# Image representations for large-scale visual recognition

**Andrea Vedaldi**

UNIVERSITY OF OXFORD

# Demo: image search

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

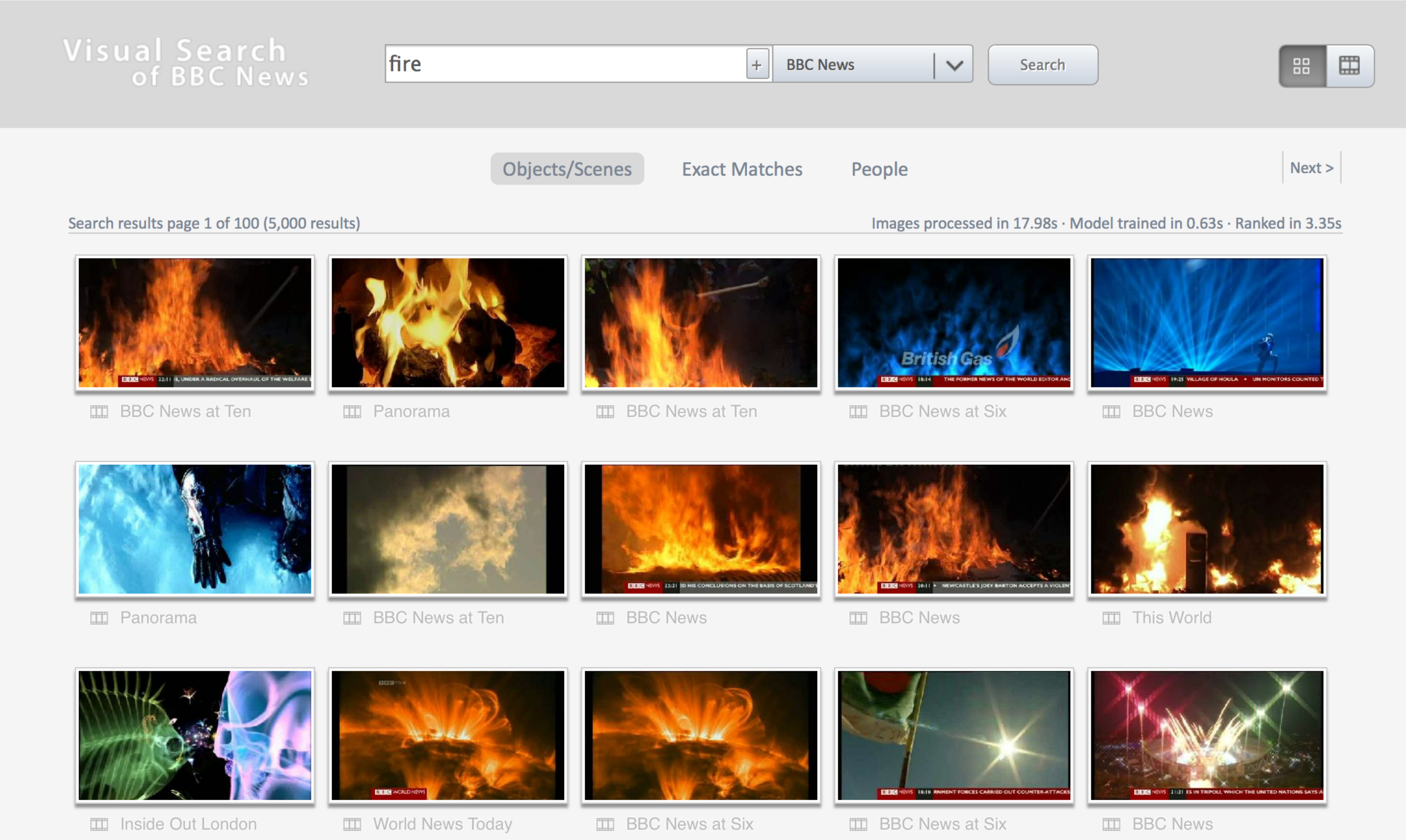


**Rob Cooper from BBC Research & Development explains how their work with Oxford University is opening up new ways to search archive footage.[1]**

