Image representations, from shallow to deep

Andrea Vedaldi

BMVC 2014 Tutorial

UNIVERSITY OF OXFORD

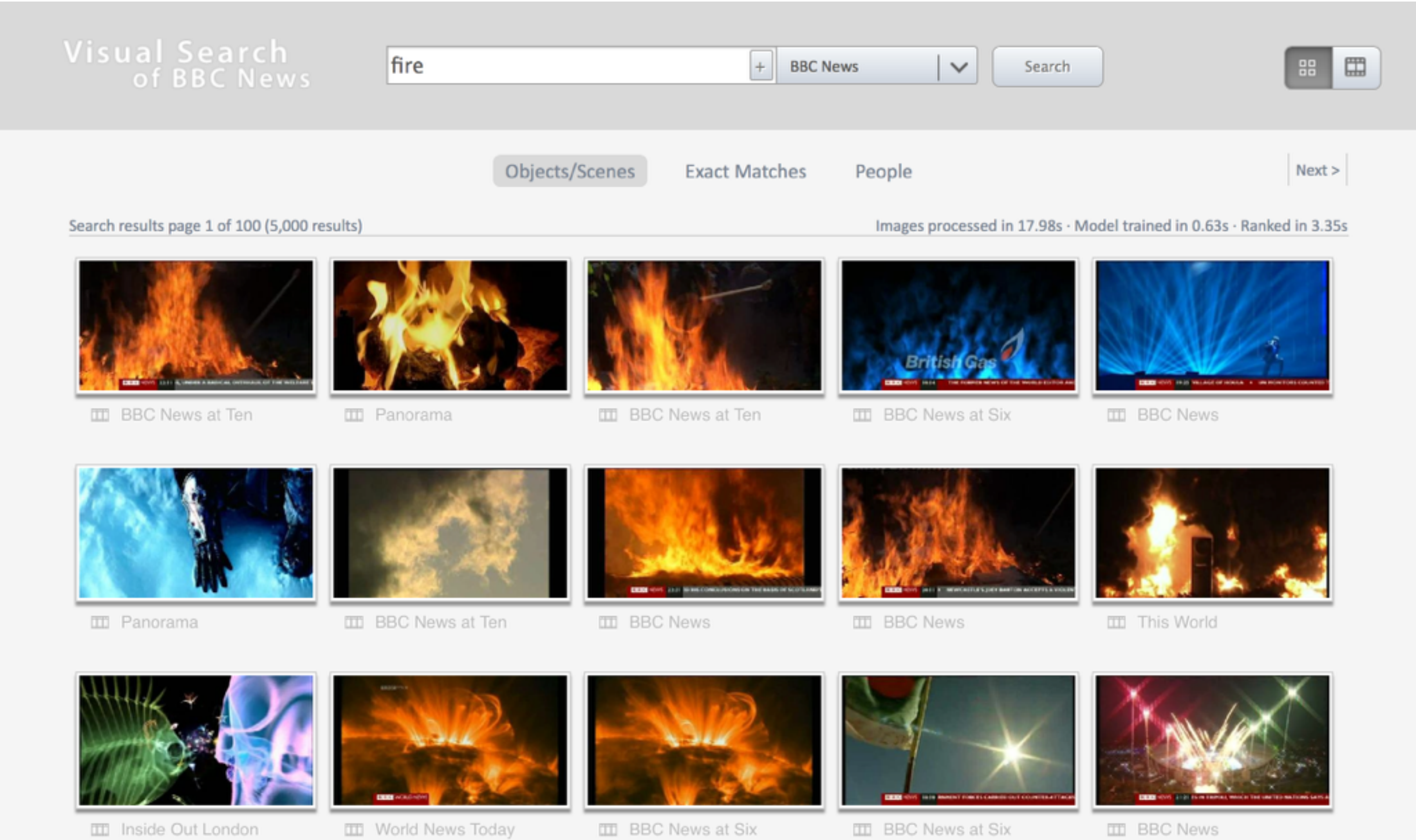
Demo: image search

http://www.robots.ox.ac.uk/~vgg/research/on-the-fly/

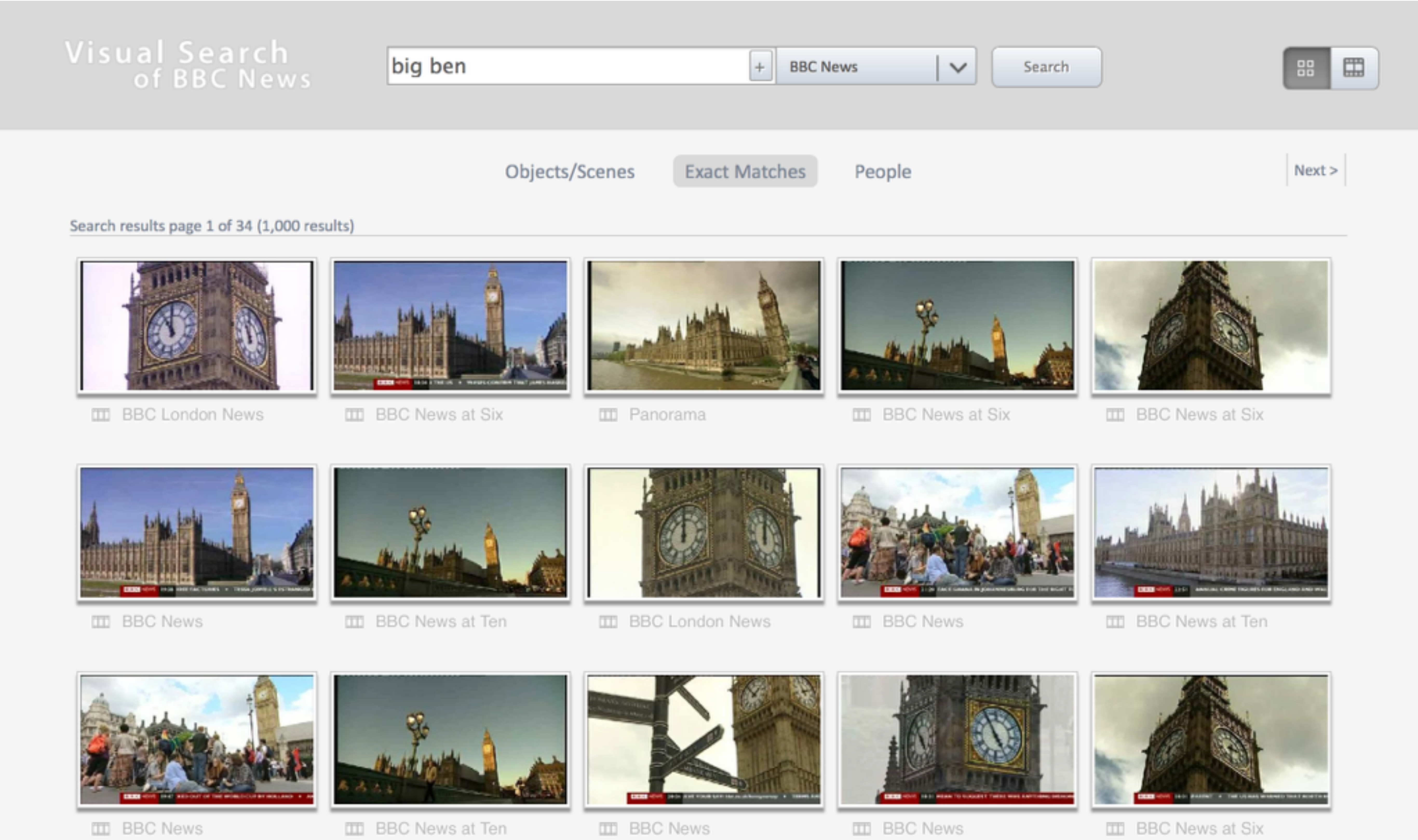
BBC Research & Development explains how their work with Oxford University is opening up new ways to search archive footage.[1]

[1]http://www.bbc.co.uk/informationandarchives/archivenews/2014/face-recognition-and-new-ways-to-search-for-archive.html

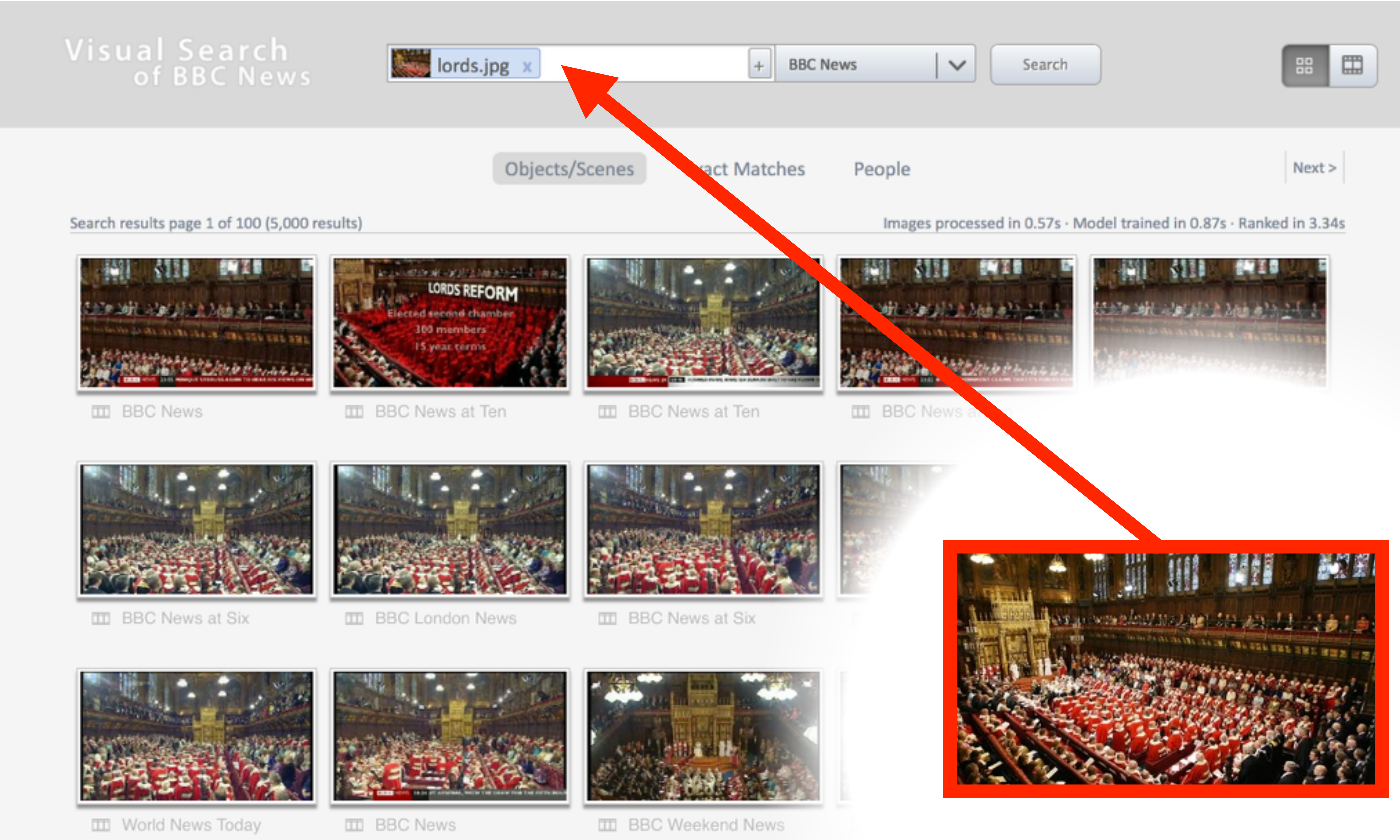
Searching by type



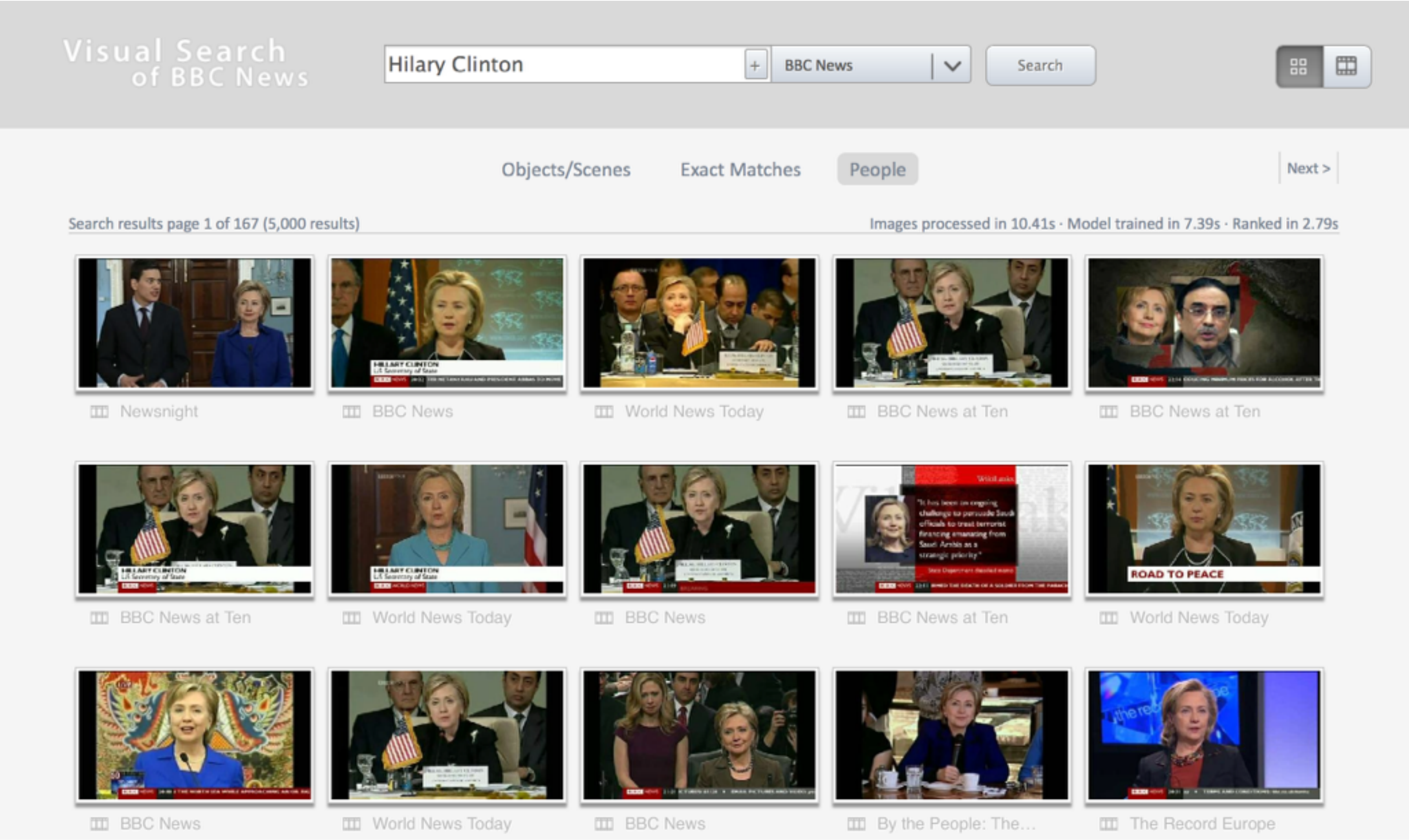
Searching by instance



Search by example

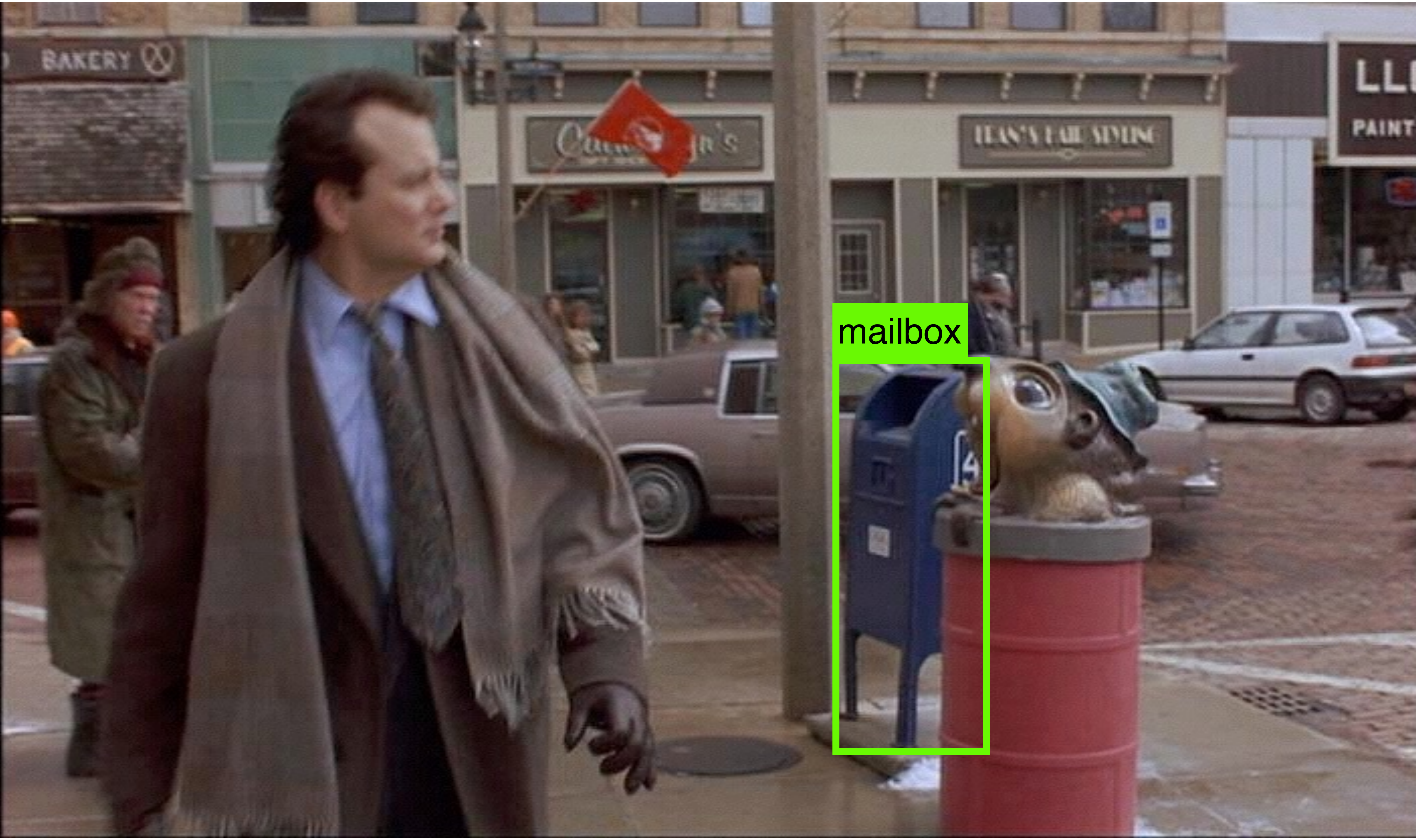


Searching by identity



Challenges: intra-class variation

Challenges: viewpoint, occlusions, clutter, illumination, …



Challenges: size

BBC Footage Duration	# of Frames	# of Keyframes	Footprint	Faces Detected
3 - 40 K hours	10 - 150 M	3 - 35 M	1 - 10 TB	5 - 20 M

Learn objects, people on the fly

- Build models for new queries on the spot

Respond fast

- Search millions of frames in a few seconds

Small footprint

- Index millions of frames in RAM

Many other applications

Exemplified applications

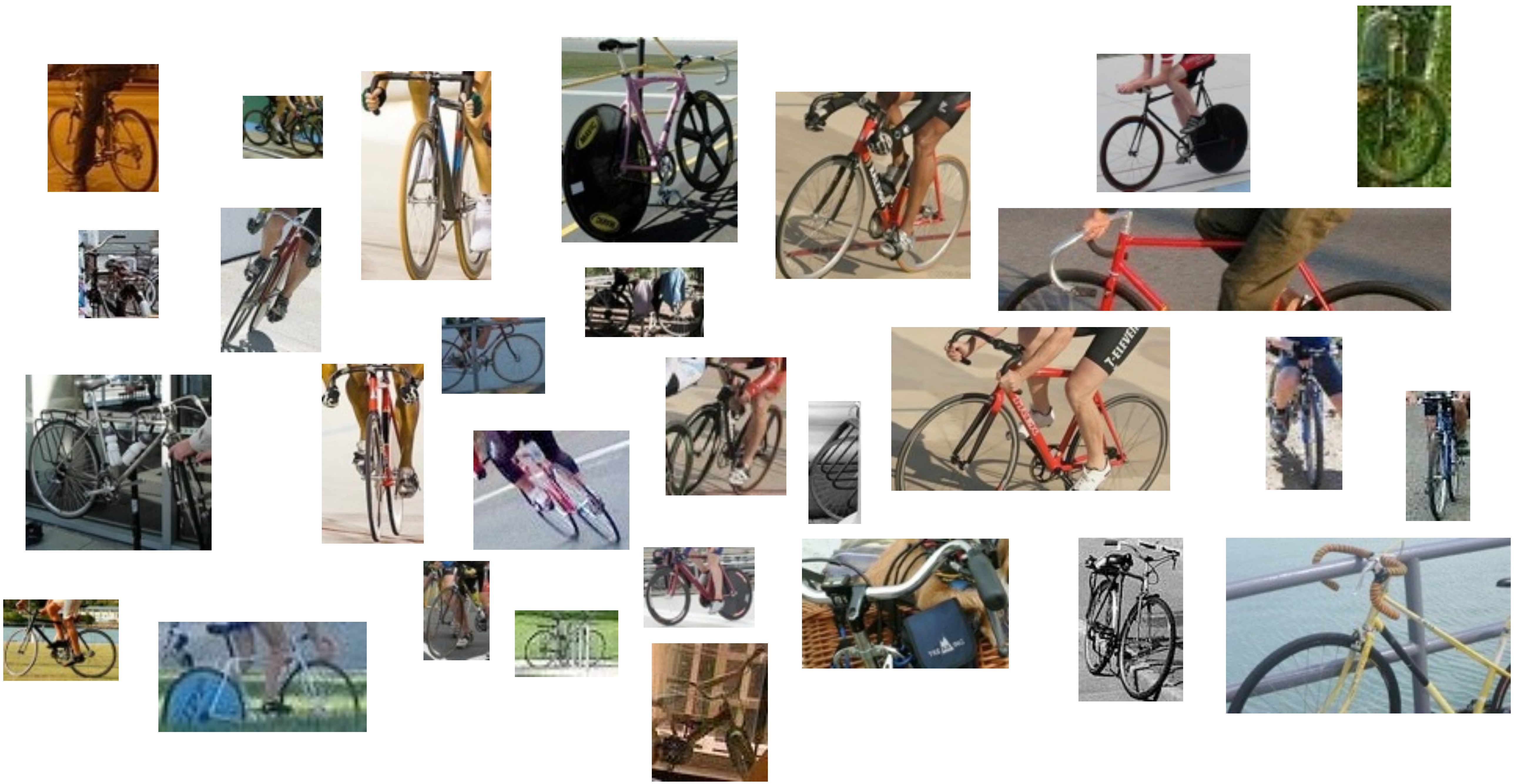
- Object **category recognition**
- Object **instance retrieval**
- **Face** detection & recognition

Image representations **apply to most areas of CV**

- **Object detection**
- Visual **tracking**
- 3D **reconstruction**
- Semantic **segmentation**
- **Pose** estimation
- Interactive segmentation
- Material recognition
- …

Image-based object models

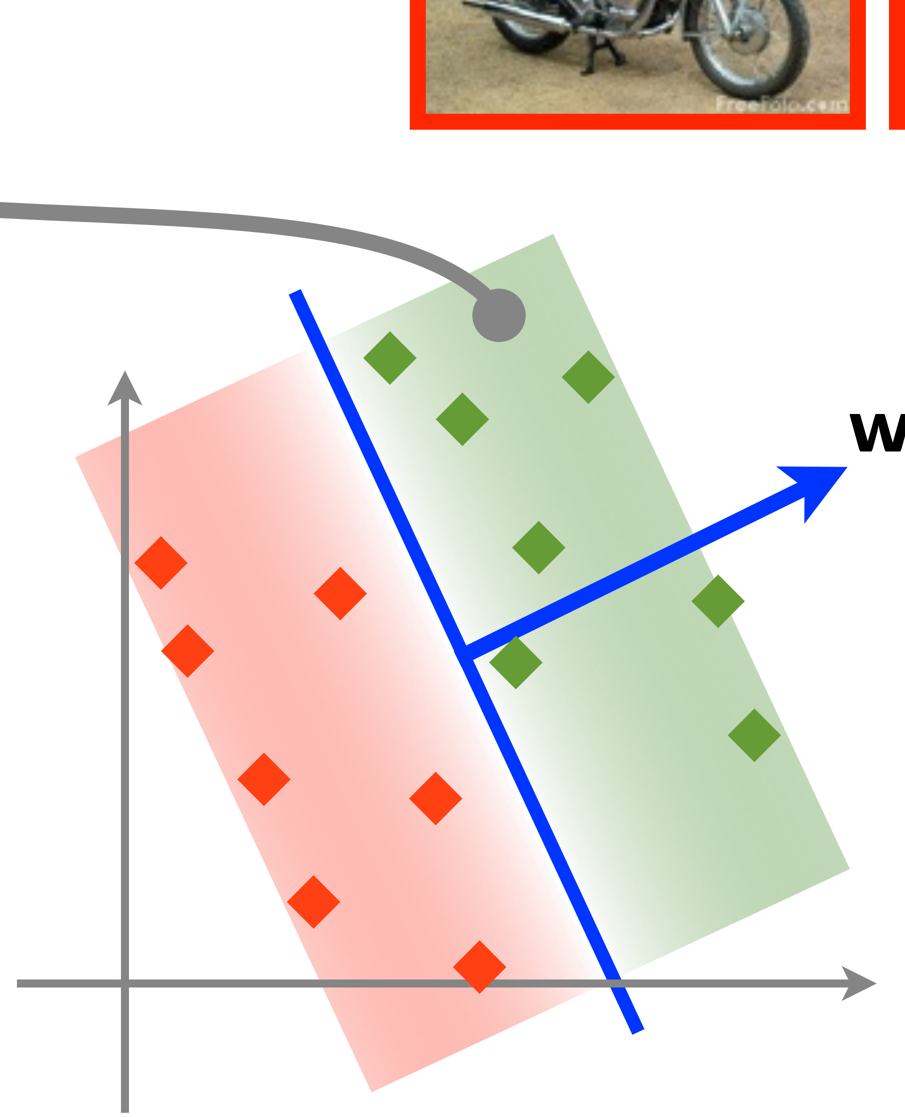
object = **distribution** of **2D patterns**



Linear predictor

bicycle?

x



linear predictor

$$F(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$$

Image representations

Using linear predictors on non-vectorial data

$\mathbf{x}$

encoder Φ $\longrightarrow$ representation $\Phi(\mathbf{x}) \in \mathbb{R}^d$

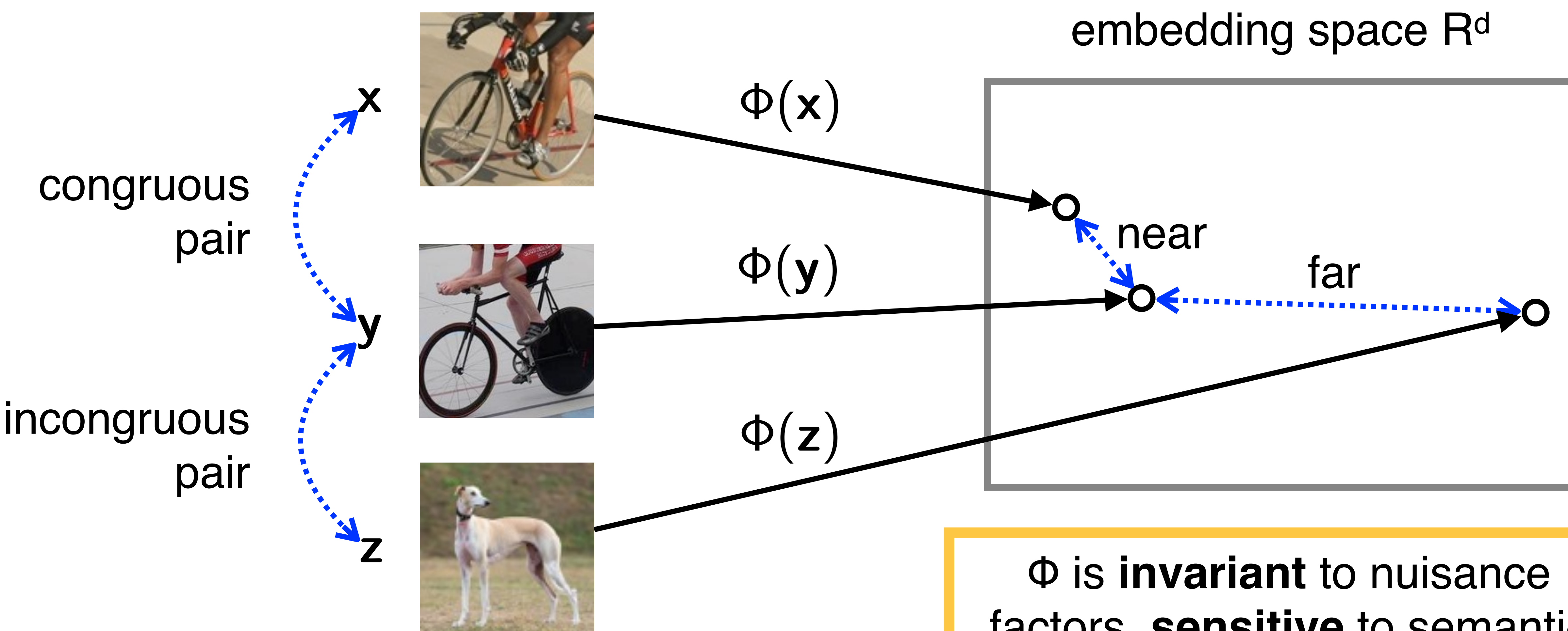
An **encoder** maps the data into a **vectorial representation**

Allows linear predictors to be applied to images, text, sound, videos, …

$$\boxed{F(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle}$$

Meaningful representation

Semantic similarity → representation → Vector similarity (distance)



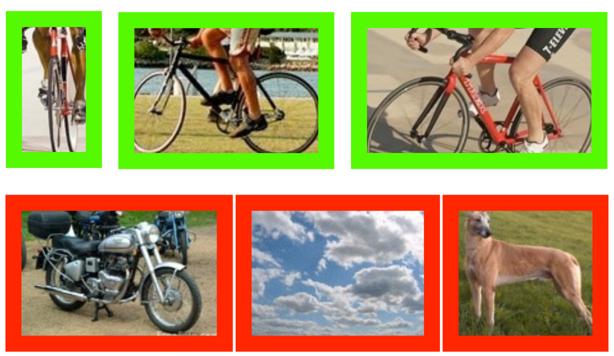
Φ is **invariant** to nuisance factors, **sensitive** to semantic variations

Learning predictors

similarity notion

smoothness hyperparameter

labelled data
($\mathbf{x}_1$,$\mathbf{y}_1$), ($\mathbf{x}_2$,$\mathbf{y}_2$), …



encoder
Φ

learning
large-scale
optimiser

predictor
parameters
w*

$$\mathbf{w}^* = \operatorname{argmin} E(\mathbf{w})$$

Support vector machines

A typical predictor

$$E(\mathbf{w}) = \underbrace{\lambda \frac{\|\mathbf{w}\|^2}{2}}_{\text{… is smooth …}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}}_{\text{… and fits the training data}}$$

The predictor …   … is smooth …   … and fits the training data

Optimisation

- Very large convex problem
- Key insight: **an accurate solution is not required**

O(N) algorithms exist

- Stochastic gradient descent, dual coordinate ascent, …
- Can learn on the fly on thousands or millions of examples

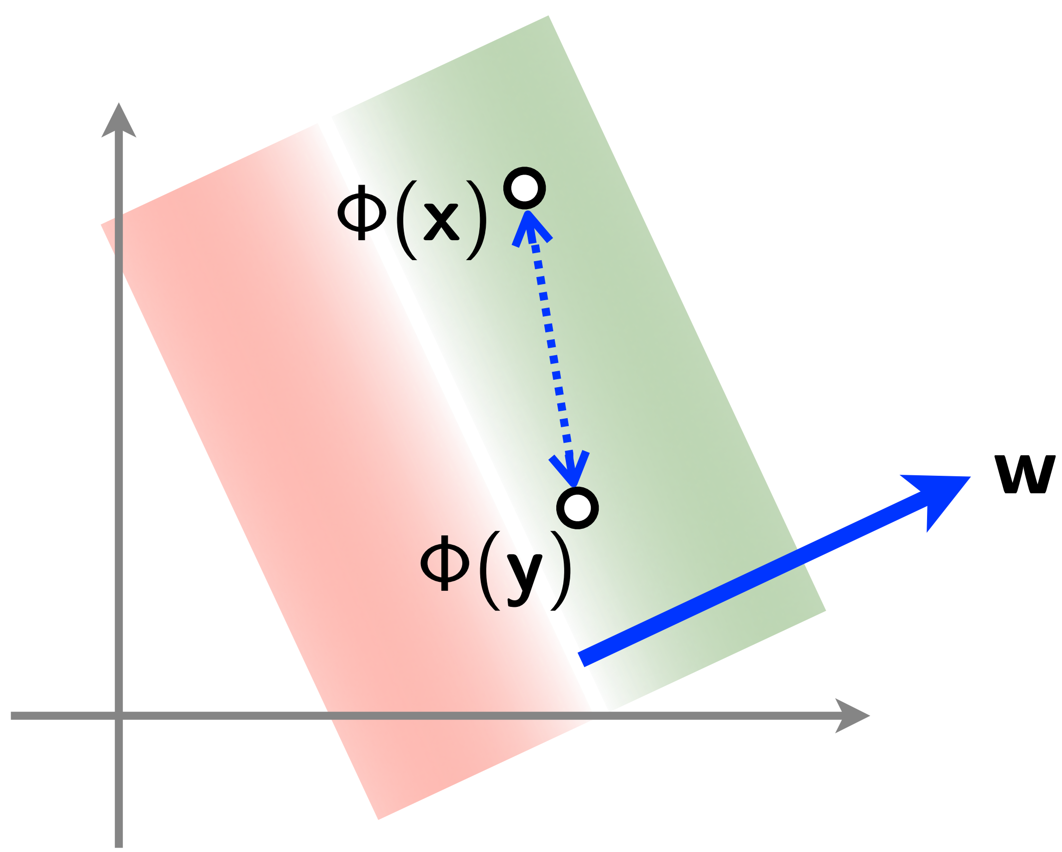
[Learning with Kernels, Schölkopf Smola 2002]

