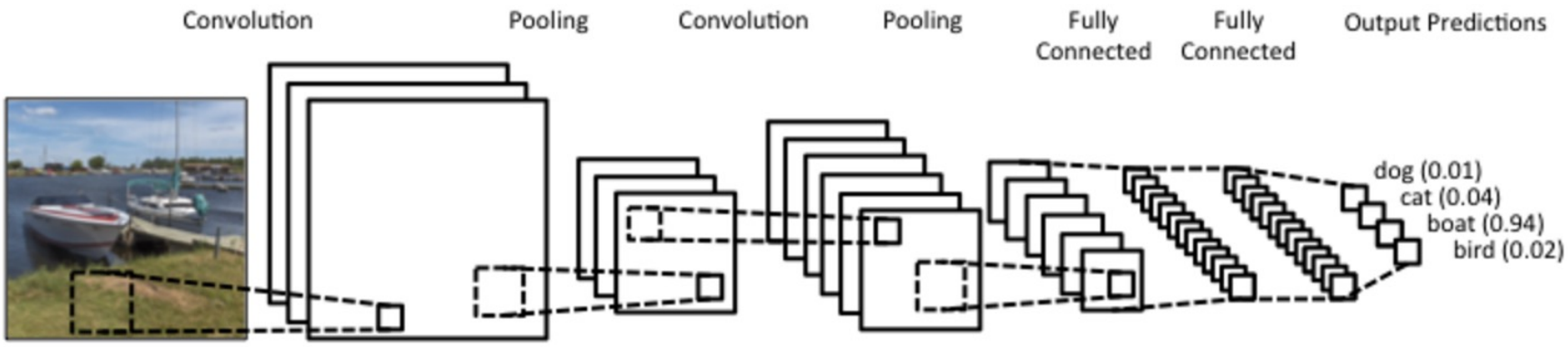
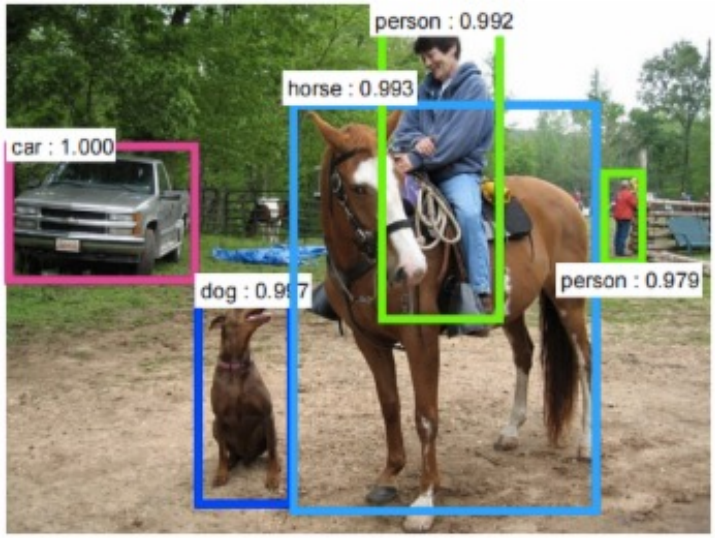
From image classification to object detection

Image classification



Object detection



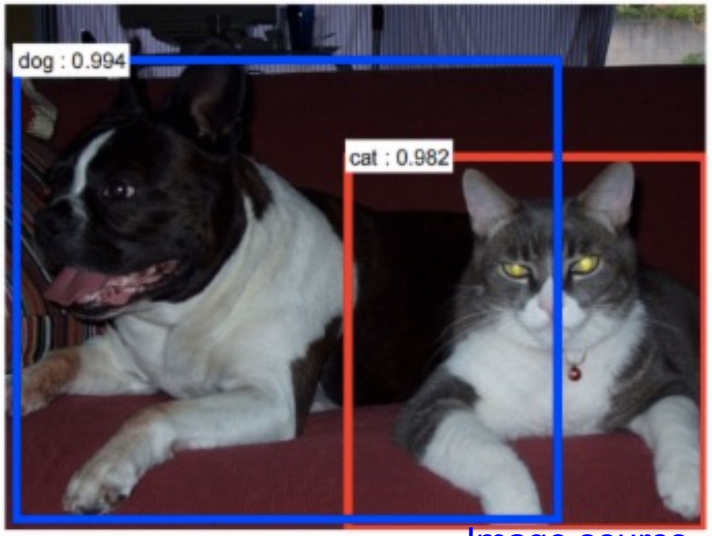


Image source

Real-time Multi-Person 2D Pose Estimation Using Part Affinity Fields

Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh

Carnegie Mellon University

What are the challenges of object detection?

- Images may contain more than one class, multiple instances from the same class
- Bounding box localization
- Evaluation

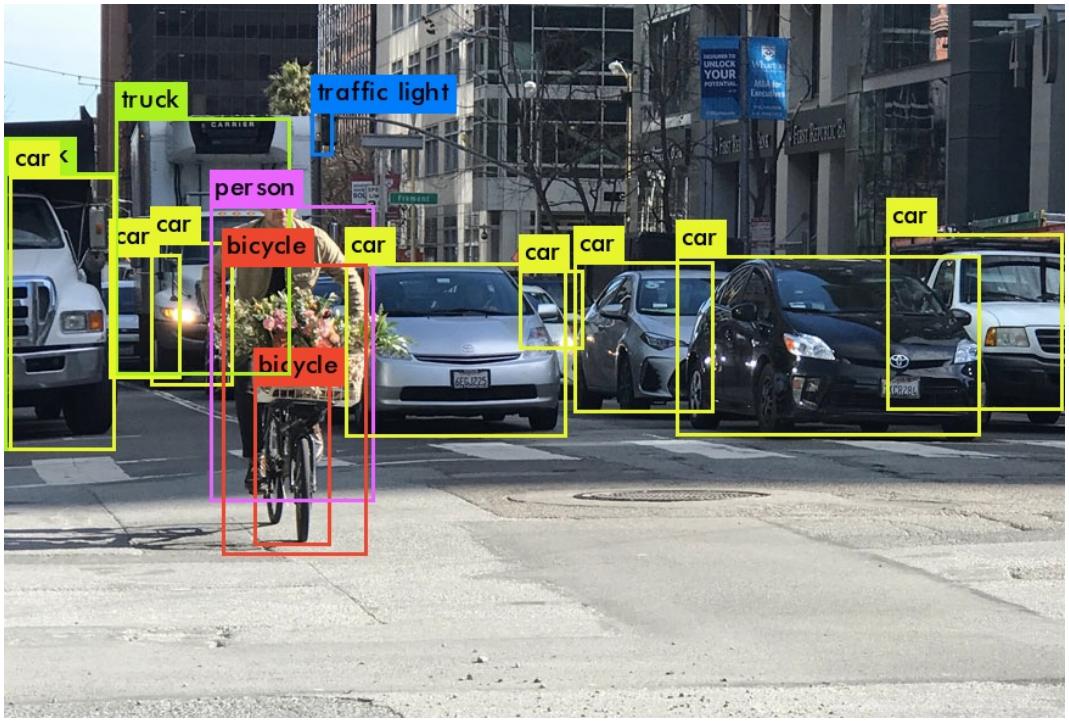


Image source

Outline

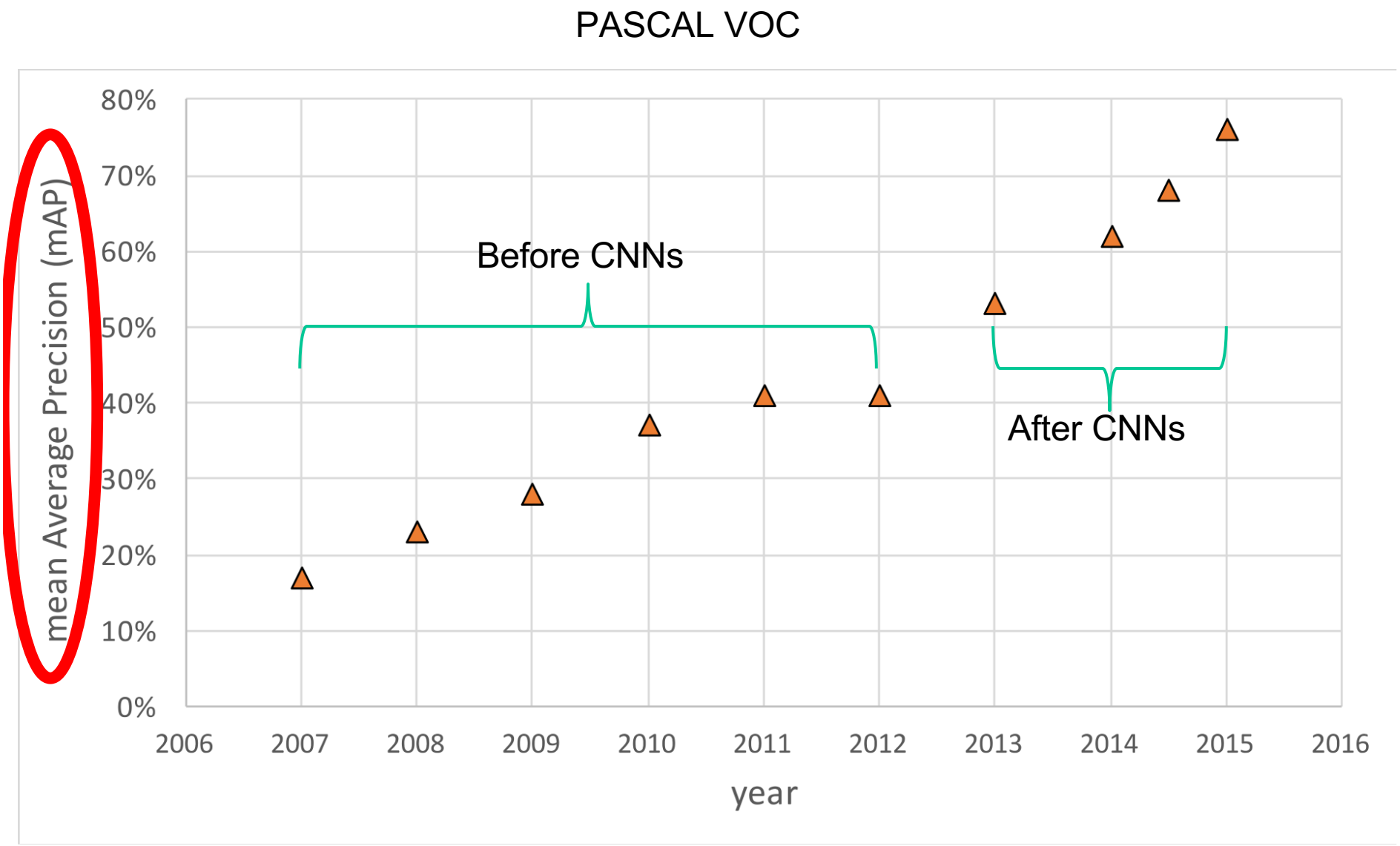
- Task definition and evaluation
- Conceptual approaches to detection
- Zoo of deep detection approaches
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - Yolo
  - SSD

PASCAL VOC Challenge (2005-2012)

- 20 challenge classes:
  - *Person*
  - *Animals:* bird, cat, cow, dog, horse, sheep
  - *Vehicles:* aeroplane, 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



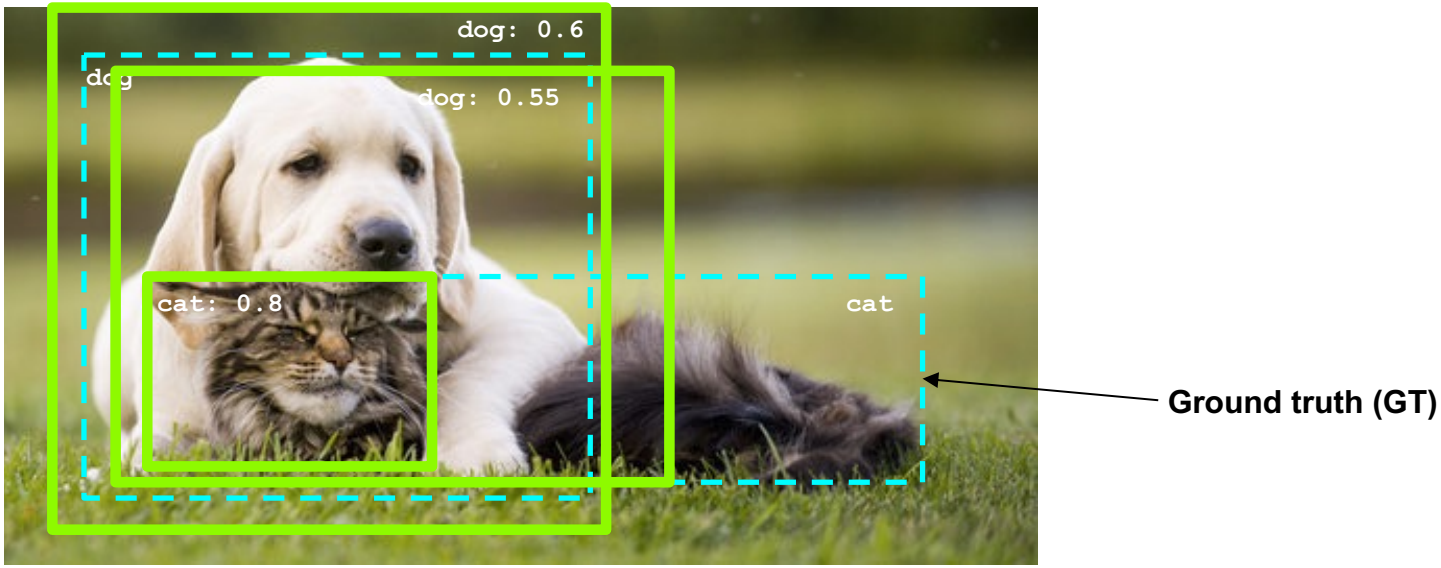
COCO detection metrics

```
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
```

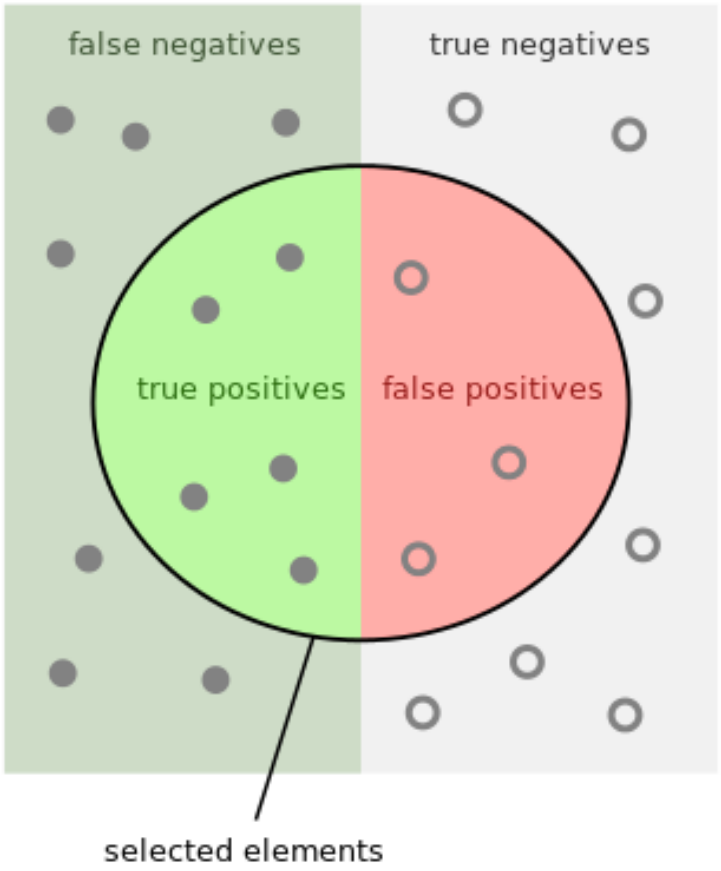
- Leaderboard: http://cocodataset.org/#detection-leaderboard
- Official COCO challenges no longer include detection
  - Emphasis has shifted to instance segmentation and dense semantic segmentation

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
  - PASCAL criterion: Area(GT∩Det) / Area(GT∪Det) > 0.5
  - For multiple detections of the same ground truth box, only one considered a true positive



Precision and Recall



How many selected items are relevant?

