI pooling



Want fixed-size RoI representation regardless of its size

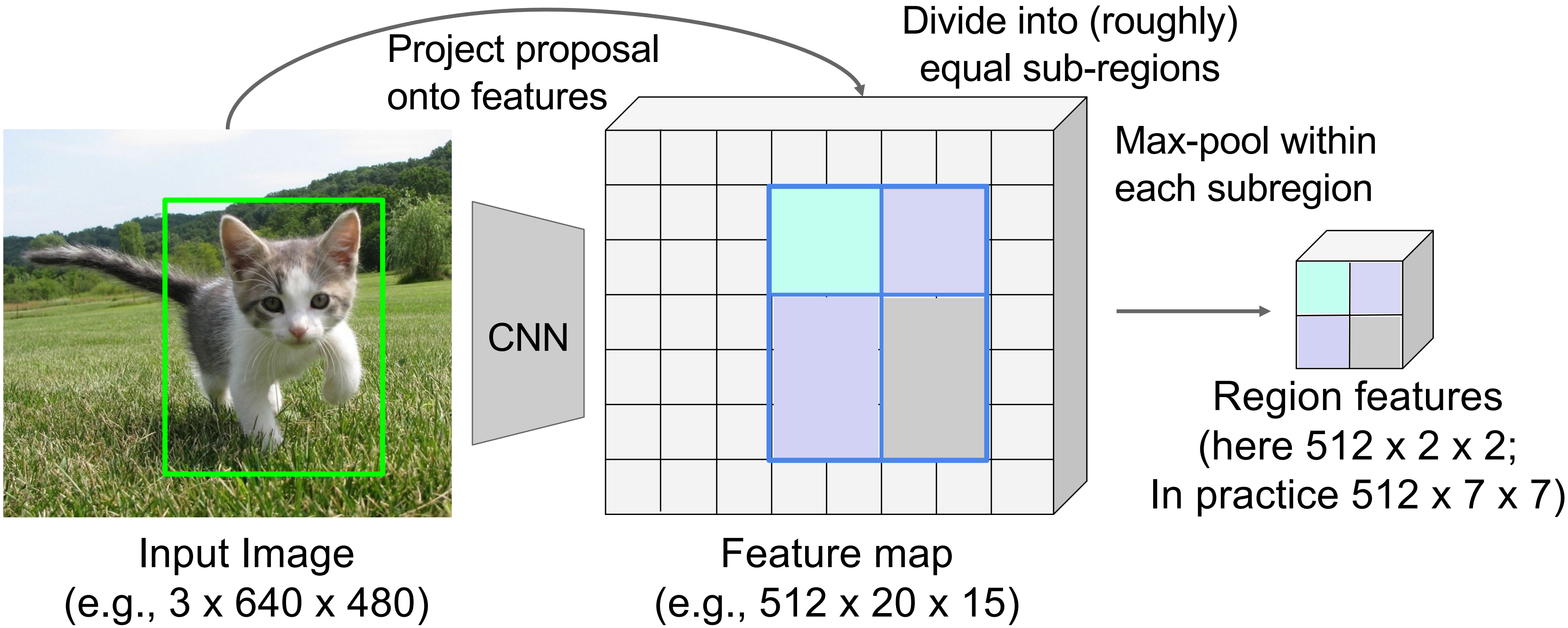
Source: J. Johnson

RoI pooling



Source: [J. Johnson]

RoI pooling



Source: J. Johnson

RoI pooling illustration

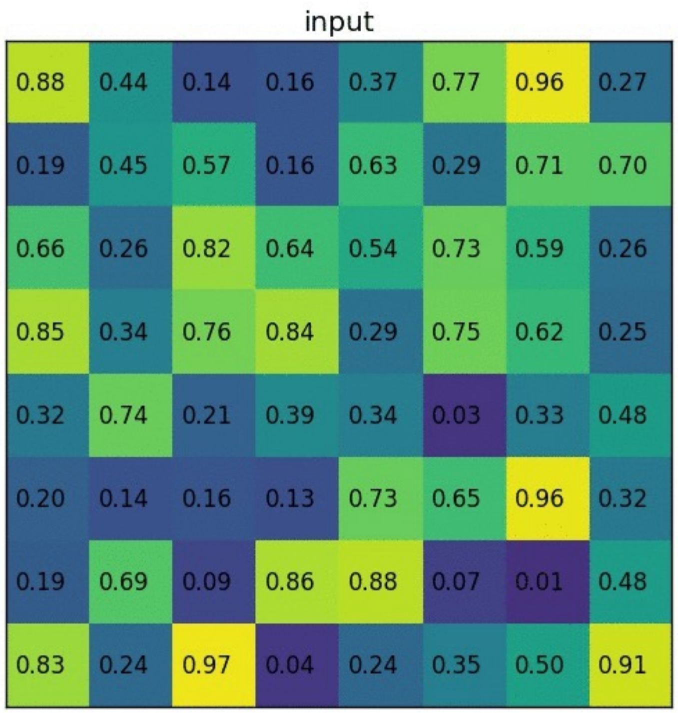
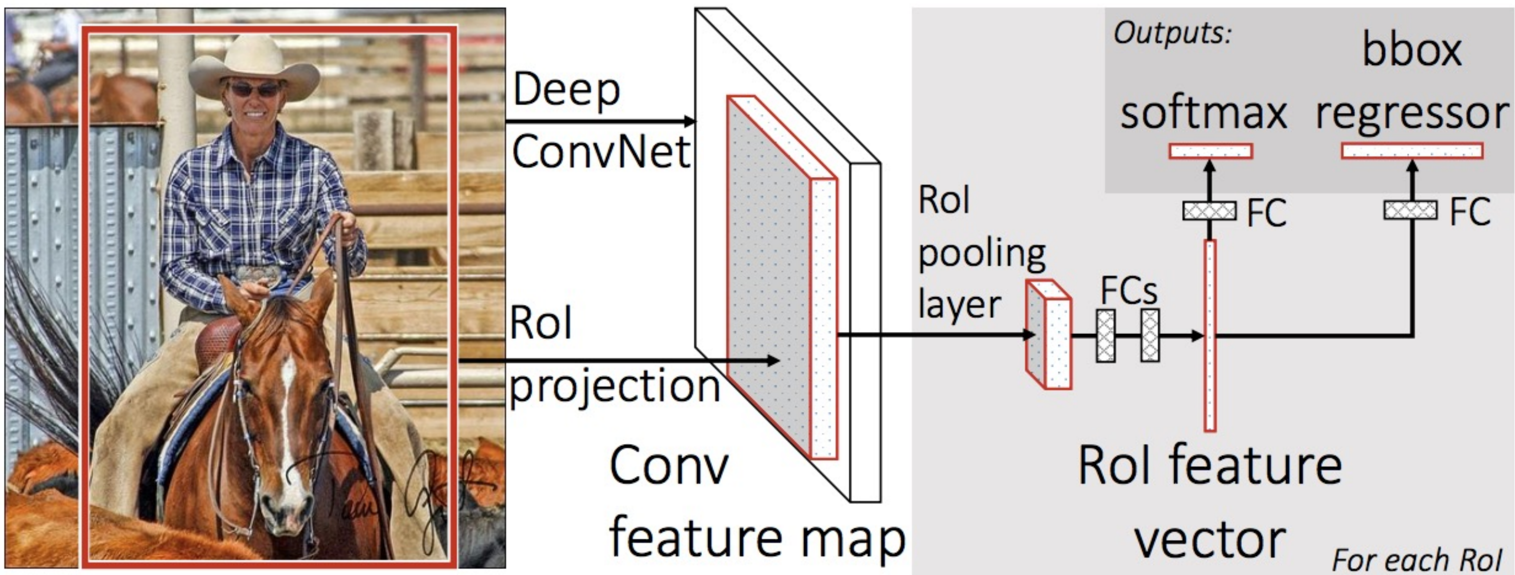


Image source

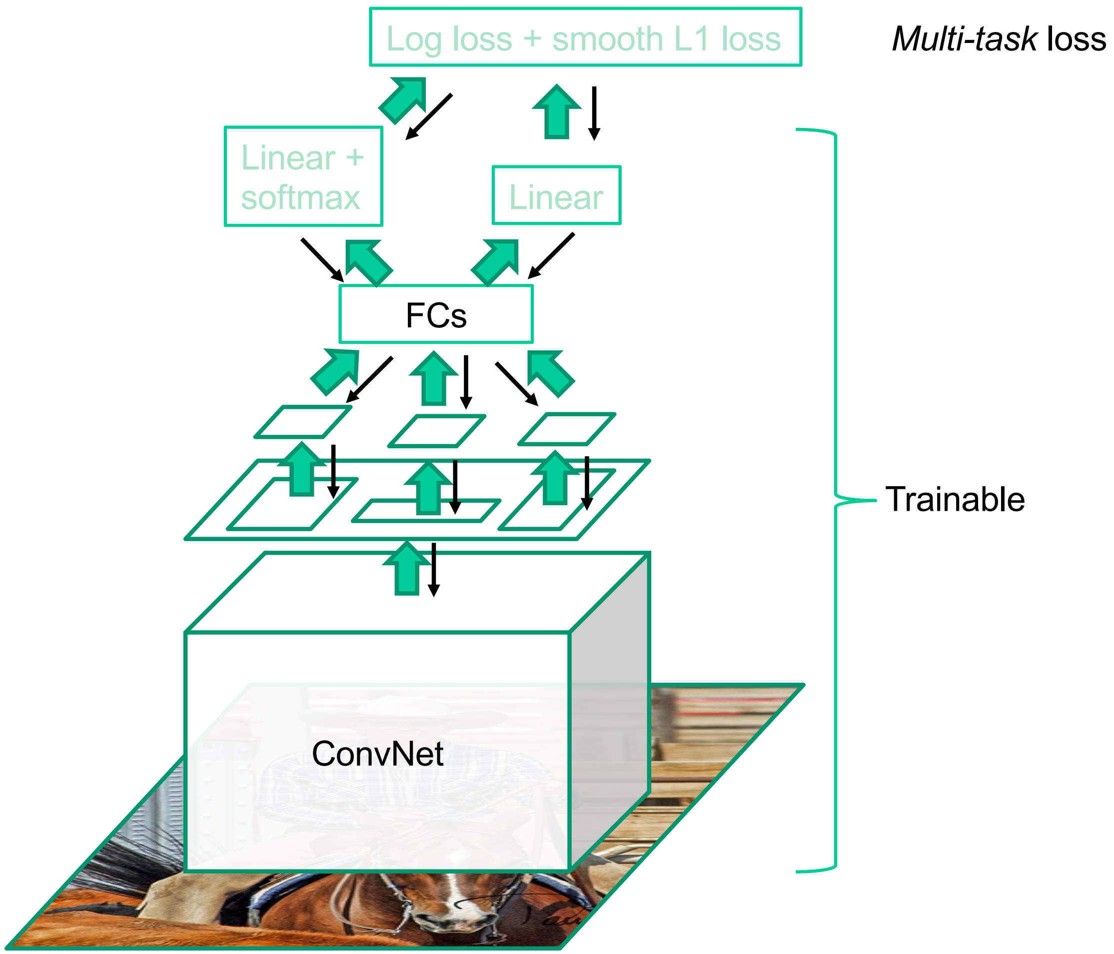
Prediction

- For each RoI, network predicts probabilities for $C + 1$ classes (class 0 is background) and four bounding box offsets for C classes



