Object detection, deep learning, and R-CNNs

Partly from Ross Girshick

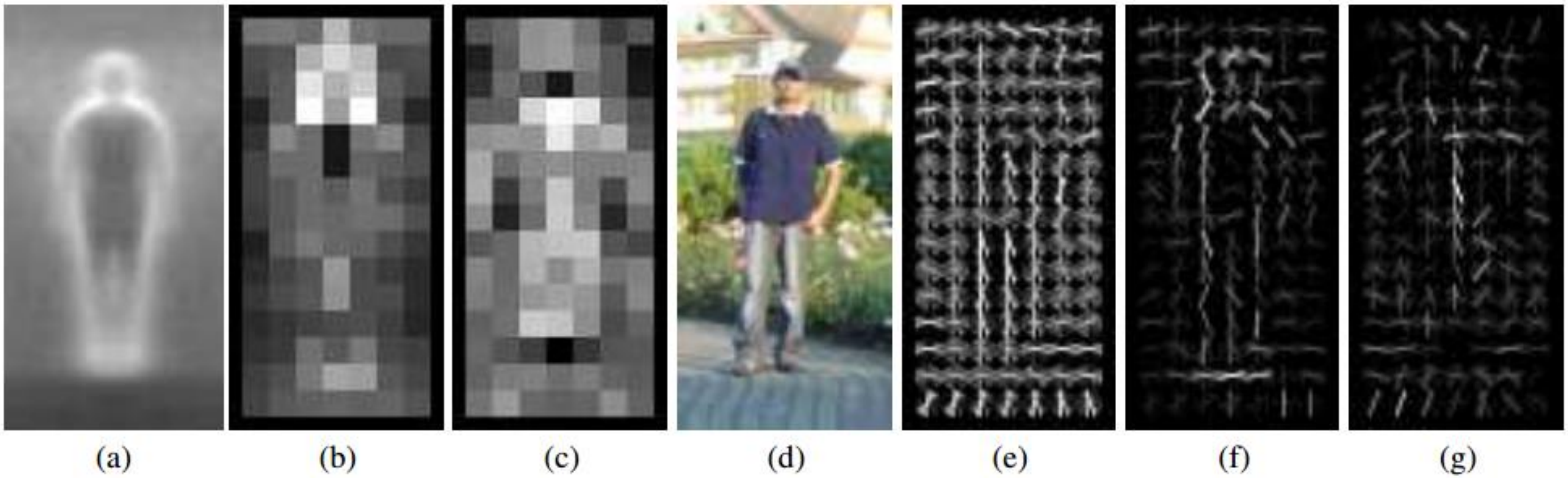
Microsoft Research

Now at Facebook

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

Pedestrians

AP ~77%
More sophisticated methods: AP ~90%



(a) average gradient image over training examples
(b) each “pixel” shows max positive SVM weight in the block centered on that pixel
(c) same as (b) for negative SVM weights
(d) test image
(e) its R-HOG descriptor
(f) R-HOG descriptor weighted by positive SVM weights
(g) R-HOG descriptor weighted by negative SVM weights

Overview of HOG Method

1. Compute gradients in the region to be described
2. Put them in bins according to orientation
3. Group the cells into large blocks
4. Normalize each block
5. Train classifiers to decide if these are parts of a human

Details

- Gradients
  [-1 0 1] and $[-1\ 0\ 1]^T$ were good enough filters.
- Cell Histograms
  Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. (9 channels worked)
- Blocks
  Group the cells together into larger blocks, either R-HOG blocks (rectangular) or C-HOG blocks (circular).

More Details

- Block Normalization

They tried 4 different kinds of normalization.
Let υ be the block to be normalized and e be a small constant.

L2-norm: $$f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

L2-hys: L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing,

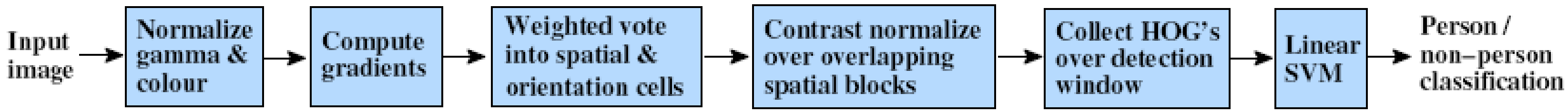
L1-norm: $$f = \frac{v}{(\|v\|_1 + e)}$$

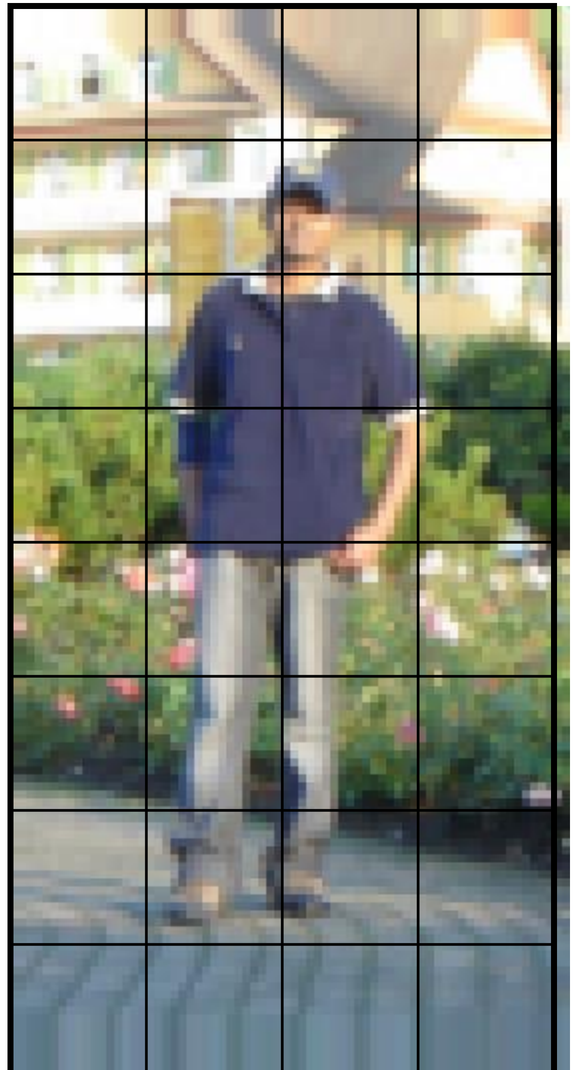
L1-sqrt: $$f = \sqrt{\frac{v}{(\|v\|_1 + e)}}$$

Example: Dalal-Triggs pedestrian

1. Extract fixed-sized (64x128 pixel) window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

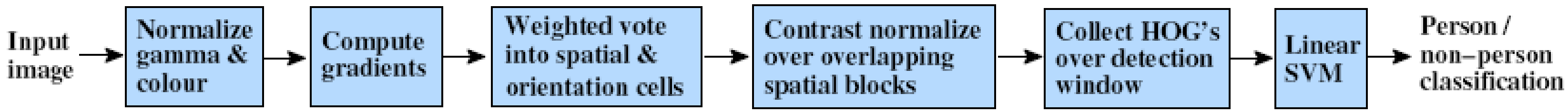
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05





Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



Outperforms

-1	0	1

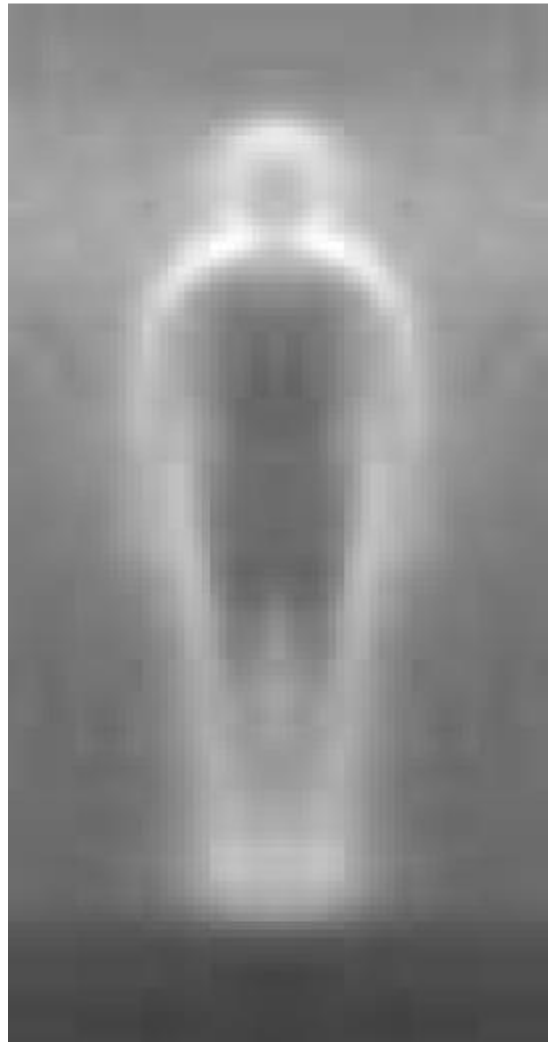
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected



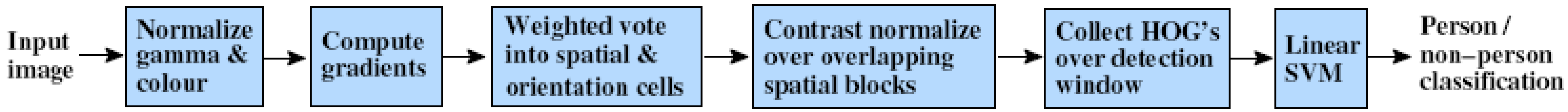
0	1
-1	0

diagonal

-1	0	1
-2	0	2
-1	0	1

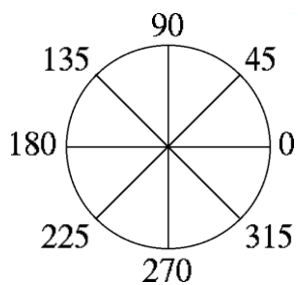
Sobel

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

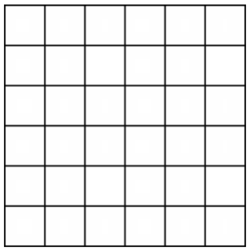


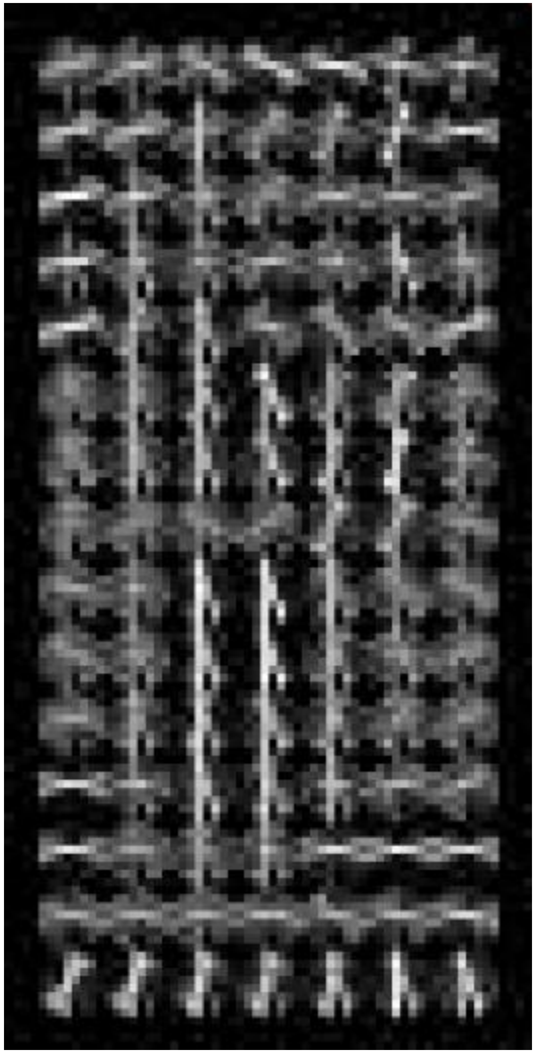
- Histogram of gradient orientations

Orientation: 9 bins (for unsigned angles)



Histograms in 8x8 pixel cells

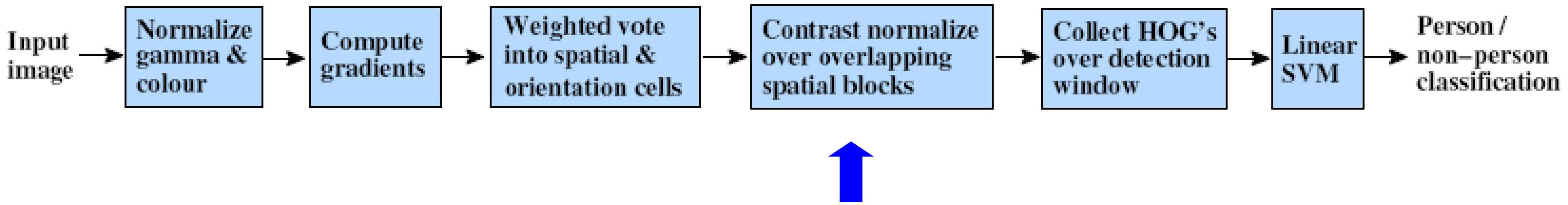




- Votes weighted by magnitude
- Bilinear interpolation between cells

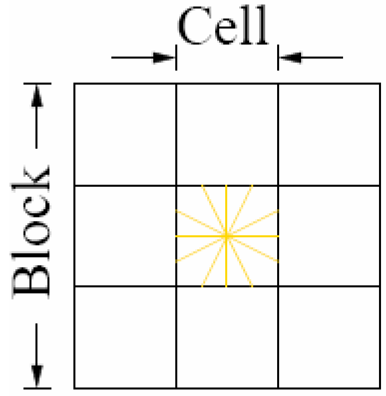
Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