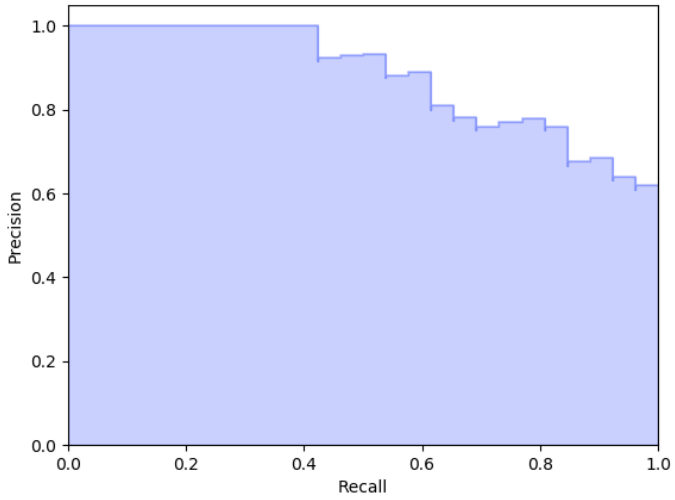
Precision = $\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$

How many relevant items are selected?

Recall = $\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$

Object detection evaluation

- For each class, plot Precision-Recall curve
- Compute Average Precision (area under curve = AUC)
- Take mean of AP over classes to get mAP



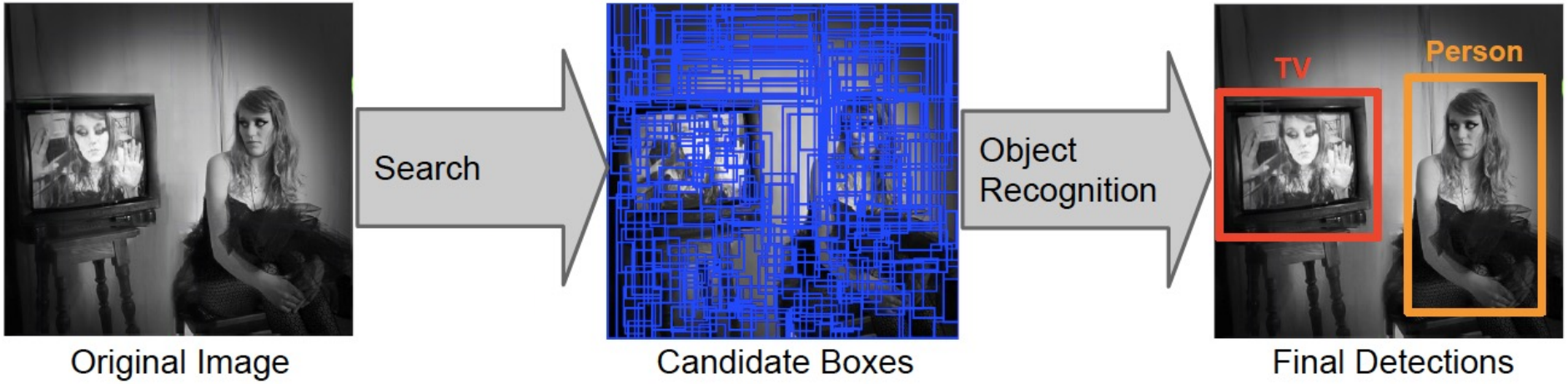
Precision:
true positive detections /
total detections

Recall:
true positive detections /
total positive test instances

Conceptual approach: Sliding window detection

- Slide a window across the image
- Evaluate a detection model at each location
  - Thousands of windows to evaluate: efficiency? false positives?
  - Difficult to extend to a large range of scales and aspect ratios

Conceptual approach: Proposal-driven detection



- Generate a few hundred *region proposals*
  - Proposals can take advantage of low-level perceptual cues
  - Proposal mechanism can be
    - category-specific or agnostic
    - hand-crafted or trained
  - Classifier can be slower but more powerful

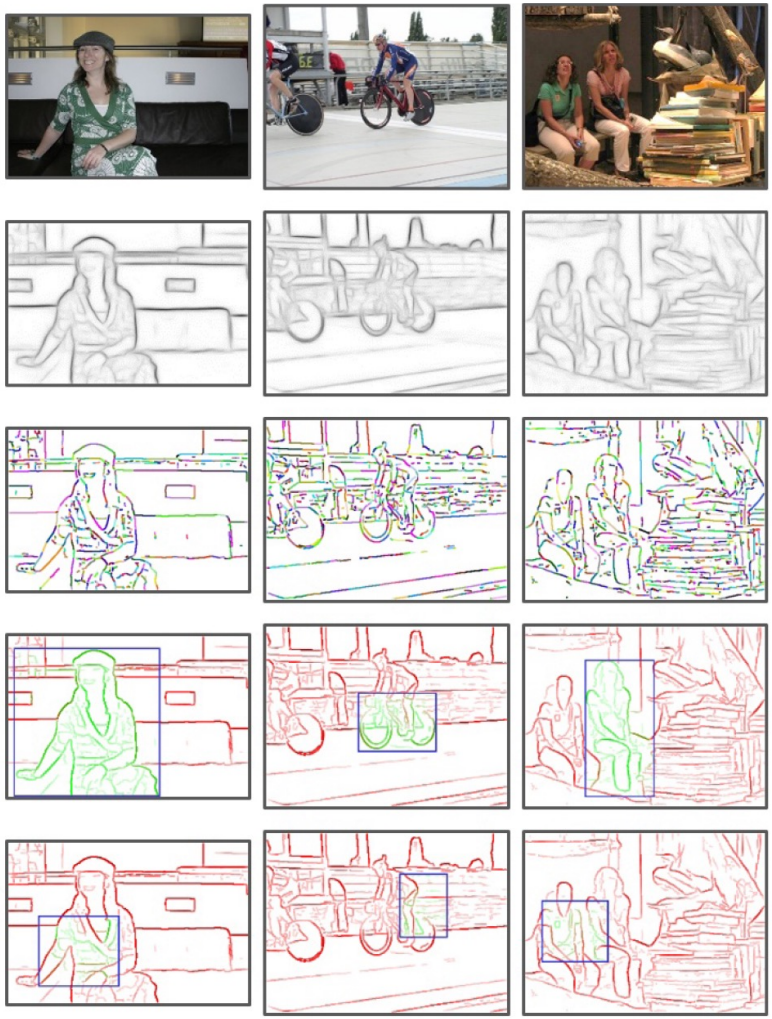
“Selective Search” for detection

- Use hierarchical segmentation: start with small *superpixels* and merge based on diverse cues

J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders
Selective Search for Object Recognition, IJCV 2013

Another proposal method: EdgeBoxes

- Box score: number of edges in the box minus number of edges that overlap the box boundary
- Uses a trained edge detector
- Uses efficient data structures (incl. integral images) for fast evaluation
- Gets 75% recall with 800 boxes (vs. 1400 for Selective Search)
- 40 times faster

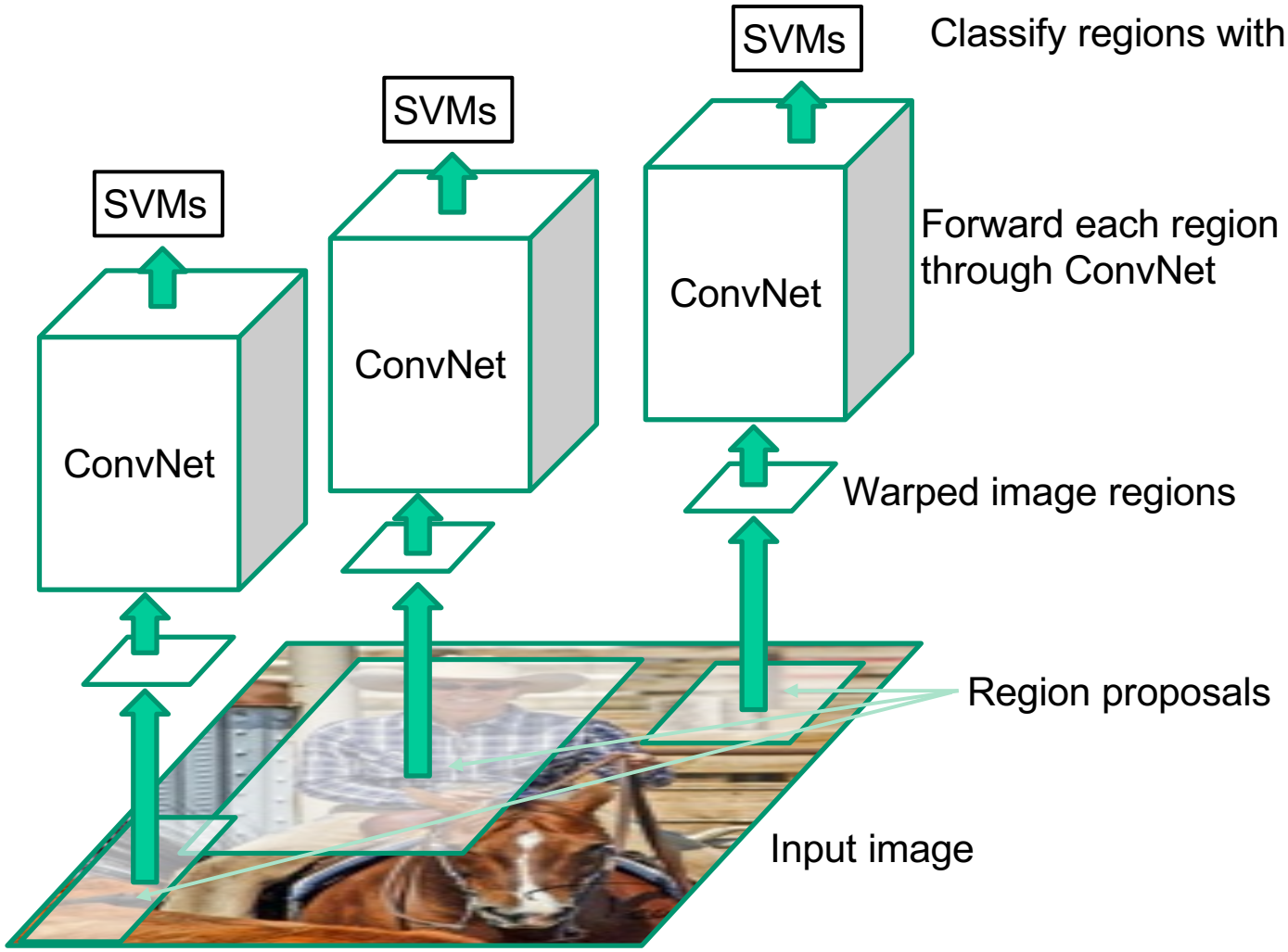


C. Zitnick and P. Dollar

Edge Boxes: Locating Object Proposals from Edges, ECCV 2014

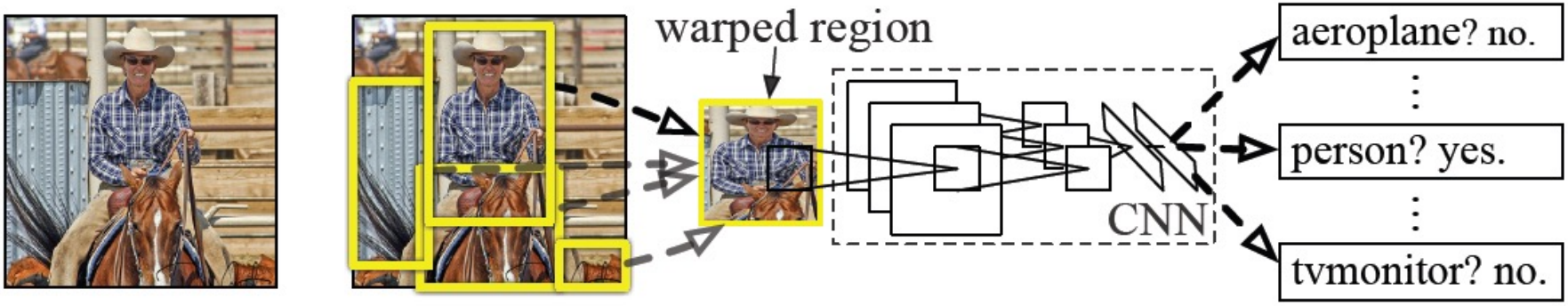
R-CNN: Region proposals + CNN features

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

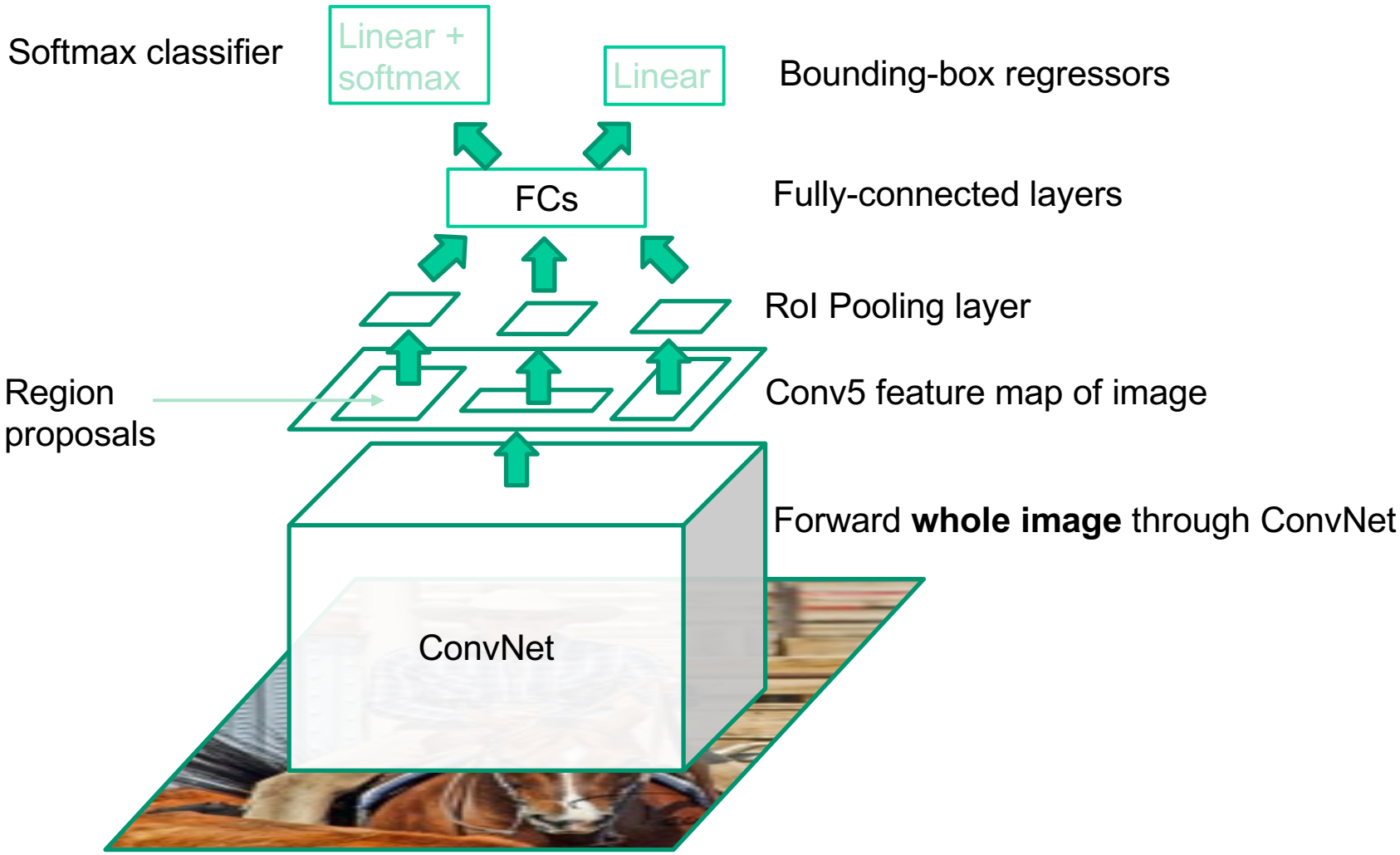


- **Regions**: ~2000 Selective Search proposals
- **Network**:
  - AlexNet *pre-trained* on ImageNet (1000 classes)
  - *fine-tuned* on PASCAL (21 classes)
- **Final detector**: warp proposal regions, extract fc7 network activations (4096 dimensions), classify with linear SVM
- **Bounding box regression**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