Smoothness and generalisation

Key challenge: **extrapolate the training data**

- Achieved by **smoothness**
- I.e. similar vectors receive similar scores

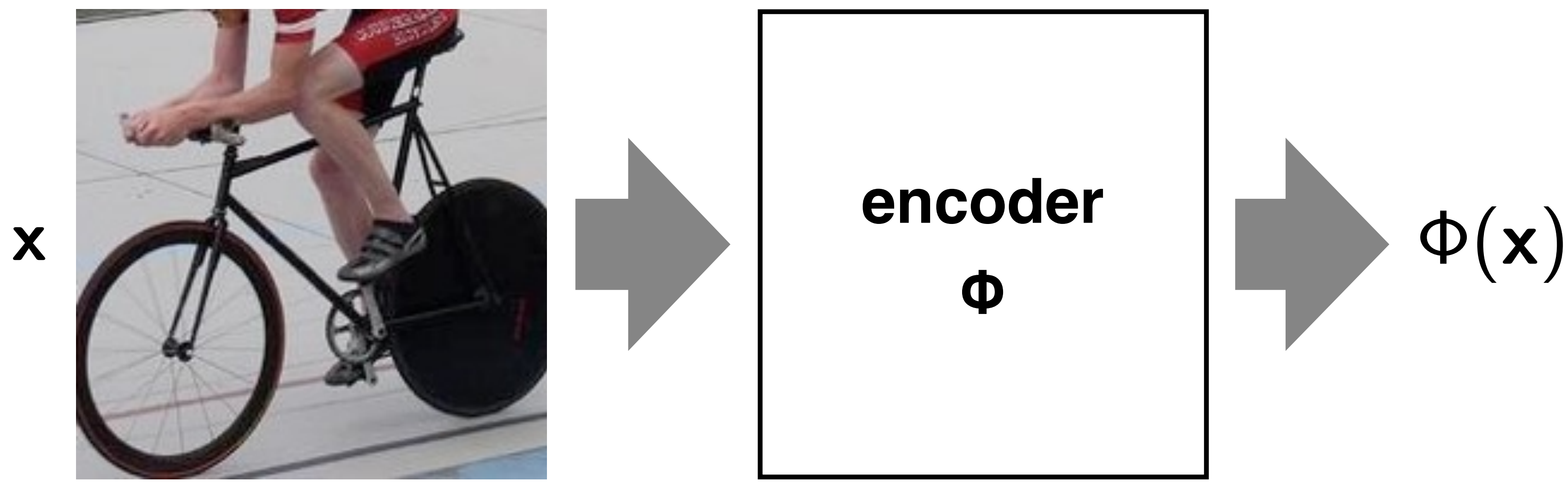
$$(F(\mathbf{x}) - F(\mathbf{y}))^2 = (\langle \mathbf{w}, \Phi(\mathbf{x}) - \Phi(\mathbf{y}) \rangle)^2 \leq \|\mathbf{w}\| \cdot \|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\|$$



linear predictor

$$F(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle$$

Good representations

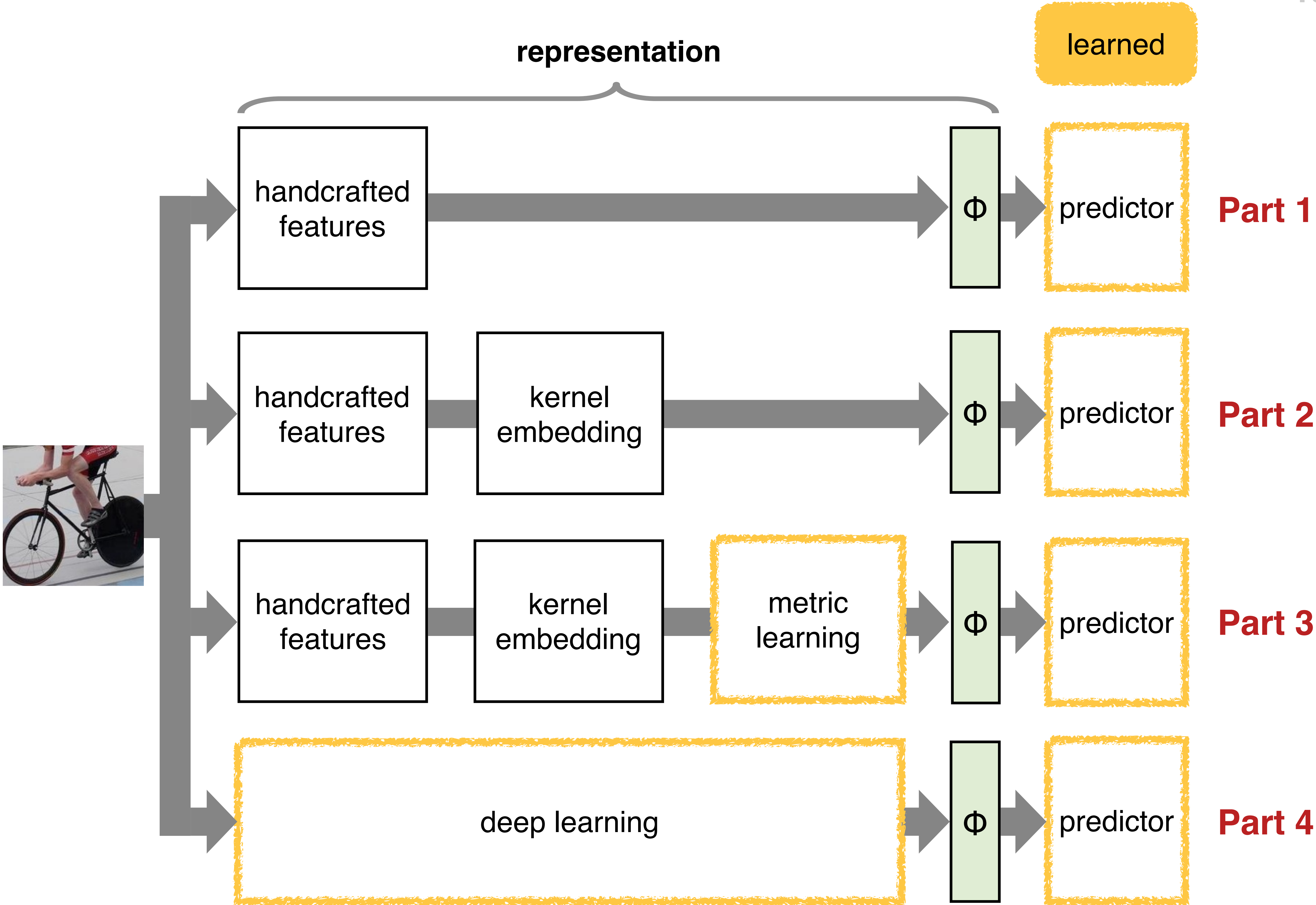


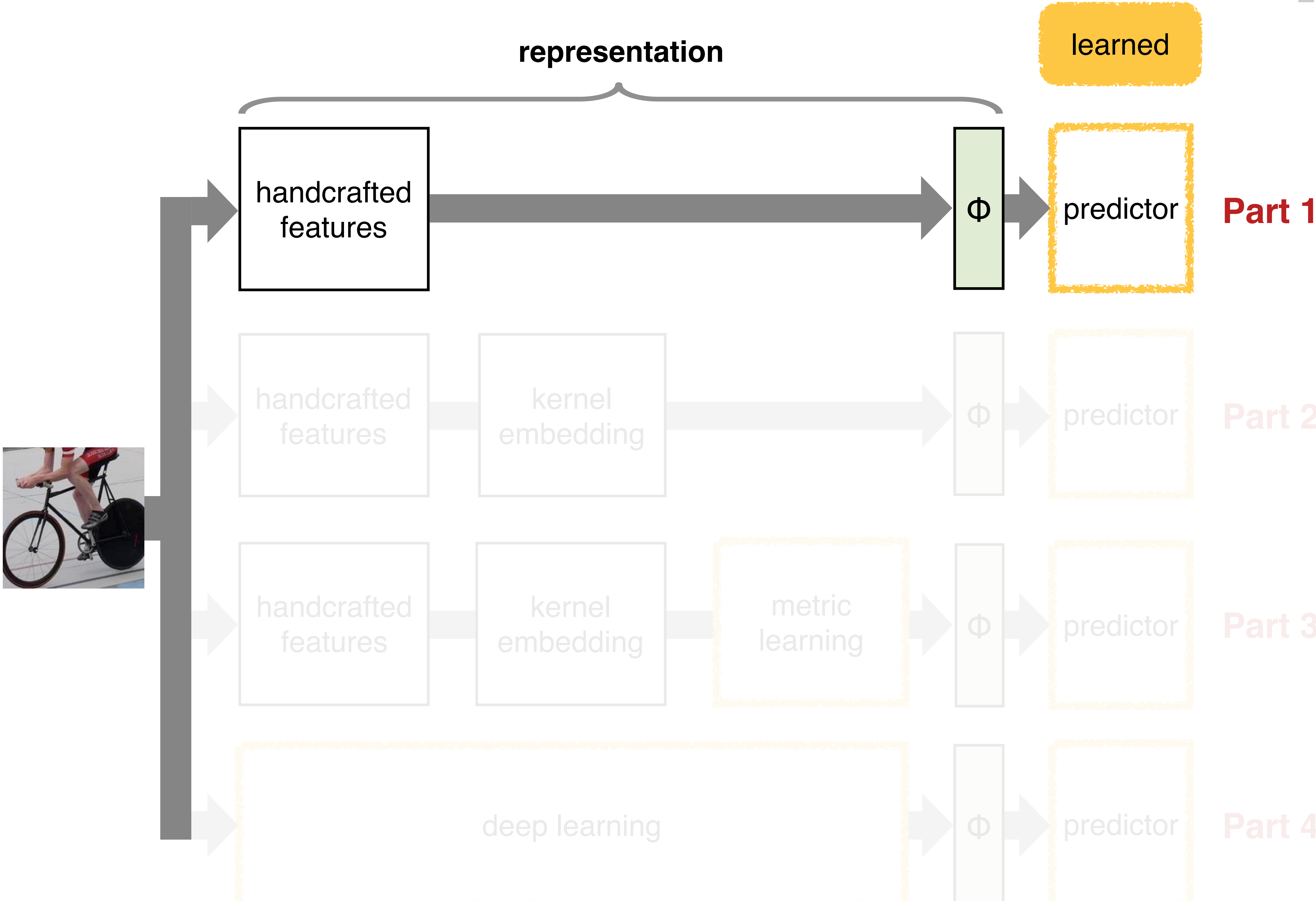
Main desiderata

- **Powerful:** meaningful similarity / untangles factors
- **Cheap:** fast to evaluate (can be computed on the fly)
- **Compact:** small code (takes little RAM, disk, IO)

Others

- Easy to learn (when not hand-crafted)
- Easy to implement


learned
representation
handcrafted features
Φ
predictor
Part 1
handcrafted features
kernel embedding
Φ
predictor
Part 2
handcrafted features
kernel embedding
metric learning
Φ
predictor
Part 3
deep learning
Φ
predictor
Part 4

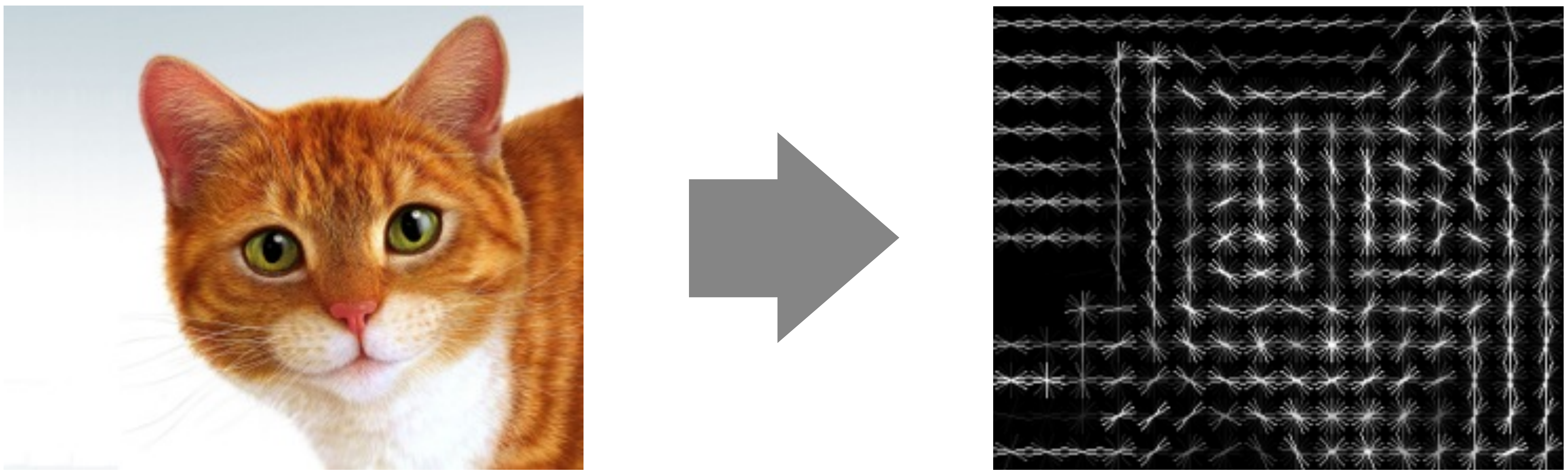


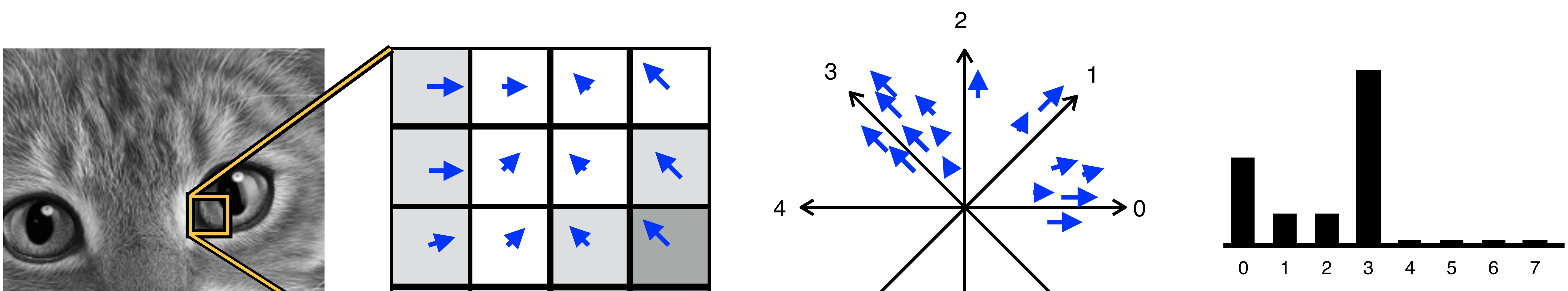
learned
representation
handcrafted features
Φ
predictor
Part 1
handcrafted features
kernel embedding
Φ
predictor
Part 2
handcrafted features
kernel embedding
metric learning
Φ
predictor
Part 3
deep learning
Φ
predictor
Part 4

Histogram of Oriented Gradients

[Lowe 1999, Dalal & Triggs 2005]

HOG captures the local gradient (edge) orientations in the image

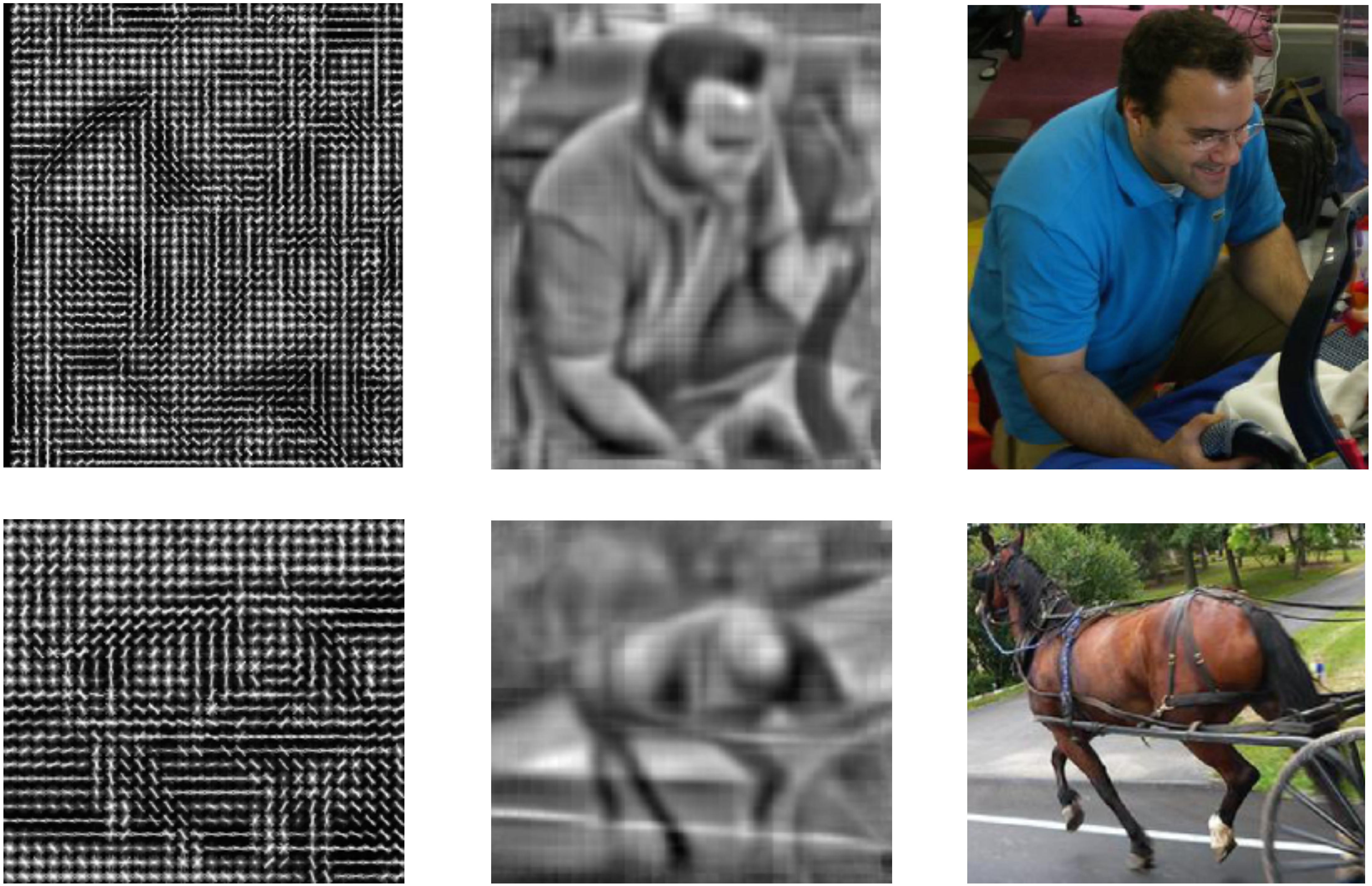




+ block l^2 normalisation

HOG challenge

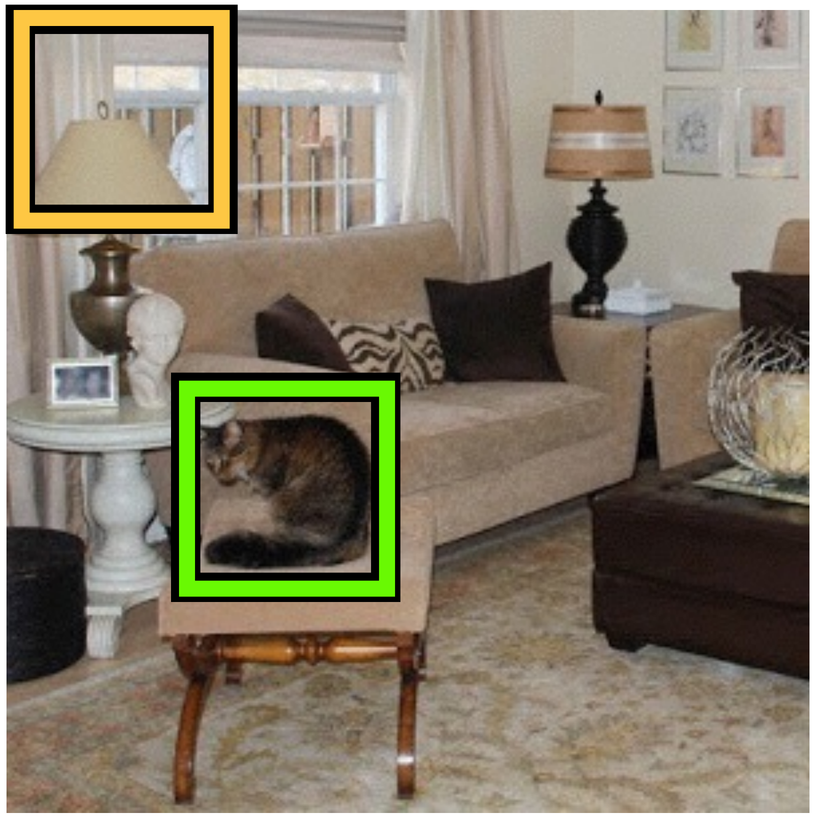
HOG(**x**) | HOG^{-1}(**x**) | **x**

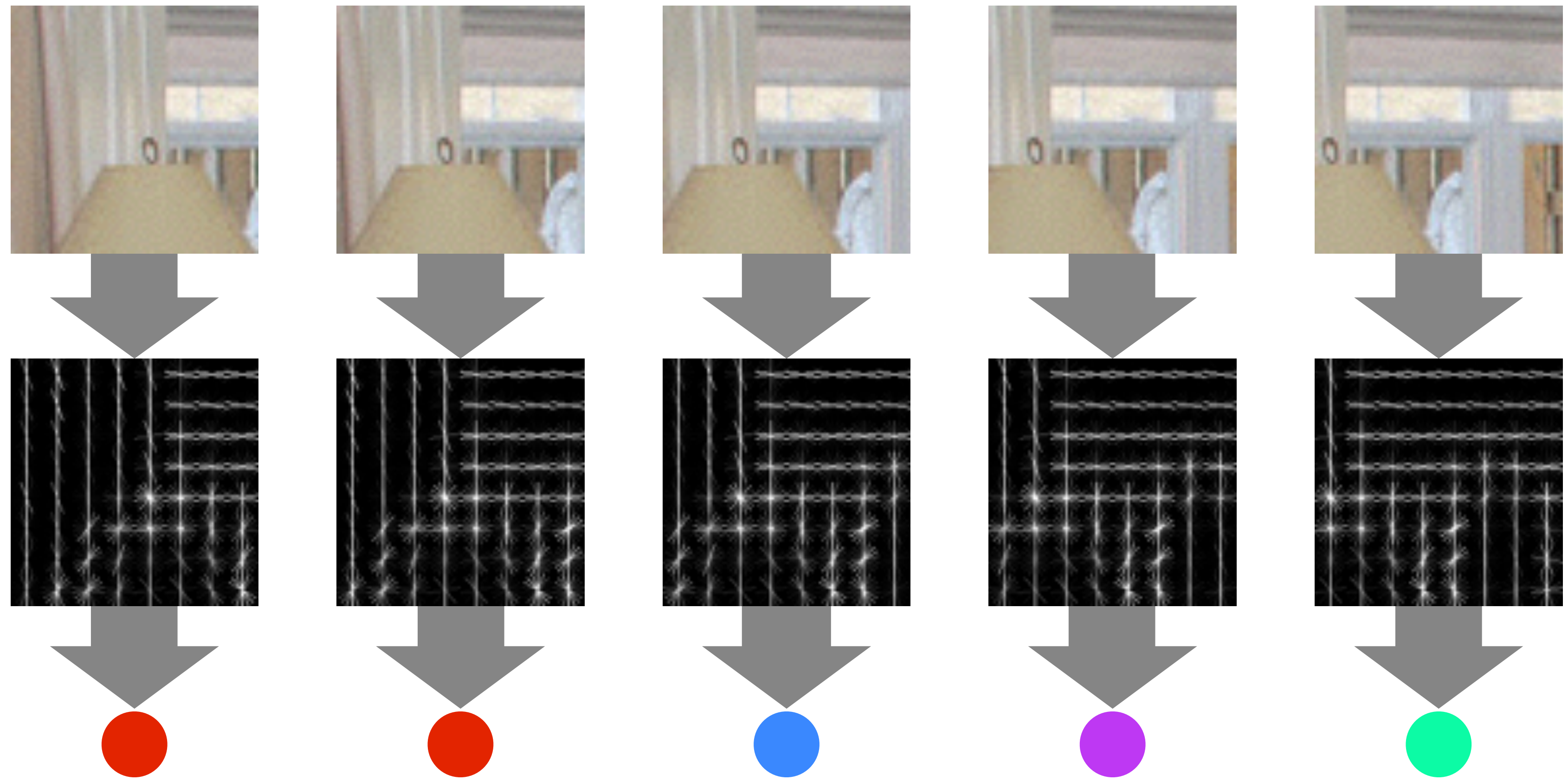


[Vondrick *et al.* 2013]

Bag of visual words

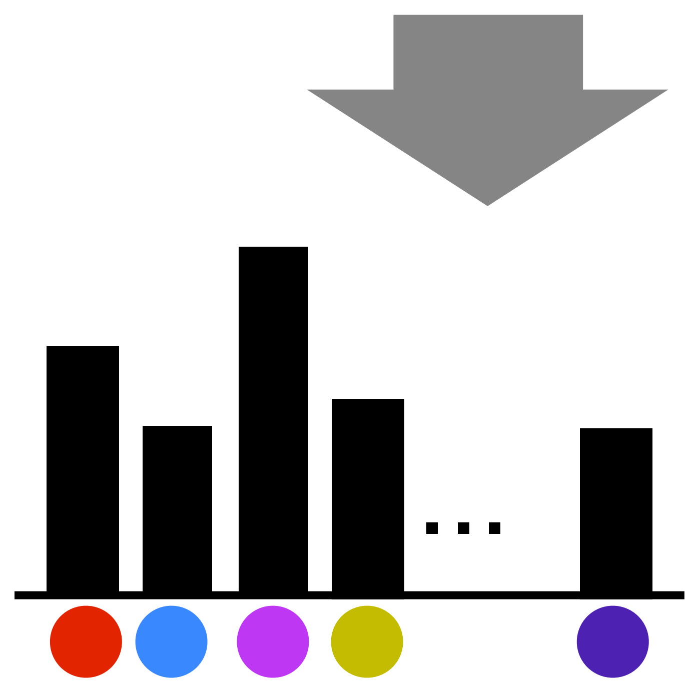
[Sivic & Zisserman 2003, Csurka *et al.* 2004, Nowak *et al.* 2006]



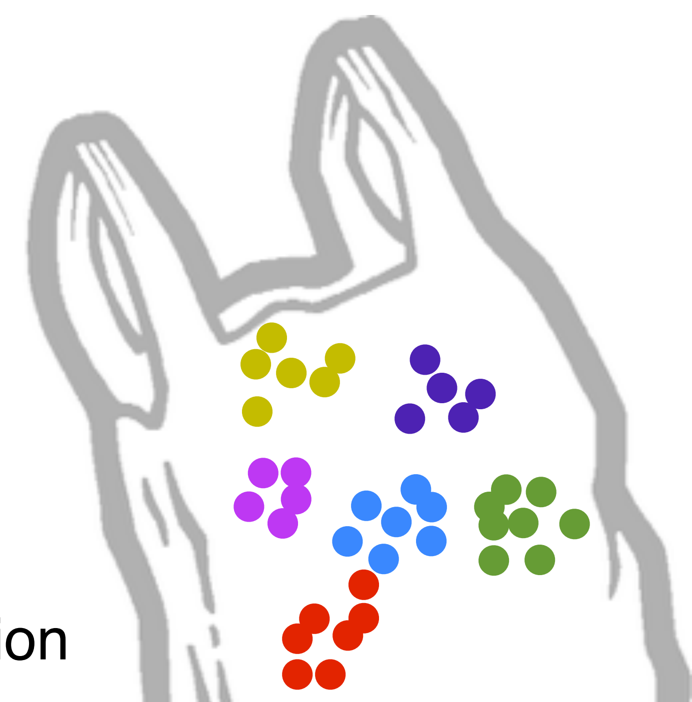


BoVW construction

1. Extract local descriptor densely
2. Quantise descriptors
3. Form histogram

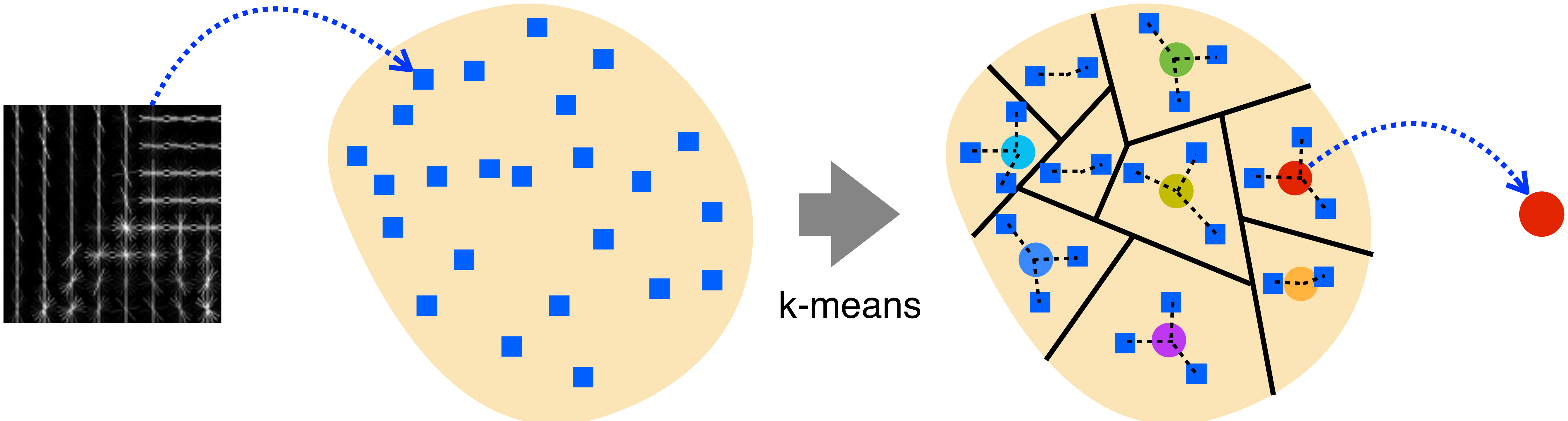


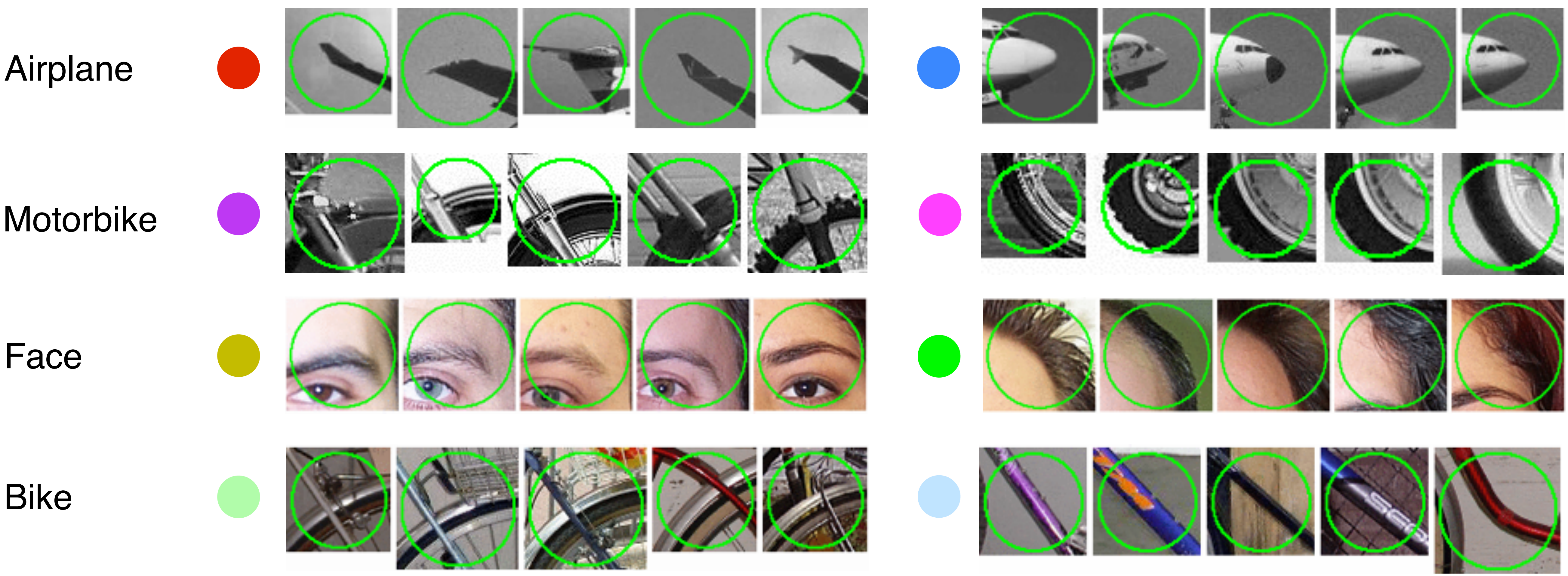
+ l^2 normalisation



2. **Discards spatial information**

Quantisation

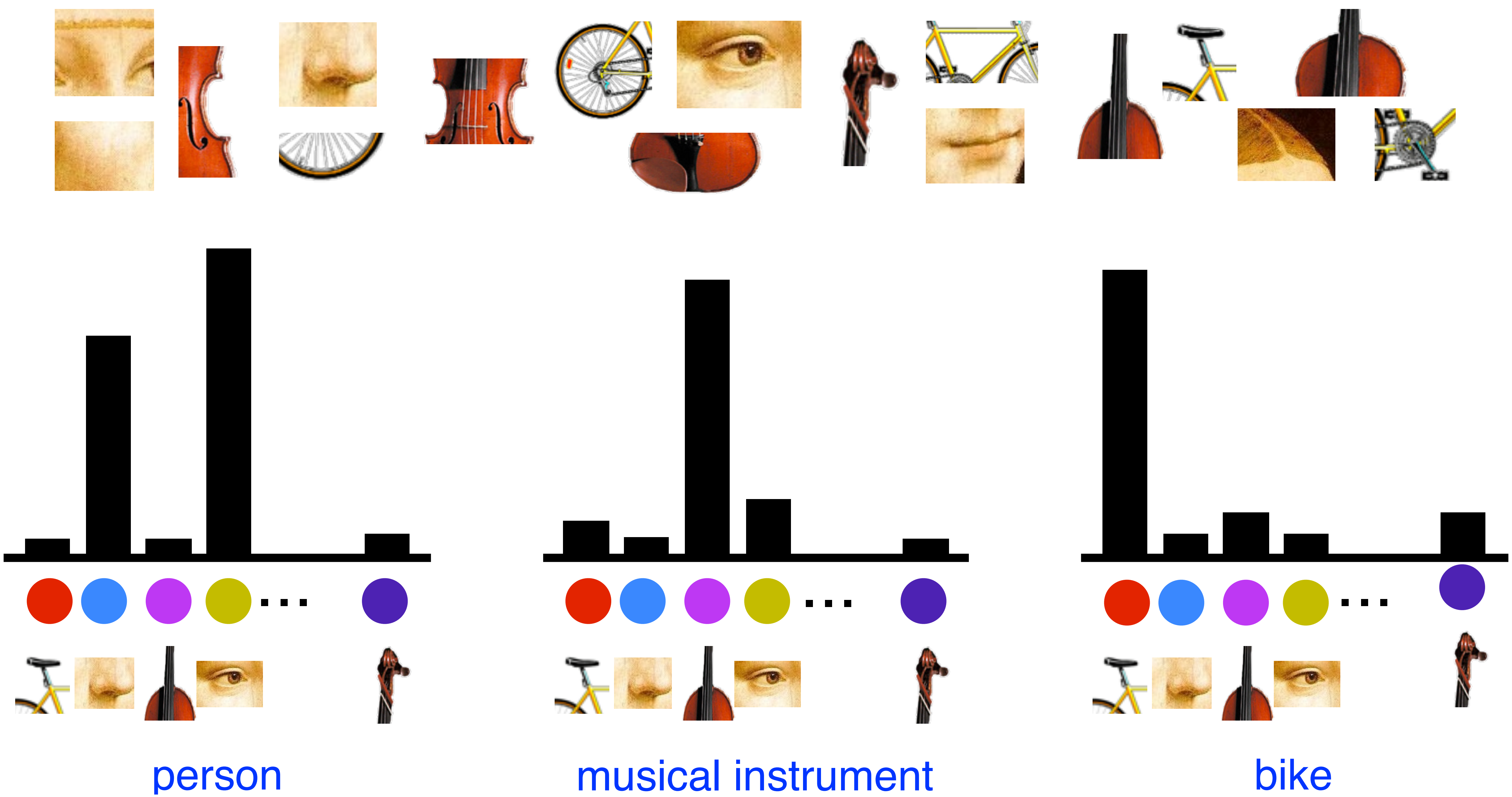




BoVW intuition

Discarding spatial information gives **lots of invariance**

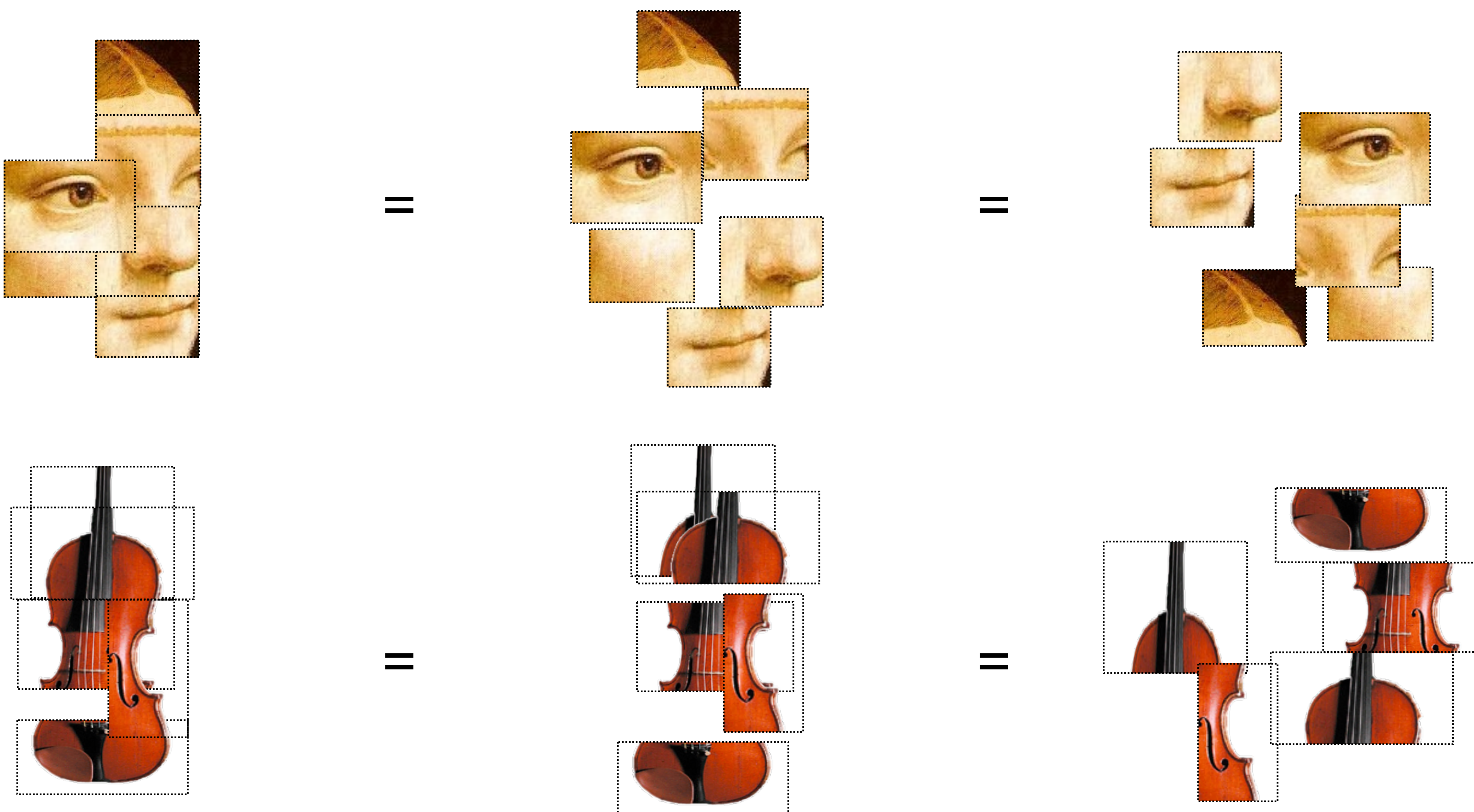
Visual words represent **“iconic” image fragments**



The loss of spatial information

Bag of features representation effectively forgets the relative location of the features