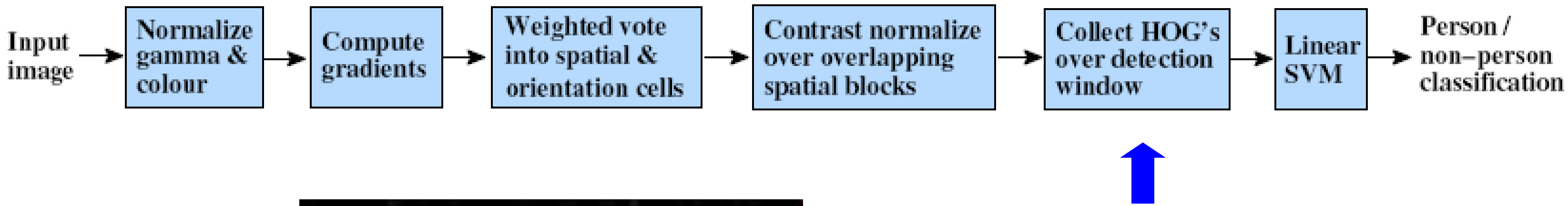
R-HOG

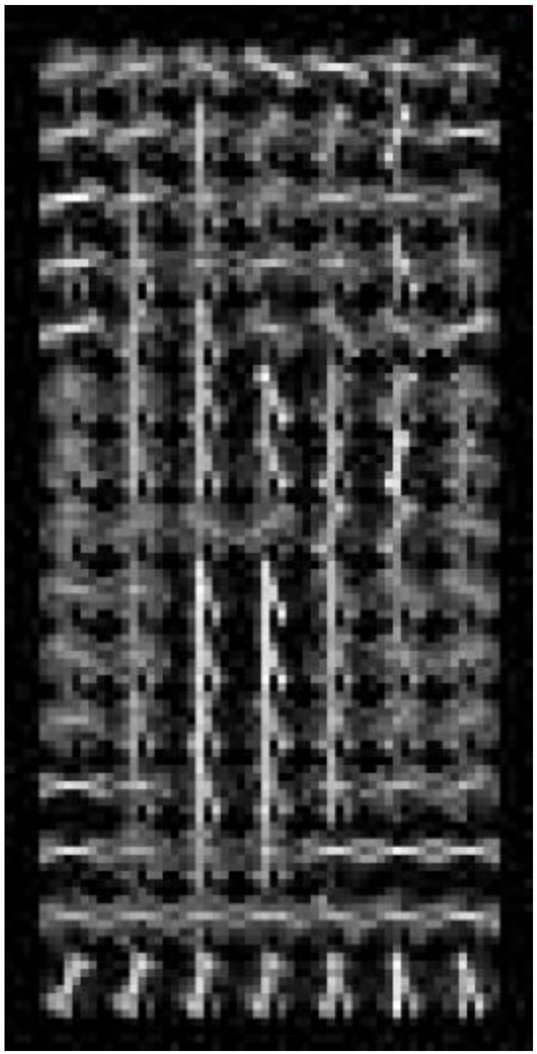
Normalize with respect to surrounding cells



$$L2-norm : v \longrightarrow v/\sqrt{||v||_2^2+\epsilon^2}$$

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



X= 

\# orientations

$\#$ features = $\underline{15 \times 7} \times 9 \times 4 = 3780$

\# cells

\# normalizations by neighboring cells

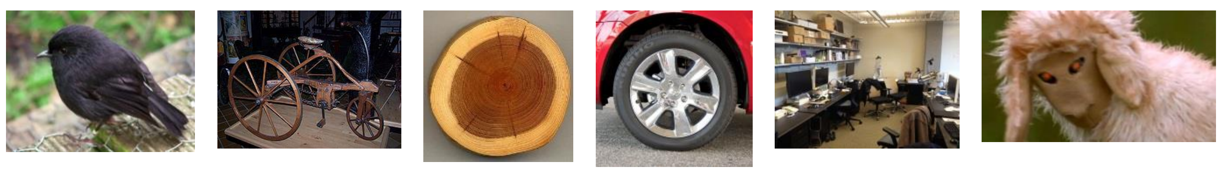
Slides by Pete Barnum

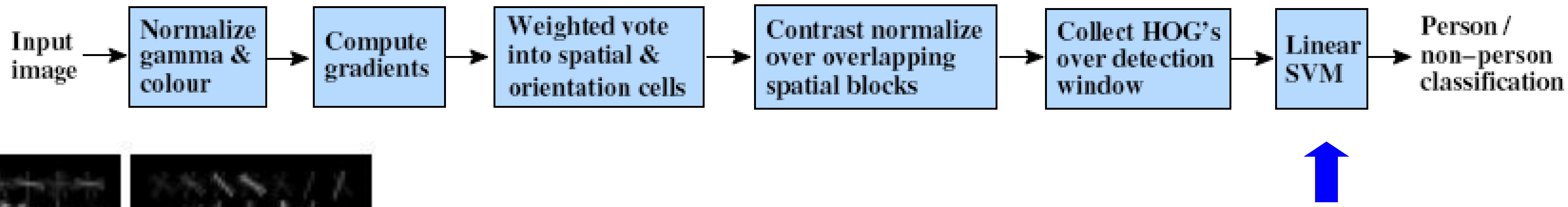
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

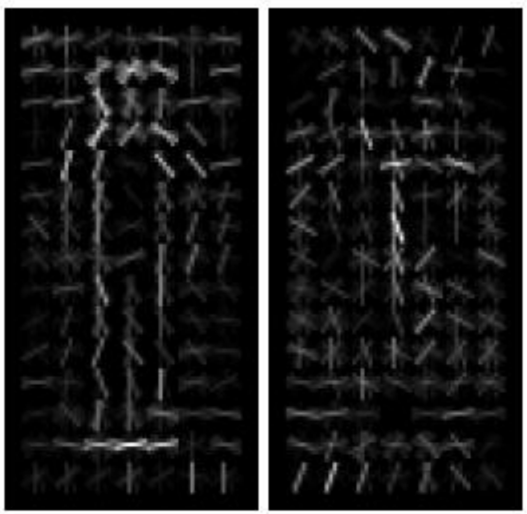
Training set

+

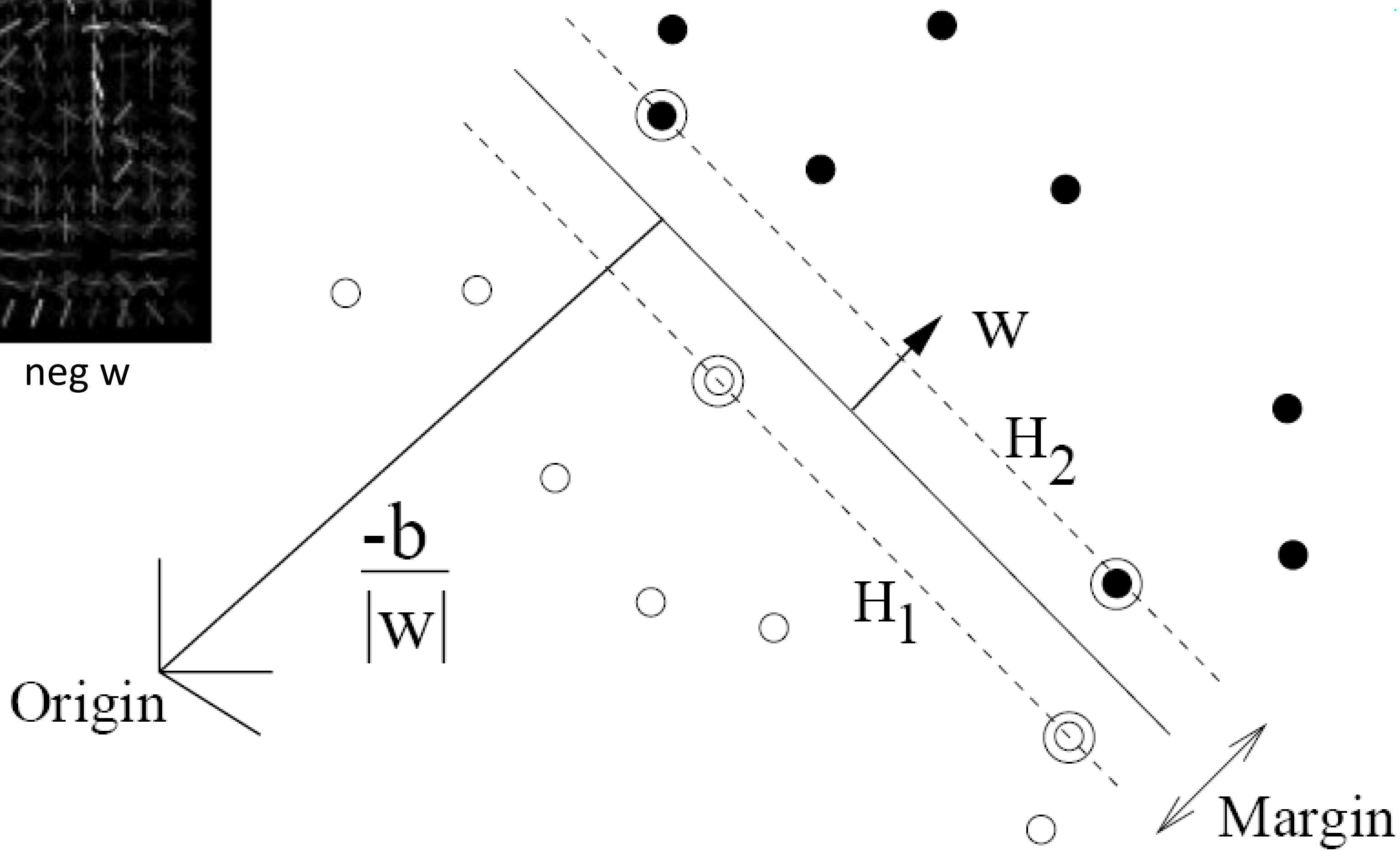
−





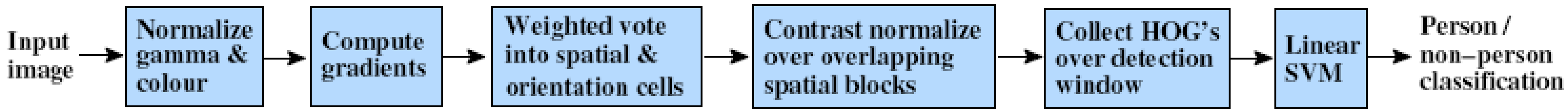


pos w　　neg w



Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



$$0.16 = w^T x - b$$

$$sign(0.16) = 1$$

$=>$ pedestrian

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Detection examples

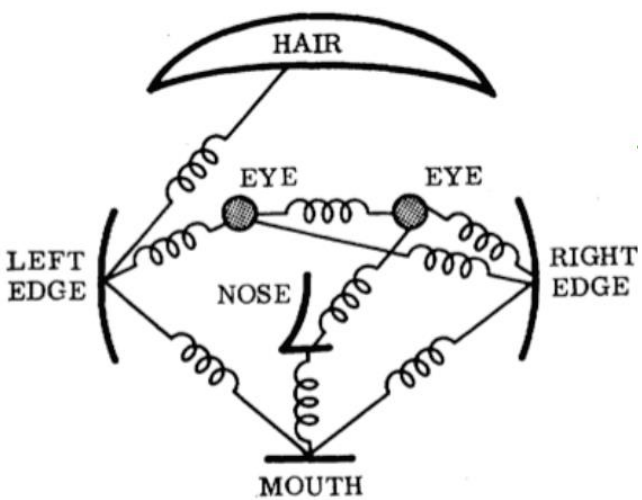
Deformable Parts Model

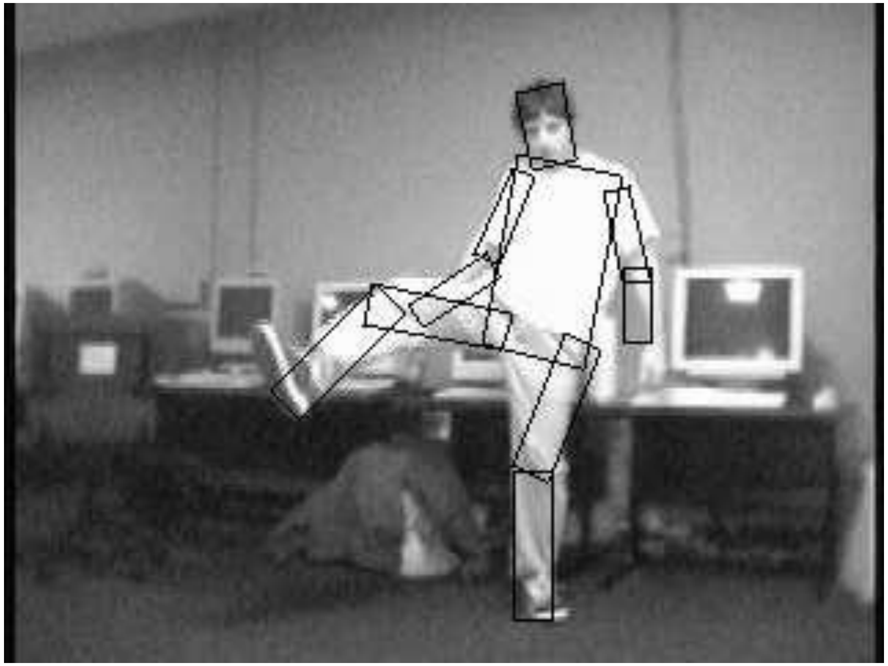
- Takes the idea a little further
- Instead of one rigid HOG model, we have multiple HOG models in a spatial arrangement
- One root part to find first and multiple other parts in a tree structure.