R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN training



Source: R. Girshick

R. Girshick, [Fast R-CNN](), ICCV 2015

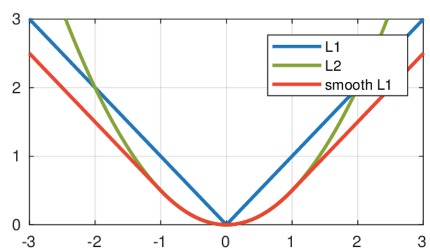
Multi-task loss

- Loss for ground truth class y, predicted class probabilities $P(y)$, ground truth box b, and predicted box $\hat{b}$:

$$L(y, P, b, \hat{b}) = \underbrace{-\log P(y)}_{\text{softmax loss}} + \lambda \mathbb{I}[y \geq 1] \underbrace{L_{\text{reg}}(b, \hat{b})}_{\text{regression loss}}$$

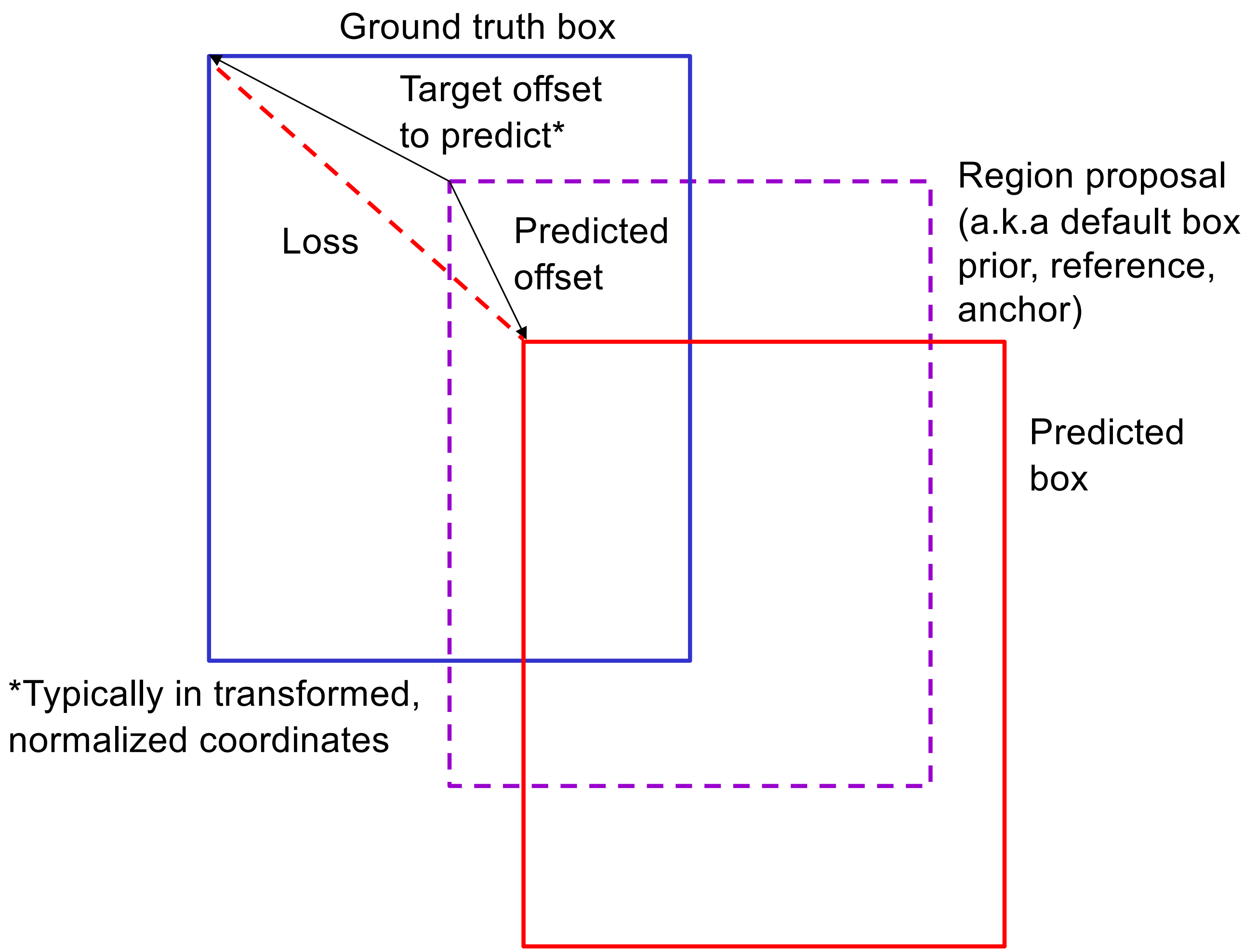
- Regression loss: *smooth* L_1 *loss* on top of log space offsets relative to proposal

$$L_{\text{reg}}(b, \hat{b}) = \sum_{i=\{x,y,w,h\}} \text{smooth}_{L_1}(b_i - \hat{b}_i)$$



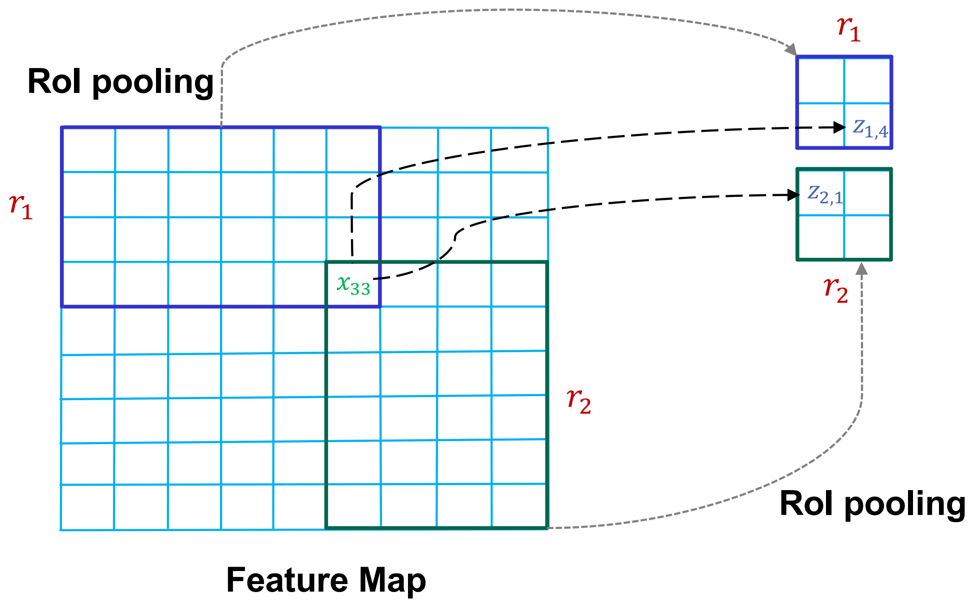
$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Bounding box regression



ROI pooling: Backpropagation

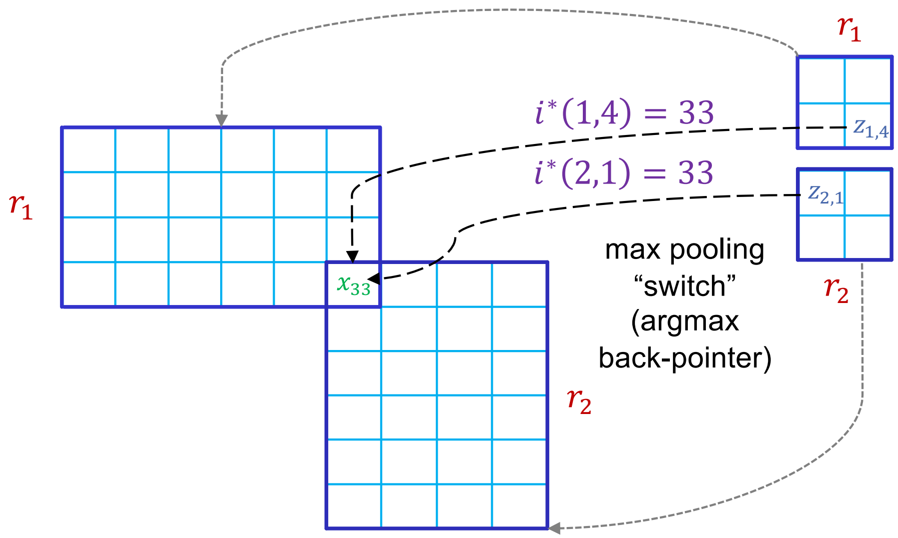
- Similar to max pooling, but has to take into account overlap of pooling regions



Source: Ross Girshick

ROI pooling: Backpropagation

- Similar to max pooling, but has to take into account overlap of pooling regions



Backward Pass:
Have $\frac{\partial e}{\partial z}$,
want $\frac{\partial e}{\partial x}$

$$\frac{\partial e}{\partial x_i} = \sum_r \sum_j \frac{\partial e}{\partial z_{rj}} \frac{\partial z_{rj}}{\partial x_i} = \sum_r \sum_j \mathbb{I}[i = i^*(r,j)] \frac{\partial e}{\partial z_{rj}}$$

Over regions r,
RoI indices j

1 if r, j “pooled”
input i; 0 o/w

Source: Ross Girshick

Fast R-CNN results

	Fast R-CNN	**R-CNN**
Train time (h)	**9.5**	84
- Speedup	**8.8x**	
Test time / image	**0.32s**	47.0s
- Test speedup	**146x**	
mAP	**66.9%**	66.0%

(vs. 53.7% for AlexNet)

Timings exclude object proposal time, which is equal for all methods.
All methods use VGG16.

Source: R. Girshick, K. He

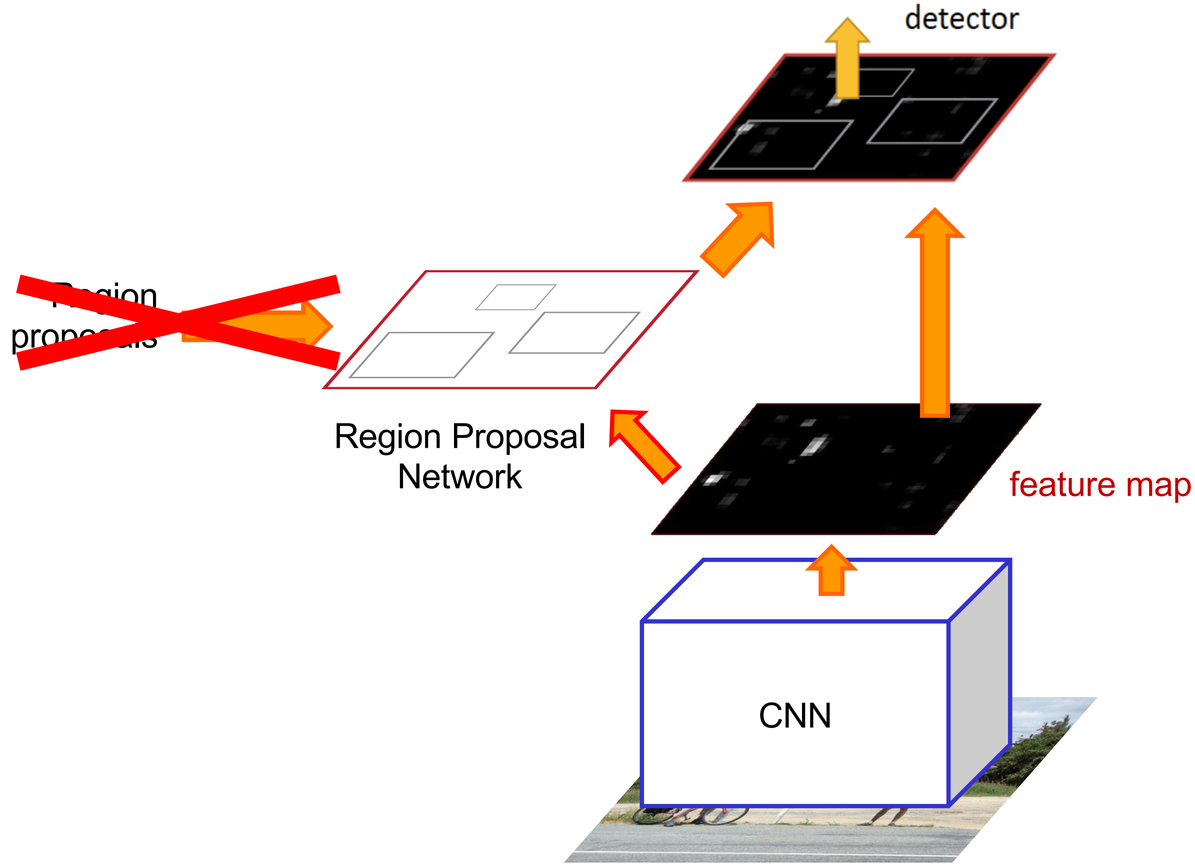
Announcements and reminders

- **Quiz 2** will be out 9AM this Friday, March 22, through 9AM **next Tuesday, March 26**
- **MP3** is out, due **Wednesday, April 3**
- **Project progress updates** will be due **Friday, April 12**