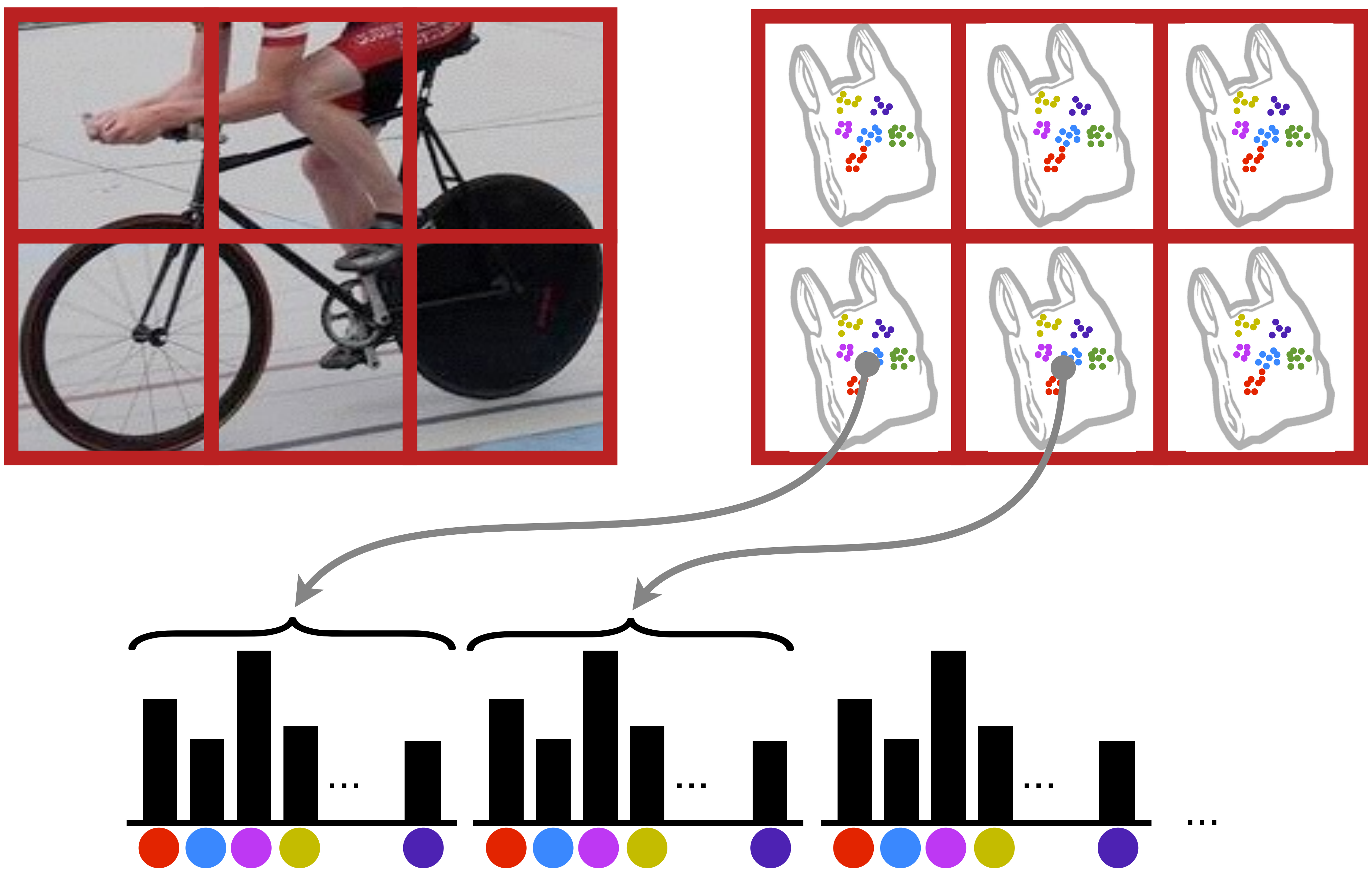
image plausible deformation implausible deformation



Spatial histograms

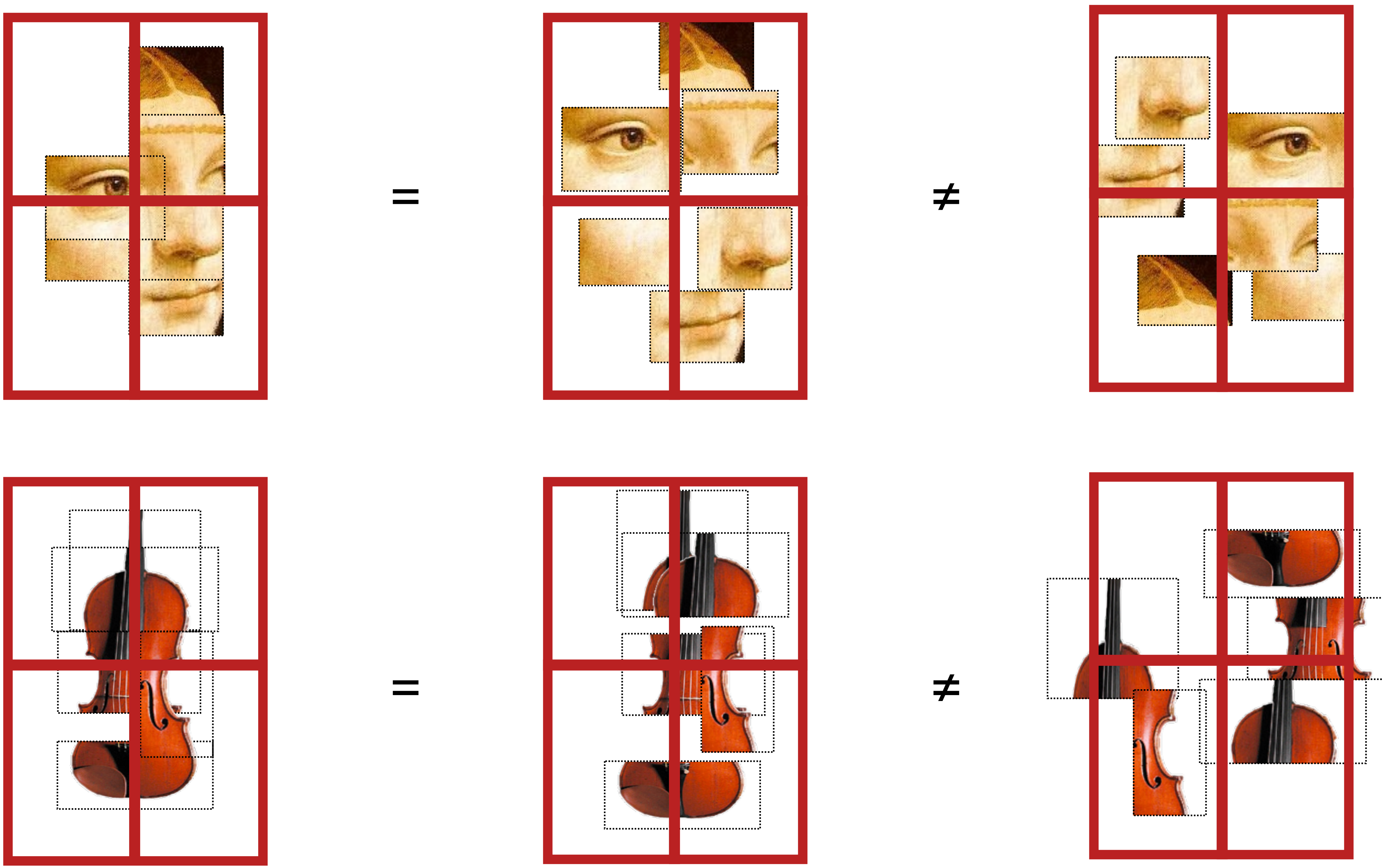
[Lazebnik *et al.* 2006]

Weak geometry: **pool spatial information locally**

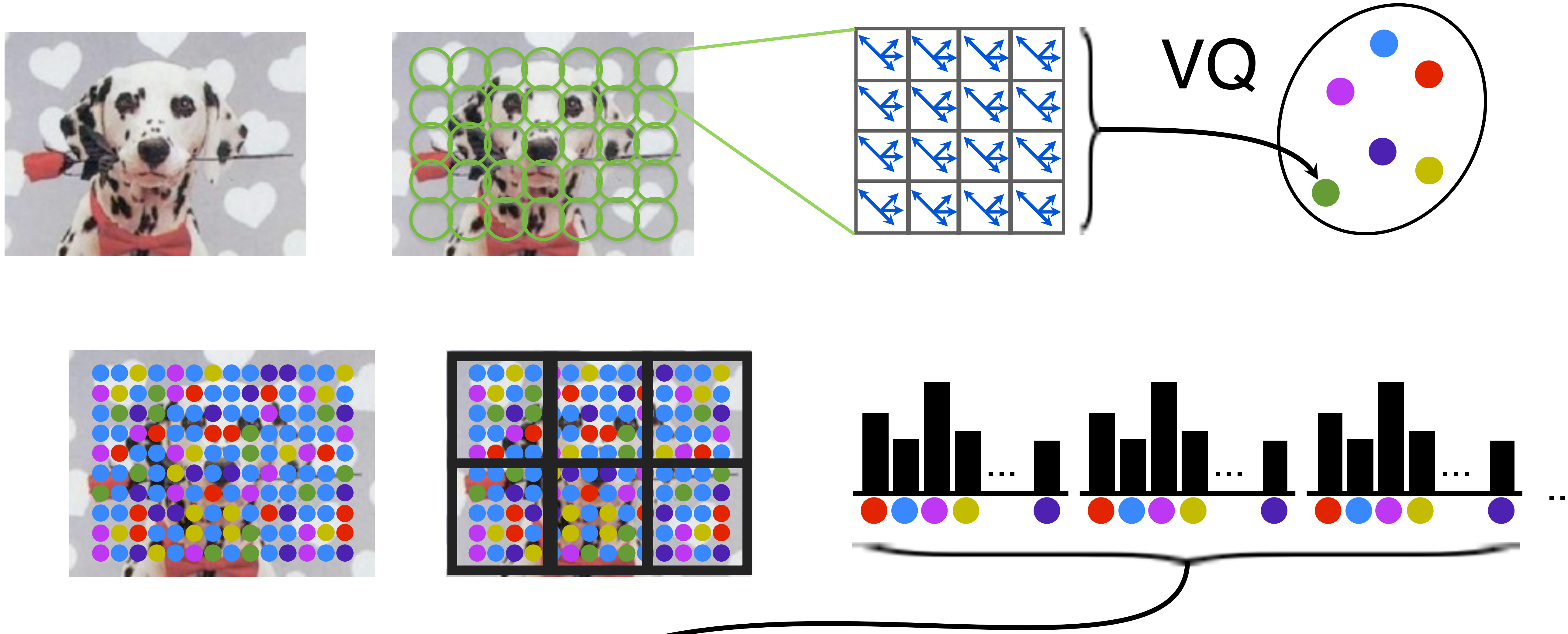


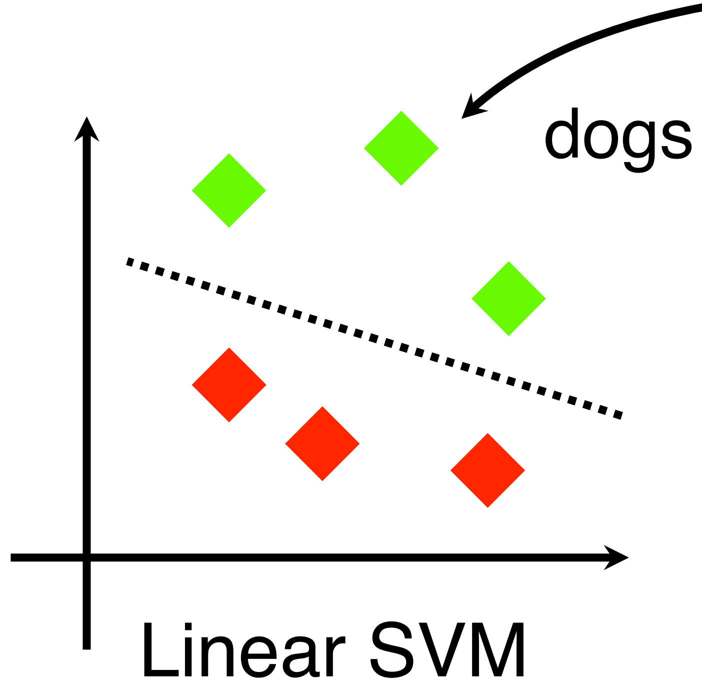
Spatial histograms capture weak geometry

image | plausible deformation | implausible deformation



Summary so far





[Luong & Malik, 1999]
[Varma & Zisserman, 2003]
[Csurka et al, 2004]
[Vogel & Schiele, 2004]
[Jurie & Triggs, 2005]
[Lazebnik et al, 2006]
[Bosch et al, 2006]

Advanced encodings

Soft and **sparse** assignments, e.g.

- [Philbin et al CVPR 08, Gemert et al ECCV 08]
- Locality-constrained linear coding (LLC) – [Wang et al CVPR 10]

Representing SIFT distribution **mean** in Voronoi cell, e.g.

- Super-Vector Coding [Zhou et al ECCV 10]
- VLAD [Jegou et al CVPR 10]

Representing SIFT distribution **mean** and **covariance** in Voronoi cell, e.g.

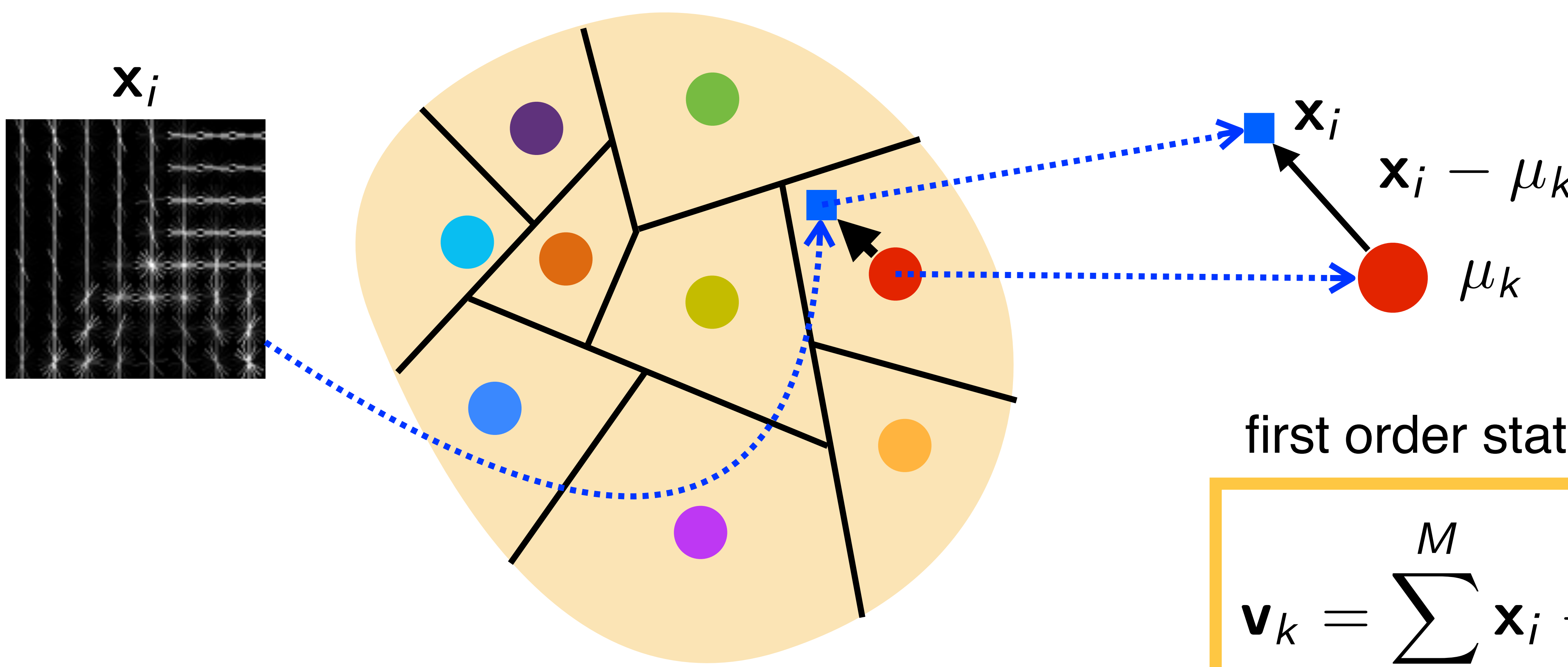
- Fisher vector [Perronnin et al CVPR 07 & 10, ECCV 10]

Improvements to **normalization**, PCA, **whitening** for VLAD/FV

- Chen et al 2011 [Jegou & Chum ECCV 12]
- All about VLAD [Arandjelovic & Zisserman CVPR 13]

Vector of locally aggregated descriptors (VLAD)

[Jegou *et al.* 2010]



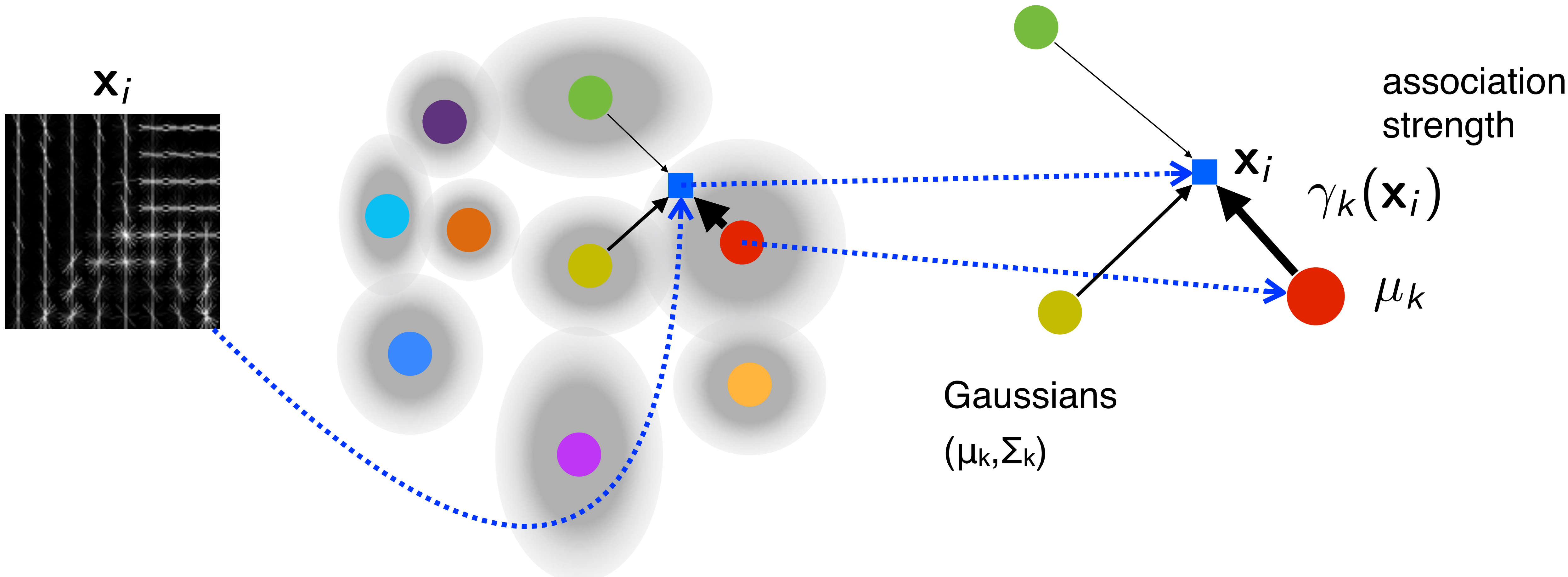
first order statistics

$$\mathbf{v}_k = \sum_{i=1}^{M} \mathbf{x}_i - \mu_k$$

VLAD encoding $\Phi = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_K \end{bmatrix}$ + l^2 normalisation

Fisher Vector (FV)

[Perronnin et al. ECCV 201, Sharma Hussain Jurie ECCV 2010, Sanchez et al. 2103]



first and second order statistics

FV encoding $\Phi = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{u}_1 \\ \mathbf{v}_2 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{v}_K \\ \mathbf{u}_K \end{bmatrix}$

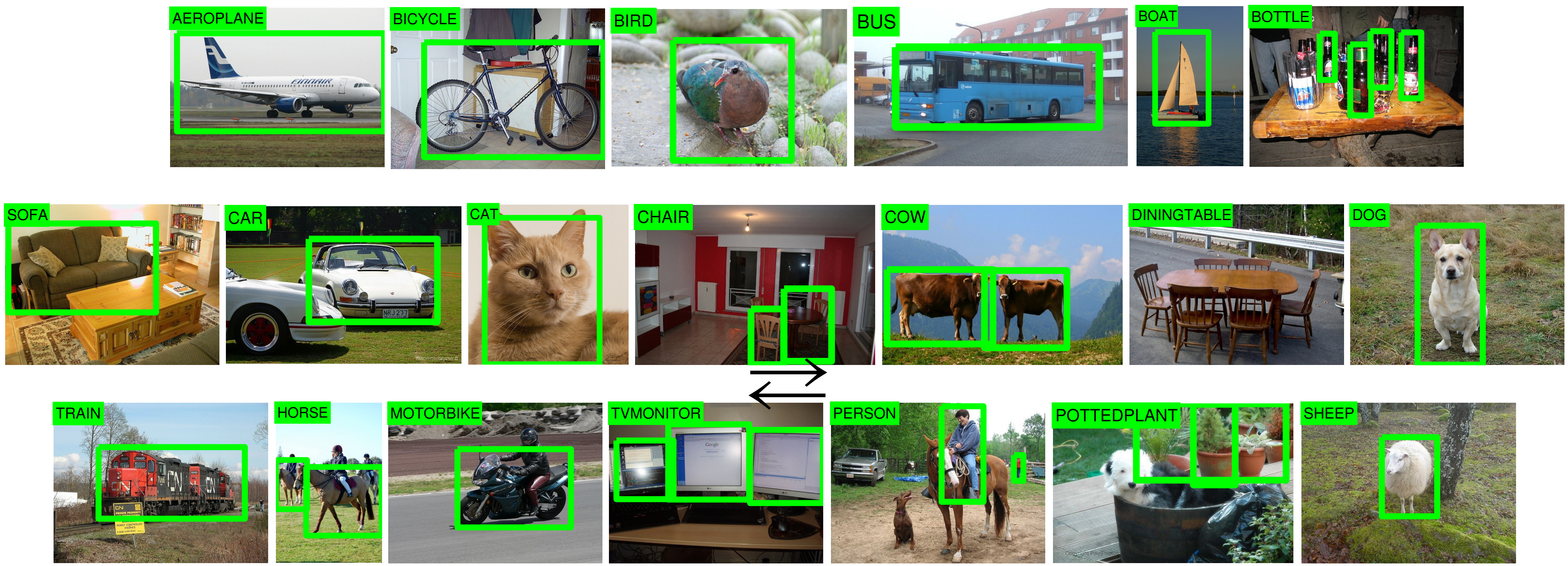
+ sqrt-l^2 normalisation

$$\mathbf{v}_k = \frac{1}{M\sqrt{\pi_k}} \sum_{i=1}^{M} \gamma_k(\mathbf{x}_i) \frac{\mathbf{x}_i - \mu_k}{\sigma_i}$$

$$\mathbf{u}_k = \frac{1}{M\sqrt{2\pi_k}} \sum_{i=1}^{M} \gamma_k(\mathbf{x}_i) \left(\frac{\mathbf{x}_i - \mu_k}{\sigma_i} - 1 \right)^2$$

Reference benchmark: PASCAL VOC

Task: decide if an image contains any of twenty object classes



Performance

mean Average Precision (mAP)

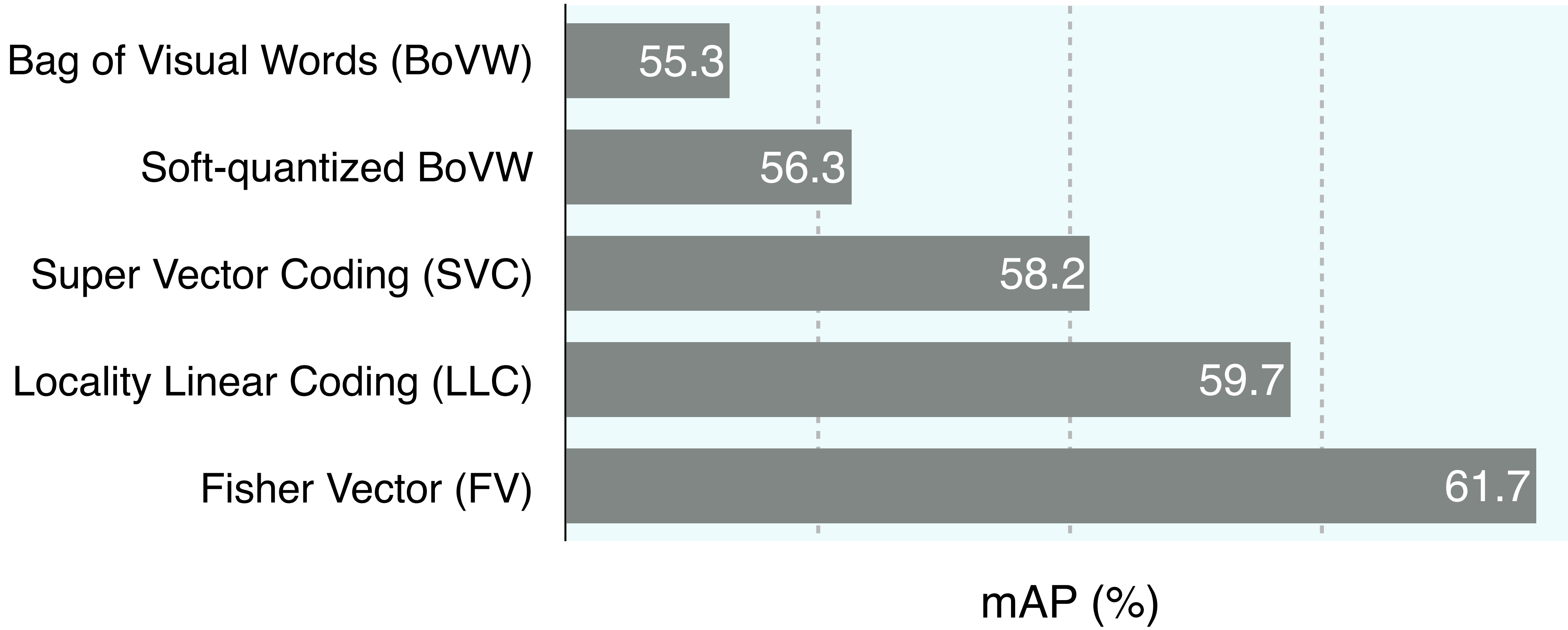
mAP = 50% $\rightleftarrows$ 50% of object occurrences are recognised reliably

roughly

[Everingham et al, 2006-12]

The devil is in the details

A comparison of encodings [Chatfield *et. al.* BMVC 2011]



2005—12: an industrial production of encodings
[Sivic et al. 03, Csurka et al. 04, Zhou et al. 10, Perronnin et al. 08, Jegou et al. 10, …]

Our evaluation compared representative ones on an equal footing

The (Improved) Fisher Vectors came out on top
[see **Tuesday's talk for a comparison with deep learning**]

Some fundamental ideas

Local and translation invariant operators

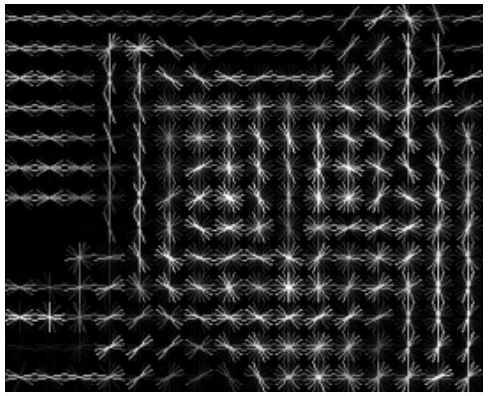
gradients, filters, visual words

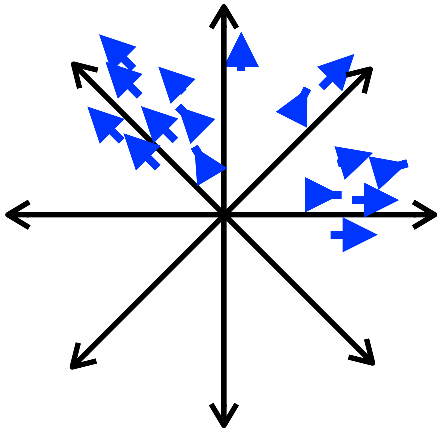
Untangling

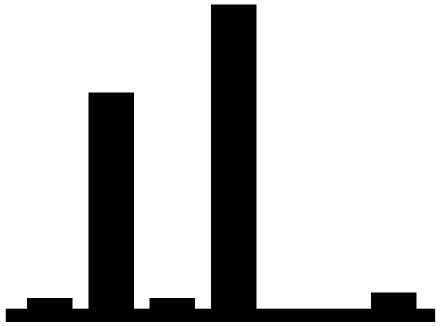
sparsity, quantisation

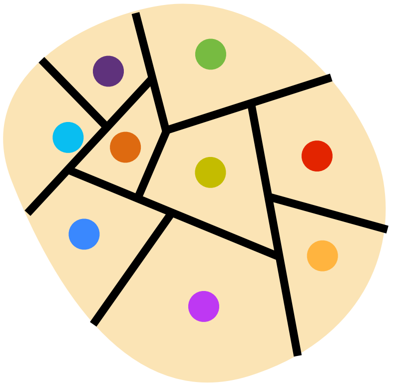
Pooling

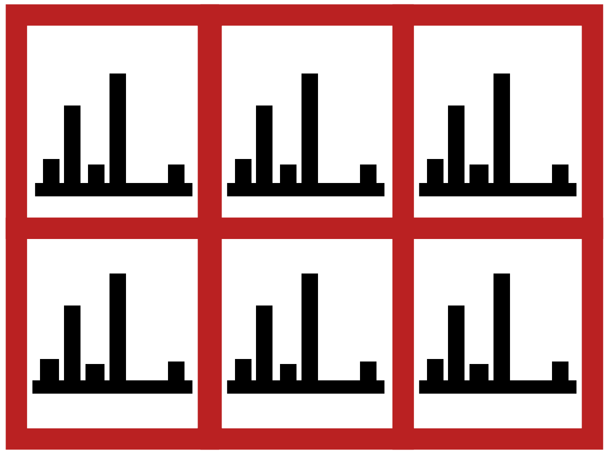
max, sum, spatial pooling

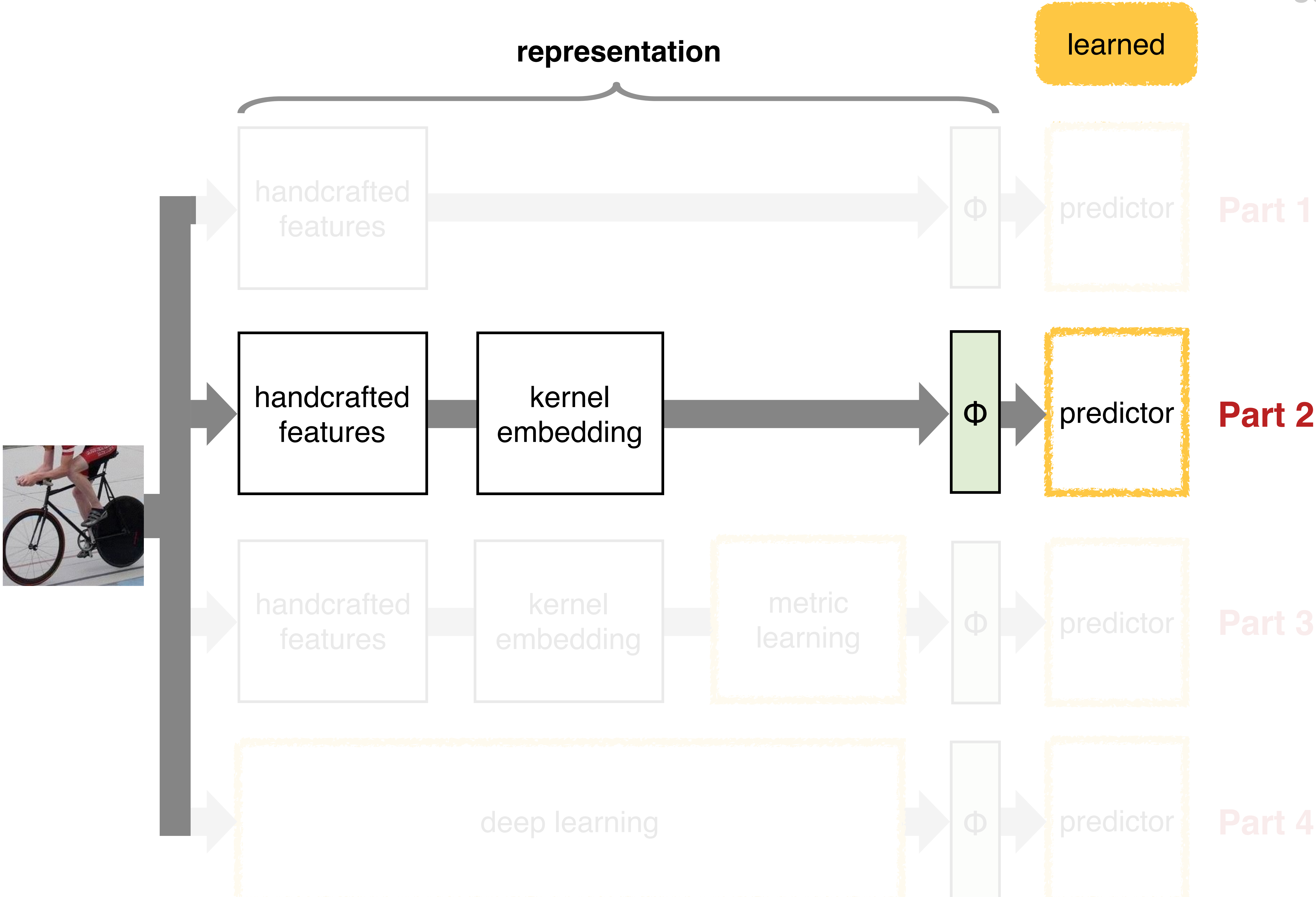










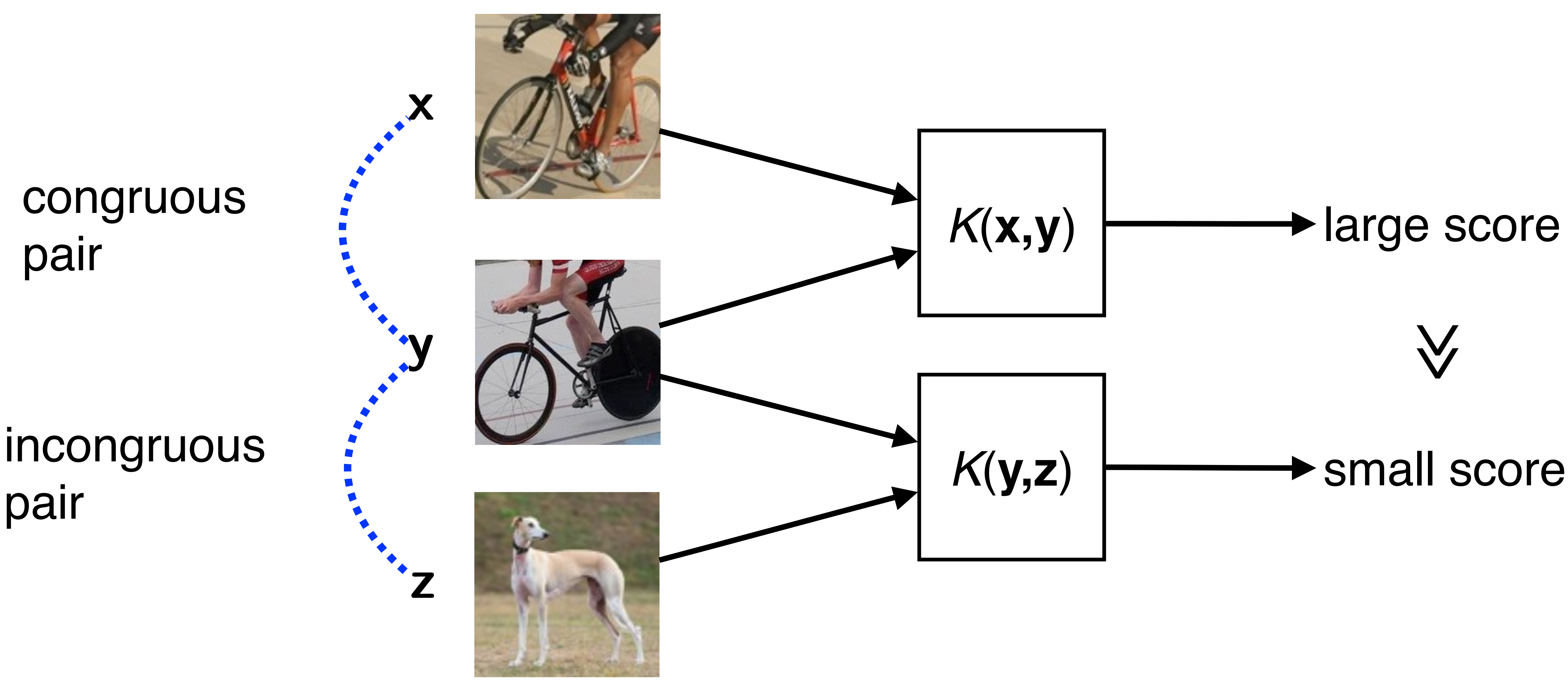


representation
learned
handcrafted features
Φ
predictor
Part 1
handcrafted features
kernel embedding
Φ
predictor
Part 2
handcrafted features
kernel embedding
metric learning
Φ
predictor
Part 3
deep learning
Φ
predictor
Part 4