R-CNN pros and cons

- Pros
  - Accurate!
  - Any deep architecture can immediately be “plugged in”
- Cons
  - **Not end-to-end**
    - Fine-tune network with softmax classifier (log loss)
    - Train post-hoc linear SVMs (hinge loss)
    - Train post-hoc bounding-box regressions (least squares)
  - Training is **slow** (84h) takes a lot of disk space
    - Why? ~2000 CNN passes per image
  - Inference is **slow** (47s / image with VGG16)

Fast R-CNN

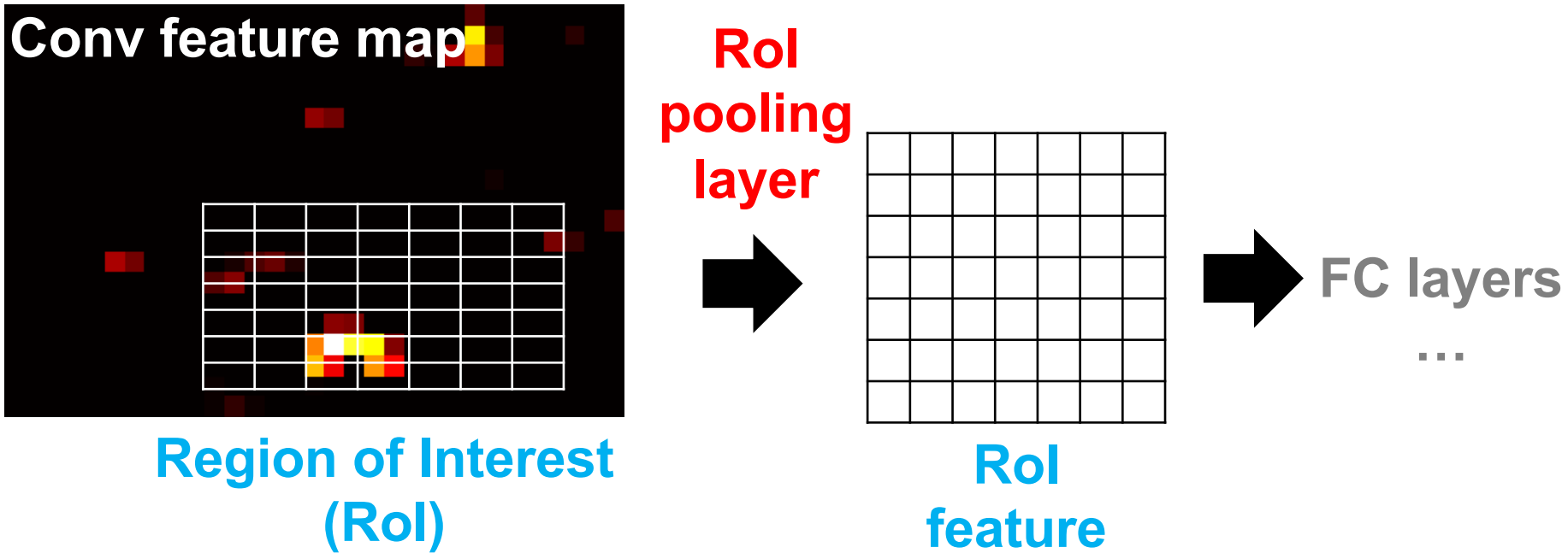


Source: R. Girshick

R. Girshick, Fast R-CNN, ICCV 2015

RoI Pooling Layer

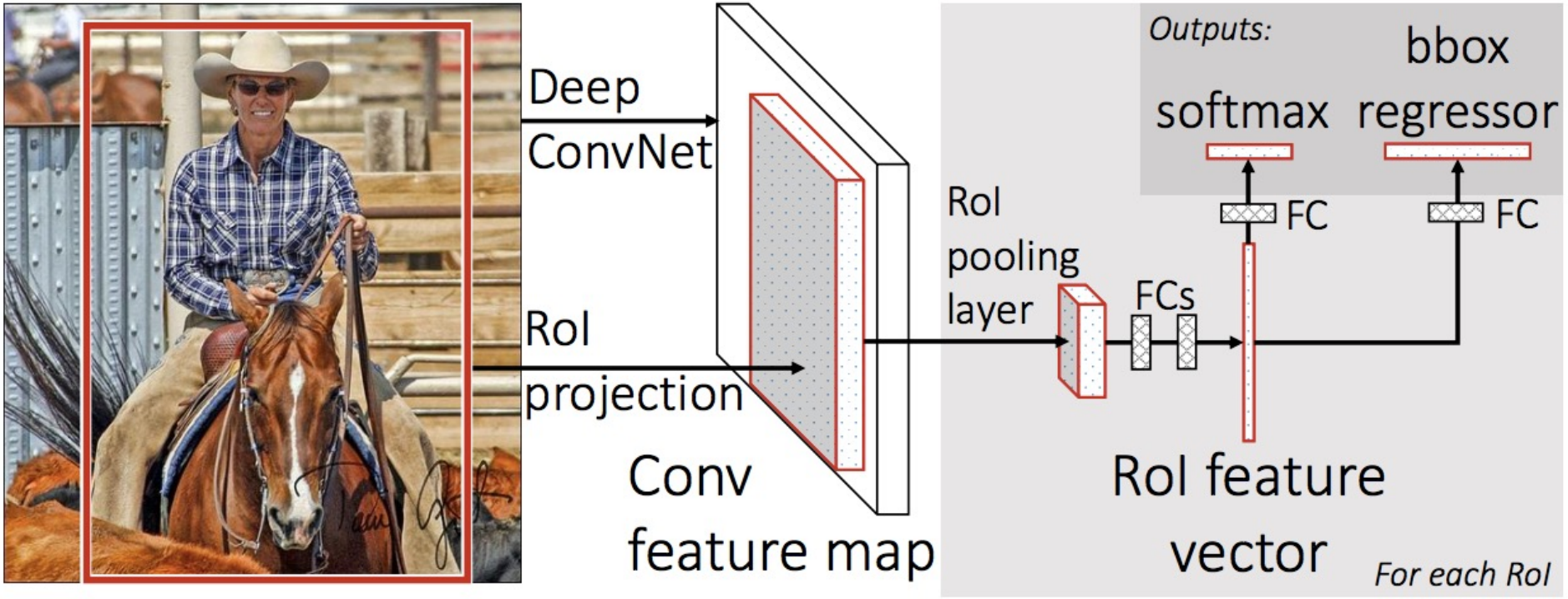
- "Crop and resample" a fixed-size feature representing a region of interest out of the outputs of the last conv layer
  - Use nearest-neighbor interpolation of coordinates, max pooling



Source: R. Girshick, K. He

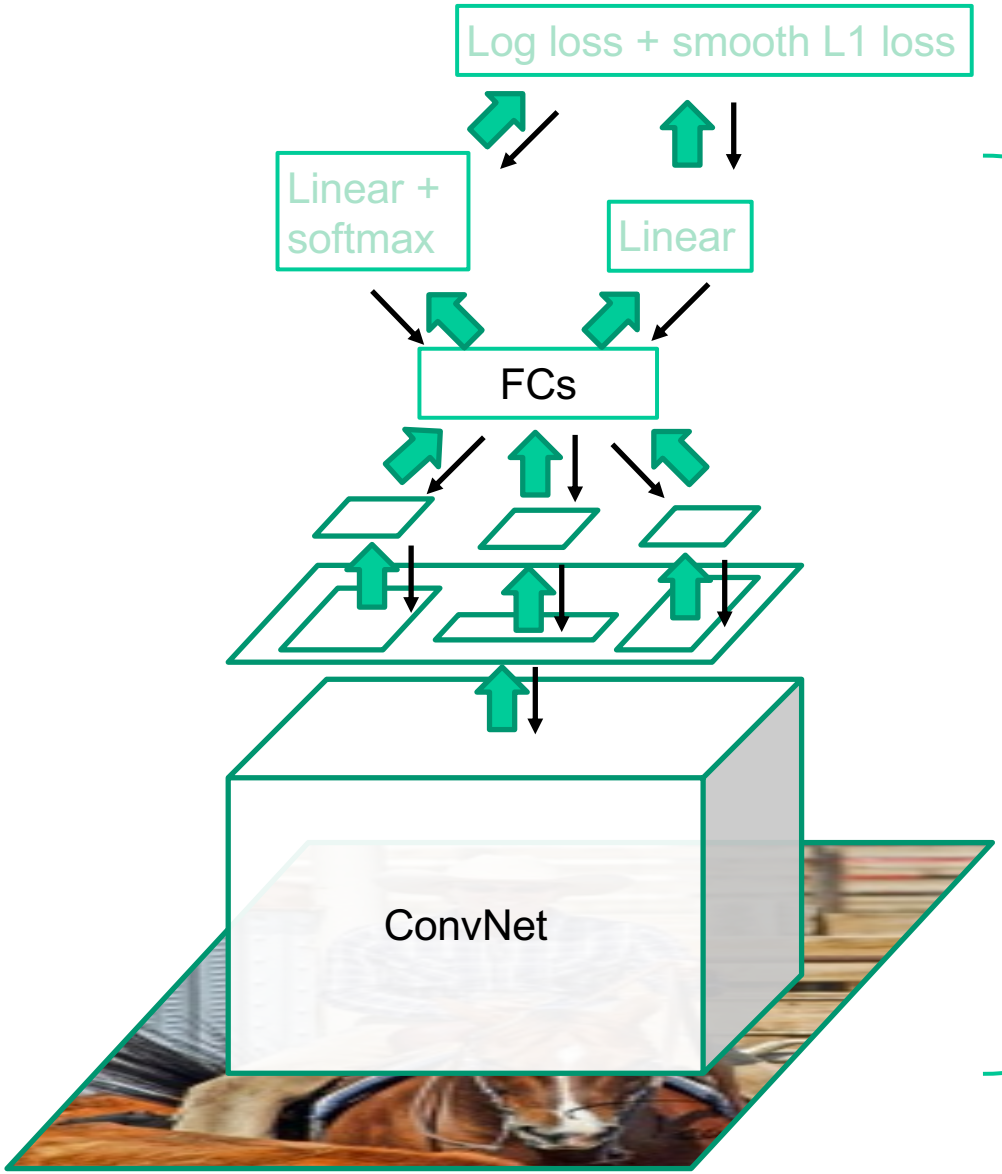
Prediction

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



R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN training



Multi-task loss

Trainable

Source: R. Girshick

R. Girshick, Fast R-CNN, ICCV 2015

Mathematical Digression

Kullback–Leibler divergence

21 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

In mathematical statistics, the **Kullback–Leibler** (**KL**) **divergence** (also called **relative entropy** and **I-divergence**[1]), denoted $D_{\mathrm{KL}}(P \parallel Q)$, is a type of statistical distance: a measure of how much a model probability distribution Q is different from a true probability distribution P.[2][3] Mathematically, it is defined as

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x)\ \log\left(\frac{P(x)}{Q(x)}\right).$$

Mathematical Digression

Softmax function

18 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

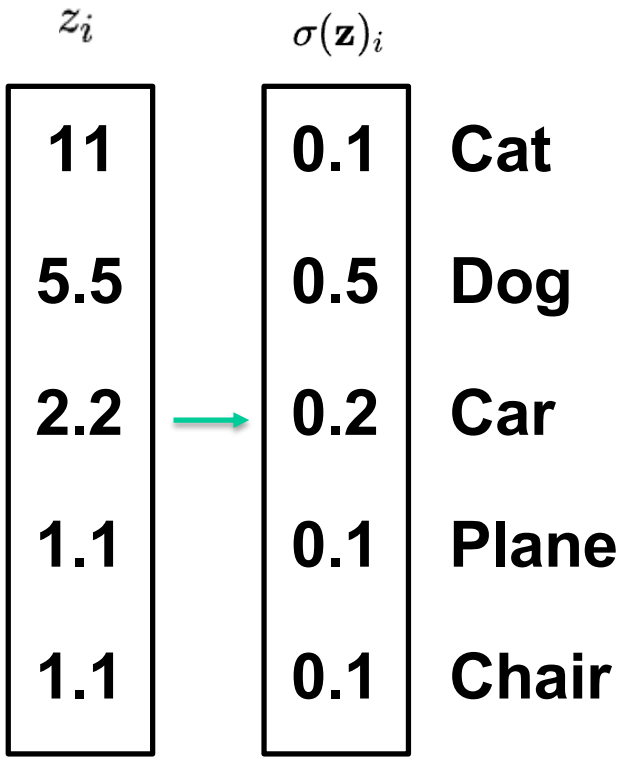
Formally, the standard (unit) softmax function $\sigma\colon \mathbb{R}^K \to (0,1)^K$, where $K > 1$, takes a vector $\mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$ and computes each component of vector $\sigma(\mathbf{z}) \in (0,1)^K$ with

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} .$$

In words, the softmax applies the standard exponential function to each element z_i of the input vector $\mathbf{z}$ (consisting of K real numbers), and normalizes these values by dividing by the sum of all these exponentials. The normalization ensures that the sum of the components of the output vector $\sigma(\mathbf{z})$ is 1. The term "softmax" derives from the amplifying effects of the exponential on any maxima in the input vector. For example, the standard softmax of $(1, 2, 8)$ is approximately $(0.001, 0.002, 0.997)$, which amounts to assigning almost all of the total unit weight in the result to the position of the vector's maximal element (of 8).