Object detection: Outline

- Task definition and evaluation
- Two-stage detectors:
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single-stage and multi-resolution detectors
- Other detectors: CornerNet, DETR

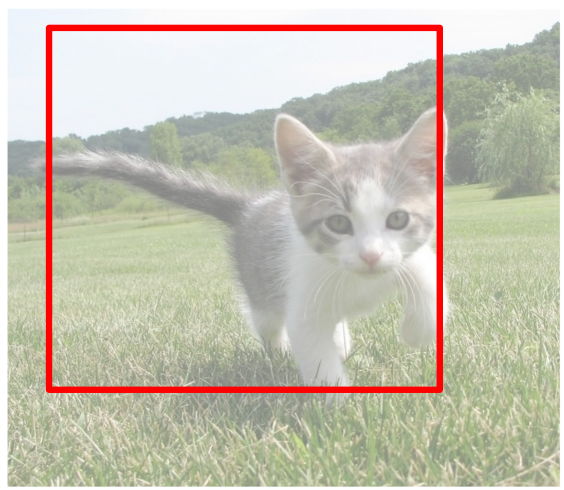
Faster R-CNN



S. Ren, K. He, R. Girshick, and J. Sun, [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks], NIPS 2015

Region proposal network (RPN)

- Idea: tile the image with "anchor boxes" of a set size and try to predict how likely each anchor is to contain an object

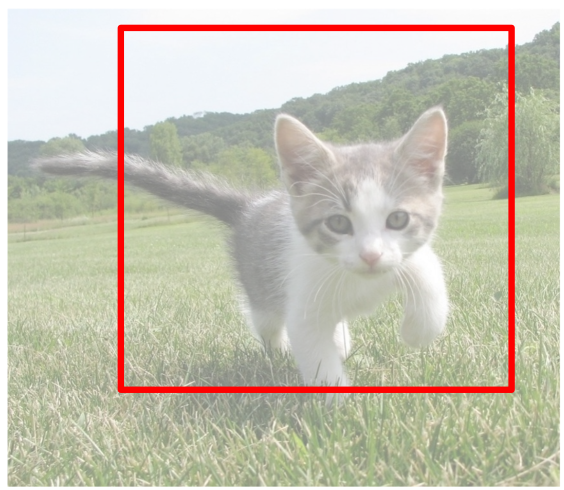


Anchor is an object?

Figure source: [J. Johnson]

Region proposal network (RPN)

- Idea: tile the image with “anchor boxes” of a set size and try to predict how likely each anchor is to contain an object

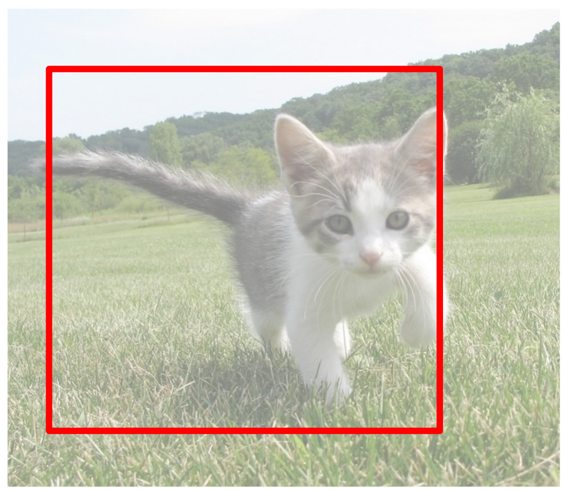


Anchor is an object?

Figure source: [J. Johnson]

Region proposal network (RPN)

- Idea: tile the image with “anchor boxes” of a set size and try to predict how likely each anchor is to contain an object

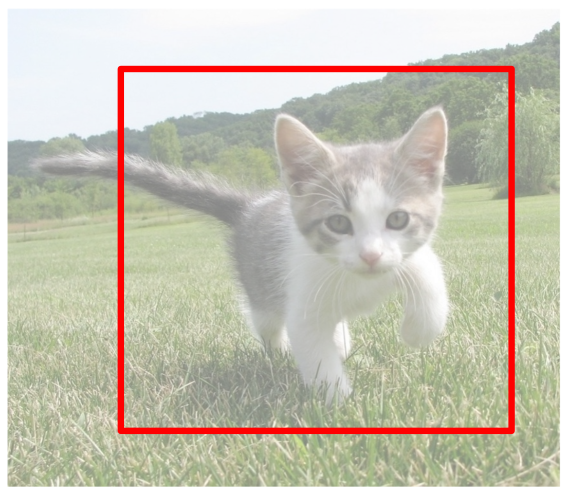


Anchor is an object?

Figure source: J. Johnson

Region proposal network (RPN)

- Idea: tile the image with "anchor boxes" of a set size and try to predict how likely each anchor is to contain an object

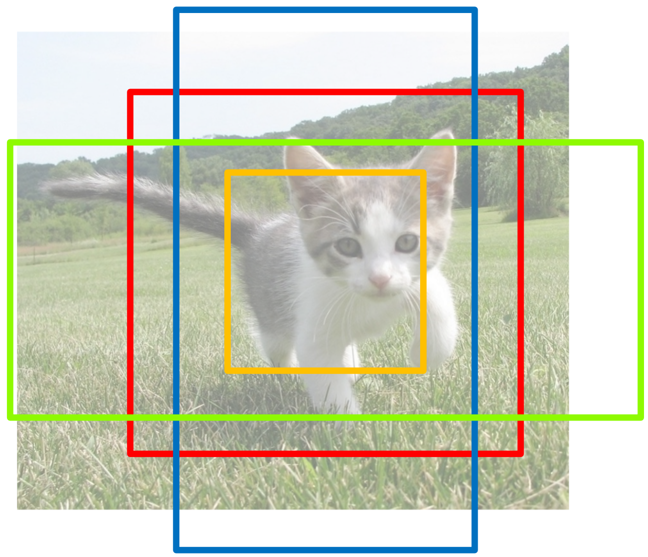


Anchor is an object?

Figure source: [J. Johnson]

Region proposal network (RPN)

- Idea: tile the image with "anchor boxes" of a set size and try to predict how likely each anchor is to contain an object
- Introduce anchor boxes at multiple scales and aspect ratios to handle a wider range of object sizes and shapes

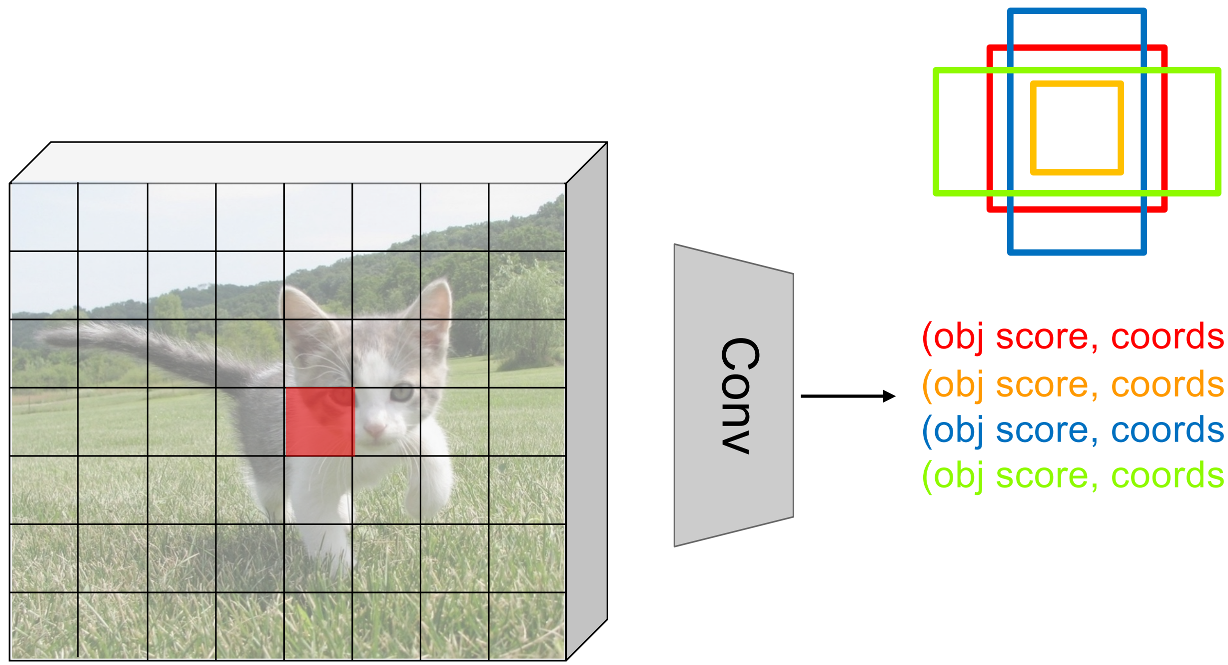


Anchor is an object?
Anchor is an object?
Anchor is an object?
Anchor is an object?

Figure source: [J. Johnson]

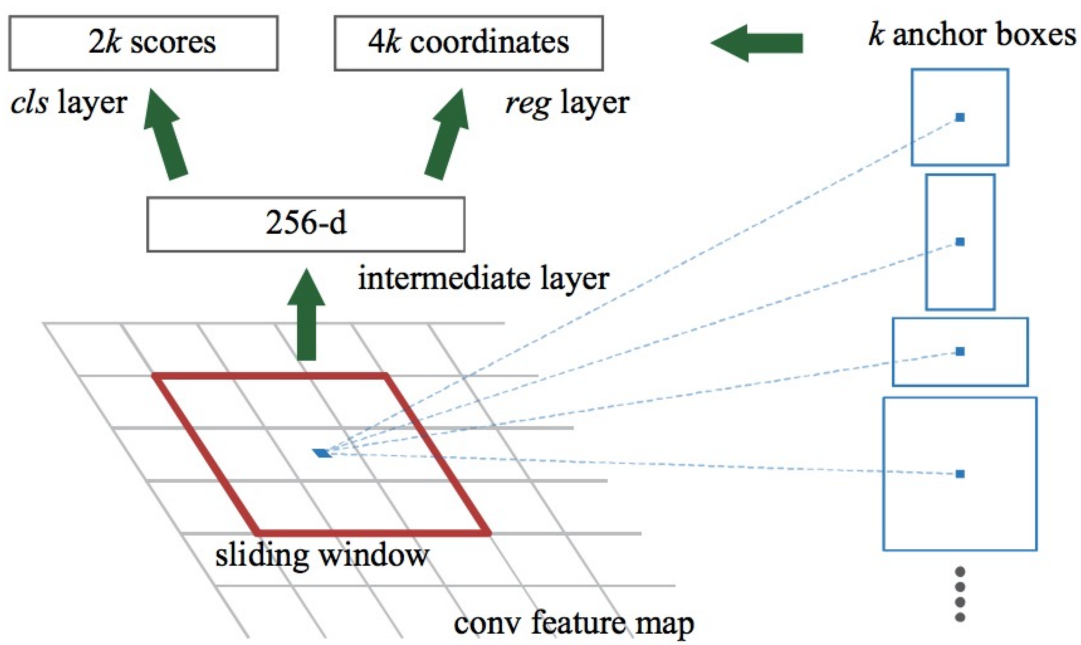
Region proposal network (RPN)

