Deep learning for object detection

Outline

1. Introduction

2. Brief history

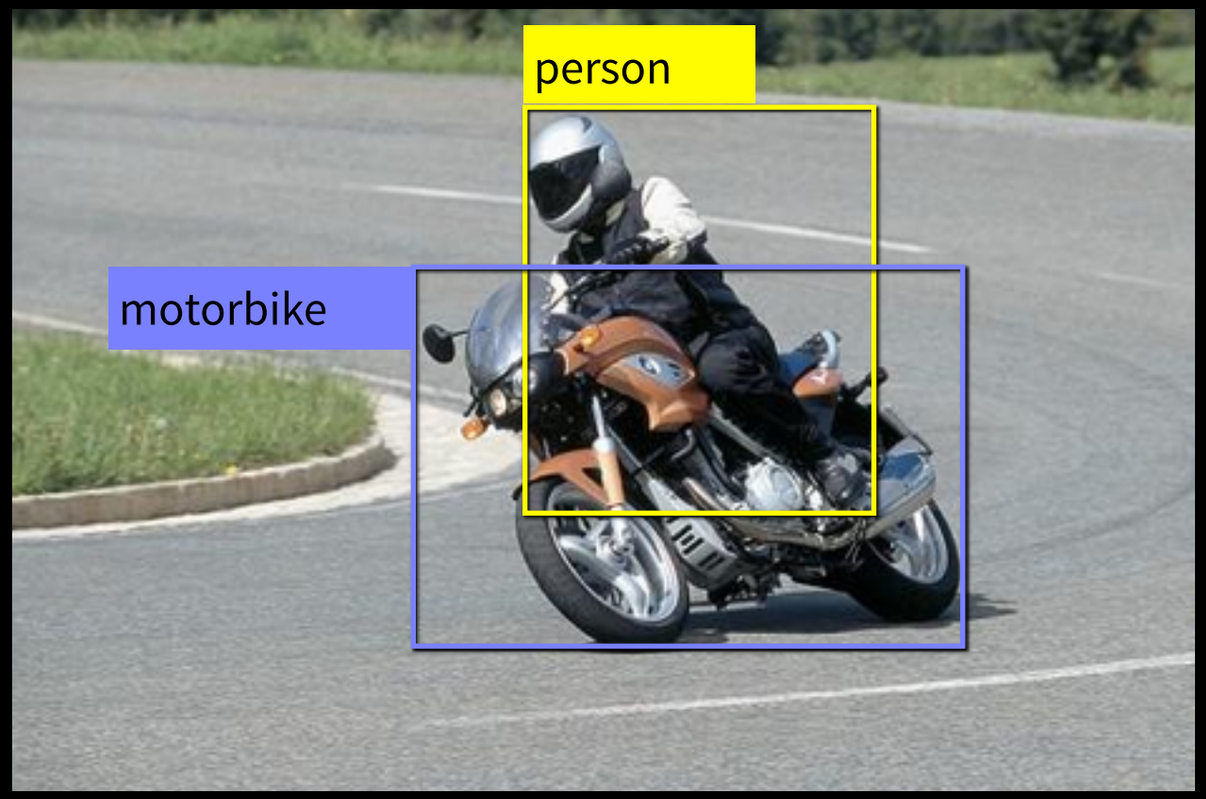
3. Case study: R-CNN & Caffe

What is object detection?

Problem formulation

{ airplane, bird, motorbike, person, sofa }

Input



Desired output

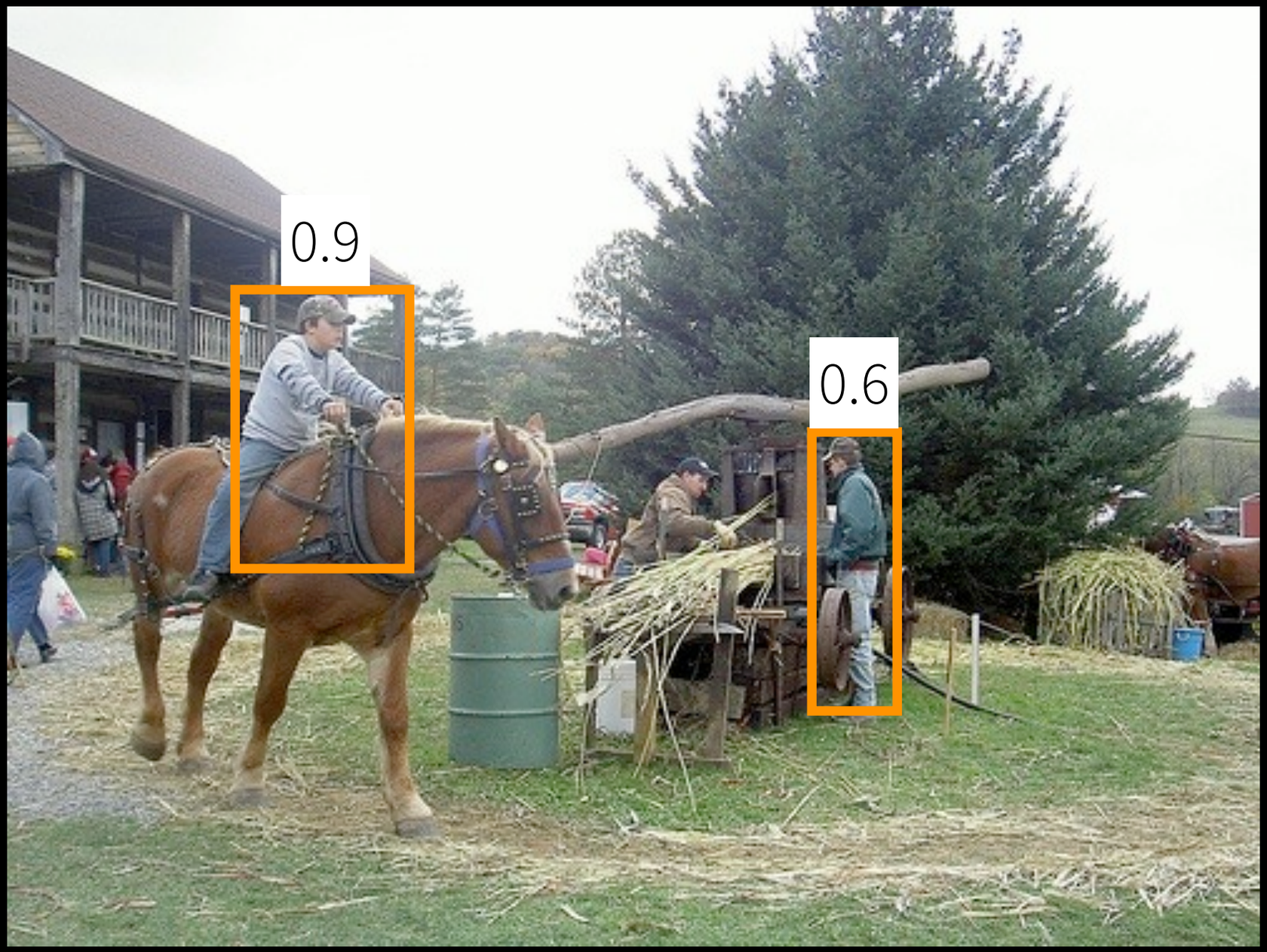
Evaluating a detector

Test image (previously unseen)

First detection ...

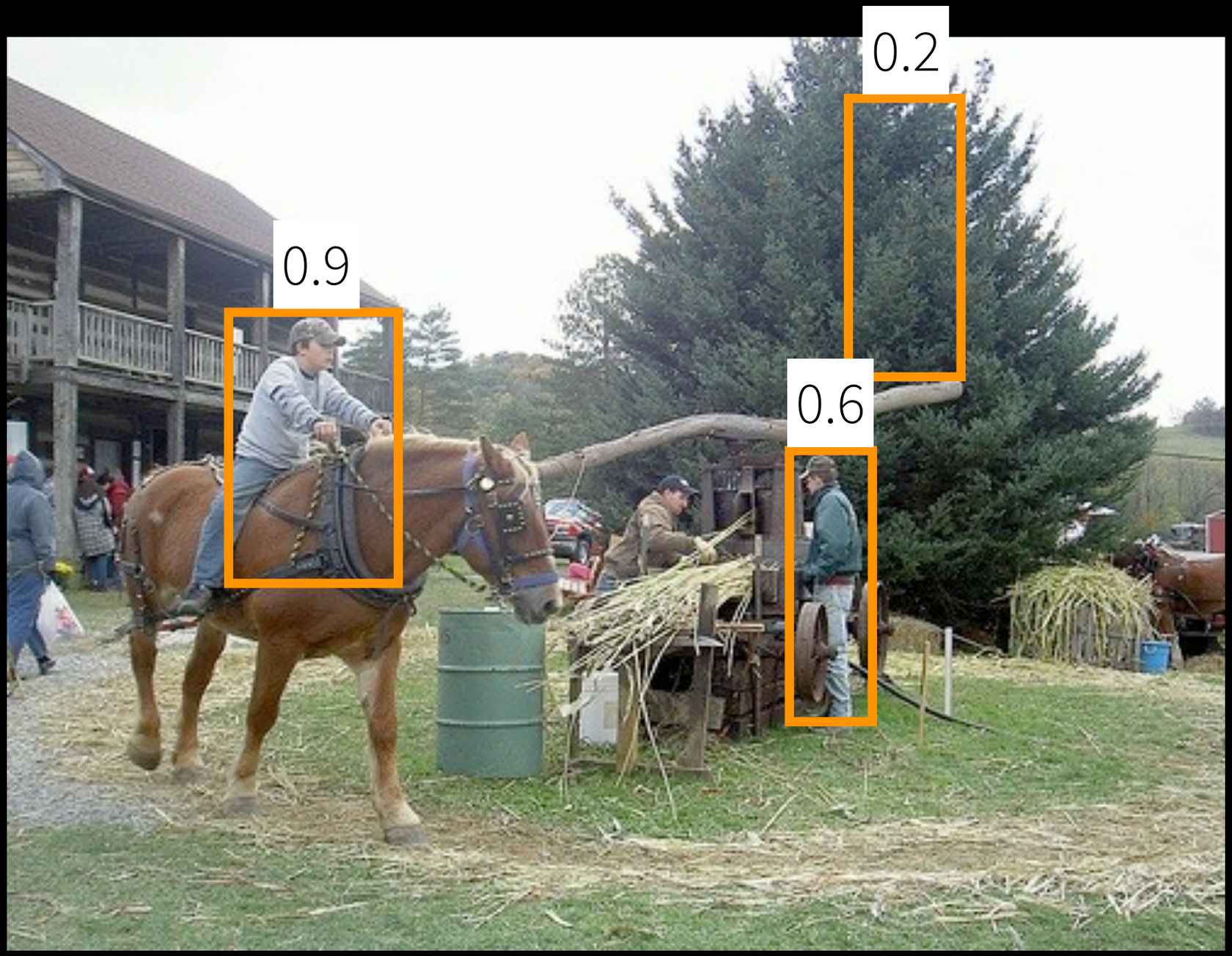
‘person’ detector predictions

Second detection ...



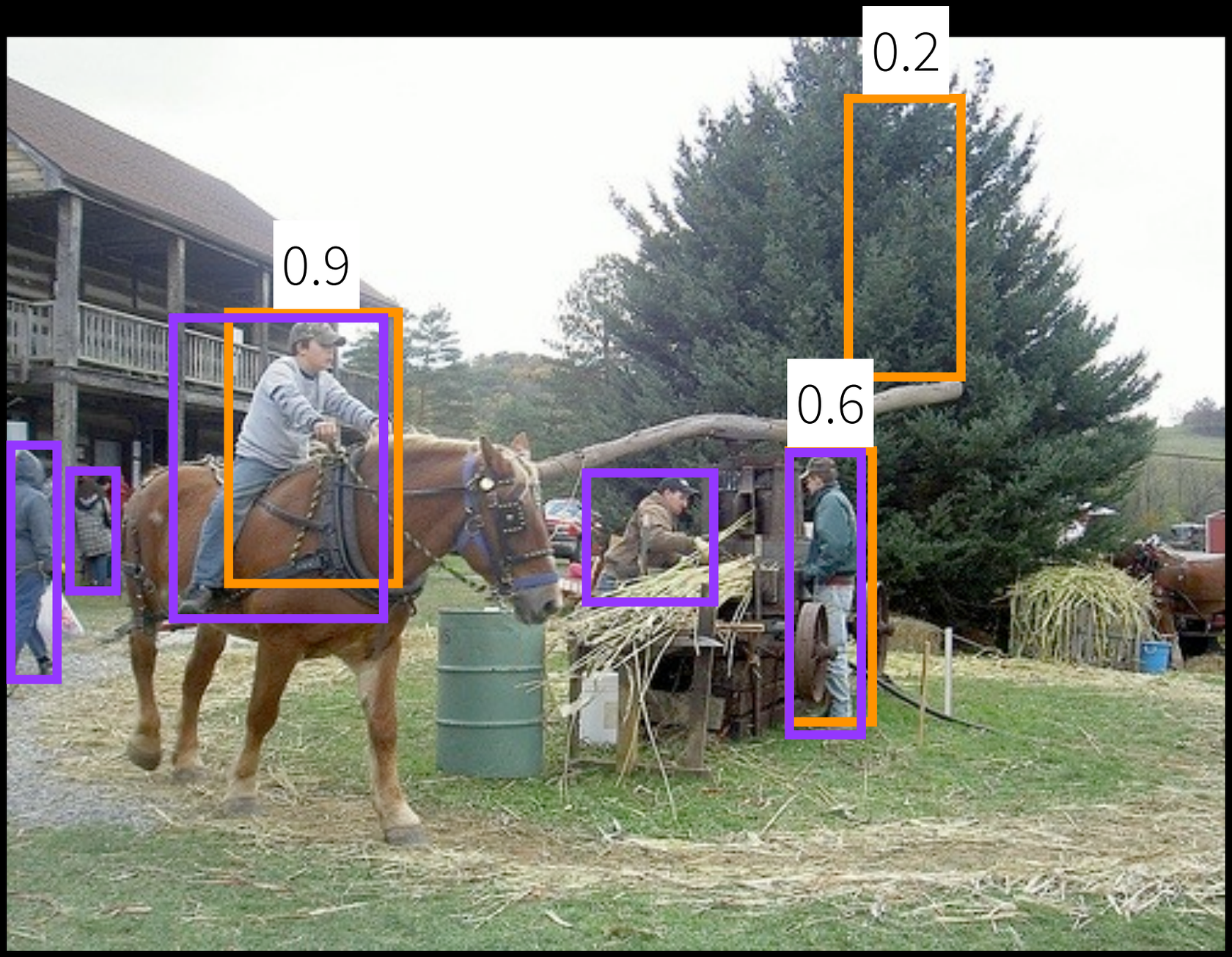
‘person’ detector predictions

Third detection ...