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

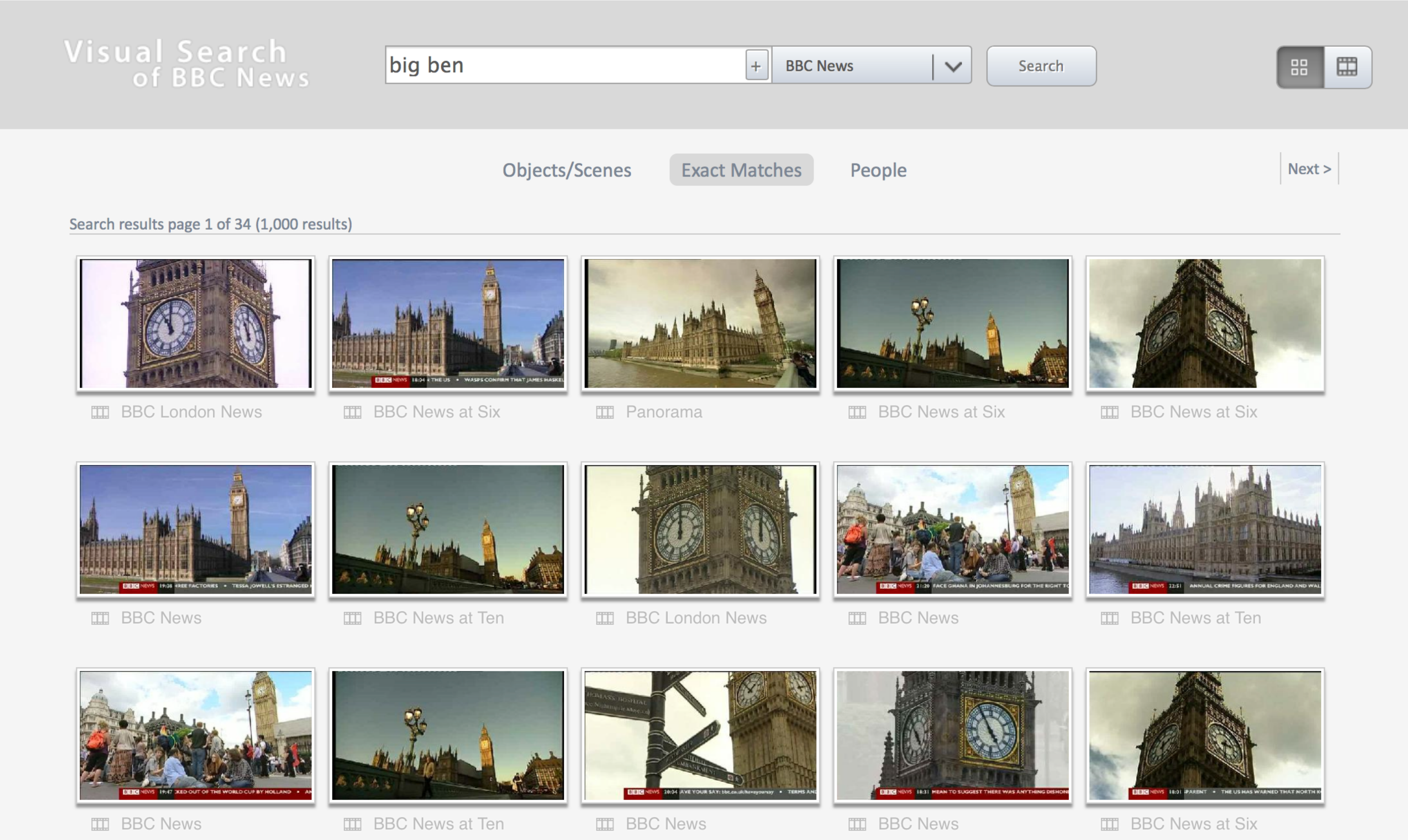
# Searching by type



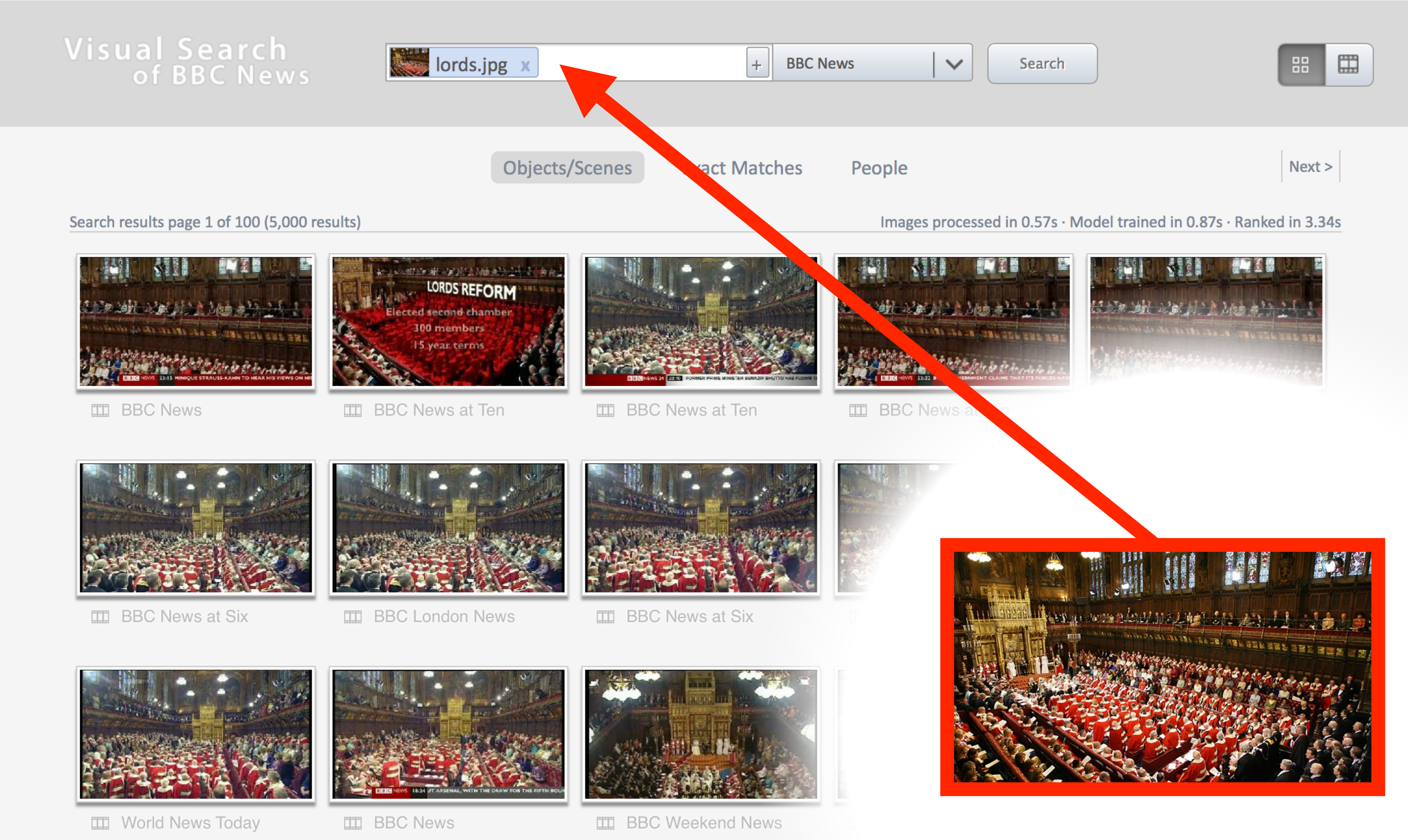


# Searching by instance



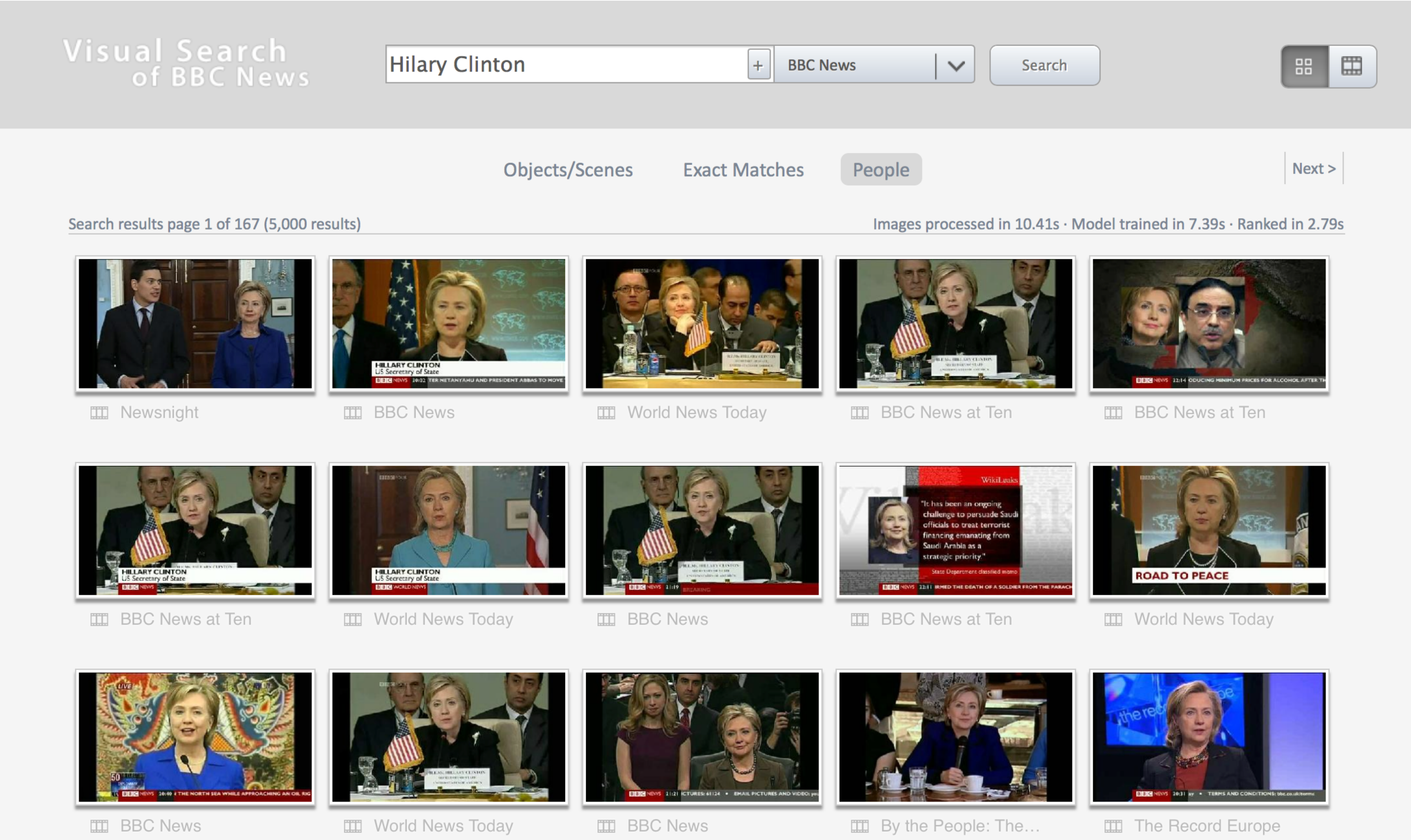


