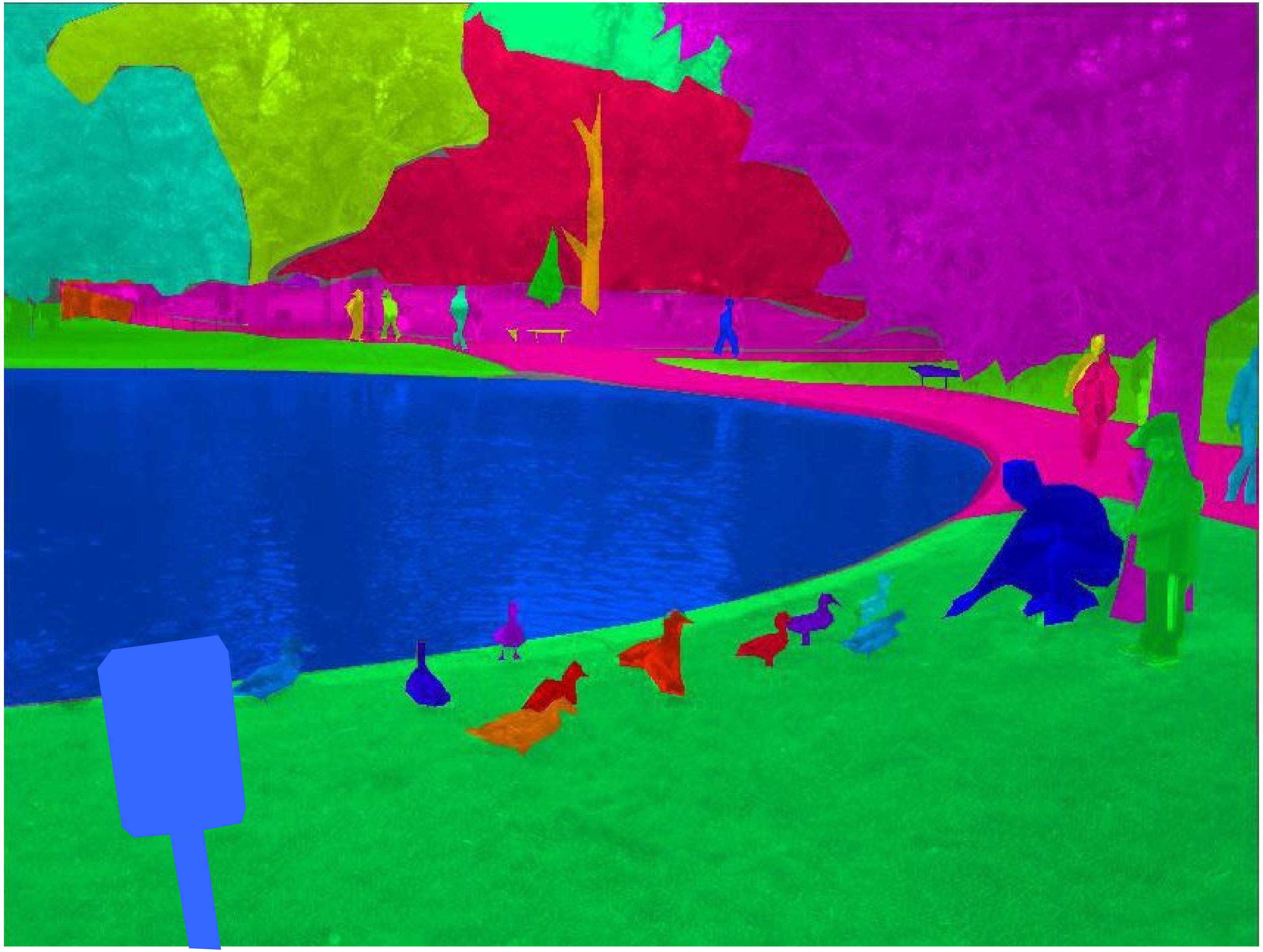
Lecture 11

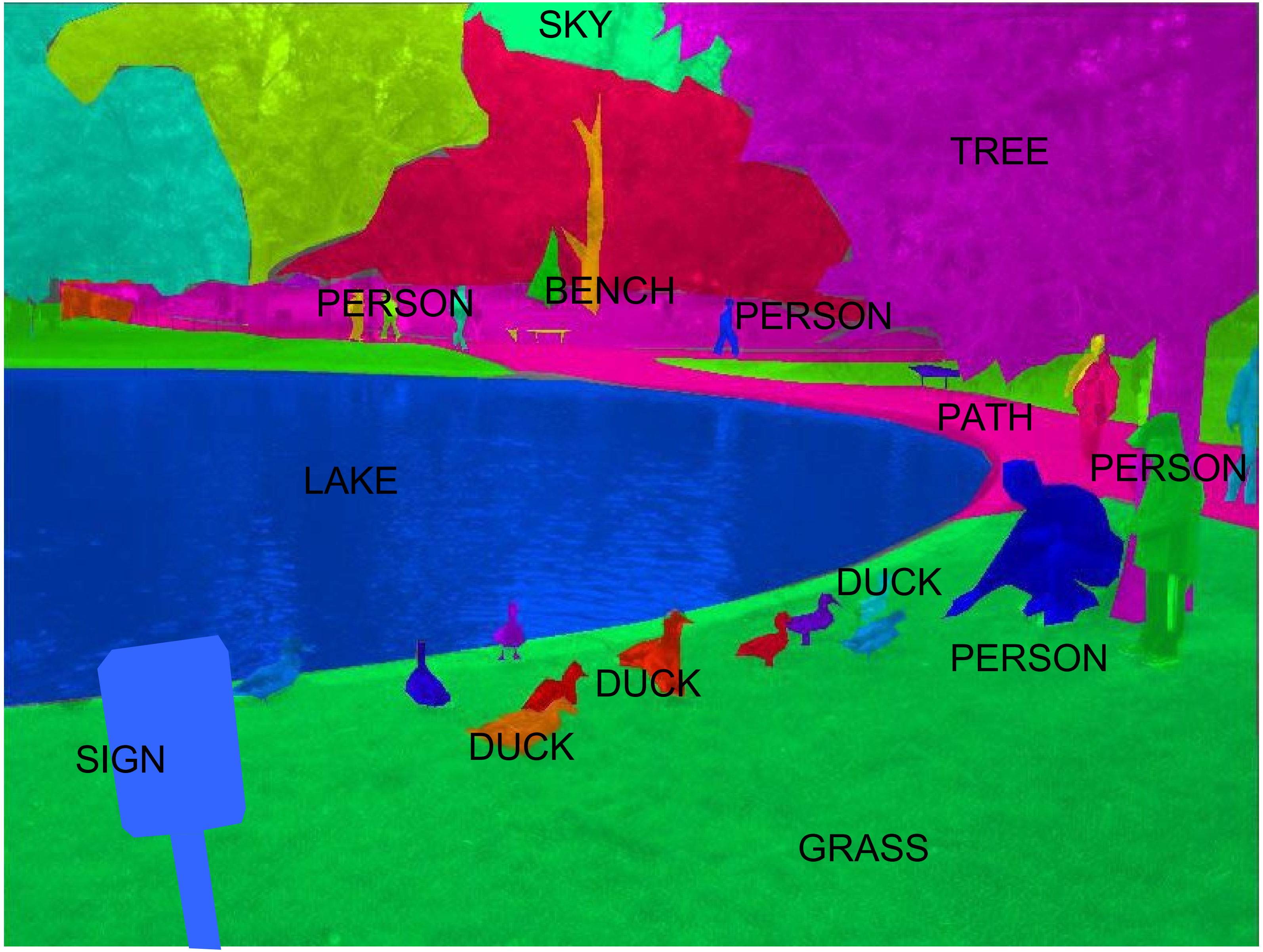
Visual Recognition

Image contains Photoshopped sign

Source: Antonio Torralba



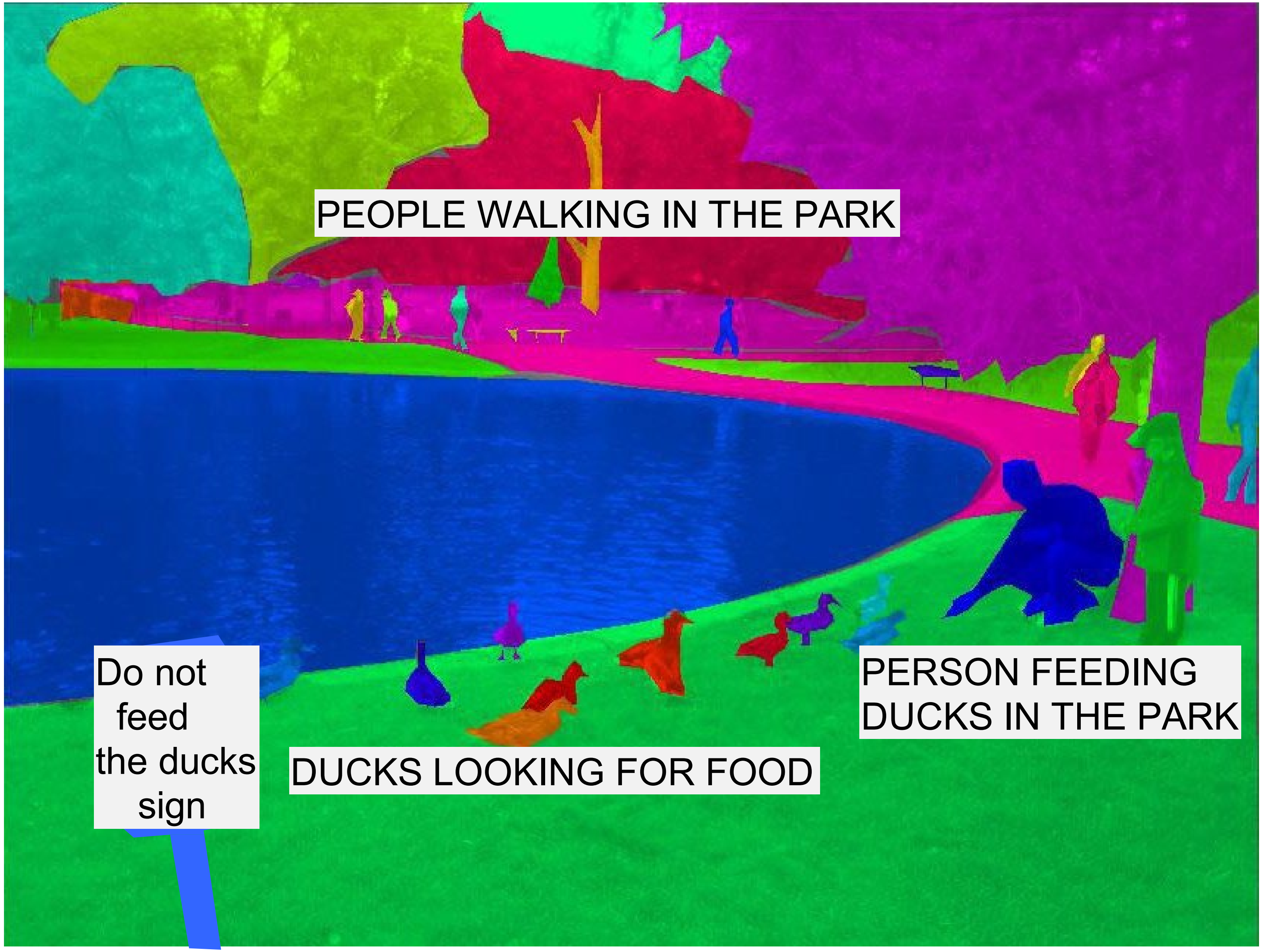
Label each pixel as a category. Each category has a unique color.



Label each pixel as a category. Each category has a unique color.

Scene-Level Classification: This is a “PARK”

Image Captioning: Describe the image in human language (e.g. English)



Dense Image Captioning: Describe several parts of the image

Why do we care about recognition?

- The concept of "**categories**" encapsulates semantic information that humans use when communicating with each other.

- Categories are also linked with what can we do with those objects.

Object categories aren't everything

A picture is worth a 1000 words…

Or just 10?

sky

building

flag

face

banner

wall

street lamp

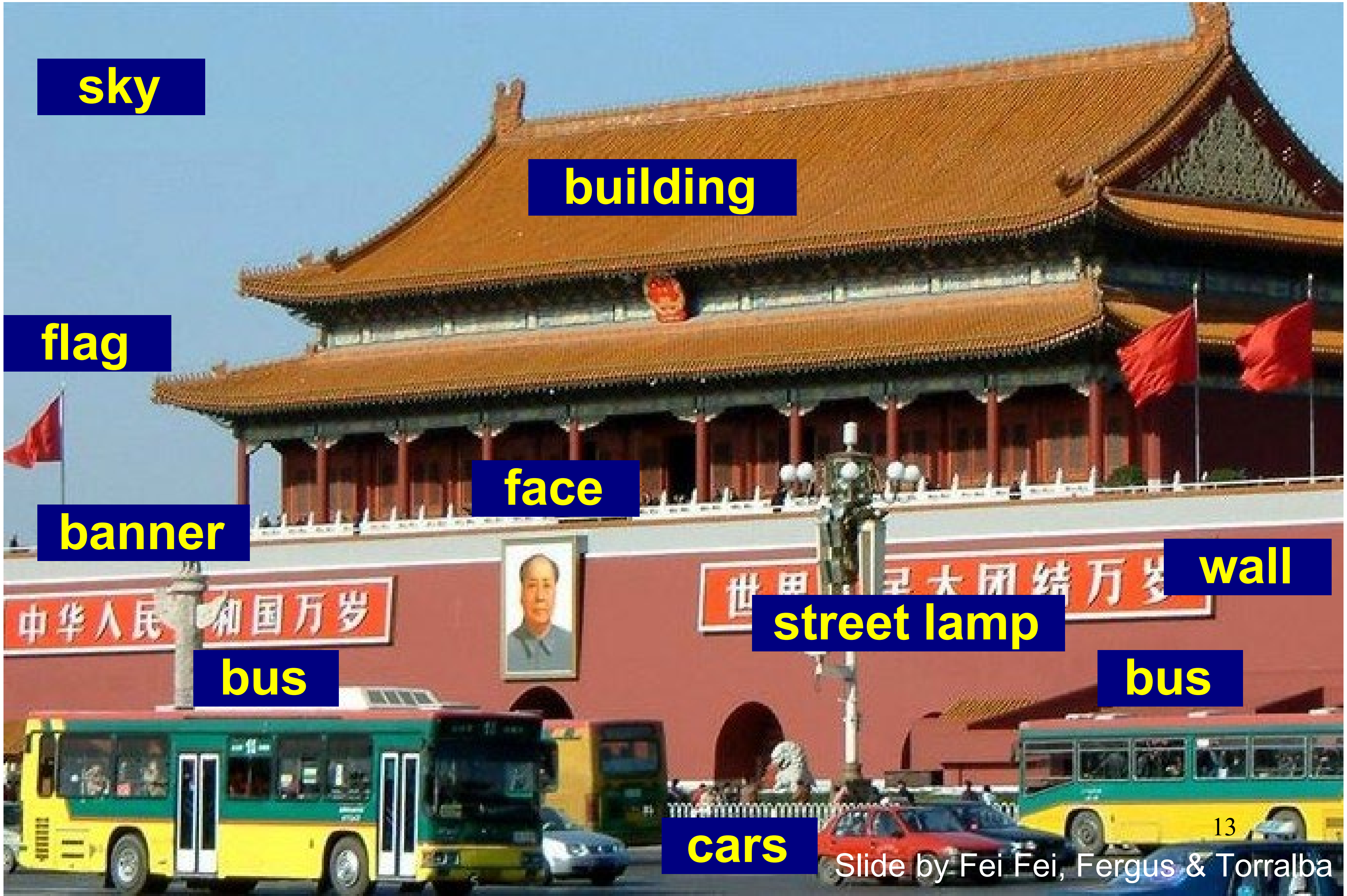
bus

bus

cars

Source: A. Efros

Object categories aren’t everything



How finegrained should categories be ?

- Bus
- Yellow & Green Bus
- …
- …
- "Beijing City Transit Bus #17, serial number 43253"

Source: A. Efros

Need more general (useful) information

What can we say the very first time we see this thing?

Functional:

- A large vehicle that may be moving fast
- Will hurt you if you stand in its way.
- However, at specified places, it will allow you to enter it and transport you quickly over large distances.

Communicational:

- bus, autobus, λεωφορείο, ônibus, автобус, 公共汽车, etc.

Source: A. Efros

What makes this challenging?

Visual challenges with categories

- A lot of categories are functional

Chair

- Categories are 3D, but images are 2D

car

- World is highly varied

train

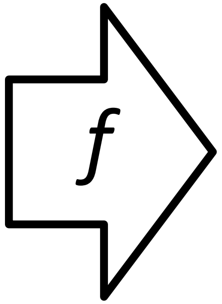
Source: A. Efros

Limits to direct perception

Today

- Introduction to scene understanding
- **Object detection models**
- Evaluating object detectors
- Future challenges

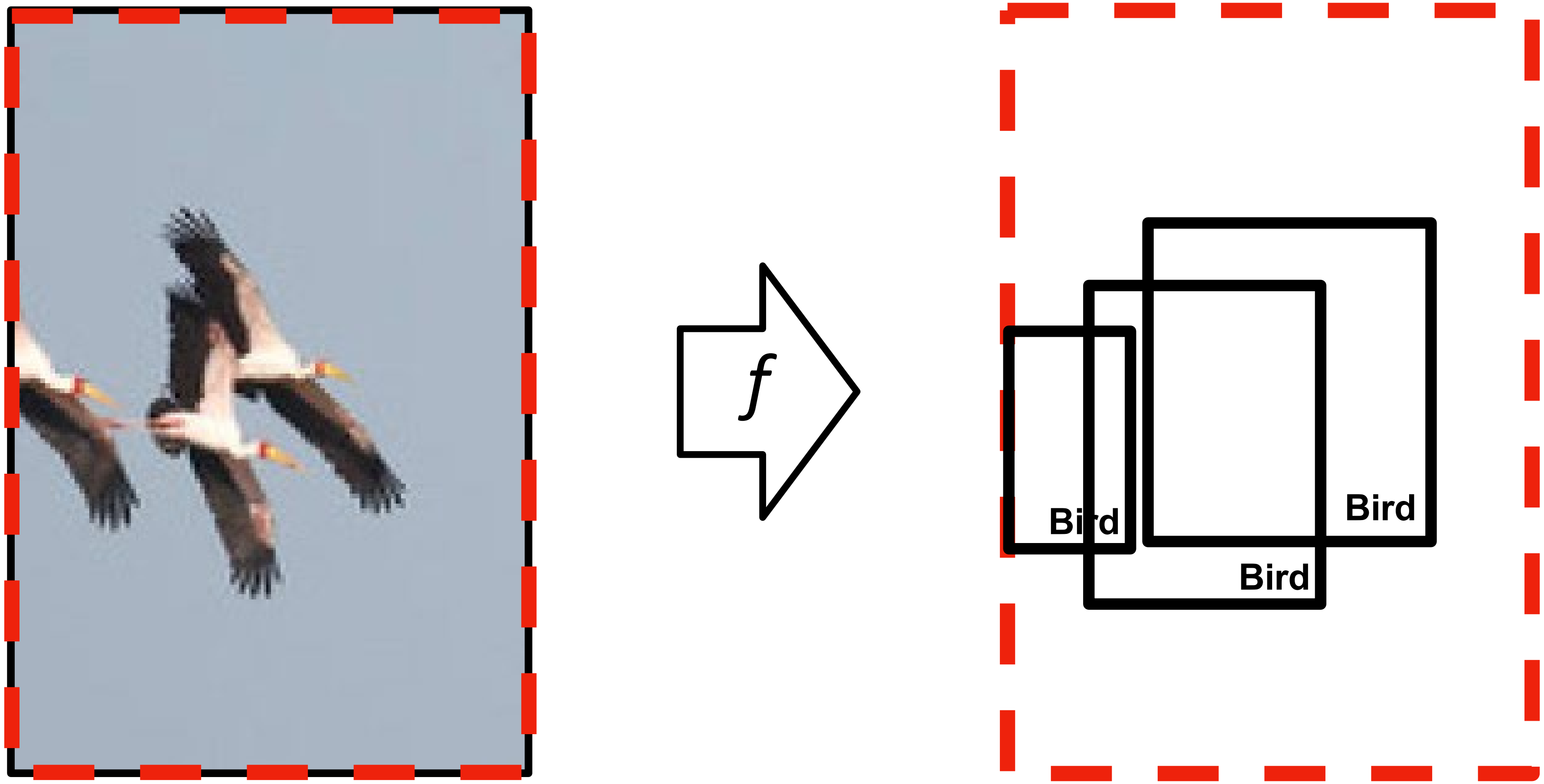
Image Classification



"Birds"