‘person’ detector predictions

Compare to ground truth



‘person’ detector predictions

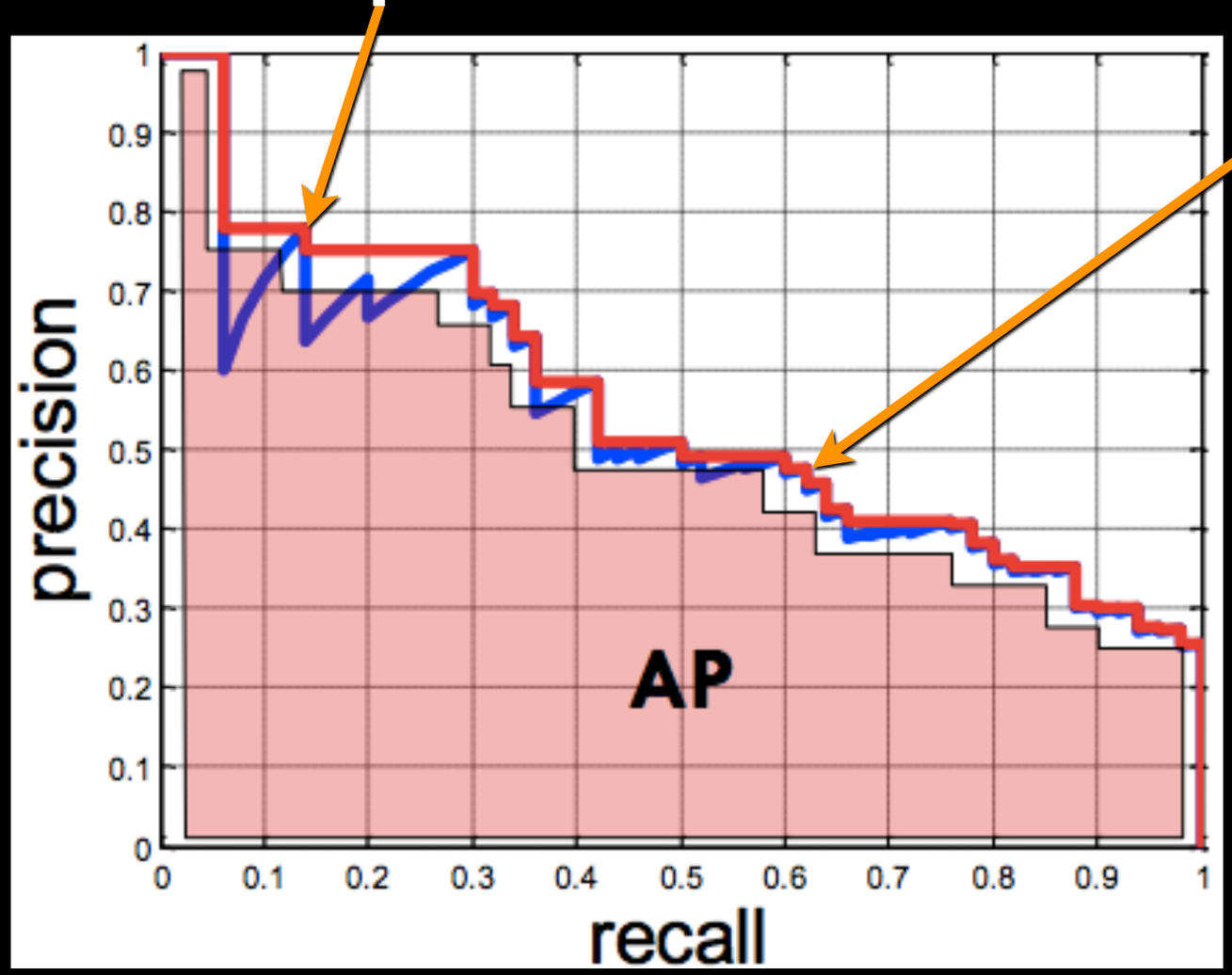
ground truth ‘person’ boxes

Sort by confidence

true positive (high IoU overlap)

false positive (low IoU overlap or duplicate)

Evaluation metric



Average Precision (**AP**)

0% is worst

100% is best

mean AP over classes (**mAP**)

Datasets

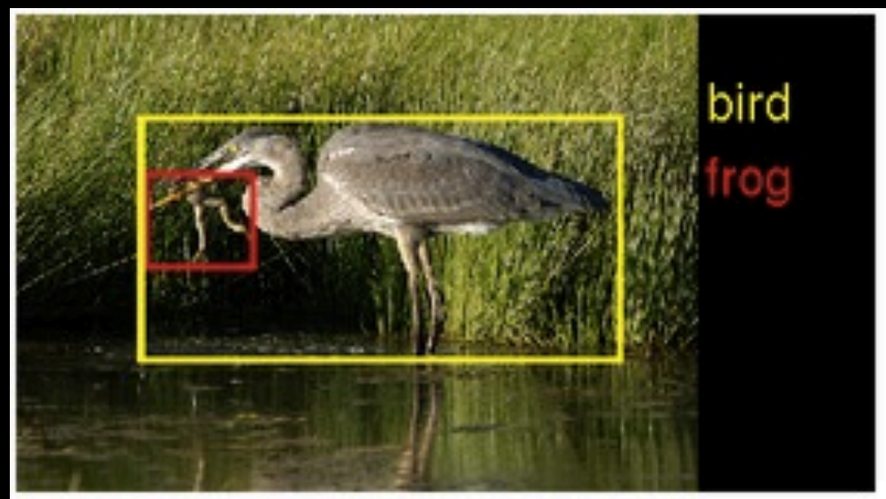
PASCAL VOC Challenge

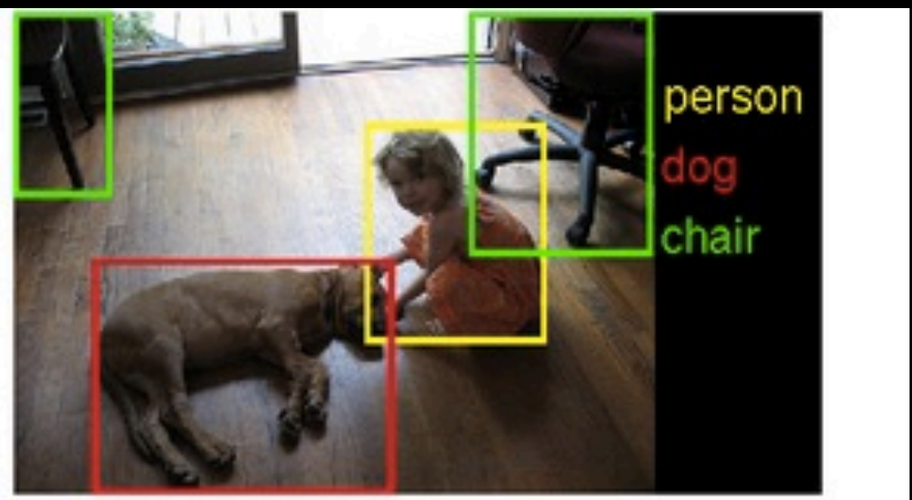
Dataset: 22k images, 50k objects, 20 classes

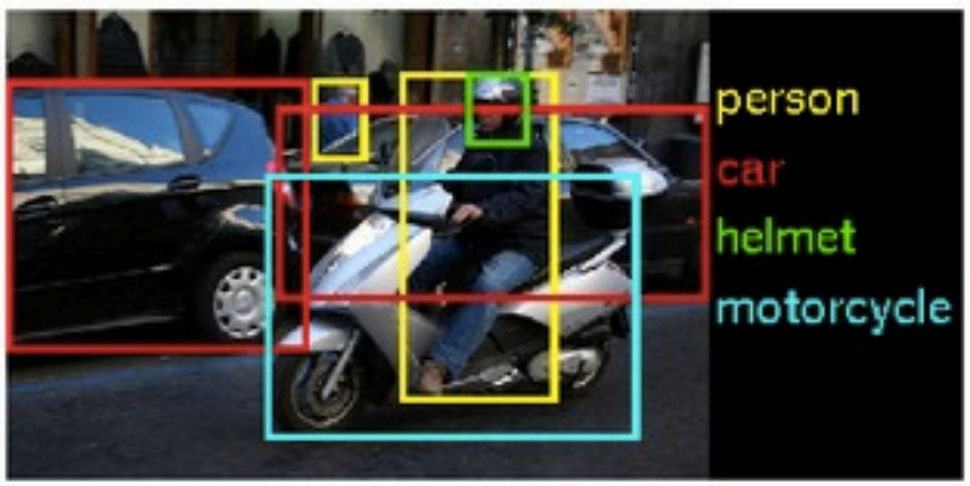
Detect: people, horses, sofas, bicycles, pottedplants, ...

ImageNet detection (ILSVRC)

Dataset: 470k images, 530k objects, 200 classes

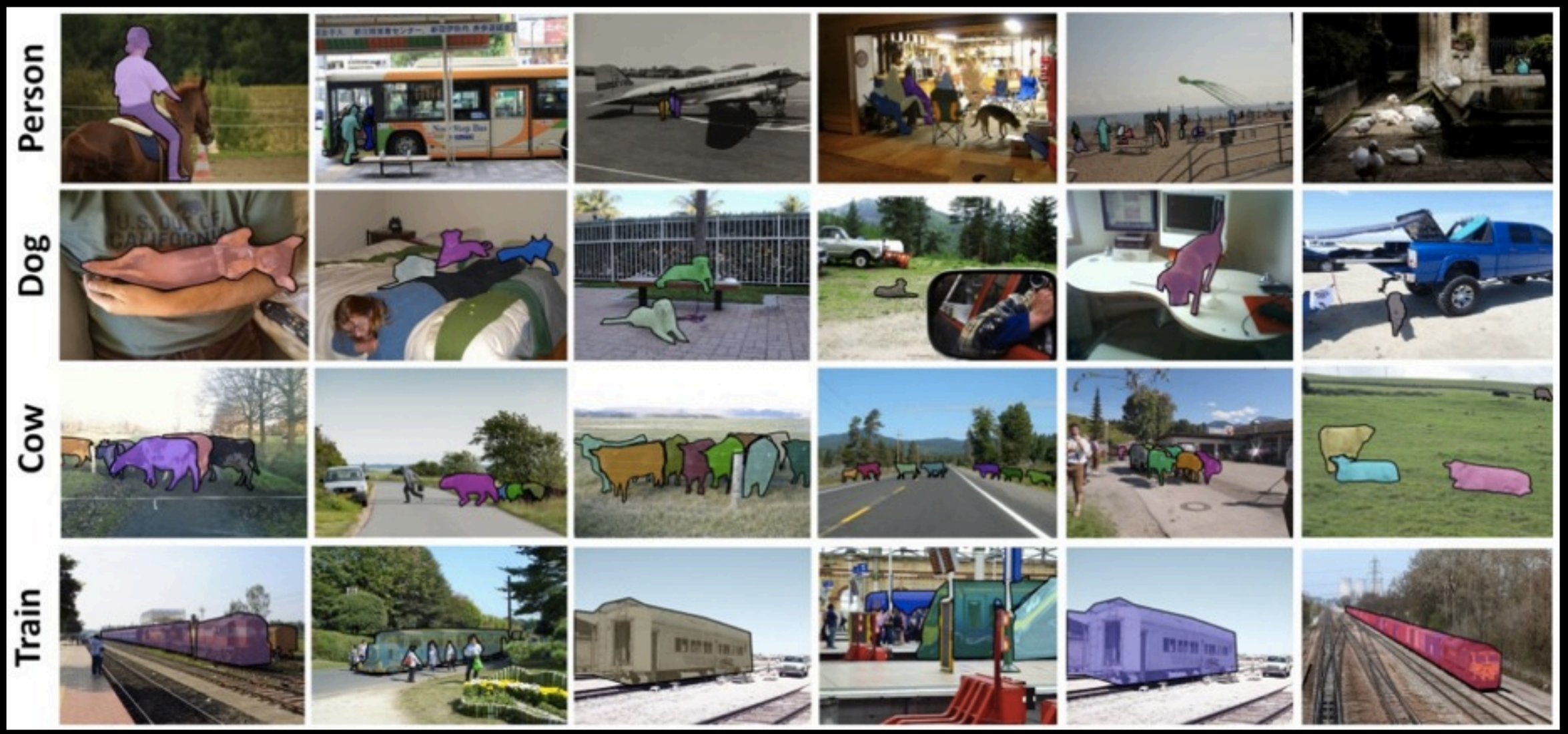






New: Microsoft COCO

Dataset: ~300k images, ~**2M** objects, 91 classes

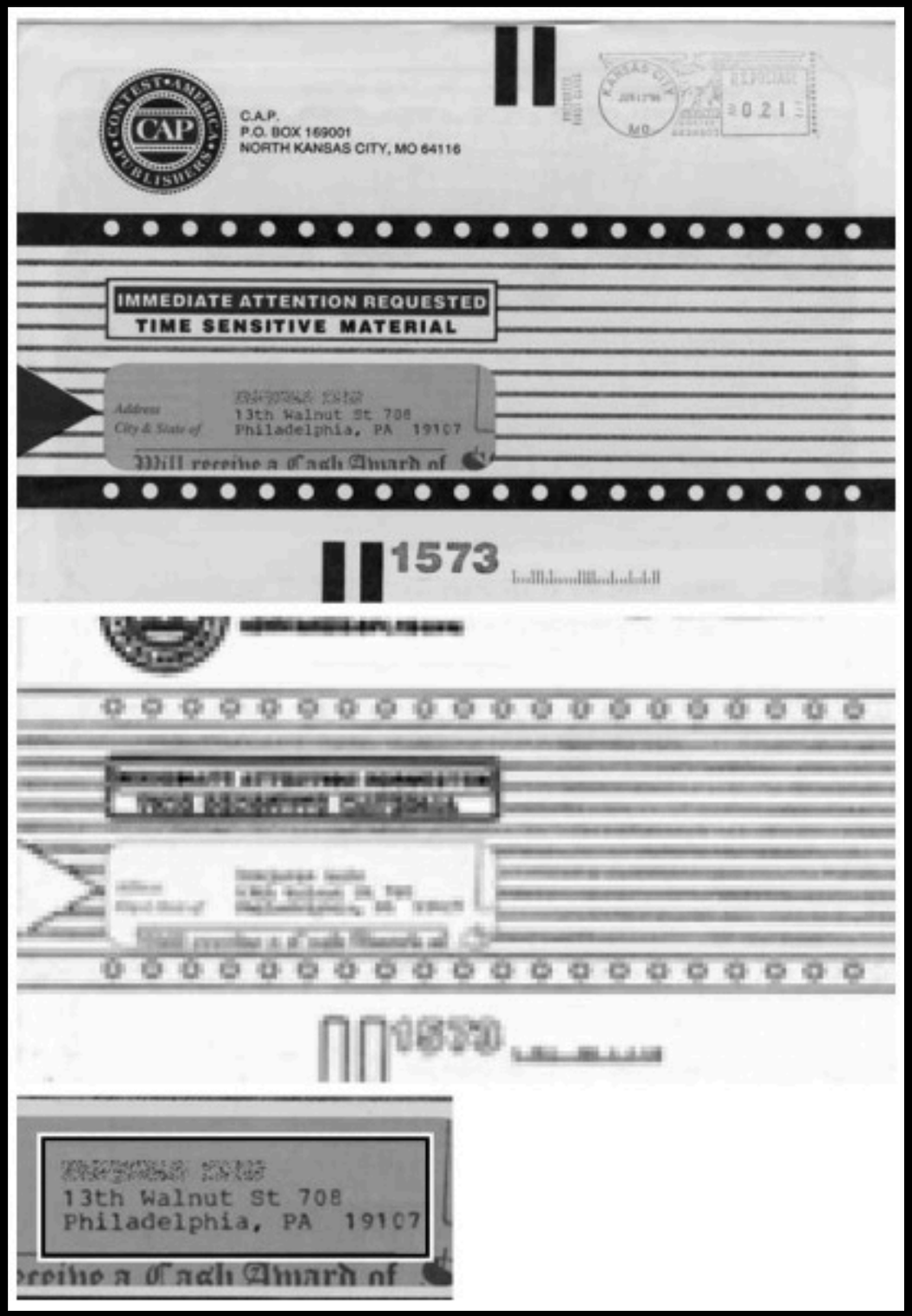


Talk: at 17:57 on Wednesday (Oral Session 3B)