The Idea

Articulated parts model

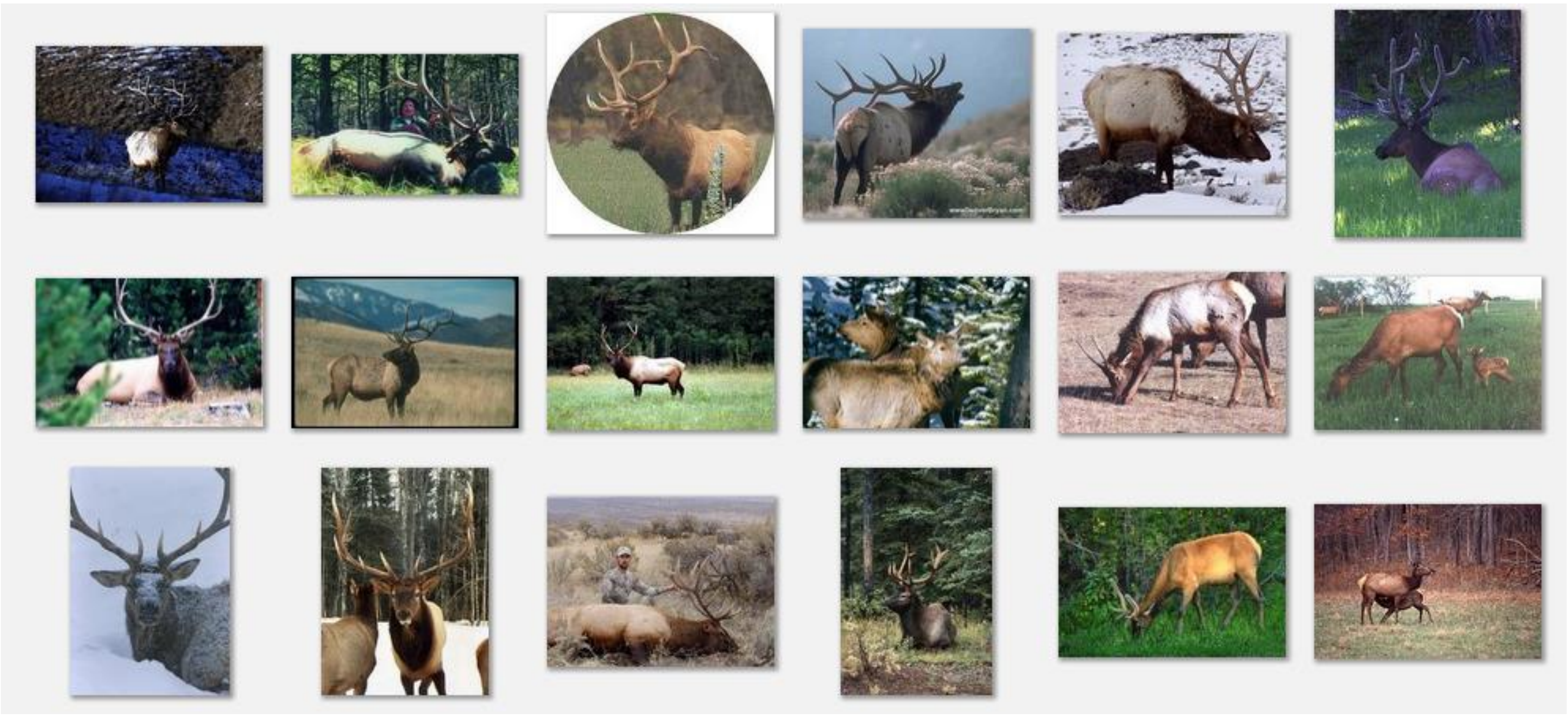
- Object is configuration of parts
- Each part is detectable





Images from Felzenszwalb

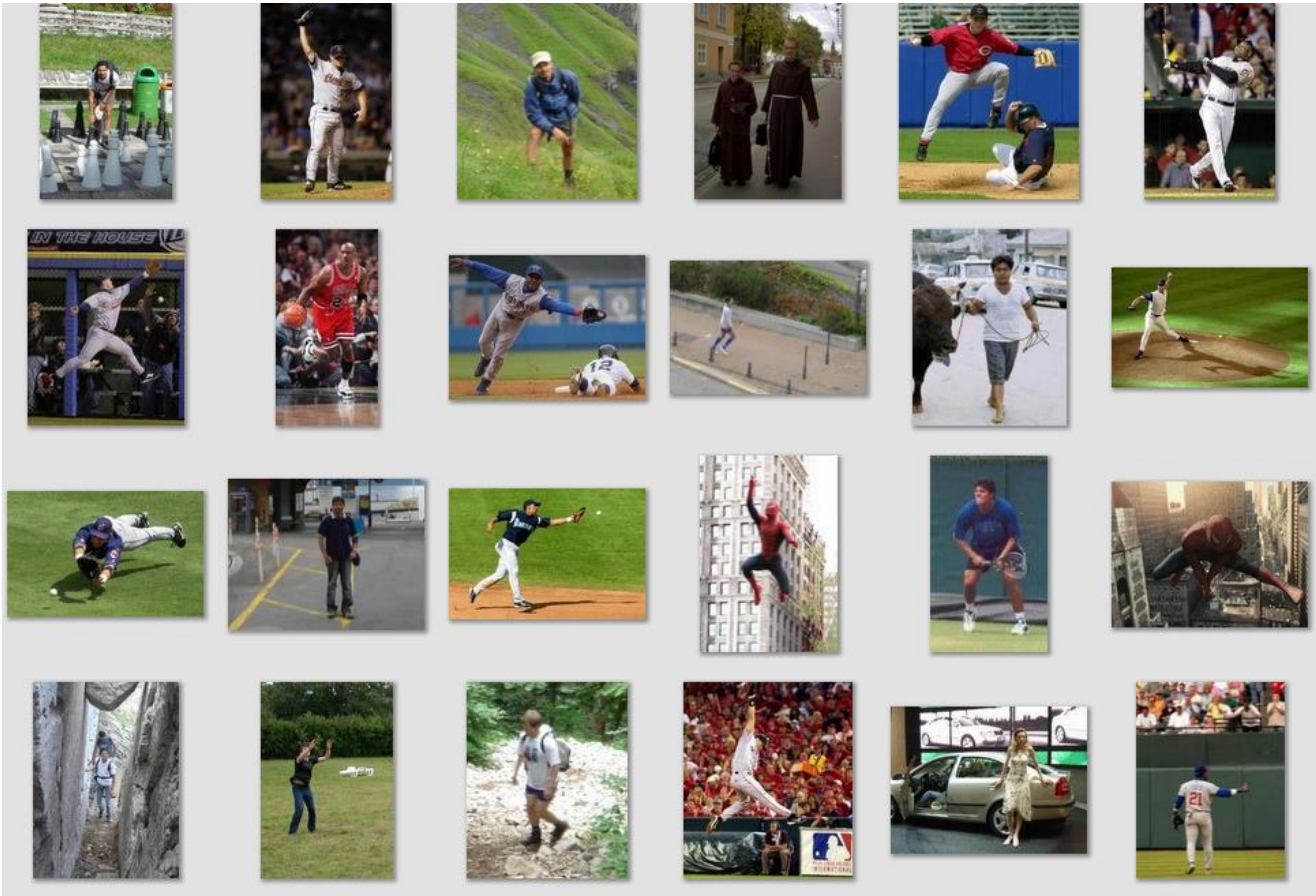
Deformable objects



Images from Caltech-256

Slide Credit: Duan Tran

Deformable objects

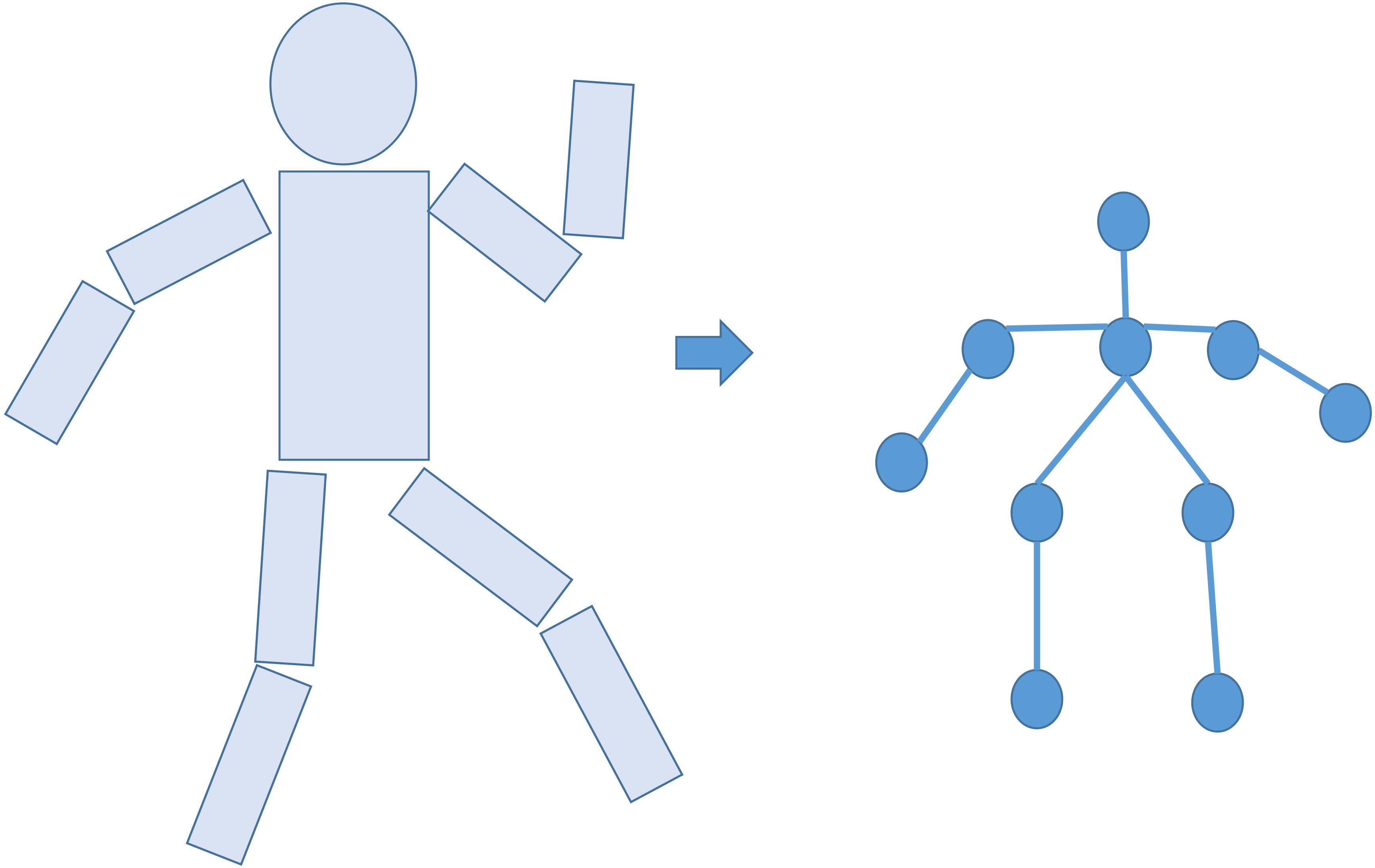


Images from D. Ramanan’s dataset

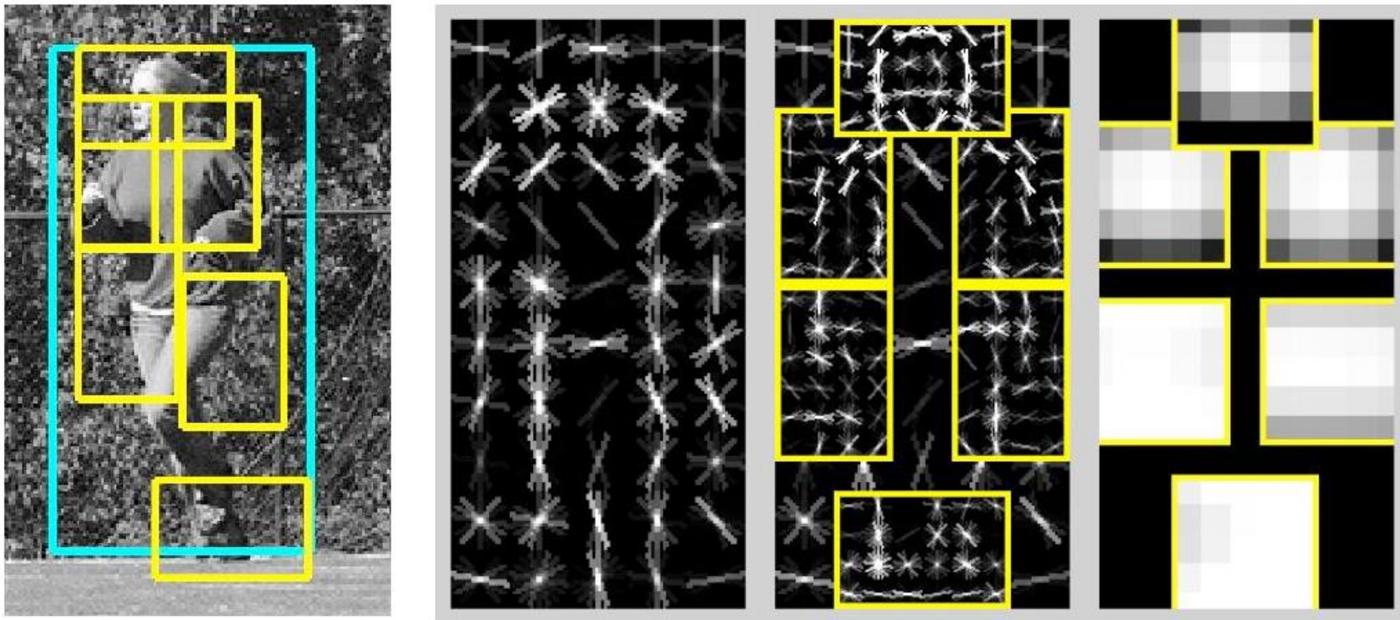
Slide Credit: Duan Tran

How to model spatial relations?

- Tree-shaped model



Model Overview

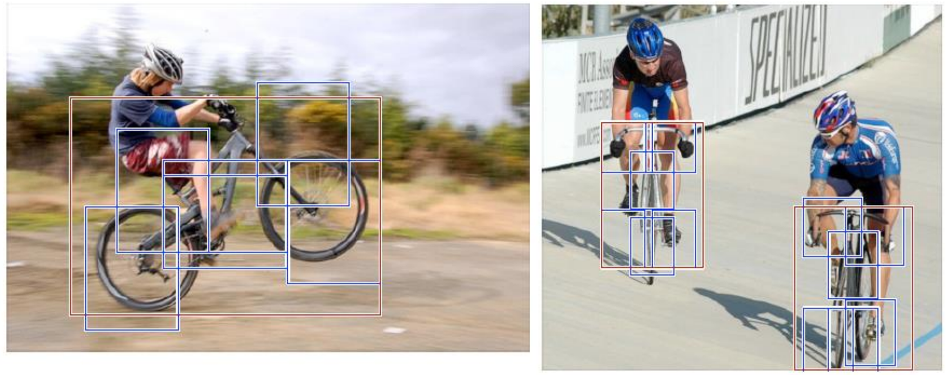


detection root filter part filters deformation models

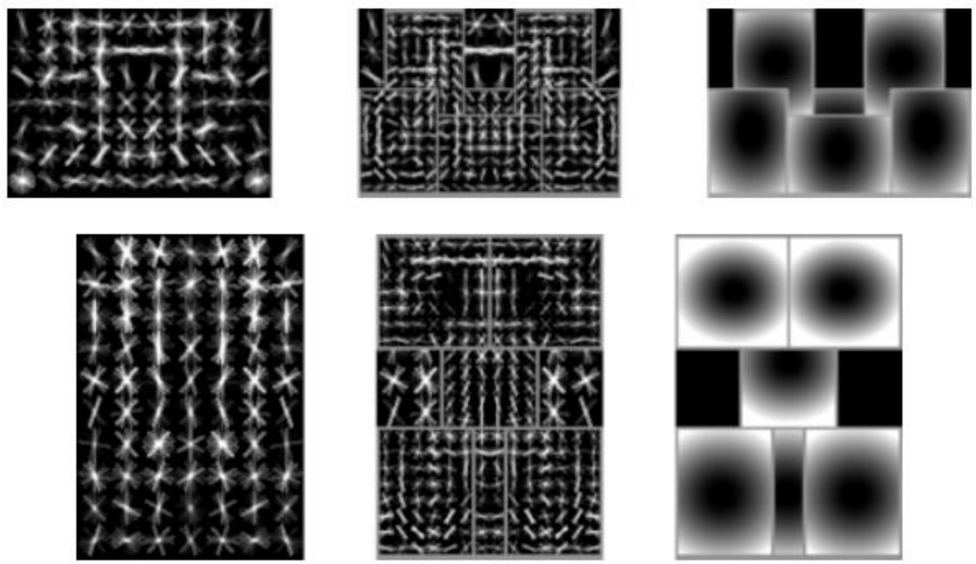
Model has a root filter plus deformable parts