- Implementation: put conv layers over low-resolution feature grid, for each grid location predict "object/no object" scores and bounding box regression coordinates

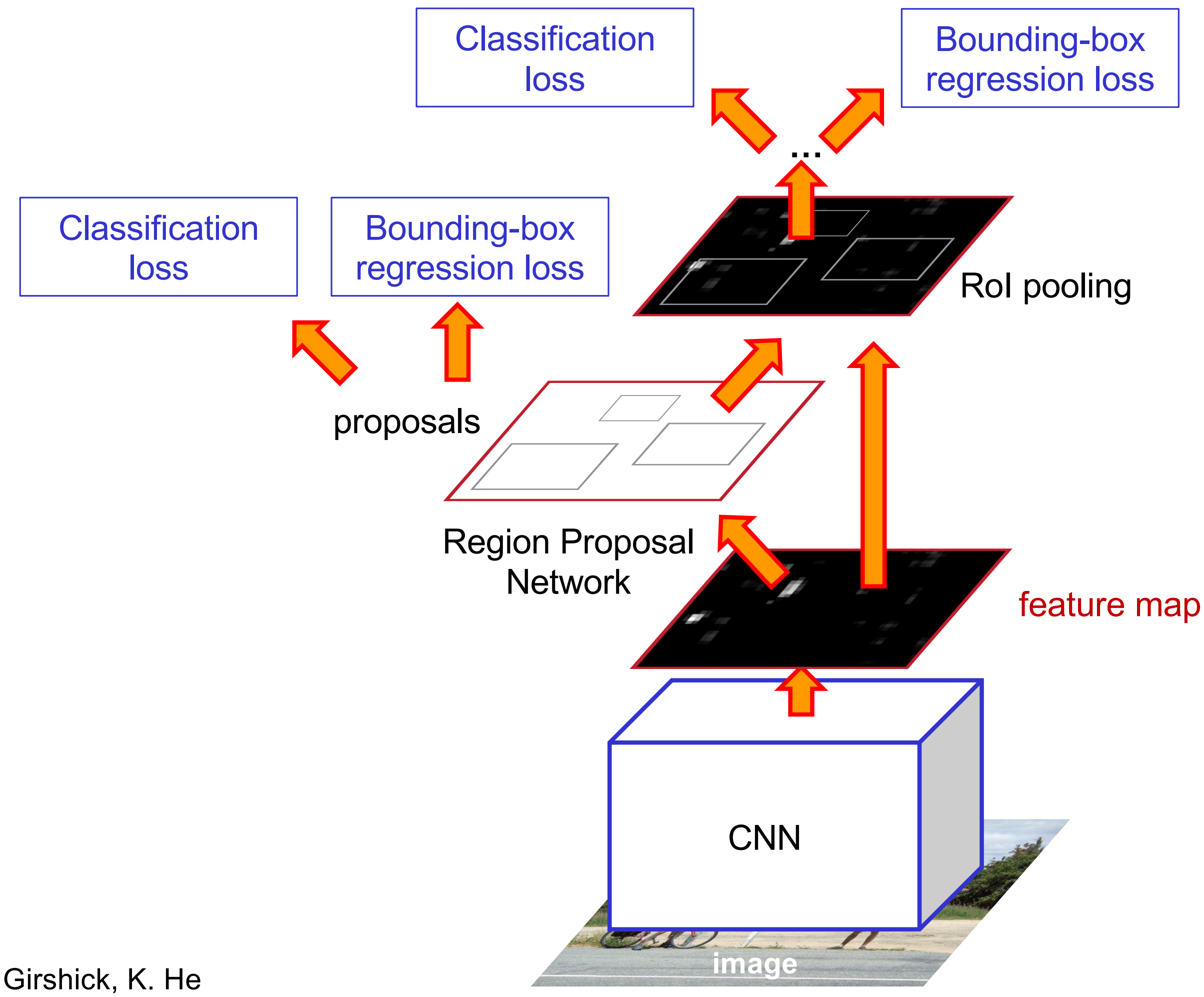


Faster R-CNN RPN design

- Slide a small window (3x3) over the conv5 layer
  - Predict object/no object
  - Regress bounding box coordinates with reference to *anchors* (3 scales x 3 aspect ratios)



One network, four losses



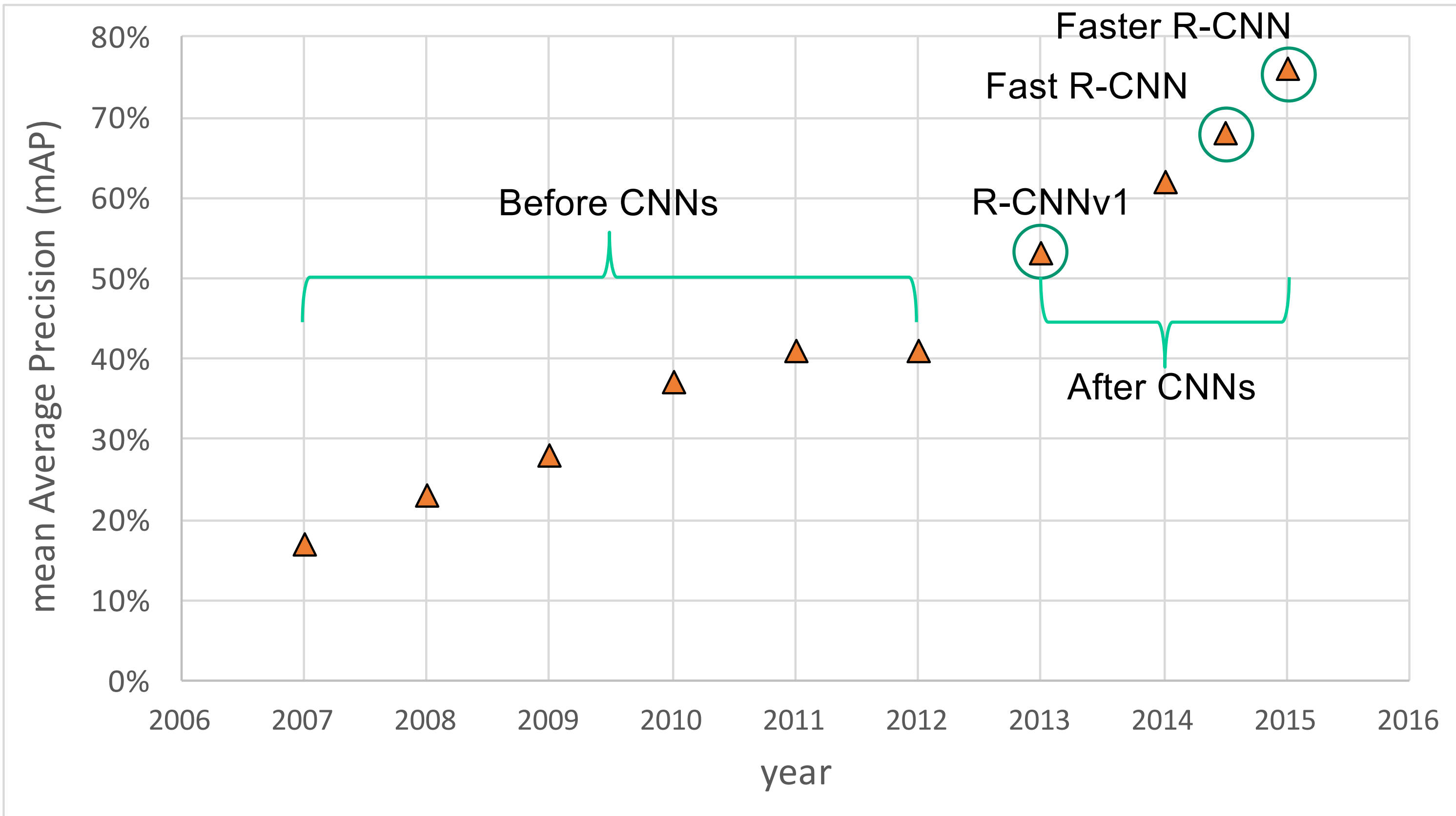
Source: R. Girshick, K. He

Faster R-CNN results

system	time	07 data	07+12 data
R-CNN	~50s	66.0	-
Fast R-CNN	~2s	66.9	70.0
Faster R-CNN	198ms	**69.9**	**73.2**

detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet

Object detection progress

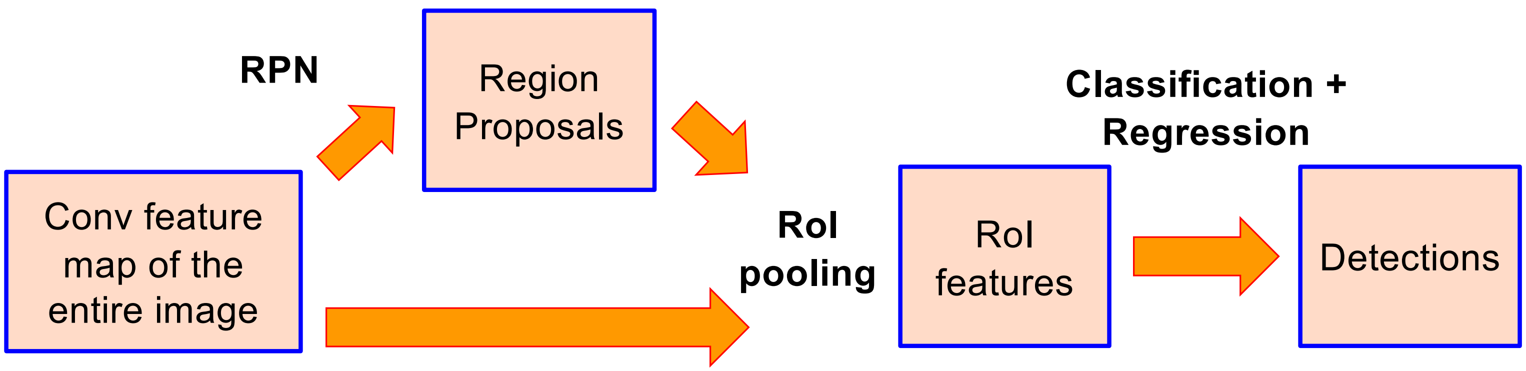


Outline

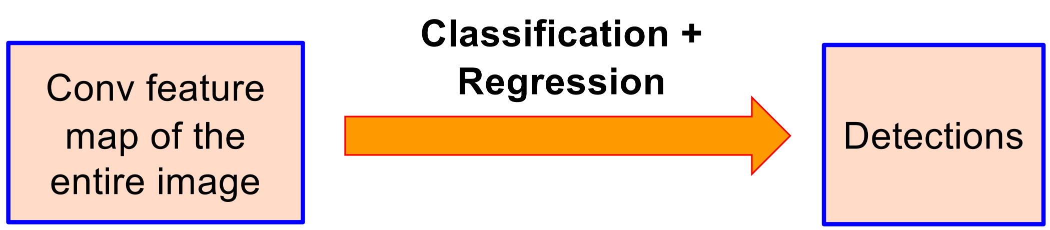
- Task definition and evaluation
- Two-stage detectors
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single-stage and multi-resolution detectors