# Search by example



# Searching by identity





# Challenges



| 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 |

- **Understand images**
  - Queries are semantic, images are not
- **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

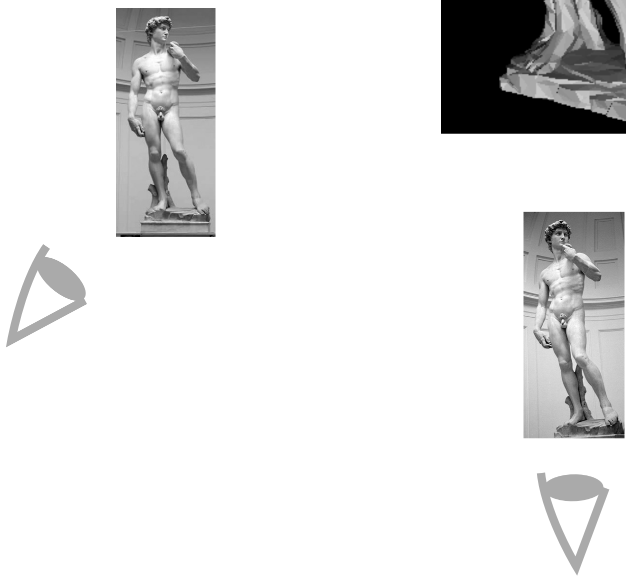
# Understanding objects



- Recognition by reconstruction
  [Vedaldi & Soatto 2005]