Hybrid template/parts model

Detections



Template Visualization



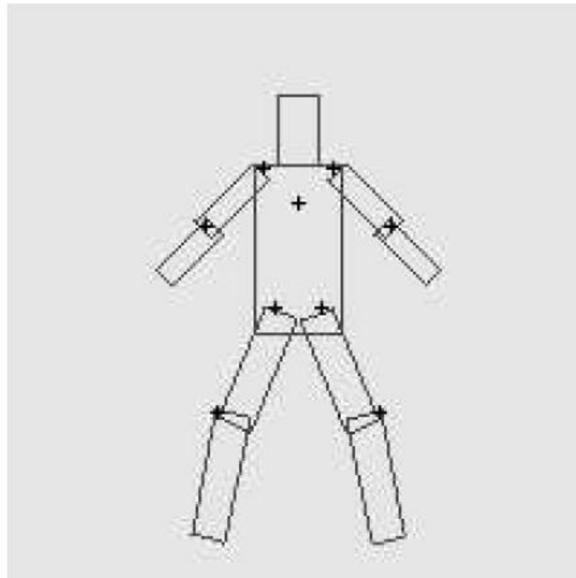
root filters
coarse resolution

part filters
finer resolution

deformation
models

Felzenszwalb et al. 2008

Pictorial Structures Model

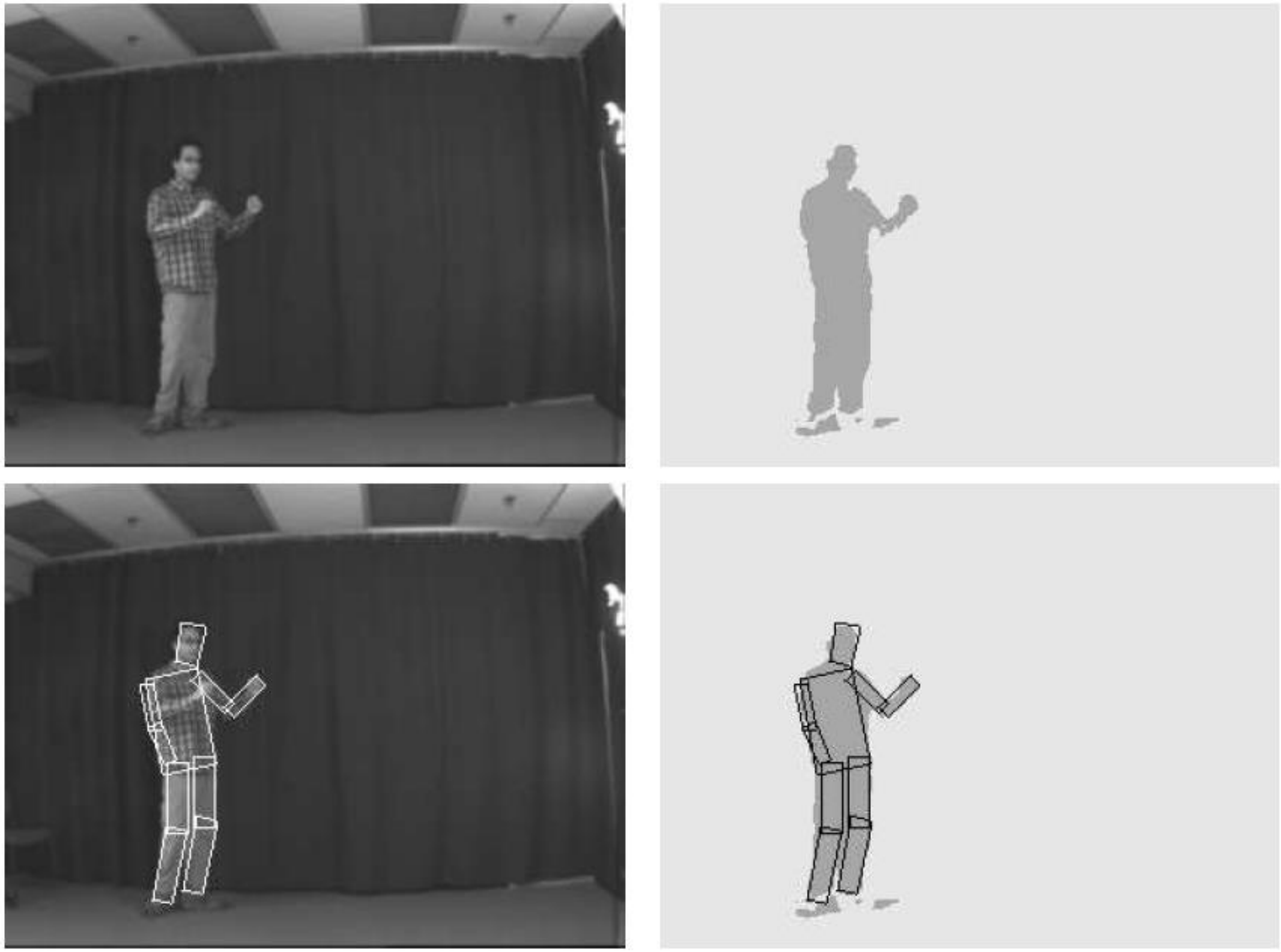


$$P(L|I,\theta) \propto \left(\prod_{i=1}^{n} p(I|l_i, u_i) \prod_{(v_i,v_j)\in E} p(l_i, l_j|c_{ij}) \right)$$

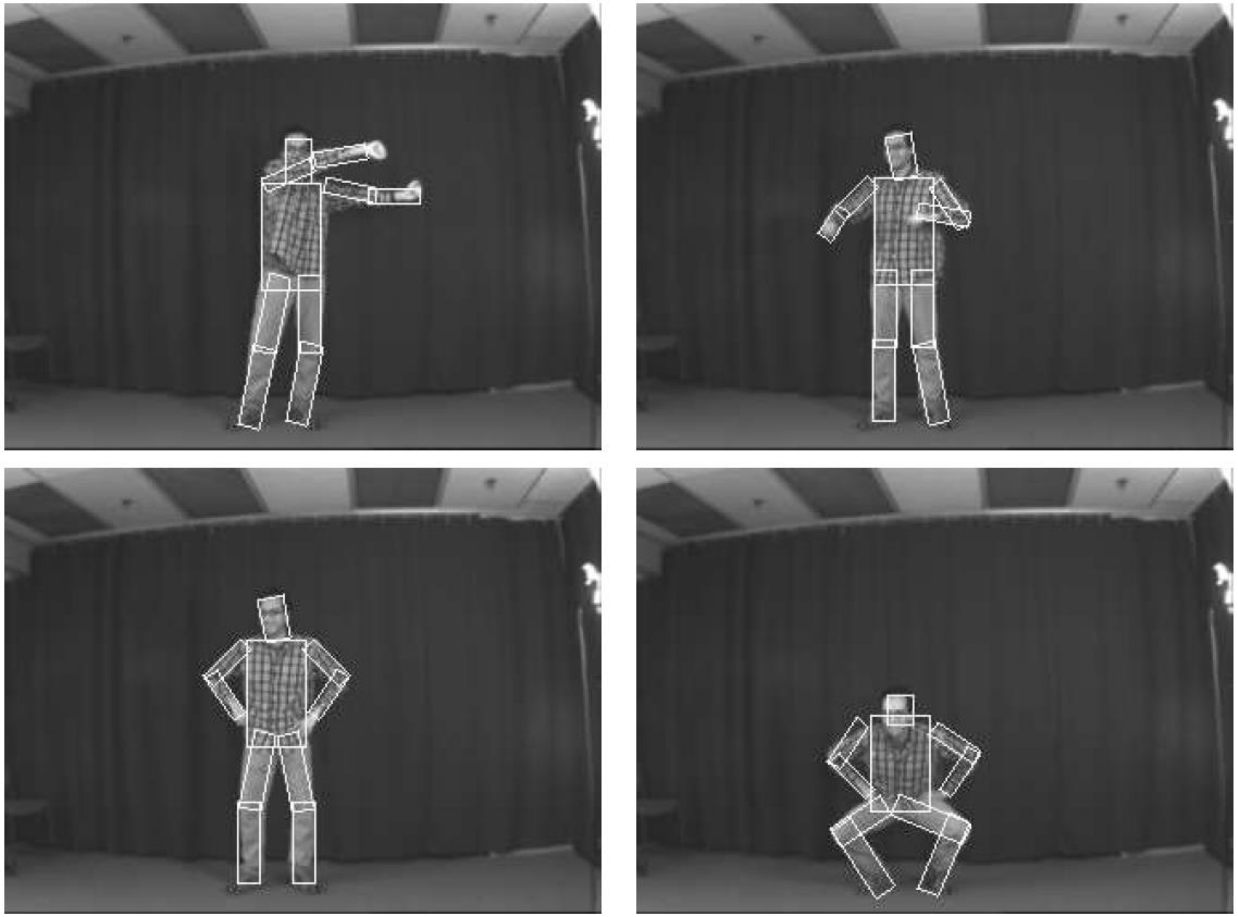
Appearance likelihood

Geometry likelihood

Results for person matching

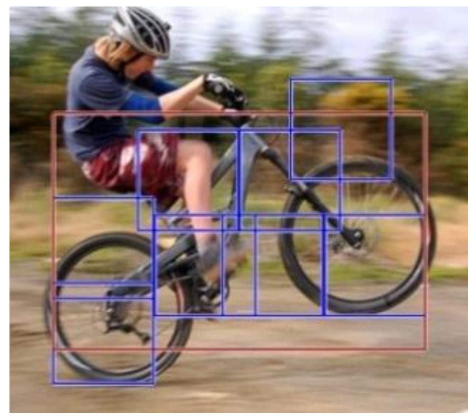


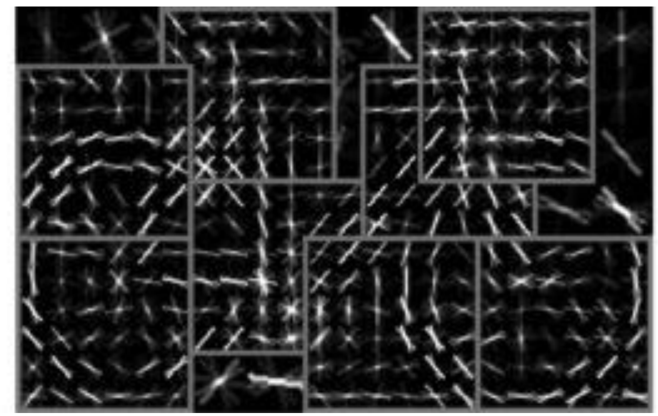
Results for person matching




1
2
3
4
5
6
a
b
c
d

2012 State-of-the-art Detector: Deformable Parts Model (DPM)





1. Strong low-level features based on HOG
2. Efficient matching algorithms for deformable part-based models (pictorial structures)
3. Discriminative learning with latent variables (latent SVM)

Felzenszwalb et al., 2008, 2010, 2011, 2012

Why did gradient-based models work?

Average gradient image

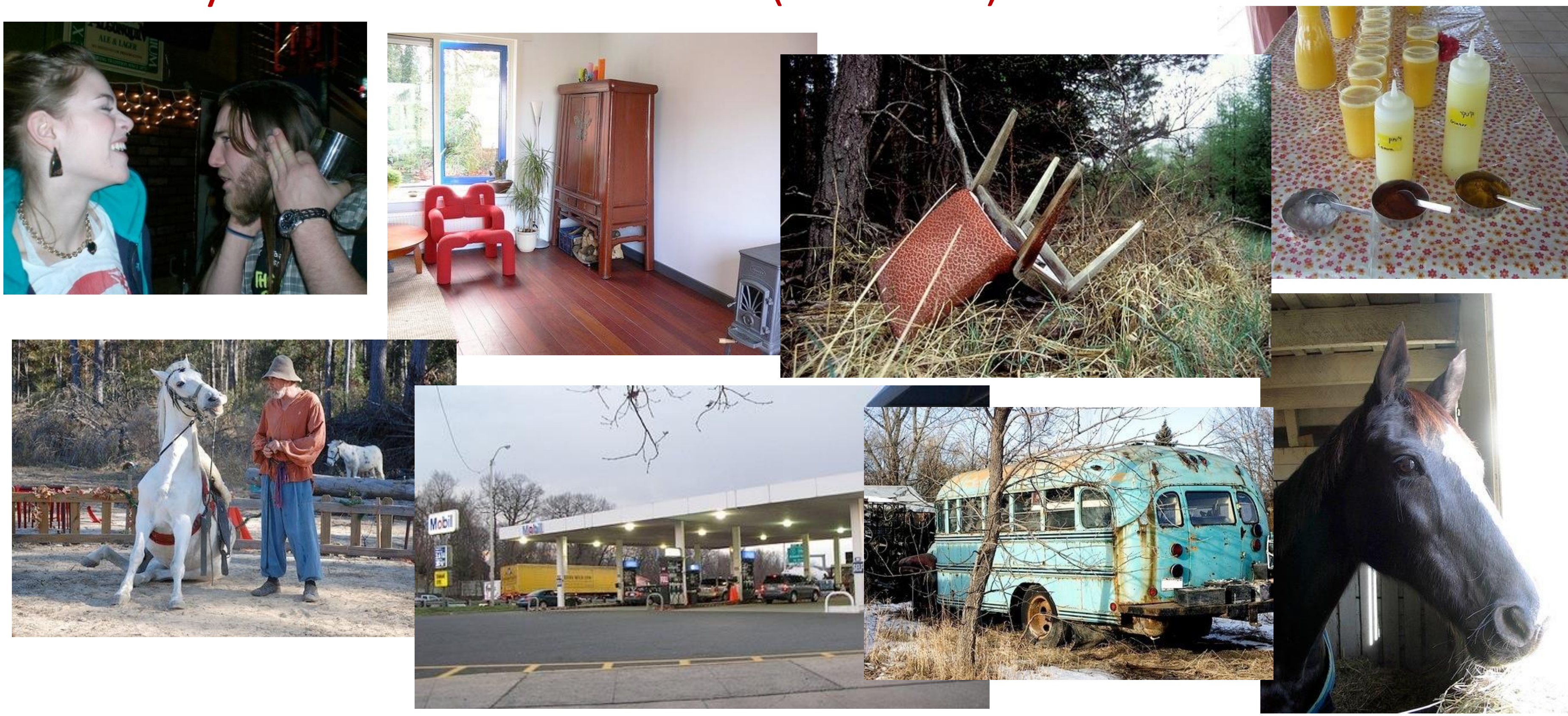
Generic categories

Can we detect people, chairs, horses, cars, dogs, buses, bottles, sheep …?