Object detection

Classification and localization

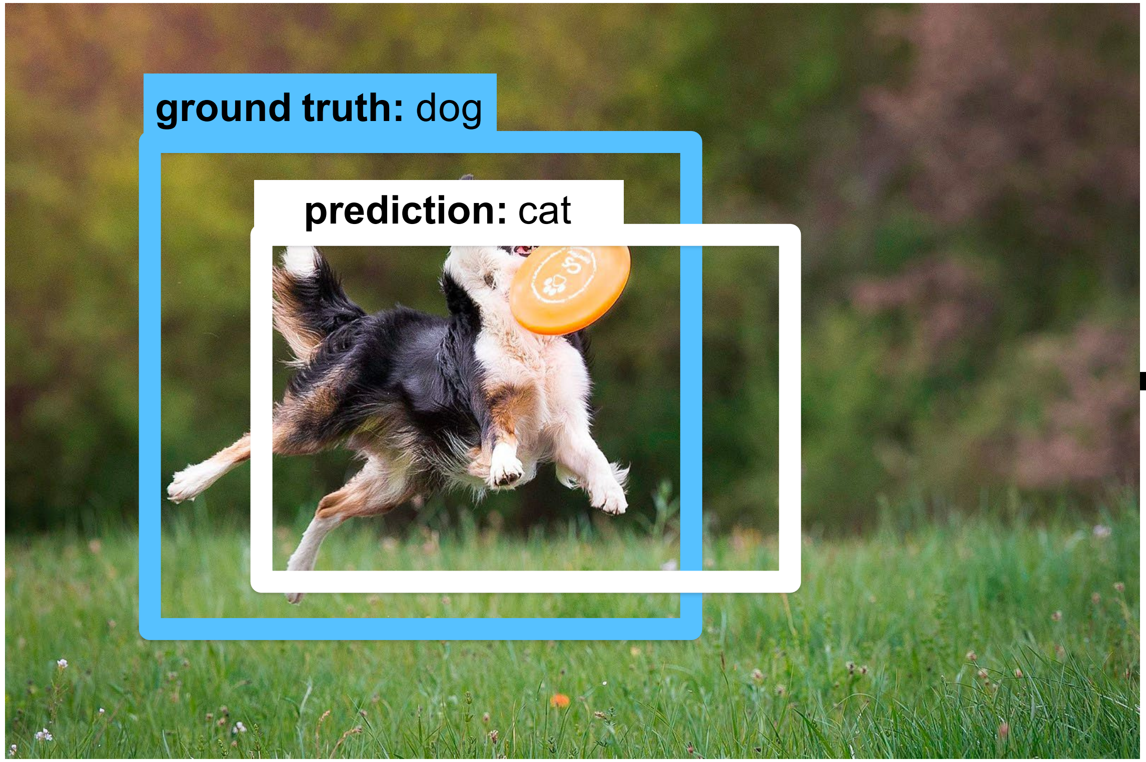


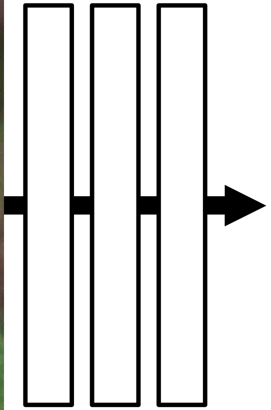
Each bounding box is:
[x,y,w,h]

Challenge: unbounded number of detections, possibly multiple detections per pixel

Idea #1: regress bounding box

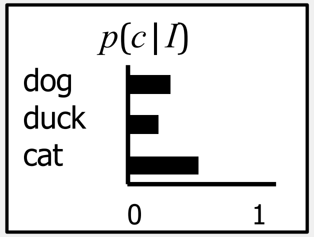
Idea #1: regress bounding box





Outputs

1. Class label



2. Box coords.

$$(x, y, w, h)$$

Losses

1. Cross entropy loss

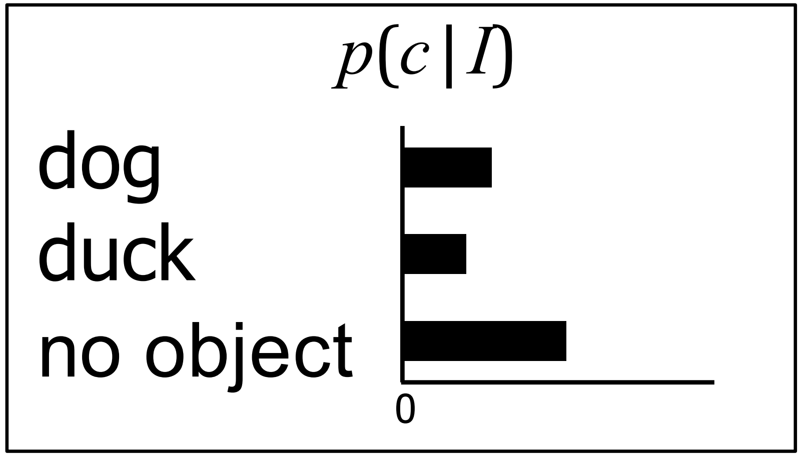
$$L_{cls} = -\log(p(y = \text{dog}))$$

2. Squared distance

$$L_{box} = \left\| \begin{bmatrix} x \\ y \\ w \\ h \end{bmatrix} - \begin{bmatrix} x_{gt} \\ y_{gt} \\ w_{gt} \\ h_{gt} \end{bmatrix} \right\|^2$$

Doesn't scale well to multiple objects.

Idea #2: sliding window

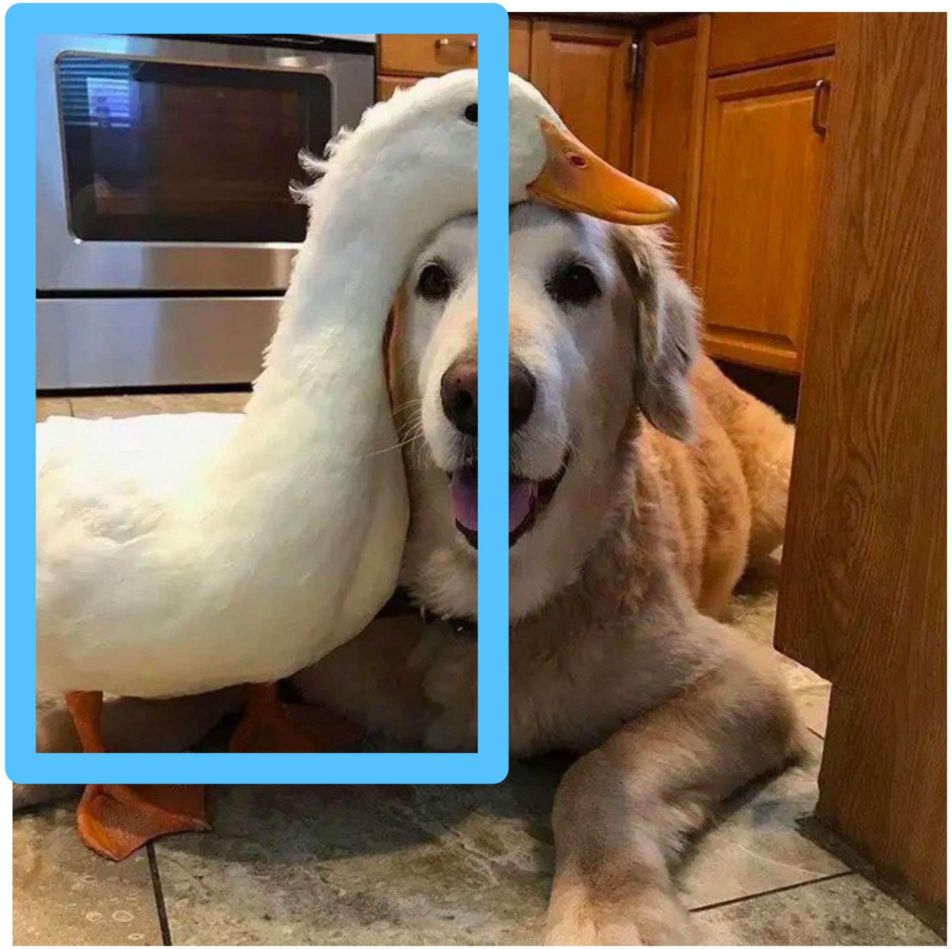


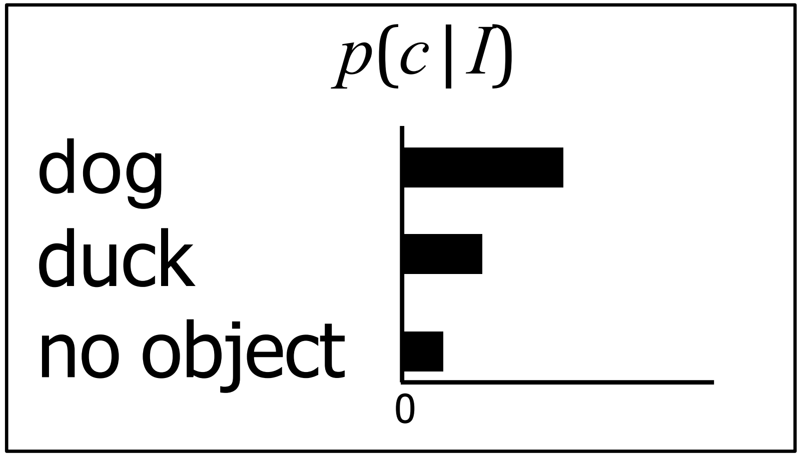
Bounding box

(x, y, w, h)

Need multiple scales and aspect ratios

Idea #2: sliding window

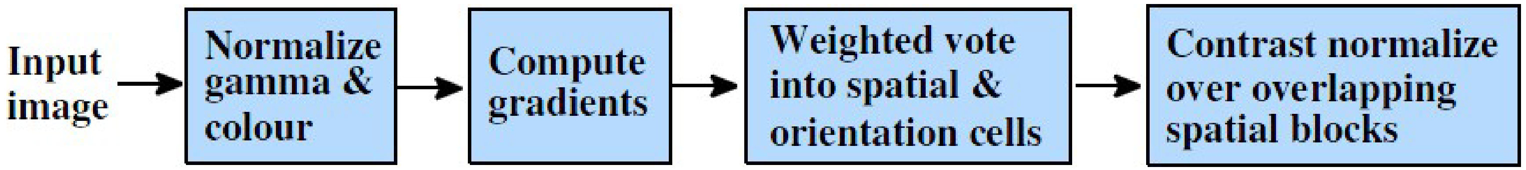




Bounding box

(x, y, w, h)

Another Sliding Window Approach:
Histograms of oriented gradients (HOG)



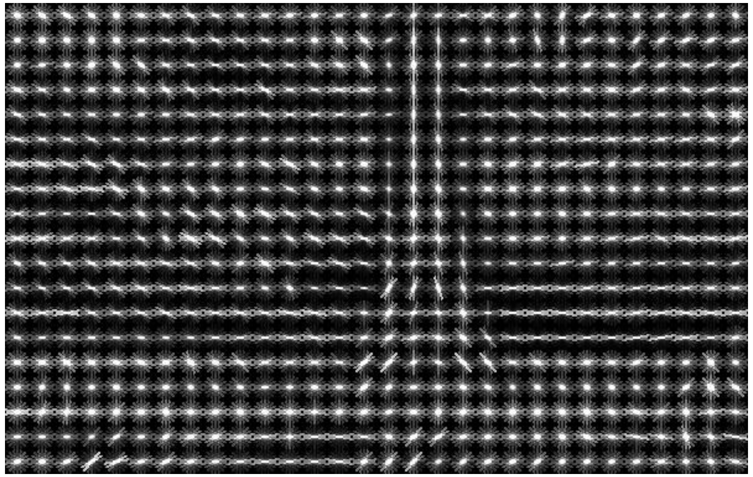


Image credit: N. Snavely

N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Source: S. Lazebnik

Example: pedestrian detection with HOG

Train a pedestrian template using a linear classifier.
Represent each window using HOG

positive training examples

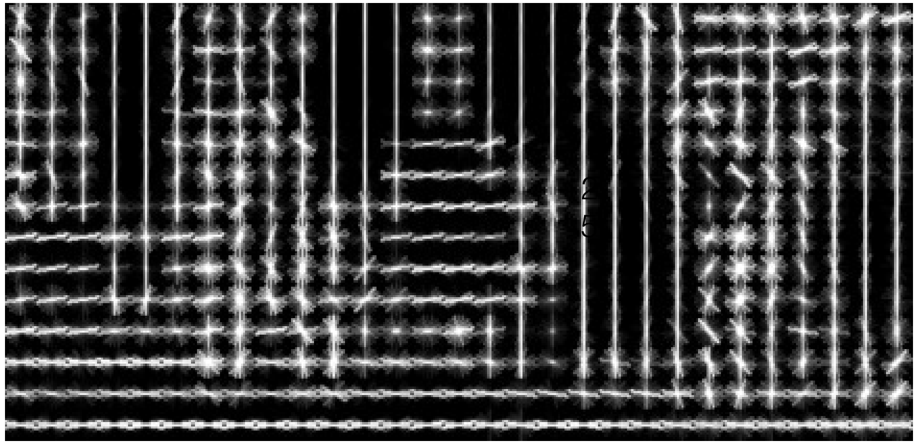
negative training examples

N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

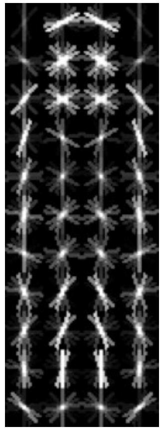
Example: pedestrian detection with HOG

For multi-scale detection, repeat over multiple levels of a HOG pyramid

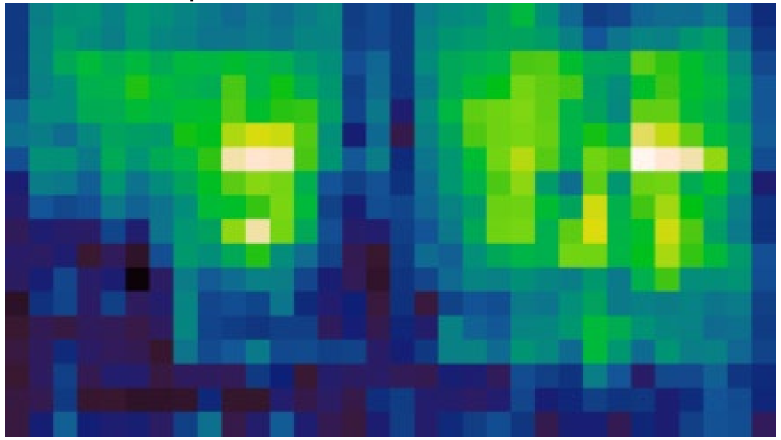
HOG feature map



Template

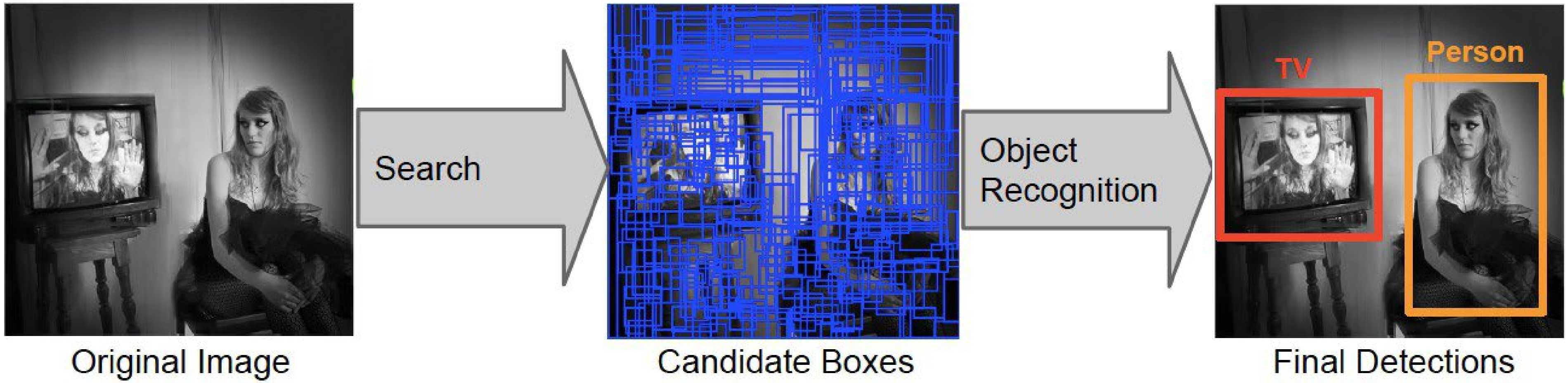


Detector response map



Source: S. Lazebnik

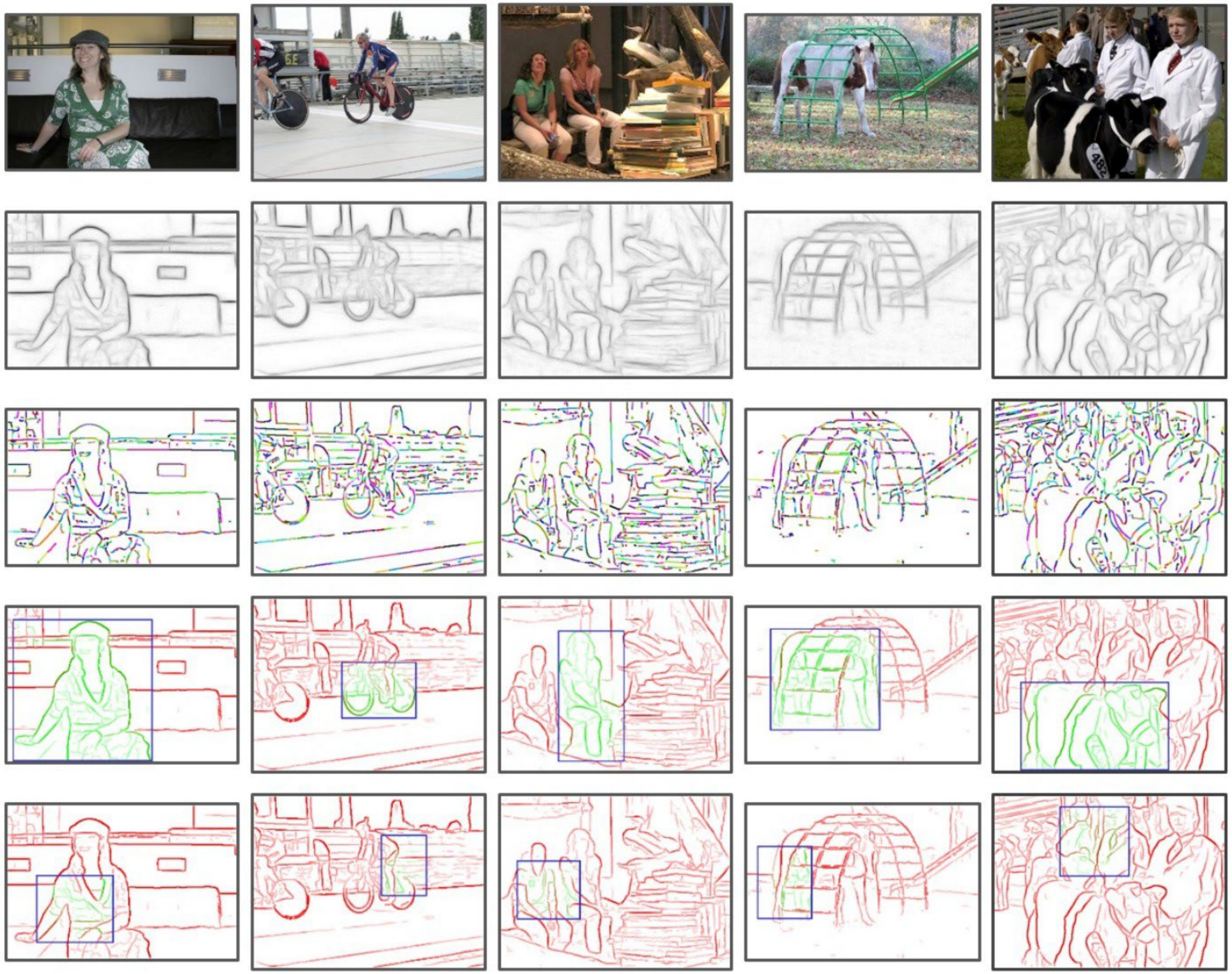
Idea #3: selective search



- Problem: evaluating a detector is very expensive
- An image with n pixels has $O(n^2)$ windows
- Only generate and evaluate a few hundred **region proposals** for regions that are “likely” to be an object of interest.