Brief history

Early work on object detection:
neural networks (the first wave)

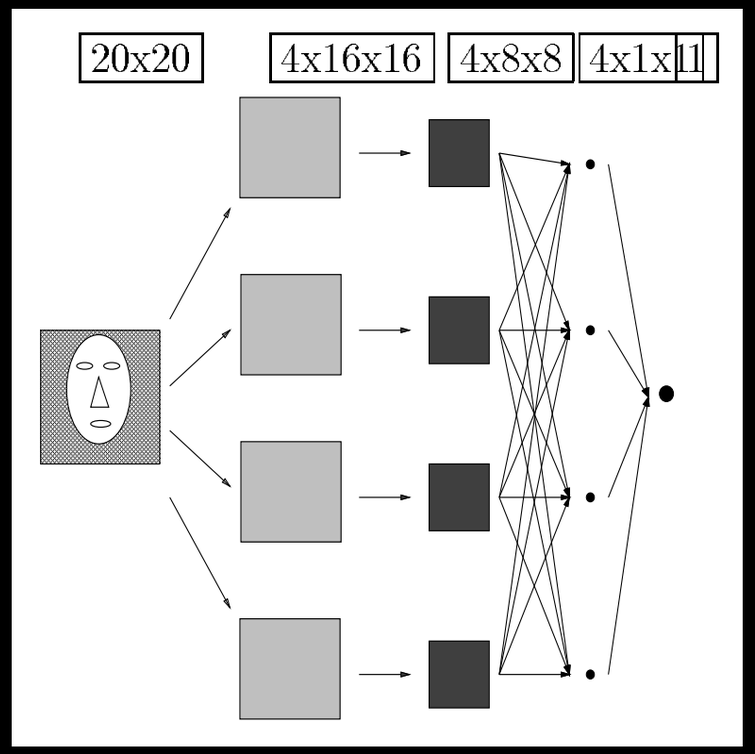
Early object detection work



Postal Address Block Location Using a Convolutional Locator Network

Wolf & Platt 1992

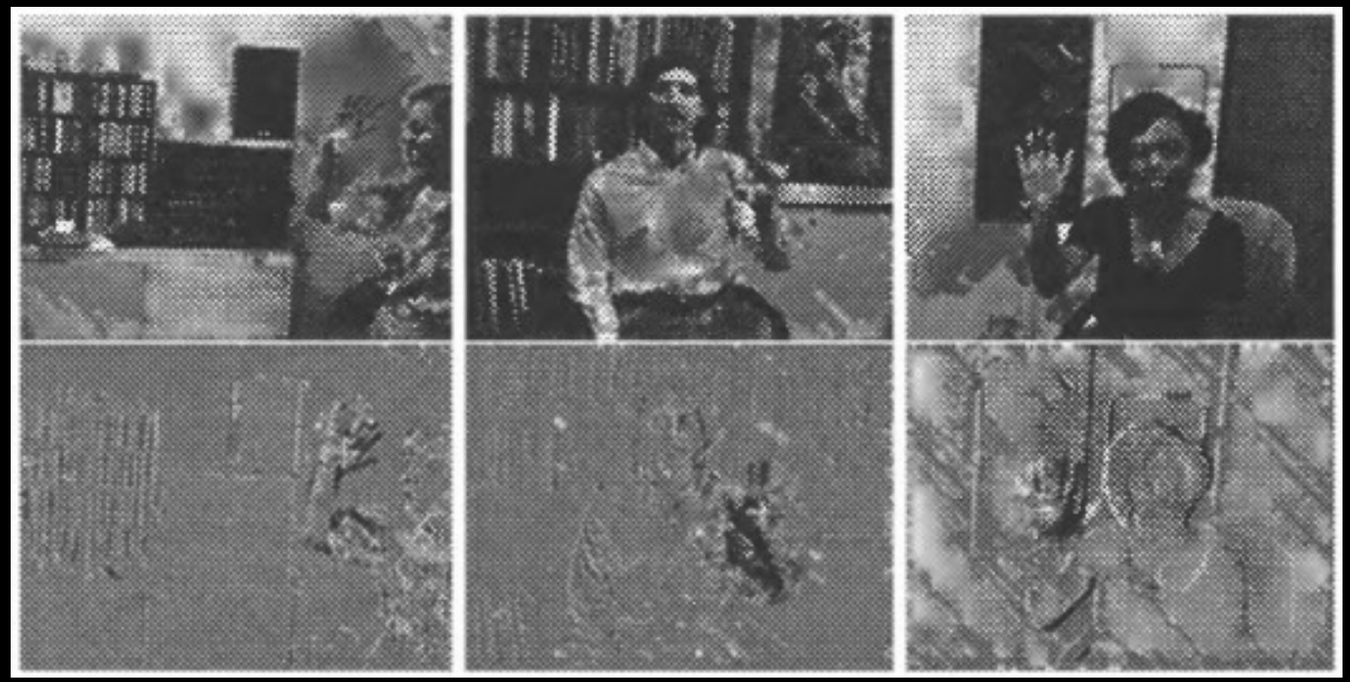
Early object detection work



Original Approach for the Localization of Objects in Images

Vaillant, Monrocq, LeCun 1994

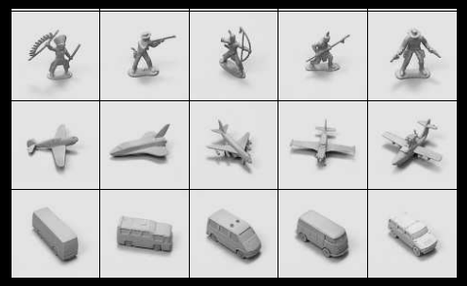
Early object detection work



A Convolutional Neural Network Hand Tracker

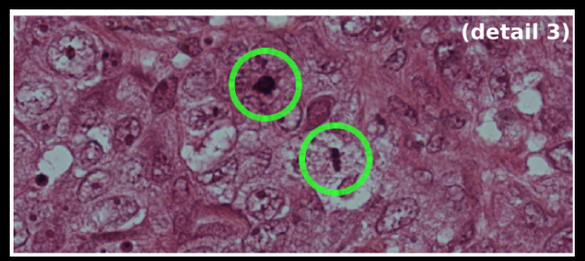
Nowlan & Platt 1995

Ongoing efforts with neural nets



LeCun, Huang, Bottou 2004

NORB dataset

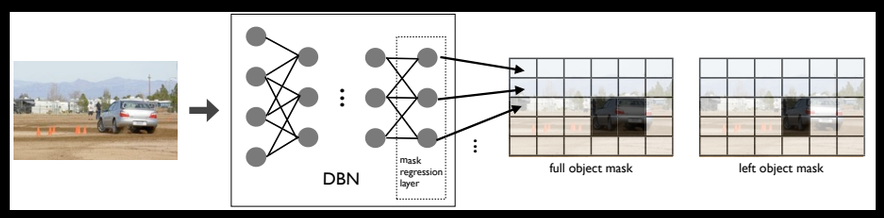


Cireşan et al. 2013

Mitosis detection

Sermanet et al. 2013

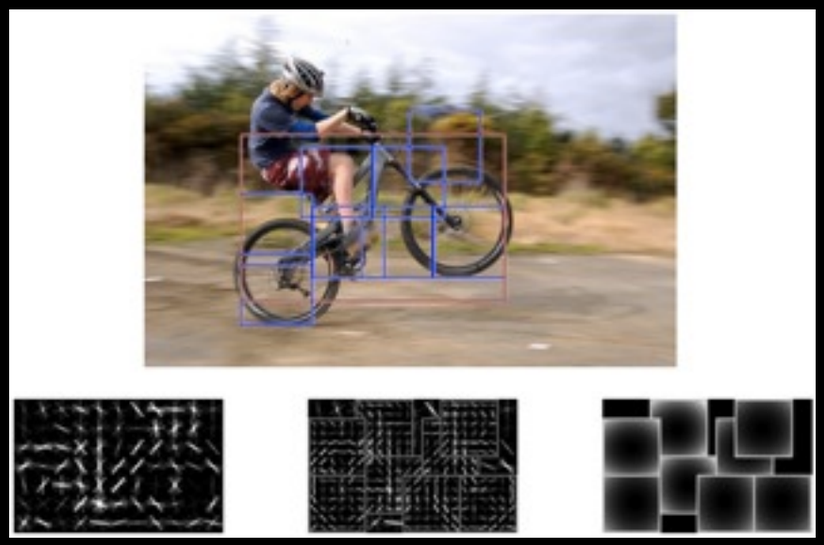
Pedestrian detection



Szegedy, Toshev, Erhan 2013

PASCAL detection (VOC'07 mAP 30.5%)

Non-neural network methods



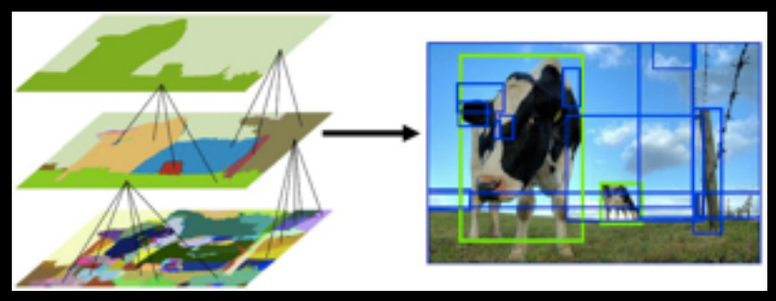
Deformable Part Models (DPM)

Felzenszwalb, Girshick, McAllester, Ramanan

type	example	$f(z)$	$g(x,y)$	eval. complexity
linear	linear	z	xy	$O(N)$
quasi-lin.	χ^2	z	$\frac{2xy}{(x+y)}$	$O(BN)$
non-lin.	RBF-χ^2	$e^{-\gamma z}$	$\frac{(x-y)^2}{(x+y)}$	$O(MBN)$