PASCAL Visual Object Categories (VOC) dataset

Generic categories

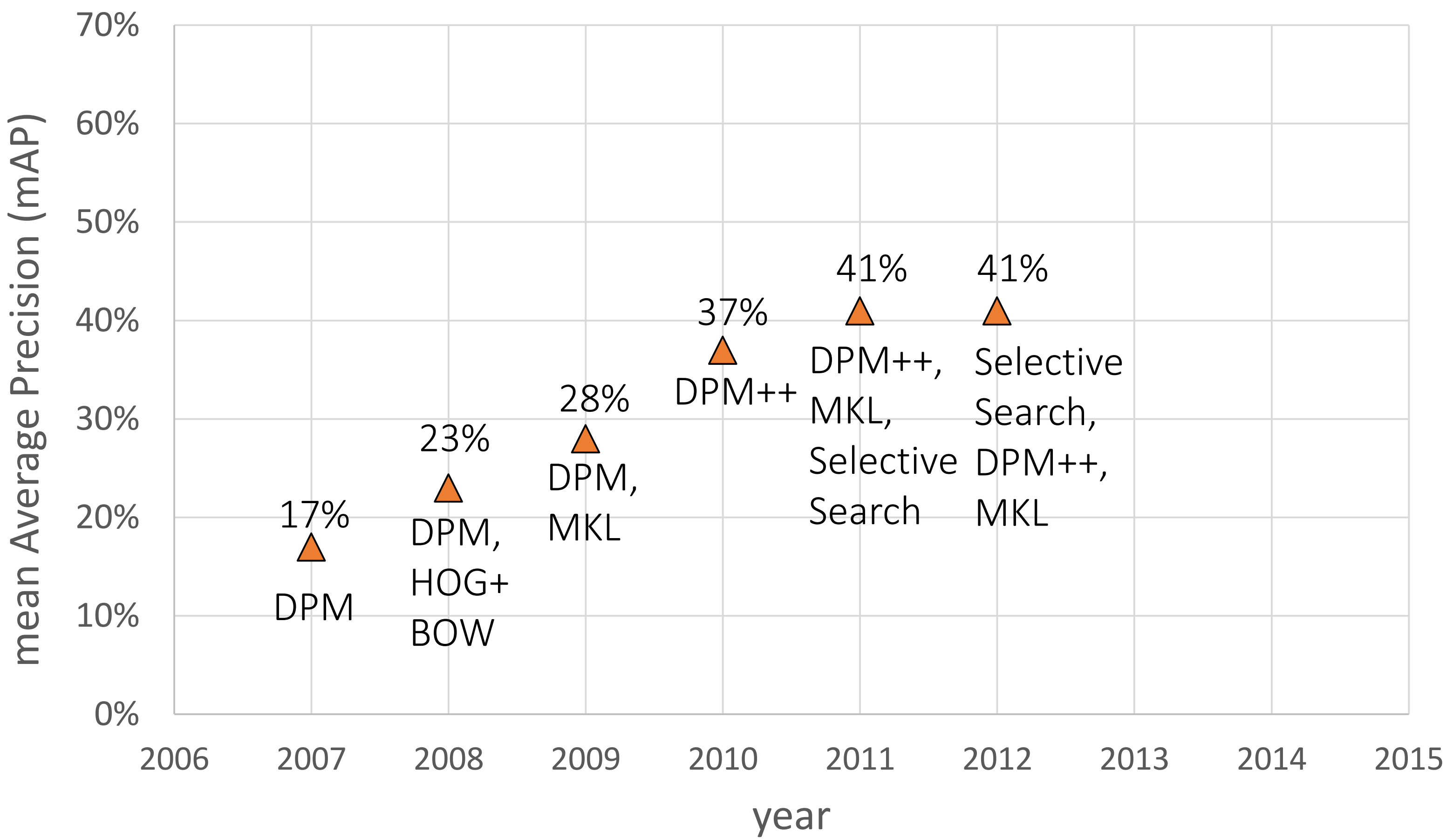
Why doesn't this work (as well)?



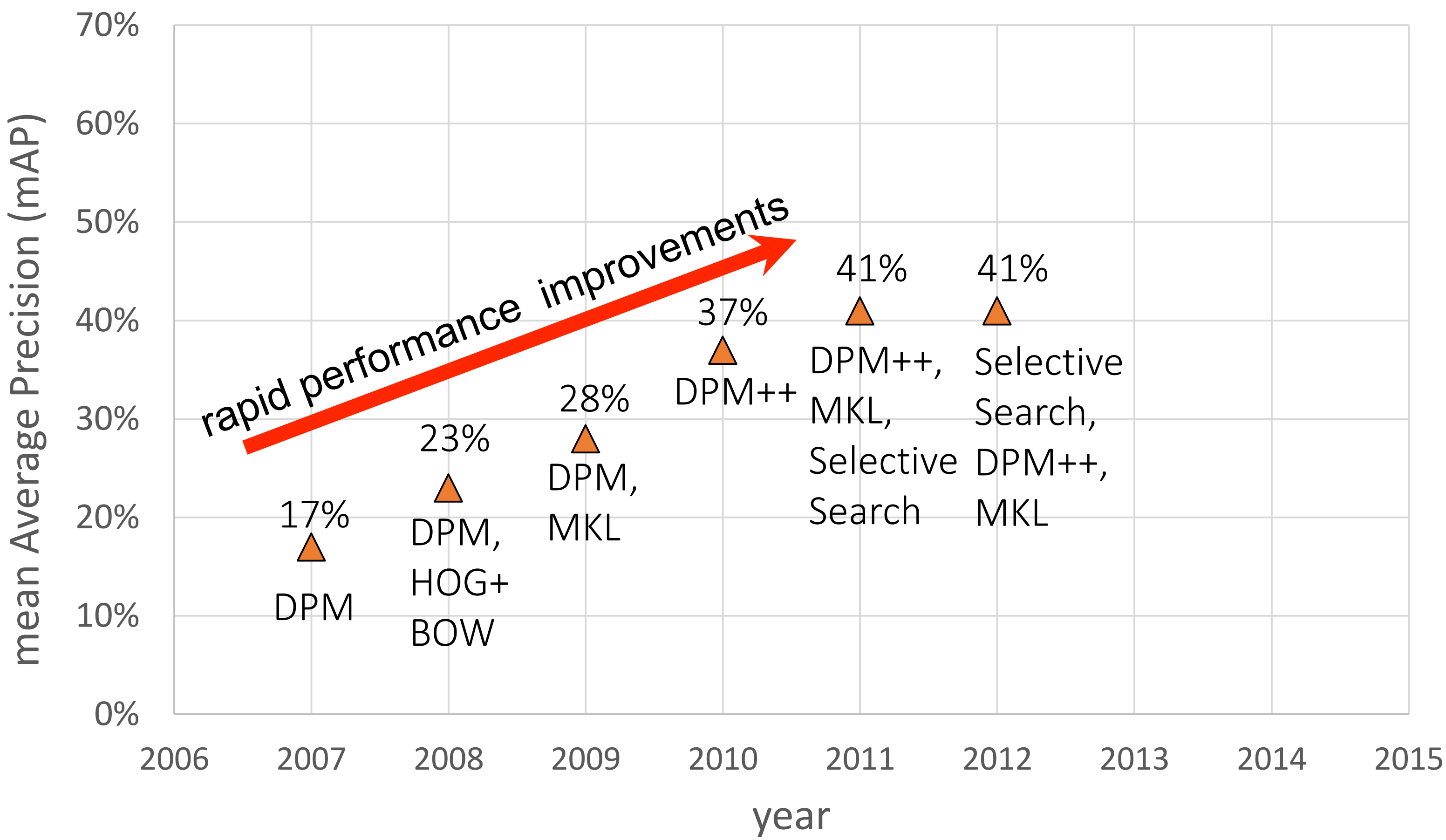
Can we detect people, chairs, horses, cars, dogs, buses, bottles, sheep …?

PASCAL Visual Object Categories (VOC) dataset

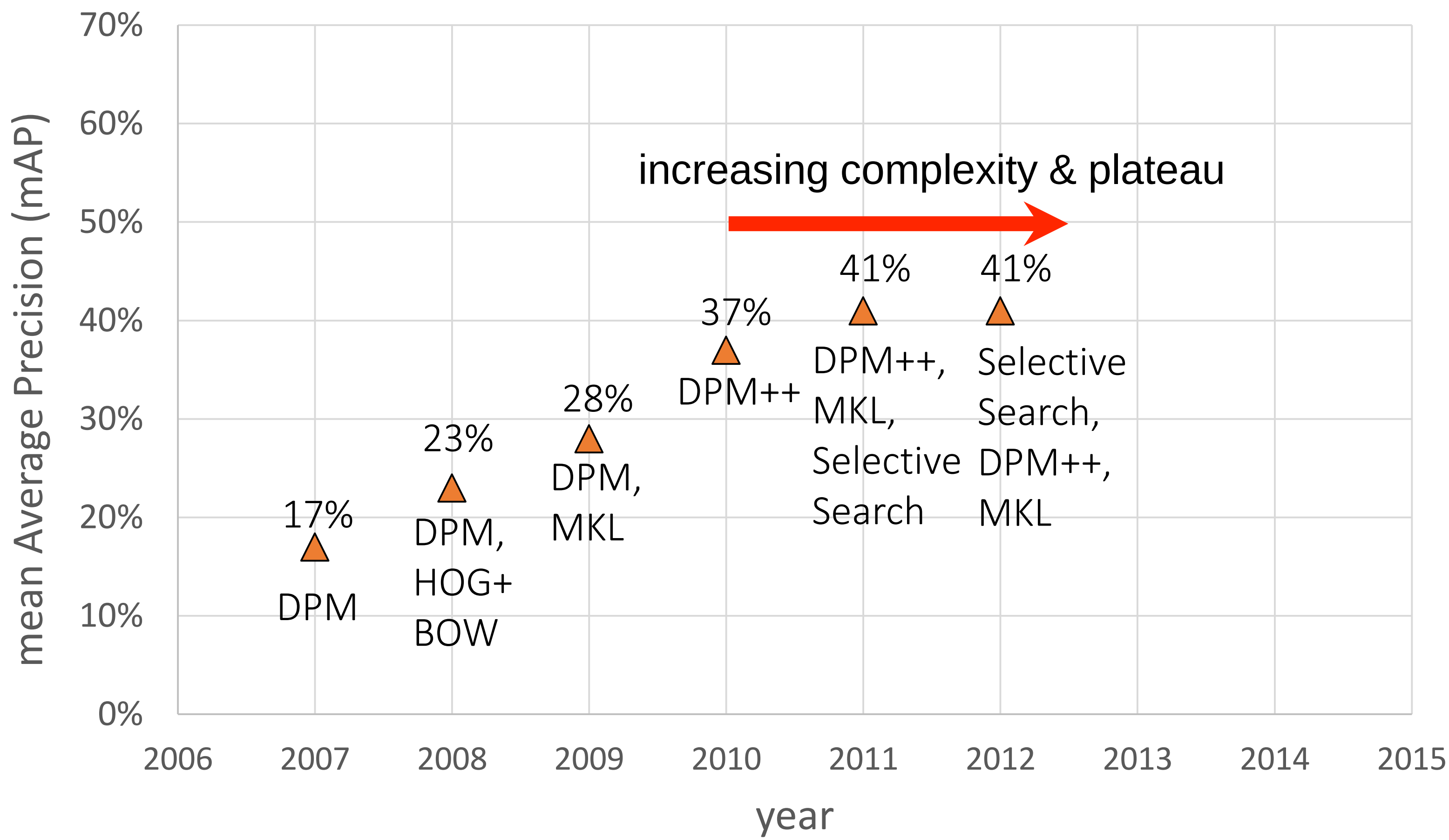
PASCAL VOC detection history



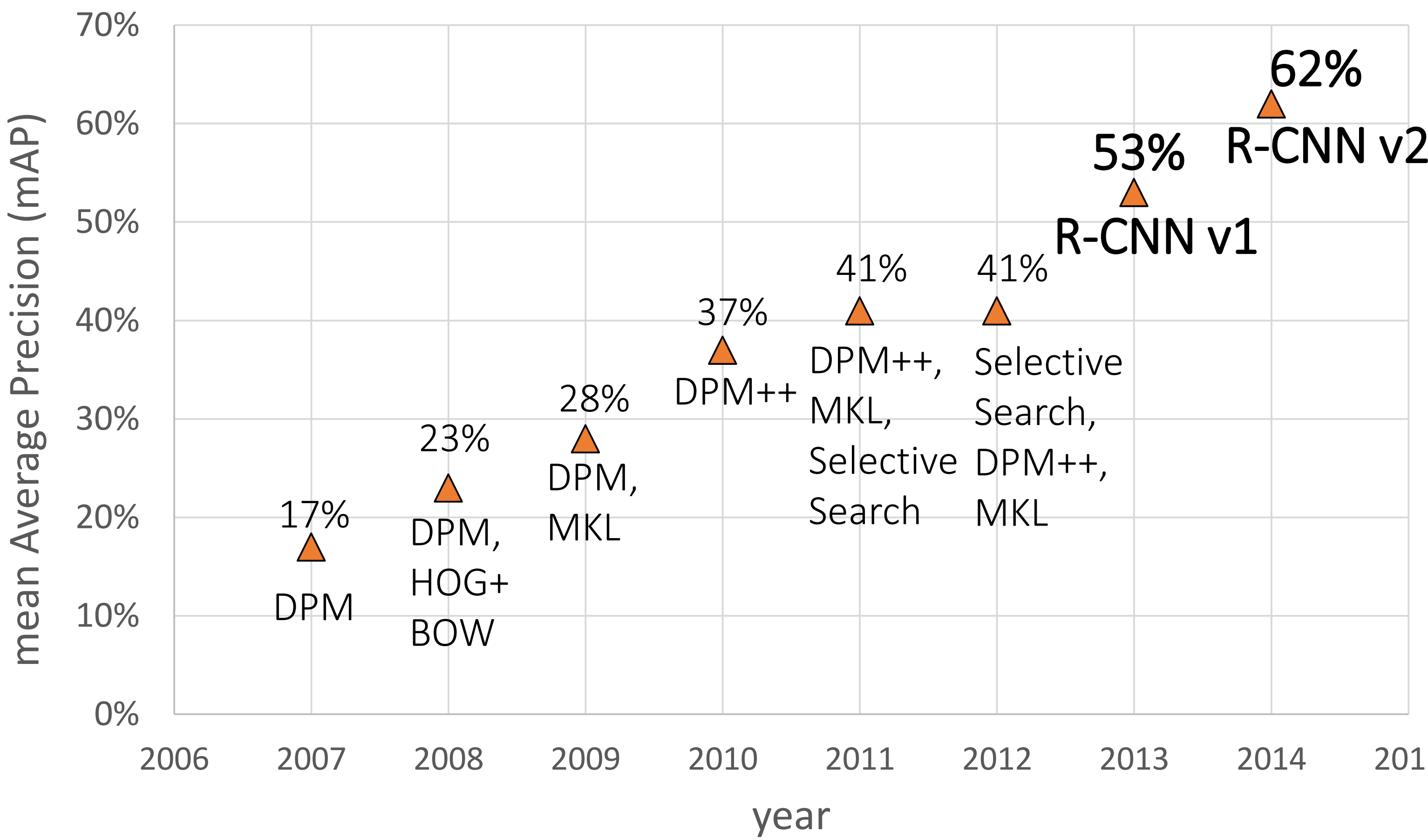
Part-based models & multiple features (MKL)