Kernels

A **kernel** *directly* encodes a notion of *data similarity*

$$\boxed{K : (\mathbf{x}, \mathbf{y}) \mapsto \mathbb{R}}$$



Similarity and kernels

Recall: the encoder ϕ(I) should embody a useful notion of similarity

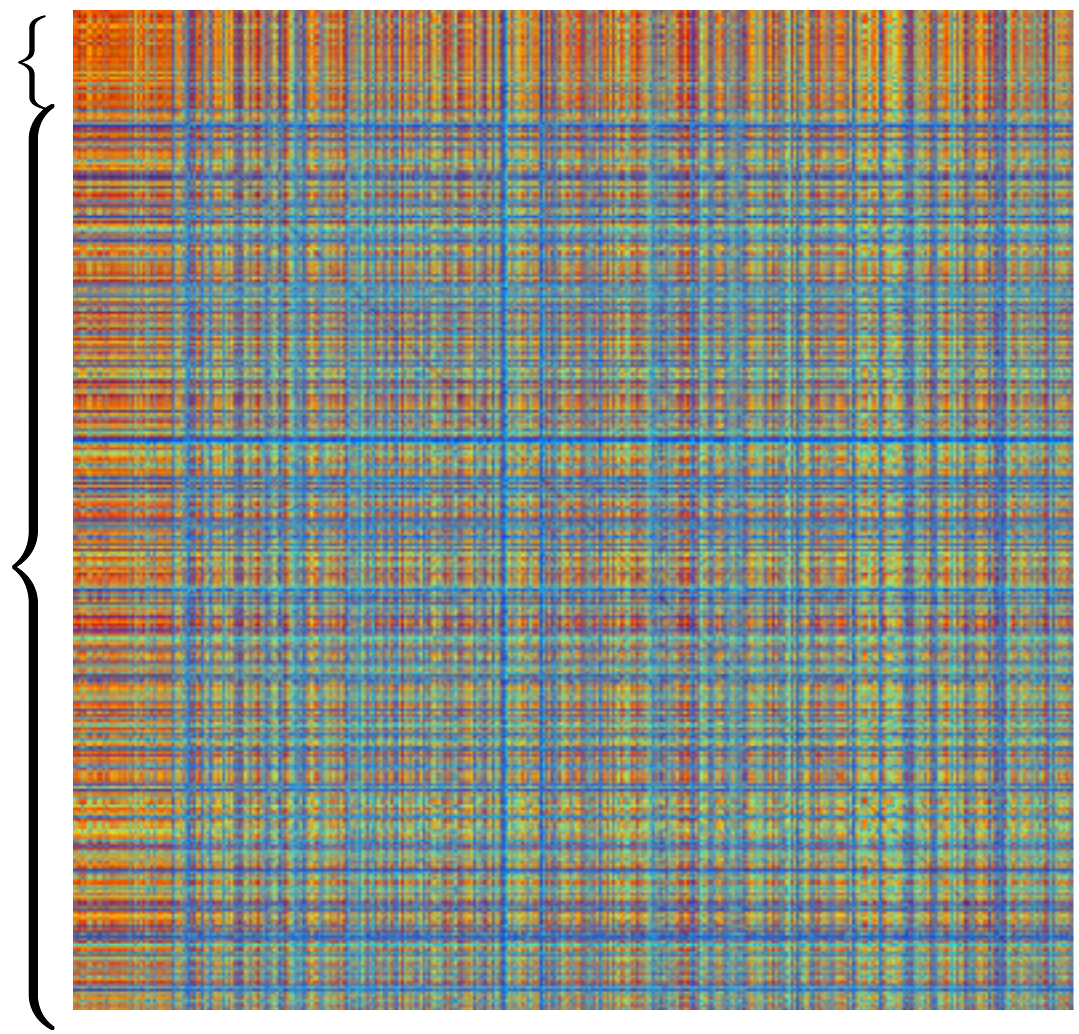
Similarity can be measured by the inner product or kernel $\langle \phi(I), \phi(I') \rangle$

linear kernel

$$\langle \Phi(I), \Phi(I') \rangle$$

bike images

other images



Normalisation

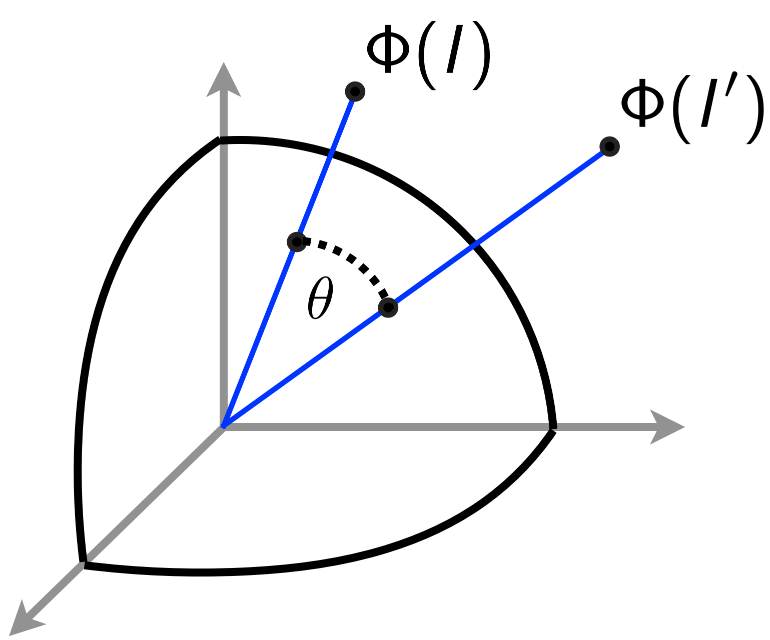
Extracting the representation ϕ(I) induces a notion of "similarity" between images

$$\textbf{kernel} \quad K(I, I') = \langle \Phi(I), \Phi(I') \rangle$$

A natural property: any object is most similar to itself

$$\forall I, I' : \quad K(I, I') \leq K(I, I)$$

This property is satisfied *provided that features are L2* ***normalised***



$$\Phi(I) \leftarrow \frac{1}{\|\Phi(I)\|}\Phi(I)$$

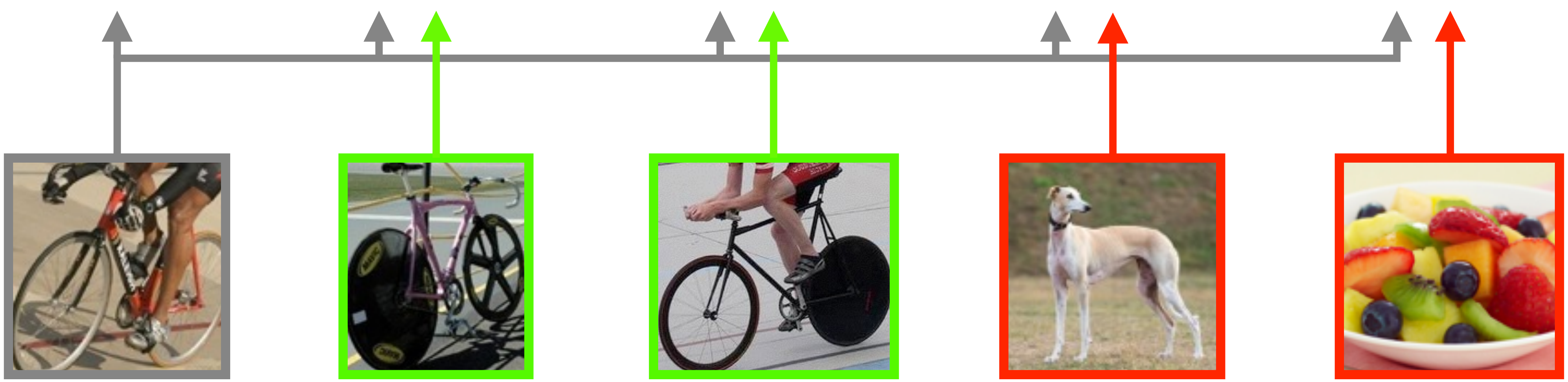
Kernel predictor

Task: predict the class of a datum **x**

How: use *K* to compare **x** to all training examples $\mathbf{x}_1$, $\mathbf{x}_2$, …

$$\boxed{F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)}$$

$$F(\mathbf{x}) = \alpha_1 K(\mathbf{x}, \mathbf{x}_1) + \alpha_2 K(\mathbf{x}, \mathbf{x}_2) + \alpha_3 K(\mathbf{x}, \mathbf{x}_3) + \alpha_4 K(\mathbf{x}, \mathbf{x}_4) + \ldots$$

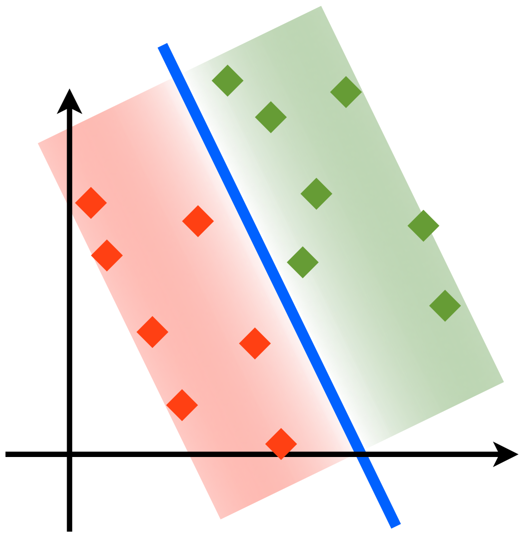


Non-linear kernels

Linear SVM

✔ fast

✘ restrictive

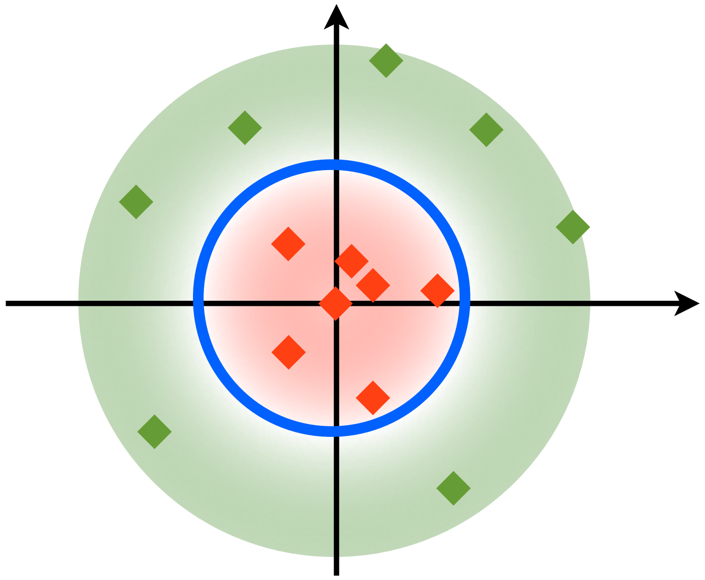


$$F(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$$

Non-linear SVM

✘ much slower

✔ powerful



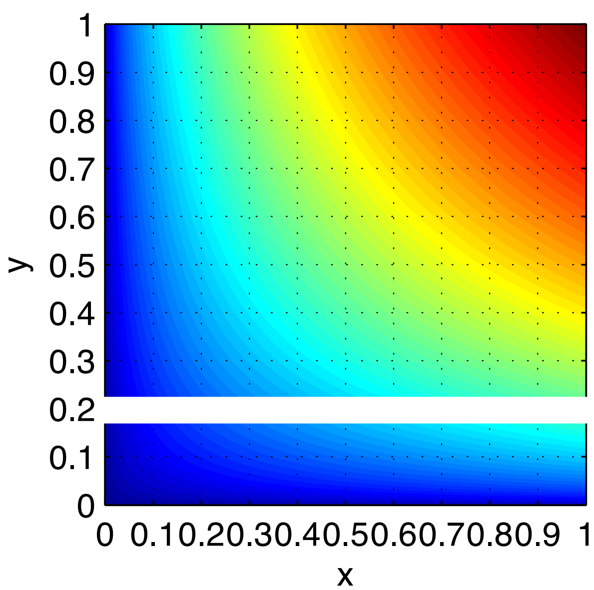
$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

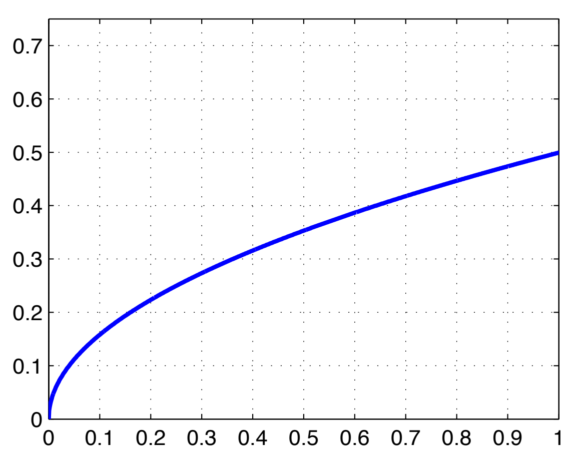
Additive homogeneous kernels

$$\boxed{K(\mathbf{x},\mathbf{y}) = \sum_{l=1}^{d} k(x_l, y_l)}$$

Hellinger

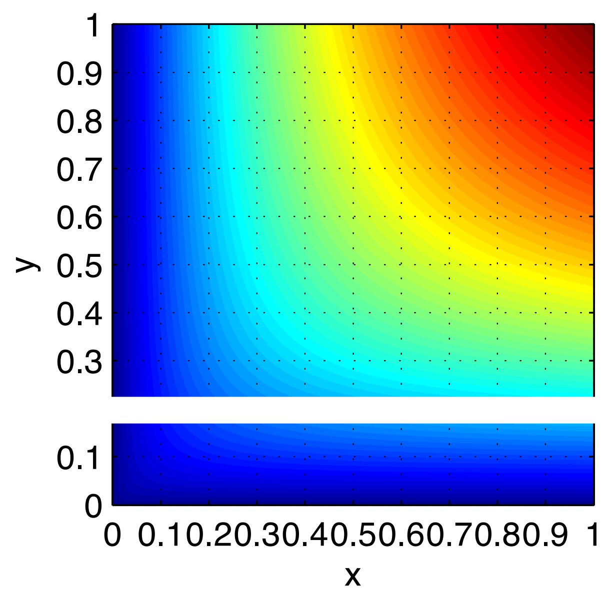
$$\sqrt{xy}$$

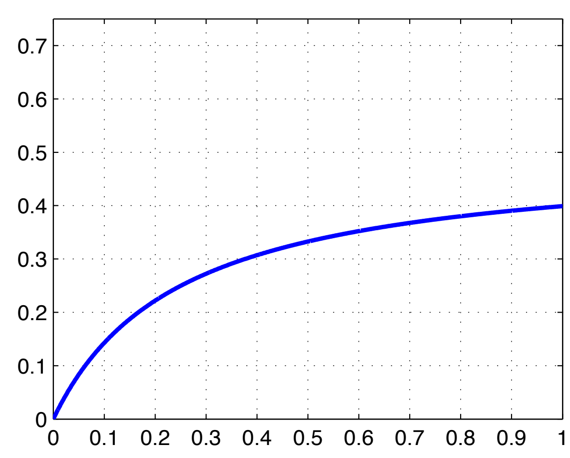




χ^2

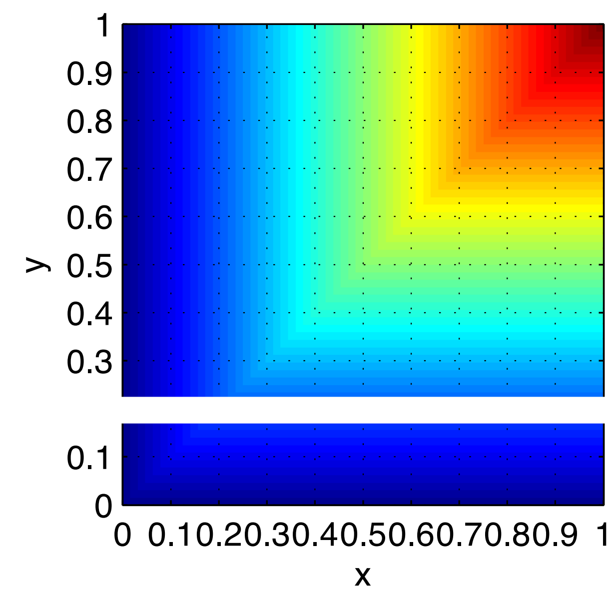
$$\frac{2xy}{x+y}$$

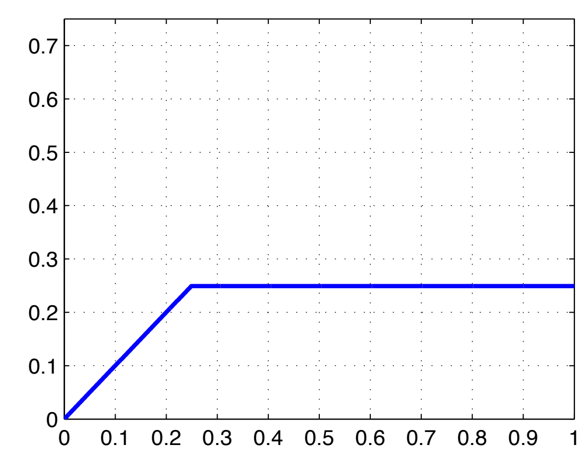




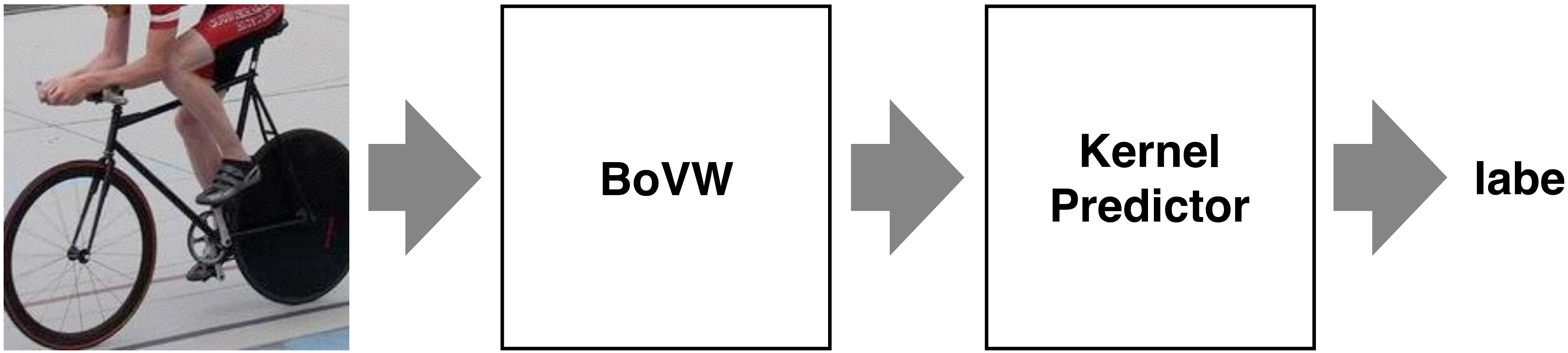
intersection

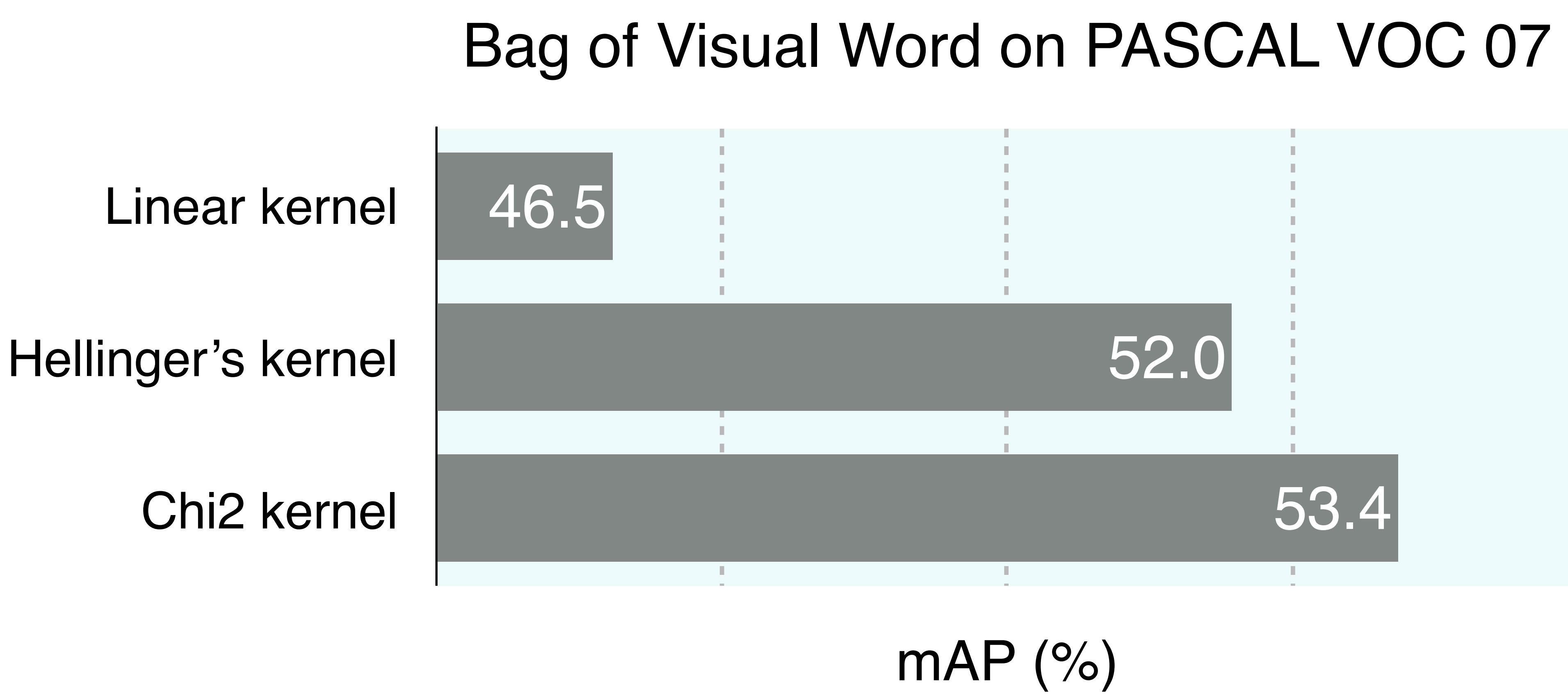
$$\min\{x, y\}$$





Additive kernels example





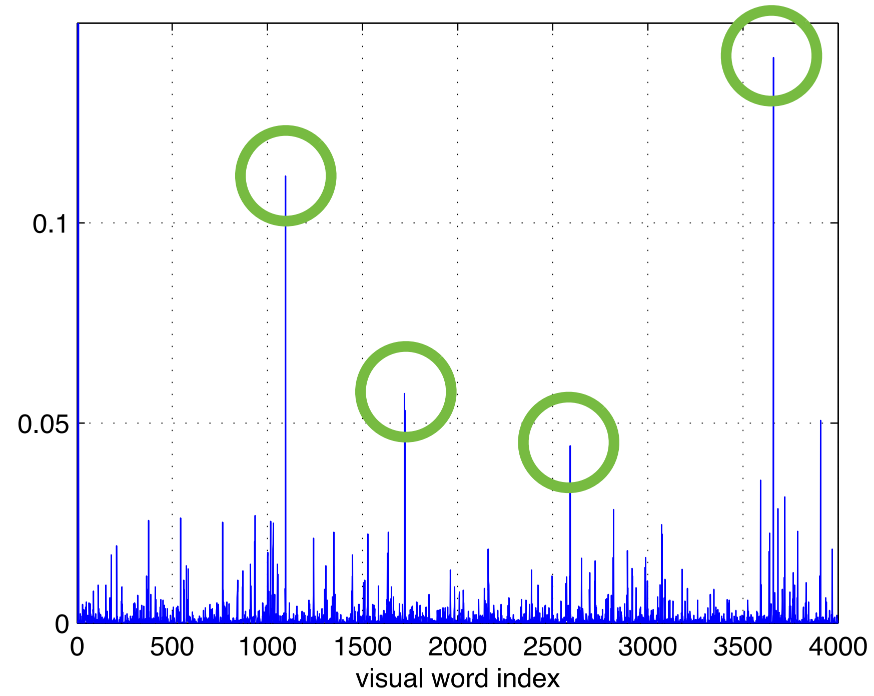
Burstiness and the square-root map

Burstiness

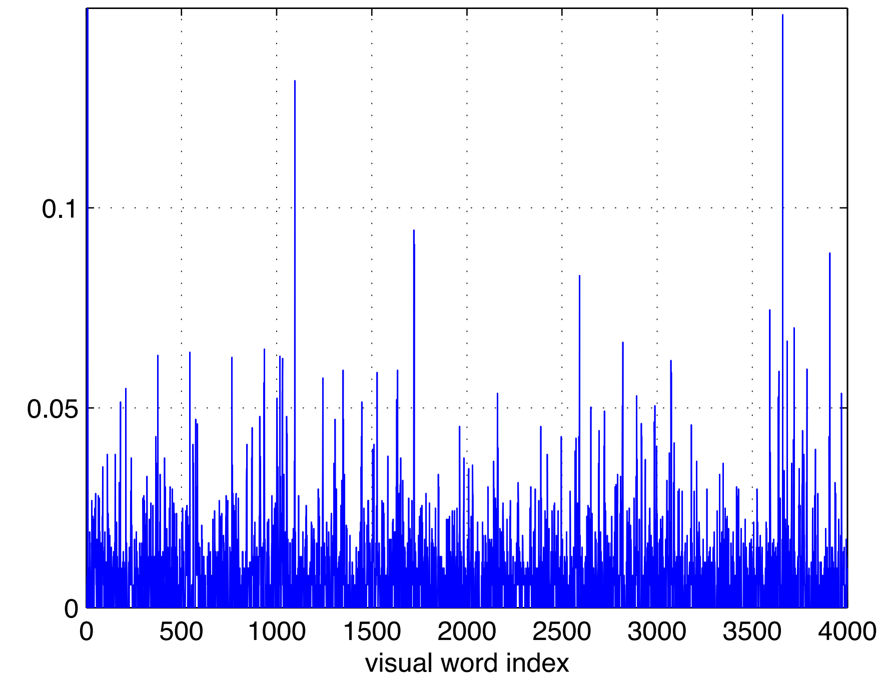
- words may occur in **bursts** [Jegou et al. 2009]
- compensate by taking the **square root**

dominated by “grass” words

$\Phi(I)$



$\Phi(I) \leftarrow \sqrt{\Phi(I)}$



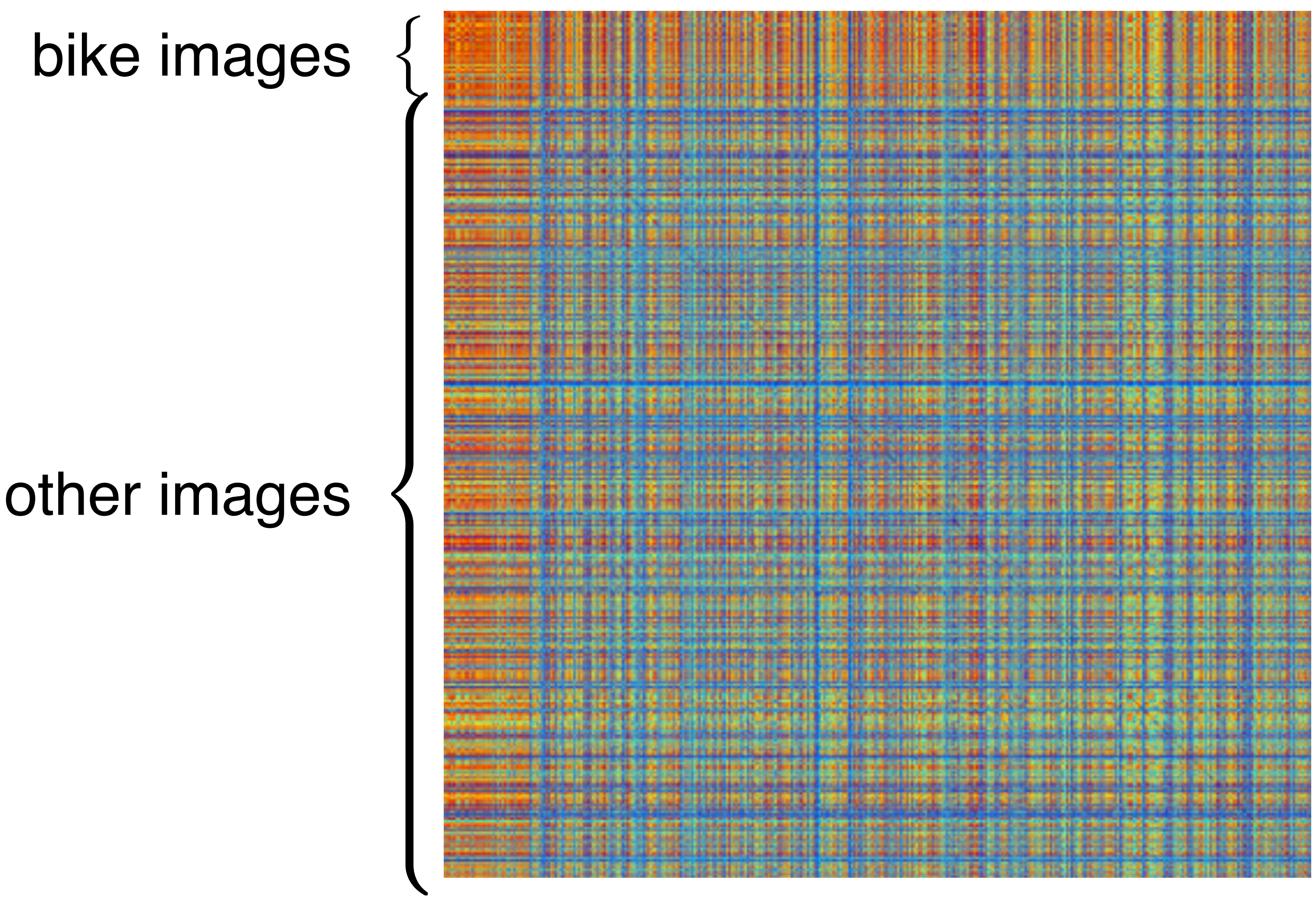
Effect of square rooting

Extracting the representation ϕ(I) induces a notion of "similarity" between images

$$\textbf{kernel}\ \ K(I, I') = \langle \Phi(I), \Phi(I') \rangle$$

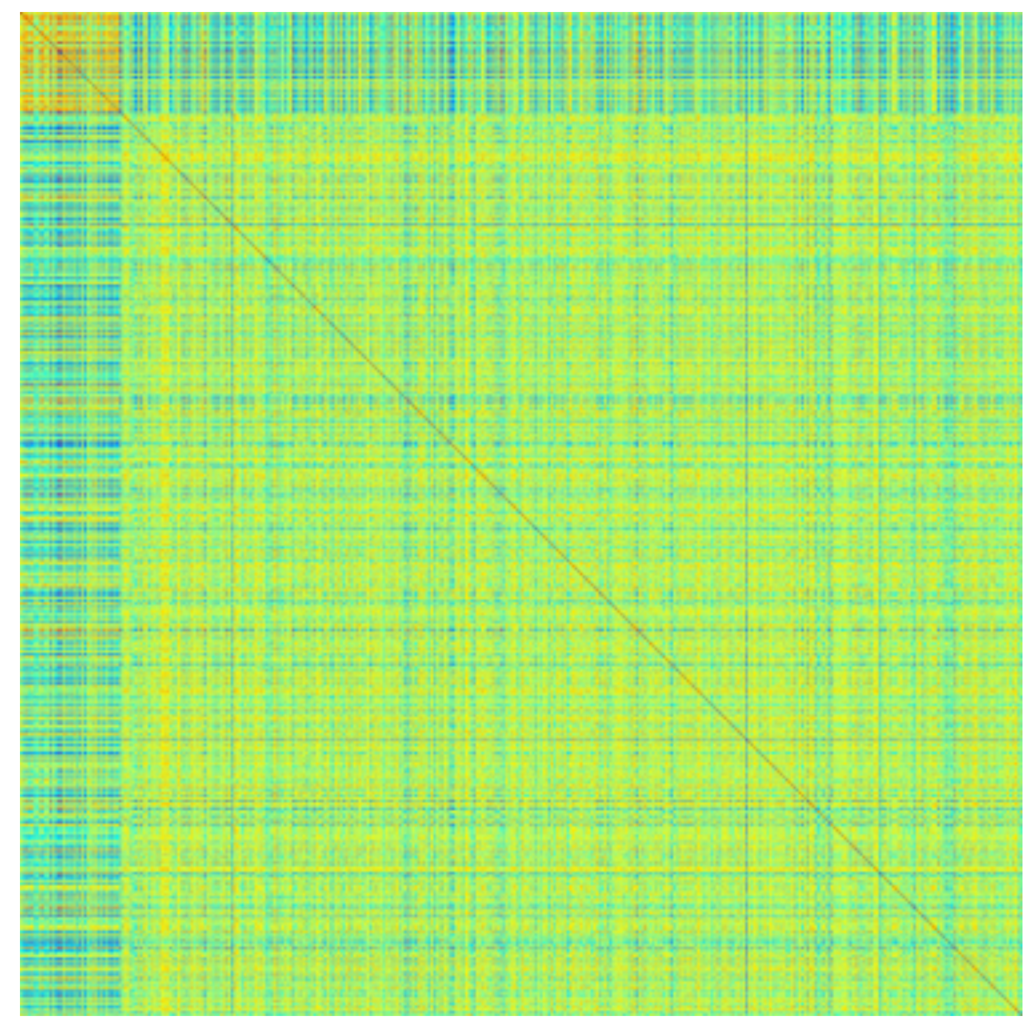
linear kernel

$$\langle \Phi(I), \Phi(I') \rangle$$



non-linear kernel

$$\langle \sqrt{\Phi(I)}, \sqrt{\Phi(I')} \rangle$$



Non-linear kernels are expensive

$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

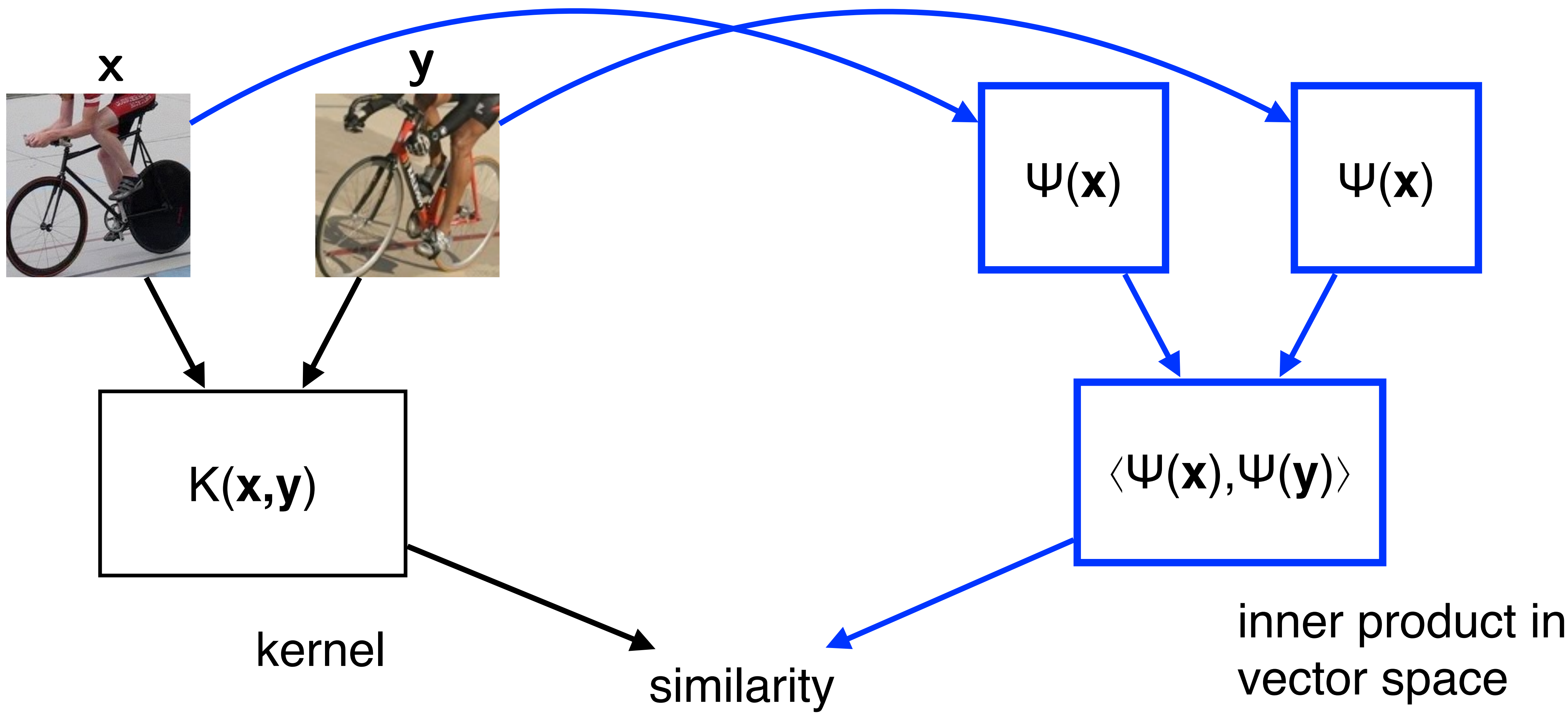
thousand bicycles

many more non-bicycle

Kernel maps

Positive definite kernel = inner product of **feature vectors**

$$\mathbf{x} \xrightarrow{\text{feature map } \Psi} \underset{\text{representation}}{\Psi(\mathbf{x}) \in V}$$



Analytical explicit maps

Empirical maps

- Numerical
- **Good**: general, adaptive
- **Bad**: slow, dataset specific

Analytical maps

- Closed-form
- **Good**: fast, dataset agnostic
- **Bad**: kernel-specific, non-adaptive

linear	$K(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y} \rangle$	$\Phi(\mathbf{x}) = \mathbf{x}$
Hellinger's	$K(x,y) = \sqrt{xy}$	$\Phi(x) = \sqrt{x}$

A few kernels have trivial maps

Which other kernels have analytical maps?