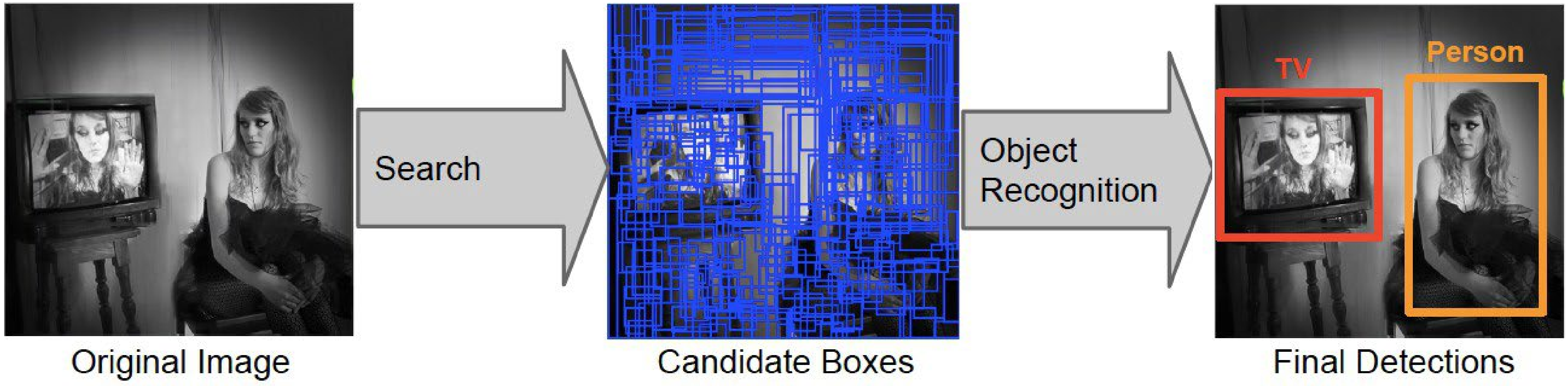
Source: S. Lazebnik

Selective search



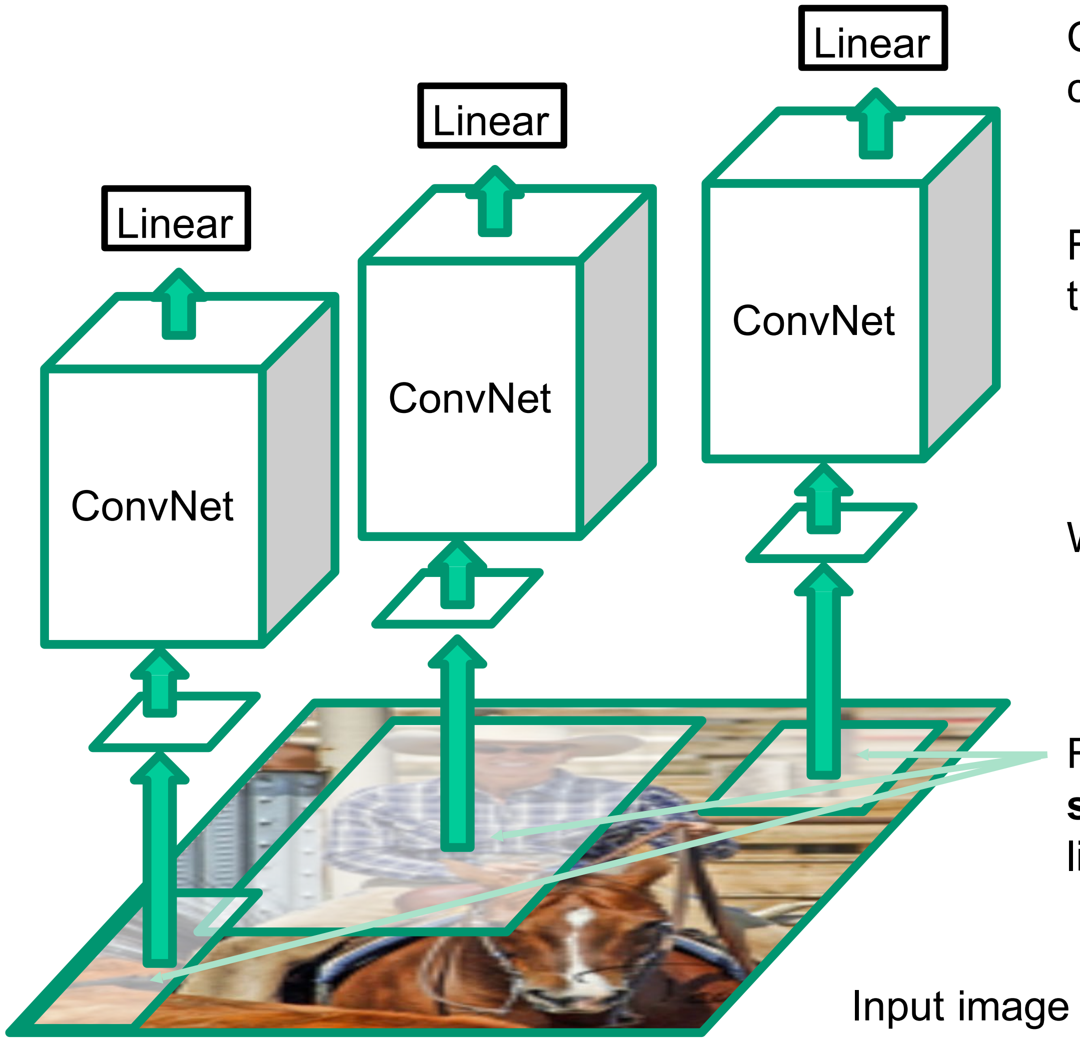
- Example: edge boxes [Zitnick & Dollar, 2014]
- Heuristic: detect edges, group them into contours
- Rank each window based on number of contours in window
- These are the only windows our detector will see

Selective search



- Problem: evaluating a detector is very expensive
- An image with n pixels has $O(n^2)$ windows
- Only generate and evaluate a few hundred **region proposals** for regions that are "likely" to be an object of interest.

R-CNN: Region proposals + CNN features



Classify regions with linear classifier

Forward each region through a CNN

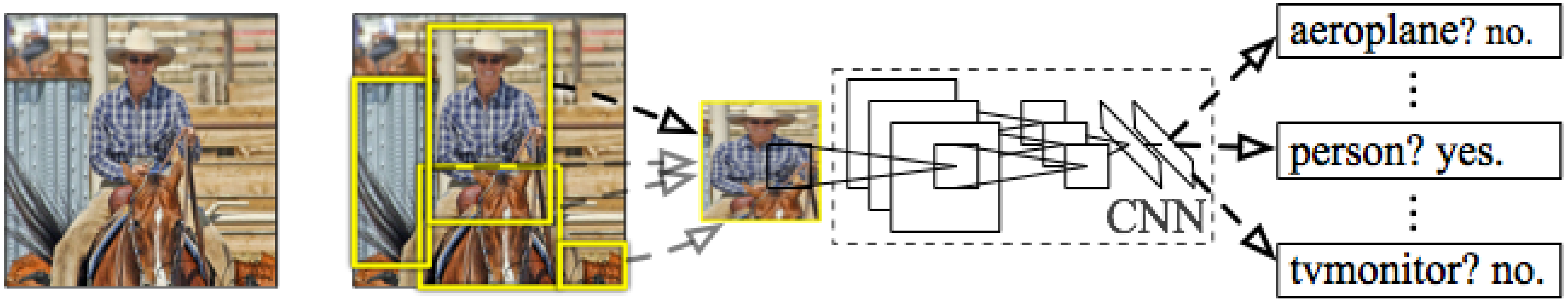
Warped image regions

Region proposals from **selective search** (~2K rectangles that are likely to contain objects)

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

Source: R. Girshick

R-CNN at test time



Input image

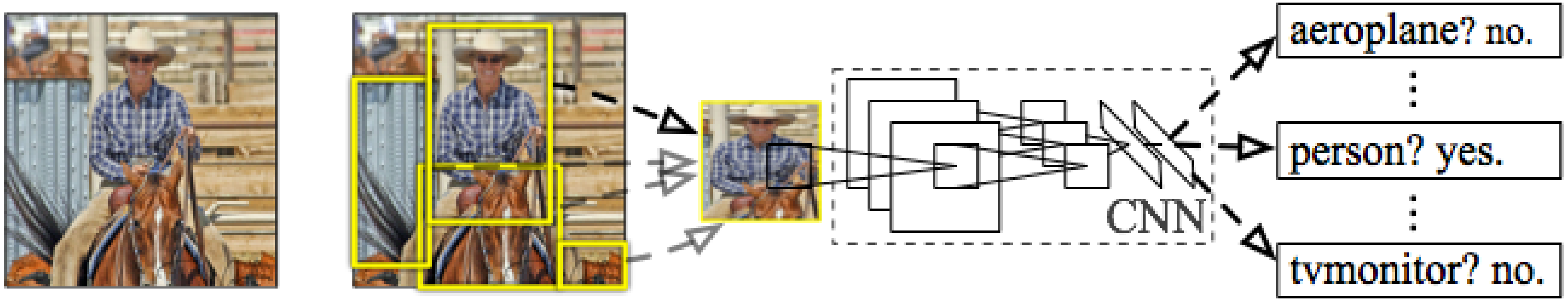
Extract region proposals (~2k / image)

Compute CNN features

a. Crop

Source: R. Girshick

R-CNN at test time

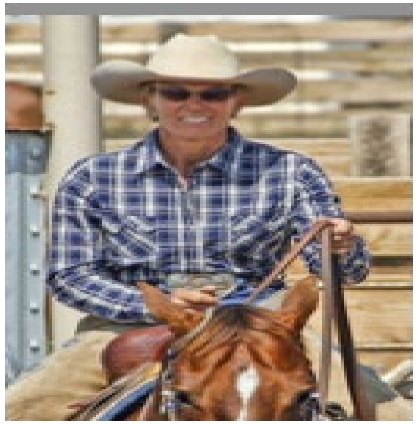


Input image

Extract region proposals (~2k / image)

Compute CNN features

a. Crop

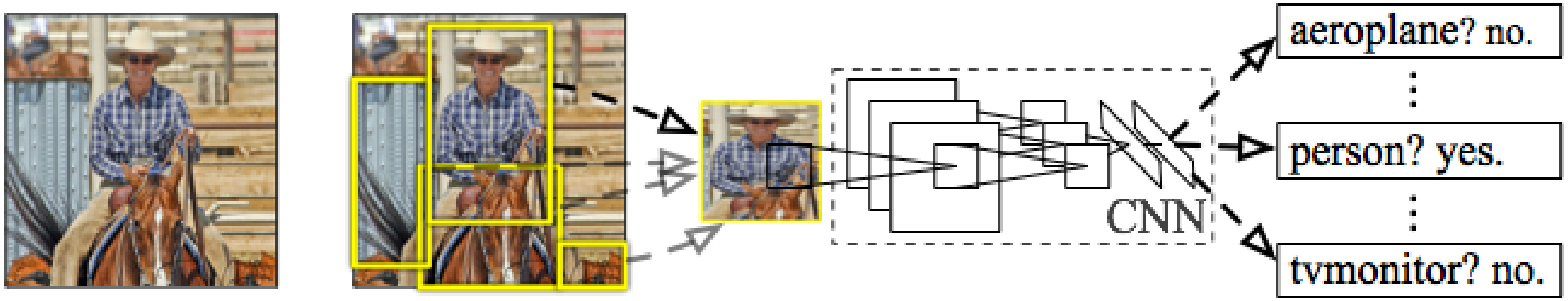


b. Scale

227 x 227

Source: R. Girshick

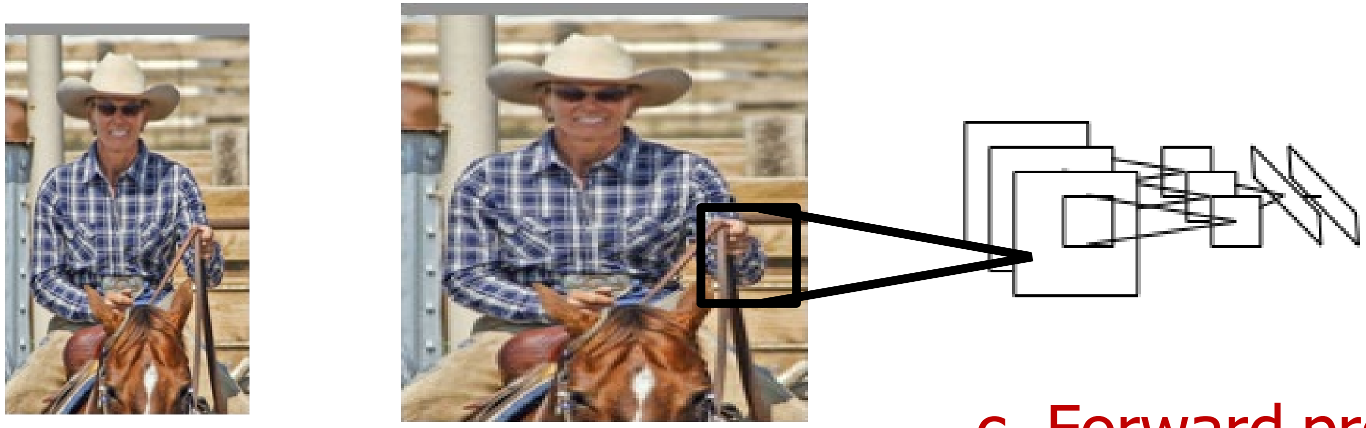
R-CNN at test time



Input image

Extract region proposals (~2k / image)

Compute CNN features



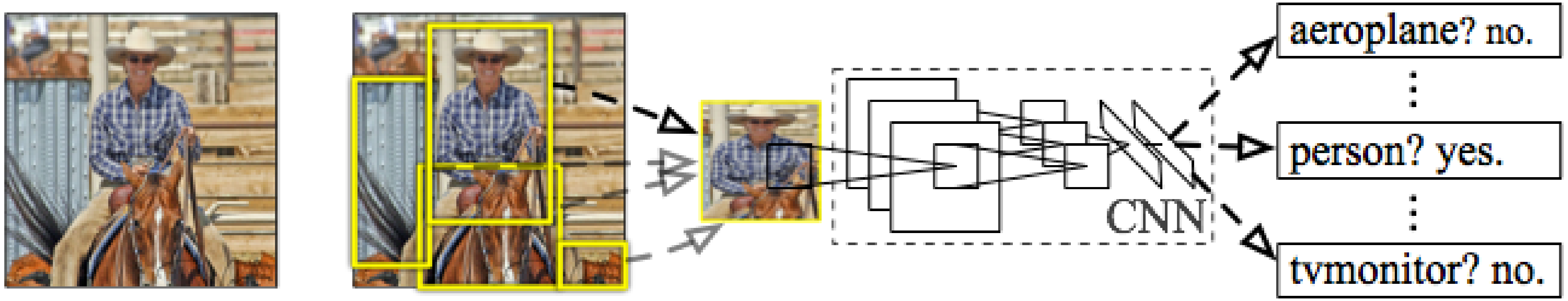
1. Crop

b. Scale

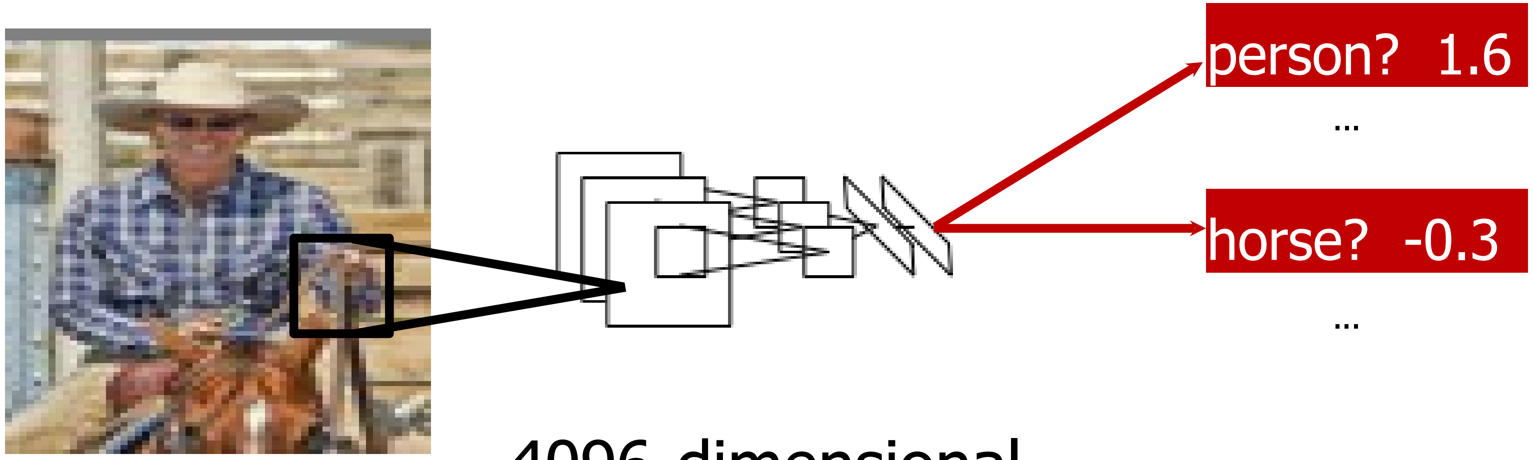
c. Forward propagate Output: “fc_7” features

Source: R. Girshick

R-CNN at test time



Input image | Extract region proposals (~2k / image) | Compute CNN features → Classify regions