Softmax Loss

logits



$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}.$$

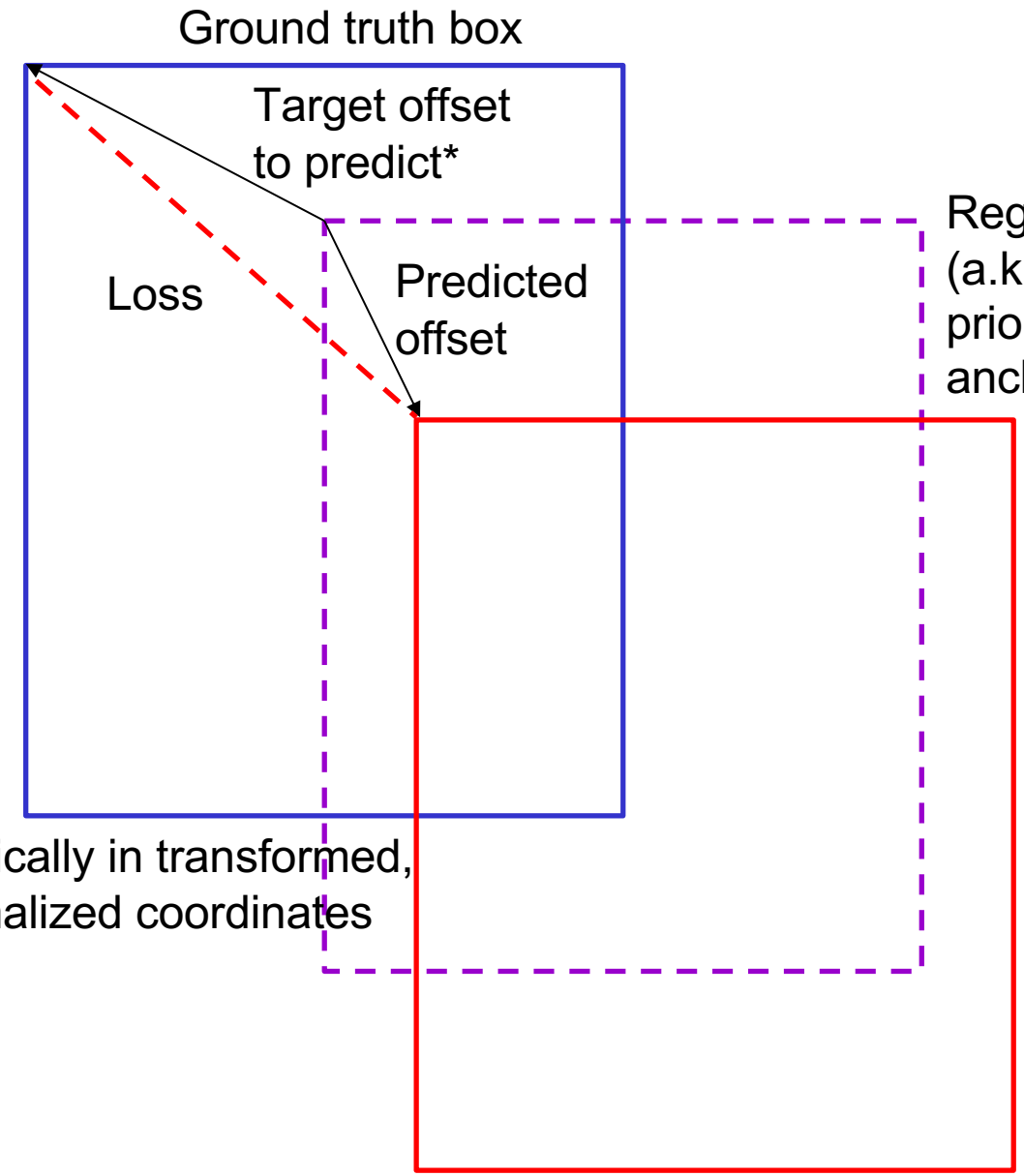
Binary Cross Entropy
(on post soft-max logits)

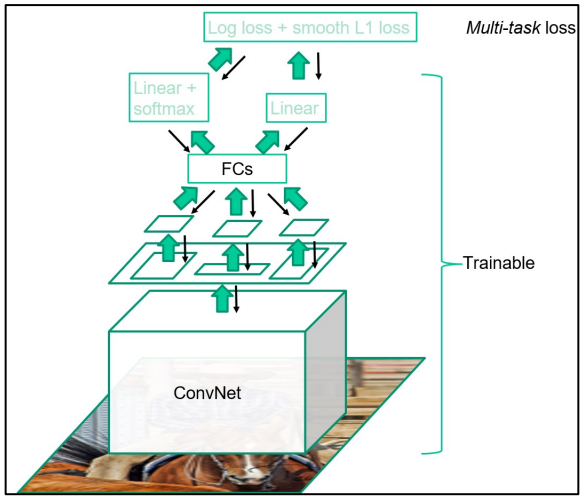
$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v), \quad (1)$$

in which $L_{\text{cls}}(p, u) = -\log p_u$ is log loss for true class u.

The second task loss, L_{loc}, is defined over a tuple of true bounding-box regression targets for class u, $v = (v_{\text{x}}, v_{\text{y}}, v_{\text{w}}, v_{\text{h}})$, and a predicted tuple $t^u = (t^u_{\text{x}}, t^u_{\text{y}}, t^u_{\text{w}}, t^u_{\text{h}})$, again for class u. The Iverson bracket indicator function $[u \geq 1]$ evaluates to 1 when $u \geq 1$ and 0 otherwise. By convention the catch-all background class is labeled $u = 0$. For background RoIs there is no notion of a ground-truth

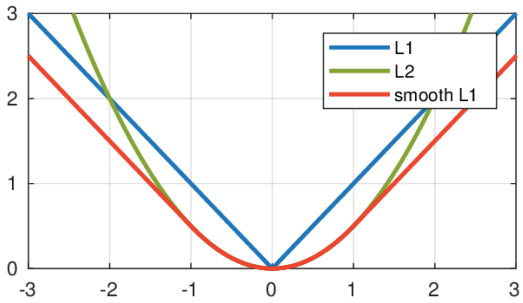
Bounding box regression





Multi-task loss

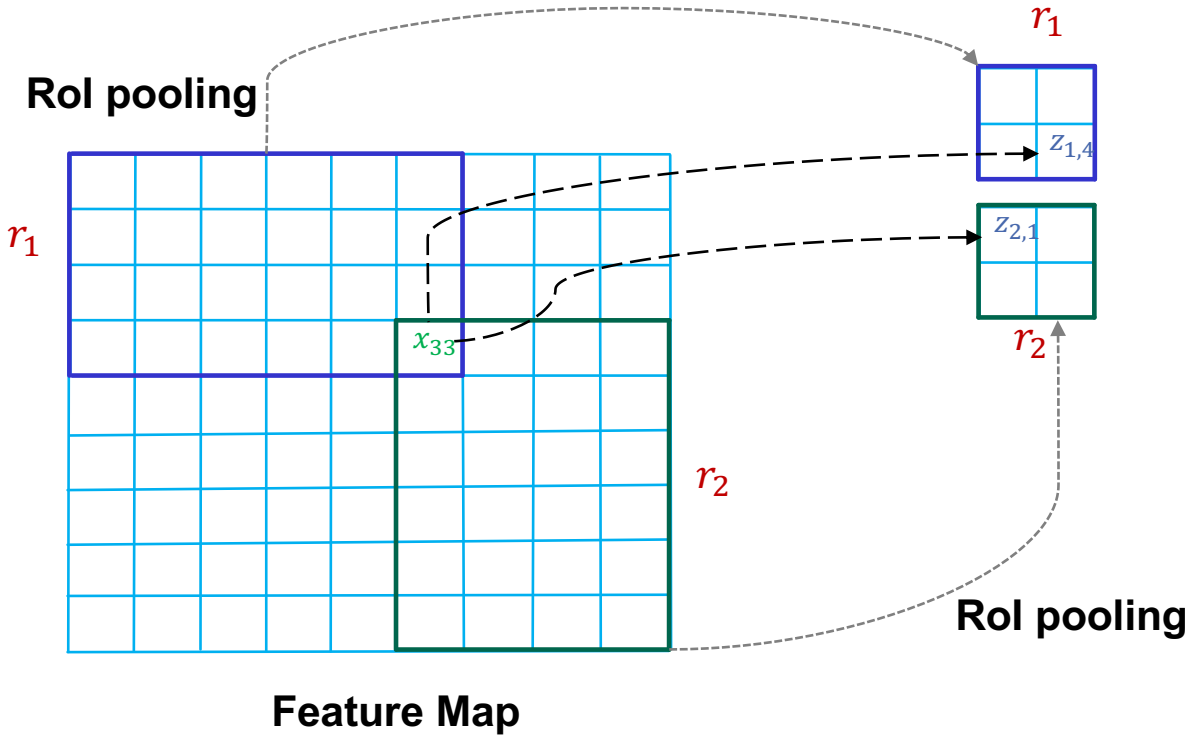
- Regression loss: *smooth L1 loss*



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

ROI pooling: Backpropagation

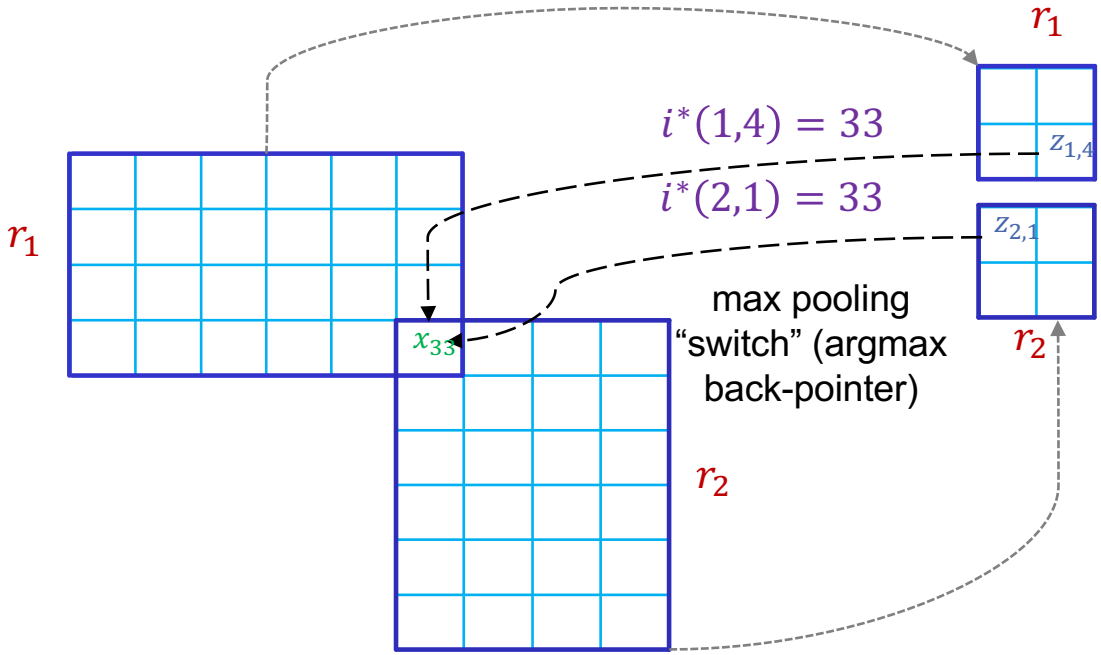
- Similar to max pooling, has to take into account overlap of pooling regions



Source: Ross Girshick

ROI pooling: Backpropagation

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



Backward Pass:

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

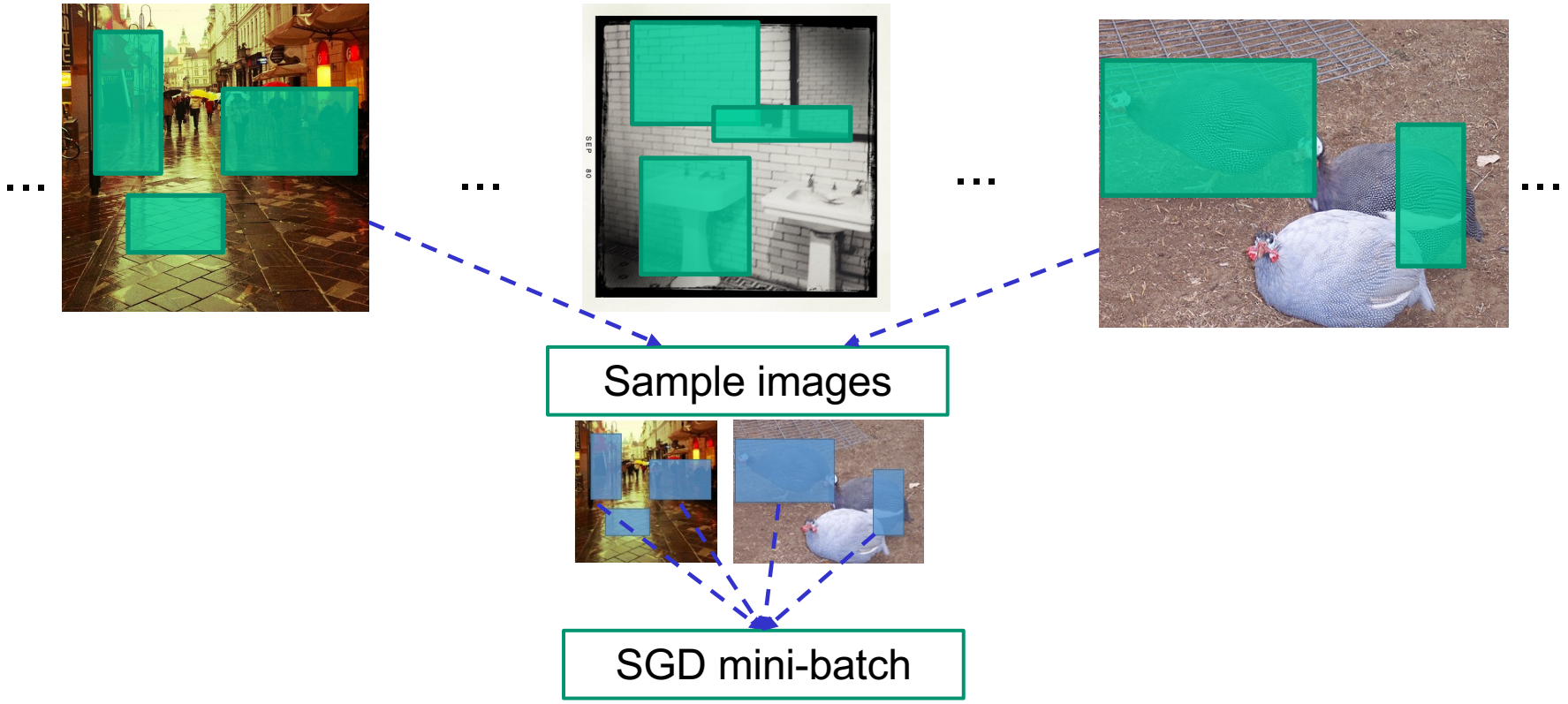
Over regions r, RoI indices j

1 if "pooled" input ; 0 o/w

Source: Ross Girshick

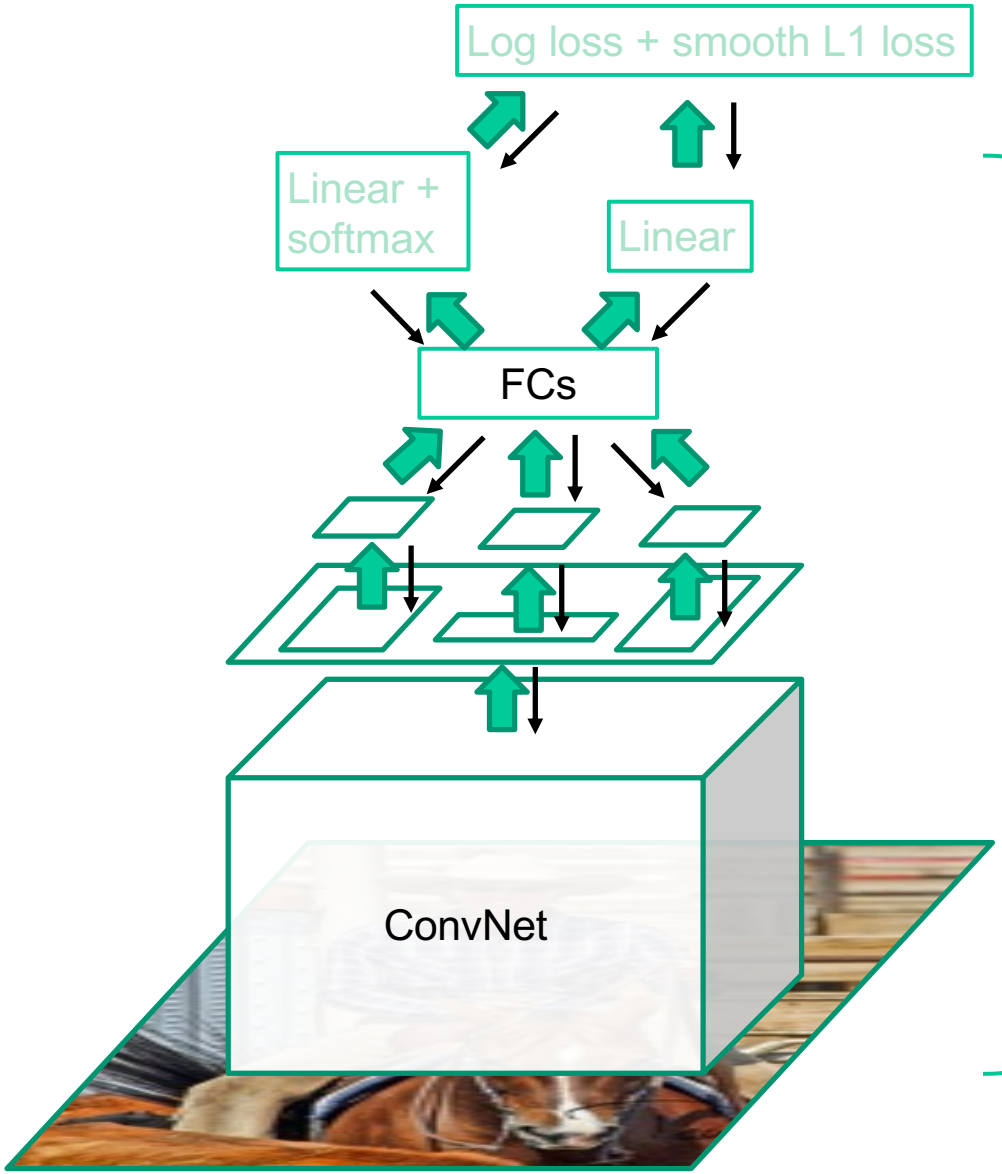
Mini-batch sampling

- Sample a few images (e.g., 2)
- Sample many regions from each image (64)



Source: R. Girshick, K. He

Fast R-CNN training



Multi-task loss

Trainable

Source: R. Girshick

R. Girshick, Fast R-CNN, ICCV 2015

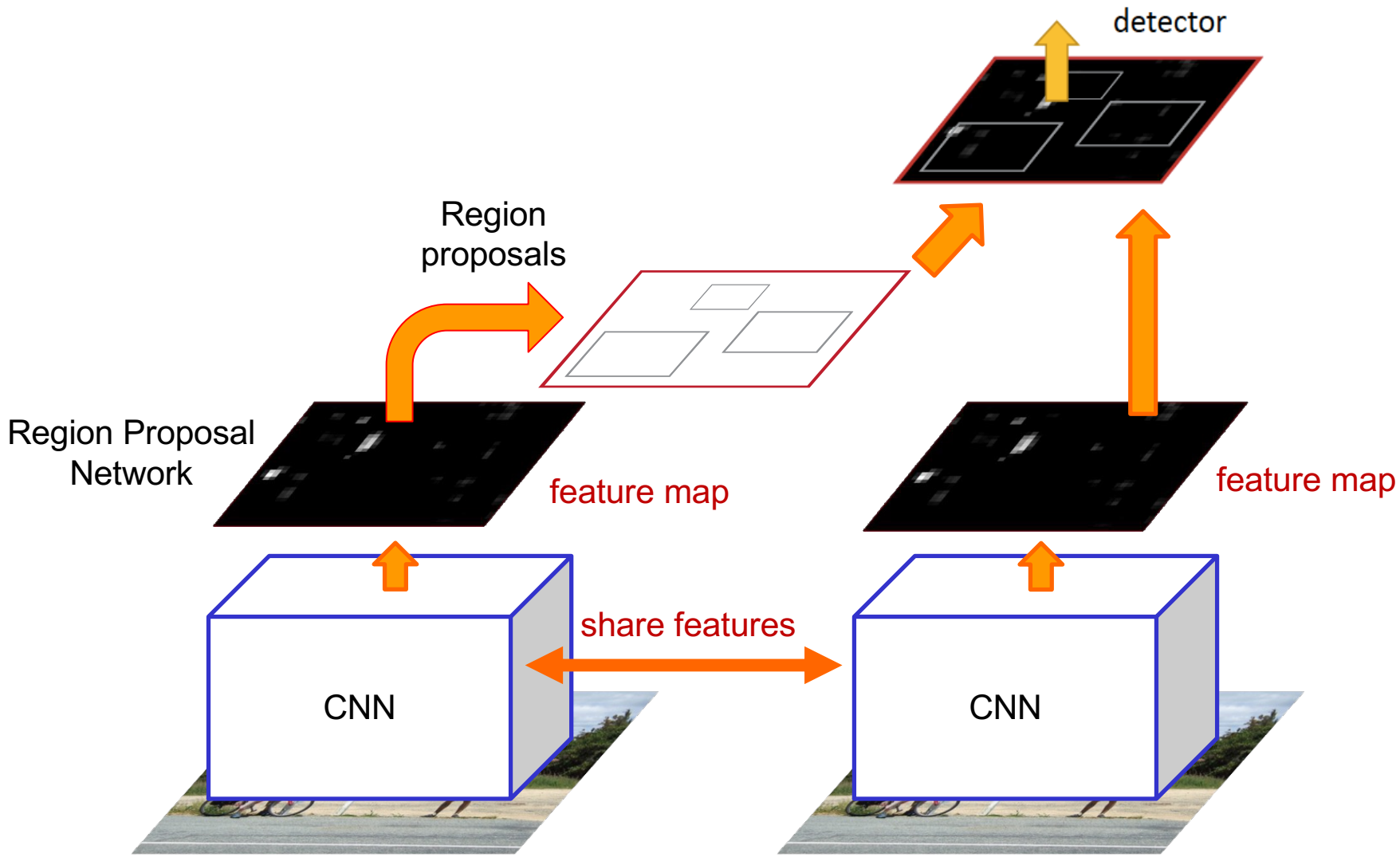
Fast R-CNN results

	Fast R-CNN	R-CNN	
Train time (h)	**9.5**	84	
- Speedup	**8.8x**	1x	
Test time / image	**0.32s**	47.0s	
- Speedup	**146x**	1x	
mAP	**66.9%**	66.0%	(vs. 53.7% for AlexNet)

Timings exclude region proposal time, which is equal for all methods.
All methods use VGG16 from Simonyan and Zisserman.

Source: R. Girshick

Faster R-CNN

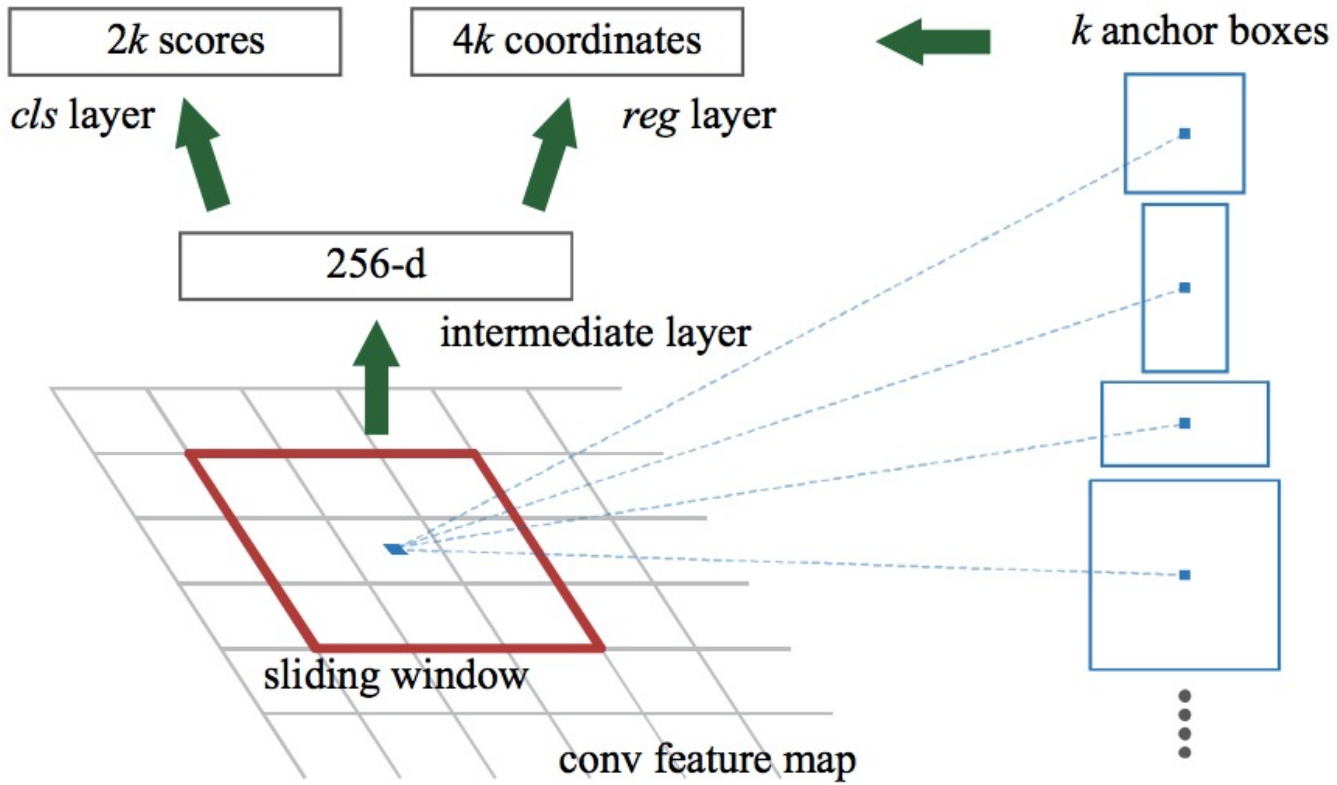


S. Ren, K. He, R. Girshick, and J. Sun

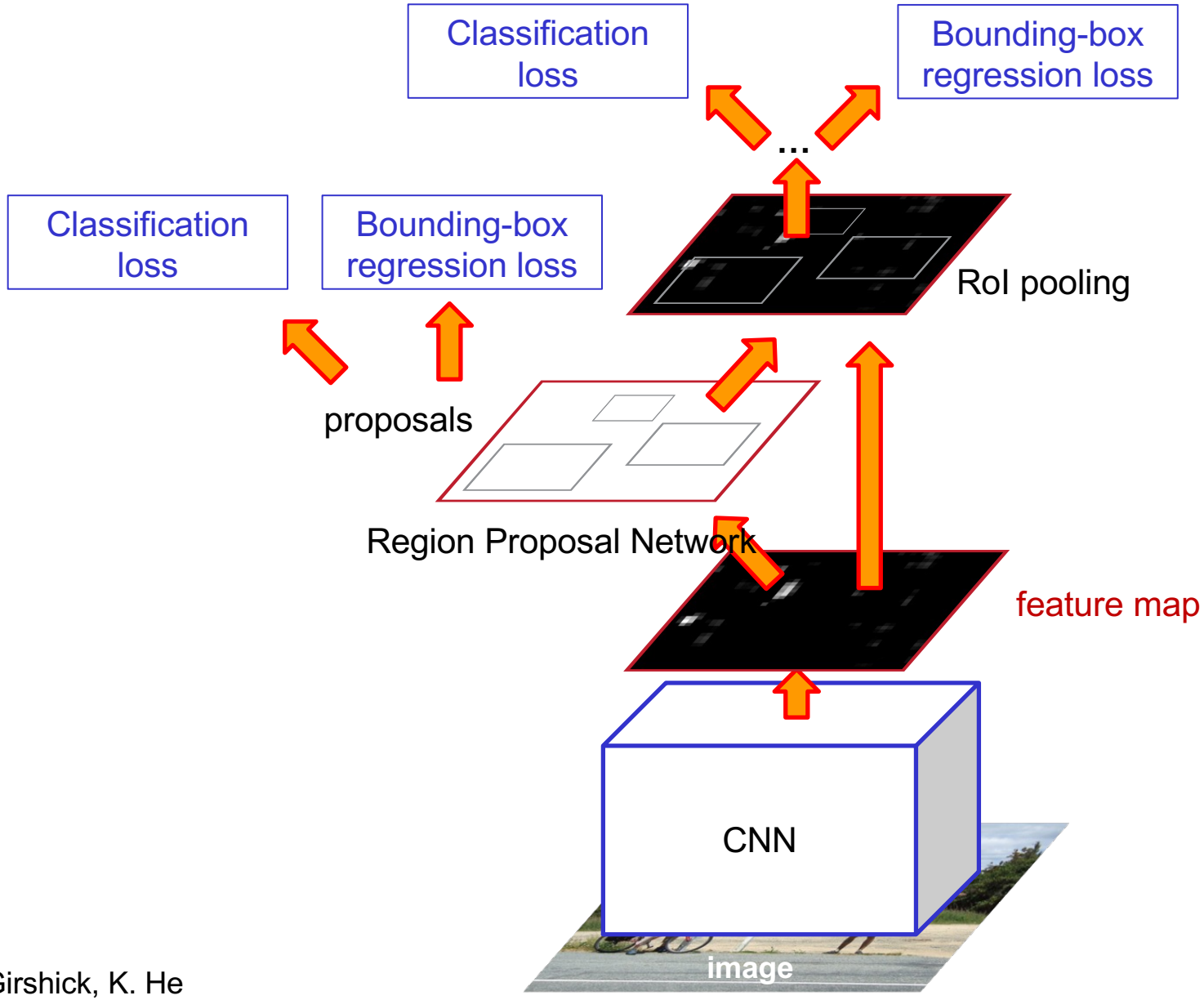
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS 2015

Region proposal network (RPN)