Kitchen-sink approaches

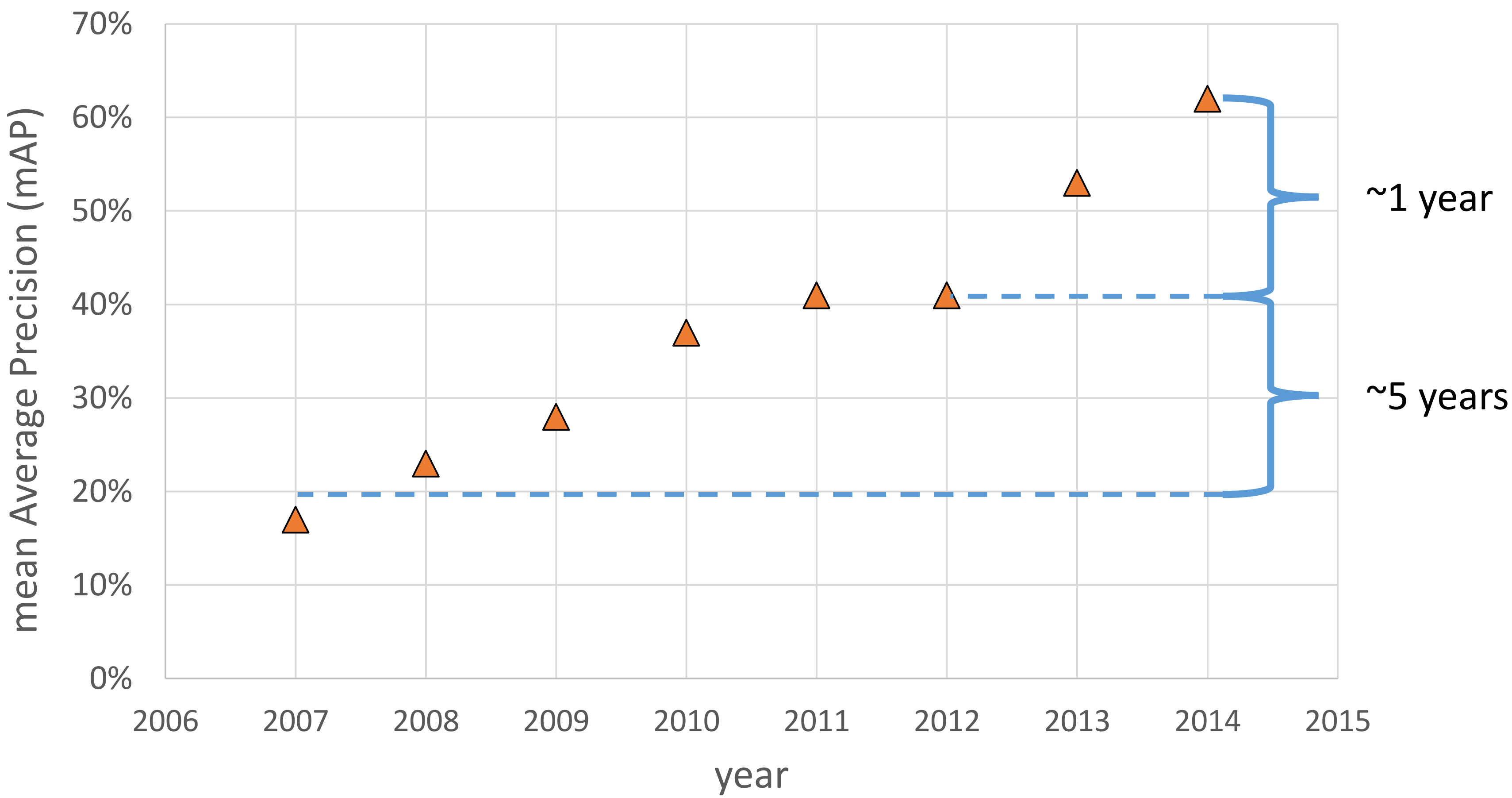


Region-based Convolutional Networks (R-CNNs)



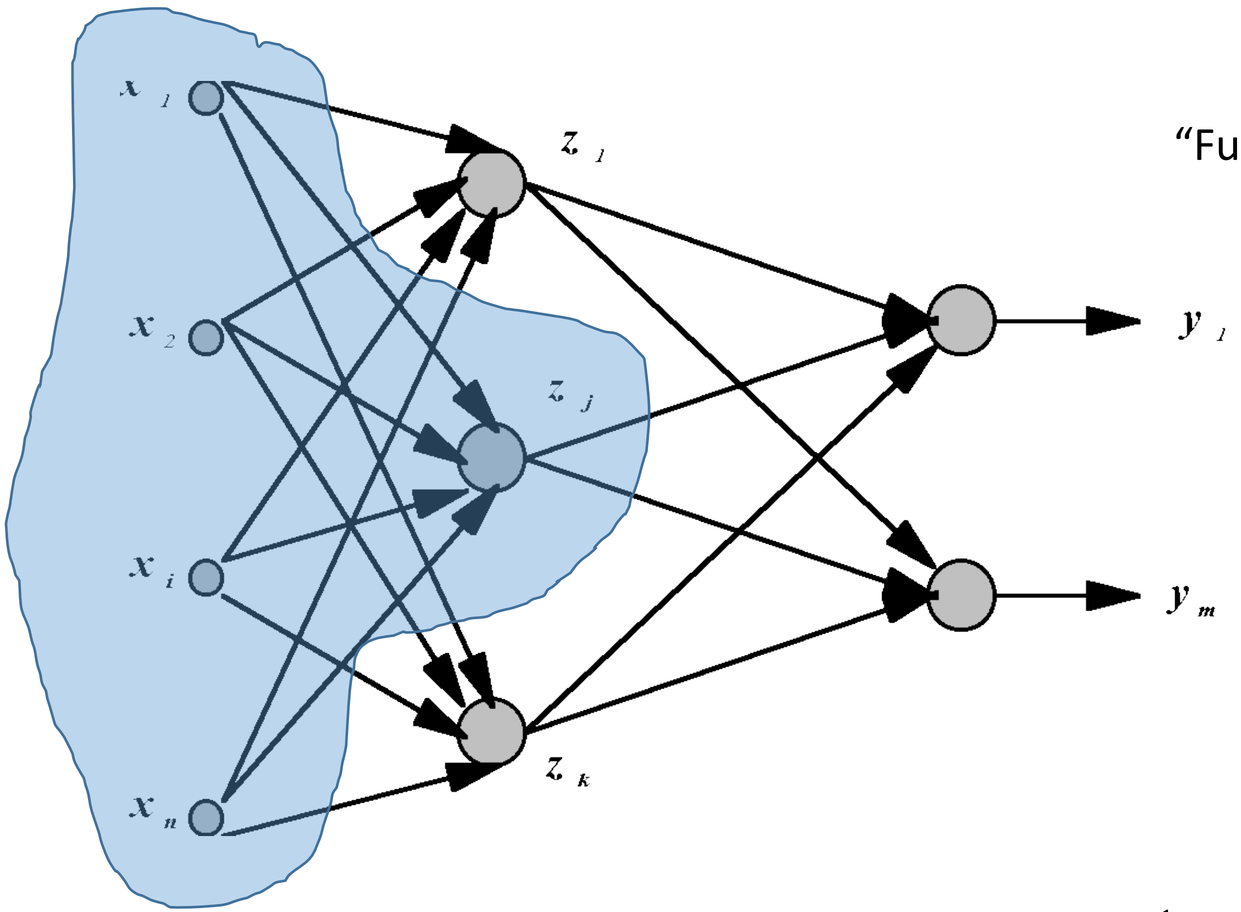
[R-CNN. Girshick et al. CVPR 2014]

Region-based Convolutional Networks (R-CNNs)



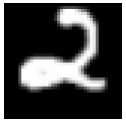
[R-CNN. Girshick et al. CVPR 2014]

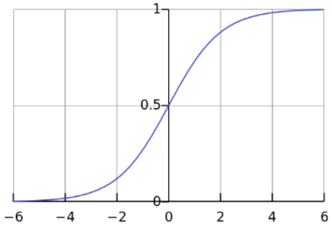
Standard Neural Networks



"Fully connected"

$\boldsymbol{x} = (x_1, \ldots, x_{784})^T$ $\quad z_j = g(\boldsymbol{w}_j^T \boldsymbol{x})$ $\quad g(t) = \dfrac{1}{1 + e^{-t}}$





From NNs to Convolutional NNs

- Local connectivity
- Shared (“tied”) weights
- Multiple feature maps
- Pooling

Just-in-Time Information

- What is a convolution?
- In signal processing, a correlation is an operation that multiplies a small mask times a small piece of the image. These are examples of such masks.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

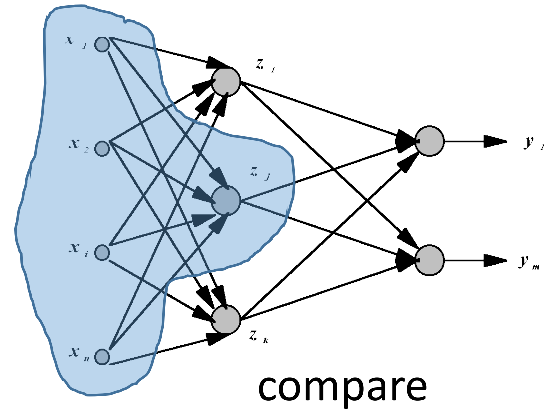
+1	+2	+1
0	0	0
-1	-2	-1

Gy

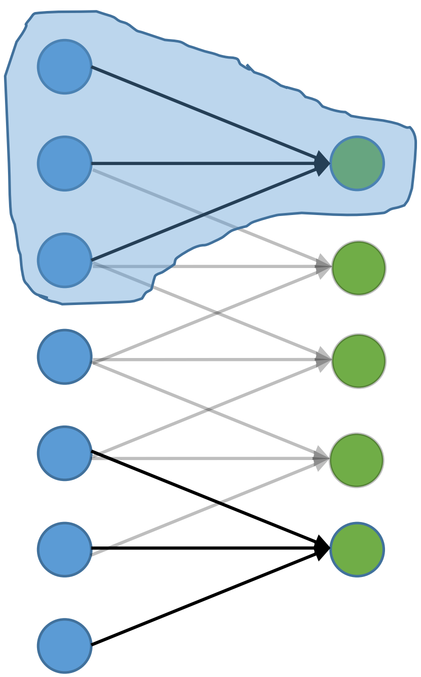
- The strict definition of convolution flips the mask.
- But in computer vision, we call everything convolution.

Convolutional NNs

- Local connectivity



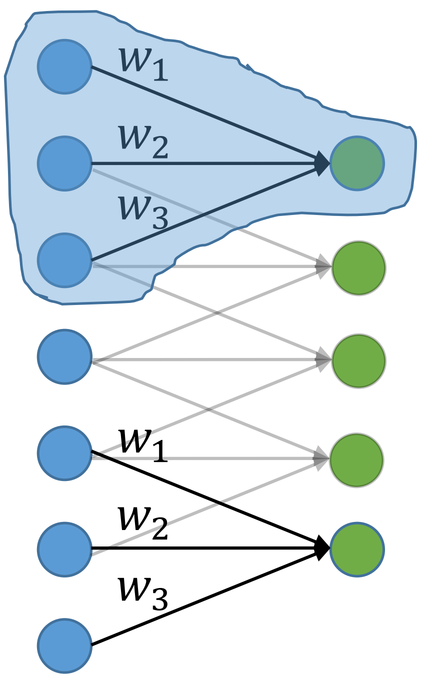
compare



- Each green unit is only connected to (3) **neighboring** blue units

Convolutional NNs

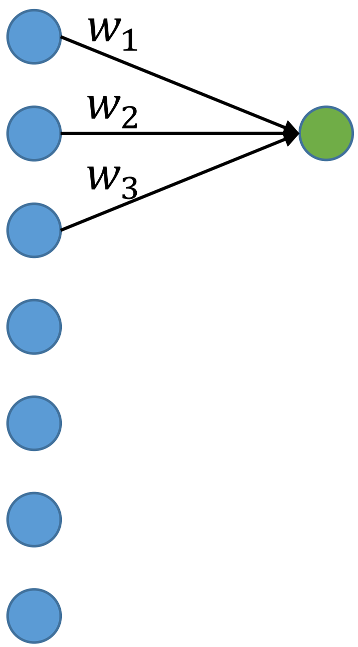
- Shared ("tied") weights



- All green units **share** the same parameters $\boldsymbol{w}$
- Each green unit computes the **same function,** but with a **different input window**

Convolutional NNs

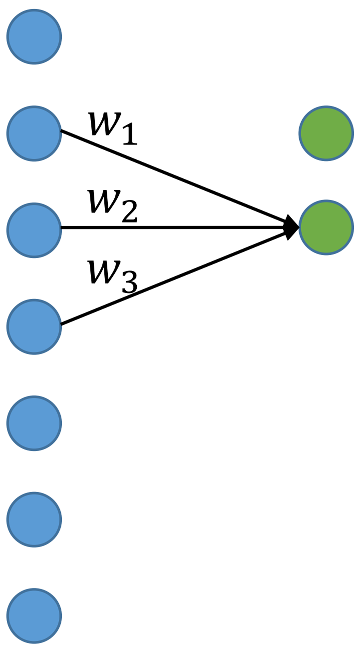
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- All green units **share** the same parameters $\boldsymbol{w}$
- Each green unit computes the **same function,** but with a **different input window**