Warped proposal | 4096-dimensional fc_7 feature vector | linear classifier

Source: R. Girshick

Proposal refinement

Linear regression

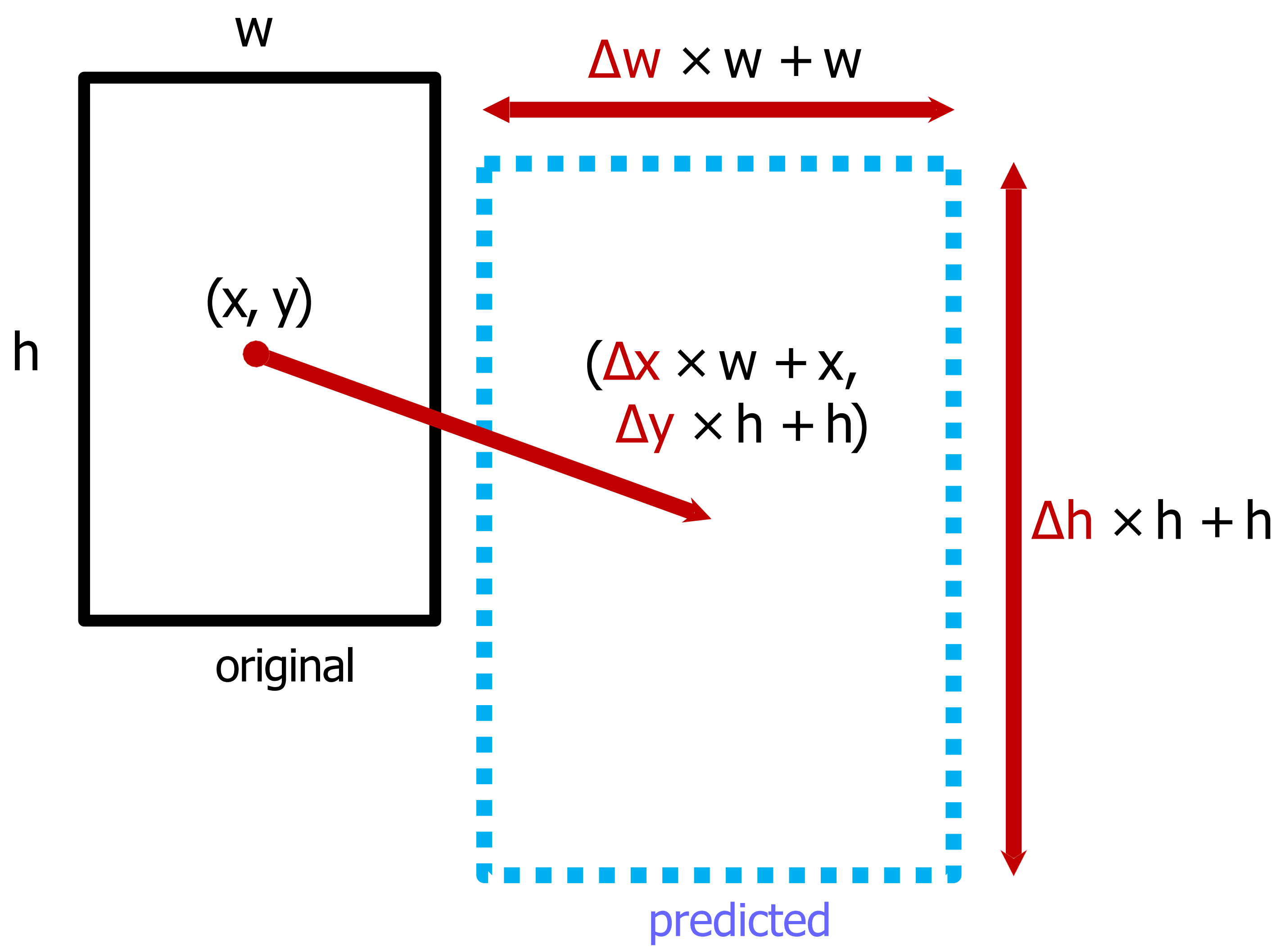
on CNN features

Original
proposal

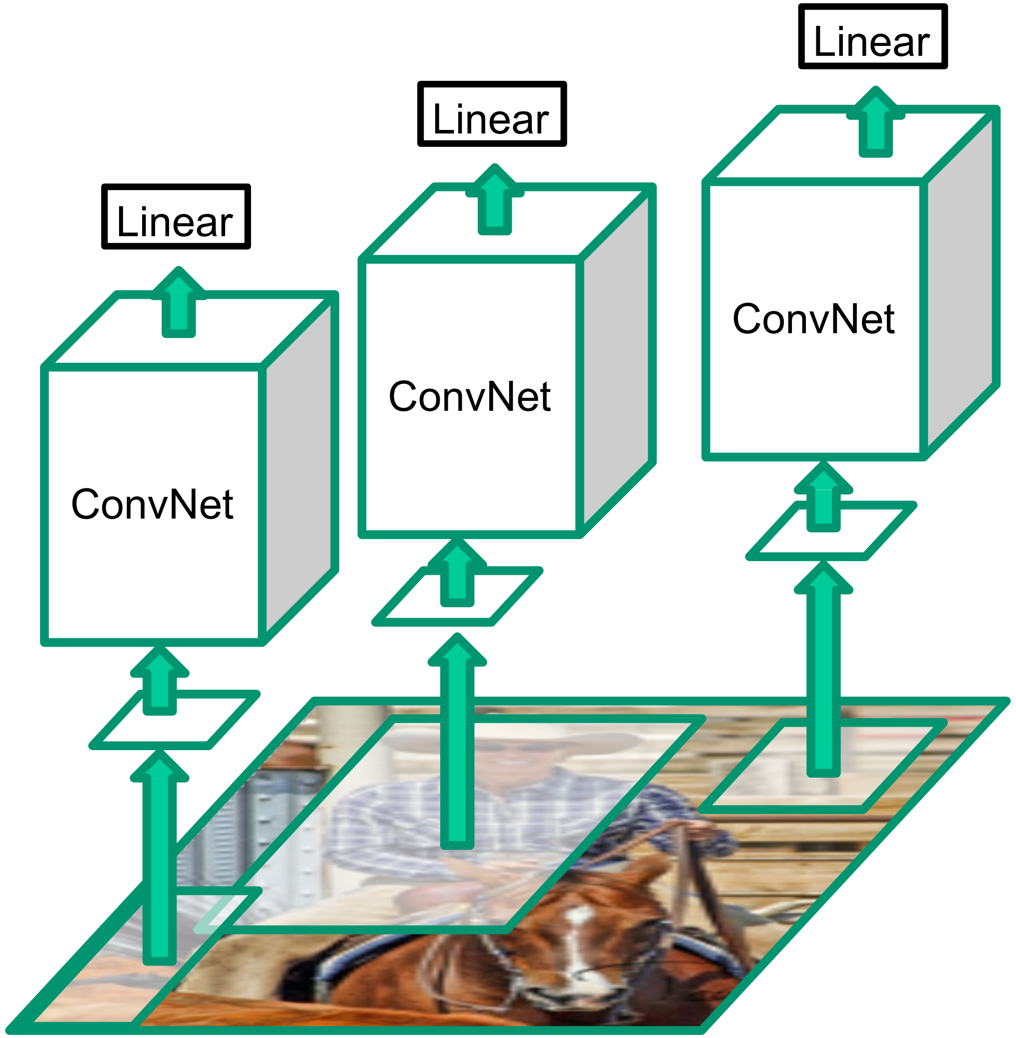
Predicted
object bounding box

Bounding-box regression

Bounding-box regression



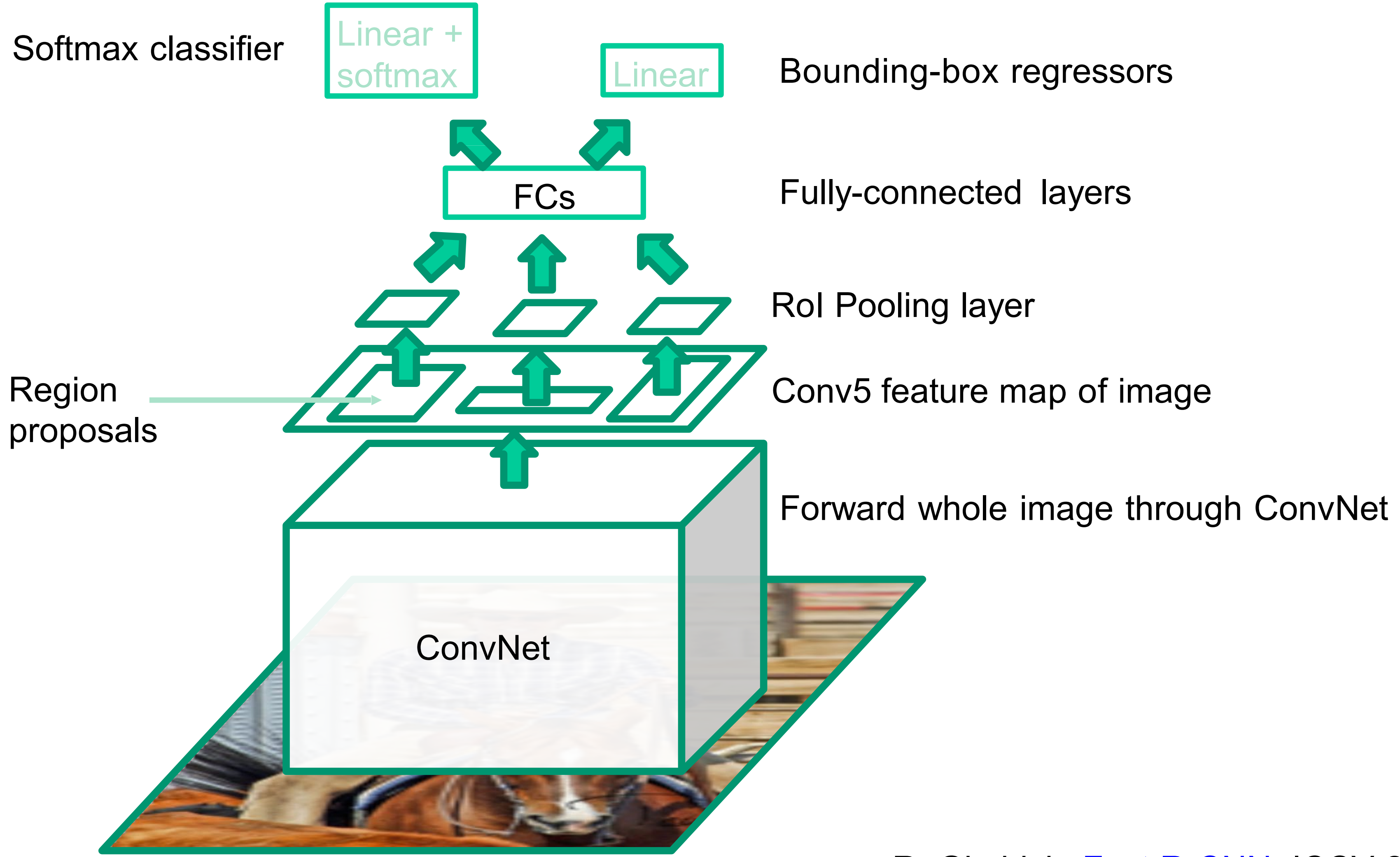
Problems with R-CNN



1. Slow! Have to run CNN per window

2. Hand-crafted mechanism for region proposal might be suboptimal.

“Fast” R-CNN: reuse features between proposals

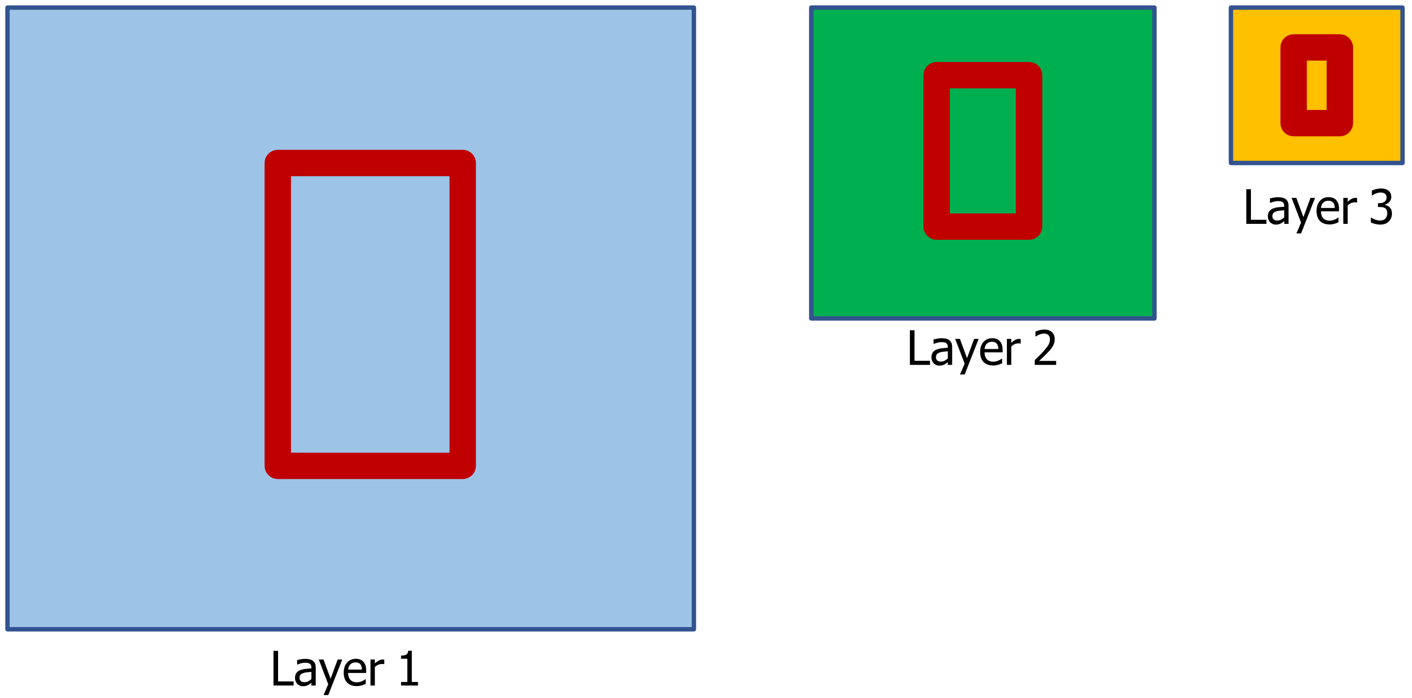


Source: R. Girshick

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

ROI Pooling

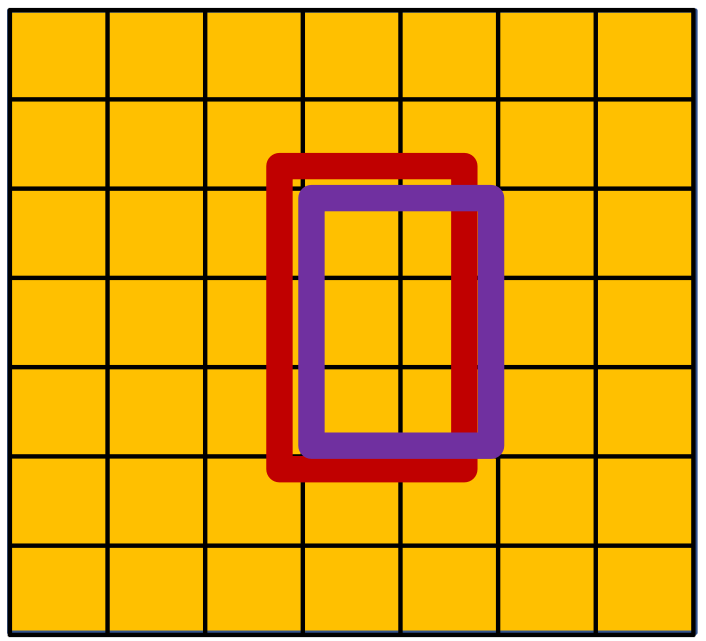
- How do we crop from a feature map?
- Step 1: Resize boxes to account for subsampling



Source: B. Hariharan

ROI Pooling

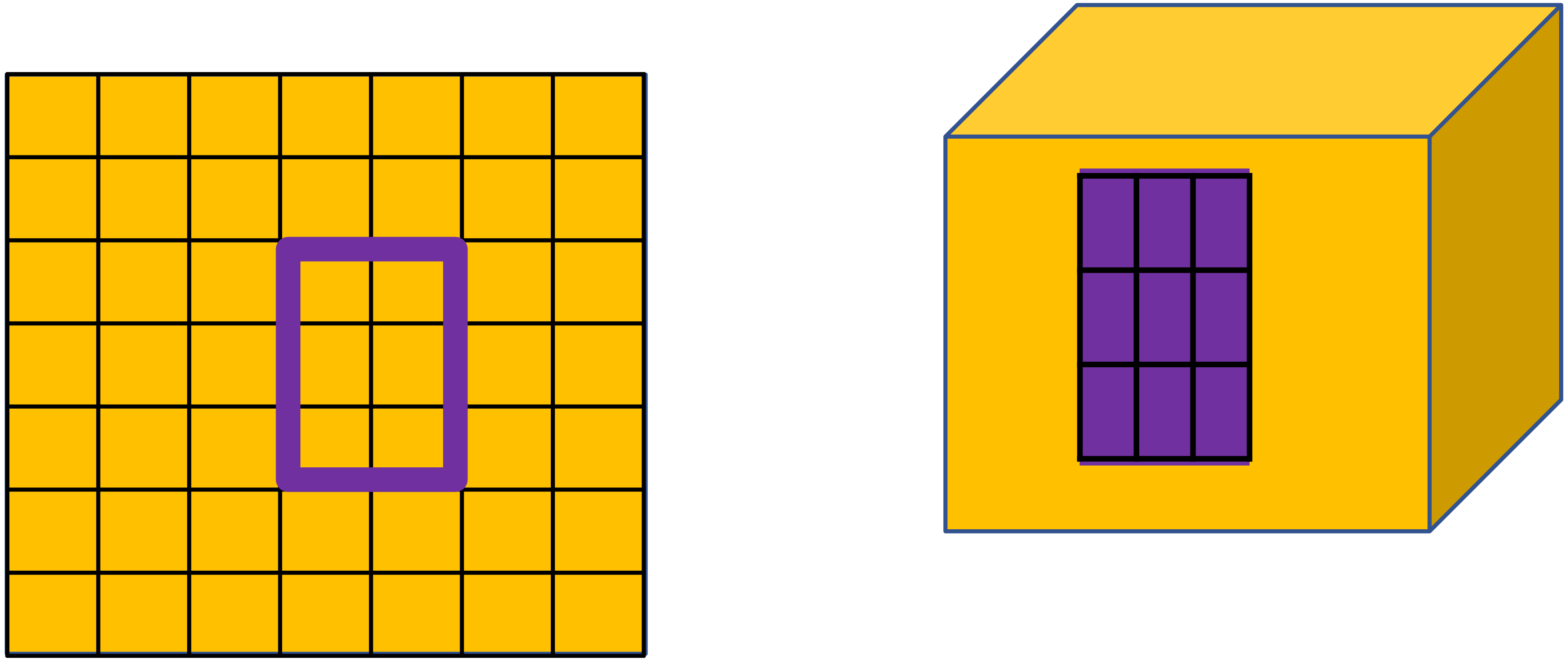
- How do we crop from a feature map?
- Step 2: Snap to feature map grid



Source: B. Hariharan

ROI Pooling

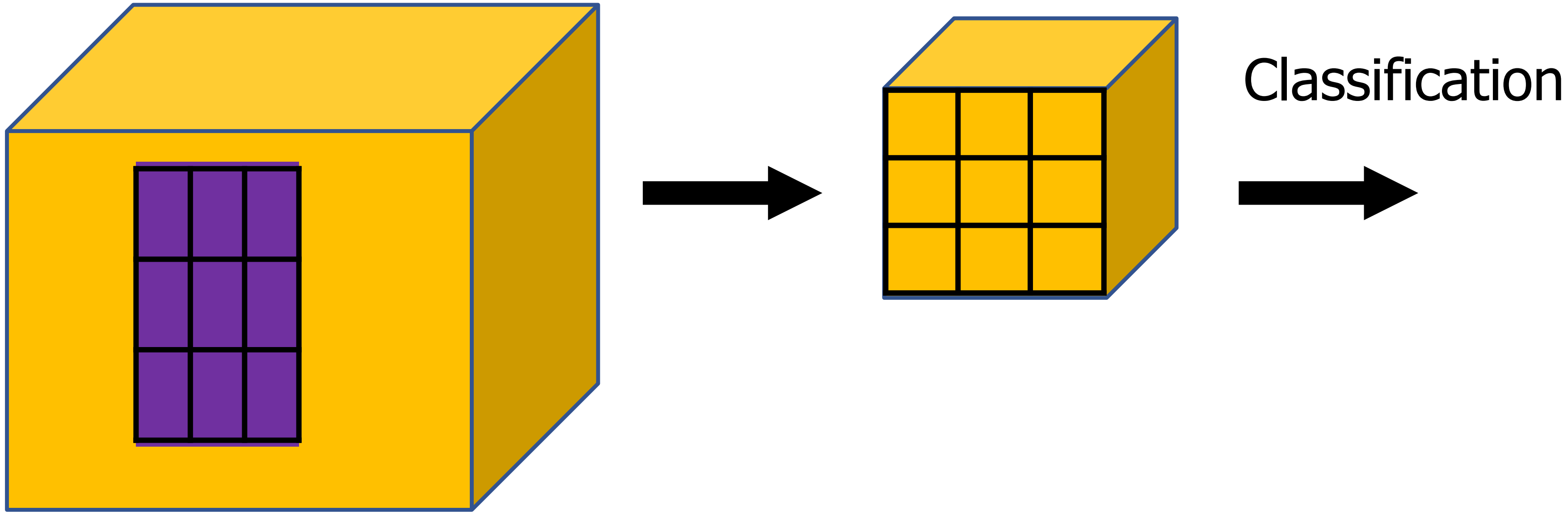
- How do we crop from a feature map?
- Step 3: Overlay a new grid of fixed size