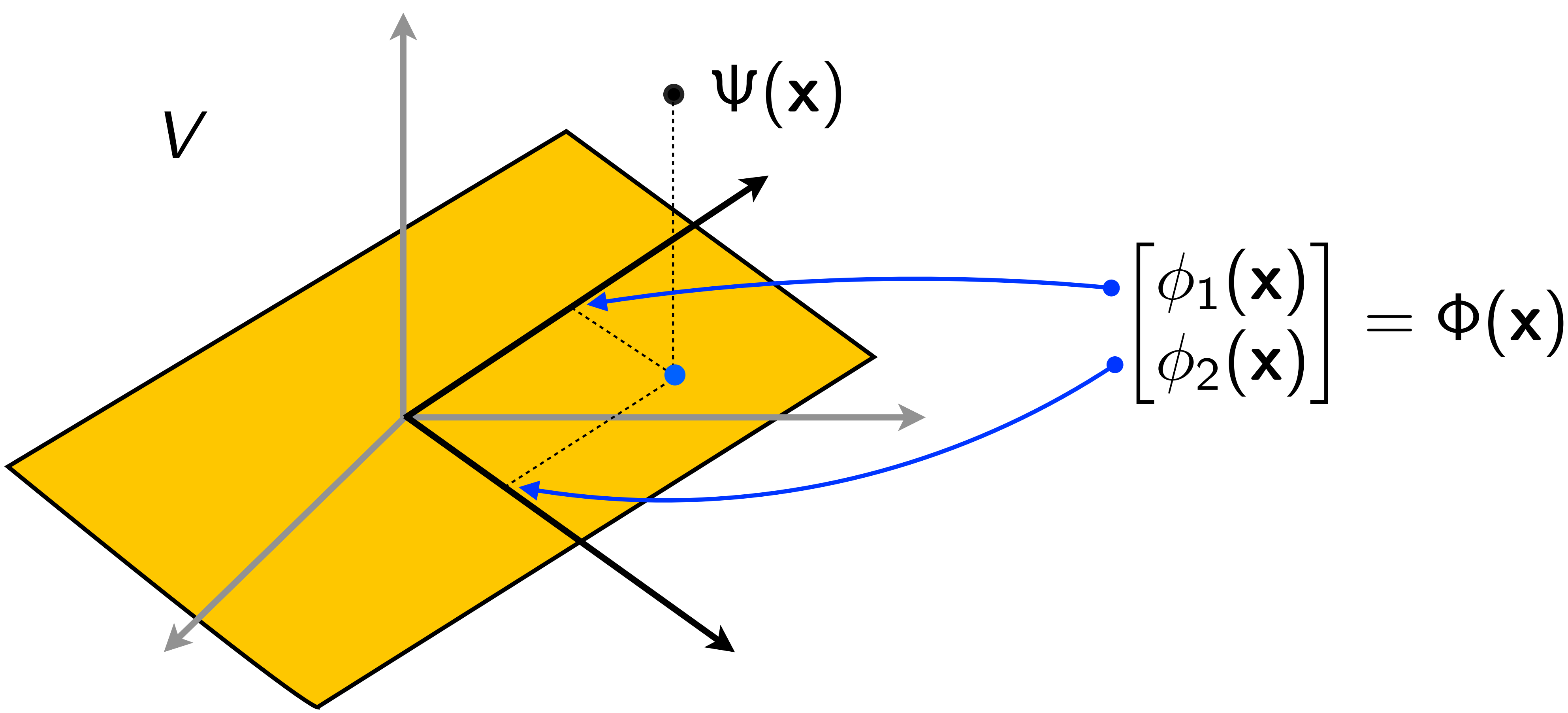
Explicit kernel maps

- **Kernel maps**
  - often infinite dimensional
  - used implicitly (kernel trick)
  - theoretical

$$K(\mathbf{x}, \mathbf{y}) = \langle \Psi(\mathbf{x}), \Psi(\mathbf{y}) \rangle$$
$$\Psi(\mathbf{x}) \in V$$

- **Explicit kernel maps**
  - finite dimensional approximation
  - used explicitly
  - practical

$$K(\mathbf{x}, \mathbf{y}) \approx \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$$
$$\Phi(\mathbf{x}) \in \mathbb{R}^d$$



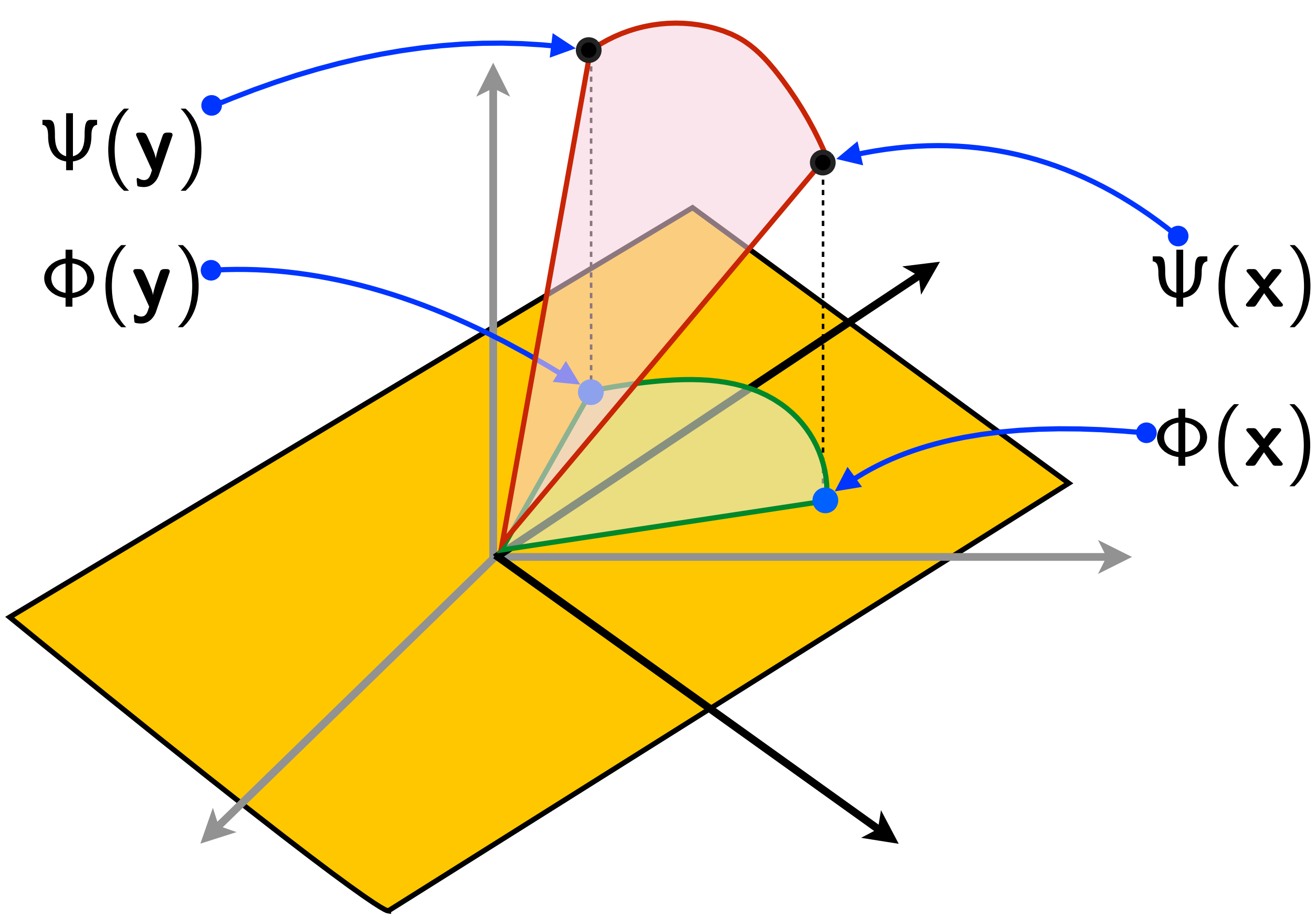
Explicit kernel maps

- **Kernel maps**
  - often infinite dimensional
  - used implicitly (kernel trick)
  - theoretical

$$K(\mathbf{x}, \mathbf{y}) = \langle \Psi(\mathbf{x}), \Psi(\mathbf{y}) \rangle$$
$$\Psi(\mathbf{x}) \in V$$

- **Explicit kernel maps**
  - finite dimensional <u>approximation</u>
  - used explicitly
  - practical

$$K(\mathbf{x}, \mathbf{y}) \approx \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$$
$$\Phi(\mathbf{x}) \in \mathbb{R}^d$$



Explicit maps are efficient

Much faster **evaluation**

$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

O(N)

explicit map

$$F(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle$$

O(1)

Much faster **learning**

Non-linear SVM
`LibSVM`

$O(N^2)$

explicit map

Linear SVM solver
`LibLinear`

O(N)

Example: Chi^2 map

MATLAB code for Chi2 kernel

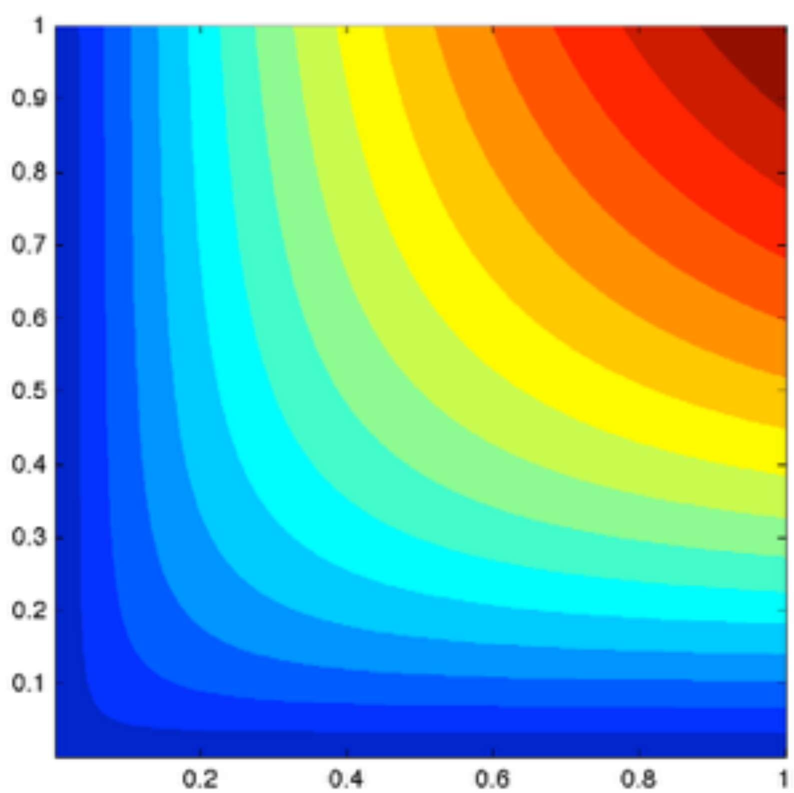
```
x = .01:.01:1 ;
for i = 1:100
  for j = 1:100
    K(i,j) = ...
      2*x(i)*x(j)/(x(i)+x(j));
  end
end
```

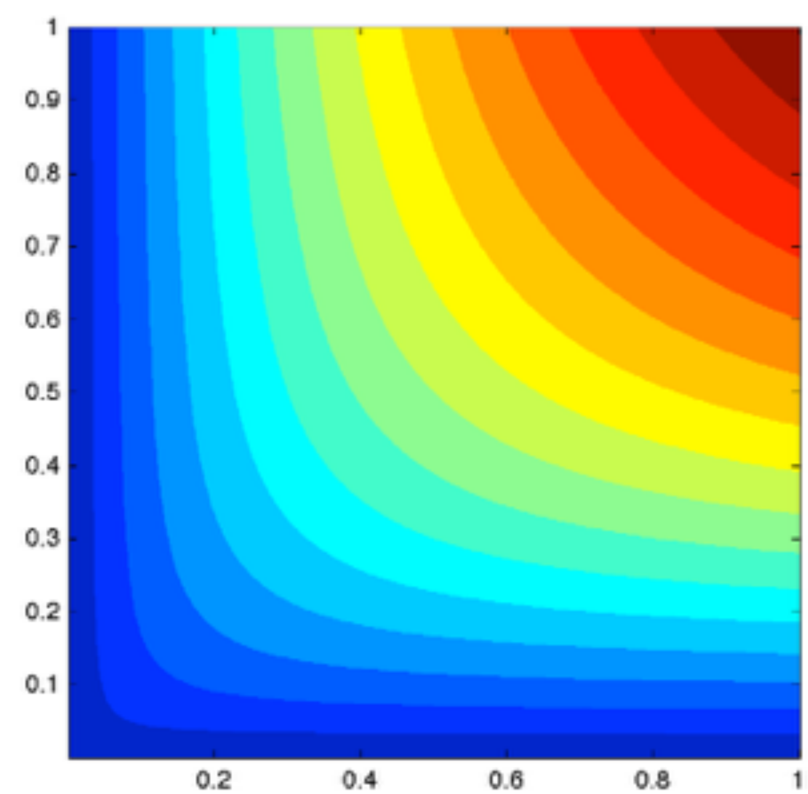
With the hom. kernel feature map

```
x = .01:.01:1 ;
psi = vl_homkermap(x,1) ;
K = psi'*psi ;
```

VLFeat Toolbox

http://www.vlfeat.org





Example: Chi² map

training time

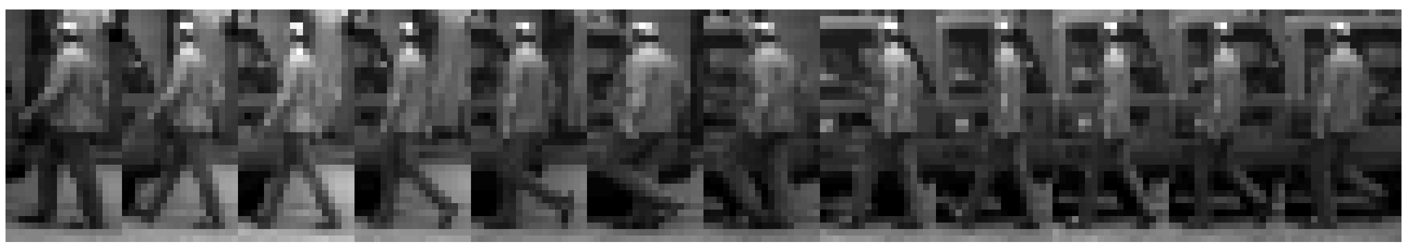
Caltech-101 category recognition

#1,500

1 h → 5 m

4× speedup

DaimlerChrylser pedestrian recognition



#20,000

1/2 h → 14 s

100× speedup

Trecvid 2009 video indexing

#70,000

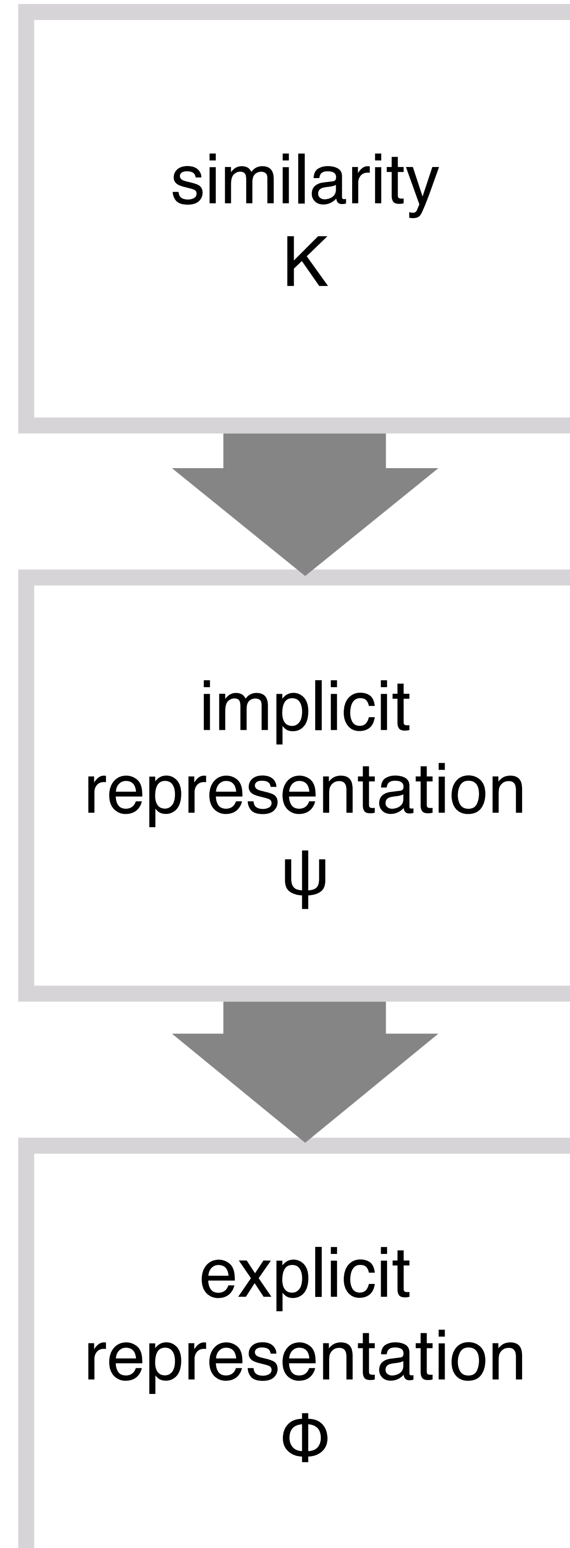
> 1 h → 22.6 s

160× speedup

Kernel as representations

From similarity to features

1. Start from a **concept of similarity**
   - homogeneous kernels = good for histograms
   - Gaussian kernels = local similarity
   - …
2. Capture it in a **positive definite function**
   - infinite dimensional feature map
   - implicit data representation
3. Find **finite dimensional approximations**
   - explicit data representation



Kernel methods are very mature

Theory

- reproducing kernel
- regularization theory
- statistical learning theory

Many kernels

- generic: linear, polynomial, Gaussian
- for histograms: homogeneous intersetion, Chi2, sqrt, log, …
- combinations: exp-chi2, MKL, …

Kernel trick: flexibility

- learn with any kernel

Data aware approximations

- (additive) Nystrom
- incomplete Cholesky

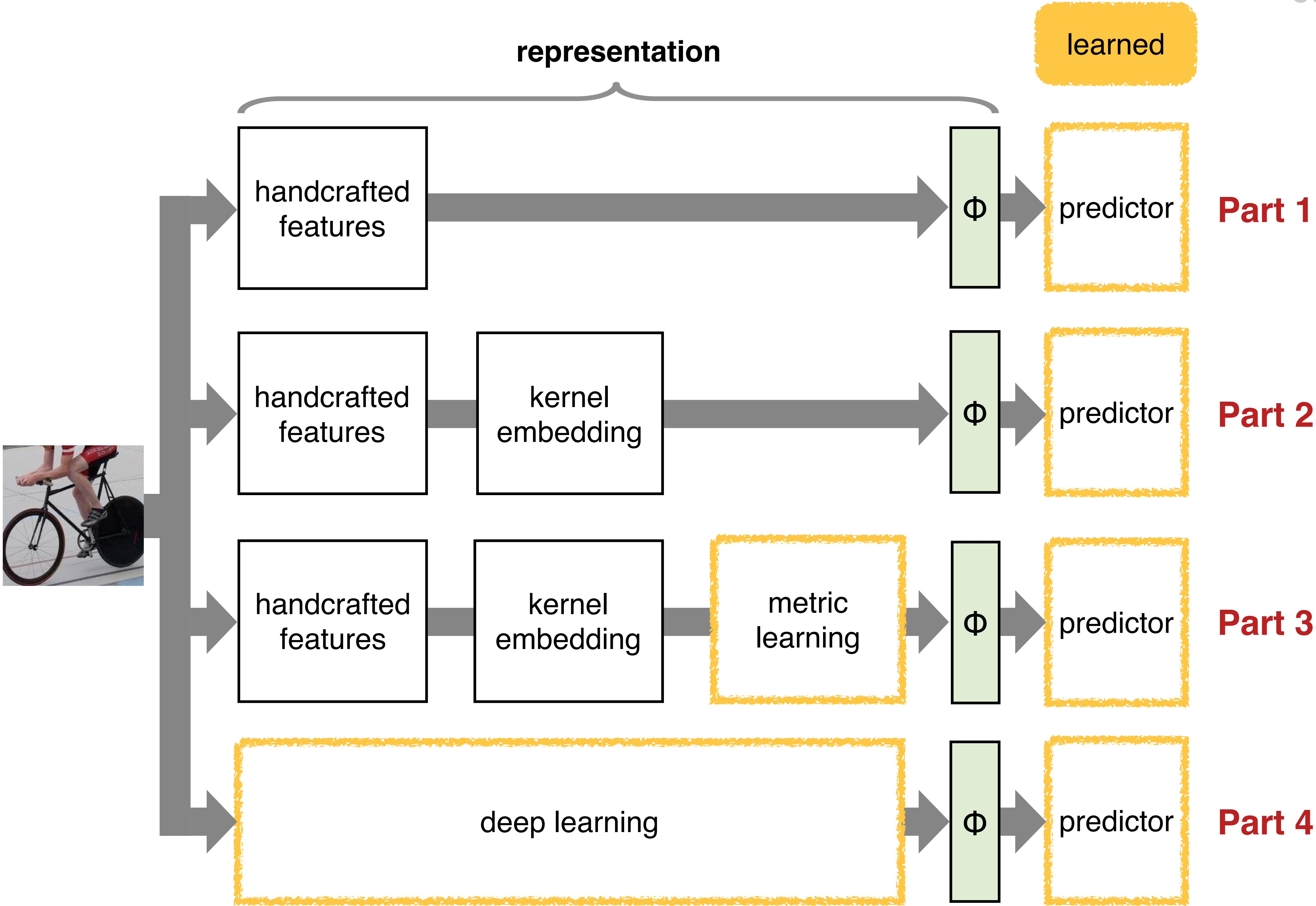
Data-agnostic approximations

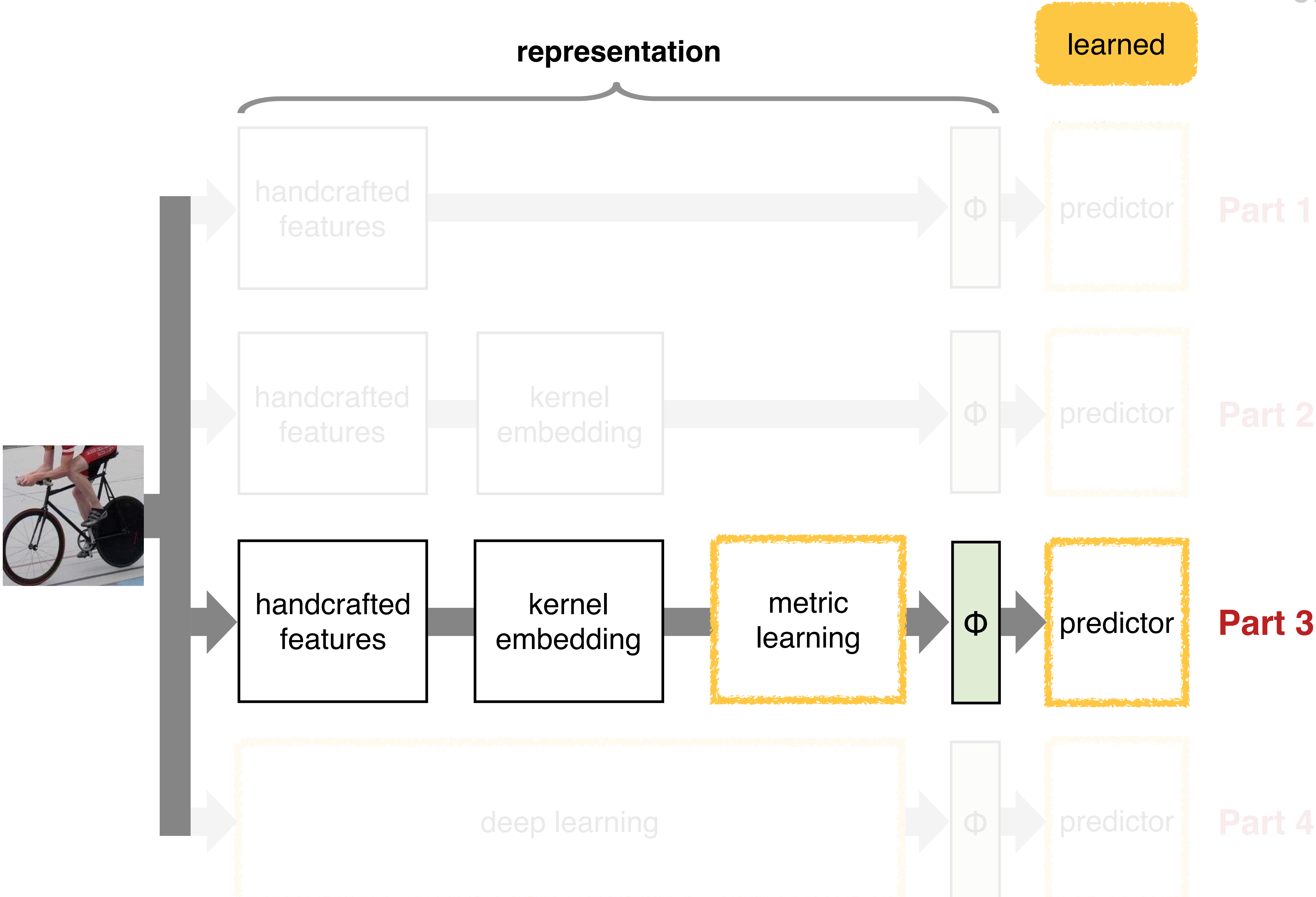
- random Fourier features
- fast food
- homogeneous kernel map
- intersection kernel map

Algorithms

- power mean for add. kernels
- online-learning with kernels

…


learned
representation
handcrafted features
Φ
predictor
Part 1
handcrafted features
kernel embedding
Φ
predictor
Part 2
handcrafted features
kernel embedding
metric learning
Φ
predictor
Part 3
deep learning
Φ
predictor
Part 4



representation
learned
handcrafted features
Φ
predictor
Part 1
handcrafted features
kernel embedding
Φ
predictor
Part 2
handcrafted features
kernel embedding
metric learning
Φ
predictor
Part 3
deep learning
Φ
predictor
Part 4

Learning to compare

For a thorough review: [Weinberger Saul JMLR 2009]

Goal

- compare (rather than classify) objects **x**, **y**
- formally, learn a distance $d^2(\mathbf{x},\mathbf{y})$

Desiderata

- if **x** and **y** are *congruous* $\Longrightarrow$ small distance
- if **x** and **y** are *incongruous* $\Longrightarrow$ large distance

Parametrisation of the distance

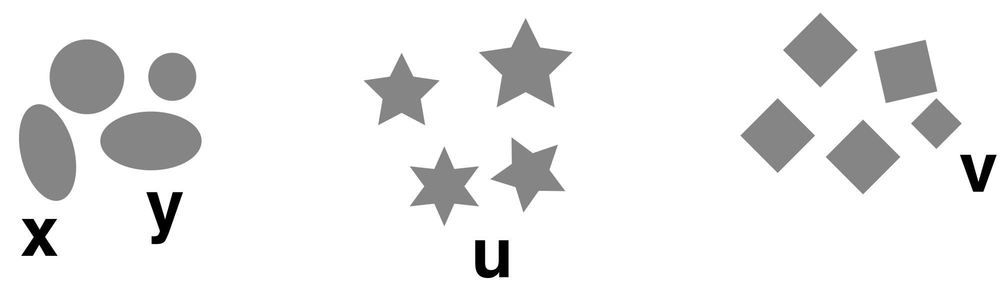
Euclidean distance + linear projection W

$$\boxed{d_W^2(\mathbf{x},\mathbf{y}) = \|W\mathbf{x} - W\mathbf{y}\|^2}$$

Classification-like constraints

For all object pairs **x**, **y**

- congruous $\Longrightarrow$ distance **smaller** than threshold - margin
- incongruous $\Longrightarrow$ distance **larger** than threshold + margin



$$d_W^2(\mathbf{x},\mathbf{y}) = \|W\mathbf{x} - W\mathbf{y}\|^2 < b$$

congruous pairs

$$d_W^2(\mathbf{u},\mathbf{v}) = \|W\mathbf{u} - W\mathbf{v}\|^2 > b$$

incongruous pairs

$$\boxed{d_W^2(\mathbf{x},\mathbf{y}) < b-1, \qquad d_W^2(\mathbf{u},\mathbf{v}) > b+1}$$

Learning formulation

$$\min_{W,b} \mathcal{R}(W) + \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{P}} \max\{0, 1 - b + d_W^2(\mathbf{x},\mathbf{y})\} + \sum_{(\mathbf{u},\mathbf{v})\in\mathcal{N}} \max\{0, 1 + b - d_W^2(\mathbf{u},\mathbf{v}))\}$$

Input: training data

- congruous pairs $\mathcal{P}$ (i.e., positive)
- incongruous pairs $\mathcal{N}$ (i.e., negative)

Input: regulariser $\mathcal{R}(W)$

- controls which type of solution is found
- may induce smoothness, sparsity, group-sparsity, low rank

Output: projection matrix W

Algorithm and variants

- Convex + sparsity: regularized dual averaging
- Non-convex + fixed dimensionality: stochastic gradient descent

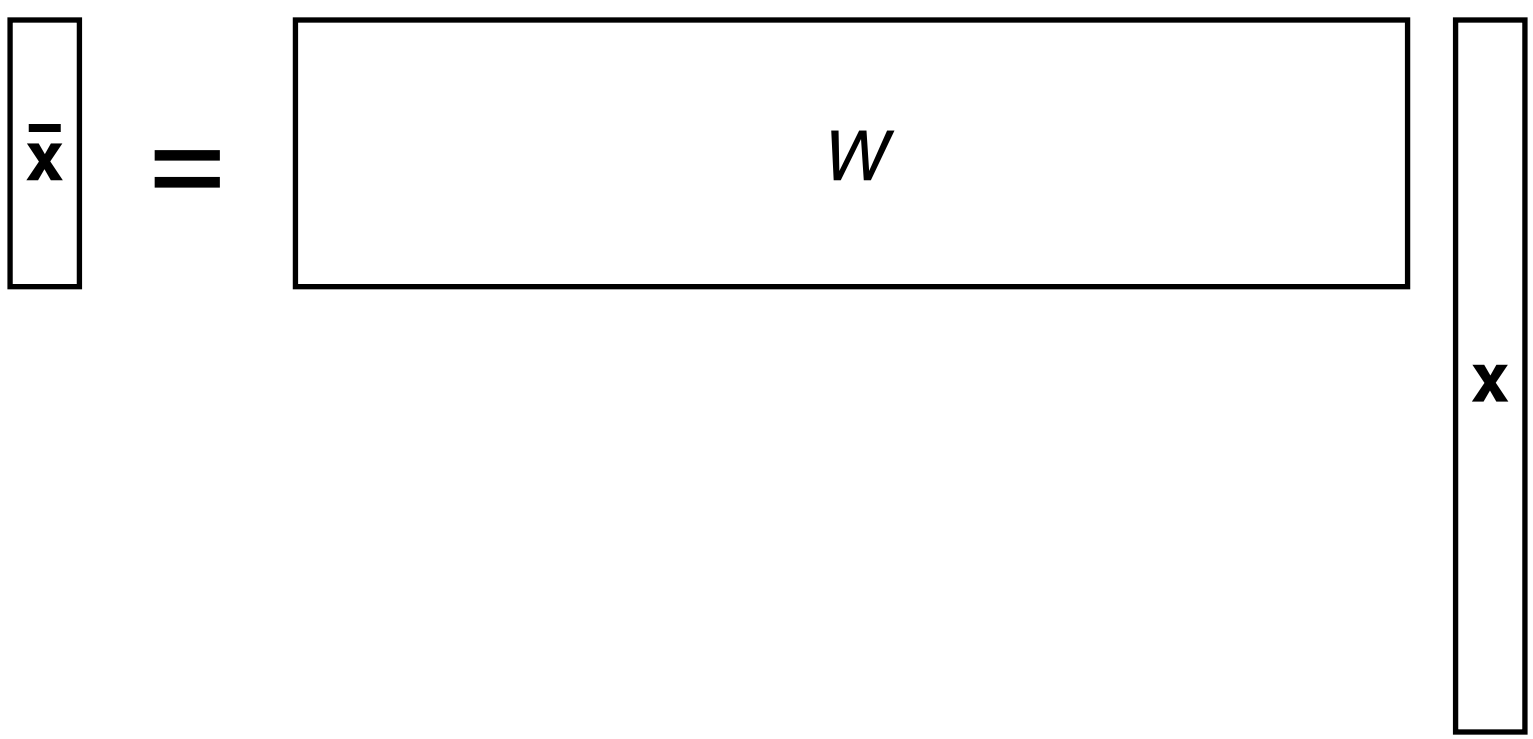
Compare & compress

Euclidean distance		linear projection
$d_W^2(\mathbf{x}, \mathbf{y}) = \lVert W\mathbf{x} - W\mathbf{y} \rVert^2$	+	$\mathbf{x} \in \mathbf{R}^n \xrightarrow{W \in \mathbf{R}^{m \times n}} \bar{\mathbf{x}} = W\mathbf{x} \in \mathbf{R}^m$

W improves the data separation (= learns a meaningful similarity)

W can also **reduce the data dimensionality**

- simply pick m ≪ n



Learning to verify people identities

[Simonyan *et al.* BMVC 2013]

Task

- decide if two pictures portray the same person
- learning accurate and compact face descriptors

Code available

- http://www.robots.ox.ac.uk/~vgg/software/face_desc/

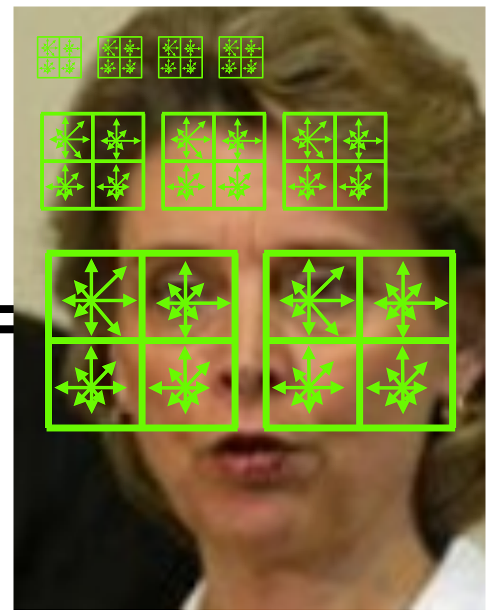
See also [Guillaumin *et al.* ICCV 2009, Sharma Hussain Jurie ECCV 2012 , Chen *et al.* CVPR 2013]

Fisher Vector Faces (FVF)

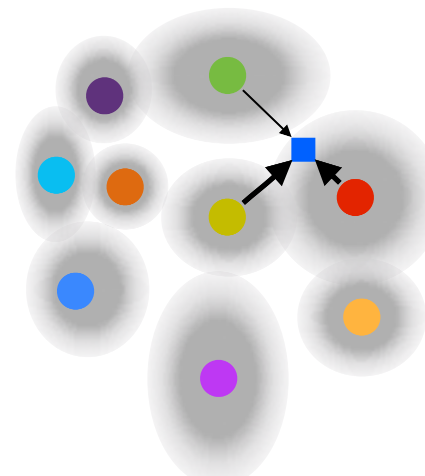
[Simonyan *et al.* BMVC 2013]

Dense SIFT | Fisher Vector | Metric learning

descriptor computation =



+



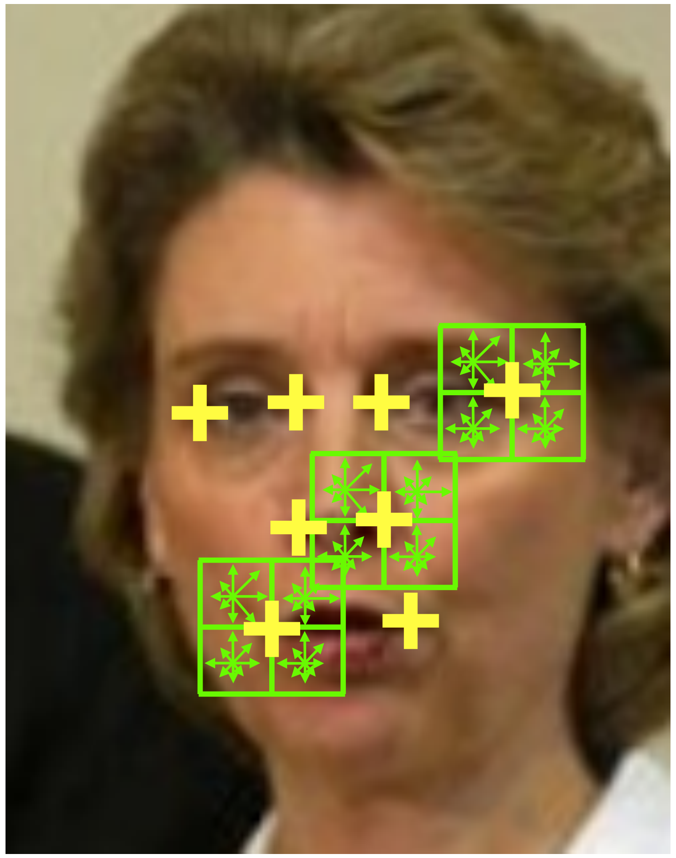
+

$$d^2_W(\mathbf{x}, \mathbf{y}) = \|W\mathbf{x} - W\mathbf{y}\|^2$$

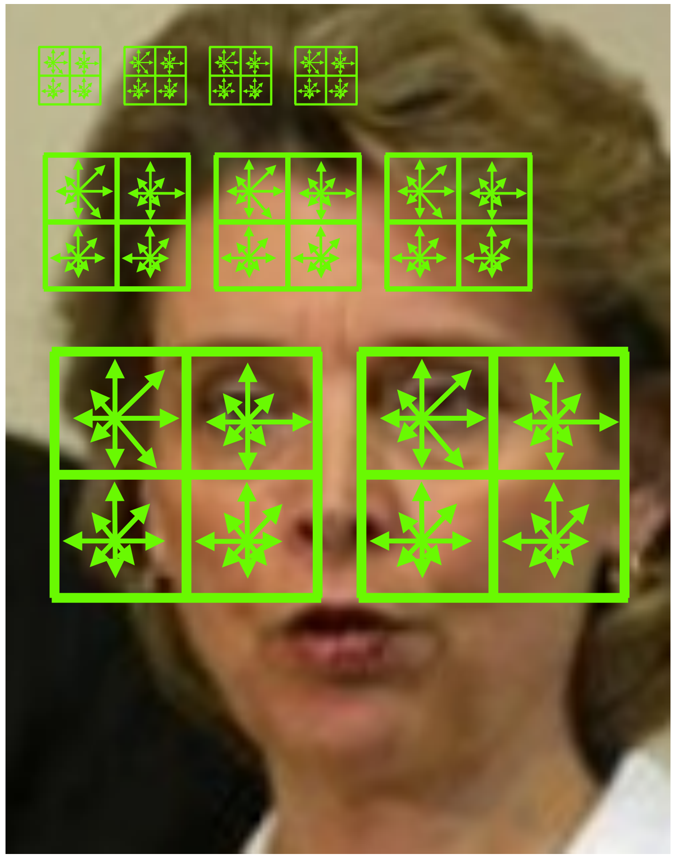
1. **FVF descriptor**
   A. Features: *densely sampled*, *spatially augmented* SIFT features
   B. Encoding: Fisher Vectors
   C. Metric learning & dimensionality reduction
   D. Optional post-processing: binarization