- 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

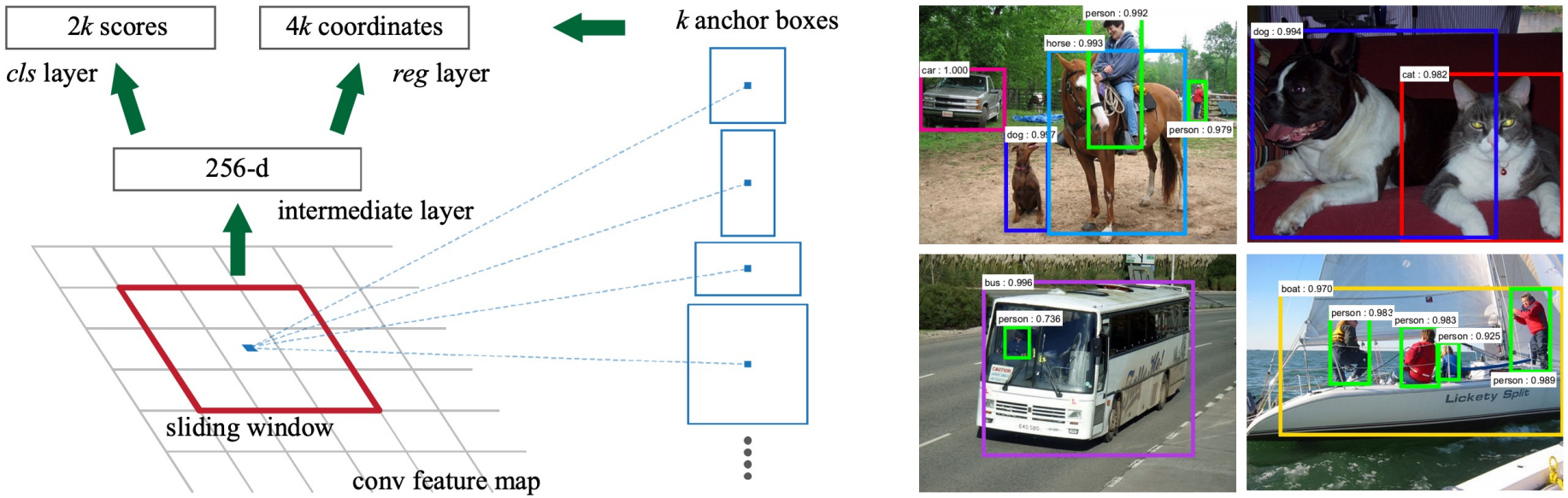


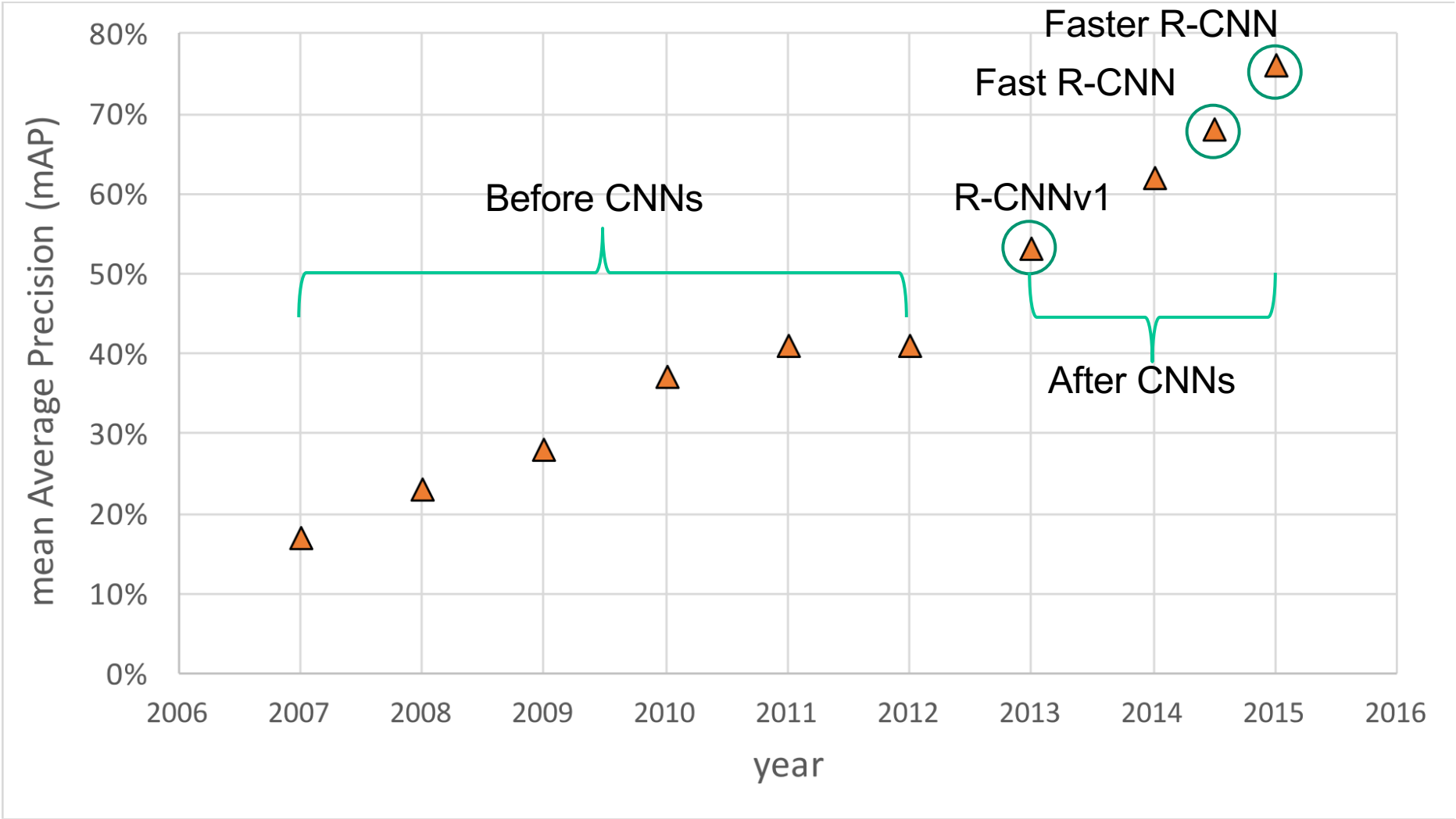
Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

Faster R-CNN results

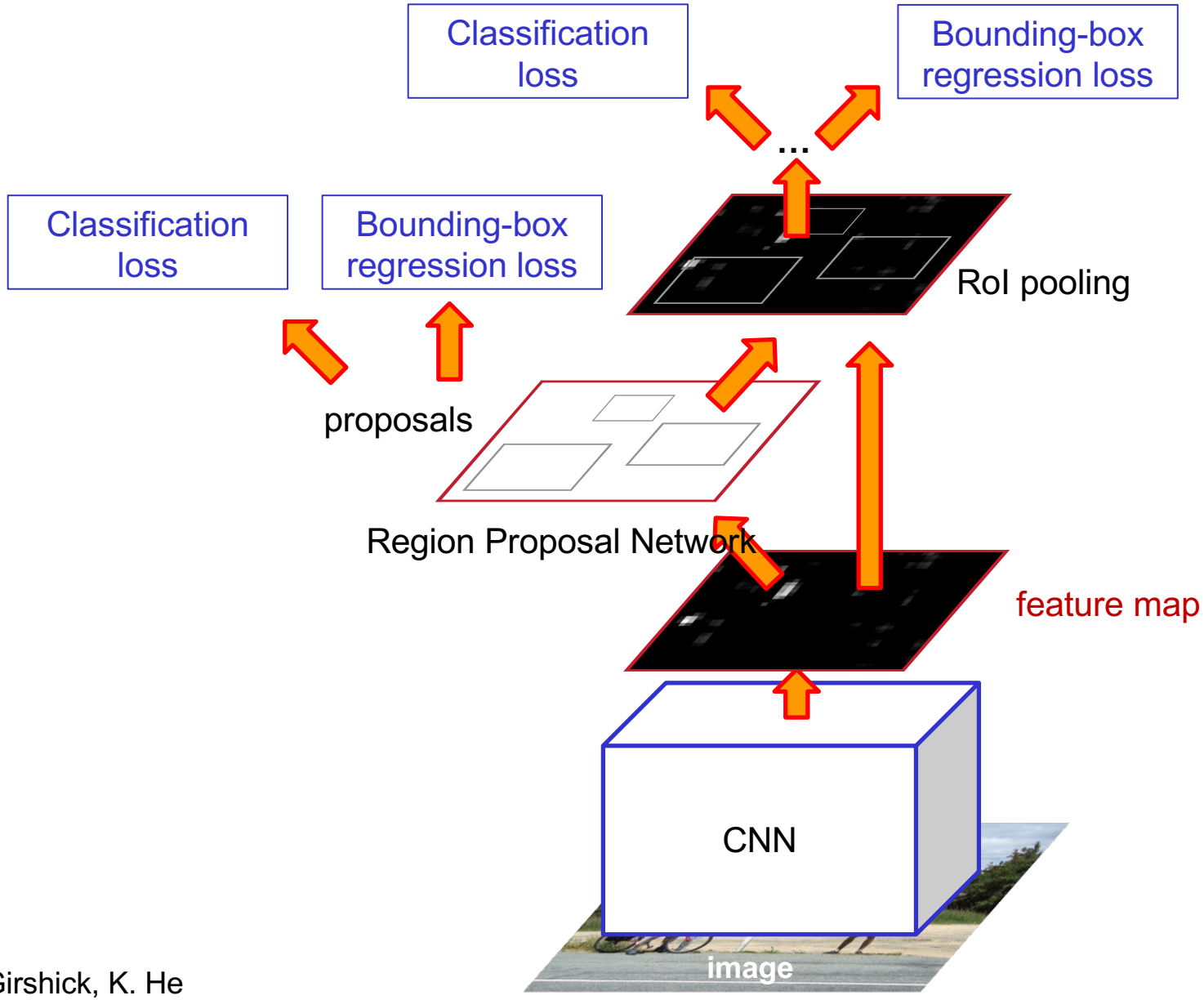
system	time	07 data	07+12 data
R-CNN	~50s	66.0	-
Fast R-CNN	~2s	66.9	70.0
Faster R-CNN	198ms	**69.9**	**73.2**

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

Object detection progress



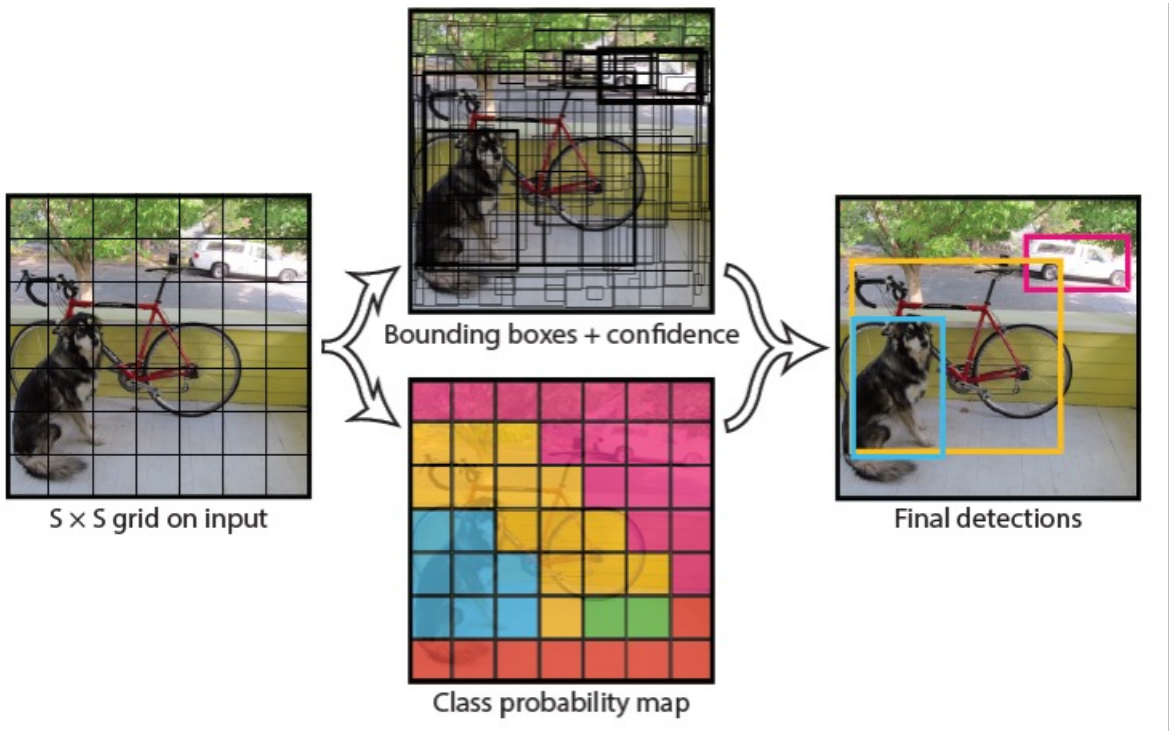
What is strange in Faster RCNN?



Source: R. Girshick, K. He

YOLO

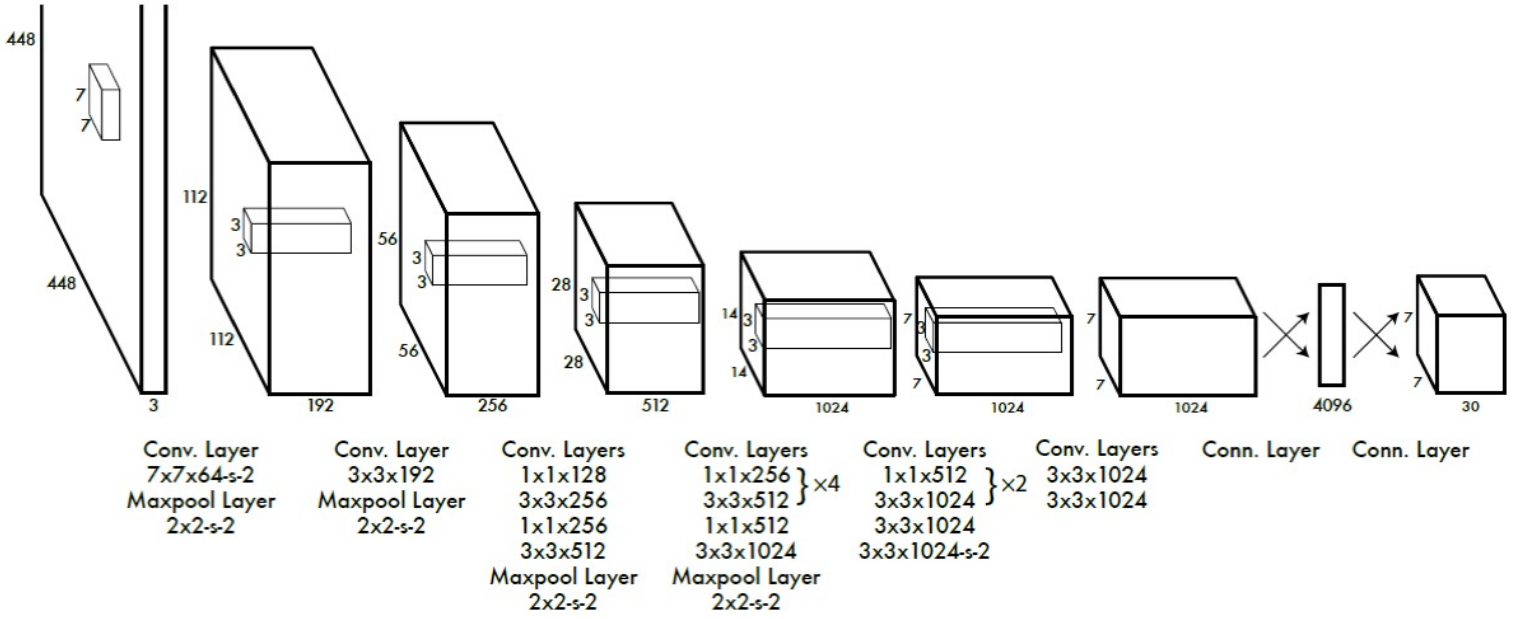
- Divide the image into a coarse grid and directly predict class label and a few candidate boxes for each grid cell



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 7x7x30 (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:

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

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

regression

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

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

[no] object confidence

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

class prediction

YOLO

- Objective function:

Cell "I" contains object, predictor "j" is responsible for it

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

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

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

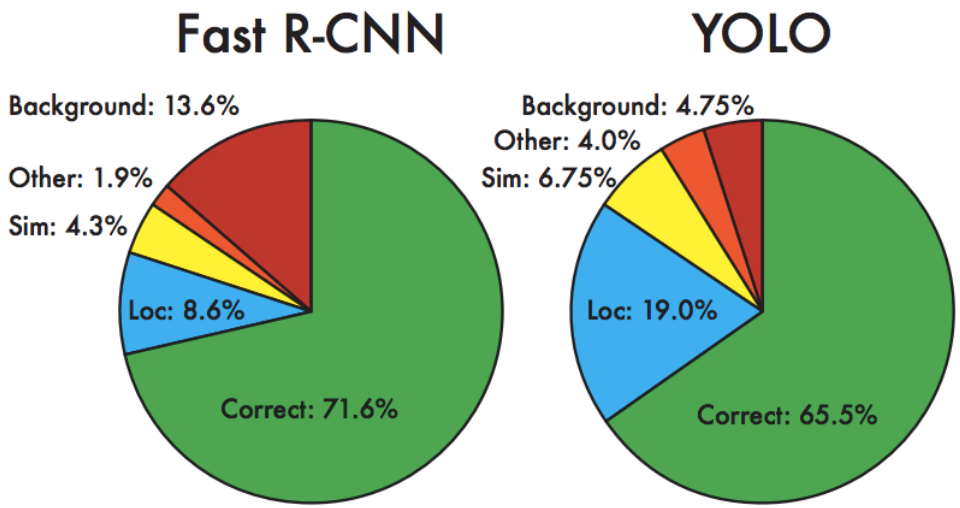
Down-weight loss from boxes that don't contain objects

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

Class probability

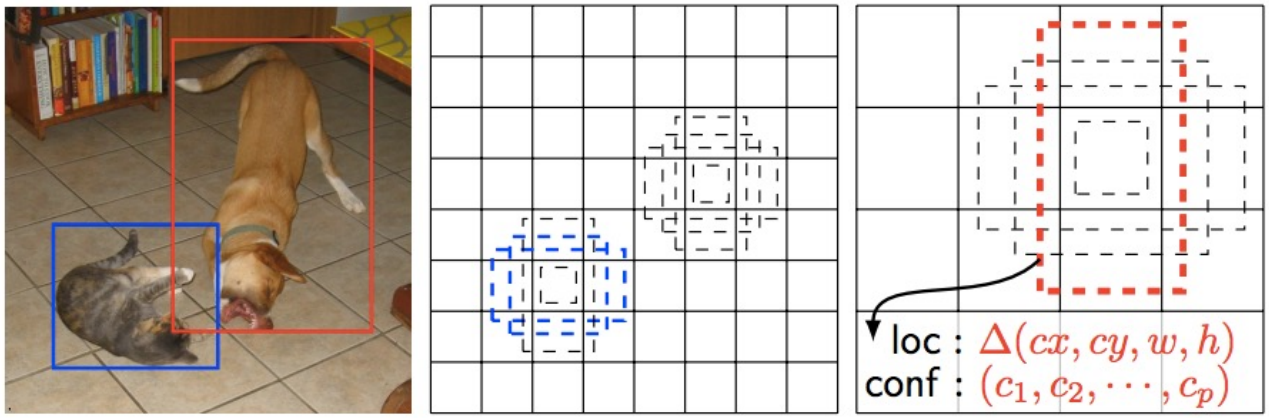
YOLO: results

- Each cell predicts only two boxes and can have only 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

SSD

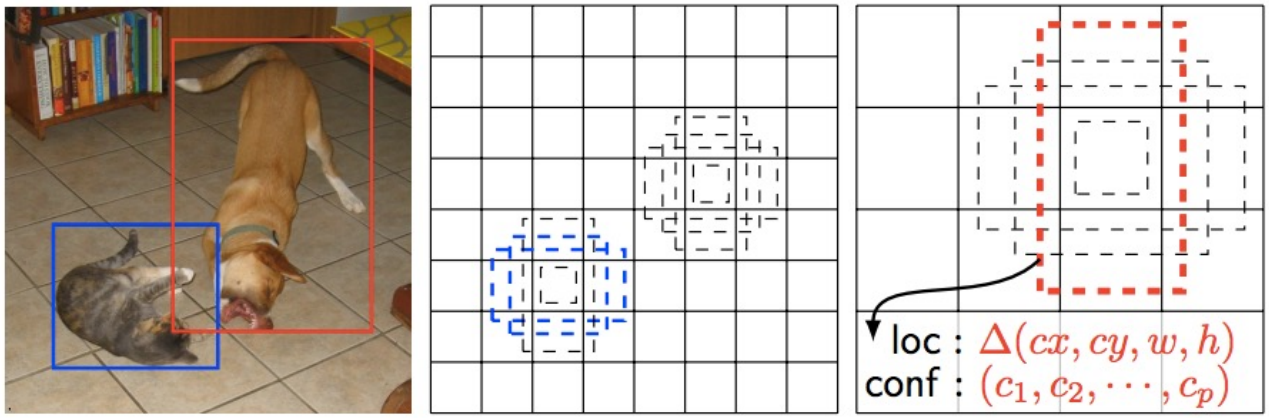


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

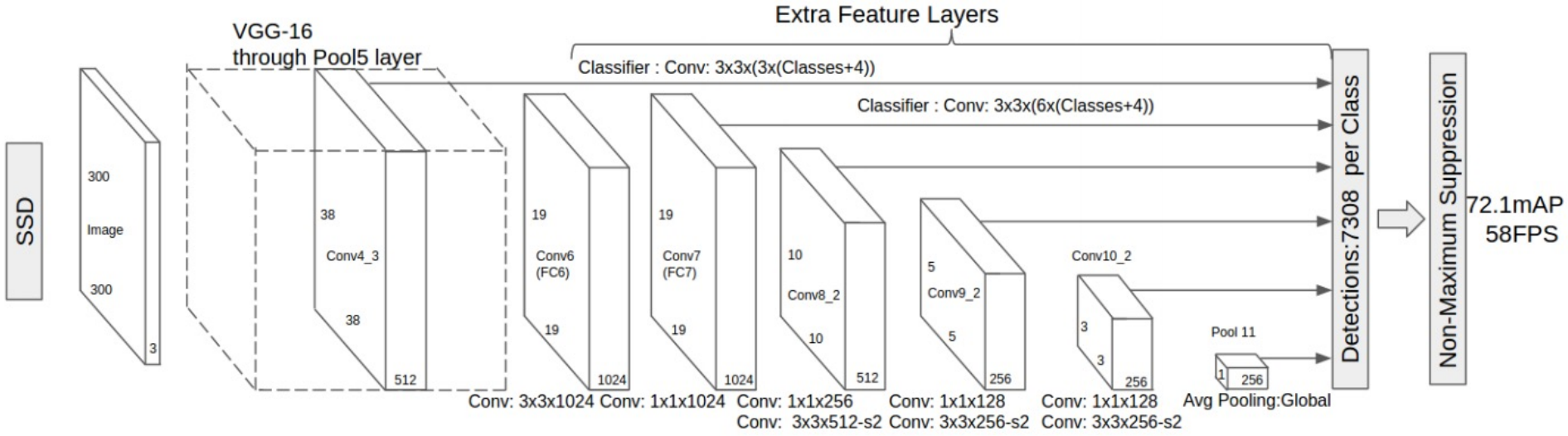
- Similarly to YOLO, predict bounding boxes directly from conv maps (i.e. end-to-end)
- Unlike YOLO, do not use FC layers and predict different size boxes from conv maps at different resolutions (i.e. multi-scale)
- Similarly to RPN, use anchors (i.e. more predictions)

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

SSD



(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.

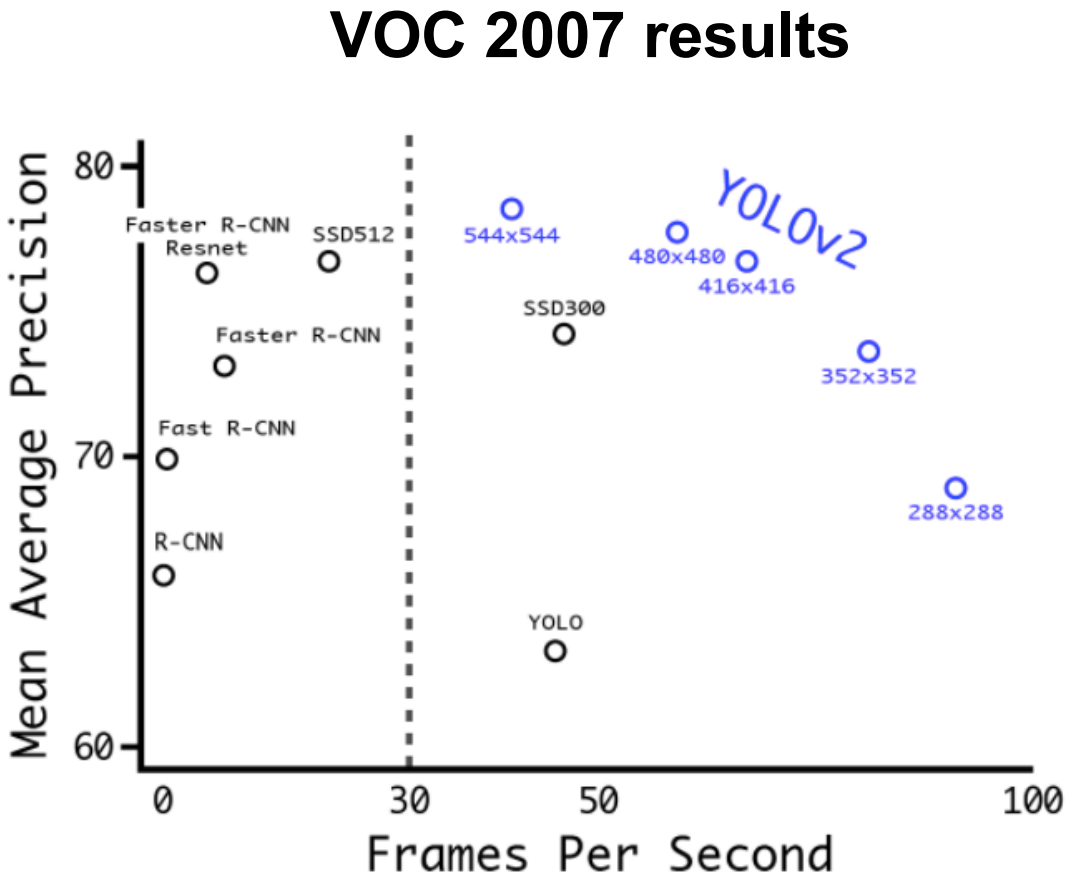
SSD: Results (PASCAL 2007)

- More accurate than Faster R-CNN
- *Faster* than YOLO

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

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-norm and other tricks



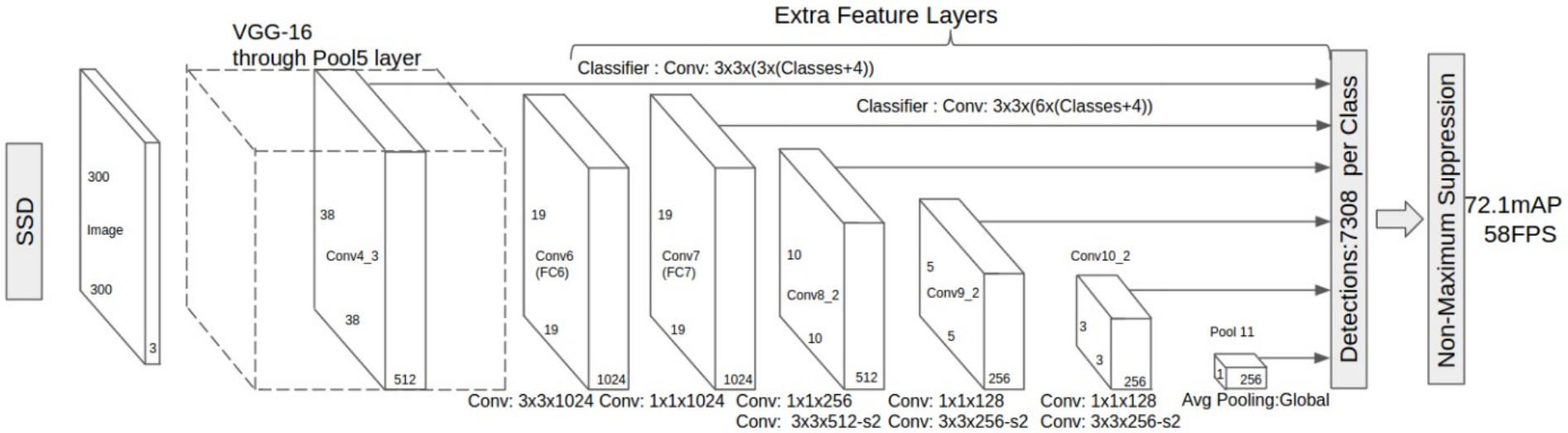
YouTube demo

J. Redmon and A. Farhadi, YOLO9000: Better, Faster, Stronger, CVPR 2017


YOLO v2
http://pjreddie.com/yolo

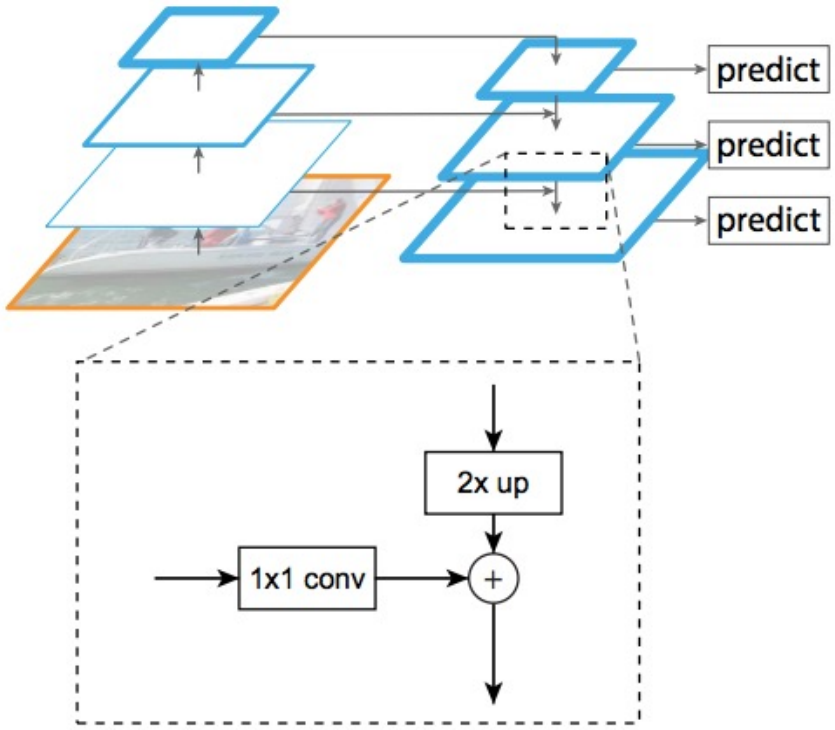
Multi-resolution prediction

- SSD predicts boxes of different size from different conv maps, but each level of resolution has its own predictors and higher-level context does not get propagated back to lower-level feature maps
- Can we have a more elegant multi-resolution prediction architecture?