Source: B. Hariharan

ROI Pooling

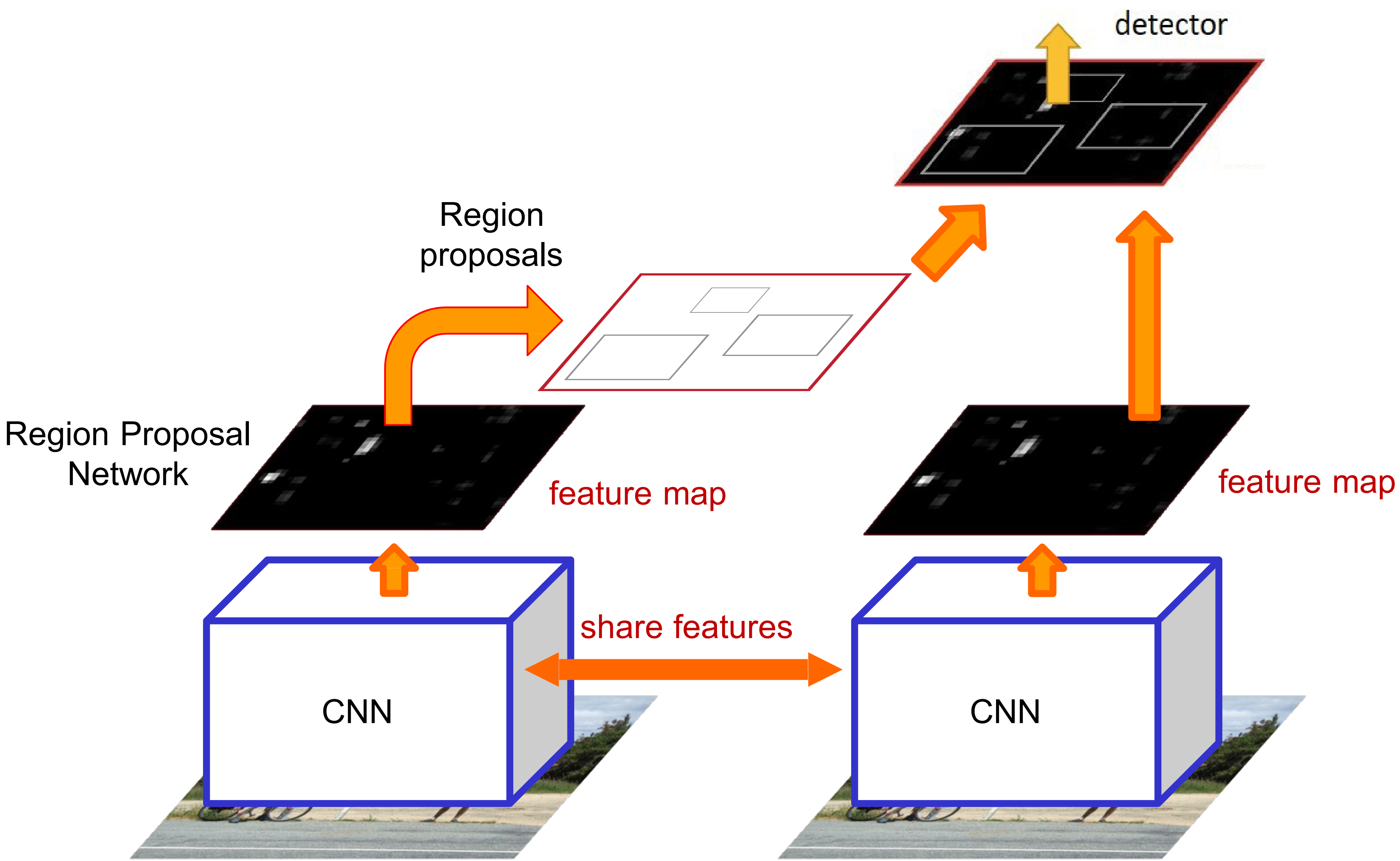
- How do we crop from a feature map?
- Step 4: Take max in each cell
- Can improve with bilinear sampling



See more here: https://deepsense.ai/region-of-interest-pooling-explained/

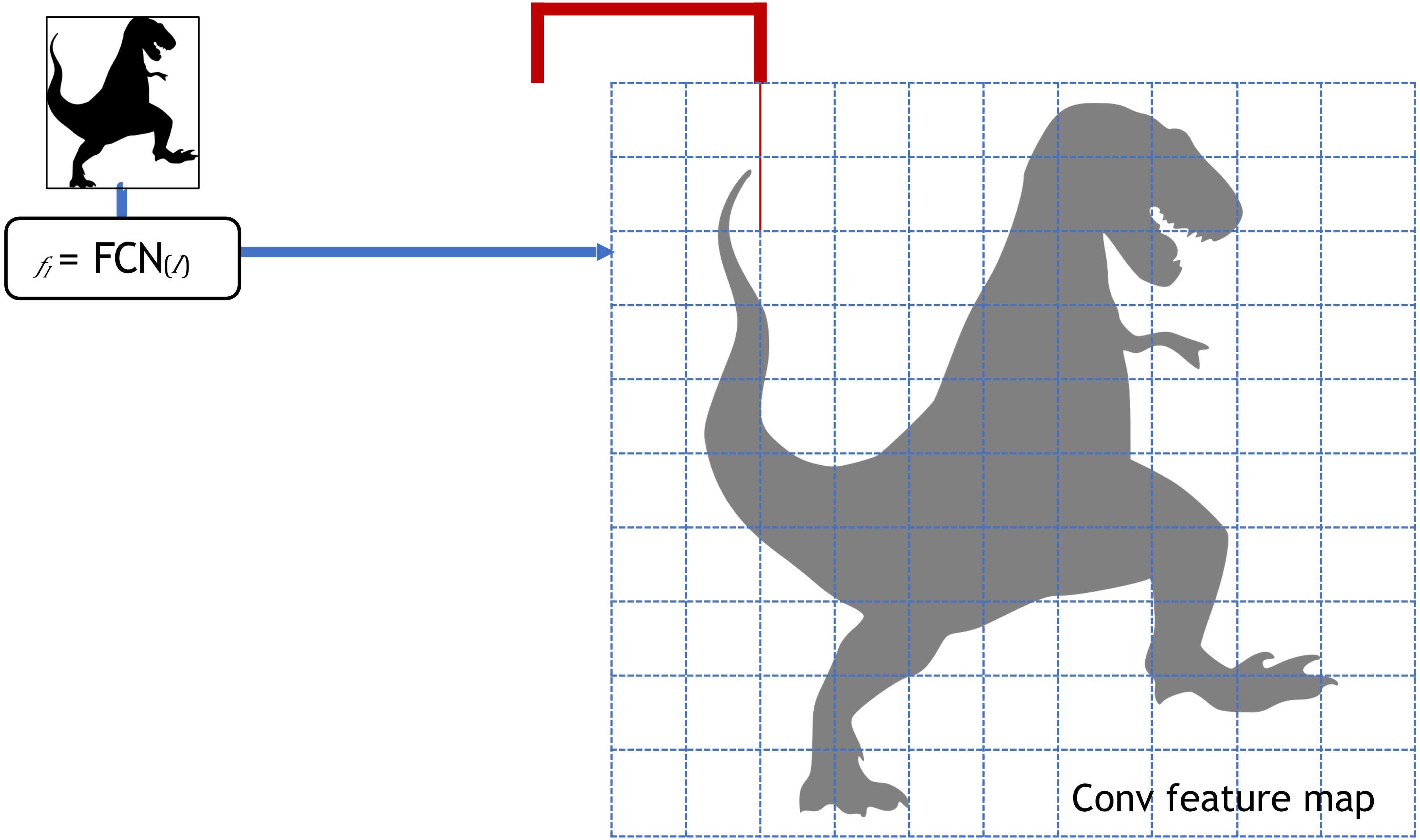
Source: B. Hariharan

“Faster” R-CNN: learn region proposals



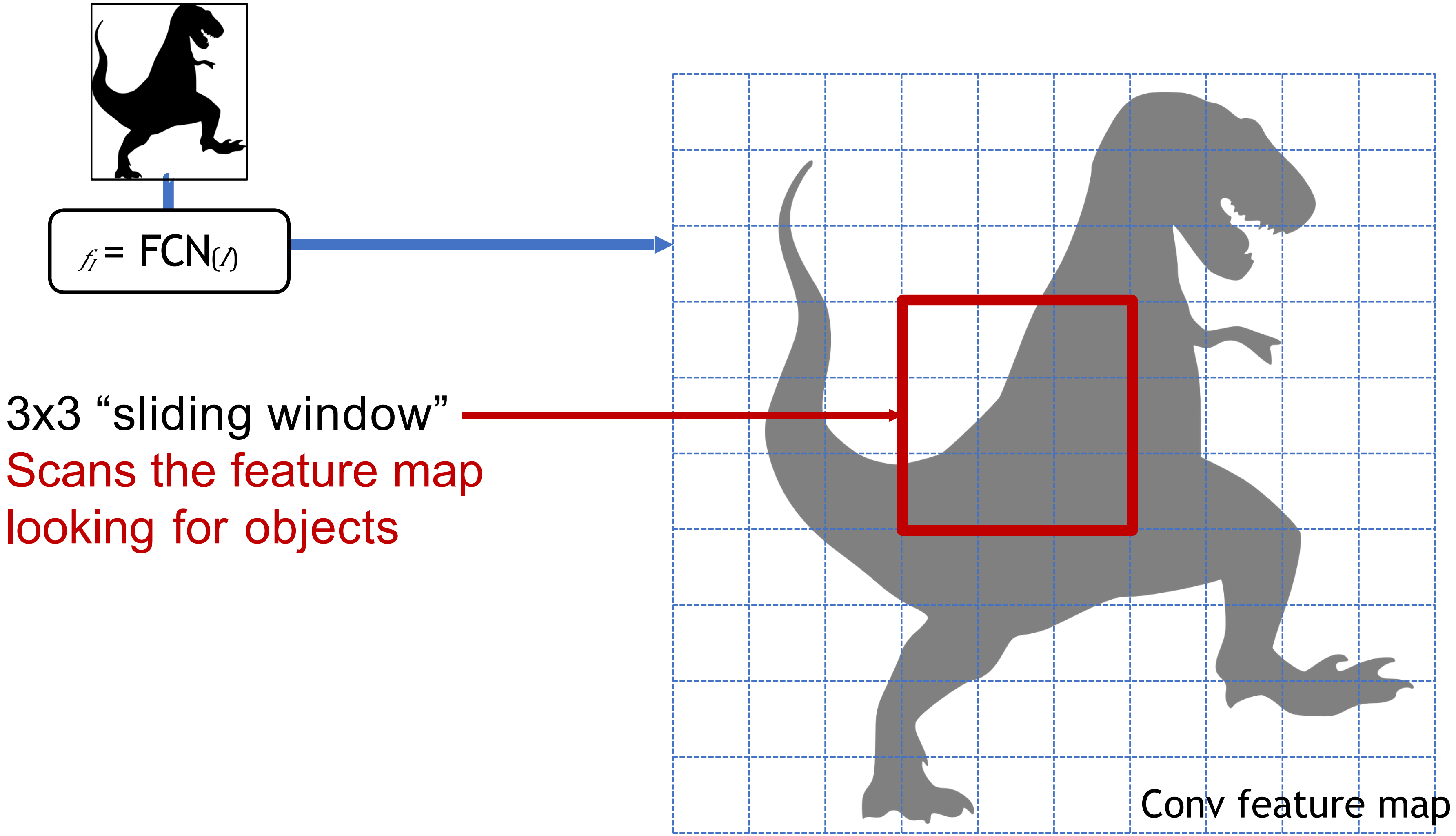
S. Ren, K. He, R. Girshick, and J. Sun, [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks], NIPS 2015

RPN: Region Proposal Network



Source: R. Girshick

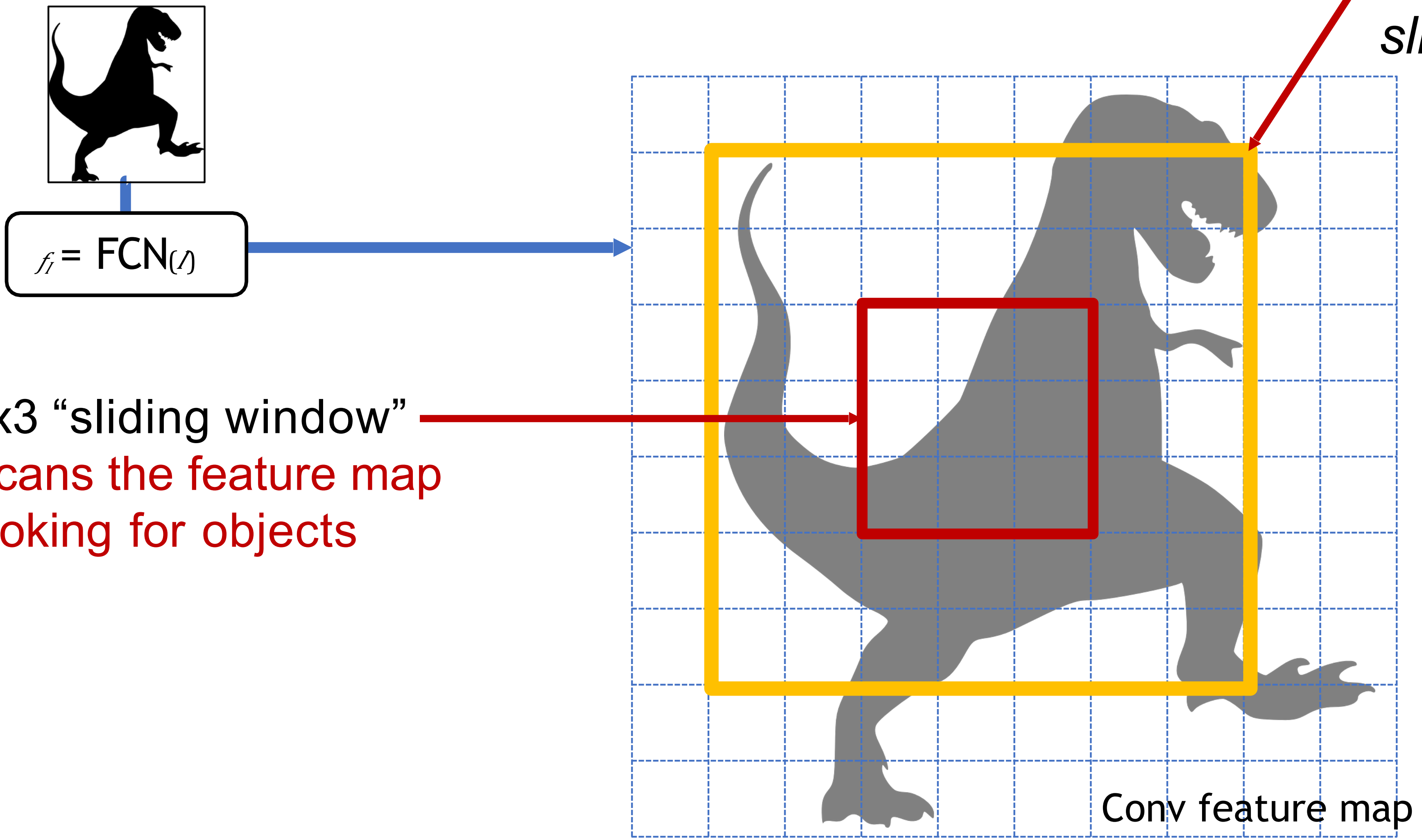
RPN: Region Proposal Network



Source: R. Girshick

RPN: Anchor Box

Anchor box: predictions are w.r.t. this box, *not the 3x3 sliding window*

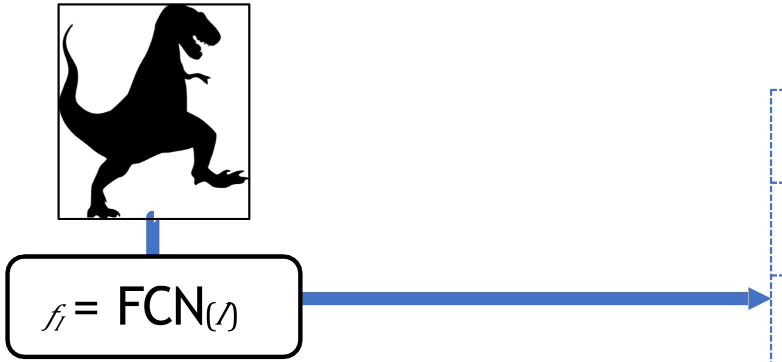


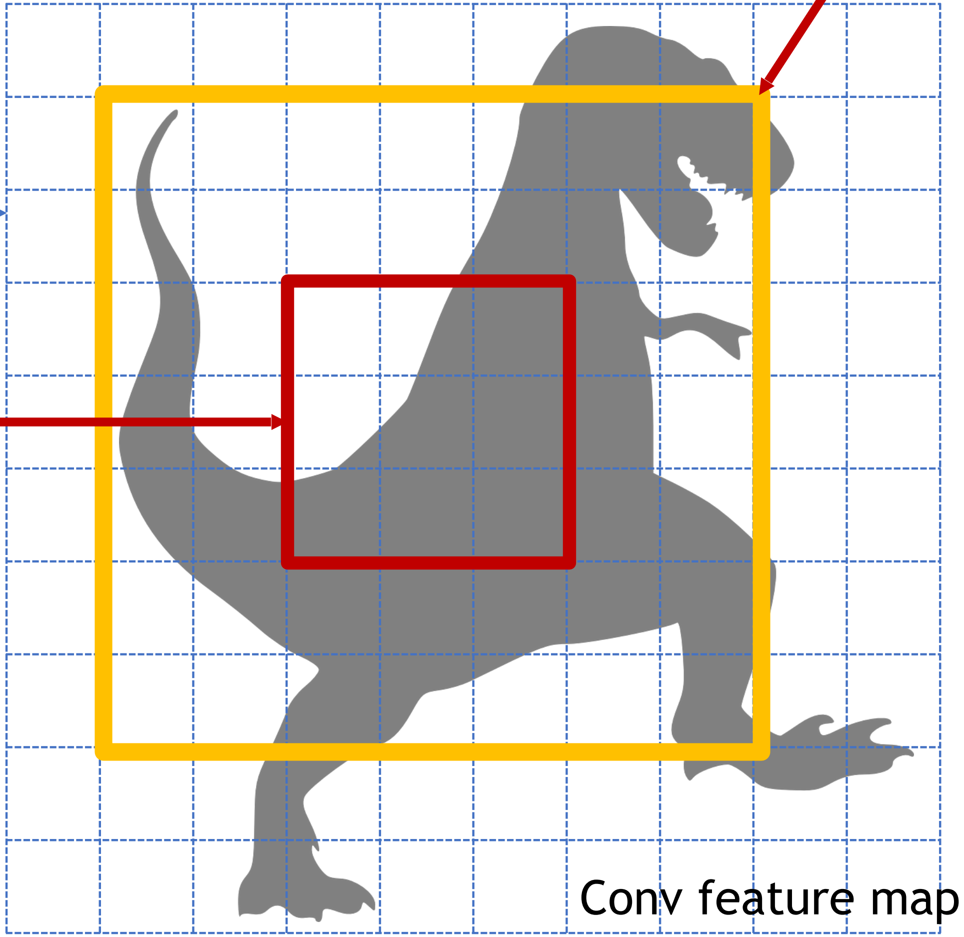
3x3 "sliding window"
Scans the feature map looking for objects

Source: R. Girshick

RPN: Anchor Box

Anchor box: predictions are w.r.t. this box, *not the 3x3 sliding window*



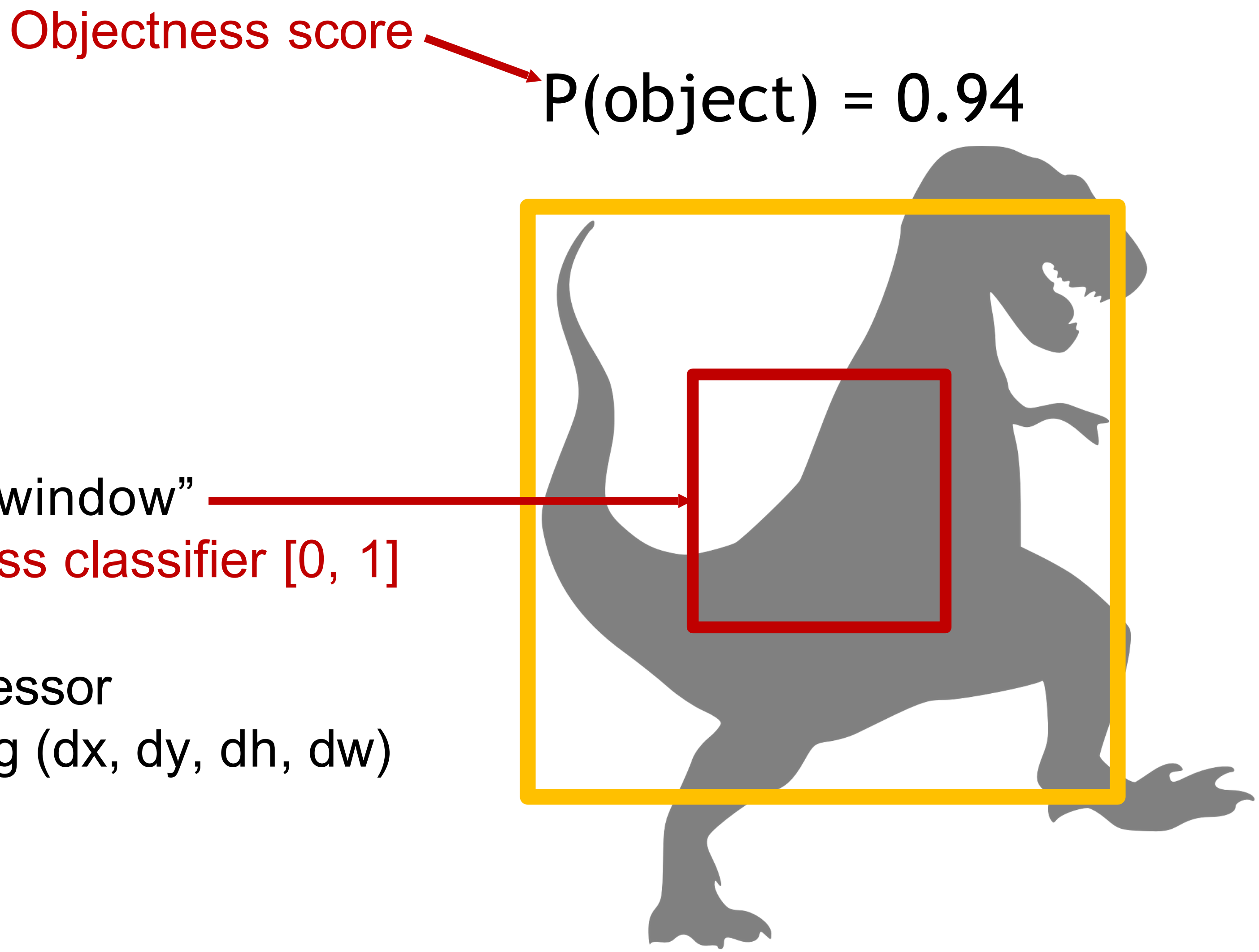


3x3 "sliding window"