Multiple Kernels for Object Detection

Vedaldi, Gulshan, Varma, Zisserman

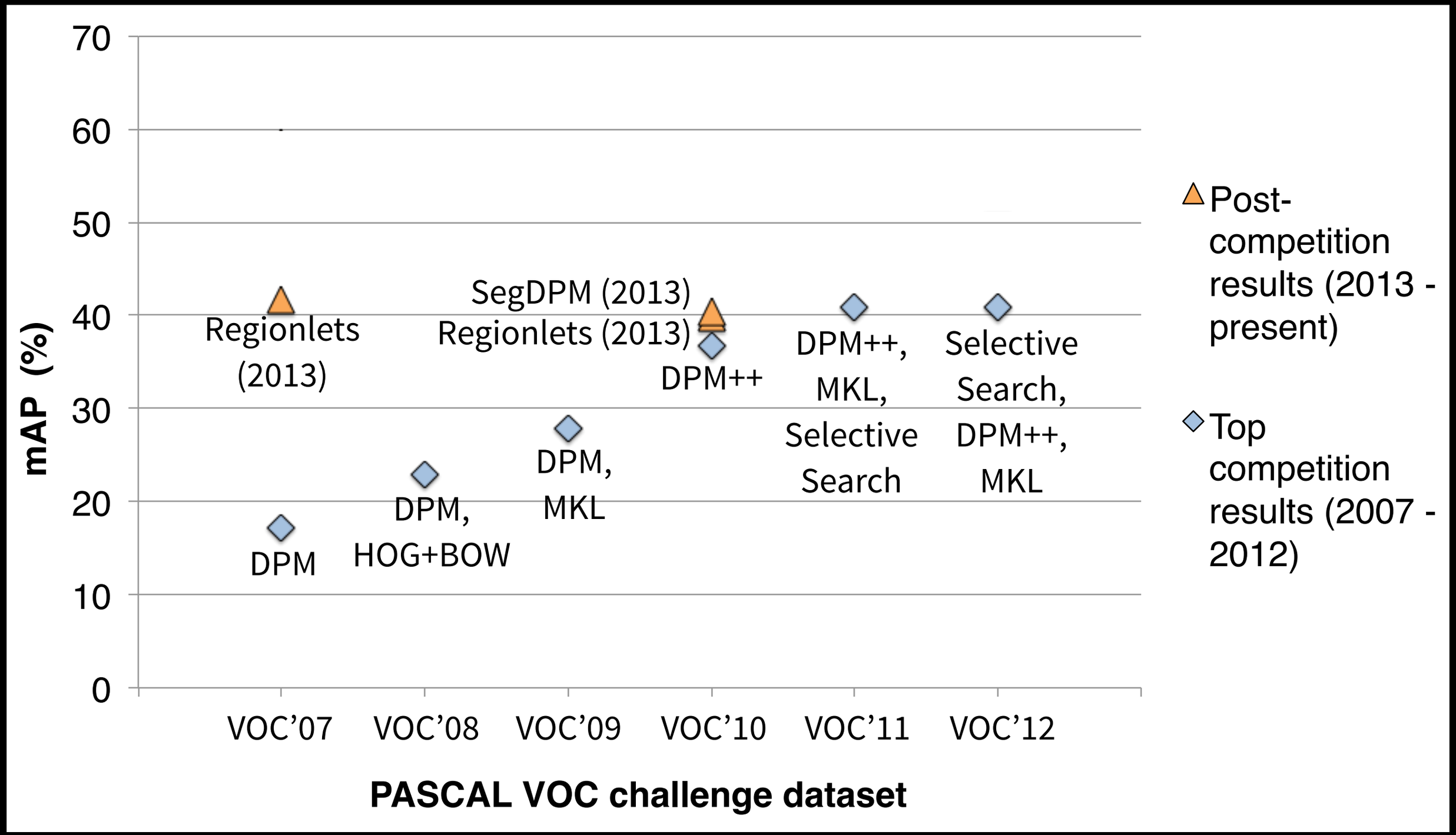


Selective Search for Object Recognition

Uijlings, van de Sande, Gevers, Smeulders

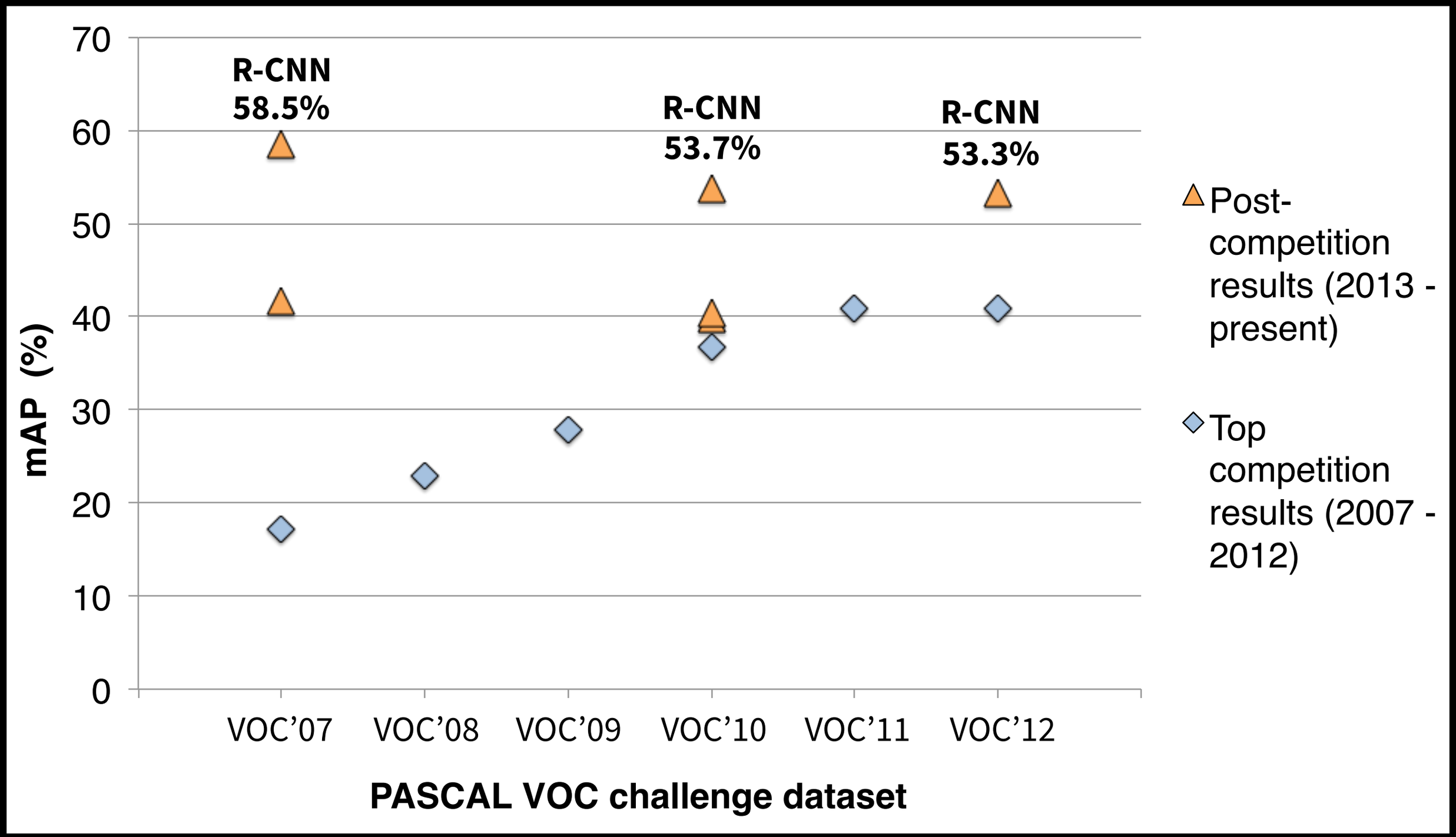
Outperformed deep learning methods until 2013

Non-neural network methods



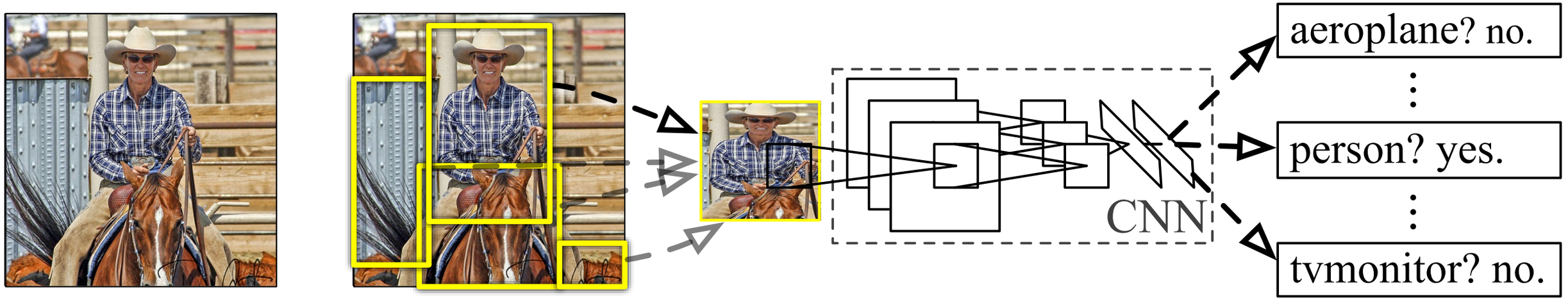
[Regionlets. Wang et al. ICCV'13] [SegDPM. Fidler et al. CVPR'13]

R-CNN: Region-based CNN



[Girshick, Donahue, Darrell, Malik CVPR 14]

R-CNN: Region-based CNN



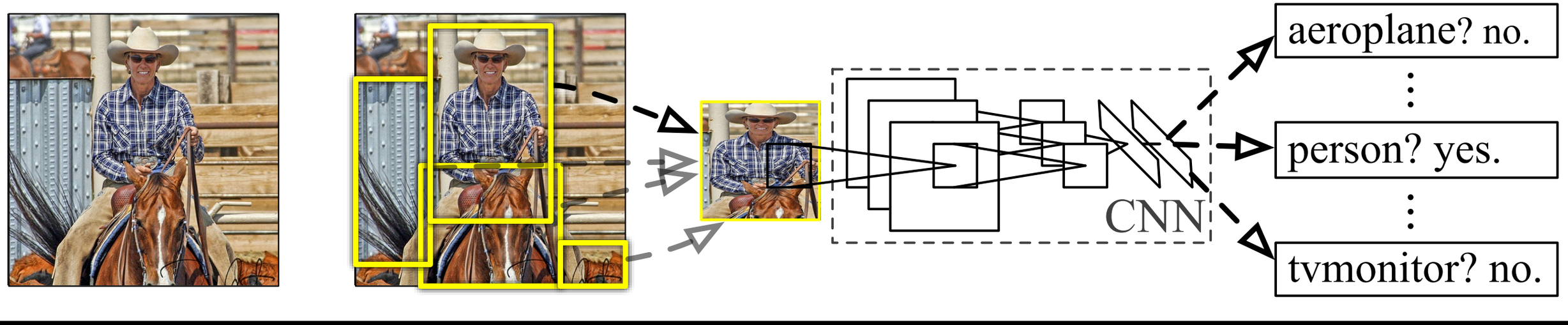
Input image

Extract region proposals (~2k / image)

Compute CNN features

Classify regions (linear SVM)

R-CNN at test time: Step 1



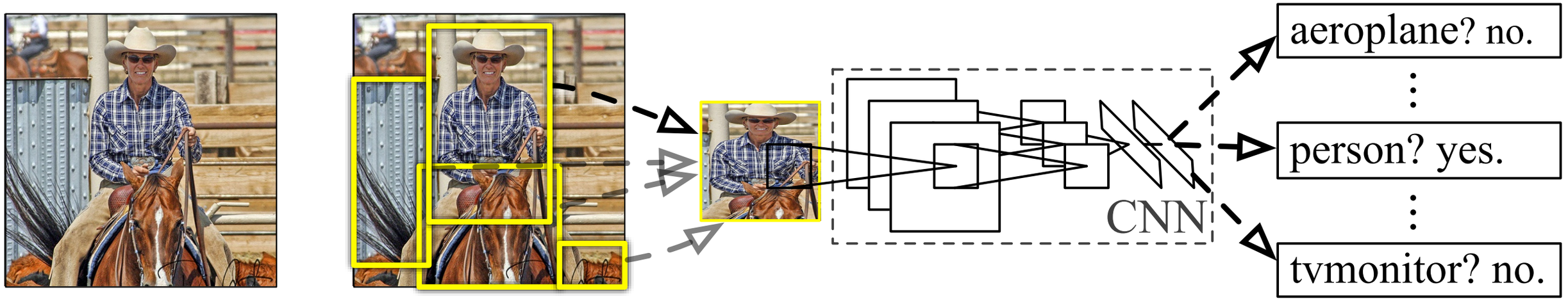
Input image

Extract region proposals (~2k / image)

Proposal-method agnostic, many choices

- Selective Search [van de Sande, Uijlings et al.] (Used in this work)
- Objectness [Alexe et al.]
- Category independent object proposals [Endres & Hoiem]
- CPMC [Carreira & Sminchisescu]
- MCG [Arbelaez et al.]
- RIGOR [Humayun et al.]
- Geodesic Object Proposals [Krähenbühl & Koltun]
- *Sliding-window*

R-CNN at test time: Step 2

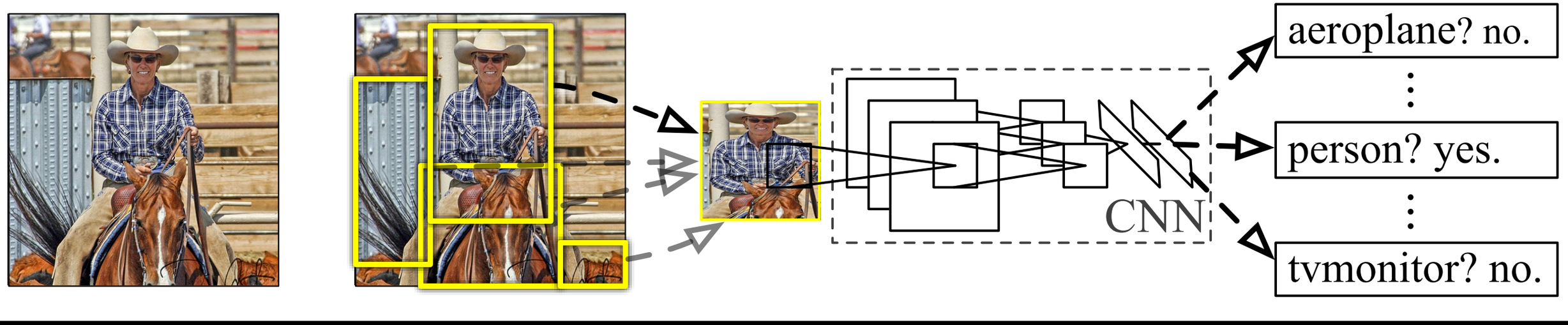


Input image

Extract region proposals (~2k / image)

Compute CNN features

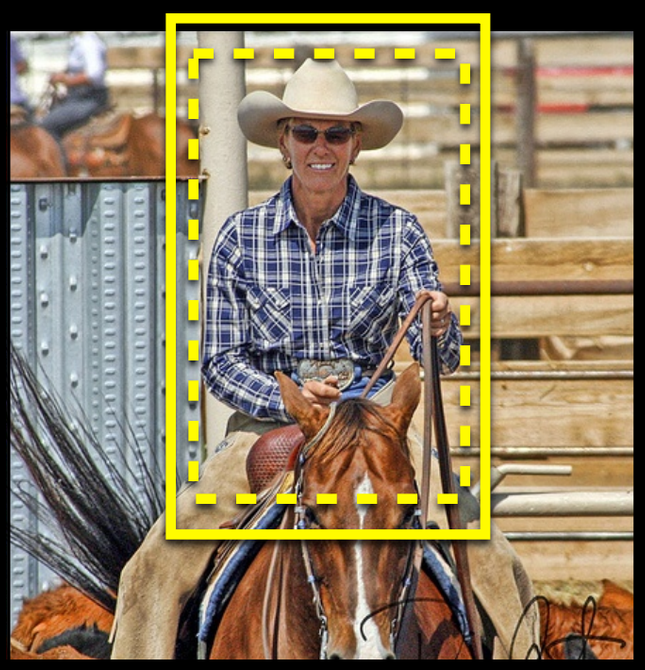
R-CNN at test time: Step 2



Input image

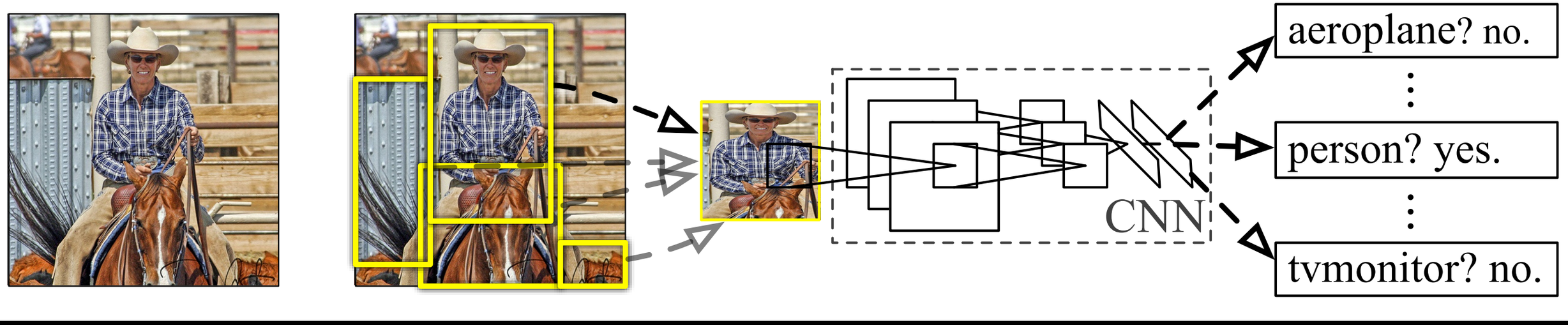
Extract region proposals (~2k / image)