Landmarks or not?

landmarks



FVF



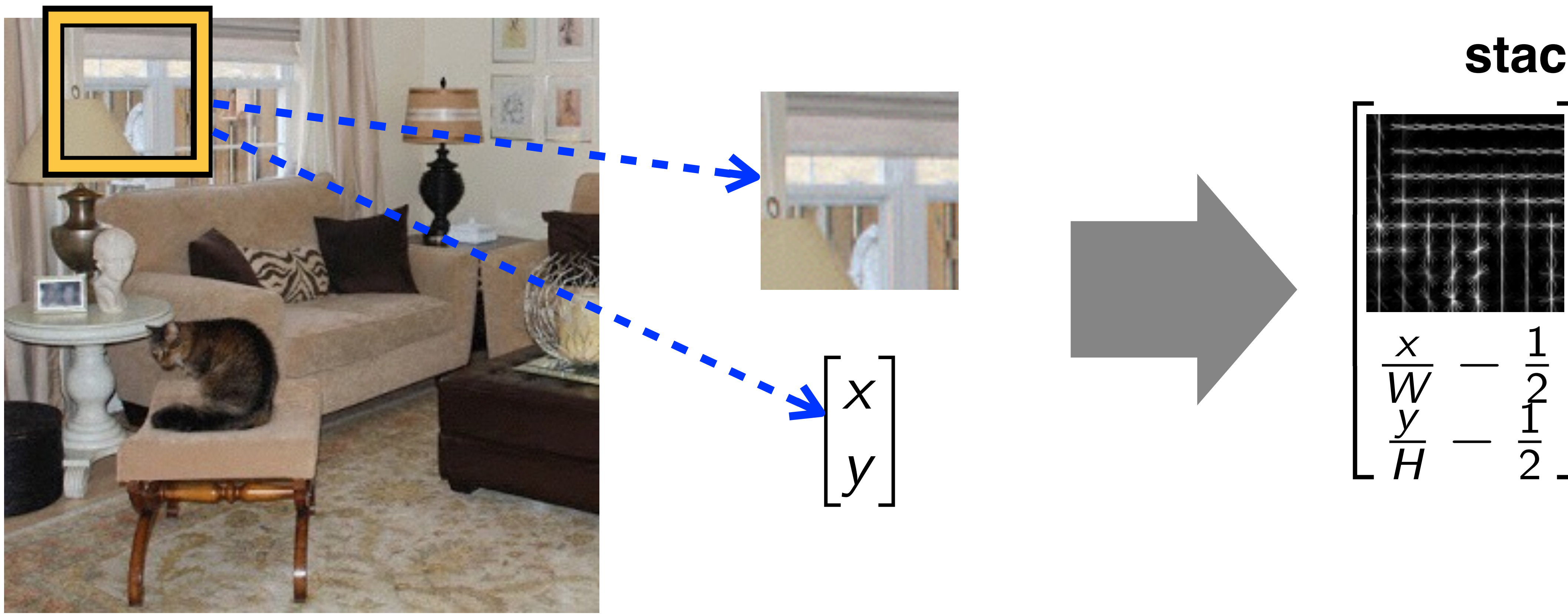
Landmarks

- sample patches at landmarks
- good: alignment
- bad: expensive, brittle

Dense sampling

- sample patches uniformly
- good: simple, robust
- bad: no alignment

Spatially-augmented descriptors

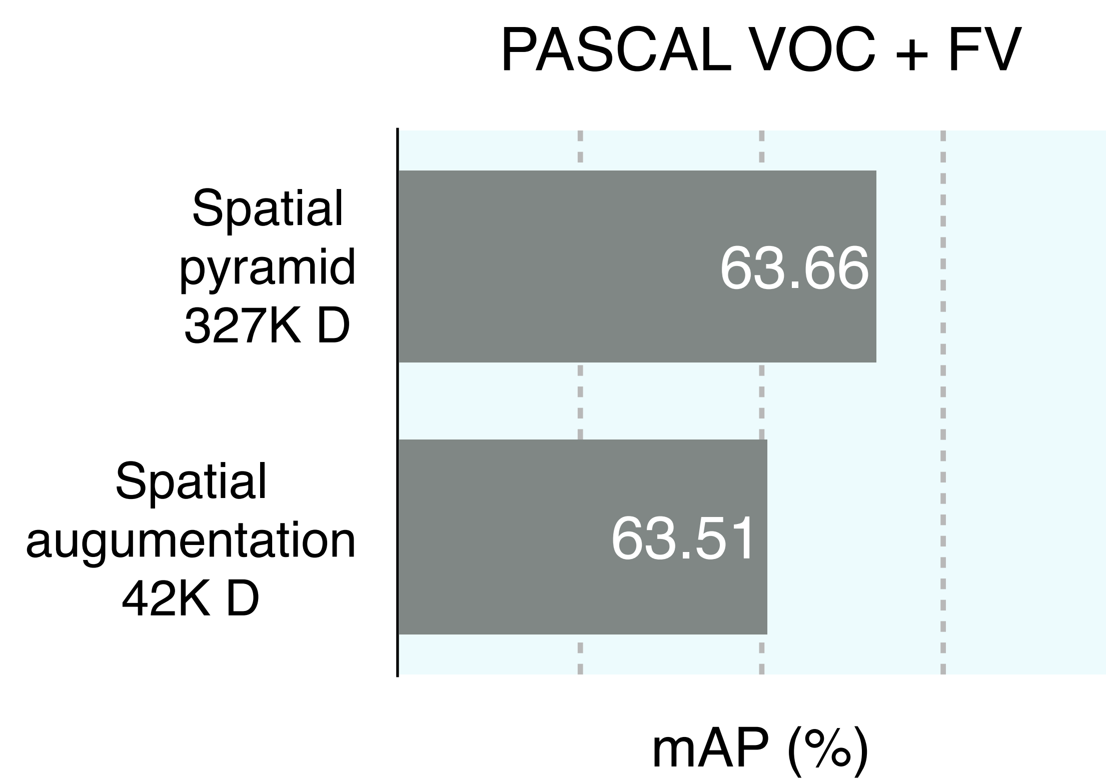


Spatial augmentation
[Sanchez *et al.* PRL 2011]

- Append (x,y) to descriptors
- Alternative to spatial pyramid

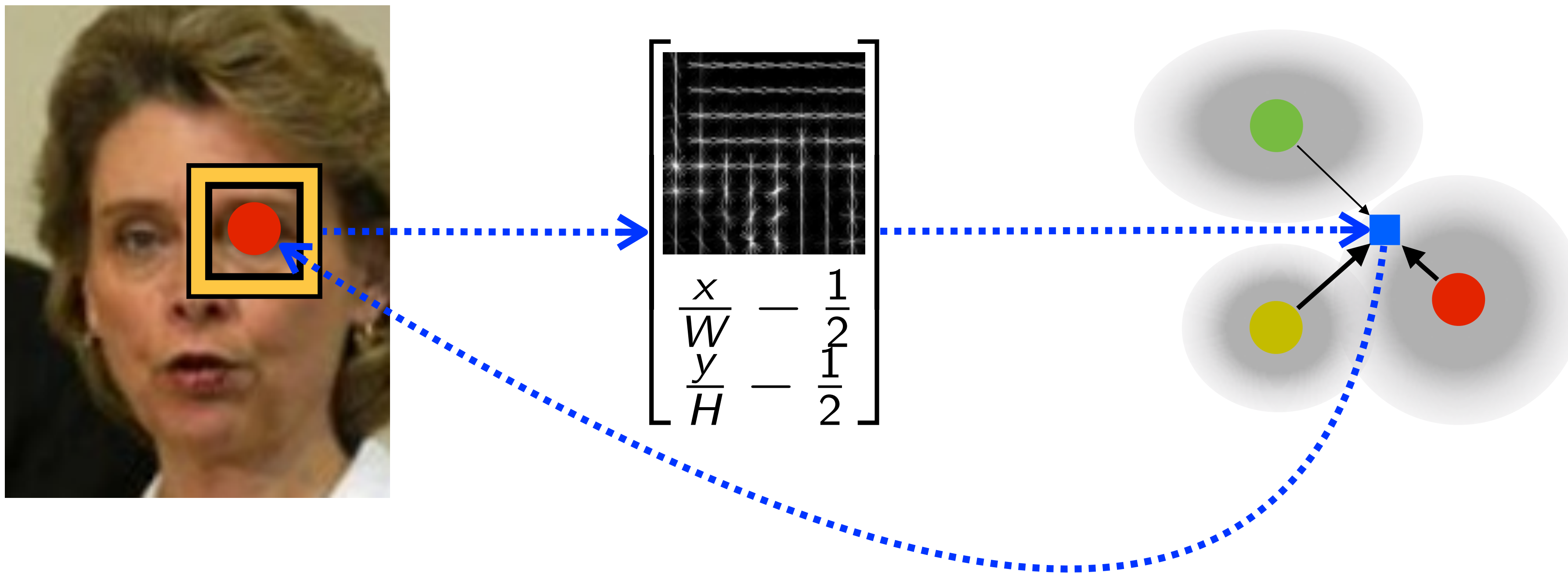
Greatly reduced dimensionality

- *e.g.* 7-fold



[Chatfield *et al.* 2014]

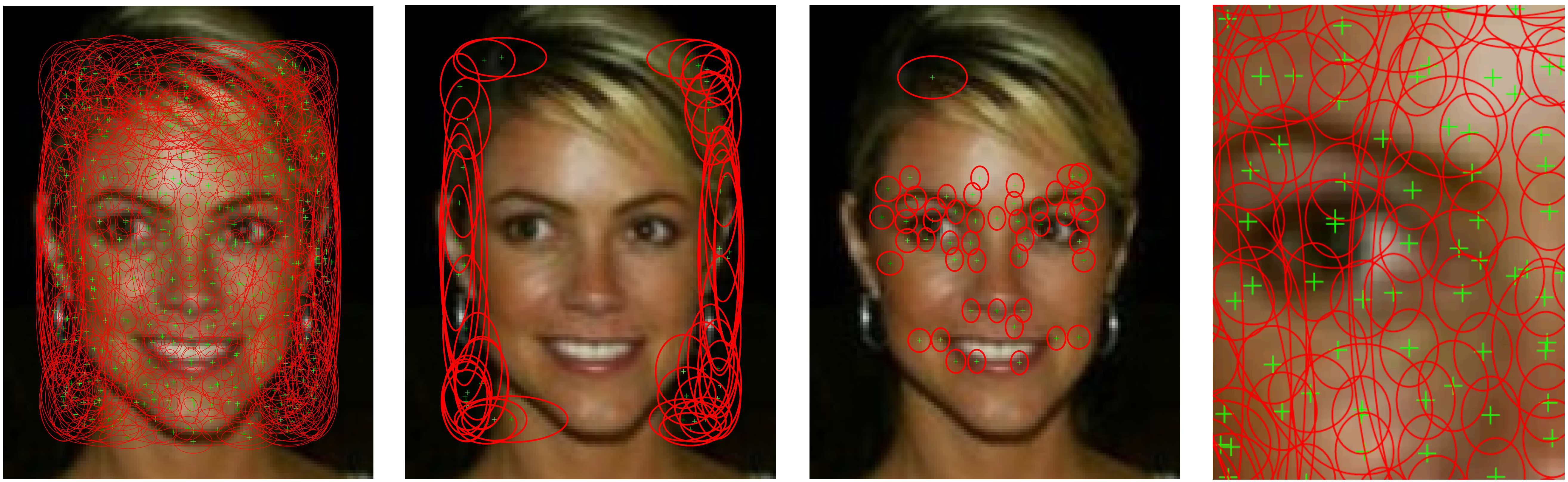
Fisher Vectors as part-based models

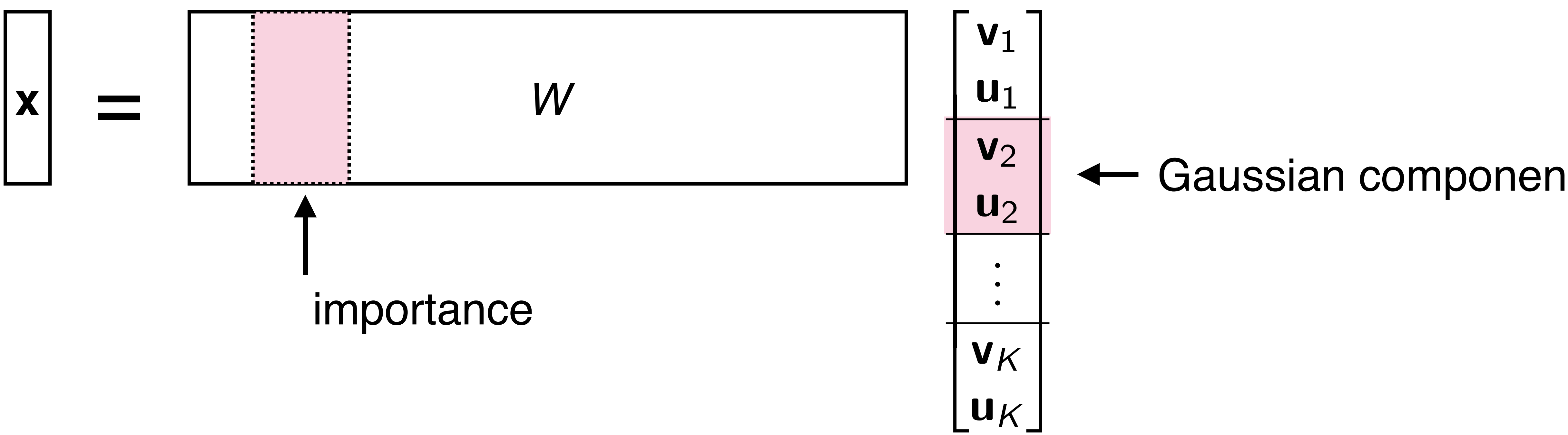


Fisher Vectors as part-based models

Distinctive face elements

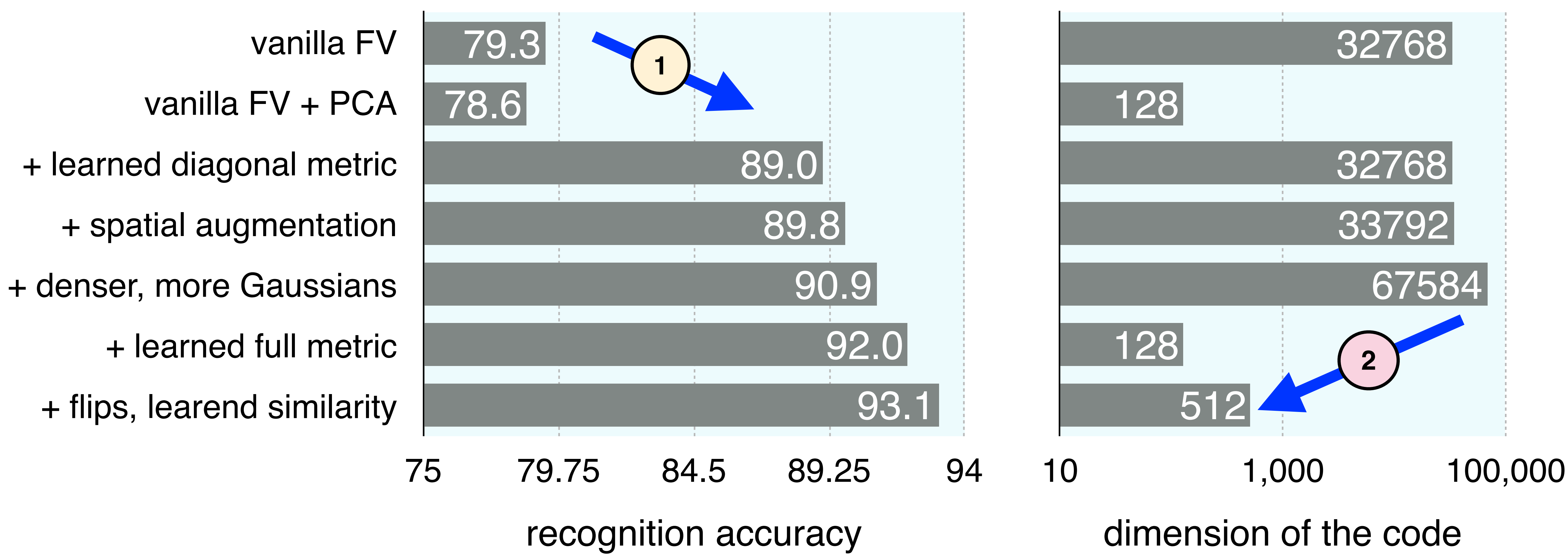
irrelevant important detail





FVF design choices

Benchmark: Labelled Faces in the Wild (LFW)



1

Metric learning dramatically boosts **performance**

2

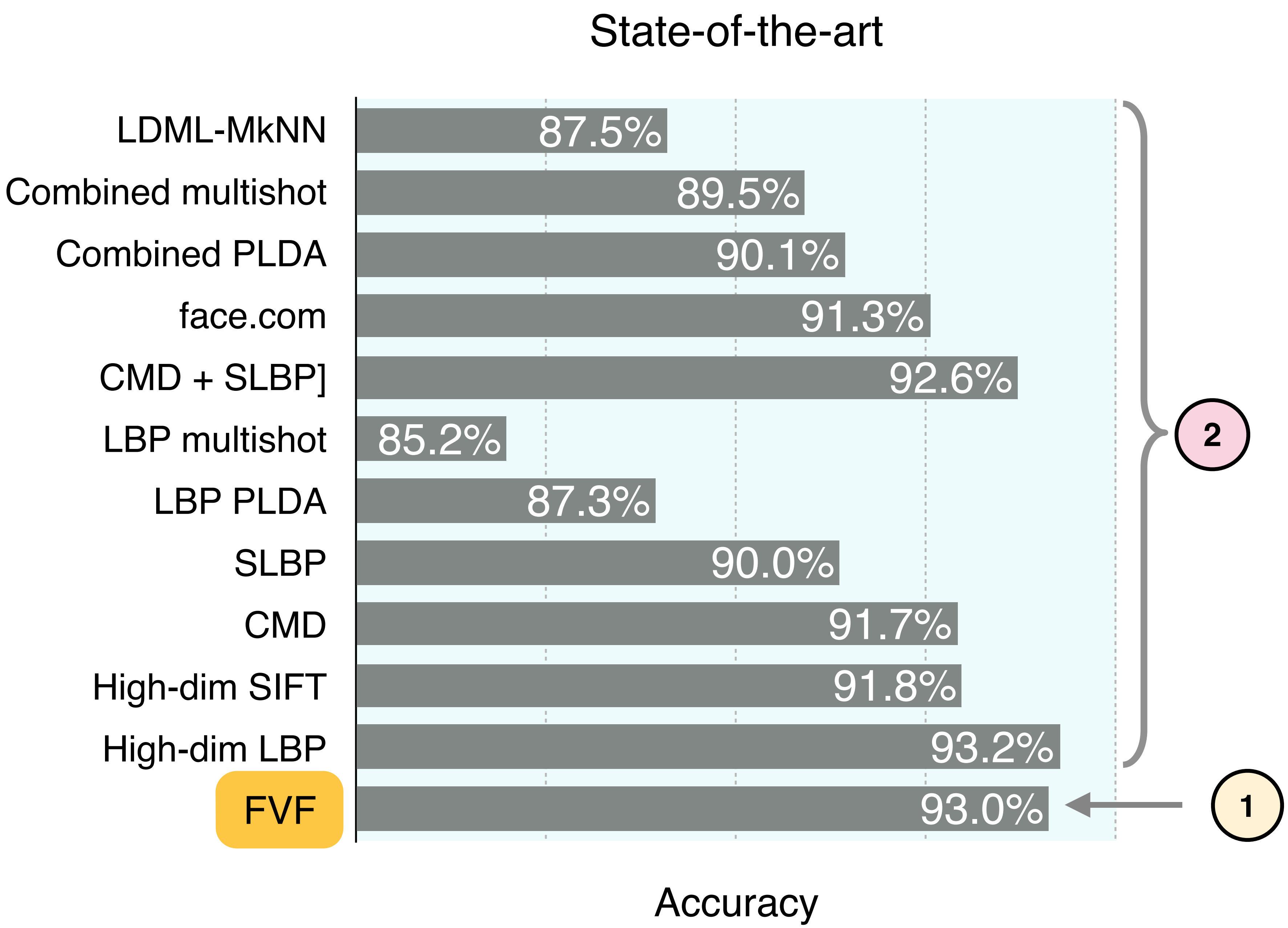
Learning a **full metric** achieves a very significant **compression**

3

Simple
(no alignment / landmarks)

FVF still image performance

Benchmark: Labelled Faces in the Wild



1

Accurate

Fast

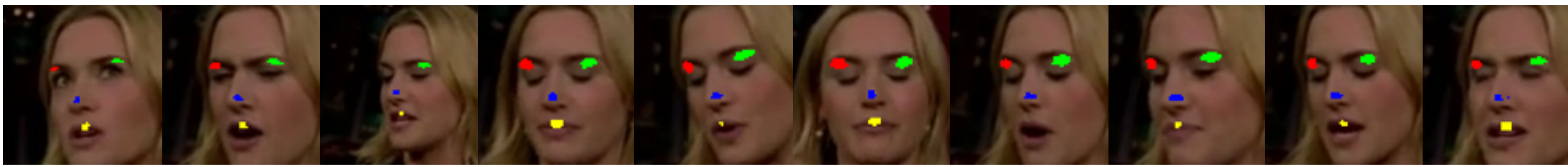
Small

2

Simpler than
most alternatives

Video Fisher Vector Faces (VF^2)

[Parkhi *et al.* CVPR 2014]



From still images to videos

- RootSIFT
- Image, video, and jittered pooling

Dimensionality reduction

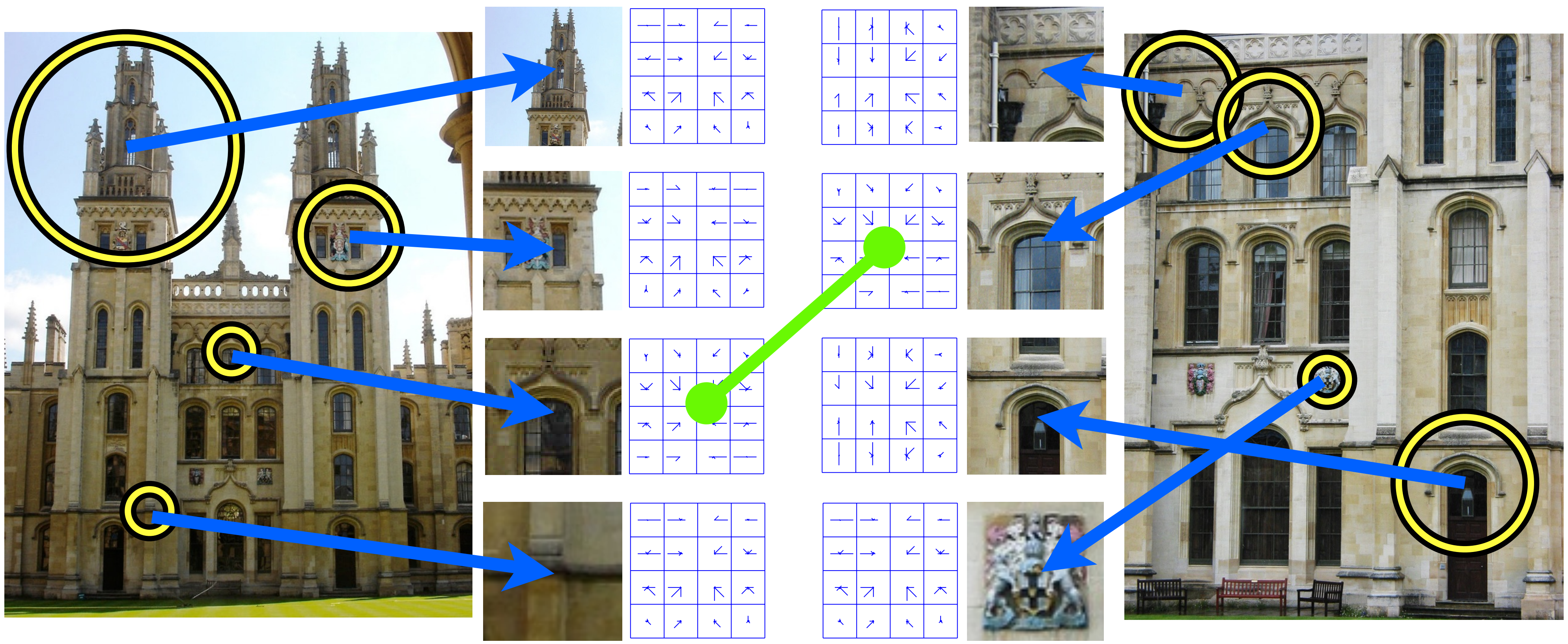
- Metric learning
- **Binarization**

Exciting area of research: hashing, binarization

https://sites.google.com/site/lsvrtutorialcvpr14/
[Jegou]

Other applications: local descriptor learning

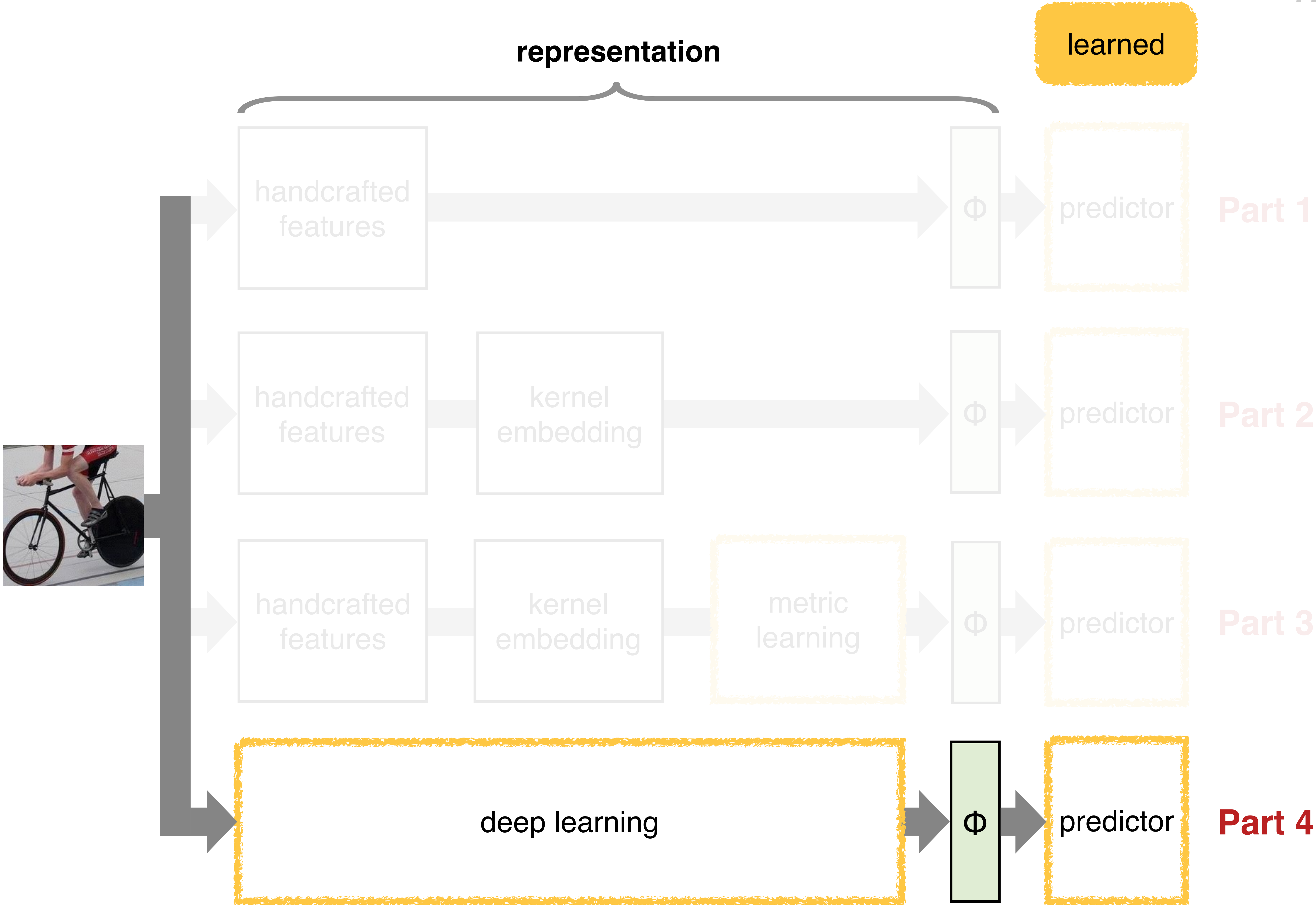
[Simonyan *et al.* ECCV 2012, PAMI 2014]



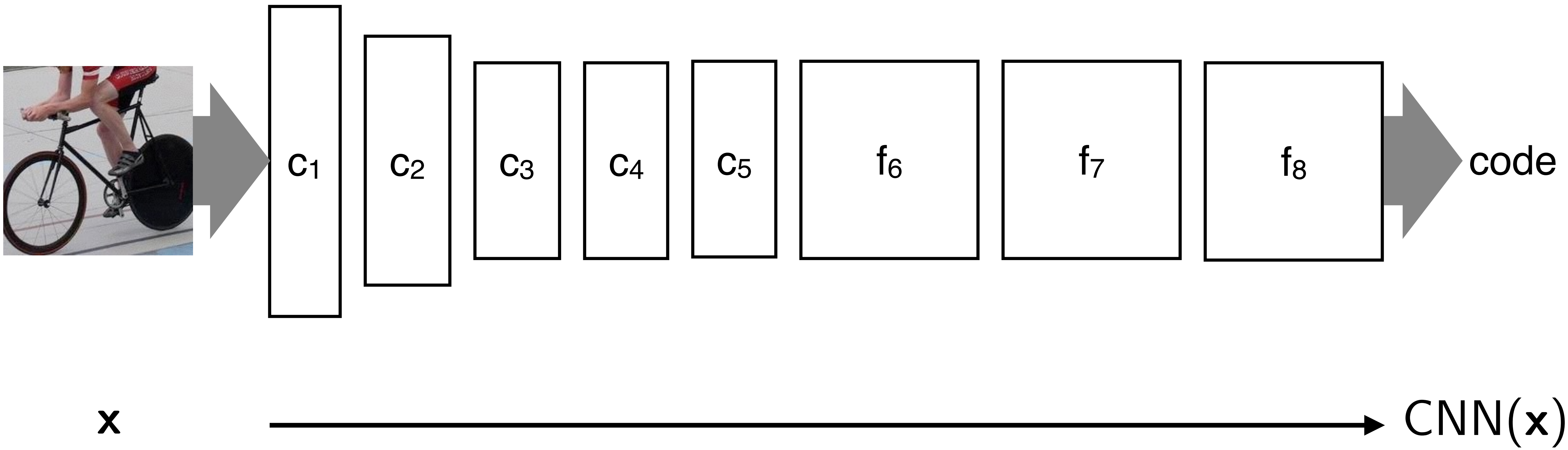
Learning to compare & compress works beyond faces

State-of-the-art **local descriptors** and **instance search**

http://www.robots.ox.ac.uk/~vgg/software/learn_desc/


representation
learned
handcrafted features
Φ
predictor
Part 1
handcrafted features
kernel embedding
Φ
predictor
Part 2
handcrafted features
kernel embedding
metric learning
Φ
predictor
Part 3
deep learning
Φ
predictor
Part 4

Convolutional neural networks (CNNs)



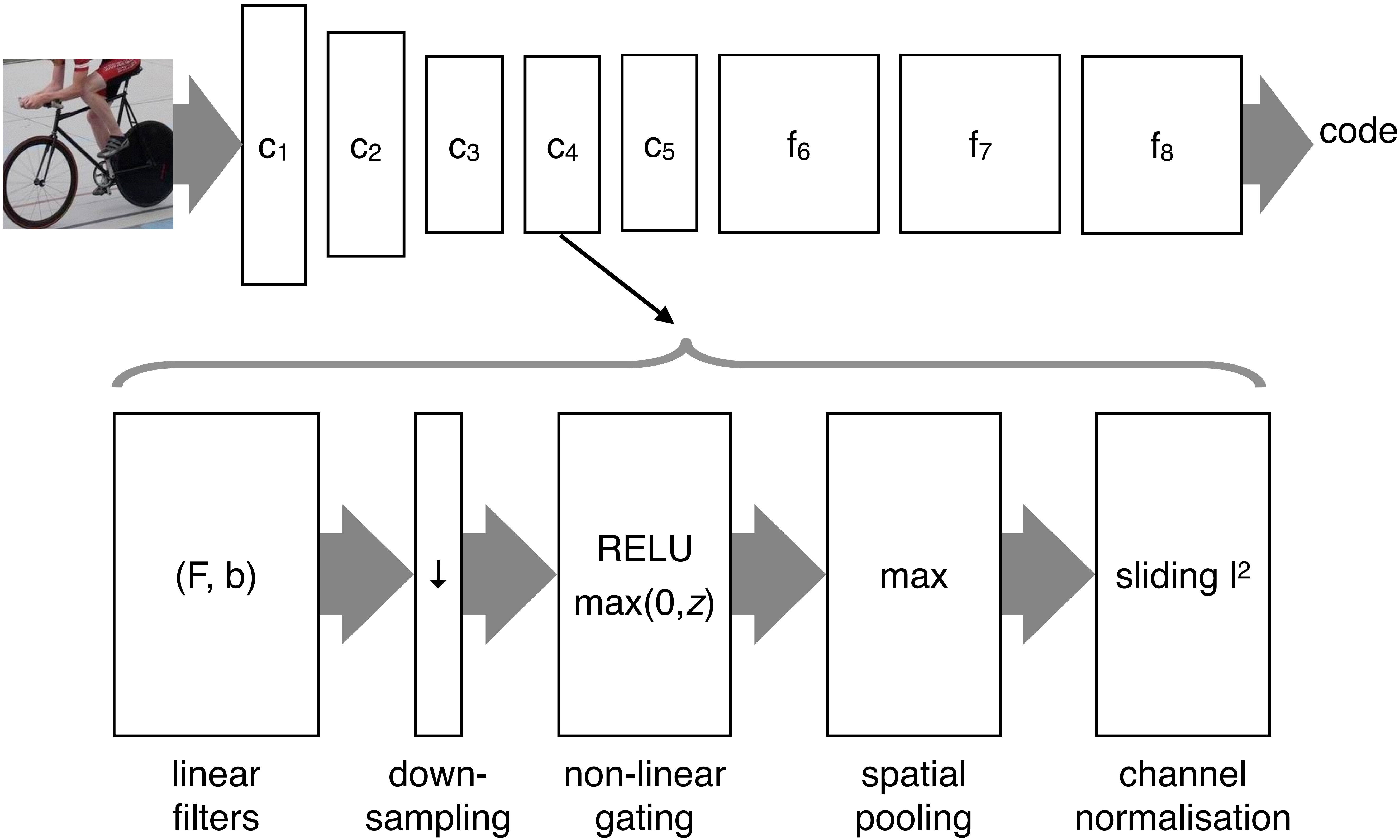
From left to right

- decreasing spatial resolution
- increasing feature dimensionality

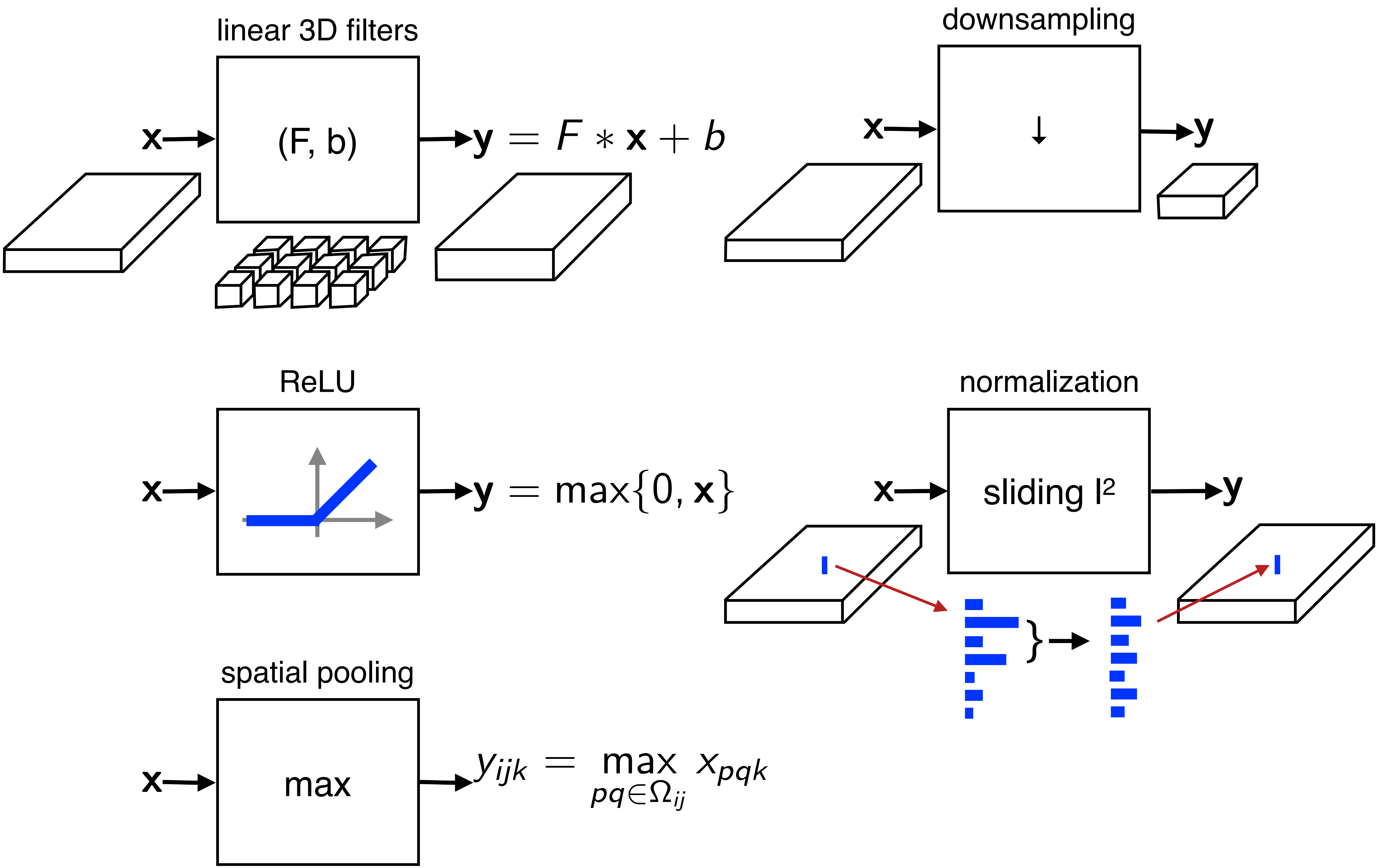
Fully-connected layers

- same as convolutional, but with 1×1 spatial resolution
- contain most of the parameters