- ➢ Objectness classifier [0, 1]
- ➢ Box regressor predicting (dx, dy, dh, dw)

Source: R. Girshick

RPN: Prediction (on object)



3x3 “sliding window”

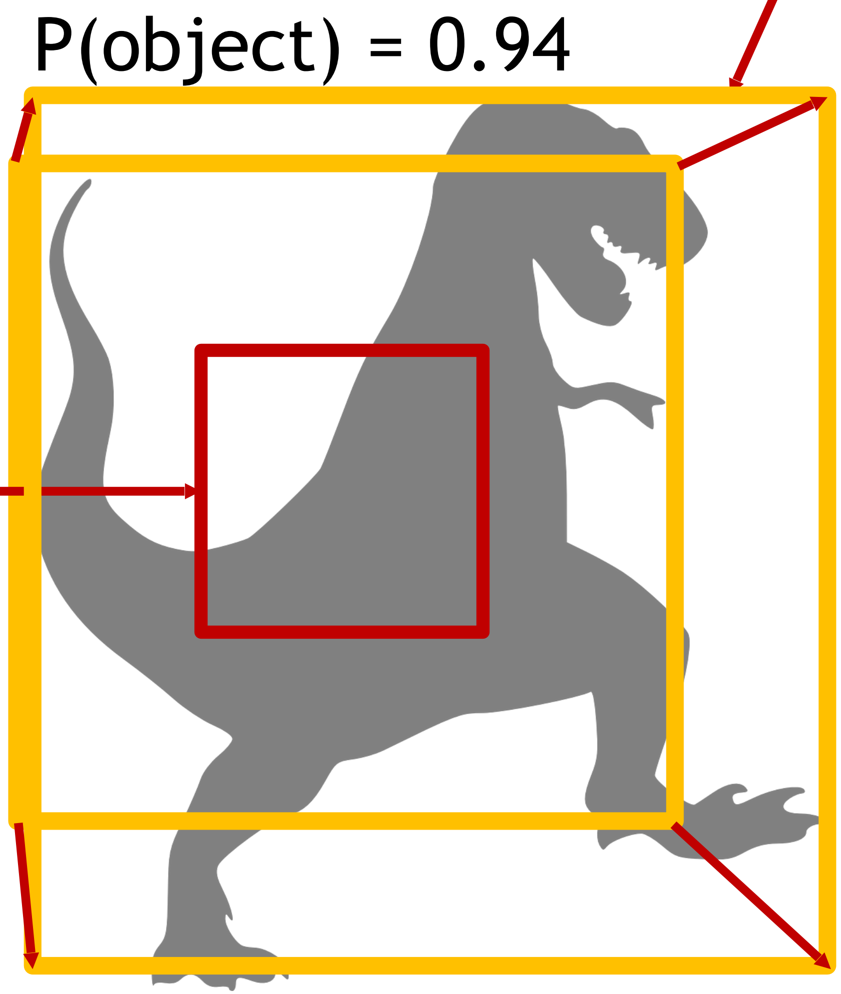
➢ Objectness classifier [0, 1]

➢ Box regressor predicting (dx, dy, dh, dw)

Source: R. Girshick

RPN: Prediction (on object)

Anchor box: transformed by box regressor



3x3 "sliding window"

- ➢ Objectness classifier [0, 1]
- ➢ Box regressor predicting (dx, dy, dh, dw)

Source: R. Girshick

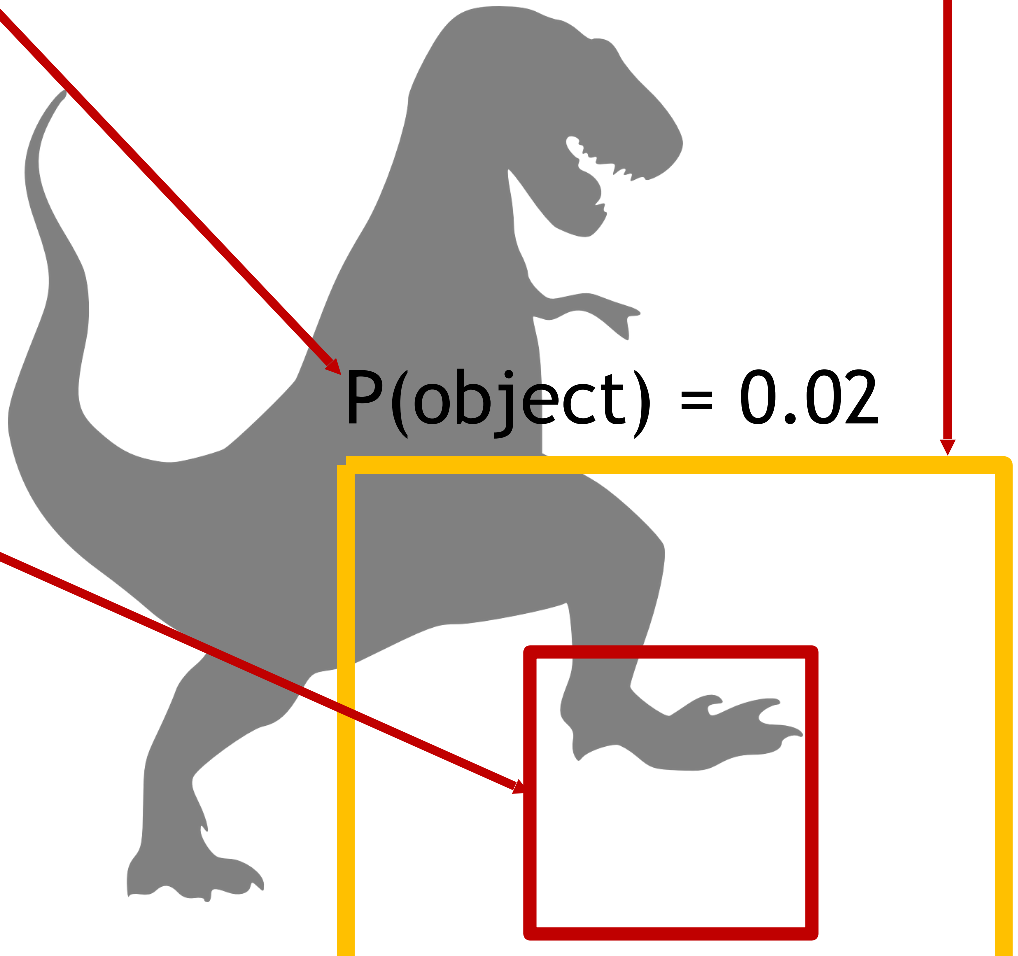
RPN: Prediction (off object)

Anchor box: transformed by box regressor

Objectness score

3x3 "sliding window"

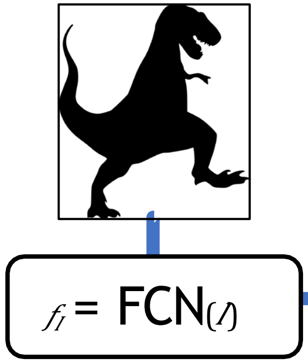
- ➢ Objectness classifier
- ➢ Box regressor predicting (dx, dy, dh, dw)

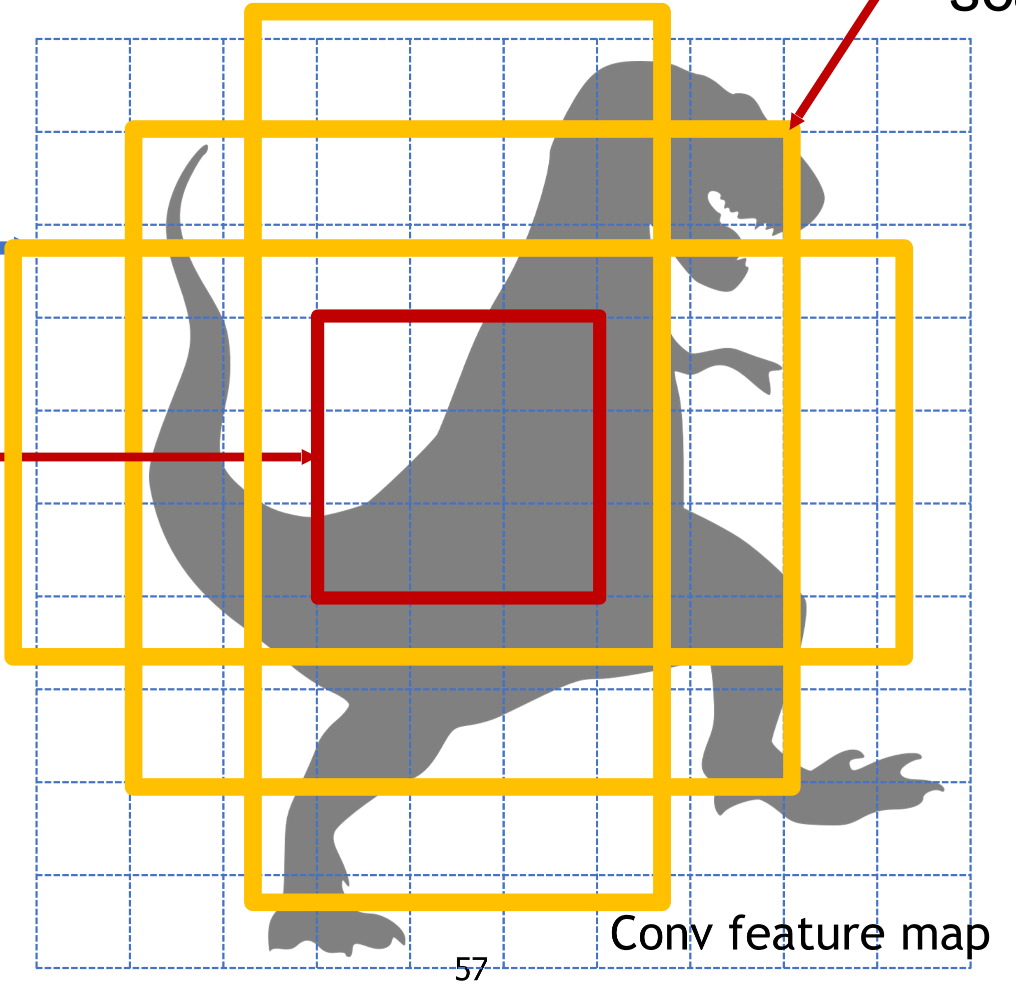


Source: R. Girshick

RPN: Multiple Anchors

Anchor boxes: *K* anchors per location with different scales and aspect ratios



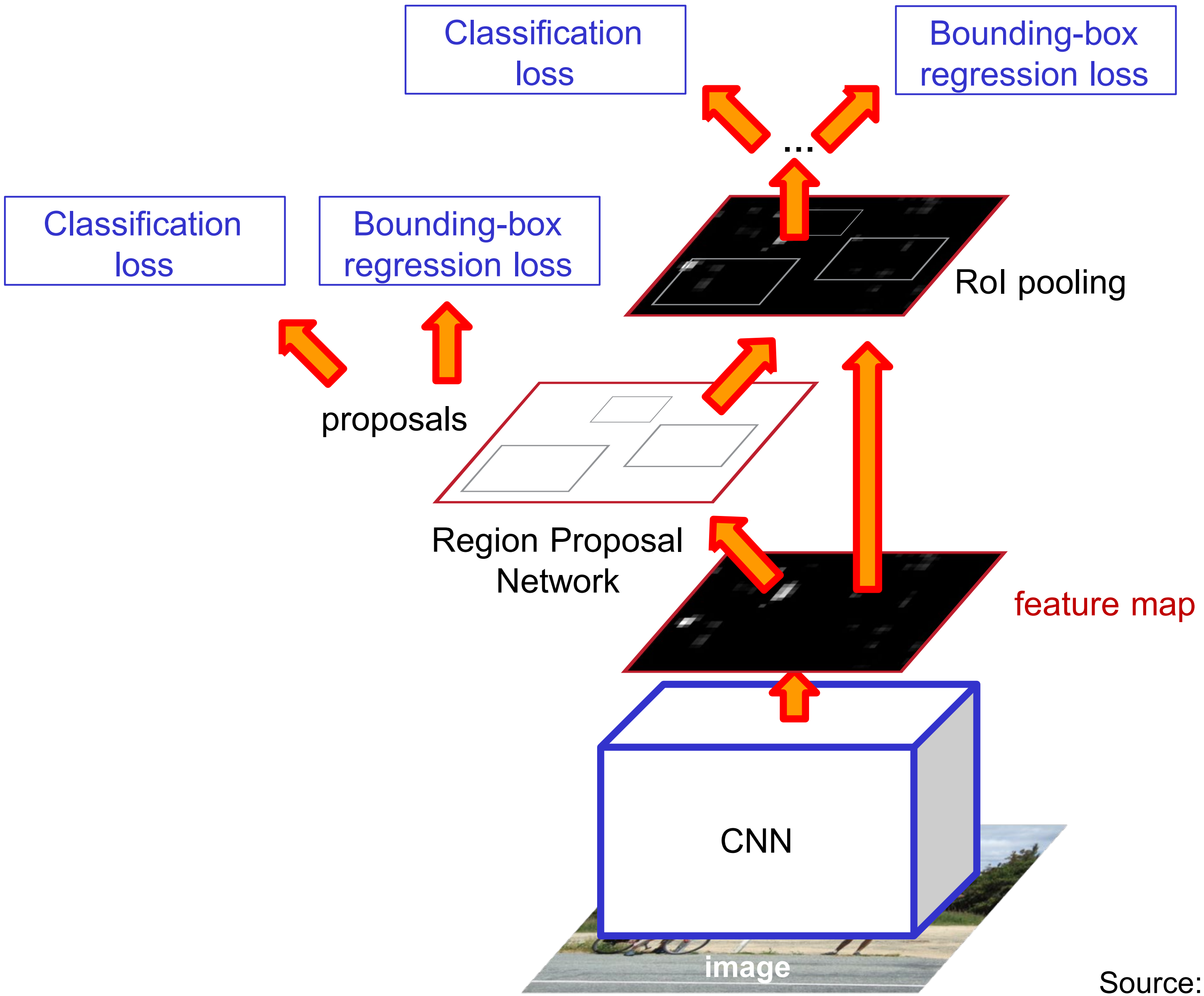


3x3 "sliding window"

- ➢ *K* objectness classifiers
- ➢ *K* box regressors

Source: R. Girshick

One network, four losses



Source: R. Girshick, K. He, S. Lazebnik

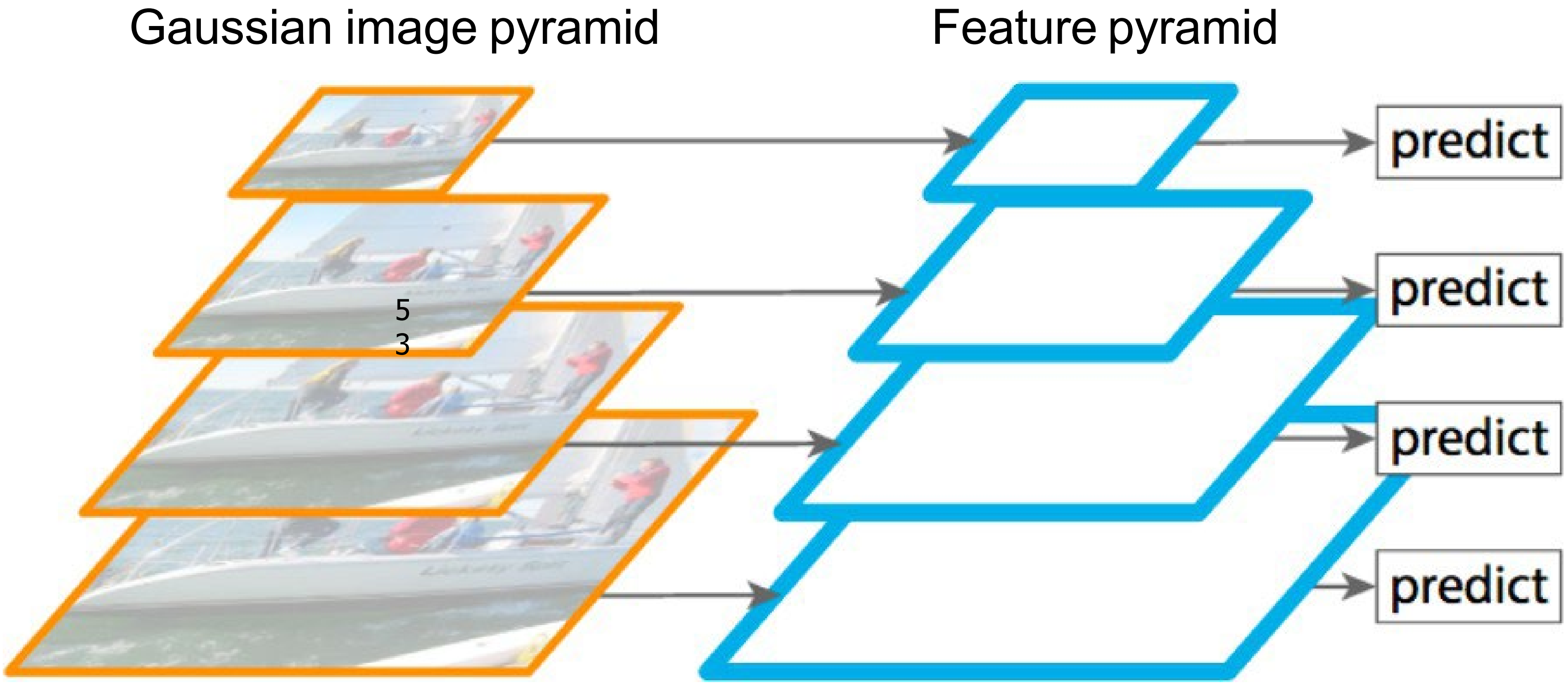
Faster R-CNN results

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

Source: S. Lazebnik

How do we deal with scale? Idea #1: Gaussian pyramid



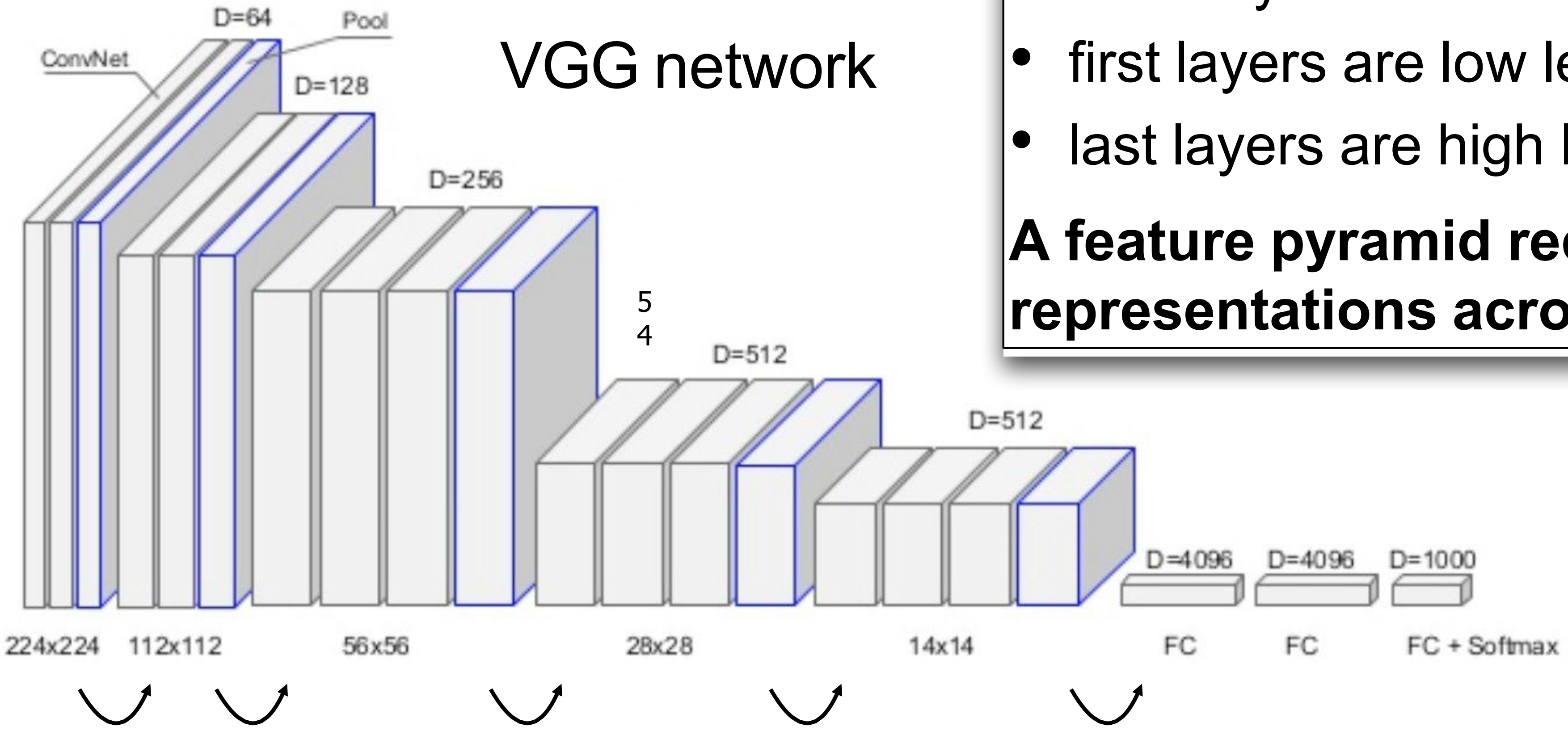
[Lin et al., "Feature Pyramid Networks for Object Detection", 2017]