Convolutional NNs

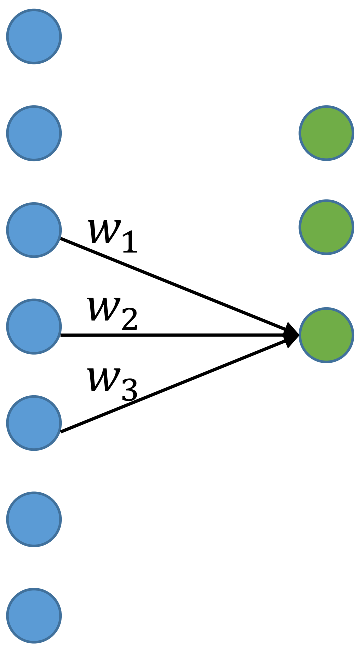
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- All green units **share** the same parameters $\boldsymbol{w}$
- Each green unit computes the **same function,** but with a **different input window**

Convolutional NNs

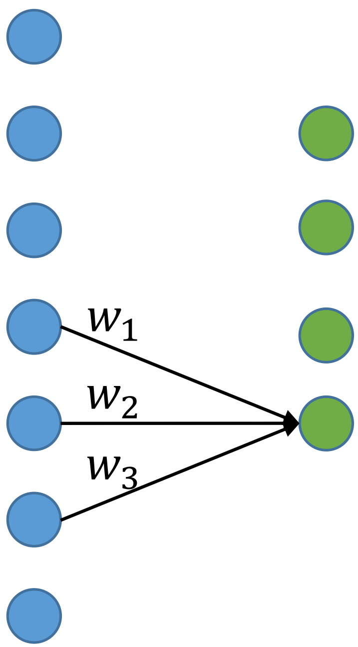
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- All green units **share** the same parameters $\boldsymbol{w}$
- Each green unit computes the **same function,** but with a **different input window**

Convolutional NNs

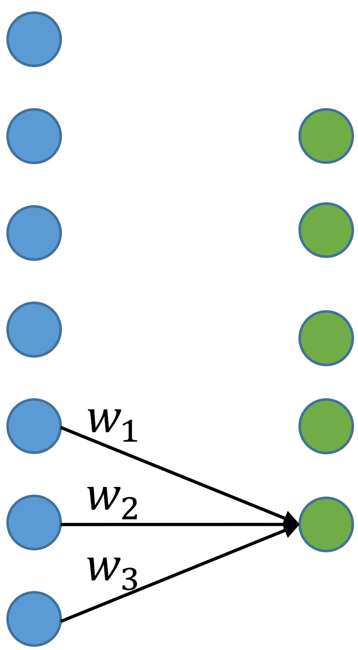
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- All green units **share** the same parameters $\boldsymbol{w}$
- Each green unit computes the **same function,** but with a **different input window**

Convolutional NNs

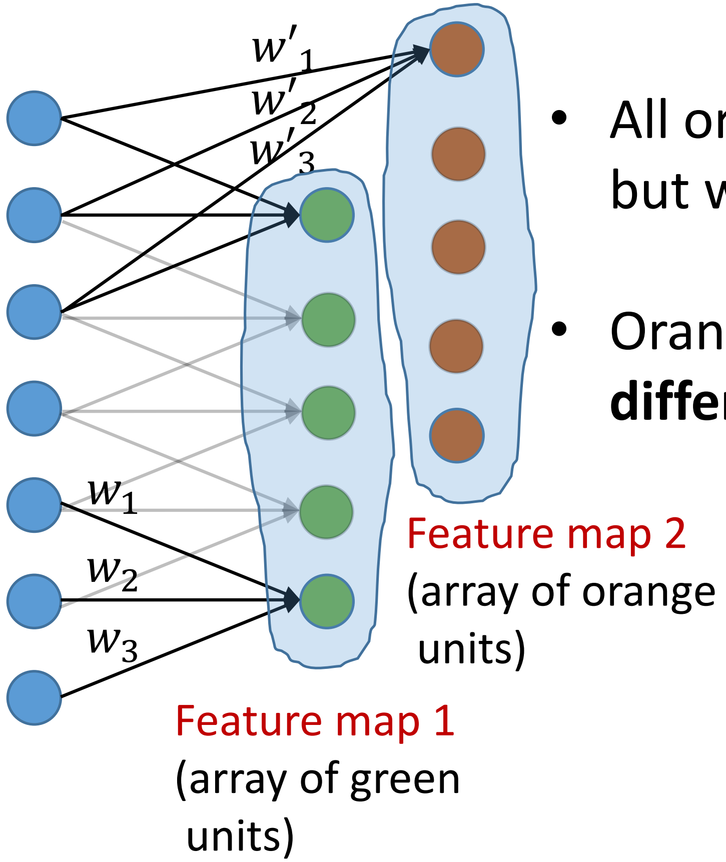
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- All green units **share** the same parameters $\boldsymbol{w}$
- Each green unit computes the **same function,** but with a **different input window**

Convolutional NNs

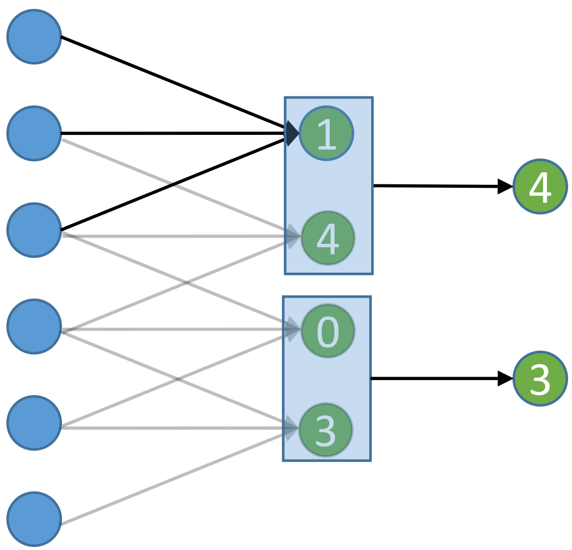
- Multiple feature maps



- All orange units compute the **same function** but with a **different input windows**
- Orange and green units **compute different functions**

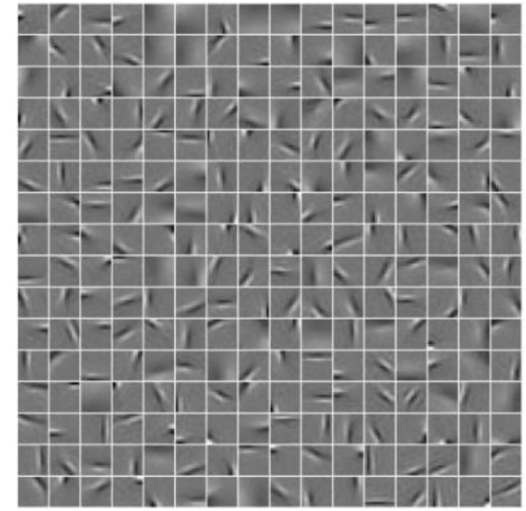
Convolutional NNs

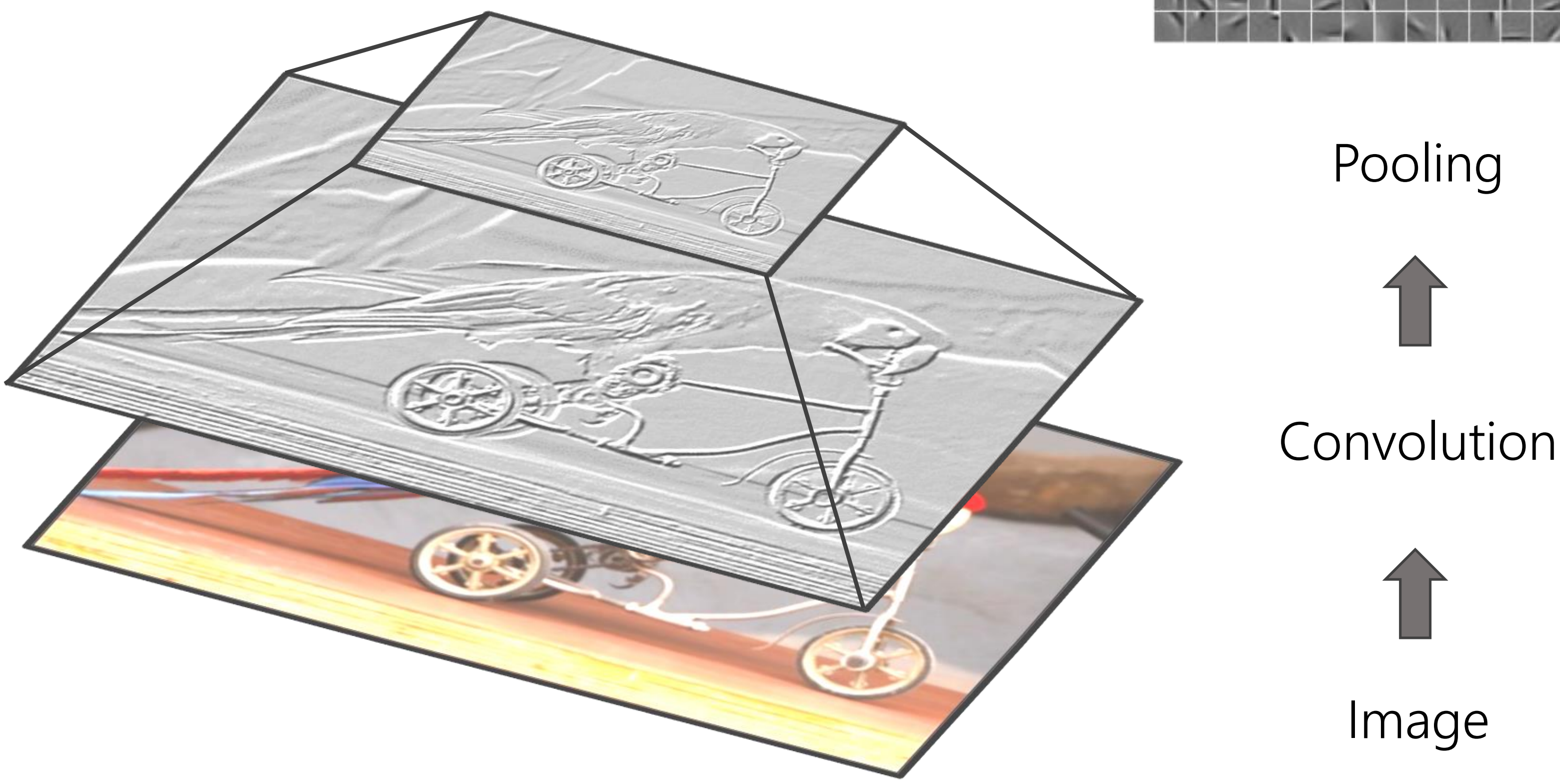
- Pooling (max, average)



- Pooling area: 2 units
- Pooling stride: 2 units
- **Subsamples** feature maps

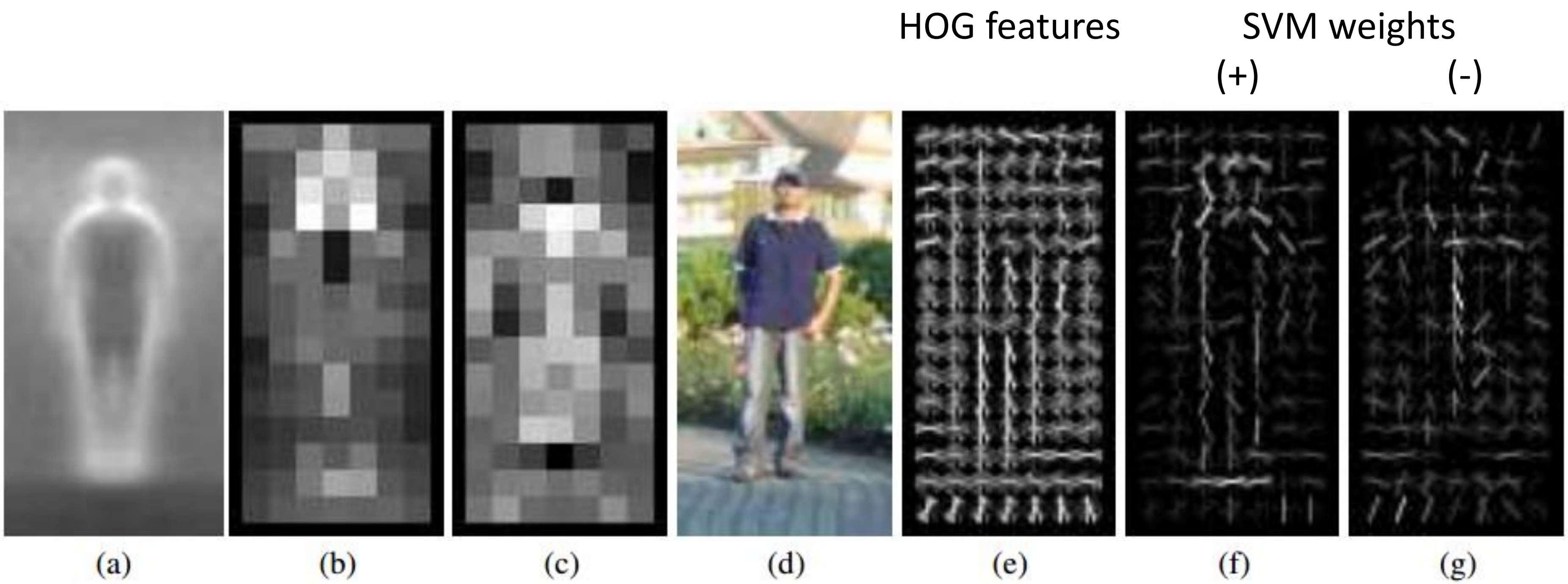
2D input





The key to SVMs

- *It's all about the features*



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

Core idea of “deep learning”

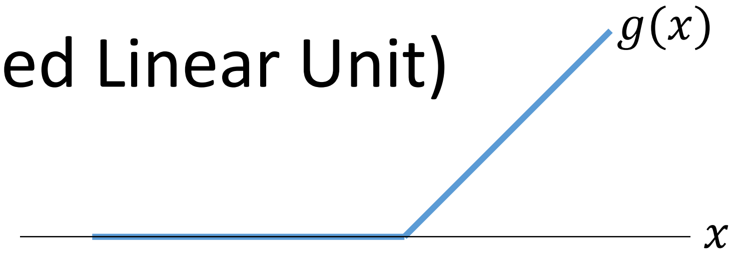
- Input: the “*raw*” signal (image, waveform, …)

- Features: hierarchy of features is *learned* from the raw input

- If SVMs killed neural nets, how did they come back (in computer vision)?

What's new since the 1980s?