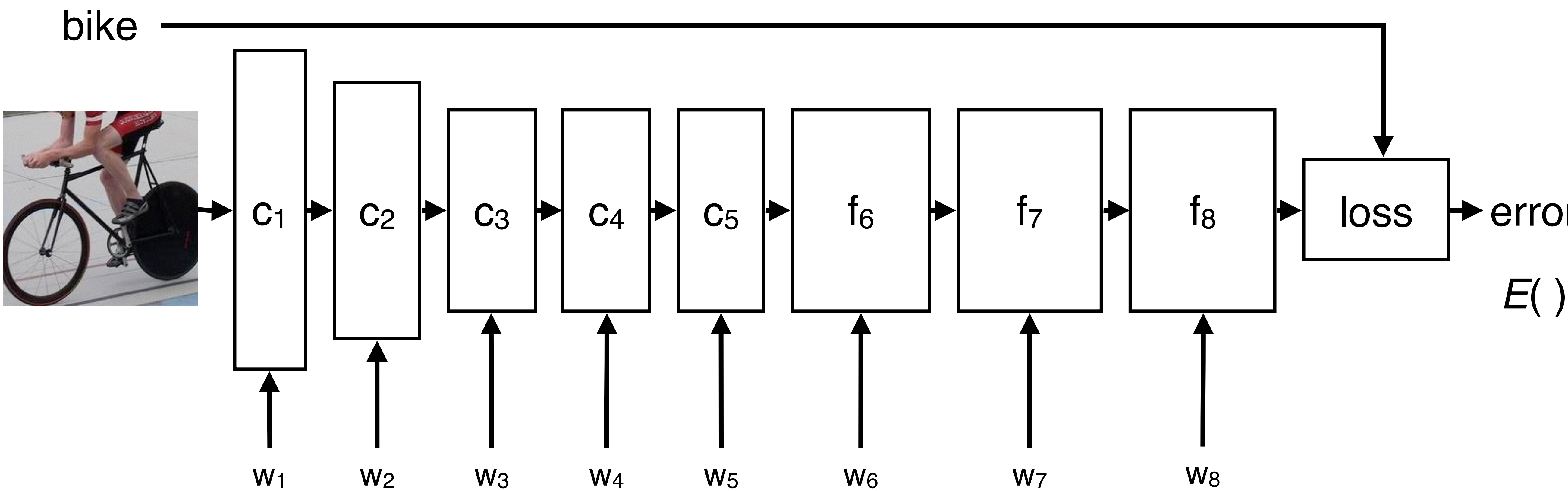
Convolutional layers



CNN components



Learning a CNN



$$\operatorname{argmin} E(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_8)$$

Stochastic gradient descent
(with momentum, dropout, …)

Learning CNNs classifiers

Challenge

- many parameters, prone to overfitting

Key ingredients

- very large annotated data
- heavy regularisation (dropout)
- stochastic gradient descent
- GPU(s)

IMAGENET

- 1K classes
- ~ 1K training images per class
- ~ 1M training images

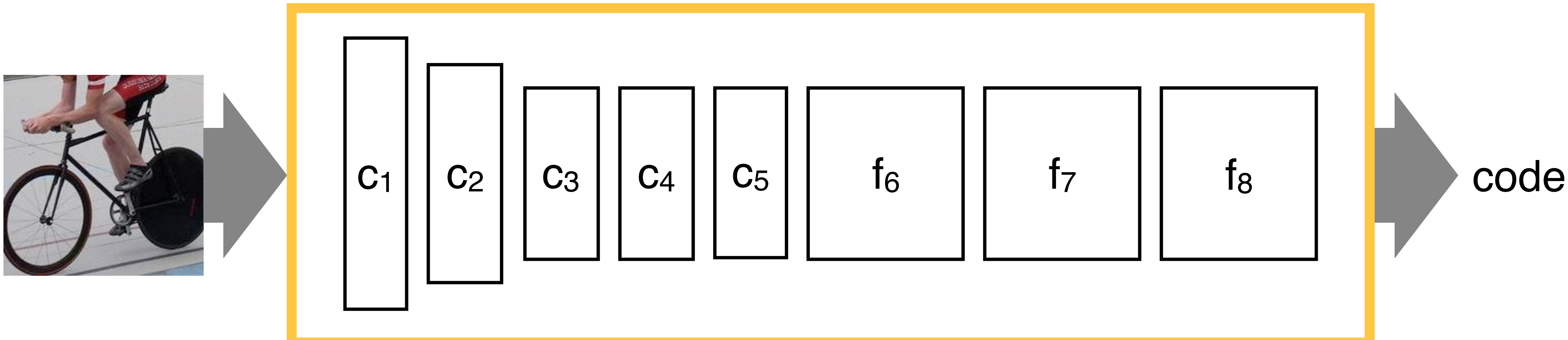
Training time

- ~ 90 epochs
- days—weeks of training
- requires processing ~150 images/sec

What do CNNs learn?

Deep dreams

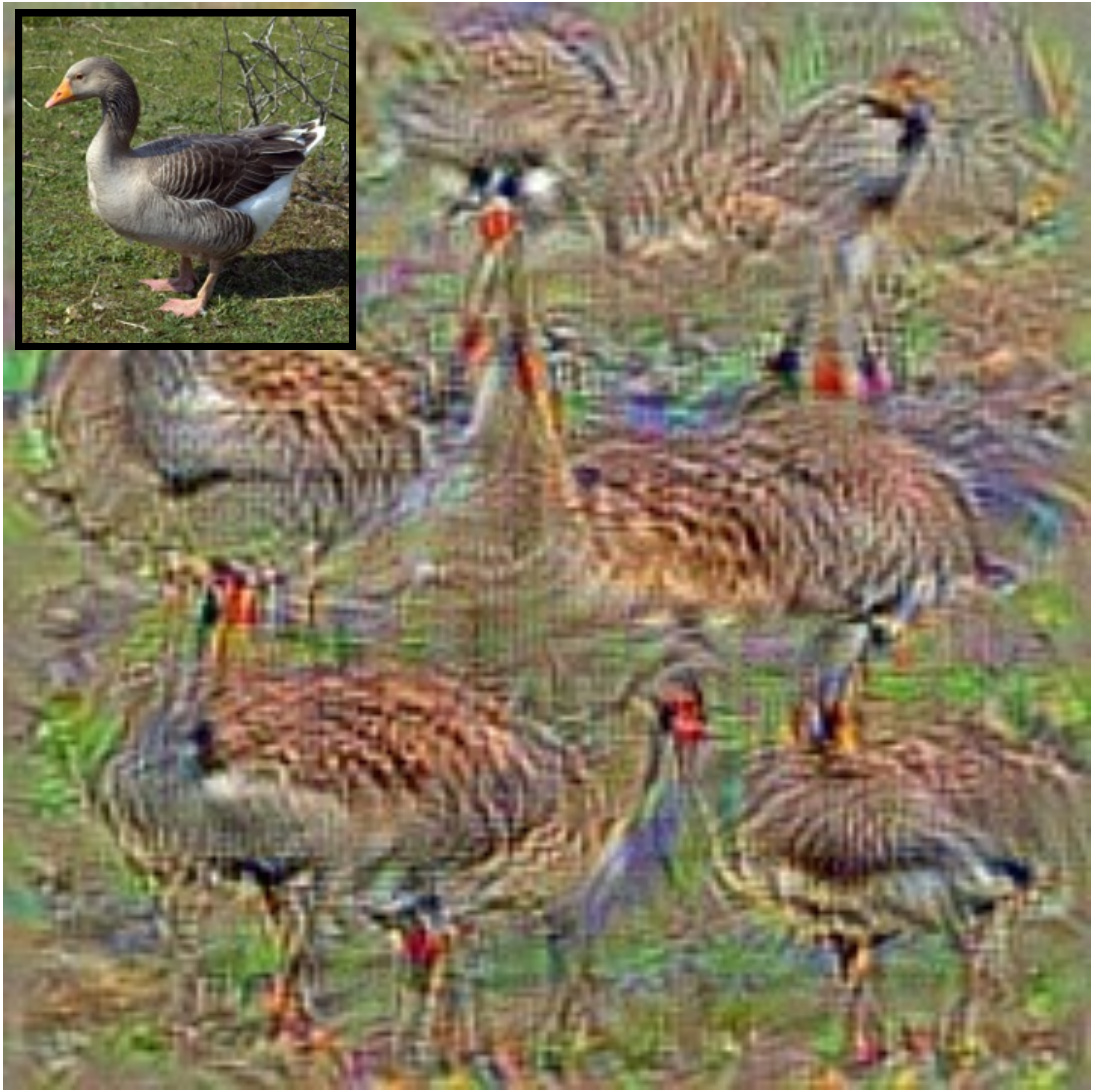
[Erhan *et al.* 2009, Simonyan *et al.* ICLR 2014]

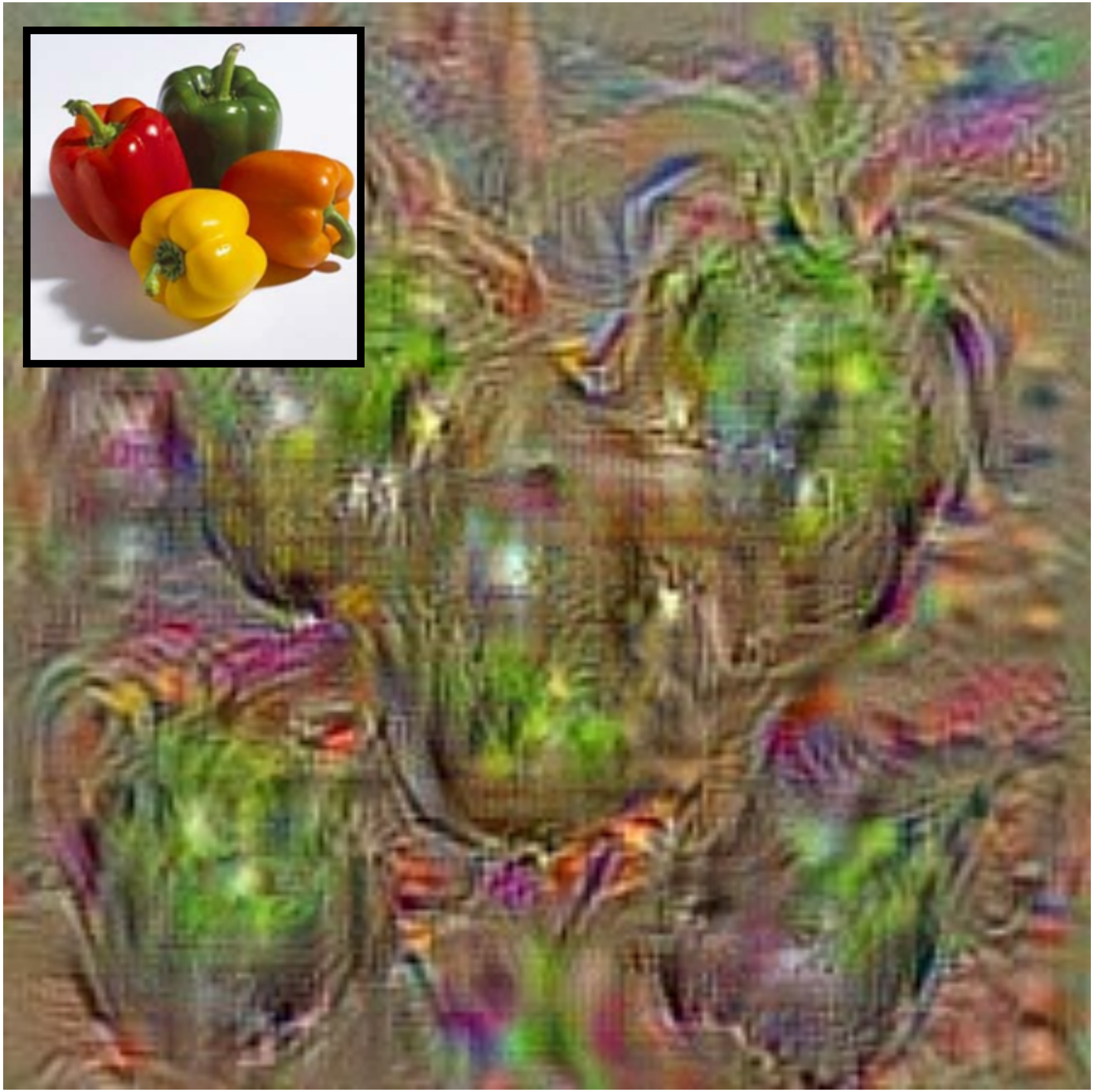


What does deep learning learn?

Invert a CNN by finding the image that maximises the output of a class

$$\mathbf{x}^* = \mathrm{argmax}_{\mathbf{x}}\, \mathrm{CNN}_c(\mathbf{x})$$

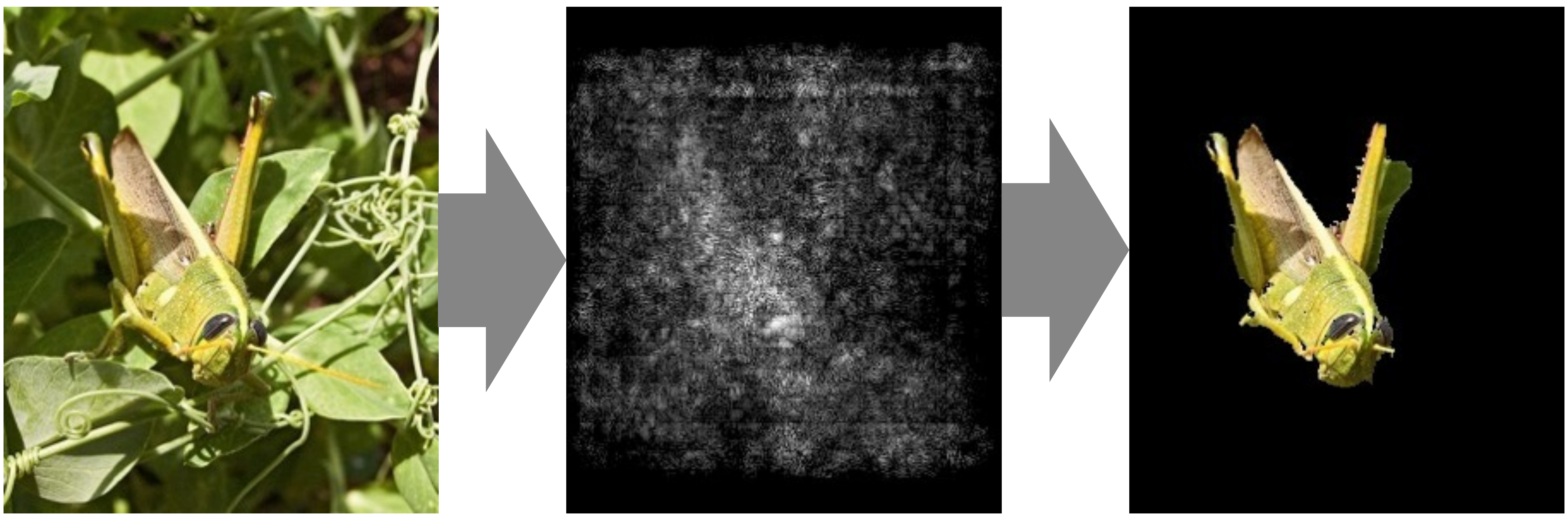




Weakly-supervised detectors

This can be used to **segment objects**

Remarkably, *no object segmentation or bounding box is given during training*



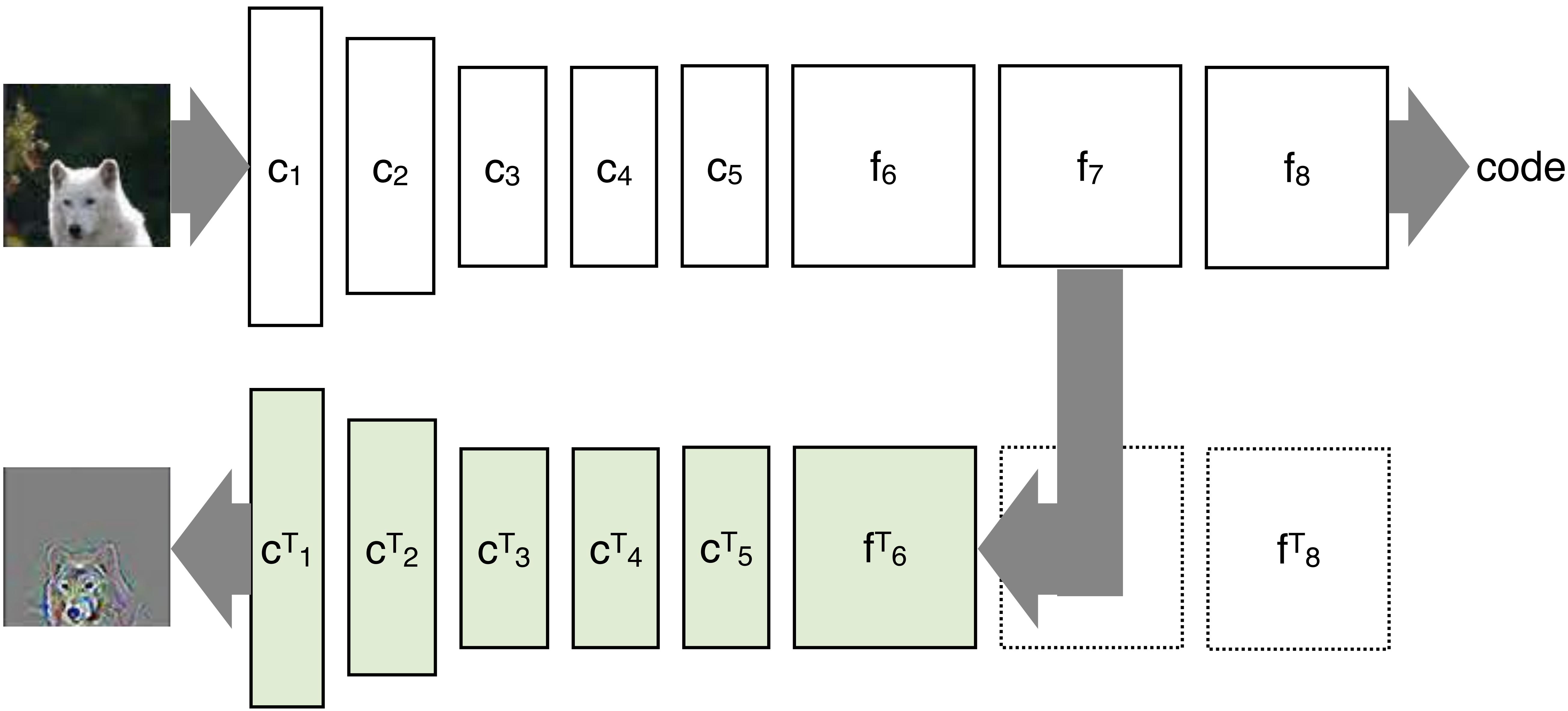
input image   input saliency   grabcut

[Simonyan *et al.* ICLR 2014]

De-convolutional networks

[Zeiler Fergus ECCV 2014]

“Transpose” the architecture to go from activations back to image

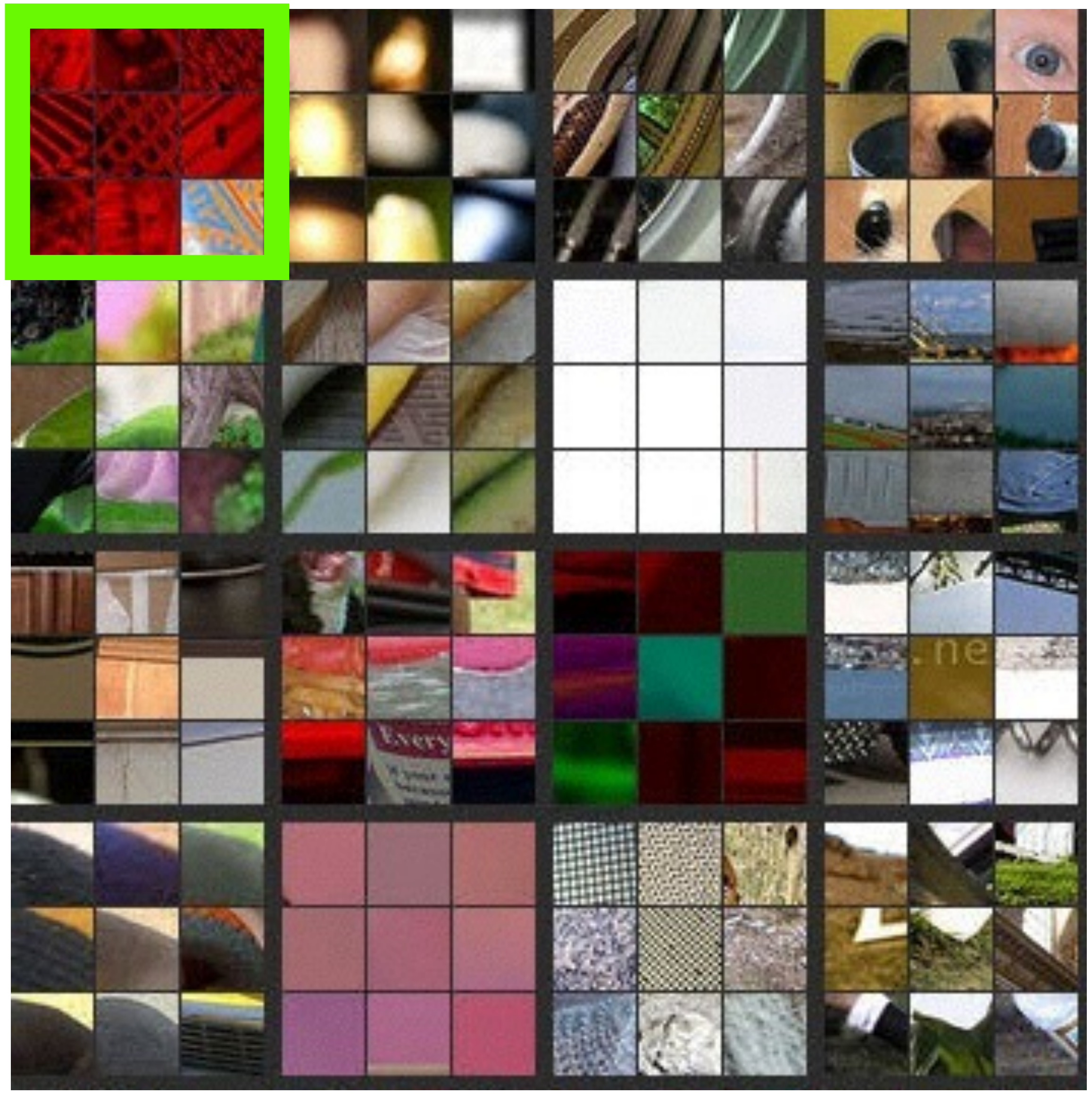


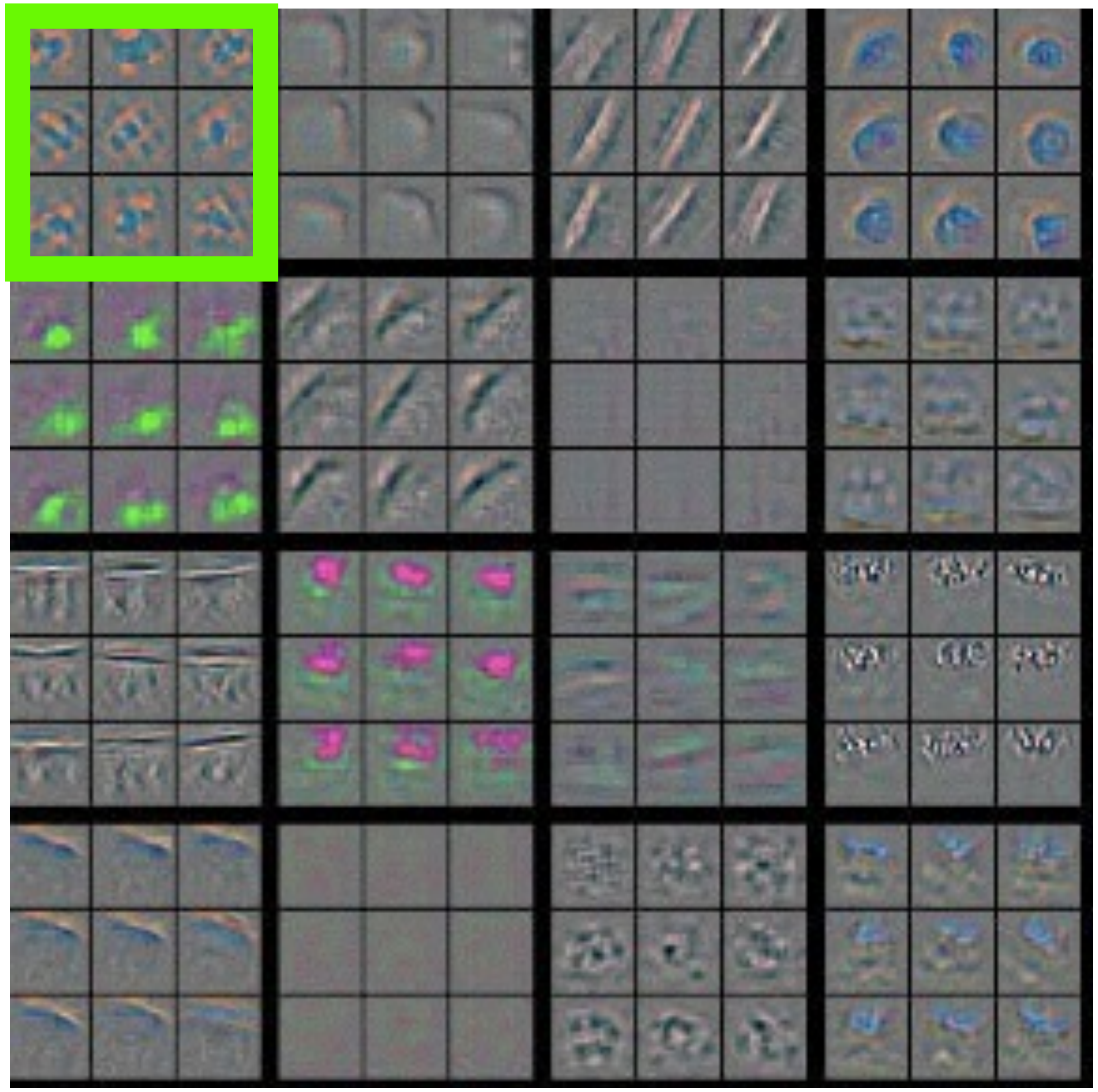
Deconvent visualization

Visualize sample images that excite a given neuron the most

Layer 1

filter
response





top 9 exciting patches
for each neuron

their deconvnet
reprojection

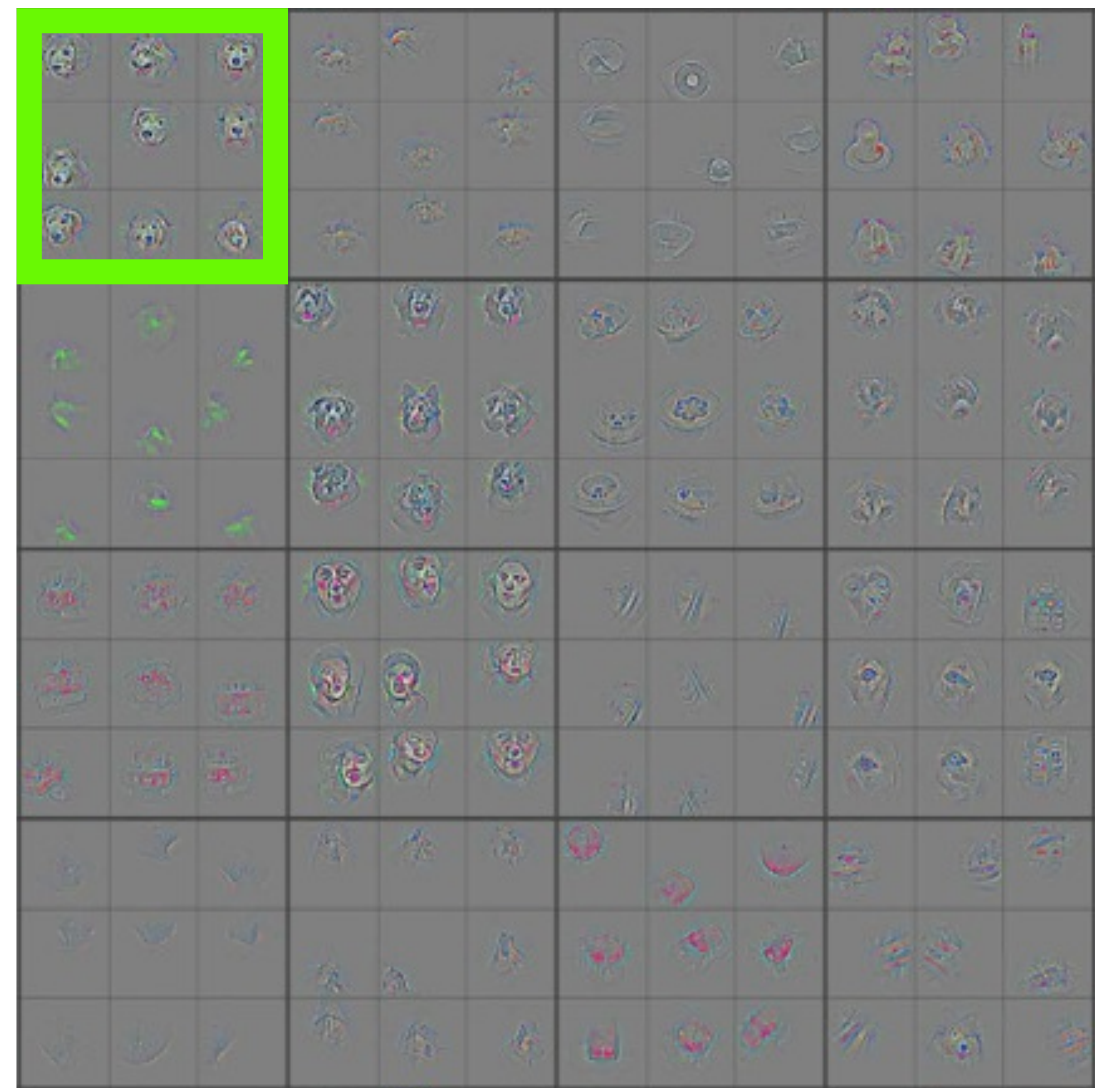
Deconvent visualization

Visualize sample images that excite a given neuron the most

Layer 5

filter
response

top 9 exciting patches
for each neuron



their deconvnet
reprojection

What is a deconvnet?