Source: Torralba, Freeman, Isola

Image and features pyramids

Each pooling reduces the resolution by
a factor of 2

VGG network



CNN architectures build:

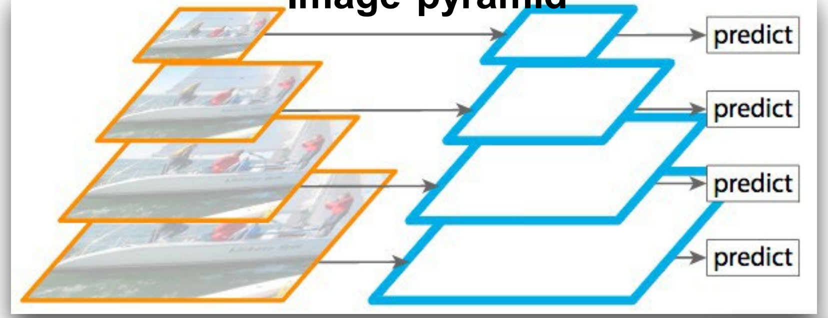
- Multiscale feature hierarchies, but
- each layer builds a different representation
- first layers are low level, while
- last layers are high level.

A feature pyramid requires a uniform representations across scales.

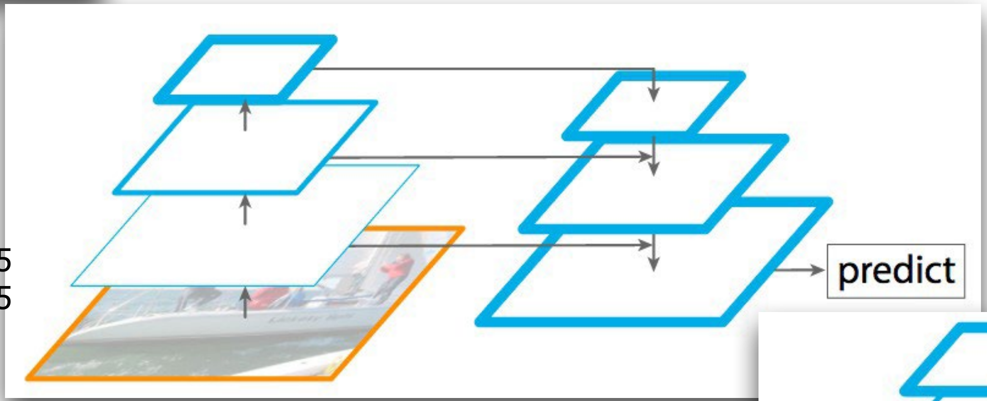
Source: Torralba, Freeman, Isola

Idea #2: Feature pyramid network

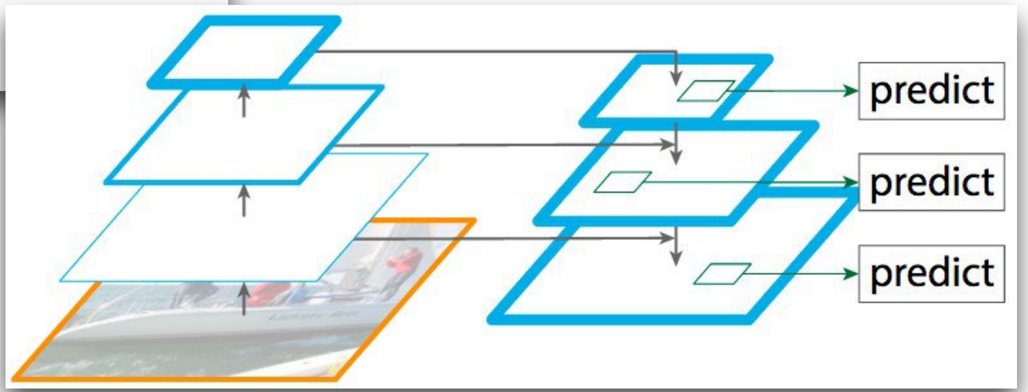
Image pyramid



Encoder-decoder architecture (U-Net)

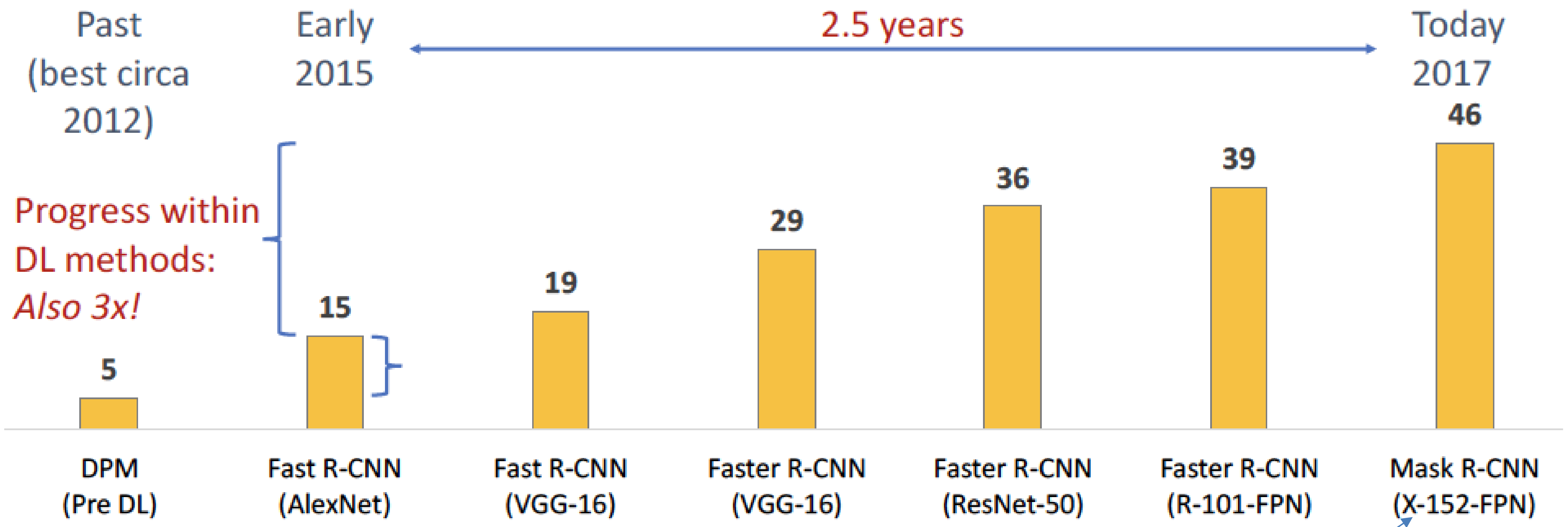


Feature pyramid



Source: Torralba, Freeman, Isola

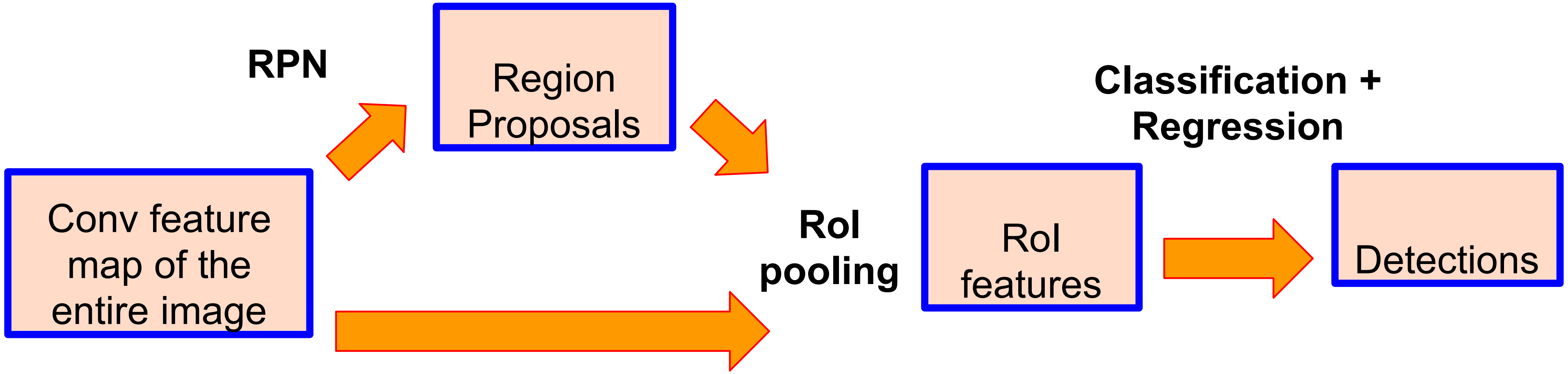
Summary: *Progress through Better Detector Design and Better Backbones*



Will talk about Mask R-CNN when
we discuss segmentation

Streamlined detection architectures

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

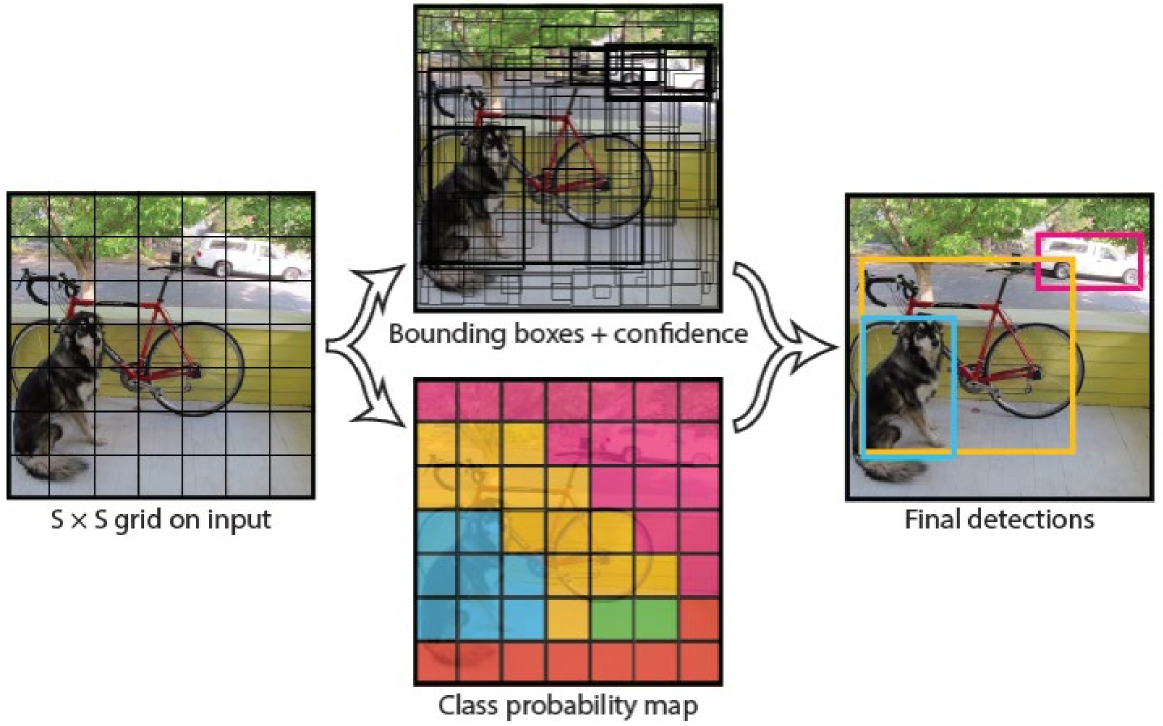


- Is it possible do detection in one shot?

Source: S. Lazebnik

Single-stage object detector

- Divide the image into a coarse grid using a fully convolutional net
- Directly predict class label, confidence, 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

Source: S. Lazebnik

YOLO detector

- Take convolutional feature maps at 7x7 resolution
- Predict, at each location, a score for each class and 2 bounding boxes (with confidence)

E.g. for 20 classes, output is 7x7x30 (30 = 20 + 2*(4+1))

7x speedup over Faster R-CNN (45-155 FPS vs. 7-18 FPS)

but less accurate (e.g. 65% vs. 72 mAP%)

Extension:

use anchor boxes in last layer to try

a few possible aspect ratios


Bounding boxes + confidence


Class probability map

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

Source: S. Lazebnik

- Introduction to scene understanding
- Object detection models
- **Evaluating object detectors**
- Future challenges

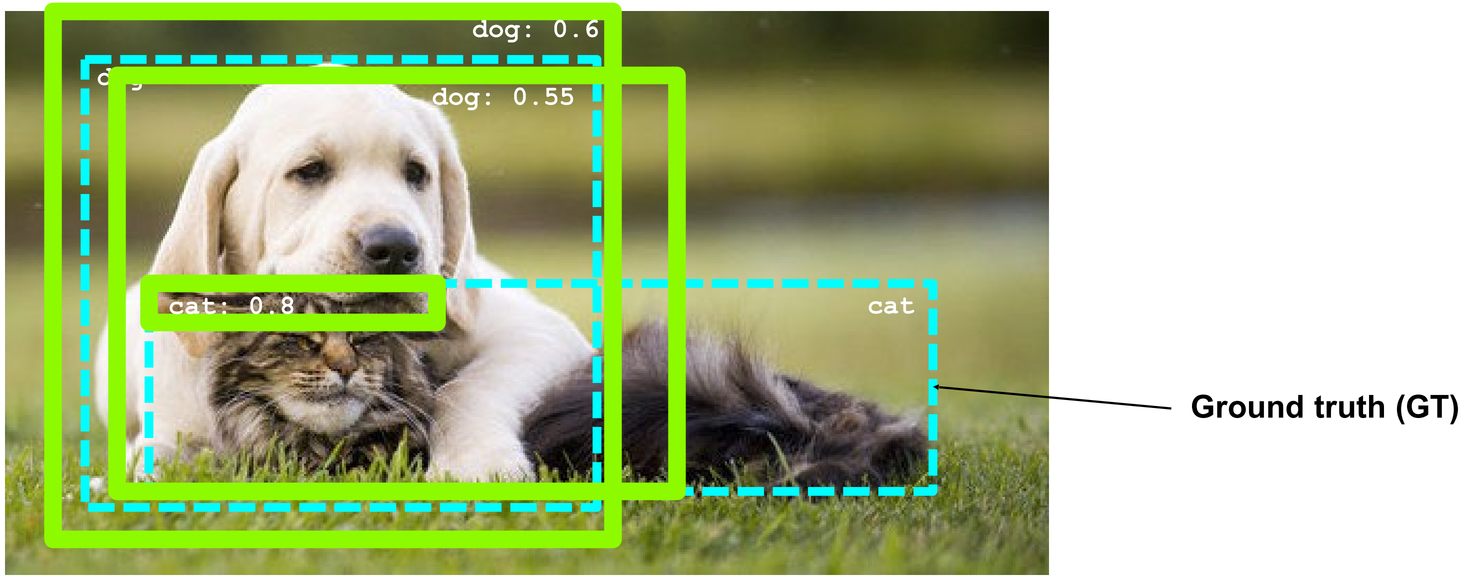
Evaluating an object detector

Evaluating an object detector

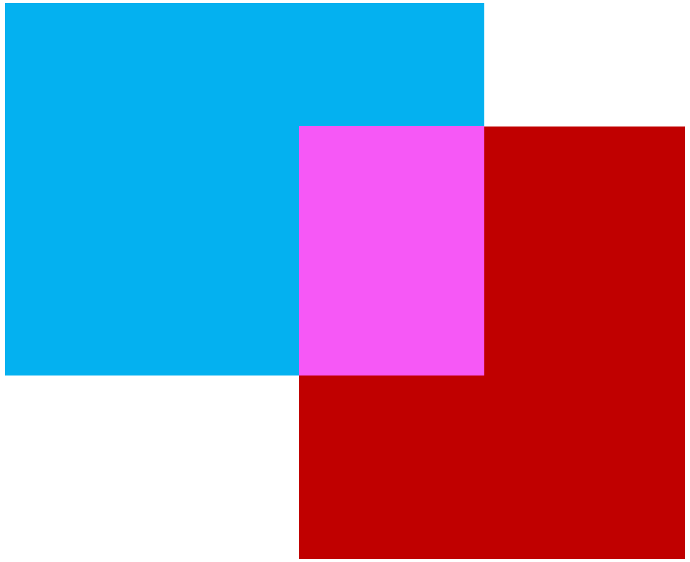
- At test time, predict bounding boxes, class labels, and confidence scores
- For each detection, determine whether it is a true or false positive

Intersection over union (IoU):

$$\text{Area}(GT \cap Det) \,/\, \text{Area}(GT \cup Det) > 0.5$$



Evaluating an object detector

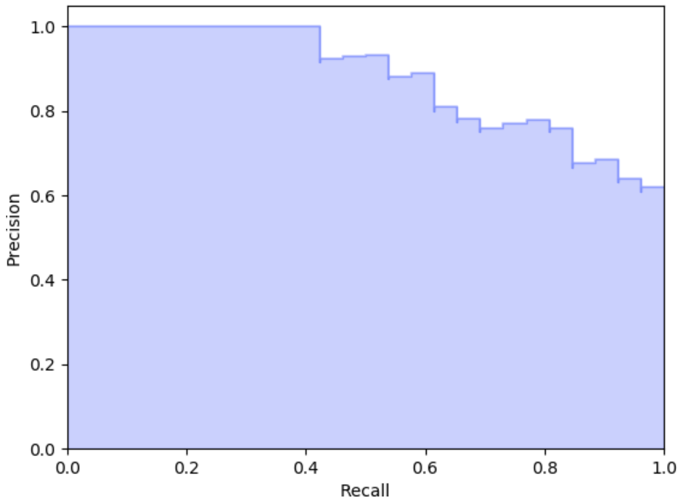


$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Intersection over union (also known as Jaccard similarity)