Streamlined detection architectures

- The Faster R-CNN pipeline separates proposal generation and region classification

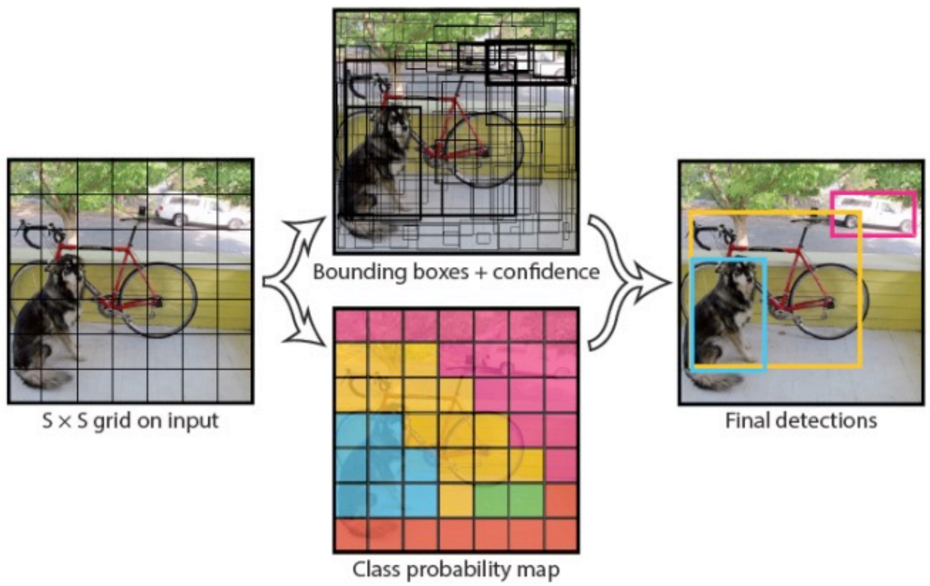


- Is it possible to do detection in one shot?



YOLO

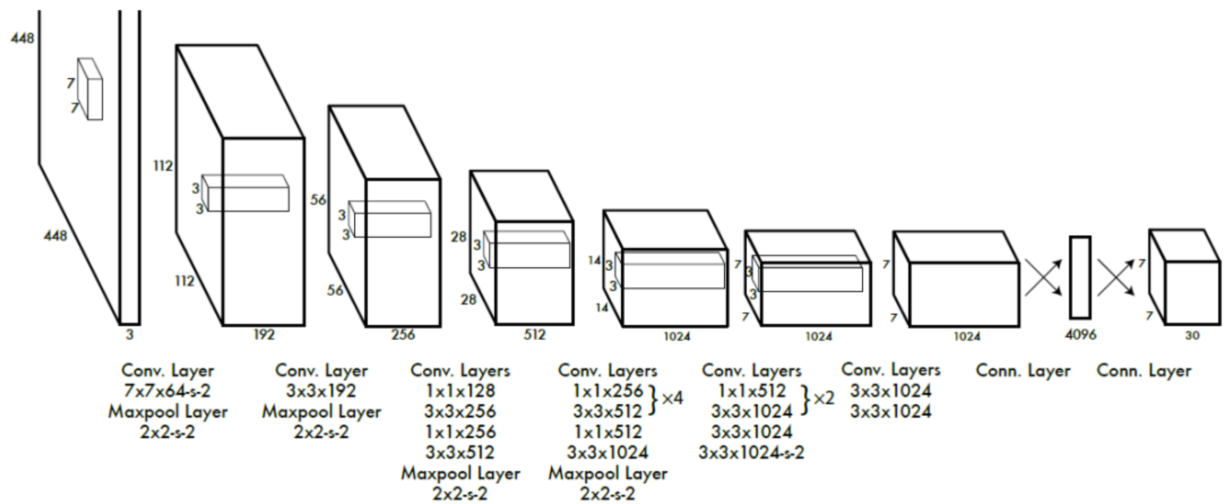
- Divide the image into a coarse grid and directly predict class label and a few candidate boxes for each grid cell



J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, [You Only Look Once: Unified, Real-Time Object Detection](), CVPR 2016

YOLO

1. Take conv feature maps at 7x7 resolution
2. Add two FC layers to predict, at each location, a score for each class and 2 bboxes w/ confidences
   - For PASCAL, output is $7 \times 7 \times 30$ $(30 = 20 + 2 * (4 + 1))$



J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, [You Only Look Once: Unified, Real-Time Object Detection](), CVPR 2016

YOLO

- Objective function:

$$
\left.
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]
\end{aligned}
\right\} \text{Regression}
$$

$$
\left.
\begin{aligned}
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2
\end{aligned}
\right\} \text{Object/no object confidence}
$$

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

YOLO

- Objective function:

Cell i contains object, predictor j is responsible for it

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

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

Small deviations matter less for larger boxes than for smaller boxes

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

Confidence for object

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

Confidence for no object

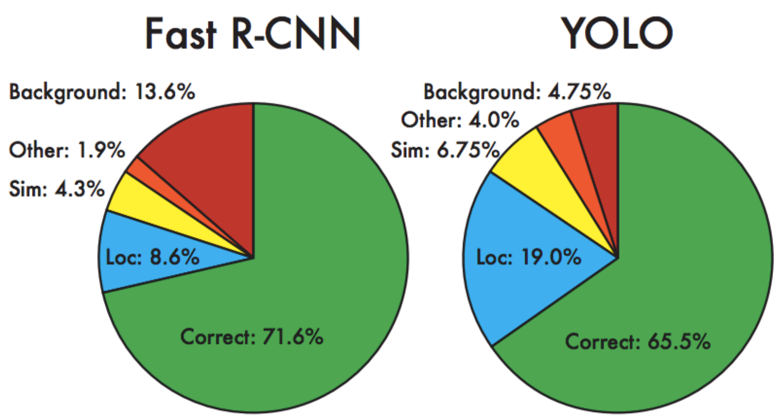
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Class probability

Down-weight loss from boxes that don’t contain objects ($\lambda_{\text{noobj}} = 0.5$)

YOLO: Results

- Each grid cell predicts only two boxes and can only have one class – this limits the number of nearby objects that can be predicted
- Localization accuracy suffers compared to Fast(er) R-CNN due to coarser features, errors on small boxes
- 7x speedup over Faster R-CNN (45-155 FPS vs. 7-18 FPS)

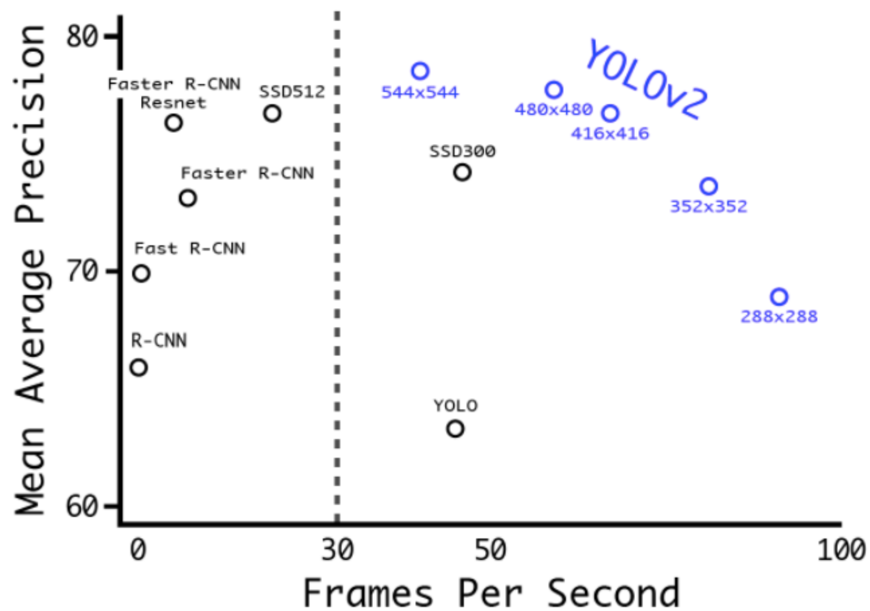


Performance on PASCAL 2007

YOLO v2

- Remove FC layer, do convolutional prediction with anchor boxes instead
- Increase resolution of input images and conv feature maps
- Improve accuracy using batch normalization and other tricks

VOC 2007 results

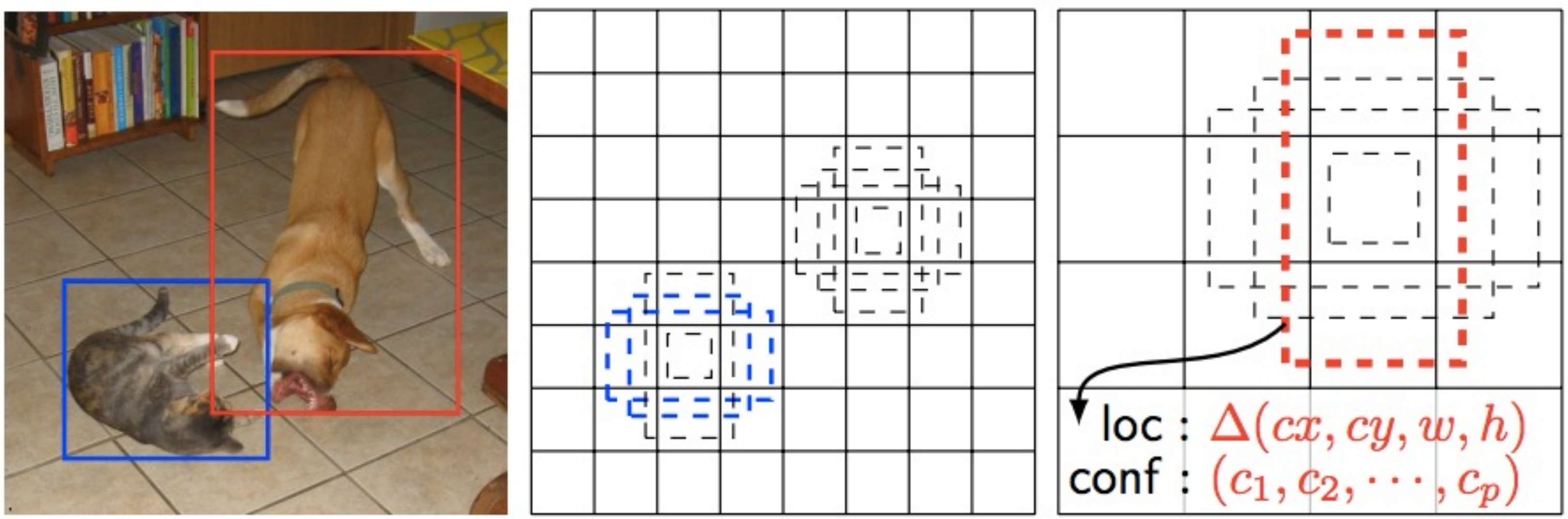


[YouTube demo](#)

J. Redmon and A. Farhadi, [YOLO9000: Better, Faster, Stronger](#), CVPR 2017

Multi-resolution prediction: SSD

- Predict boxes of different size from different conv maps
- Each level of resolution has its own predictor