Compute CNN features



Dilate proposal

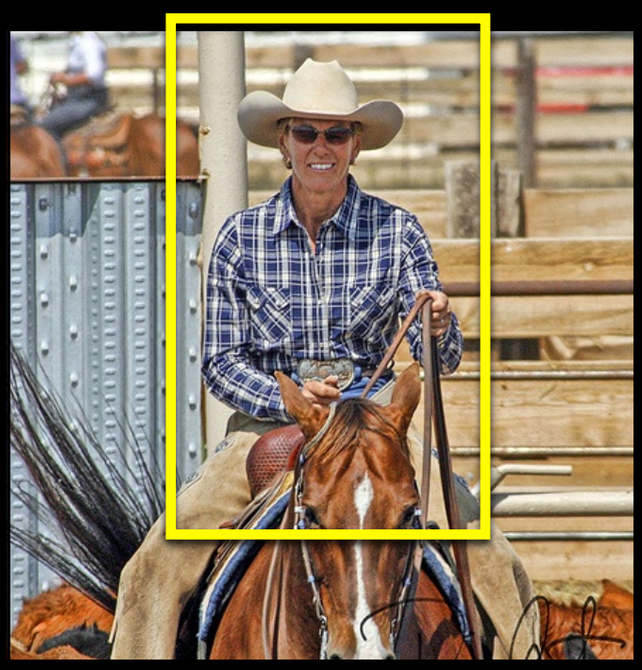
R-CNN at test time: Step 2



Input image

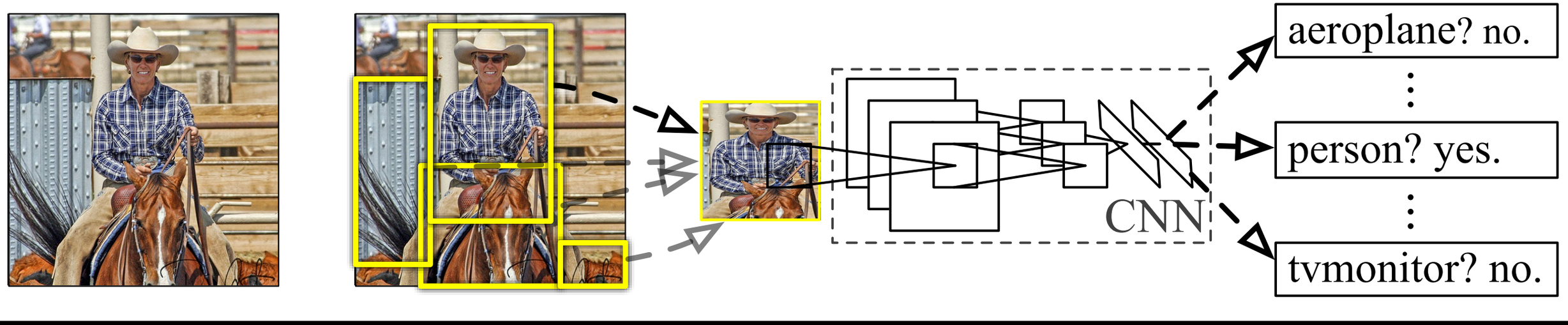
Extract region proposals (~2k / image)

Compute CNN features



a. Crop

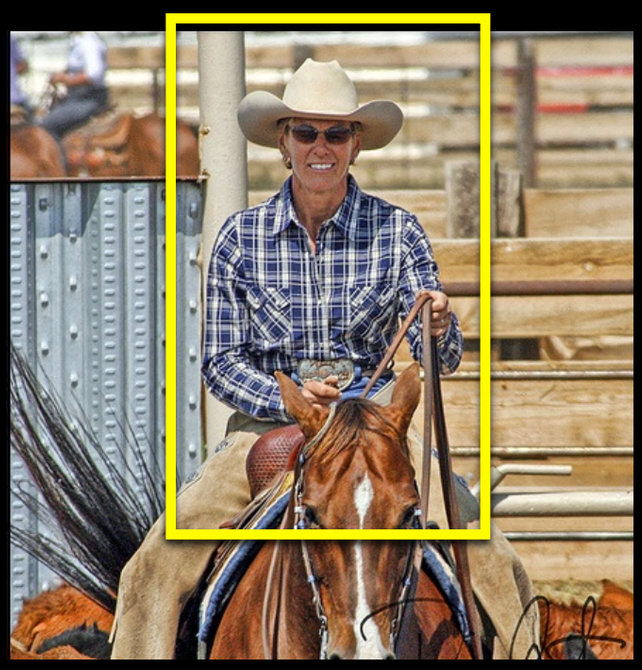
R-CNN at test time: Step 2



Input image

Extract region proposals (~2k / image)

Compute CNN features

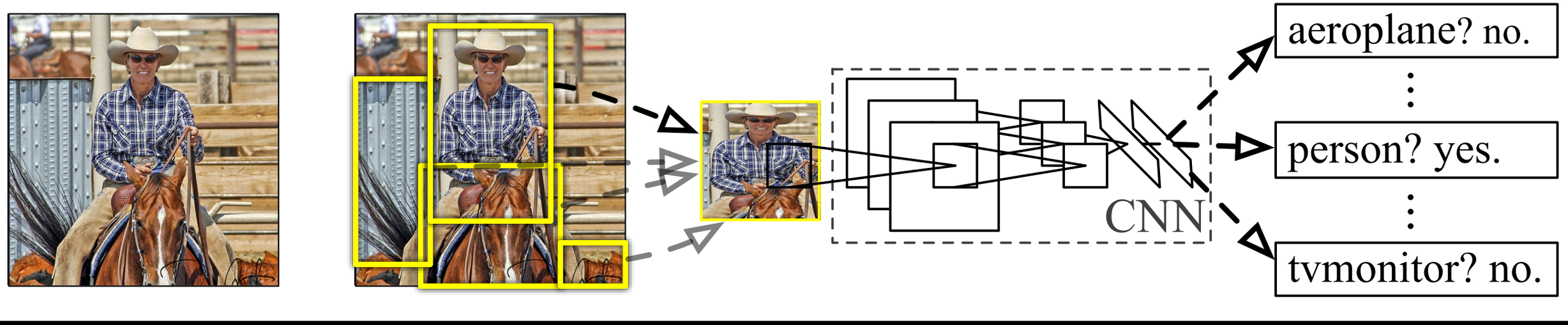


a. Crop

227 x 227

b. Scale (anisotropic)

R-CNN at test time: Step 2



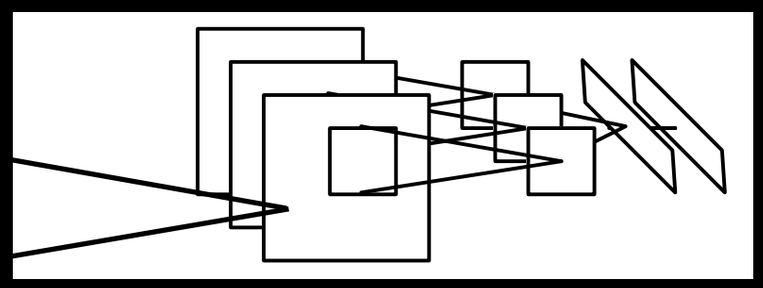
Input image

Extract region proposals (~2k / image)

Compute CNN features

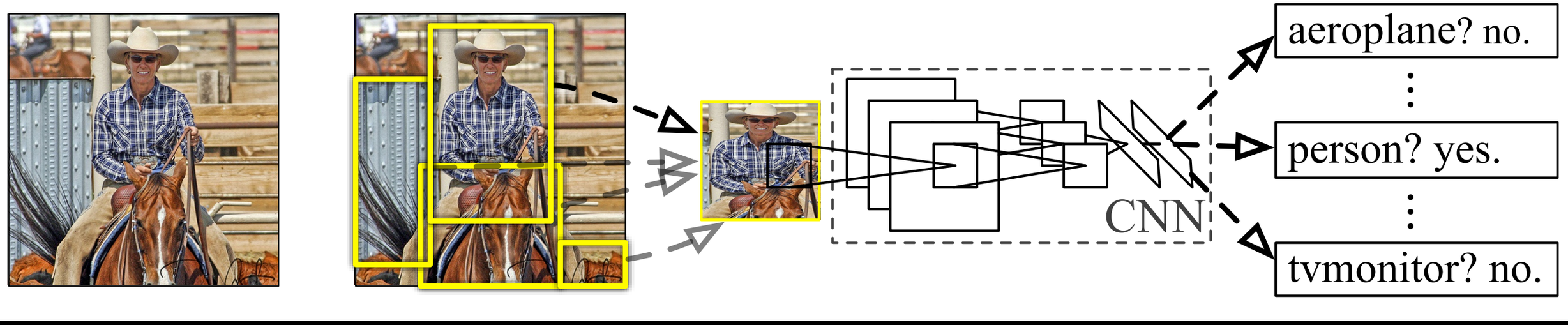
. Crop

b. Scale (anisotropic)



c. Forward propagate
Output: “fc_7” features

R-CNN at test time: Step 3

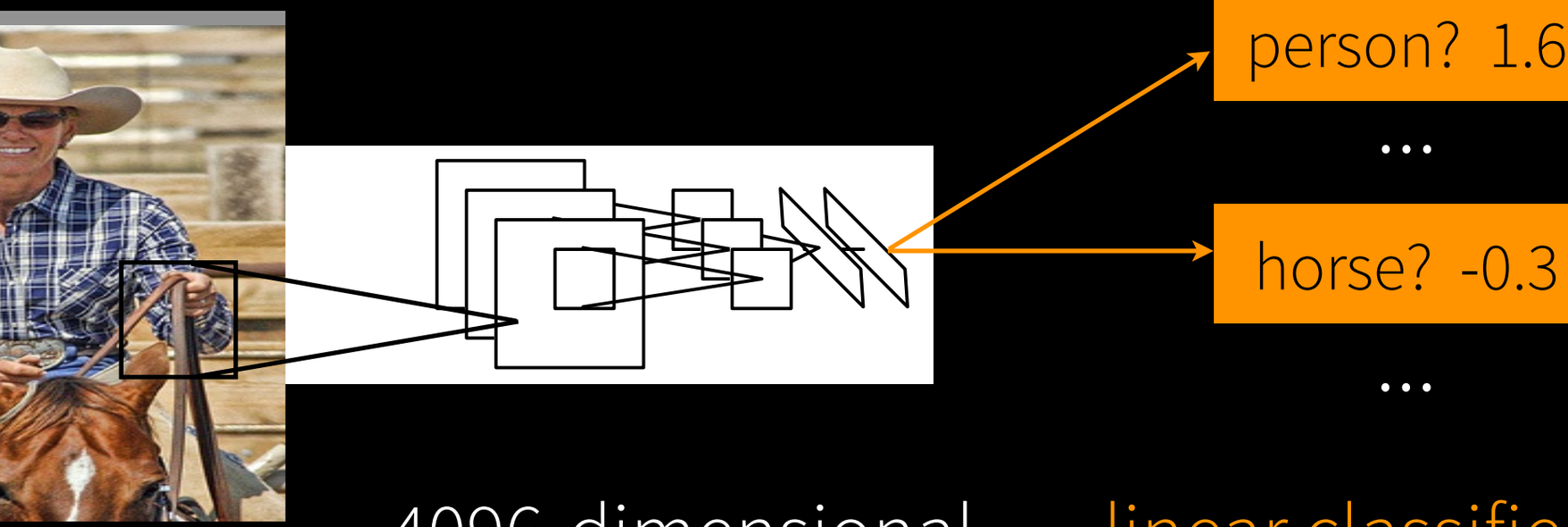


Input image

Extract region proposals (~2k / image)

Compute CNN features

Classify regions



proposal

4096-dimensional fc_7 feature vector

linear classifiers (SVM or softmax)

R-CNNs on ILSVRC

Basis of top-performing entries in ILSVRC14 detection

— GoogLeNet is an R-CNN

— Ensemble of 6 very deep CNNs

— See ILSVRC workshop on Friday

Training an R-CNN

Bounding-box labeled detection data is scarce

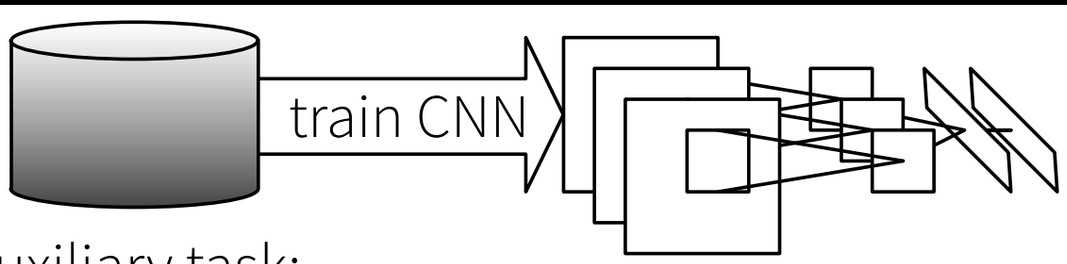
Key insight:

Use *supervised* pre-training on a data-rich auxiliary task and *transfer* to detection

R-CNN training: Step 1

Supervised pre-training

Train a CNN for the 1000-way ILSVRC image classification task

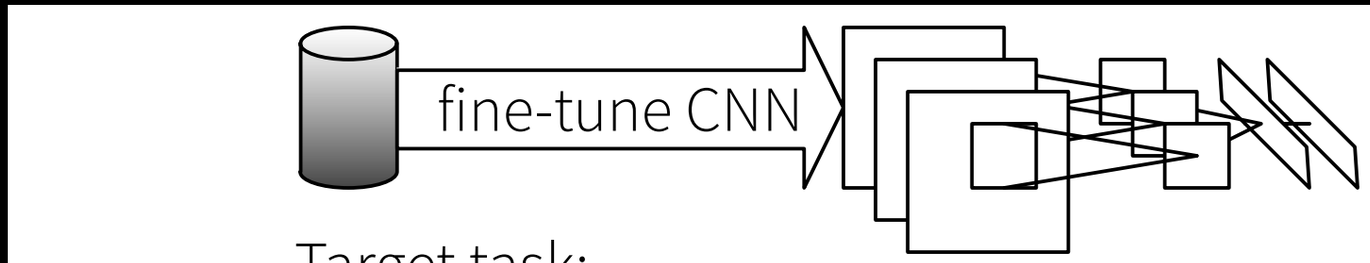


Auxiliary task:
ILSVRC 2012 *classification*
(1.2 million images)