Feature pyramid networks

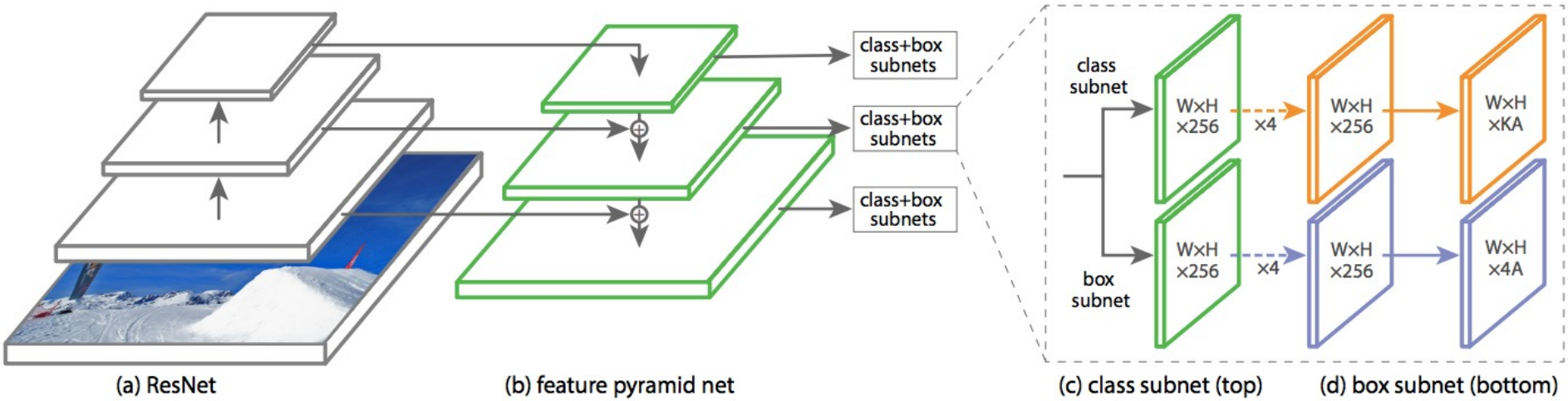
- Improve predictive power of lower-level feature maps by adding contextual information from higher-level feature maps
- Predict different sizes of bounding boxes from different levels of the pyramid (but share parameters of predictors)



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

RetinaNet

- Combine feature pyramid network with focal loss to reduce 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.

Focal Loss (RetinaNet)

The *Focal Loss* is designed to address the one-stage object detection scenario in which there is an extreme imbalance between foreground and background classes during training (*e.g.*, 1:1000). We introduce the focal loss starting from the cross entropy (CE) loss for binary classification[1]:

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases} \tag{1}$$

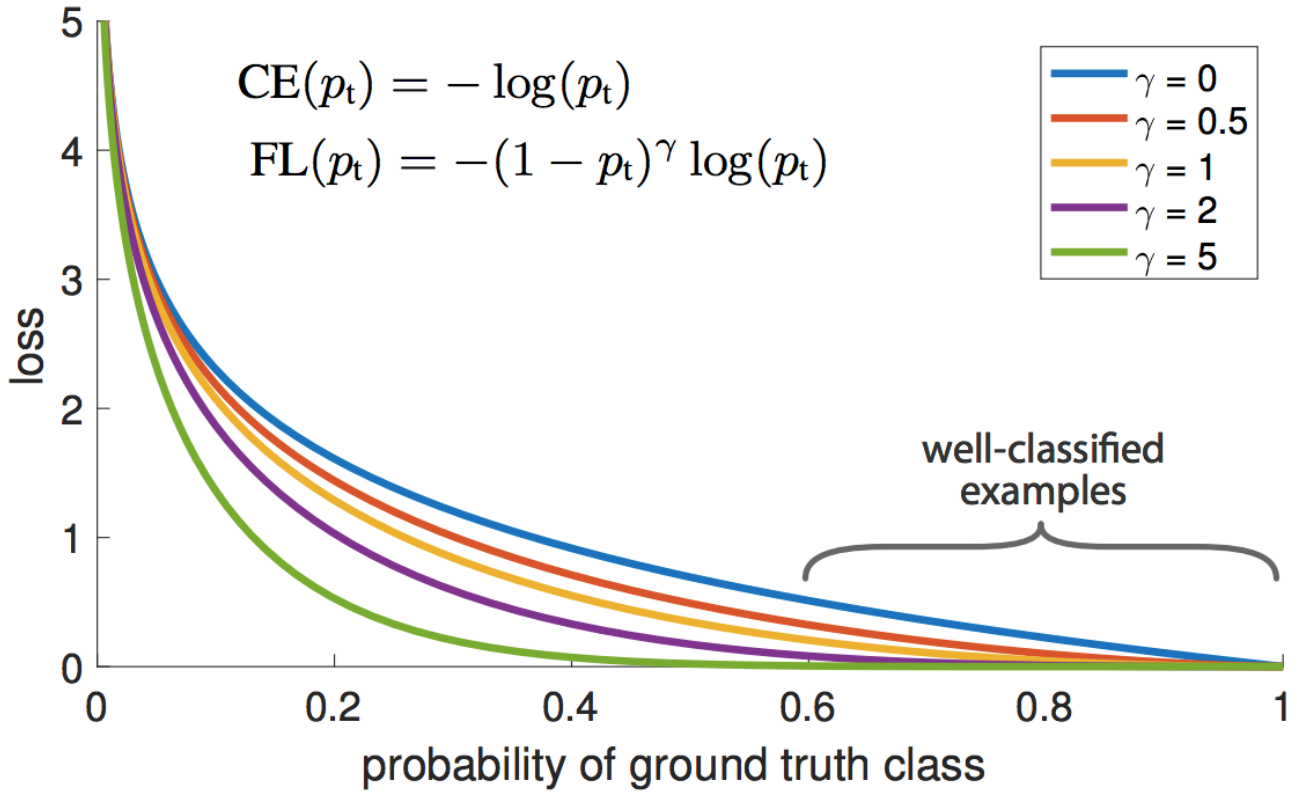
In the above $y \in \{\pm 1\}$ specifies the ground-truth class and $p \in [0, 1]$ is the model's estimated probability for the class with label $y = 1$. For notational convenience, we define p_t:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases} \tag{2}$$

and rewrite $\mathrm{CE}(p, y) = \mathrm{CE}(p_t) = -\log(p_t)$.

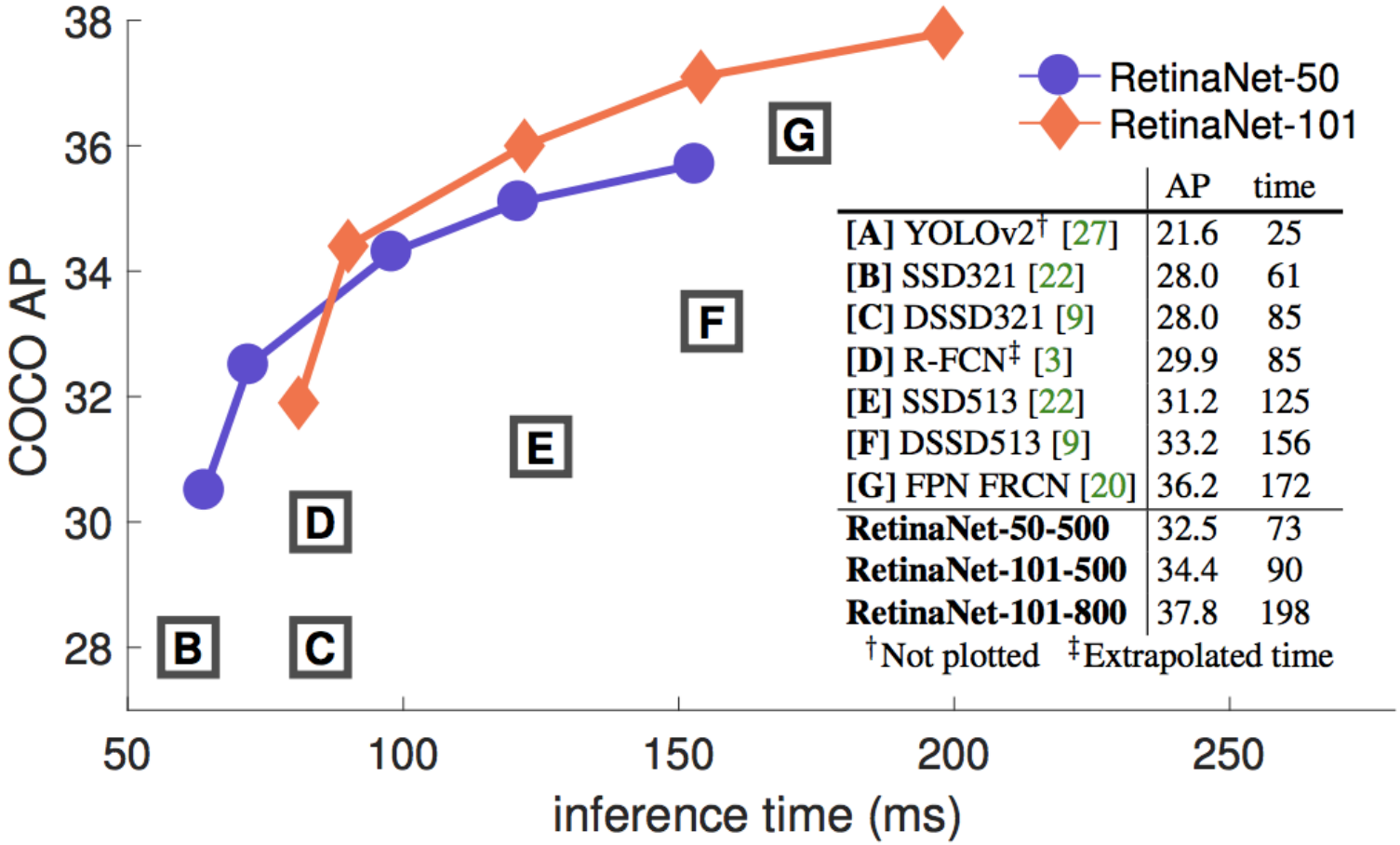
T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar
Focal loss for dense object detection, ICCV 2017.

Focal Loss (RetinaNet)



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

RetinaNet: Results



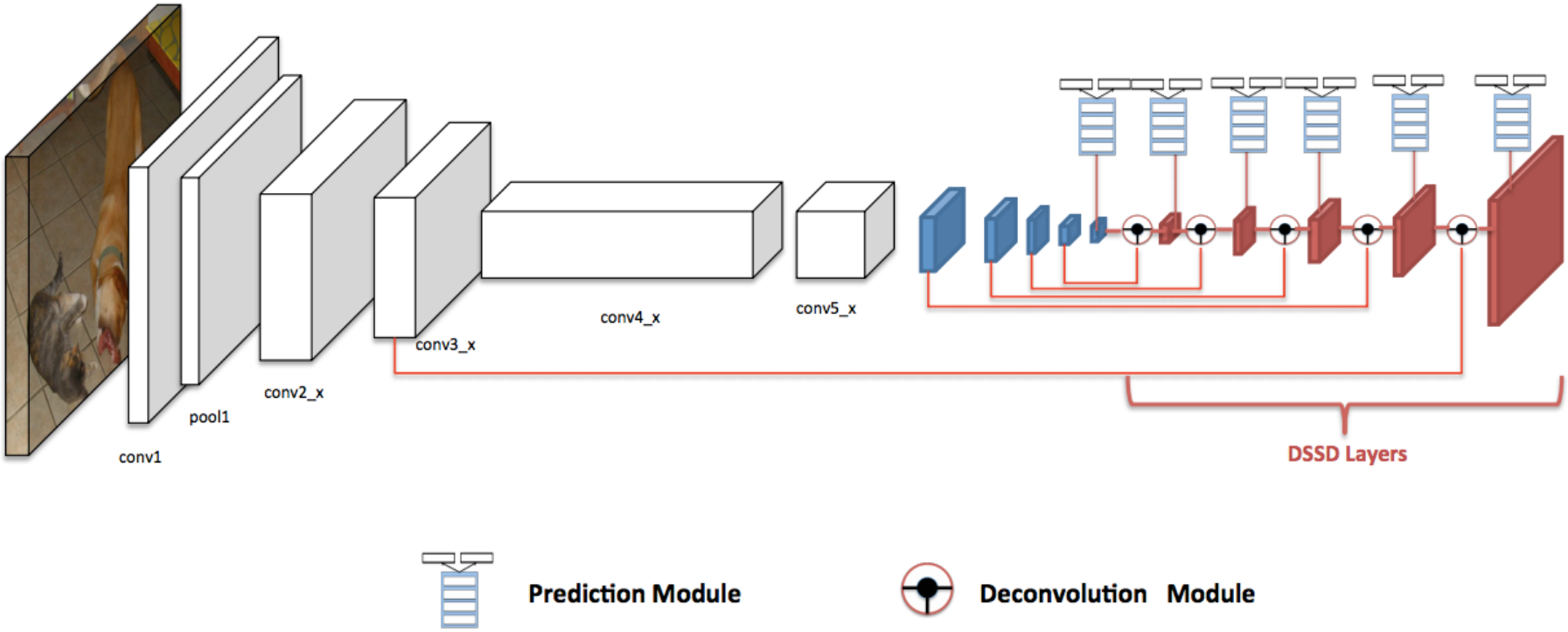
	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

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

Deconvolutional SSD

- Improve performance of SSD by increasing resolution through learned “deconvolutional” layers



C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, A. Berg
DSSD: Deconvolutional single-shot detector, arXiv 2017.

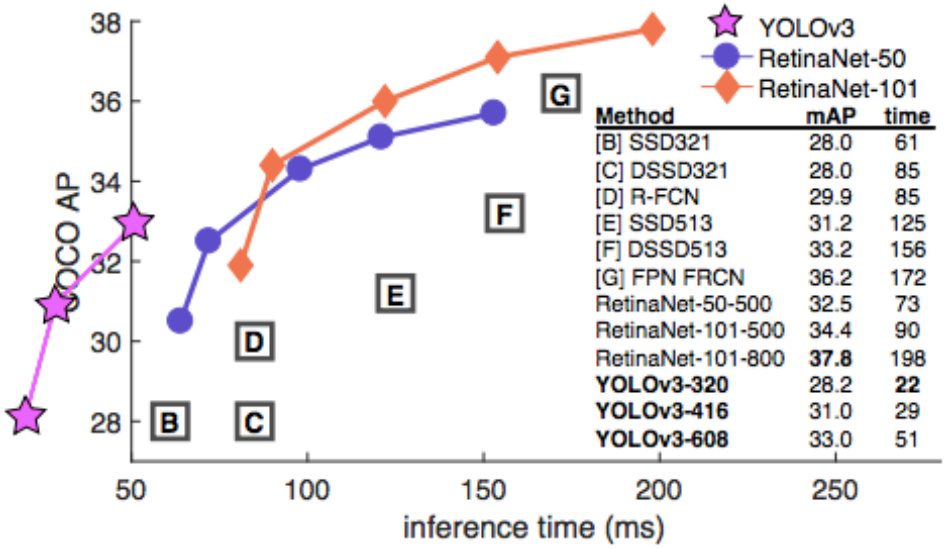
YOLO v3

YOLOv3: An Incremental Improvement

Joseph Redmon, Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

1. Introduction

https://pjreddie.com/media/files/papers/YOLOv3.pdf

Summary: Object detection with CNNs

- R-CNN: region proposals + CNN on cropped, resampled regions
- Fast R-CNN: region proposals + RoI pooling on top of a conv feature map
- Faster R-CNN: RPN + RoI pooling
- Next generation of detectors
  - Direct prediction of BB offsets, class scores on top of conv feature maps
  - Get better context by combining feature maps at multiple resolutions