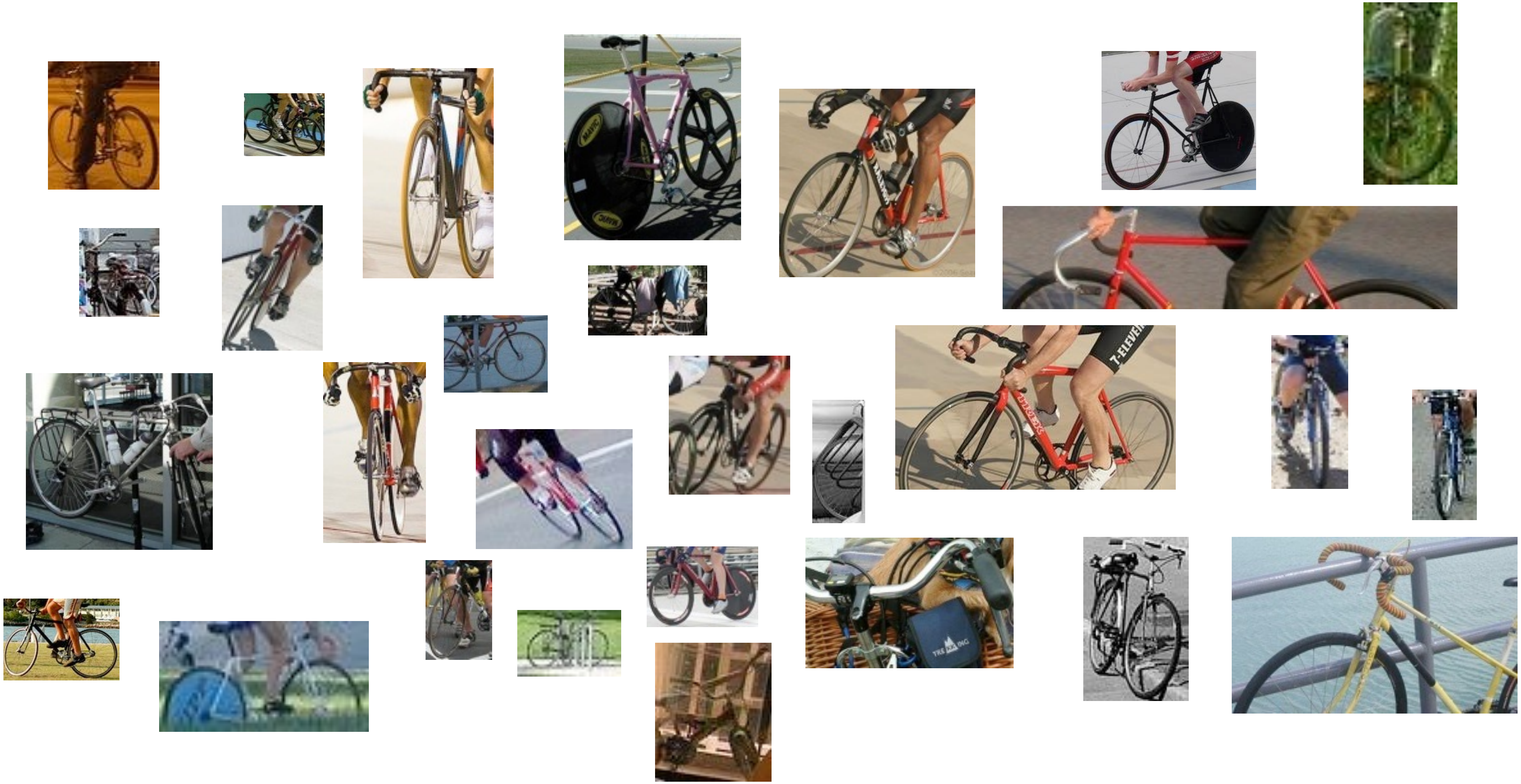
$\stackrel{?}{=}$

Is there a 3D scene that generates both images?