R-CNN training: Step 2

Fine-tune the CNN for detection

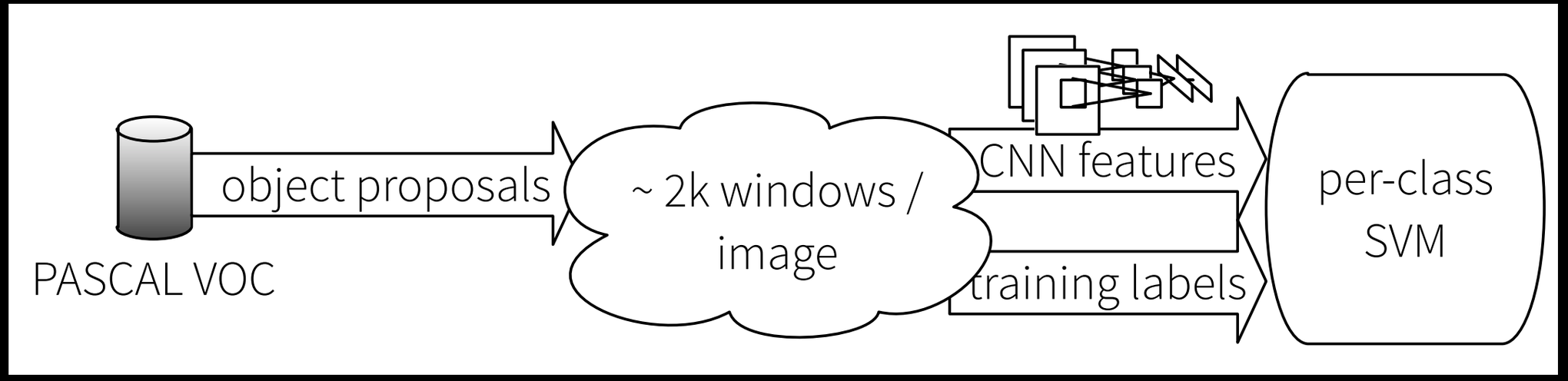
Transfer the representation learned for ILSVRC classification to PASCAL (or ImageNet detection)



Target task:
PASCAL VOC detection
(~25k object labels)

R-CNN training: Step 3

Train detection SVMs
(With the softmax classifier from fine-tuning
mAP decreases from 54% to 51%)



Ignore SVMs for now, we'll just focus on the Caffe bits

R-CNN fine-tuning (step 2)

Three changes relative to the reference model

1. Learning rates

2. Input layer

3. Last fully-connected layer (fc8)

Caffe details

Fine-tuning command

```
$ caffe train \
      -solver pascal_finetune_solver.prototxt \
      -weights /path/to/net.caffemodel
```

Smaller base learning rate

```
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 100000
display: 20
max_iter: 450000
```

Solver for Caffe ImageNet reference

```
base_lr: 0.001
lr_policy: "step"
gamma: 0.1
stepsize: 20000
display: 20
max_iter: 100000
```

Solver for R-CNN fine-tuning

Input: Window Data Layer

```
layers {
  name: "data"
  type: WINDOW_DATA
  top: "data"
  top: "label"
  window_data_param {
    source: "window_file_2007_trainval.txt"
    mean_file: "../../data/ilsvrc12/imagenet_mean.binaryproto"
    batch_size: 128
    crop_size: 227
    mirror: true
    fg_threshold: 0.5
    bg_threshold: 0.5
    fg_fraction: 0.25
    context_pad: 16
    crop_mode: "warp"
  }
}
```

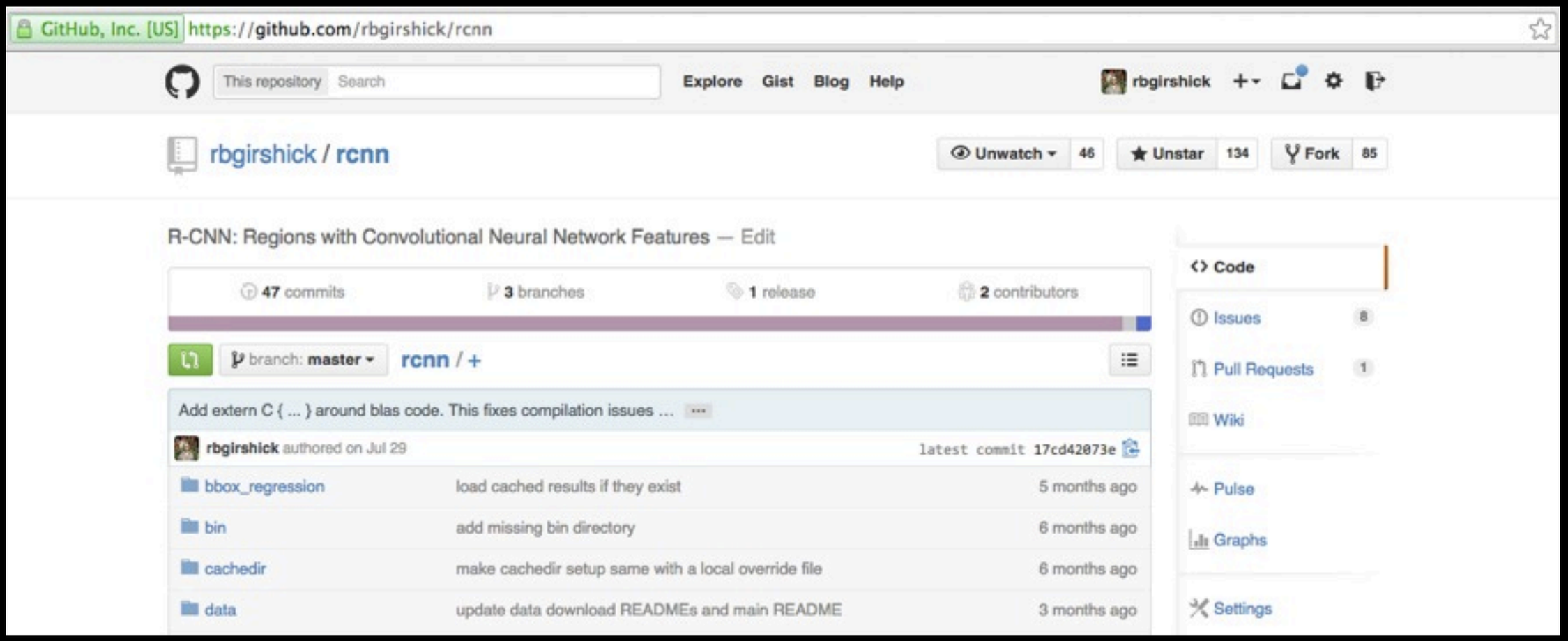
Window File Format

Documented in src/caffe/layers/window_data_layer.cpp

```
// window_file format
// repeated:
//    # image_index
//    img_path (abs path)
//    channels
//    height
//    width
//    num_windows
//    class_index overlap x1 y1 x2 y2
```

```
# 0
/work4/rbg/VOC2007/VOCdevkit/VOC2007/JPEGImages/000001.jpg
3
500
353
1830
12 1.000 47 239 194 370
15 1.000 7 11 351 497
...
```

Generating window files



https://github.com/rbgirshick/rcnn

finetuning/rcnn_make_window_file.m

Last weight layer: fc8_pascal

```
layers {
  name: "fc8"
  type: INNER_PRODUCT
  bottom: "fc7"
  top: "fc8"
  blobs_lr: 1
  blobs_lr: 2
  weight_decay: 1
  weight_decay: 0
  inner_product_param {
    num_output: 1000
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

Caffe ImageNet reference model

```
layers {
  name: "fc8_pascal"
  type: INNER_PRODUCT
  bottom: "fc7"
  top: "fc8_pascal"
  blobs_lr: 10
  blobs_lr: 20
  weight_decay: 1
  weight_decay: 0
  inner_product_param {
    num_output: 21
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

Swap in new fc8 layer for PASCAL

Tip: two-stage fine-tuning

Stage 1: set blobs_lr: 0 for all layers except fc8_pascal
— optimize to initialize fc8_pascal
— solves a convex problem

Stage 2: set blobs_lr to default values (1 for weights and 2 for biases) for all weight layers
— optimize all parameters

Improved Bird Species Recognition Using Pose Normalized Deep Convolutional Nets
Branson, Van Horn, Perona, Belongie. BMVC 2014.

Some empirical observations

@ Thursday poster session 4B

Analyzing The Performance of Multi Layer Neural Networks for Object Recognition

Pulkit Agrawal, Ross Girshick, Jitendra Malik

More pre-training is better

of pre-training iterations

	50k	105k	205k	305k
SUN-CLS	53.0 ± 0.2	54.6 ± 0.1	56.3 ± 0.2	56.6 ± 0.2
PASCAL-DET	50.2	52.6	55.3	55.4

One might think pre-training for too long could hurt generalization to a new dataset

Training for detection from scratch

SUN-CLS			PASCAL-DET		
scratch	pre-train	fine-tune	scratch	pre-train	fine-tune
40.4 ± 0.2	53.1 ± 0.2	56.8 ± 0.2	40.7	45.5	54.1

no pre-training

just pre-training

pre-train & fine-tune

Training from scratch on VOC 2007 trainval -> ok performance (would have been s.o.t.a. in early 2013)

No fine-tuning

SUN-CLS			PASCAL-DET		
scratch	pre-train	fine-tune	scratch	pre-train	fine-tune
40.4 ± 0.2	53.1 ± 0.2	56.8 ± 0.2	40.7	45.5	54.1

no pre-training

just pre-training

pre-train & fine-tune

Using the ImageNet pre-trained model and training SVMs works better than from scratch

Fine-tuning

SUN-CLS			PASCAL-DET		
scratch	pre-train	fine-tune	scratch	pre-train	fine-tune
40.4 ± 0.2	53.1 ± 0.2	56.8 ± 0.2	40.7	45.5	54.1

no pre-training

just pre-training

pre-train & fine-tune

Fine-tuning improves performance dramatically

What happens with more data?

SUN-CLS			PASCAL-DET			PASCAL-DET + DATA		
scratch	pre-train	fine-tune	scratch	pre-train	fine-tune	scratch	pre-train	fine-tune
40.4 ± 0.2	53.1 ± 0.2	56.8 ± 0.2	40.7	45.5	54.1	52.3	45.5	59.2

no pre-training

just pre-training

pre-train & fine-tune

Use VOC 2007 trainval + VOC 2012 trainval (~3x more data)

Training from scratch (with more data)

SUN-CLS			PASCAL-DET			PASCAL-DET + DATA		
scratch	pre-train	fine-tune	scratch	pre-train	fine-tune	scratch	pre-train	fine-tune
40.4 ± 0.2	53.1 ± 0.2	56.8 ± 0.2	40.7	45.5	54.1	52.3	45.5	59.2

no pre-training

just pre-training

pre-train & fine-tune

More data makes a big difference

The from-scratch model is almost as good as the old fine-tuned model

Fine-tuning (with more data)

SUN-CLS			PASCAL-DET			PASCAL-DET + DATA		
scratch	pre-train	fine-tune	scratch	pre-train	fine-tune	scratch	pre-train	fine-tune
40.4 ± 0.2	53.1 ± 0.2	56.8 ± 0.2	40.7	45.5	54.1	52.3	45.5	59.2

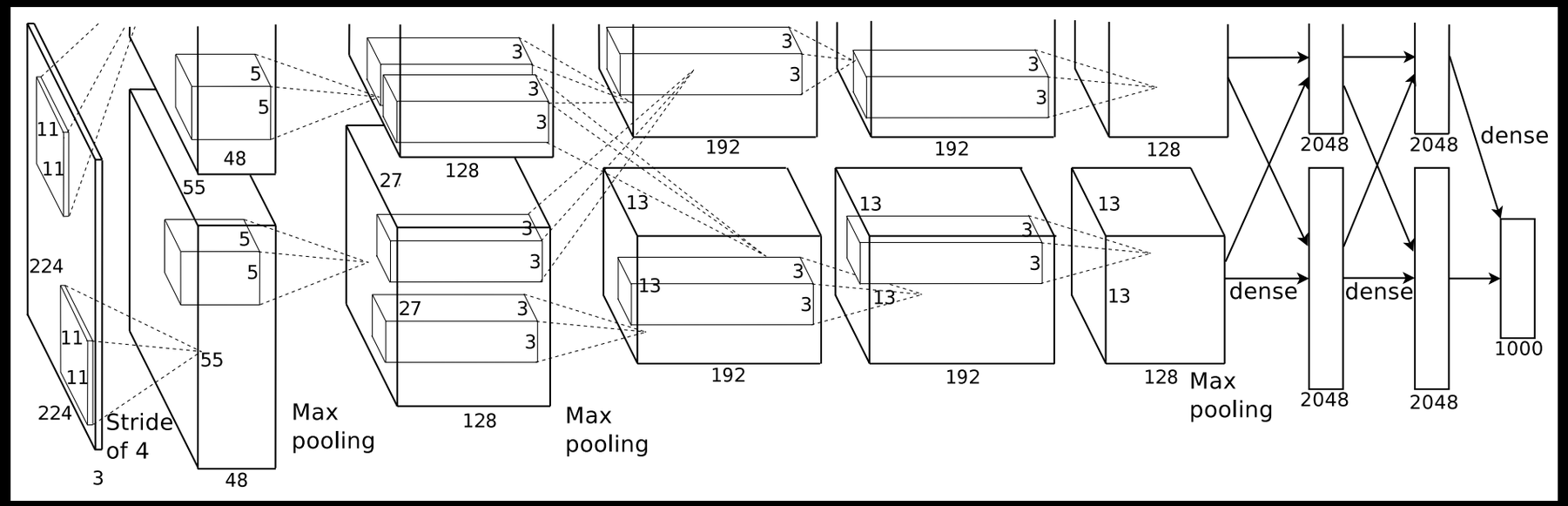
no pre-training

just pre-training

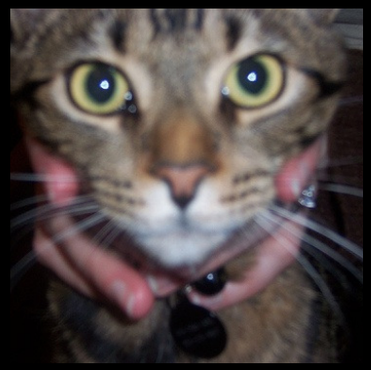
pre-train & fine-tune

Even with more data, pre-training still leads to significantly better performance

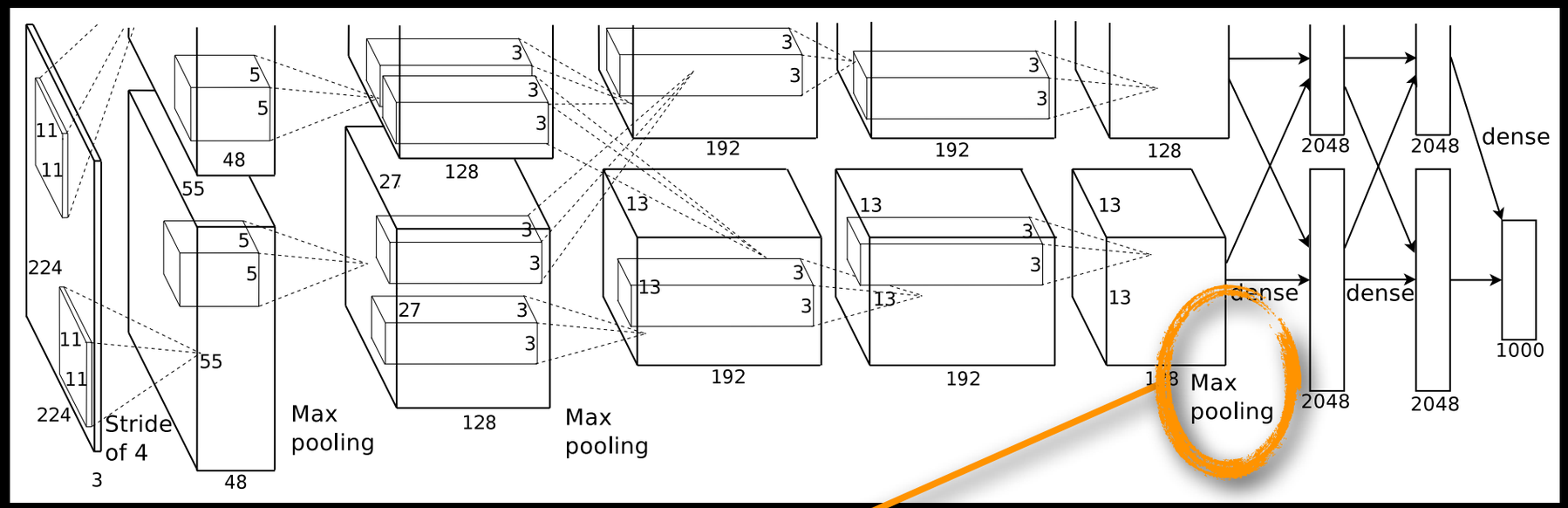
What did the network learn?