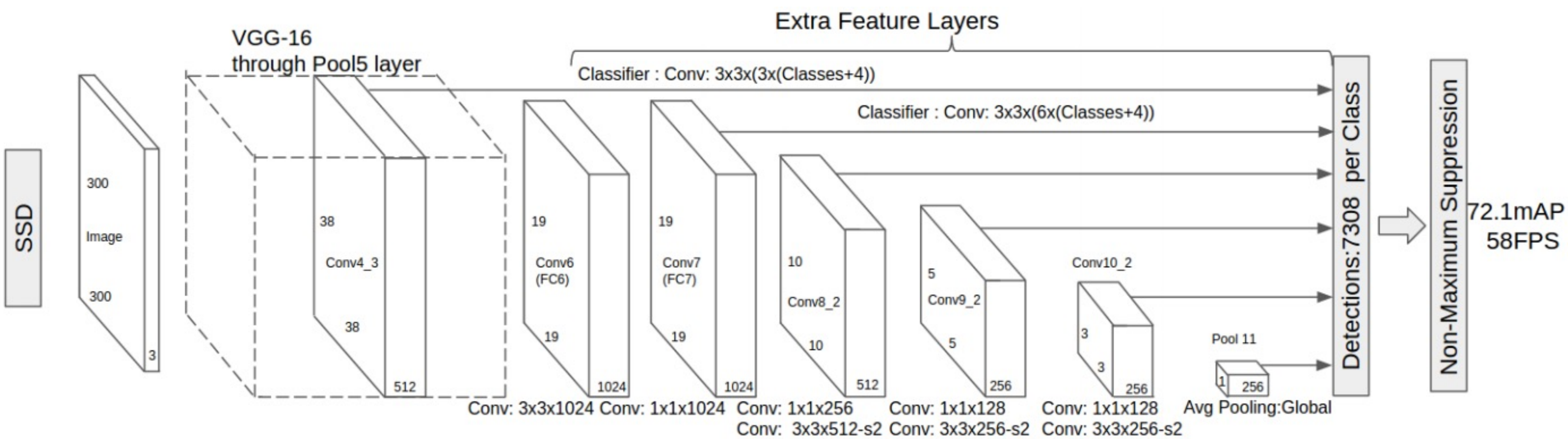


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg, [SSD: Single Shot MultiBox Detector,]() ECCV 2016

Multi-resolution prediction: SSD

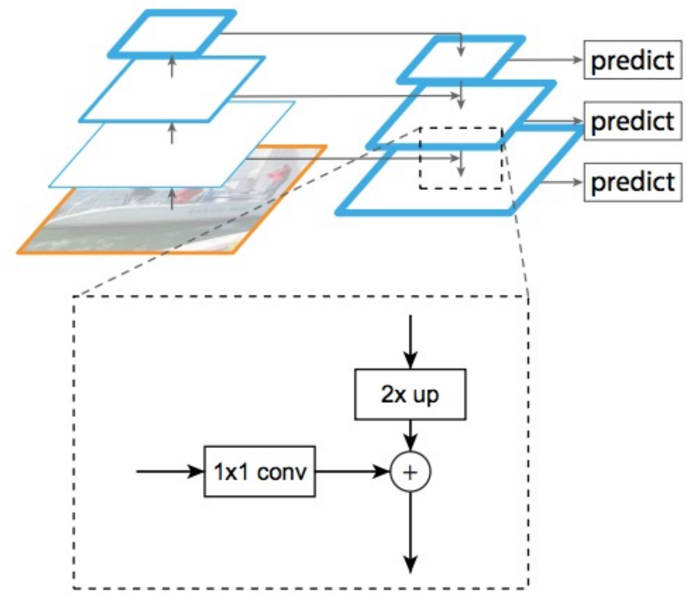
- Predict boxes of different size from different conv maps
- Each level of resolution has its own predictor



W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg, [SSD: Single Shot MultiBox Detector,](#) ECCV 2016

Feature pyramid networks

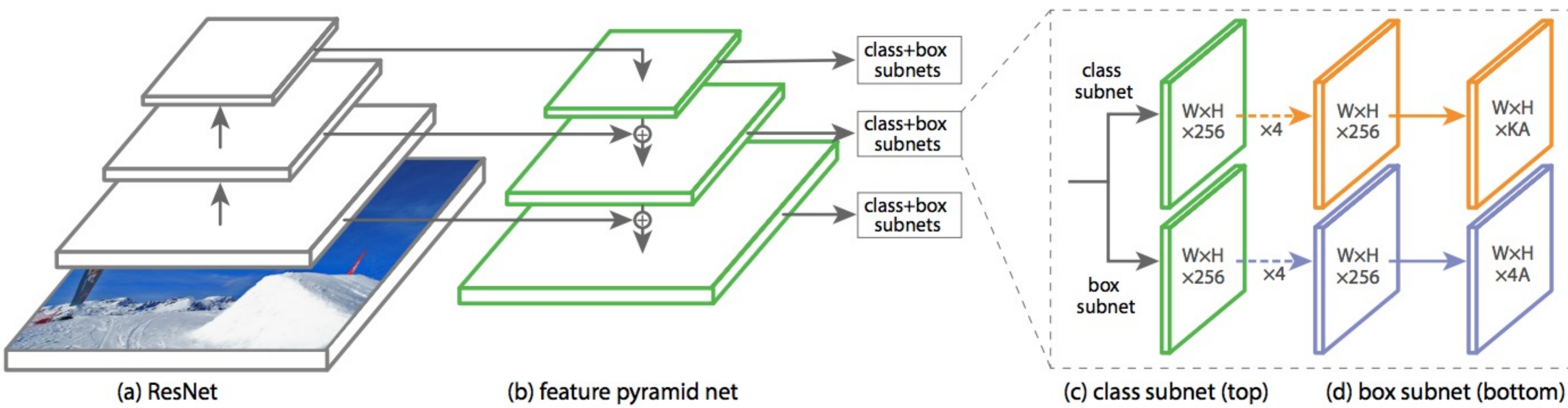
- Improve predictive power of lower-level feature maps by adding contextual information from higher-level feature maps
- Predict different sizes of bounding boxes from different levels of the pyramid (but share parameters of predictors)



T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, Feature pyramid networks for object detection, CVPR 2017

RetinaNet

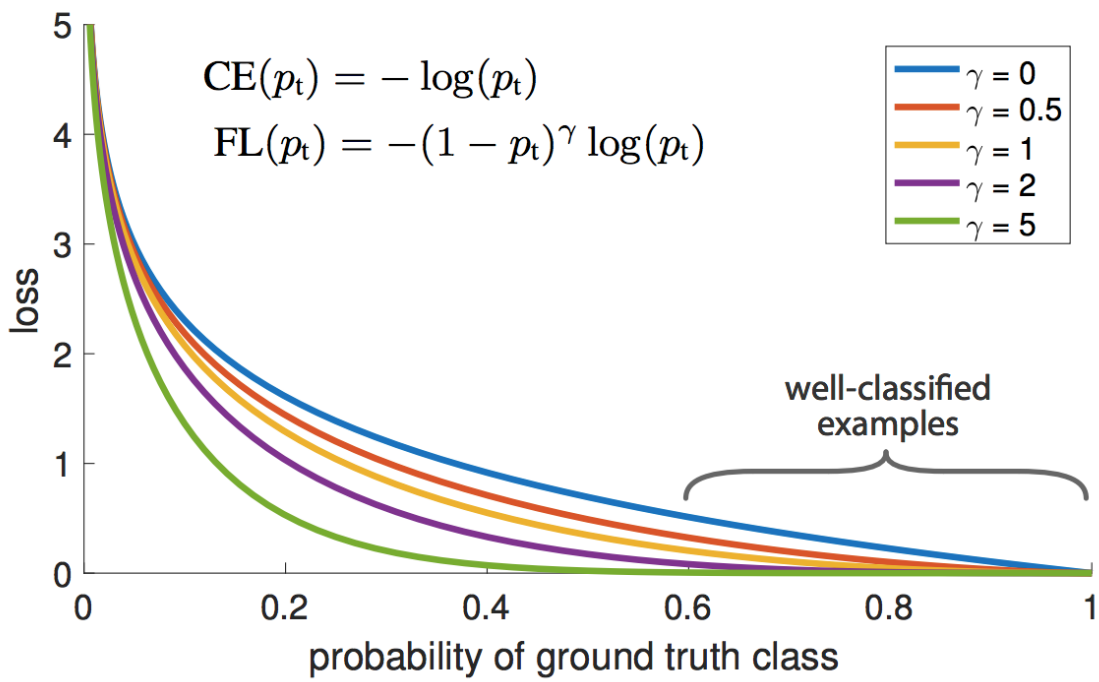
- **Classification subnet**: predict the probability of object at each position for each of *A* anchors and *K* object classes
- **Box subnet**: for each position and each anchor, predict offset to ground truth box (if any)



T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, ICCV 2017

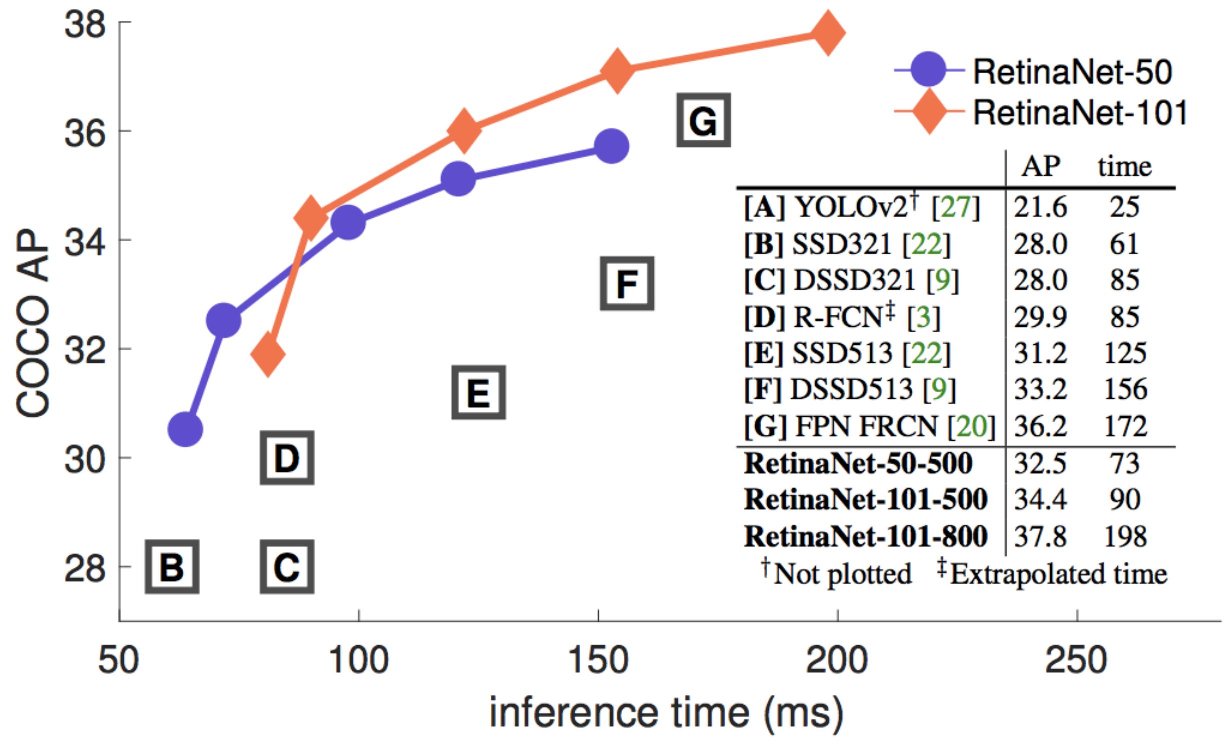
RetinaNet

- **Focal loss**: down-weight the standard cross-entropy loss for well-classified examples



T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, ICCV 2017

RetinaNet: Results



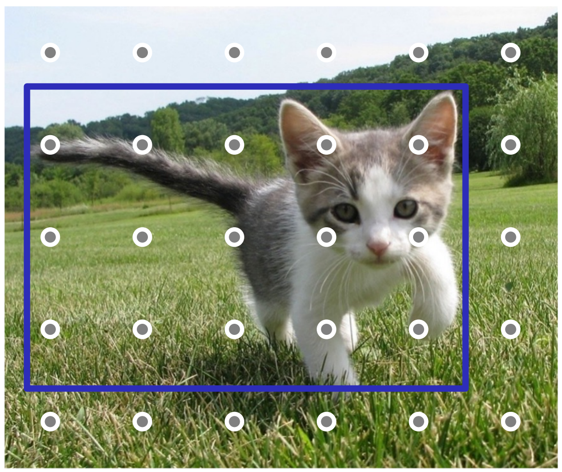
	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, ICCV 2017