# Image-based object models

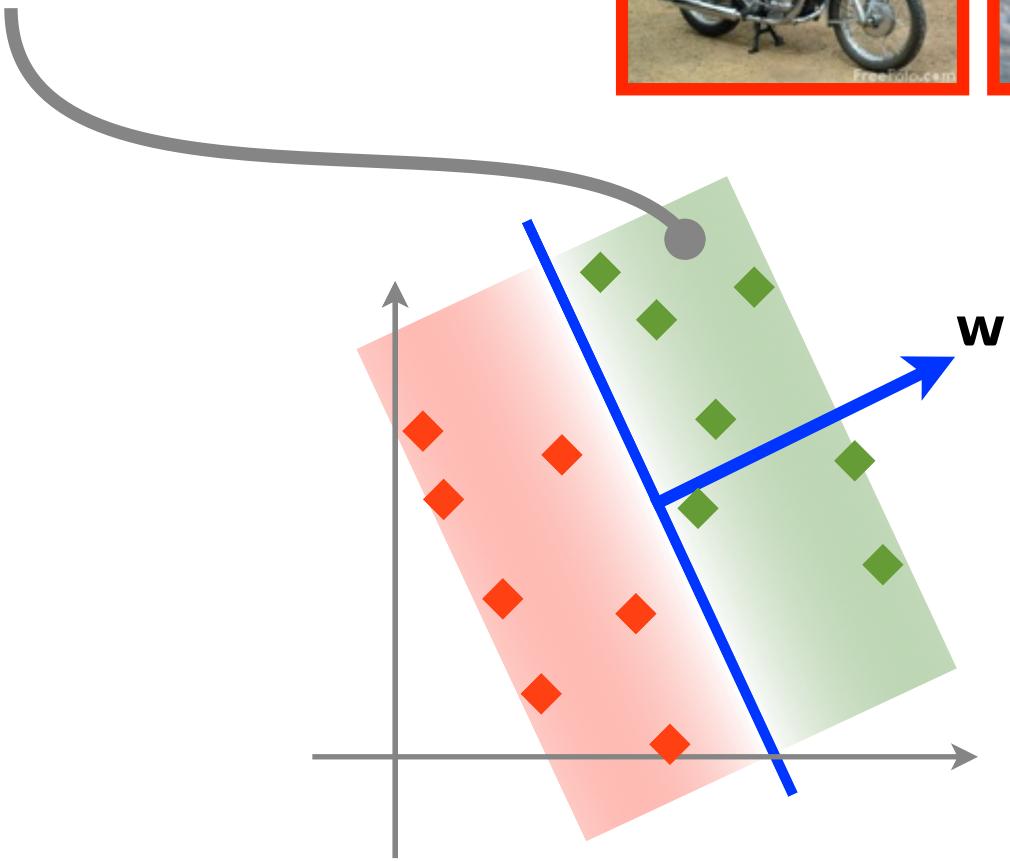
object = **distribution** of **2D patterns**



# Linear predictor

bicycle?

x




**linear predictor**

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

# Encoder

**Using linear predictors on non-vectorial data**

$\mathbf{x}$

$$\mathbf{x} \xrightarrow{\text{encoder } \Phi} \Phi(\mathbf{x}) \in \mathbb{R}^d$$

representation

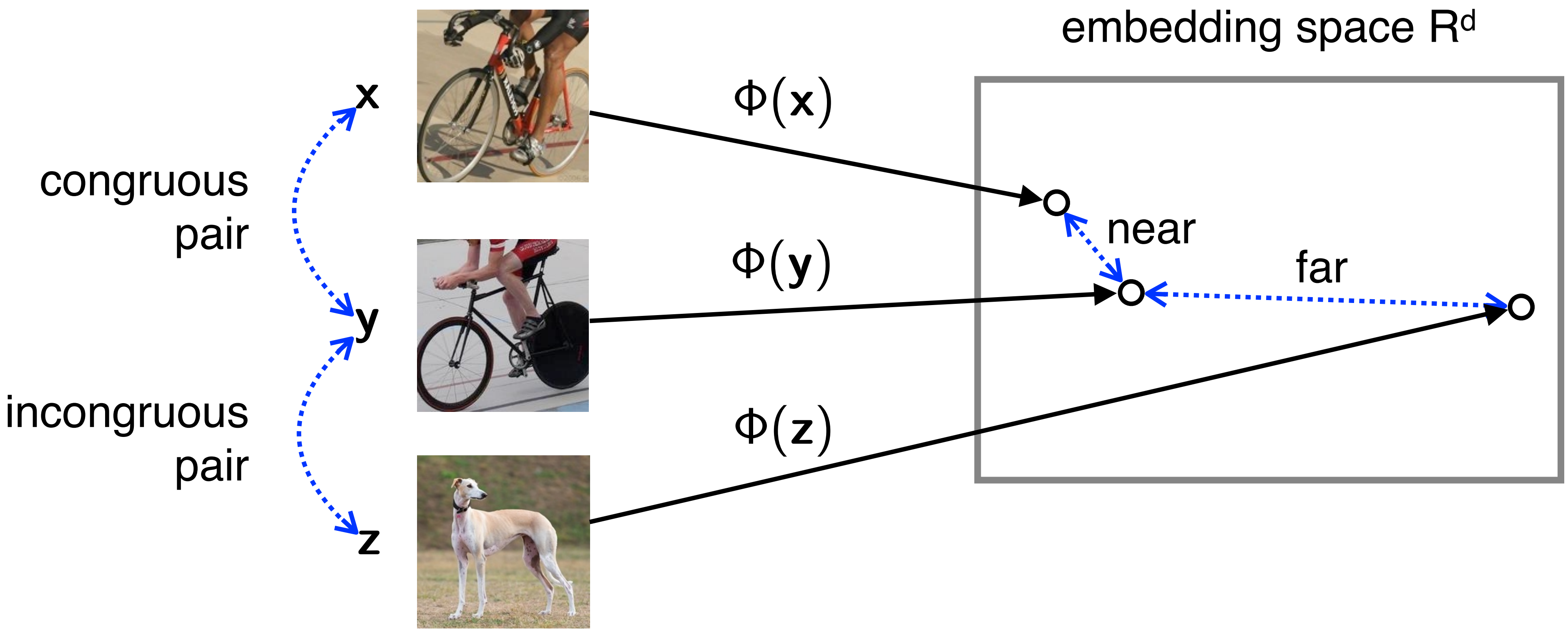
- An **encoder** maps the data into a **vectorial representation**
- Allows linear predictors to be applied to images, text, sound, videos, …