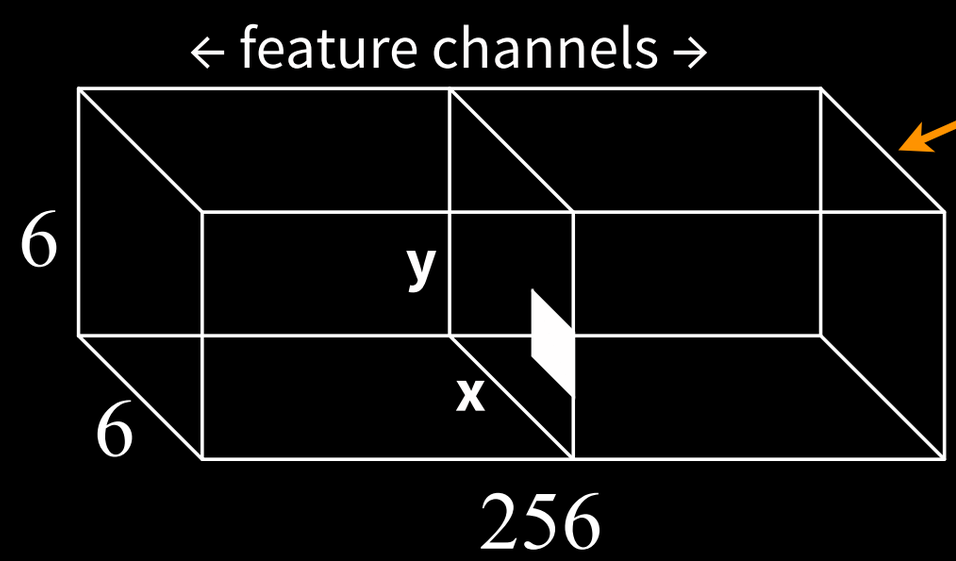


What did the network learn?



"stimulus"

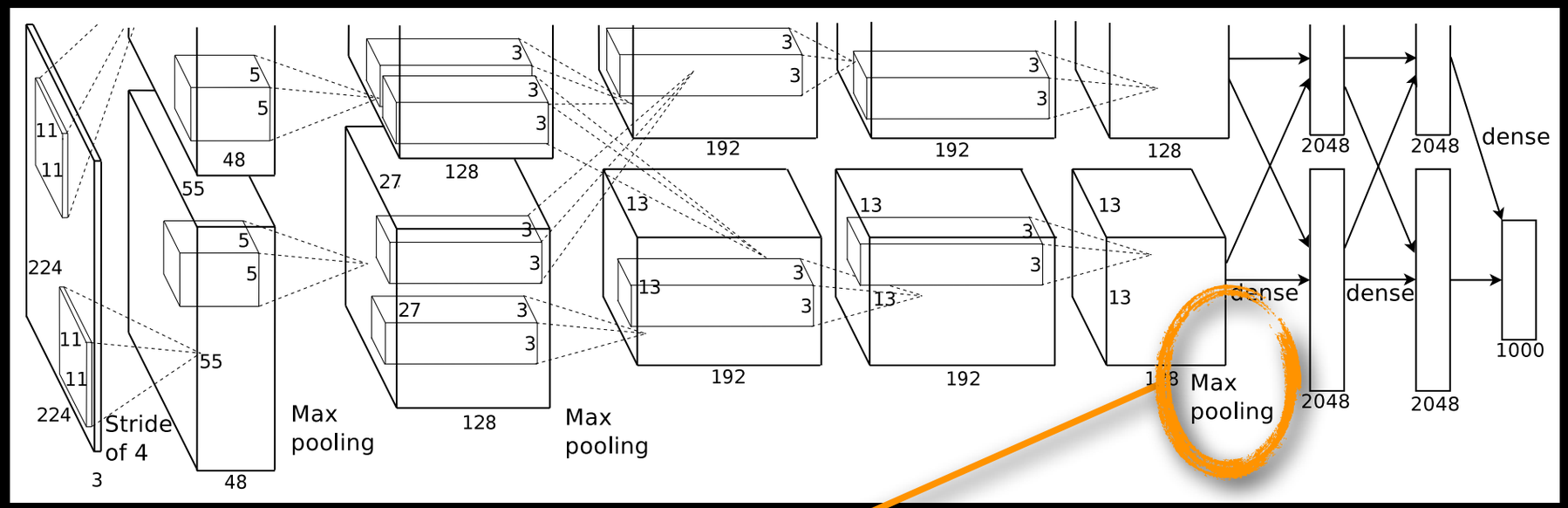


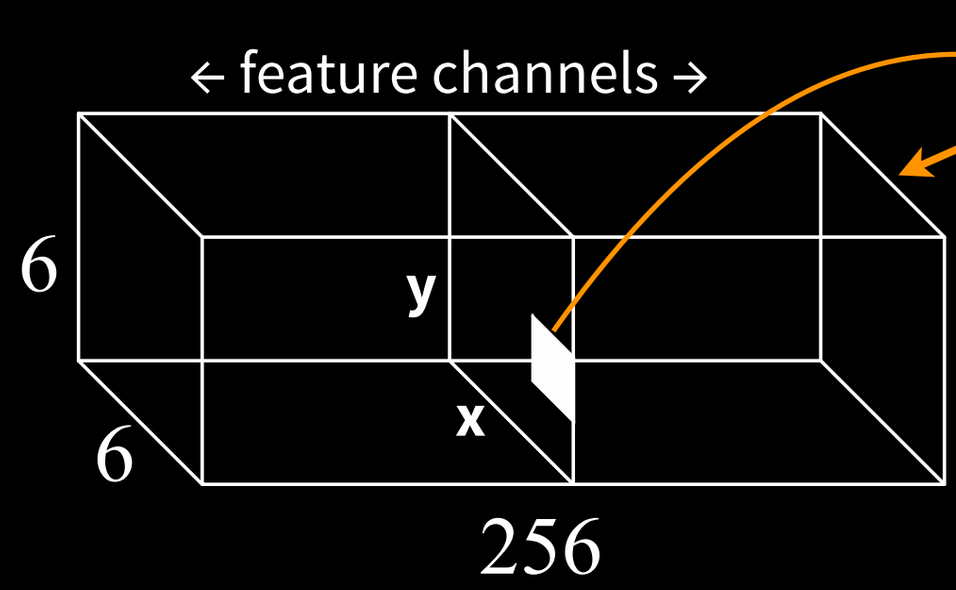


$pool_5$ feature map

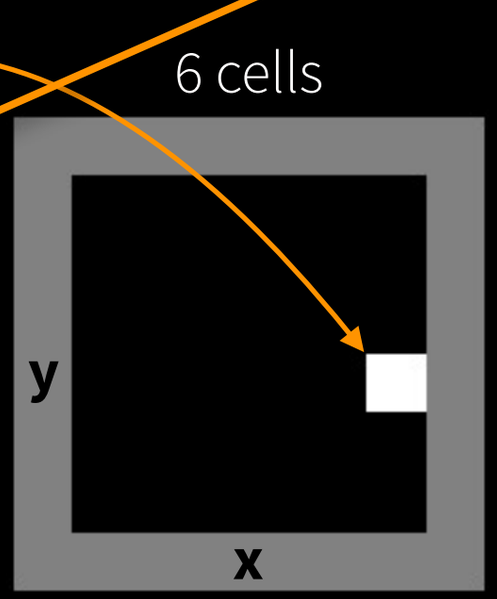
What did the network learn?

"stimulus"