Fully convolutional one-stage detector (FCOS)

- “Anchor-free” approach



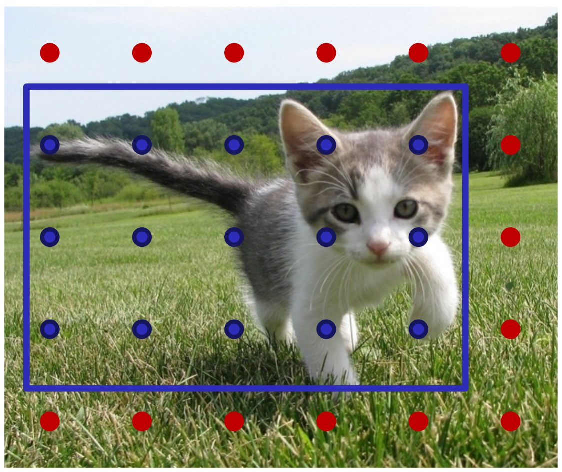
Run backbone CNN to get features aligned to input image

Figure source: [J. Johnson]

Tian et al., FCOS: Fully Convolutional One-Stage Object Detection, ICCV 2019

Fully convolutional one-stage detector (FCOS)

- “Anchor-free” approach



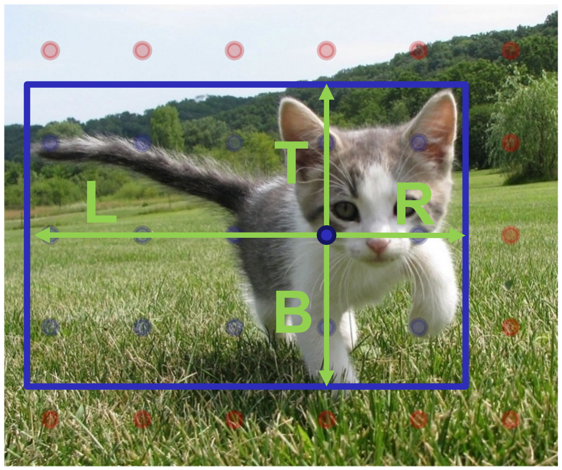
For each class, predict whether location falls inside a GT bounding box

Figure source: J. Johnson

Tian et al., FCOS: Fully Convolutional One-Stage Object Detection, ICCV 2019

Fully convolutional one-stage detector (FCOS)

- "Anchor-free" approach



For positive points, also regress distance to left, right, top, and bottom of GT box (with L2 loss)

Figure source: J. Johnson

Tian et al., FCOS: Fully Convolutional One-Stage Object Detection, ICCV 2019

Fully convolutional one-stage detector (FCOS)

- “Anchor-free” approach

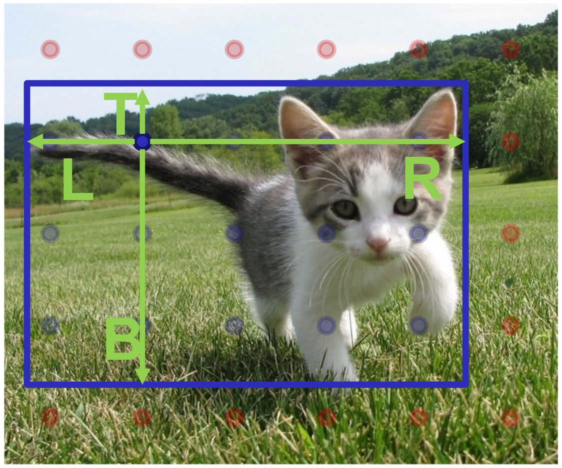


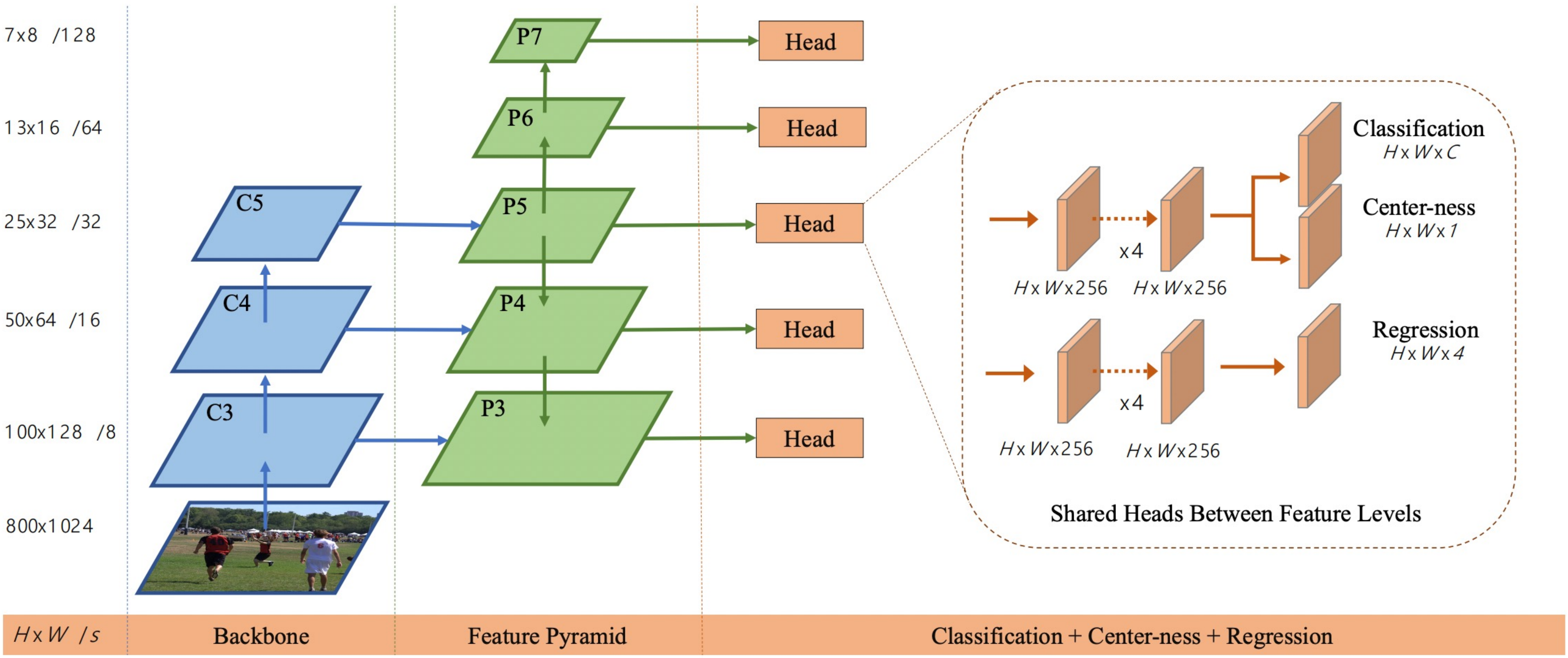
Figure source: [J. Johnson](#)

For positive points, also regress distance to left, right, top, and bottom of GT box (with L2 loss)

Weight detections by “centerness” and confidence, perform NMS

Tian et al., [FCOS: Fully Convolutional One-Stage Object Detection](#), ICCV 2019

Fully convolutional one-stage detector (FCOS)



Tian et al., [FCOS: Fully Convolutional One-Stage Object Detection](), ICCV 2019

Outline

- Task definition and evaluation
- Two-stage detectors
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single-stage and multi-resolution detectors
- Other detectors: CornerNet, DETR

CornerNet

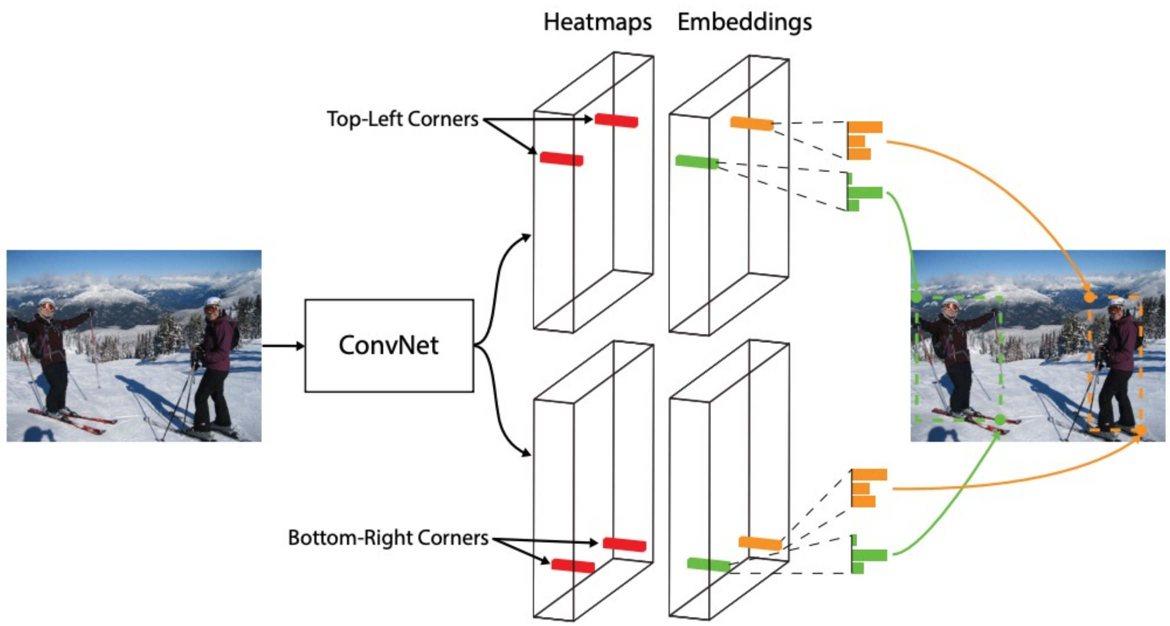
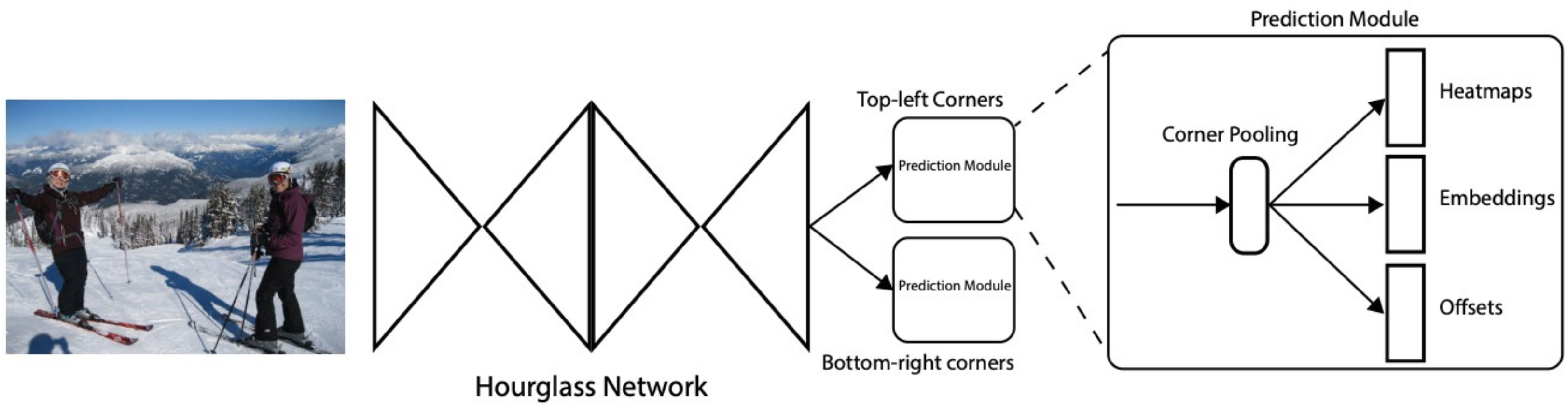


Fig. 1. We detect an object as a pair of bounding box corners grouped together. A convolutional network outputs a heatmap for all top-left corners, a heatmap for all bottom-right corners, and an embedding vector for each detected corner. The network is trained to predict similar embeddings for corners that belong to the same object.