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

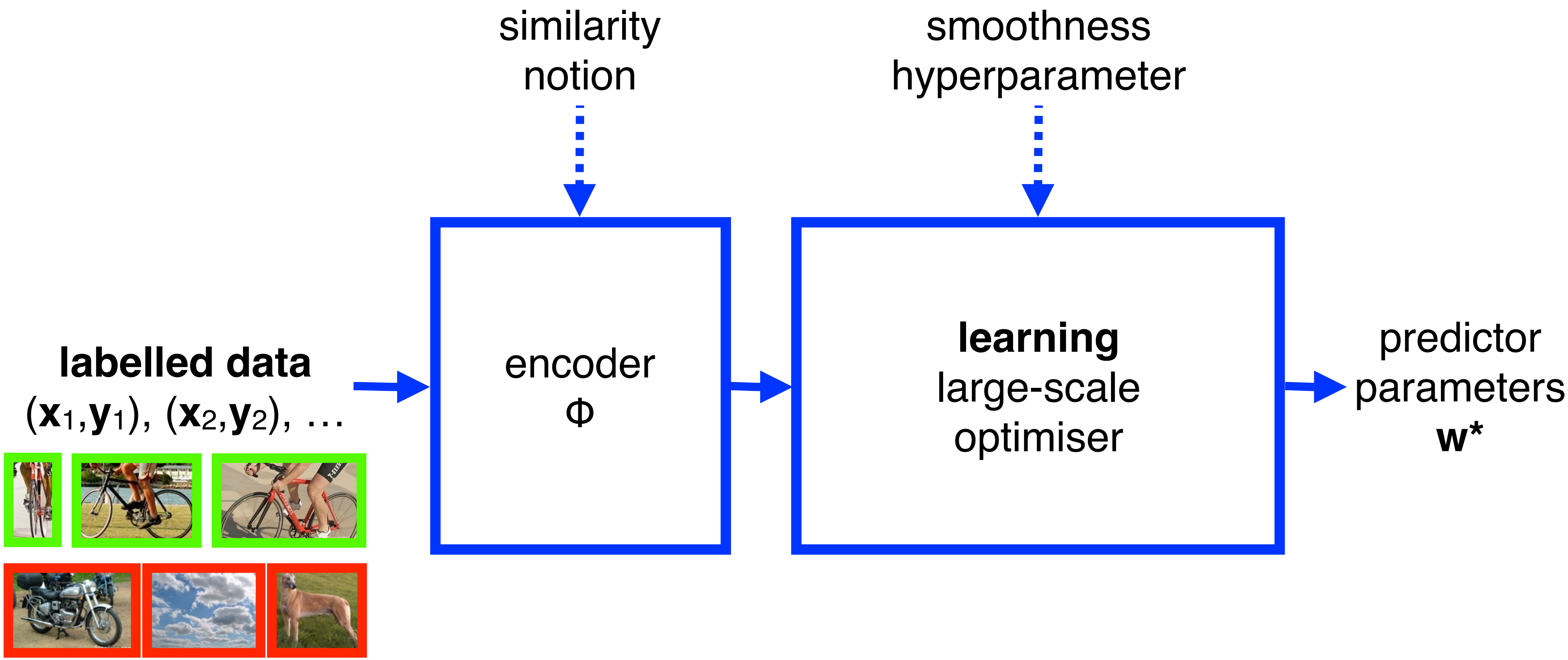
# The goal of an encoder



Semantic similarity $\xrightarrow{\text{encoder}}$ Vector similarity (distance)