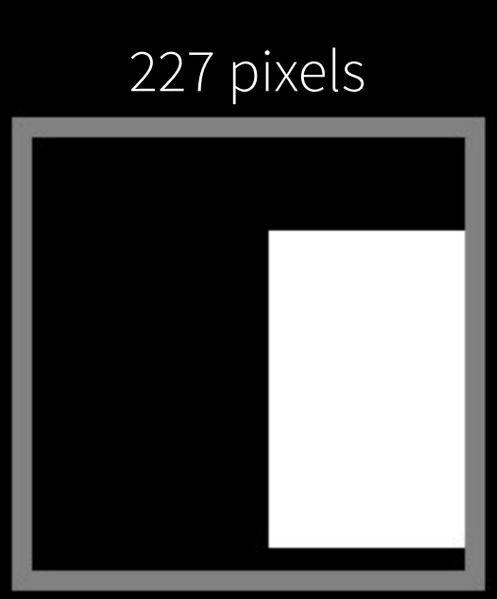




$pool_5$ feature map

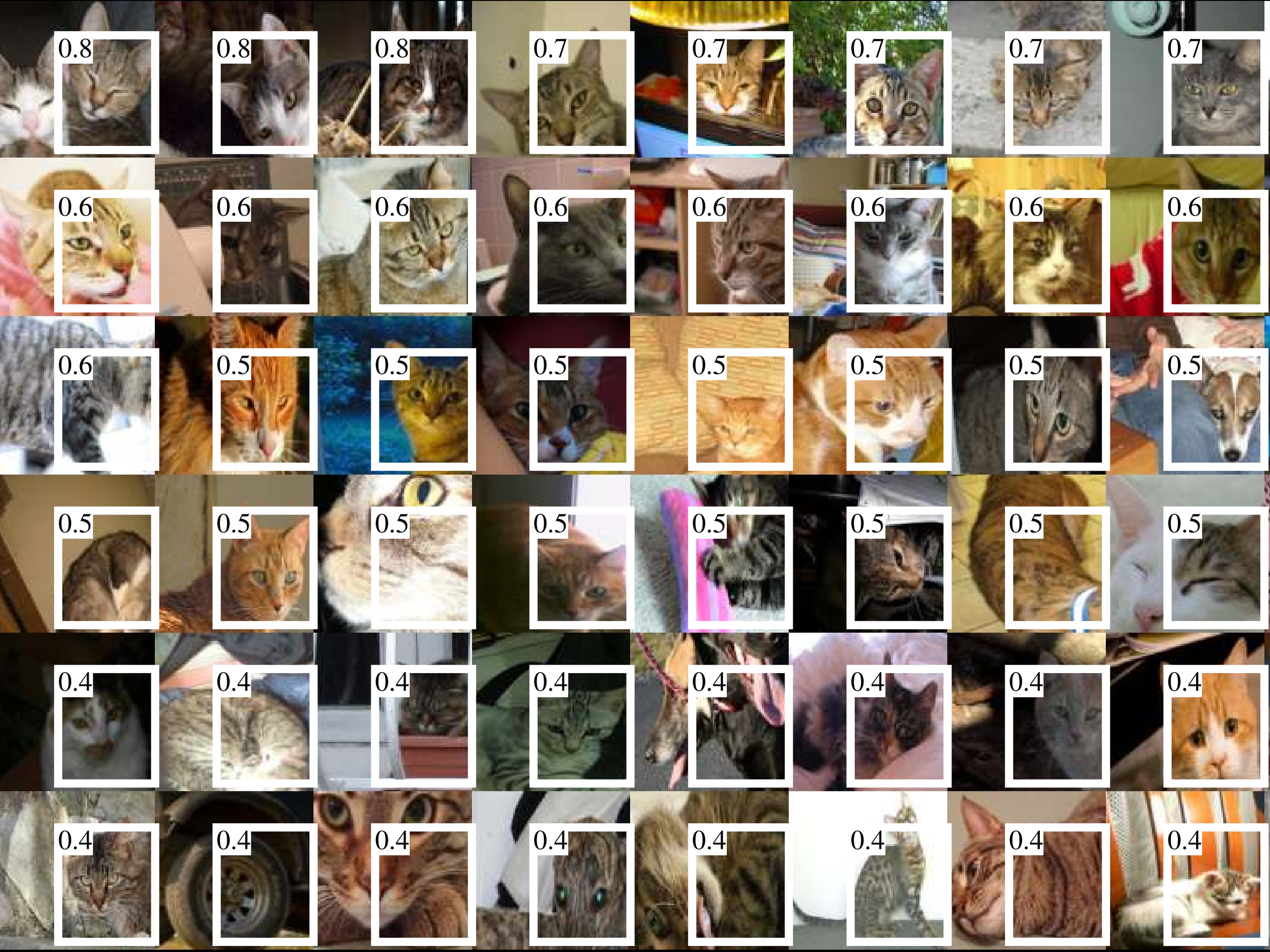


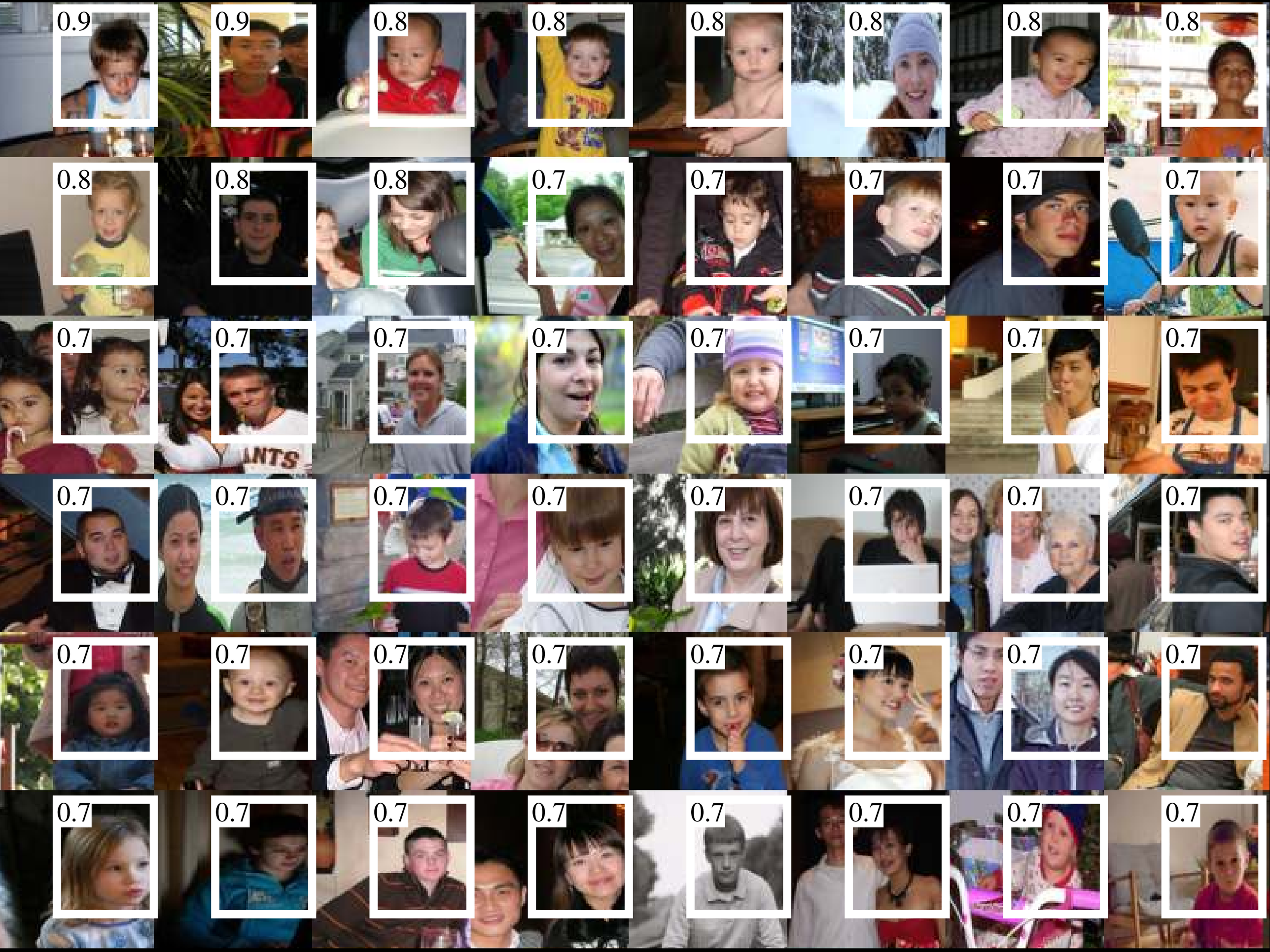
feature position

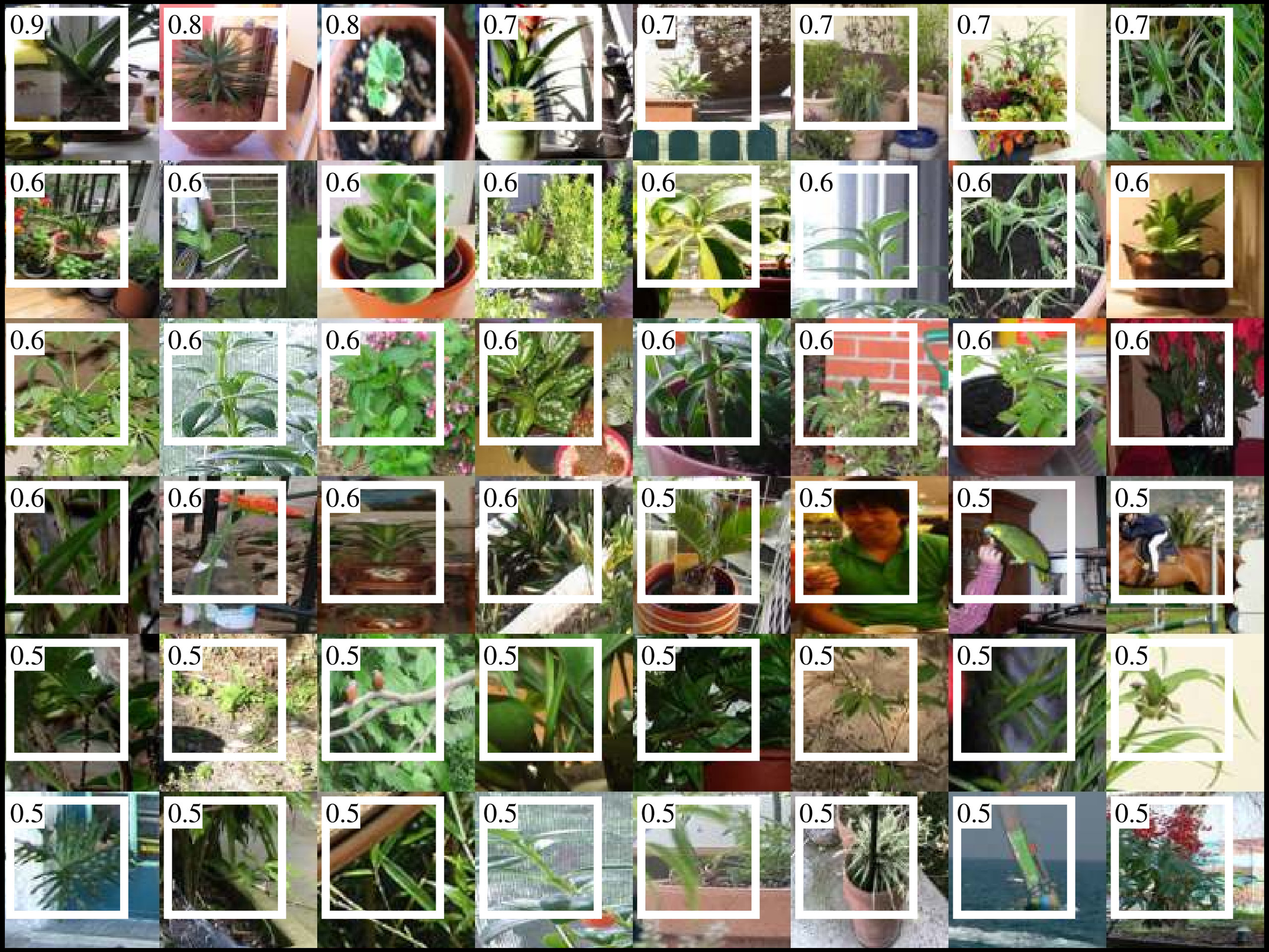


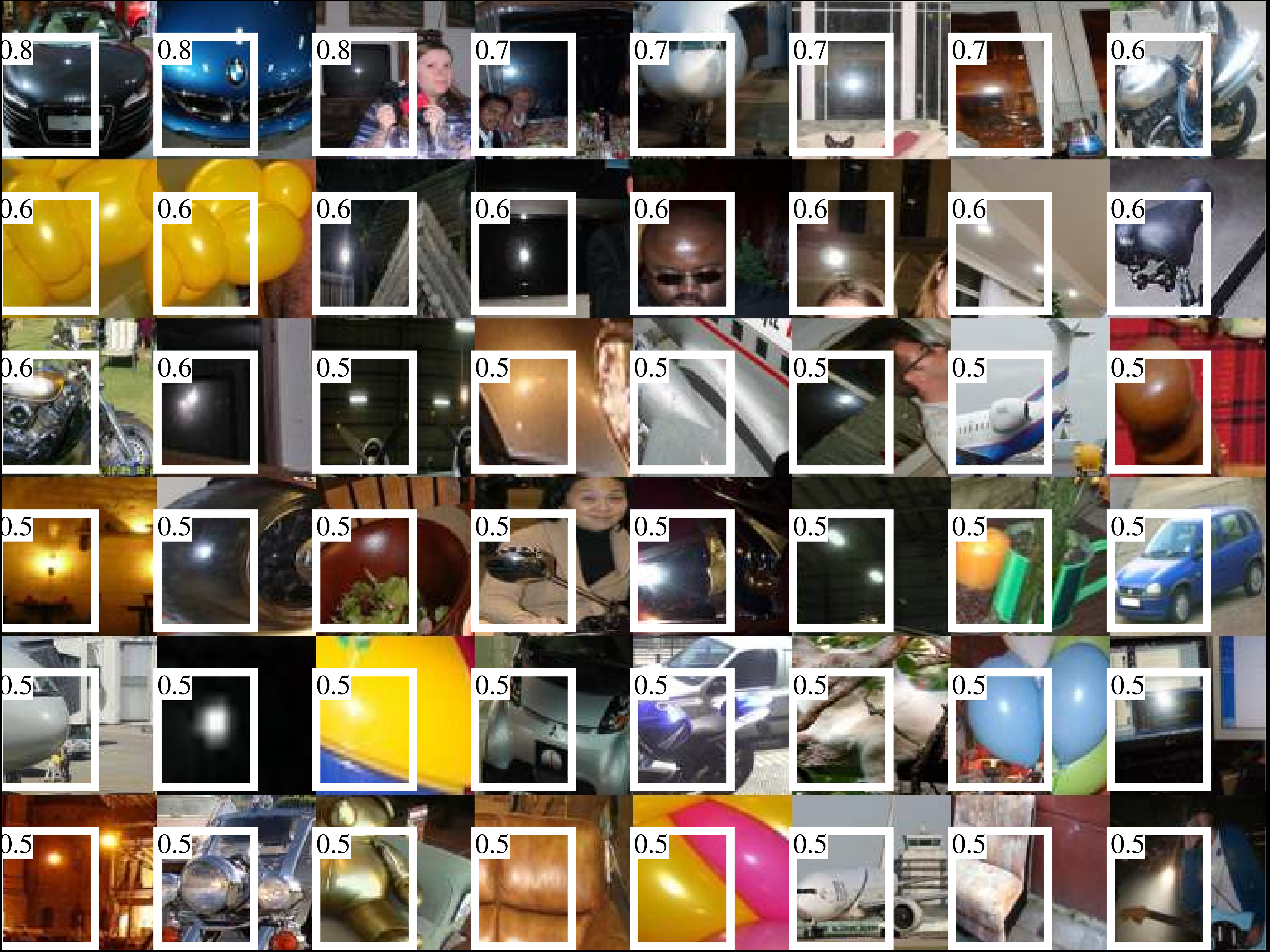
receptive field

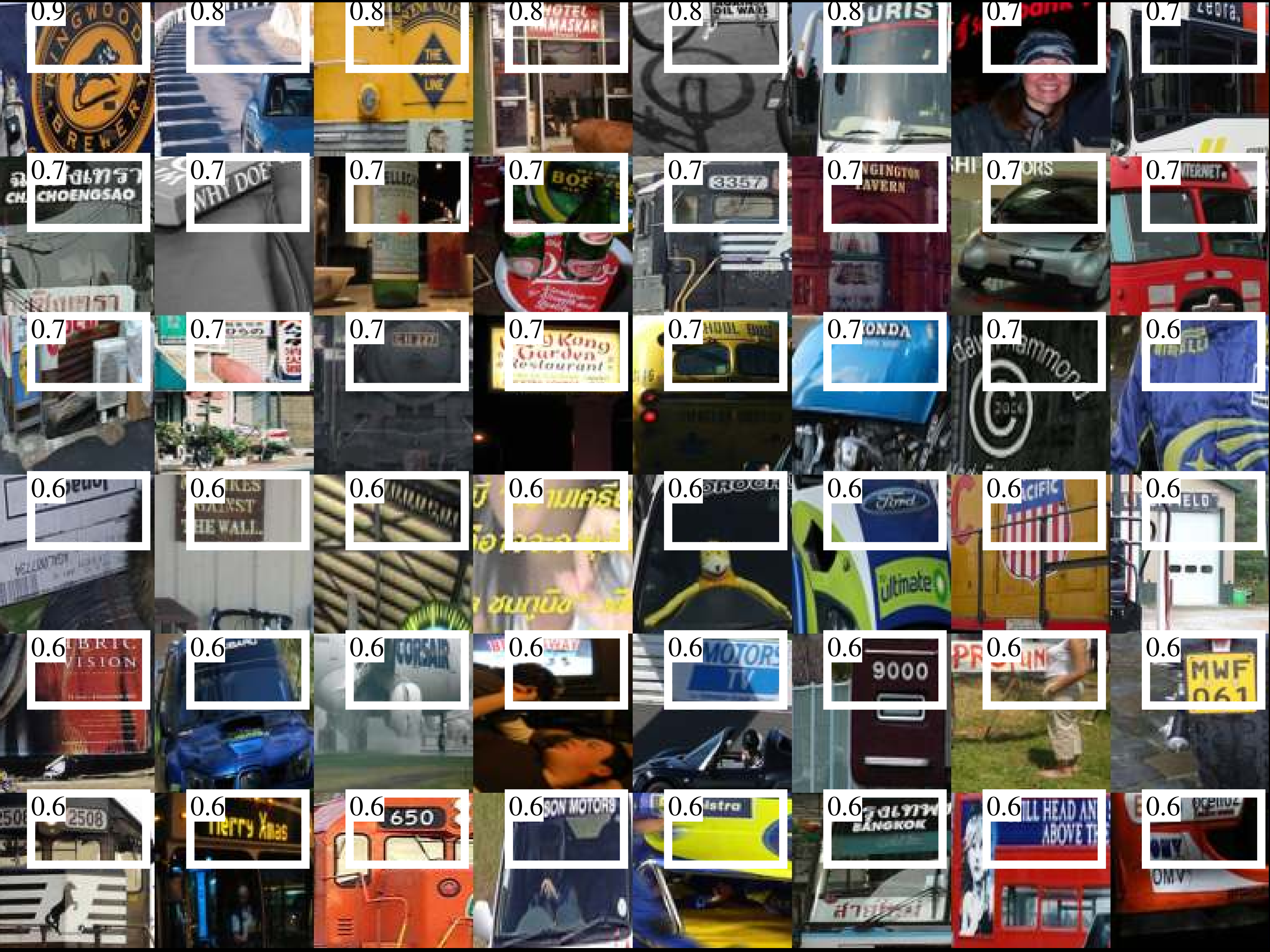
Visualize images that activate $pool_5$ a feature


0.8
0.8
0.8
0.7
0.7
0.7
0.7
0.7
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4
0.4


0.9
0.9
0.8
0.8
0.8
0.8
0.8
0.8
0.8
0.8
0.8
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7
0.7


0.9
0.8
0.8
0.7
0.7
0.7
0.7
0.7
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5


0.8
0.8
0.8
0.7
0.7
0.7
0.7
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.6
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5


0.9 0.8 0.8 0.8 0.8 0.8 0.7 0.7
0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7
0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.6
0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6
0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6
0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6

Are there “Grandmother cells”?