H. Law and J. Deng, [CornerNet: Detecting Objects as Paired Keypoints,]() ECCV 2018

CornerNet



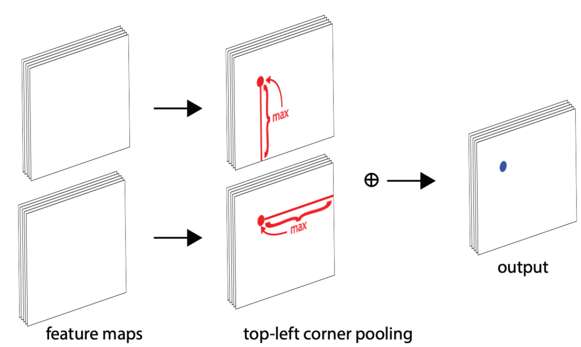
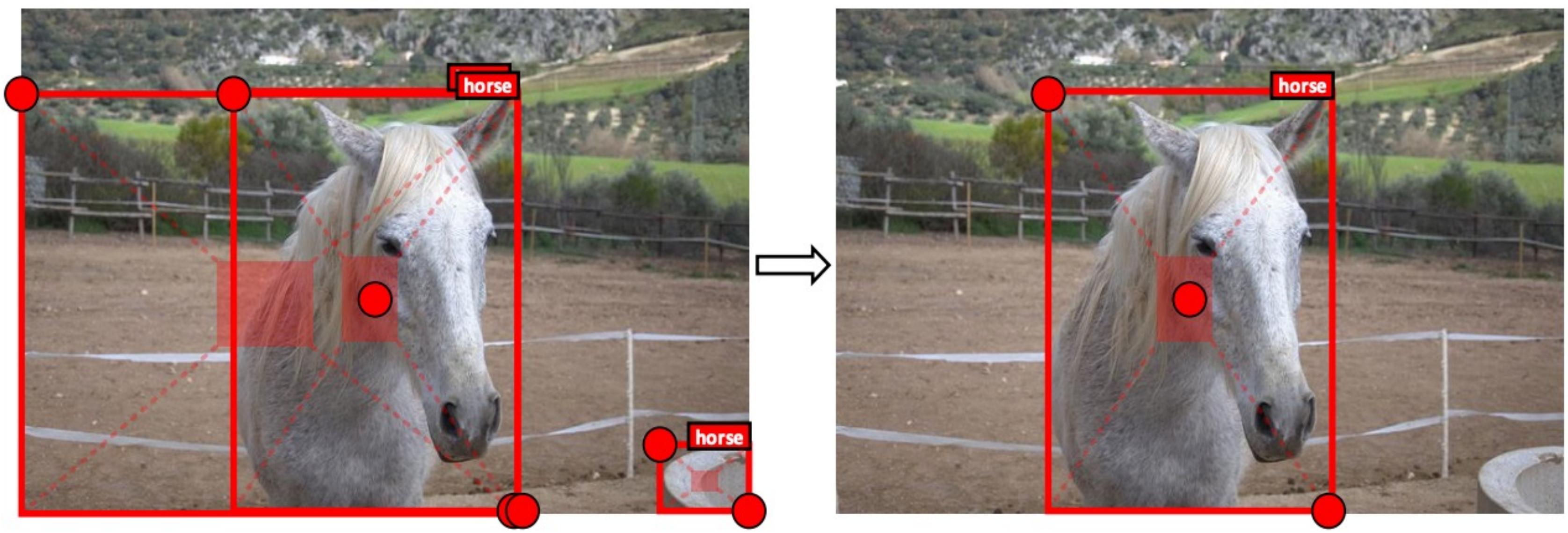


Fig. 2. Often there is no local evidence to determine the location of a bounding box corner. We address this issue by proposing a new type of pooling layer.

H. Law and J. Deng, [CornerNet: Detecting Objects as Paired Keypoints,](#) ECCV 2018

CenterNet

- Use an additional center point to verify predictions:



K. Duan et al. [CenterNet: Keypoint Triplets for Object Detection](), ICCV 2019

CenterNet

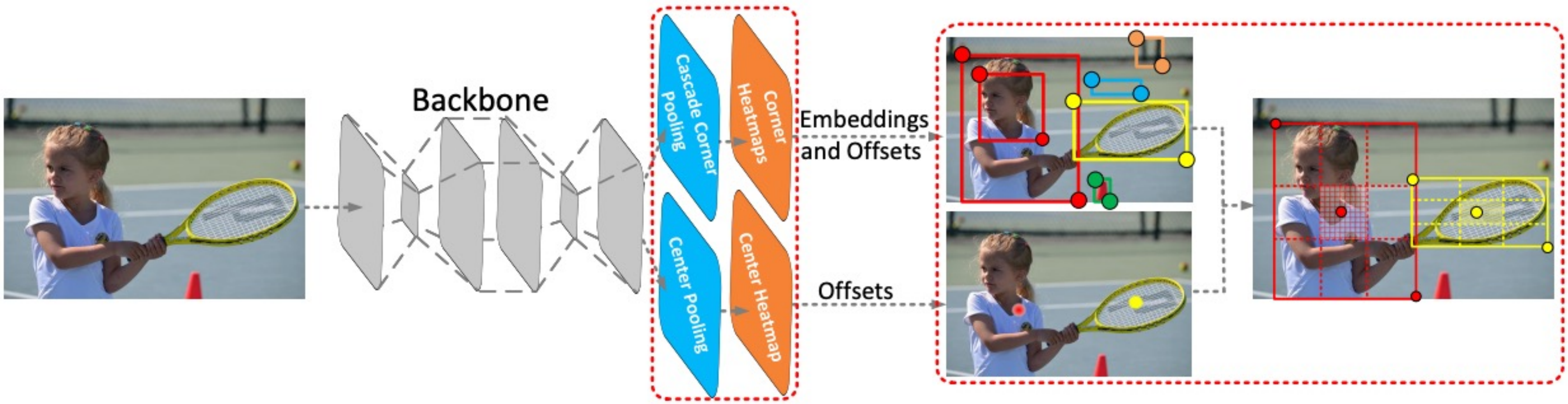


Figure 2: Architecture of CenterNet. A convolutional backbone network applies cascade corner pooling and center pooling to output two corner heatmaps and a center keypoint heatmap, respectively. Similar to CornerNet, a pair of detected corners and the similar embeddings are used to detect a potential bounding box. Then the detected center keypoints are used to determine the final bounding boxes.

K. Duan et al. [CenterNet: Keypoint Triplets for Object Detection,](#) ICCV 2019

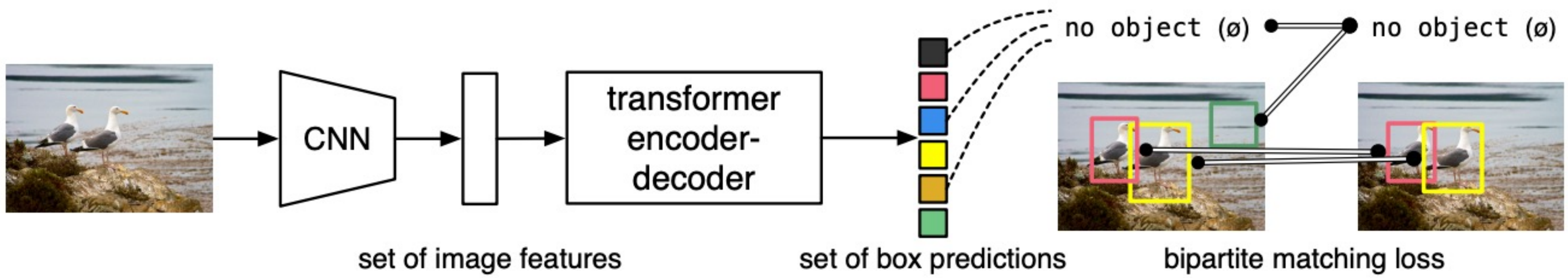
CenterNet

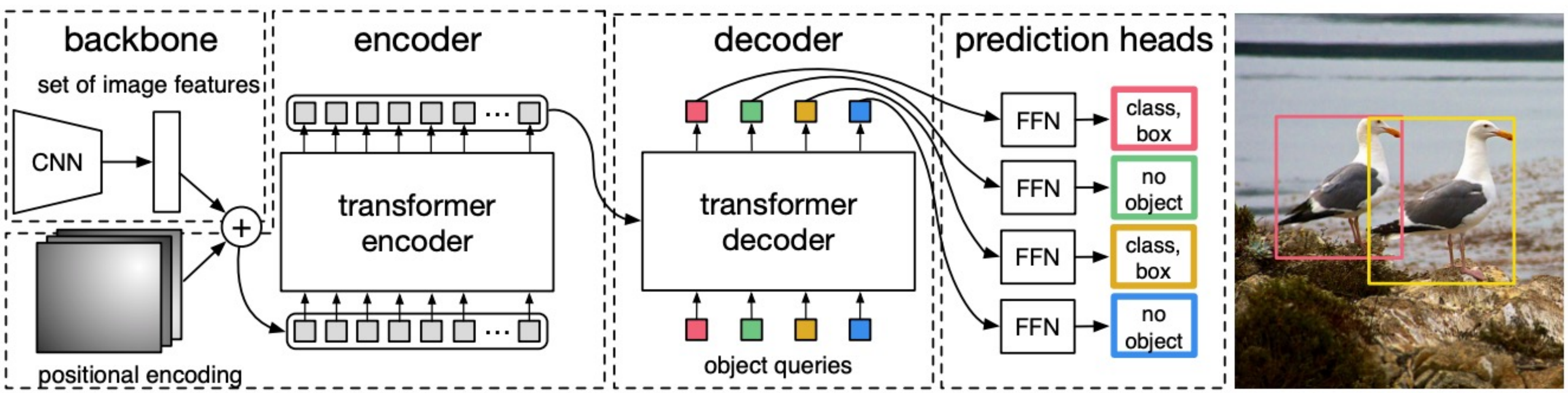
Method	FD	FD_5	FD_{25}	FD_{50}	FD_S	FD_M	FD_L
CornerNet511-52	40.4	35.2	39.4	46.7	62.5	36.9	28.0
CenterNet511-52	**35.1**	**30.7**	**34.2**	**40.8**	**53.0**	**31.3**	**24.4**
CornerNet511-104	37.8	32.7	36.8	43.8	60.3	33.2	25.1
CenterNet511-104	**32.4**	**28.2**	**31.6**	**37.5**	**50.7**	**27.1**	**23.0**

Table 3: Comparison of the false discovery rates (%) of CornerNet and CenterNet on the MS-COCO validation dataset. The results suggest that CenterNet avoids a large number of incorrect bounding boxes, especially for small incorrect bounding boxes.

K. Duan et al. CenterNet: Keypoint Triplets for Object Detection, ICCV 2019

Detection Transformer (DETR)





N. Carion et al., [End-to-end object detection with transformers](), ECCV 2020