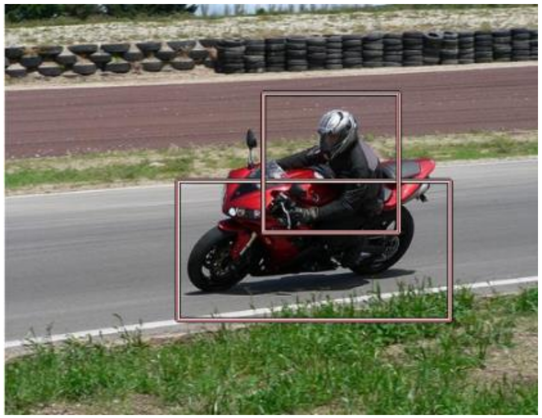
- More layers
  - LeNet-3 and LeNet-5 had 3 and 5 learnable layers
  - Current models have 8 – 20+
- "ReLU" non-linearities (Rectified Linear Unit)
  - $g(x) = \max(0, x)$
  - Gradient doesn't vanish



- "Dropout" regularization
- Fast GPU implementations
- More data

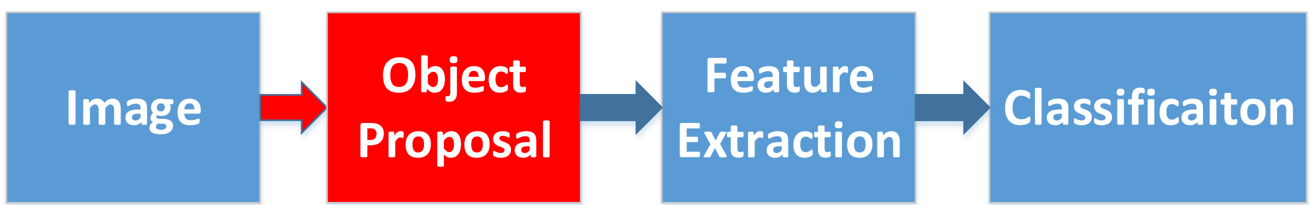
What else? Object Proposals

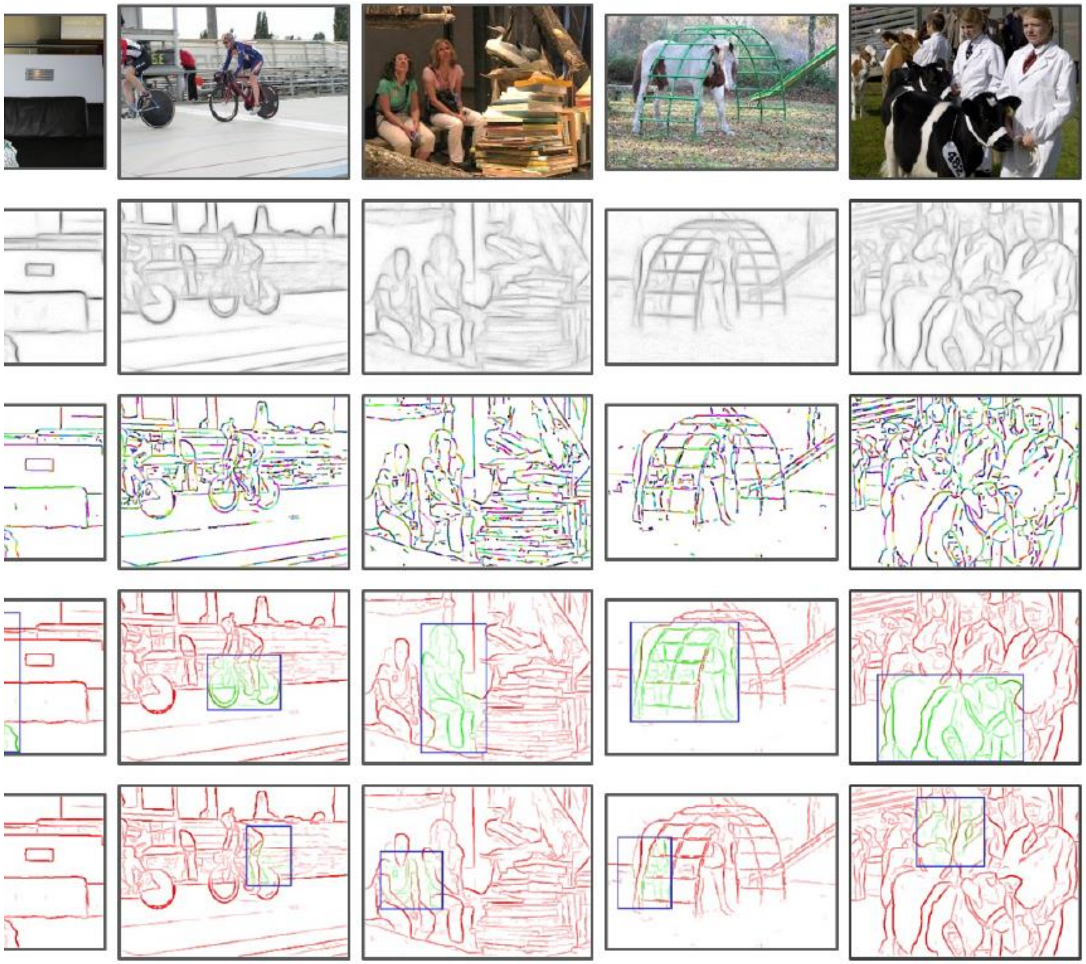
- Sliding window based object detection



Iterate over window size, aspect ratio, and location

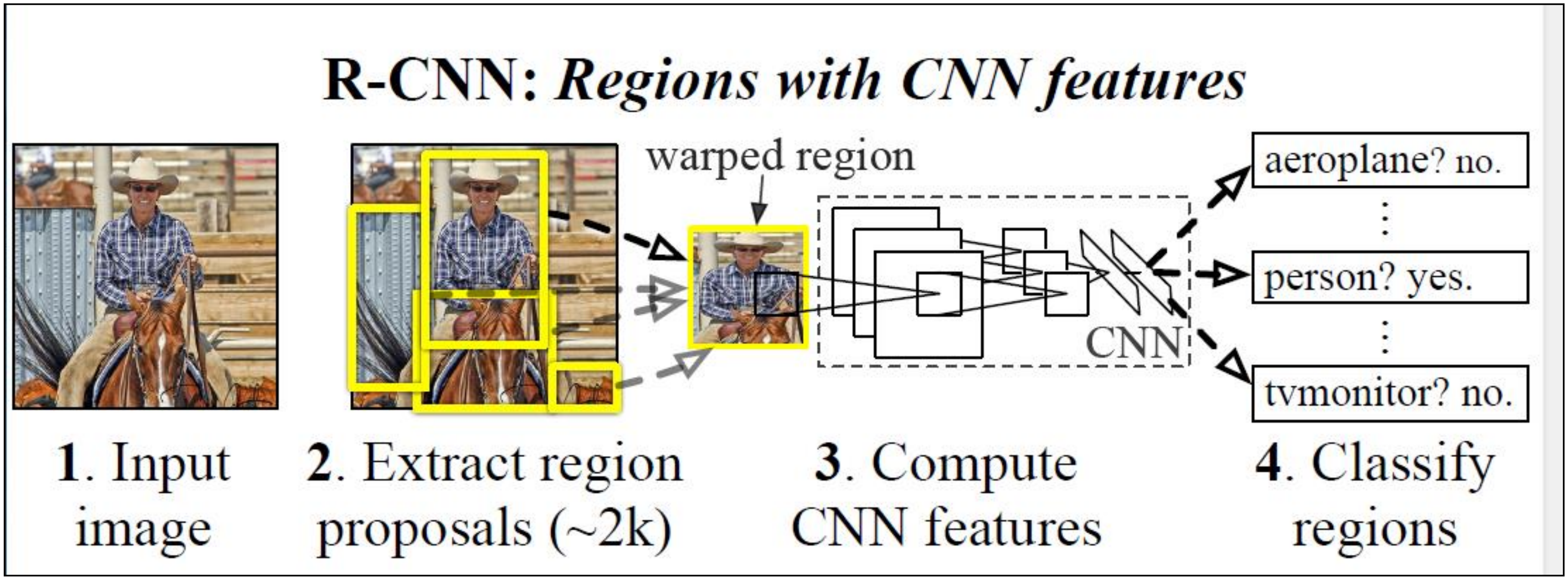
- Object proposals
  - Fast execution
  - High recall with low # of candidate boxes





The number of contours wholly enclosed by a bounding box is indicative of the likelihood of the box containing an object.

Ross’s Own System: Region CNNs



Competitive Results

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [20]†	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [39]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [41]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [18]†	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	**71.8**	**65.8**	**53.0**	**36.8**	**35.9**	**59.7**	**60.0**	**69.9**	**27.9**	**50.6**	**41.4**	**70.0**	**62.0**	**69.0**	**58.1**	**29.5**	**59.4**	**39.3**	**61.2**	**52.4**	**53.7**

Table 1: Detection average precision (%) on VOC 2010 test. R-CNN is most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding-box regression (BB) is described in Section C. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. †DPM and SegDPM use context rescoring not used by the other methods.

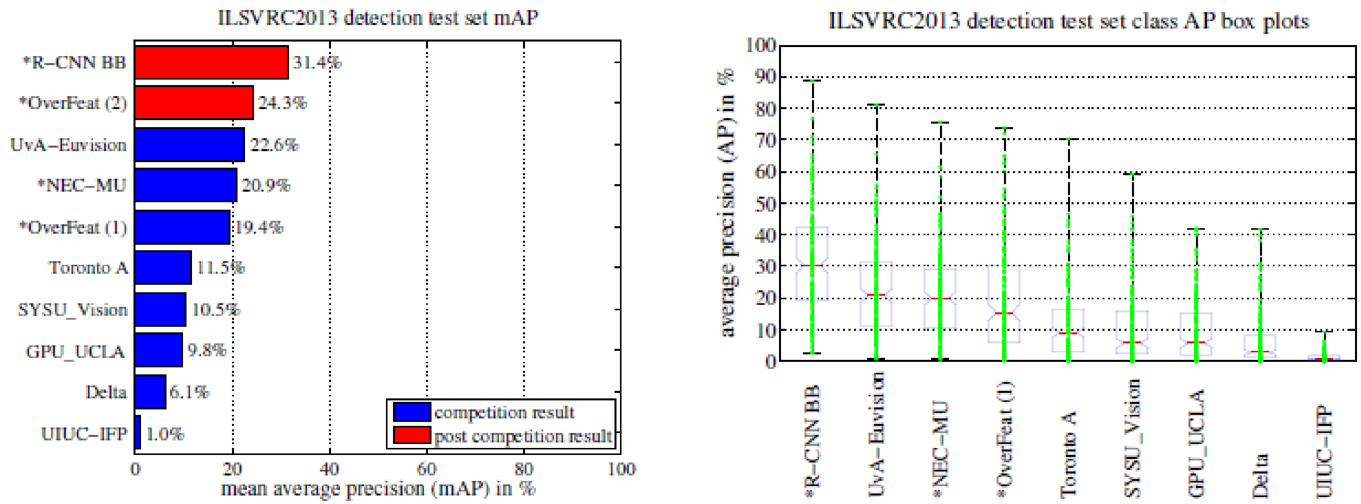


Figure 3: (Left) Mean average precision on the ILSVRC2013 detection test set. Methods preceeded by * use outside training data (images and labels from the ILSVRC classification dataset in all cases). **(Right) Box plots for the 200 average precision values per method.** A box plot for the post-competition OverFeat result is not shown because per-class APs are not yet available (per-class APs for R-CNN are in Table 8 and also included in the tech report source uploaded to arXiv.org; see `R-CNN-ILSVRC2013-APs.txt`). The red line marks the median AP, the box bottom and top are the 25th and 75th percentiles. The whiskers extend to the min and max AP of each method. Each AP is plotted as a green dot over the whiskers (best viewed digitally with zoom).

Top Regions for Six Object Classes

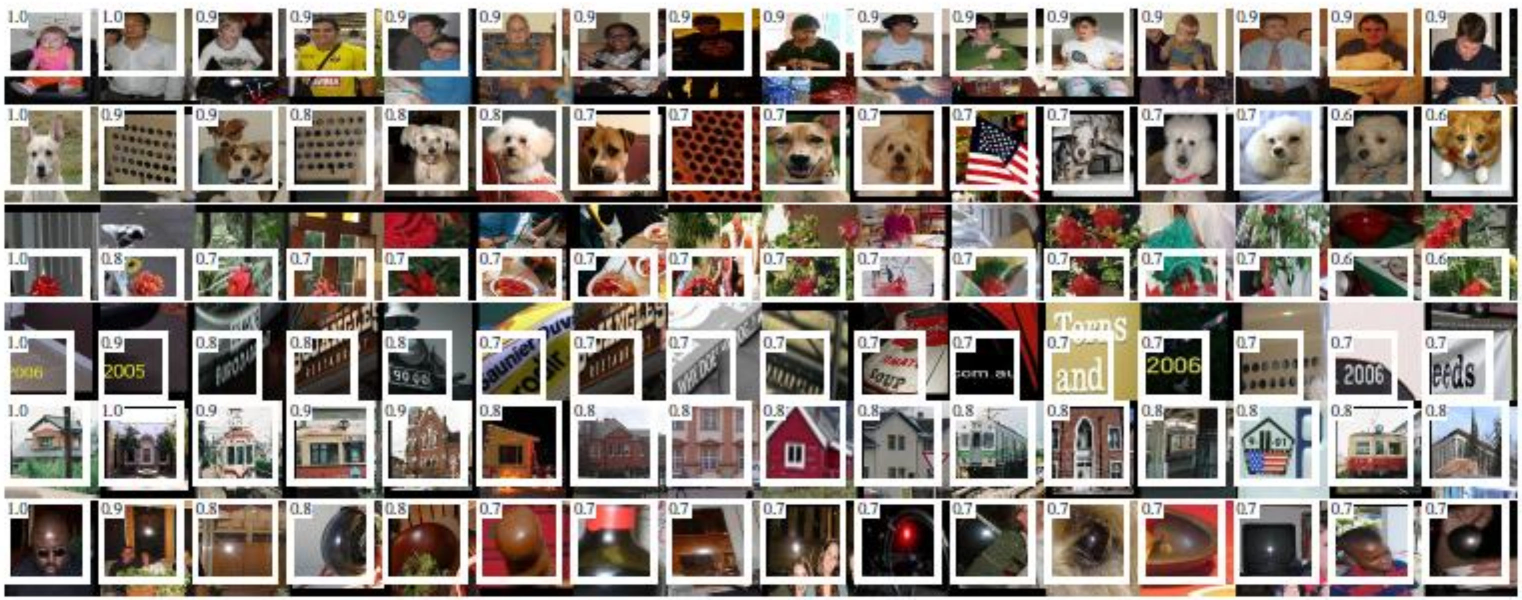


Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).