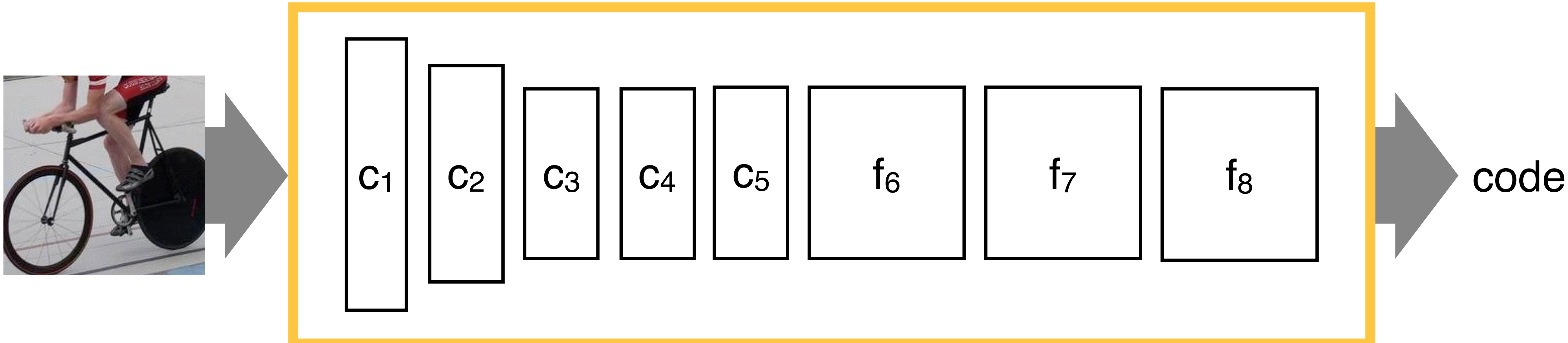
The "transpose" of the CNN

- transpose of the filters (as linear operators)
- max-pooling: remembers activations from forward pass

Alternative interpretation
[Simonyan *et al.* 2014]

- backpropagation applied to the maximum activation problem is neary the same as a deconvnet
- approximate equivalence of "deep dreams" and deconvnets

CNNs as general purpose encoders



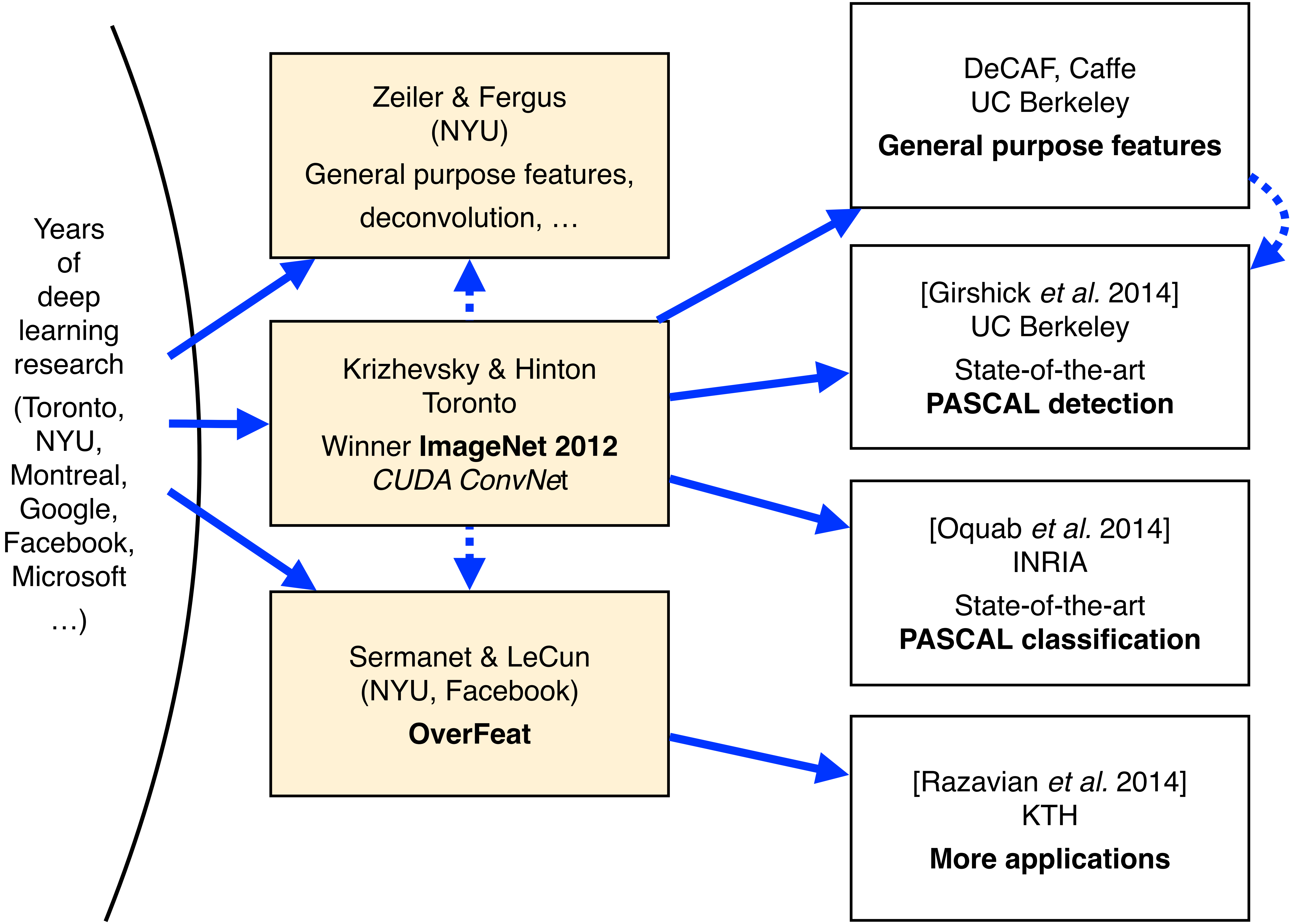
Pre-trained CNN encoders

- Architecture trained on ~ 1M ImageNet images
- Last softmax layer chopped off
- Output used as image encoding

Used as general-purpose features

- Applied to PASCAL VOC, Caltech, UCSD Birds, MIT Scene 67, …
- [Zeiler & Fergus, DeCAF, Caffe, …]

Deep visual encodings



Evaluating deep and shallow encoders

A preview of Tuesday talk

Deep or shallow? | Linear SVM

encoder → **predictor** → **label**

Shallow encoder

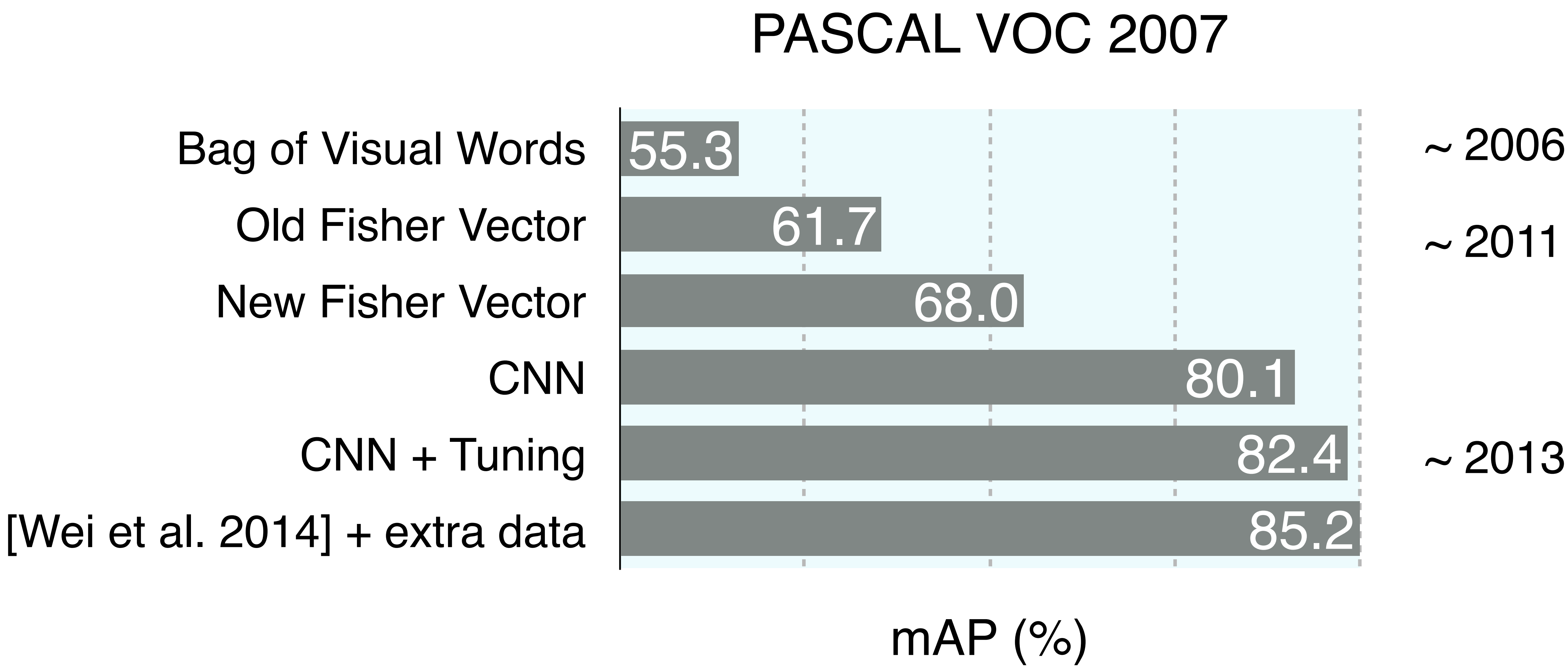
- Further Improved Fisher Vector

Deep encoders

- CNN Fast (CNN-F)
- CNN Medium (CNN-M)
- CNN Slow (CNN-S)

[Return of the devil is in the details, *Chatfield et al.* 2014]

Deep *vs.* shallow



CNNs

- Outperform shallow encodings
- Are expensive to train, but fast to evaluate
- Do provide low-dimensional, general-purpose codes
- Will definitely get much better

See Tuesday's talk for a thorough evaluation

Software & models

http://www.robots.ox.ac.uk/~vgg/software/deep_eval/

Feature generality

How large a gap can pre-trained features jump?

ImageNet pre-training

Object classification (PASCAL VOC)

- [Chatifield et al. 2014, Razavian et al. 2014, Zeiler et al. 2014]

Object detection (PASCAL VOC)

- R-CNN [Girshick et al. 2014]
- Requires region proposals and adaptation for accurate localisation

Fine-grained classification (UCSD birds)

- Part-R-CNN [Zhang et al. 2014]

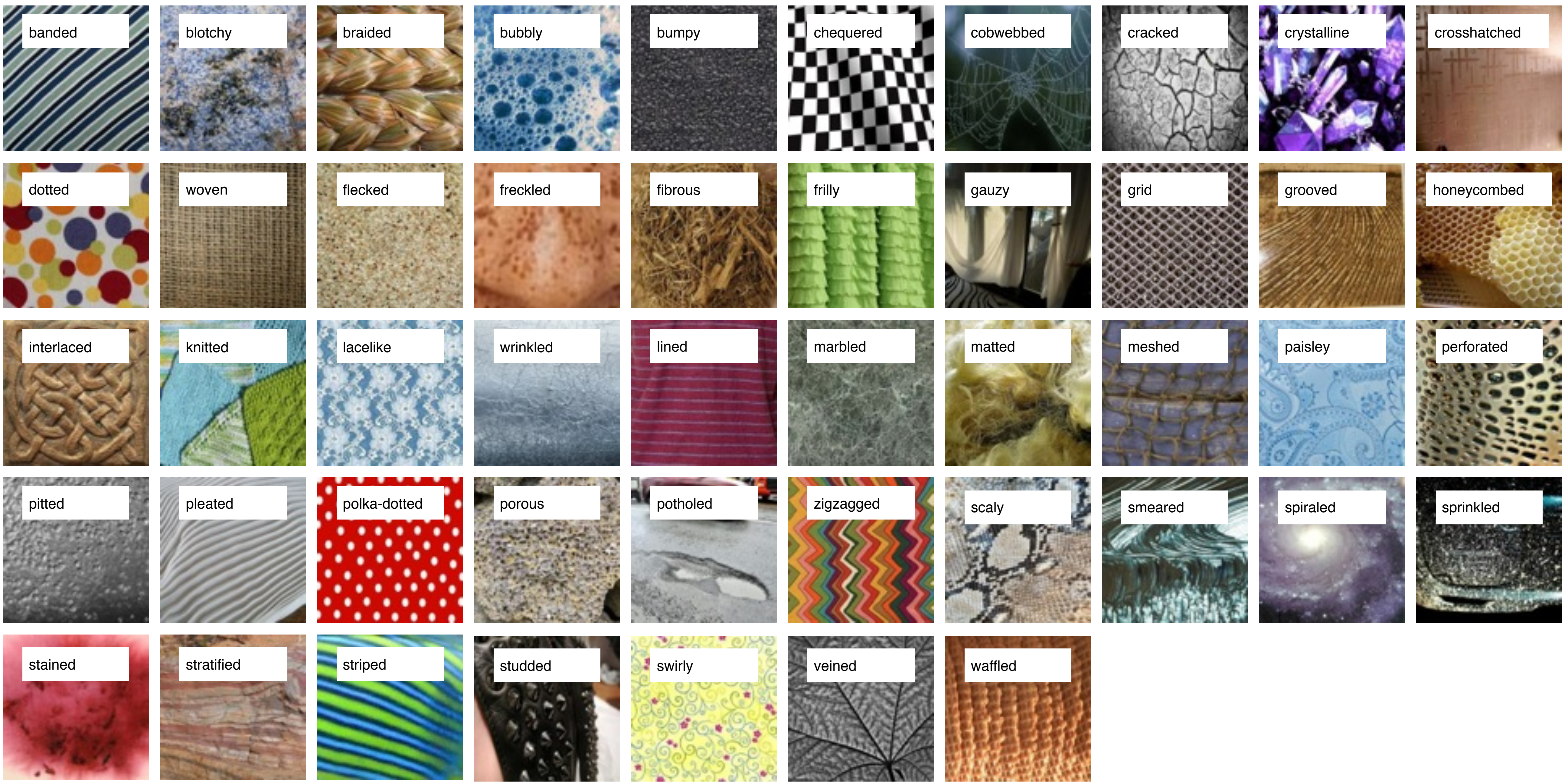
MIT 67 scene classification

- [Razavin et al. 2014]

Beyond objects?

Feature generality

ImageNet pre-trained features achieve **state-of-the-art material recognition** and **texture naming** (but similar to Fisher Vector) [Cimpoi et al. 2014]



[Describable textures dataset]

Feature sharing

The **same** CNN-based **representations** apply to **different tasks**

- ImageNet classification
- object category classification & detection
- scene recognition
- fine-grained bird classification
- texture recognition

Not dissimilar from SIFT, HOG

Can we learn features jointly from multiple tasks?

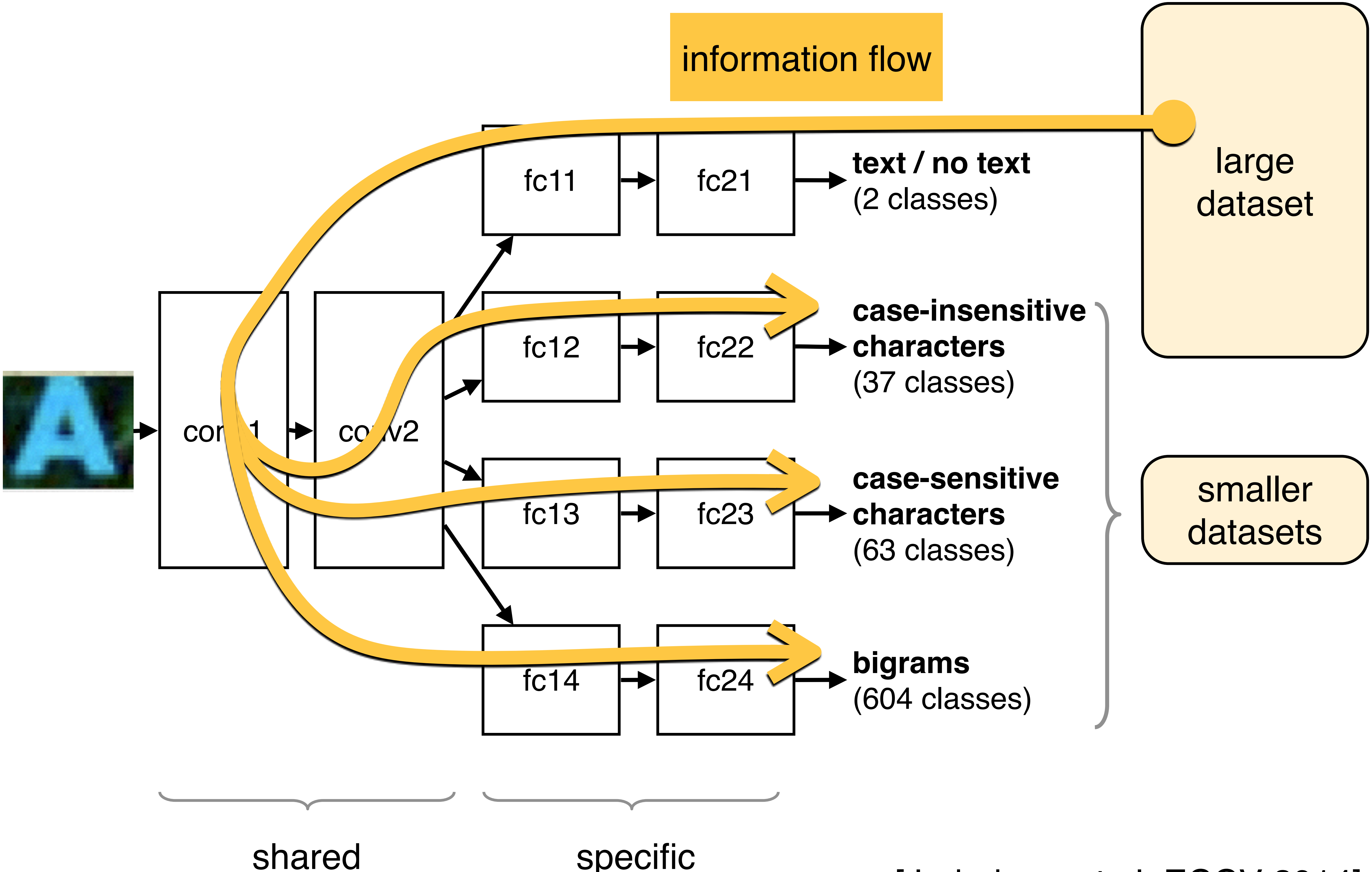
See e.g. [Bengio Courville Vincent PAMI 2013] for a great overview

Example: text spotting

Automatically detect & recognise text in natural images

Also known as PhotoOCR

Tasks are learned synergistically

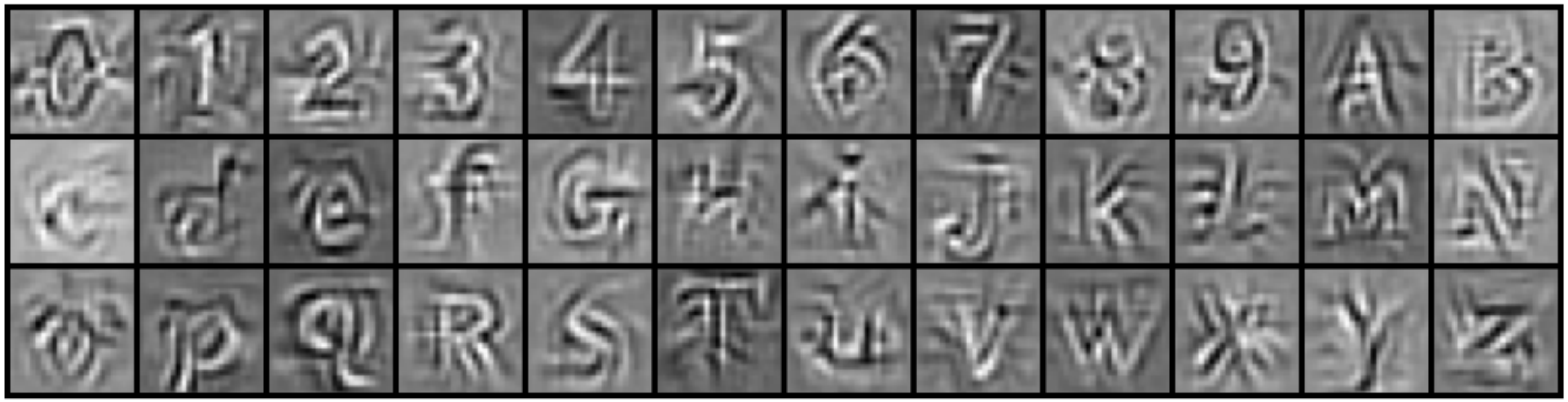


[Jaderberg et al. ECCV 2014]

What have we learned?

Diagnose the model

Use the “deep dreams” trick to visualise the learned character classes:



[Jaderberg et al. ECCV 2014]

Do it yourself

Software

- **CUDA-Convnet 1 & 2**
  https://code.google.com/p/cuda-convnet/
- **Overfeat / Torch** [Lua]
  http://cilvr.nyu.edu/doku.php?id=code:start
- **Berkeley Caffe** [Python]
  http://caffe.berkeleyvision.org
- **Theano** [Python]
  http://deeplearning.net/software/theano/
- **LibCCV**
  http://libccv.org

Pre-trained models

- Return of the Devil in the Details
  http://www.robots.ox.ac.uk/~vgg/research/deep_eval/
- Caffé reference models
  http://caffe.berkeleyvision.org/getting_pretrained_models.html

MatConvNet

http://www.vlfeat.org/matconvnet

A MATLAB toolbox for CNNs

- Similar in spirit to VLFeat.org
- Expose the fundamental computational blocks as MATLAB functions
- Designed for quick experimentation in this environment

Flexibility

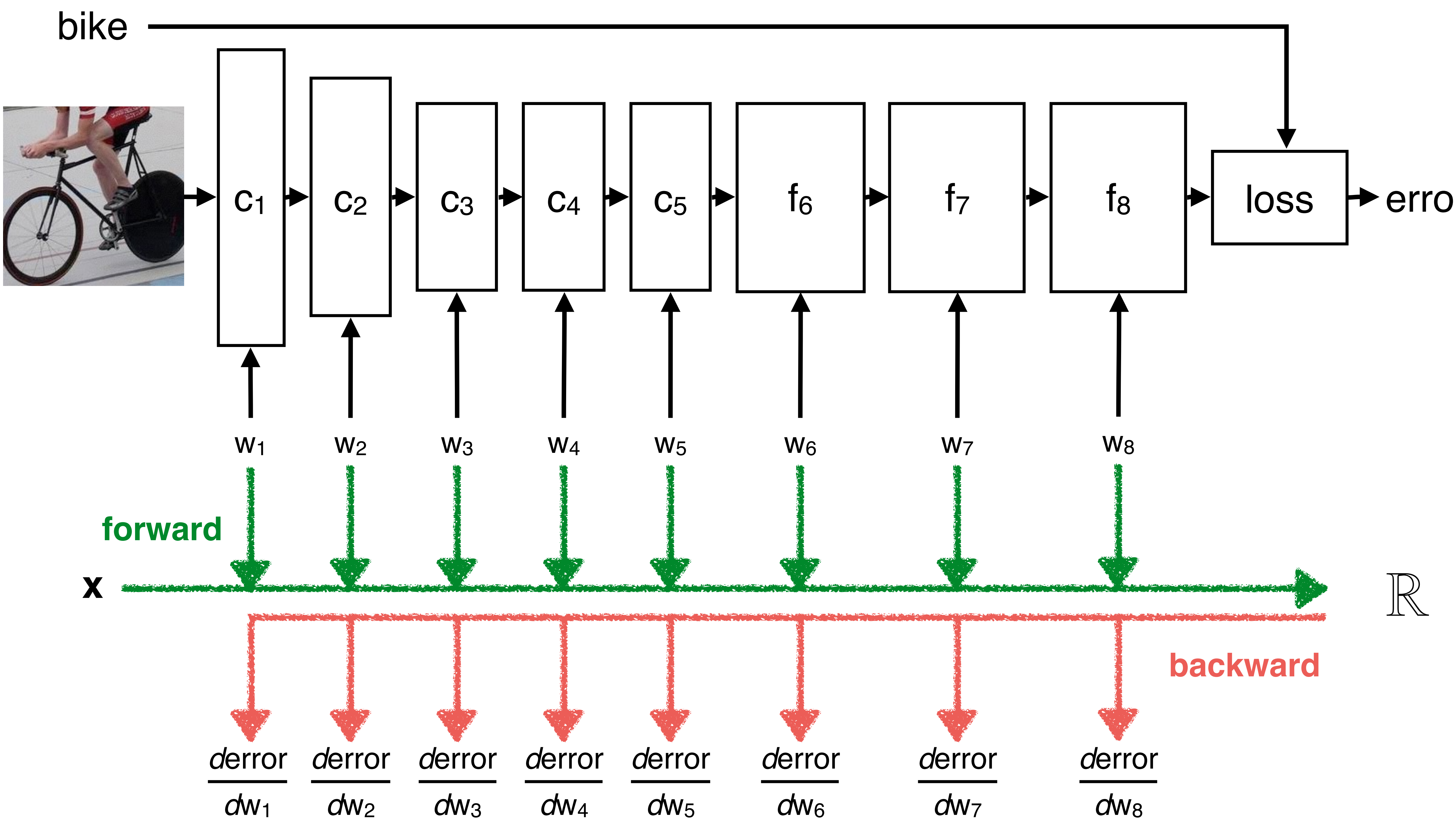
- Can run Caffe models
- Pre-trained models form Caffe and VGG

Efficiency

- Computations are inspired by Berkeley Caffe
- *Native MATLAB GPU support*
- 60-70% training speed of Caffe (and improving)

Backpropagation

Compute derivatives using the chain rule

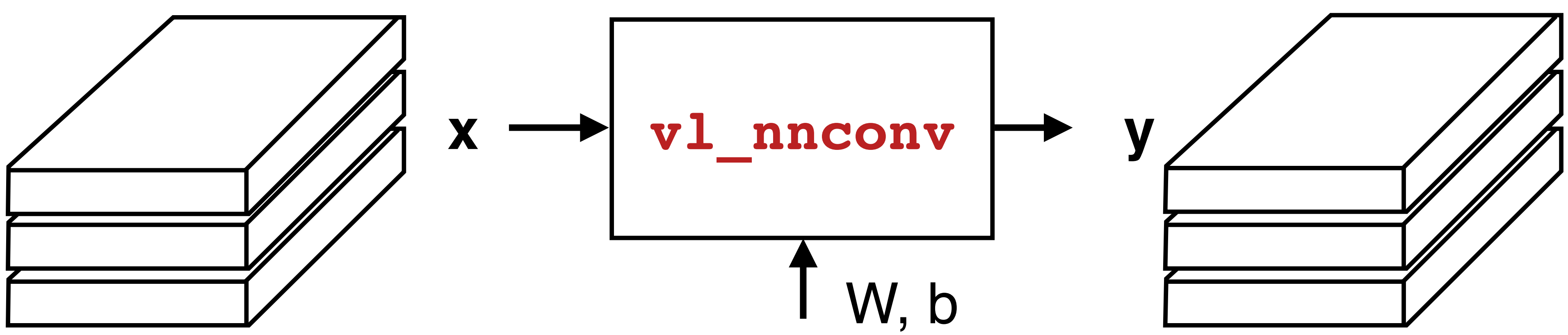


MatConvNet

A CNN toolbox for MATLAB

Forward computation

- operates on a *stack of images*
- each image has *d* feature channels



Available blocks

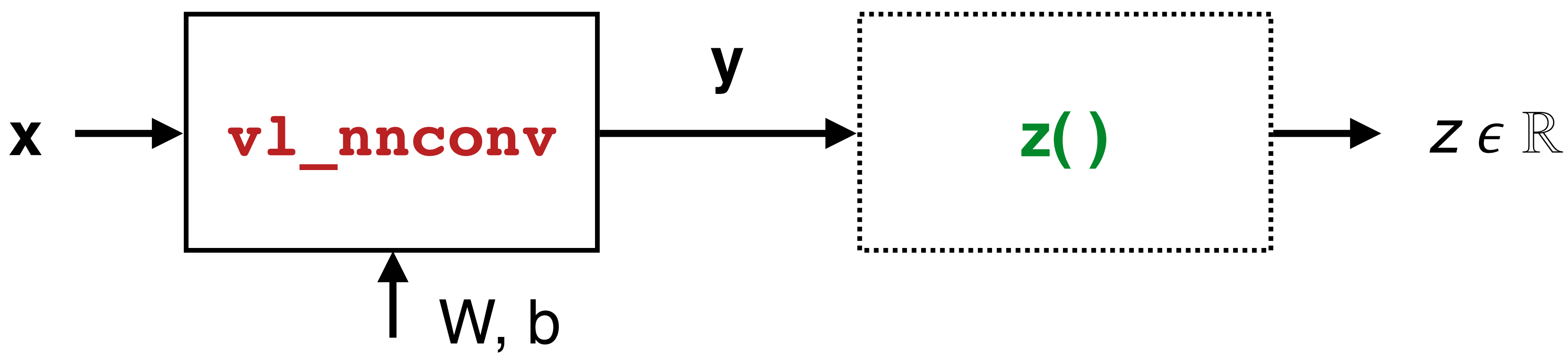
- convolution, pooling, normalization, loss, ReLU, softmax, dropout
- easily extensible (often directly in MATLAB code)

MatConvNet

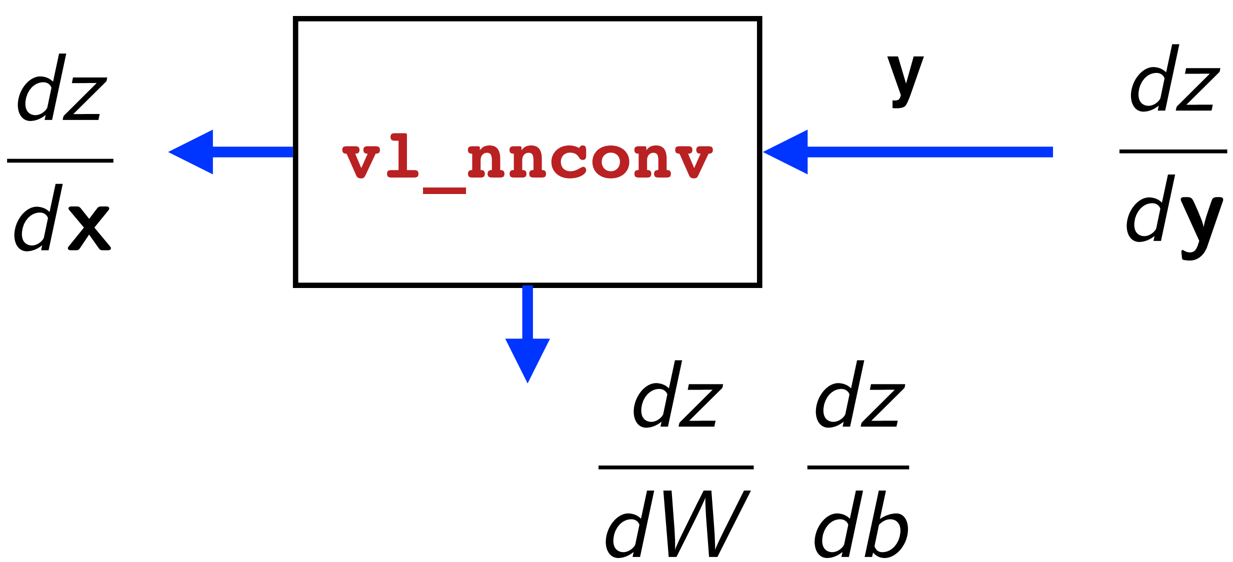
A CNN toolbox for MATLAB

Backward computation

- require network derivatives from block downstream



- chain rule



MatConvNet

Example

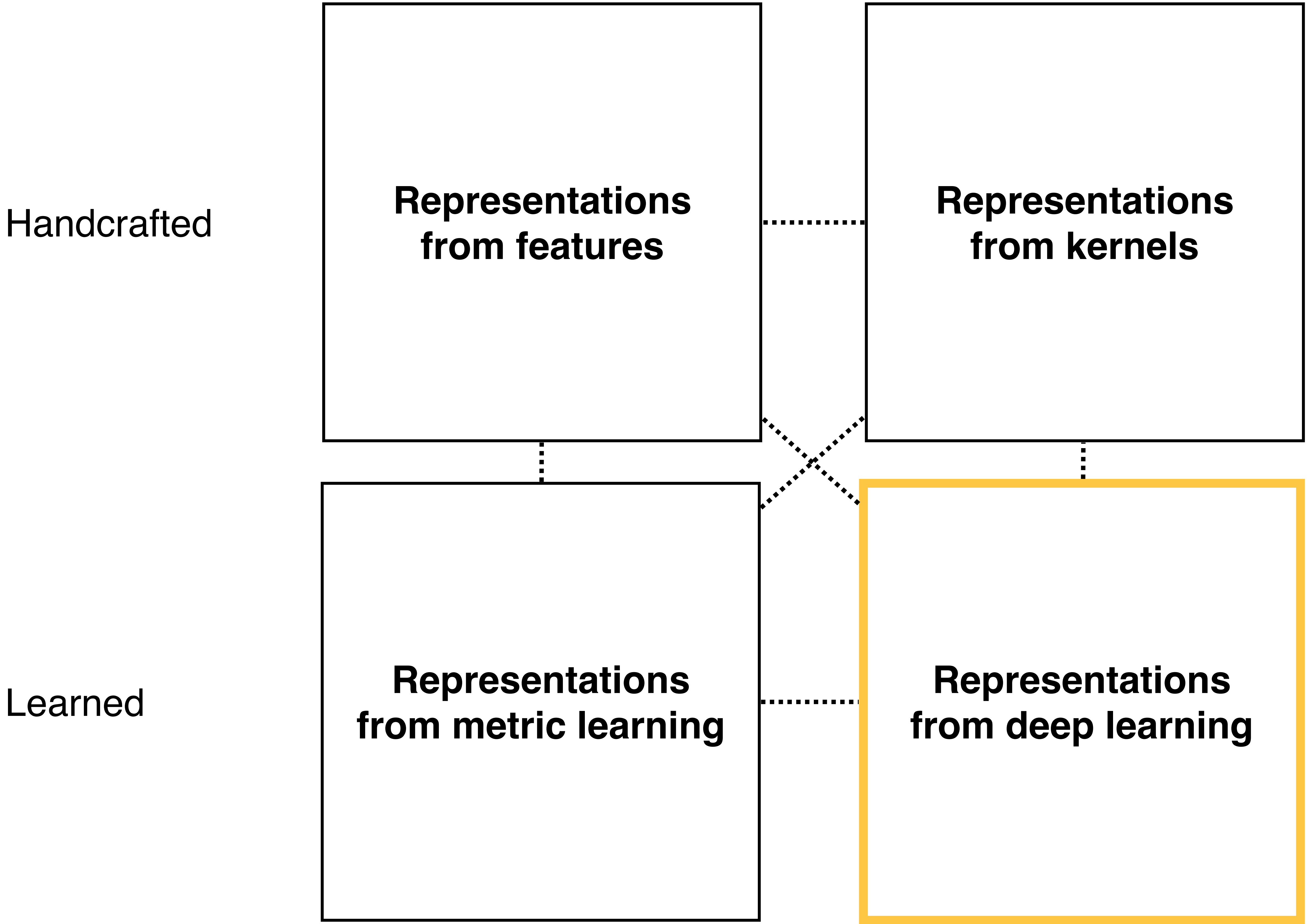
```
% download a pre-trained CNN from the web
urlwrite(...
    'http://www.vlfeat.org/matconvnet/models/imagenet-vgg-f.mat', ...
    'imagenet-vgg-f.mat') ;
net = load('imagenet-vgg-f.mat') ;

% obtain and preprocess an image
im = imread('peppers.png') ;
im_ = single(im) ;
im_ = imresize(im_, net.normalization.imageSize(1:2)) ;
im_ = im_ - net.normalization.averageImage ;

% run the CNN
res = vl_simplenn(net, im_) ;

% show the classification result
scores = squeeze(gather(res(end).x)) ;
[bestScore, best] = max(scores) ;
figure(1) ; clf ; imagesc(im) ;
title(sprintf('%s (%d), score %.3f',...
   net.classes.description{best}, best, bestScore)) ;
```

Wrapping up


Handcrafted
Learned
Representations from features
Representations from kernels
Representations from metric learning
Representations from deep learning

Wrapping Up

Represent & predict

- A good representation captures a useful notion of similarity
- Works as a prior in prediction

Representations from hand-crafted features

- HOG, BoVW, VLAD, Fisher Vectors

Representations from kernels

- Derive implicit and explicit representation from a concept of similarity

Representations from metric learning

- Compare & compress with metric learning

Representations from deep learning

- Visualisation, transfer learning, feature sharing
- Excellent performance

References

[1] F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In ICML, 2005.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. Speeded-up robust features (SURF). Computer Visionand Image Understanding, 2008.

[3] M. B. Blaschko, R. B. Girshick, J. Kannala, I. Kokkinos, S. Mahendran, S. Maji, S. Mohammed, E. Rahtu, N. Saphra, K. Simonyan, B. Taskar, D. Weiss, and A. Vedaldi. Towards a detailed under- standing of objects and scenes in natural images. Technical report, Johns Hopkins Center For Signal and Language Processing, 2012.

[4] L. Bo and C. Sminchisescu. Efficient match kernels between sets of features for visual recognition. In Proc. NIPS, 2009.

[5] A. Bosch, A. Zisserman, and X. Mun˜oz. Scene classification via pLSA. In Proc. ECCV, 2006.

[6] A. Bosch, A. Zisserman, and X. Mun˜oz. Image classification using random forests and ferns. In Proc.ICCV, 2007.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001.

[8] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. PAMI, 24(5), 2002.

[9] G. Csurka, C. R. Dance, L. Dan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In Proc. ECCV Workshop on Stat. Learn. in Comp. Vision, 2004.

[10] C. Elkan. Using the triangle inequality to accelerate k-means. In Proc. ICML, 2003.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9, 2008.

[12] L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of objectcategories. In Proc. ICCV, 2003.

[13] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. Technical report,Cornell University, 2004.

[14] B. J. Frey and D. Dueck. Clustering by passing messages between data points. Science, 315, 2007.

[15] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software, 1977.

[16] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In Proc. ECCV, 2008.

[17] T. Hastie. Support vector machines, kernel logistic regression, and boosting. Lecture Slides, 2003.

[18] T. Joachims. Making large-scale support vector machine learning practical. In Advances in kernel methods: support vector learning, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.[19] T. Joachims. Training linear SVMs in linear time. In Proc. KDD, 2006.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neuralnetworks. In Proc. NIPS, 2012.

[21] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bag of features: Spatial pyramid matching for recognising natural scene categories. In Proc. CVPR, 2006.

[22] B. Leibe, K. Micolajckzyk, and B. Schiele. Efficient clustering and matching for object class recognition.In Proc. BMVC, 2006.

[23] D. G. Lowe. Object recognition from local scale-invariant features. In Proc. ICCV, 1999.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 2(60):91–110, 2004.

[25] S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In Proc. ICCV, 2009.

[26] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In Proc. BMVC, 2002.

[27] D. Nist´er and H. Stew´enius. Scalable recognition with a vocabulary tree. In Proc. CVPR, 2006.

[28] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In Proc. CVPR, 2014.

[29] O. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face descriptor. In Proc. CVPR, 2014.

[30] M. Paulin, J. Revaud, Z. Harchaoui, C. Schidm, and F. Perronnin. Transformation pursuit in imageclassification. In Proc. CVPR, 2014.

[31] F. Perronnin, J. S´anchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In Proc. CVPR, 2010.

[32] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In Proc. CVPR, 2007.

[33] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In Proc. NIPS, 2007.

[34] B. Sch¨olkopf. The kernel trick for distances. Proc. NIPS, 2001.

[35] B. Sch¨olkopf and A. Smola. Learning with Kernels, chapter Robust Estimators, pages 75 – 83. MIT Press, 2002.

[36] B. Sch¨olkopf and A. J. Smola. Learning with Kernels. MIT Press, 2002.

[37] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal Estimated sub-GrAdient
SOlver for SVM. MBP, 2010.

[38] J. Shawe-Taylor and N. Cristianini. Support Vector Machines and other kernel-based learning methods. Cambridge University Press, 2000.

[39] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In Proc. BMVC, 2013.

[40] K. Simonyan, A. Vedaldi, and A. Zisserman. Descriptor learning using convex optimisation. In Proc. ECCV, 2012.

[41] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep fisher networks for large-scale image classification. In Proc. NIPS, 2013.

[42] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep fisher networks and class saliency maps for object classification and localisation. In ILSVRC workshop, 2014.

[43] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Proc. ICLR, 2014.

[44] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Gen. Gpu-based video feature tracking and matching. In Workshop on Edge Computing Using New Commodity Architectures, 2006.

[45] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In Proc. ICCV, 2003.

[46] N. Slonim and N. Tishby. Agglomerative information bottleneck. In Proc. NIPS, 1999.

[47] E. Tola, V. Lepetit, and P. Fua. DAISY: An efficient dense descriptor applied to wide-baseline stereo. PAMI, 2010.

[48] A. Vedaldi, S. Mahendran, S. Tsogkas, S. Maji, R. Girshick, J. Kannala, E. Rahtu, I. Kokkinos, M. B. Blaschko, D. Weiss, B. Taskar, K. Simonyan, N. Saphra, and S. Mohamed. Understanding objects in detail with fine-grained attributes. In Proc. CVPR, 2014.

[49] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In Proc. ECCV, 2008.

[50] G. Wang, Y. zhang, and L. Fei-Fei. Using dependent regions for object categorization in a generative framework. In Proc. CVPR, 2006.

[51] Z. Wang, B. Fan, and F. Wu. Local intensity order pattern for feature description. In Proc. ICCV 2011.

[52] J. Weston, S. Bengio, and N. Usunier. WSABIE: Scaling up to large vocabulary image annotation. In Proc. IJCAI, 2011.

[53] C. K. I. Williams and M. Seeger. Using the Nystr¨om method to speed up kernel machines. In Proc. NIPS, 2001.

[54] Jegou et al. 09 Douze, Schmid, "On the burstiness of visual elements", Proc. CVPR, 2009