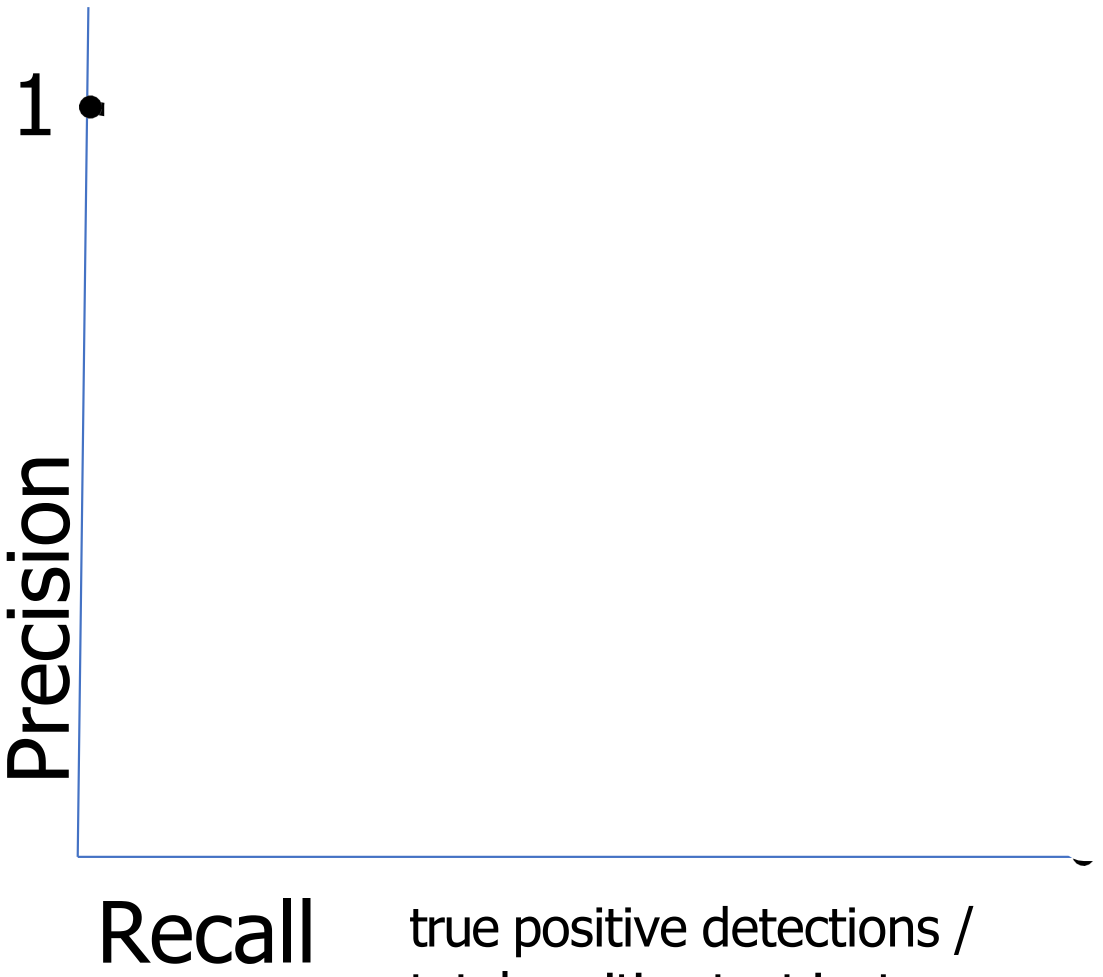
Evaluating an object detector

- For each class, plot Precision-Recall curve and compute Average Precision (area under the curve)
- Take mean of AP over classes to get mAP



Precision:
true positive detections /
total detections
Recall:
true positive detections /
total positive test instances

Average precision

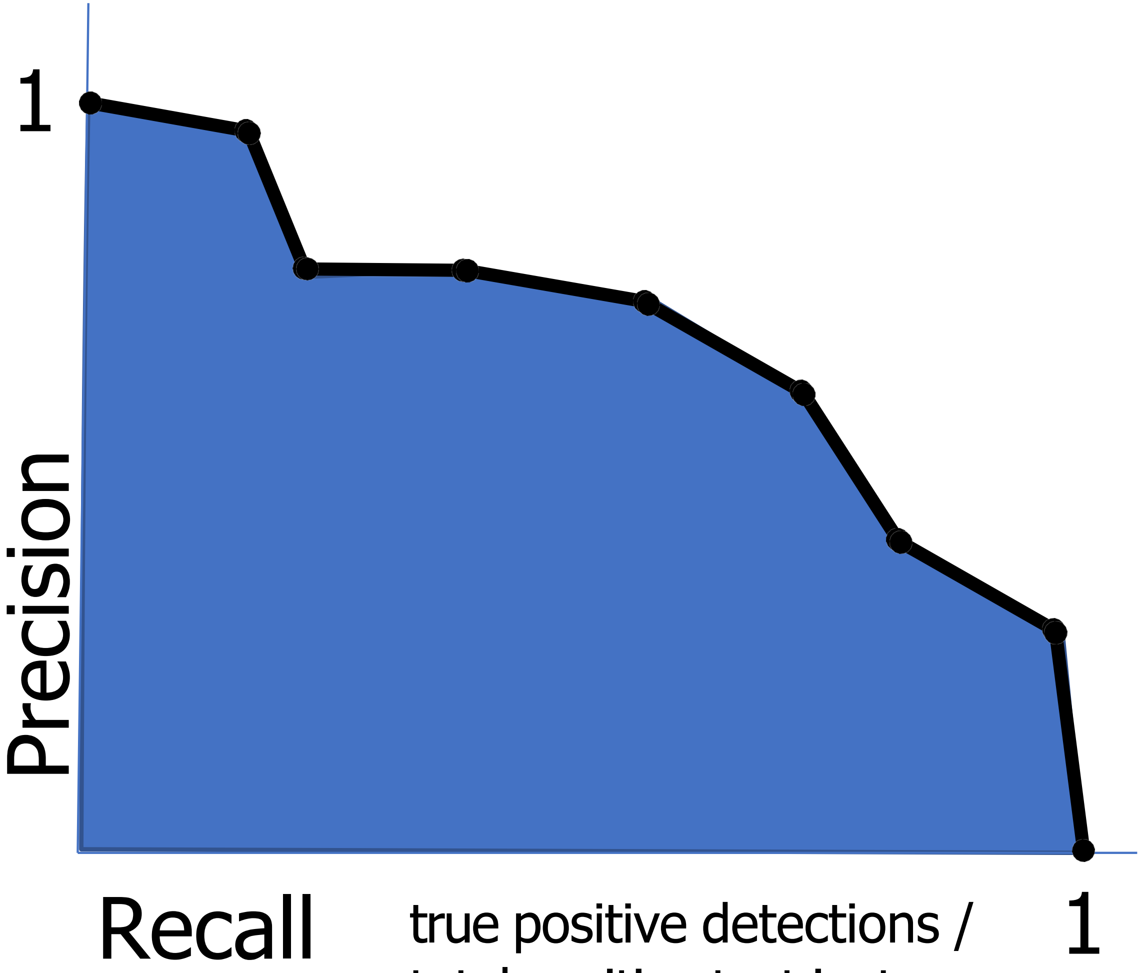


true pos. /
total detections

true positive detections /
total positive test instances

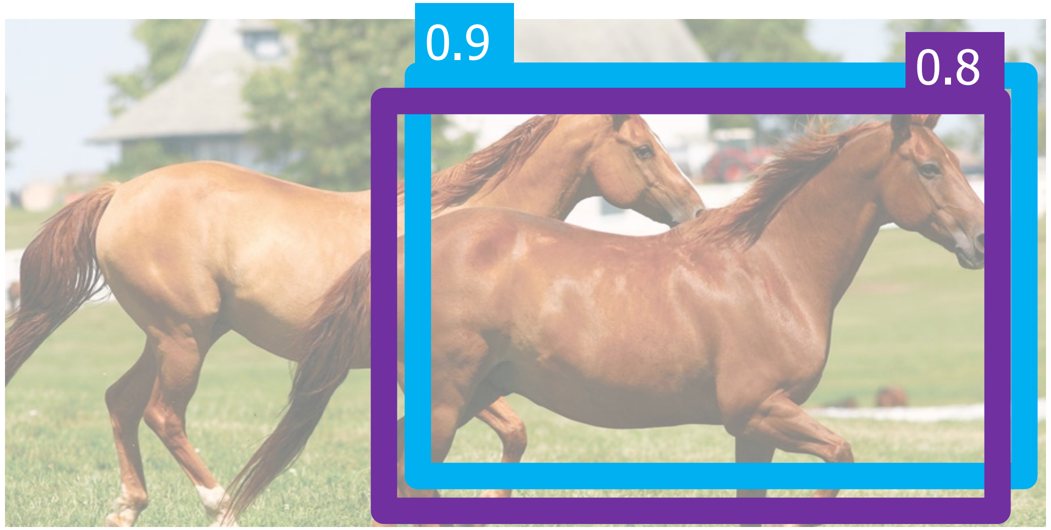
Average precision

true pos. /
total detections



true positive detections /
total positive test instances

Non-maximum suppression



- Subtlety: we predict a bounding box for every sliding window. Which ones should we keep?
- Keep only "peaks" in detector response.
- Discard low-prob boxes near high-prob ones
- Often use a simple greedy algorithm

Non-maximum suppression

Greedy algorithm, run on each class independently

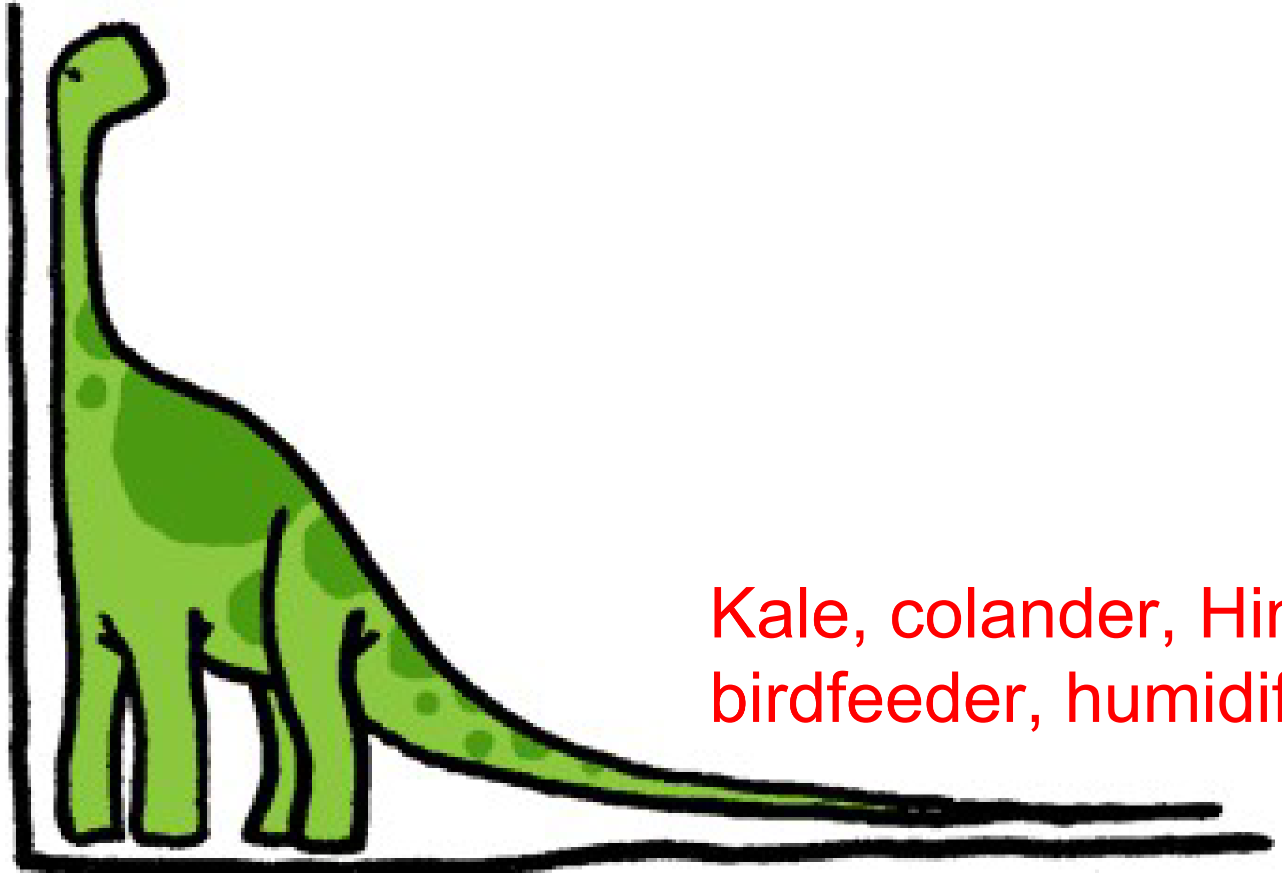
let be the set of all bounding boxes
let be the set of detections we'll keep, $D = \varnothing$
while $A \neq \varnothing$:
remove the box with highest probability from
if doesn't significantly overlap with an existing box in (e.g. IoU > 0.5):
$D = D \cup \{x\}$
return

- Introduction to scene understanding
- Object detection models
- Evaluating object detectors
- **Challenges**

Handle the long tail of the distribution

Person, dog, table, …

Frequency



Kale, colander, Himalayan Salt, birdfeeder, humidifier, …

Object categories

Handle the “long tail” of the distribution

From COCO (80 categories)
[Lin et al., 2014]

LVIS dataset (1000+ categories)
“Few shot” (e.g. < 20 examples)
[Gupta et al., 2019]

Image source: R. Girshick

OWL-ViT: Open-vocabulary object detection model

[Submitted on 12 May 2022 (v1), last revised 20 Jul 2022 (this version, v2)]

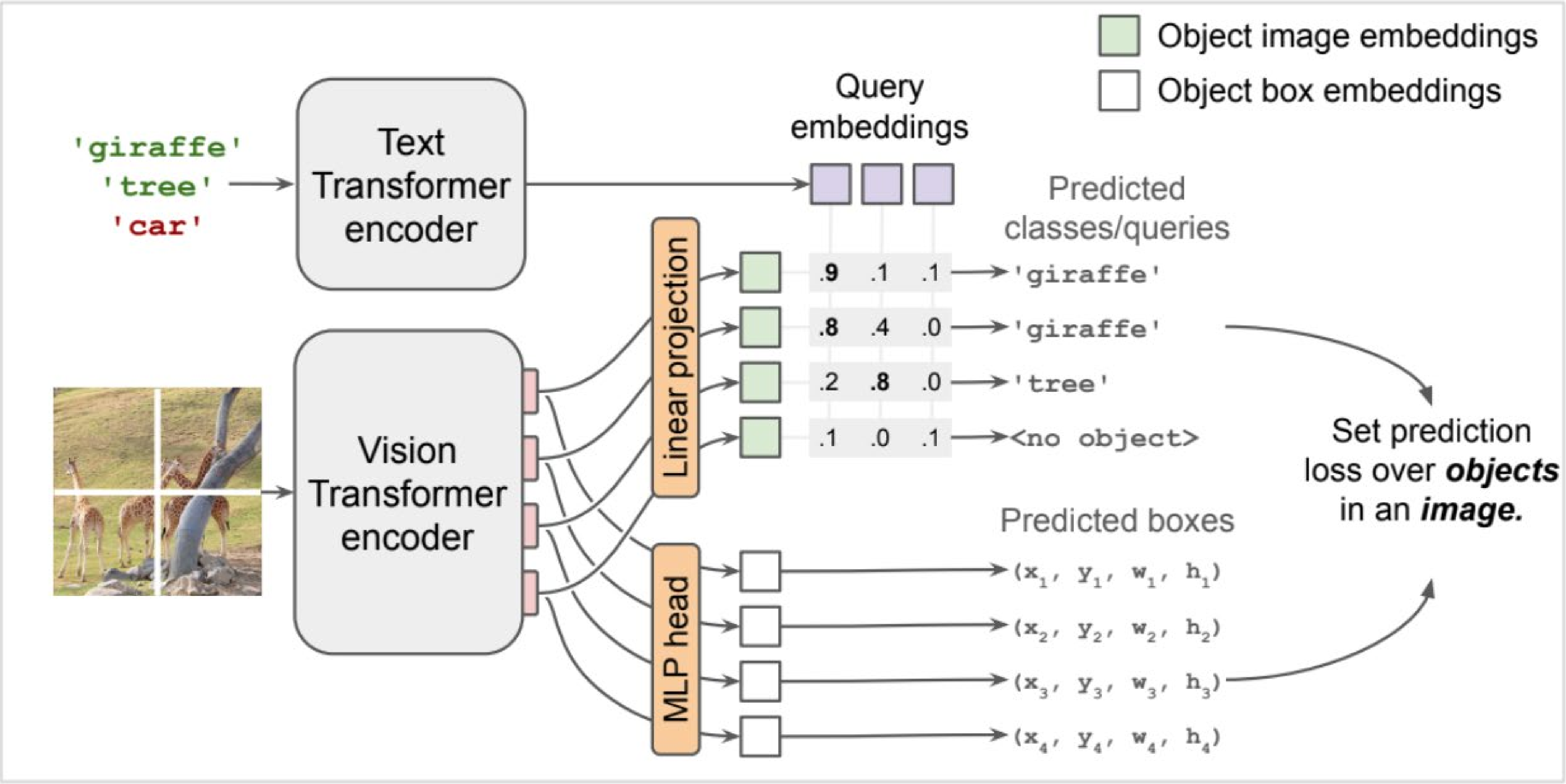
Simple Open-Vocabulary Object Detection with Vision Transformers

Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, Neil Houlsby

Combining simple architectures with large-scale pre-training has led to massive improvements in image classification. For object detection, pre-training and scaling approaches are less well established, especially in the long-tailed and open-vocabulary setting, where training data is relatively scarce. In this paper, we propose a strong recipe for transferring image-text models to open-vocabulary object detection. We use a standard Vision Transformer architecture with minimal modifications, contrastive image-text pre-training, and end-to-end detection fine-tuning. Our analysis of the scaling properties of this setup shows that increasing image-level pre-training and model size yield consistent improvements on the downstream detection task. We provide the adaptation strategies and regularizations needed to attain very strong performance on zero-shot text-conditioned and one-shot image-conditioned object detection. Code and models are available on GitHub.

Comments: ECCV 2022 camera-ready version

OWL-ViT: Open-vocabulary object detection model



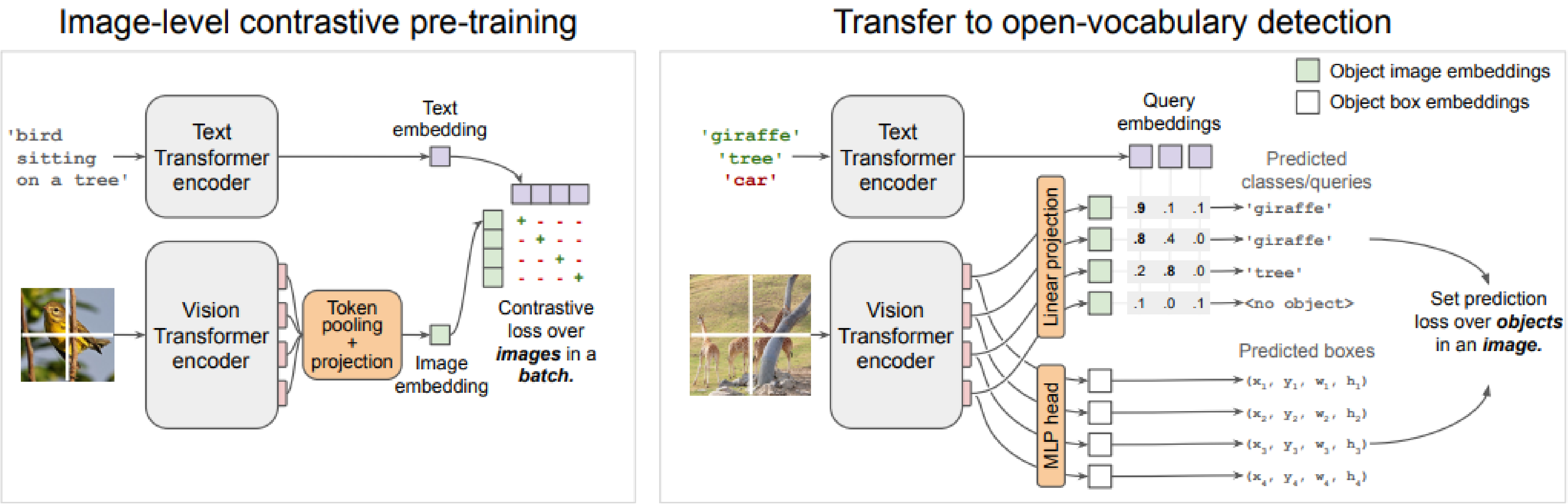


Fig. 1. Overview of our method. *Left:* We first pre-train an image and text encoder contrastively using image-text pairs, similar to CLIP [33], ALIGN [19], and LiT [44]. *Right:* We then transfer the pre-trained encoders to open-vocabulary object detection by removing token pooling and attaching light-weight object classification and localization heads directly to the image encoder output tokens. To achieve open-vocabulary detection, query strings are embedded with the text encoder and used for classification. The model is fine-tuned on standard detection datasets. At inference time, we can use text-derived embeddings for open-vocabulary detection, or image-derived embeddings for few-shot image-conditioned detection.

OWL-ViT Demo

https://colab.research.google.com/github/huggingface/notebooks/blob/main/examples/zeroshot_object_detection_with_owlvit.ipynb

https://colab.research.google.com/drive/1evZkcsq4FTreFxGV6JXDmymnYcqWK43n