# Learning predictors

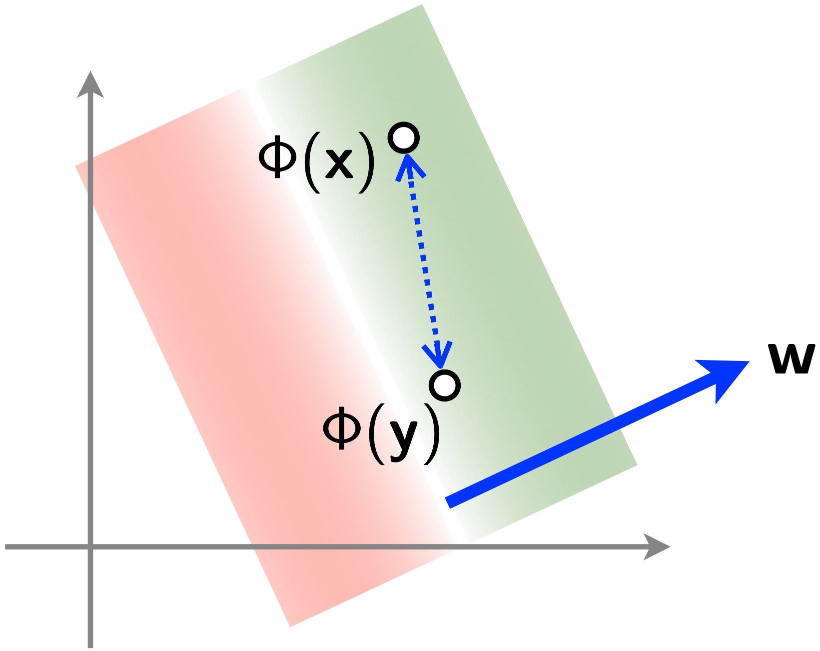



$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}}\, E(\mathbf{w})$$

# Smoothness and generalisation

- Key challenge: **extrapolate the training data**
  - Achieved by **smoothness**
  - I.e. similar vectors receive similar scores

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




**linear predictor**

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

# Support vector machines

**A representative predictor**

$$E(\mathbf{w}) = \underbrace{\lambda \frac{\|\mathbf{w}\|^2}{2}} + \underbrace{\frac{1}{N} \sum_{i=1}^{N} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}}$$

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

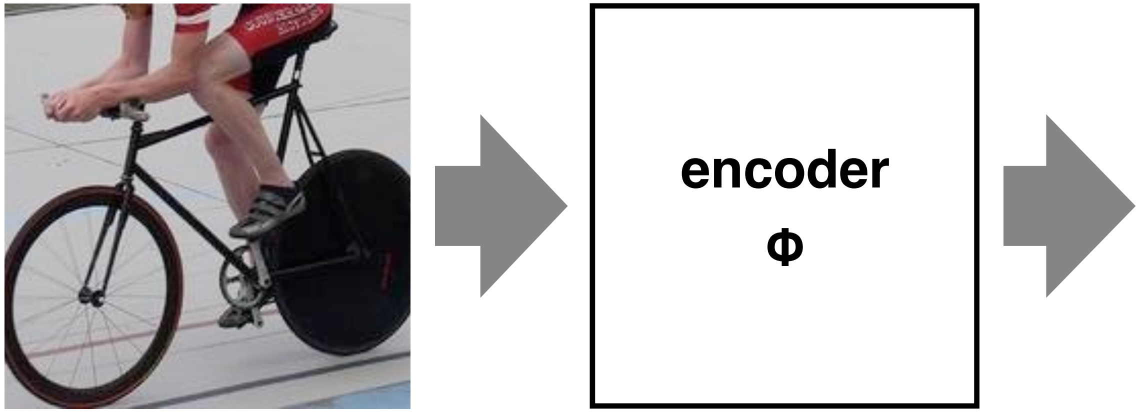
- **Optimisation**
  - Very large convex problem
  - Key insight: **an accurate solution is not required**

- O(N) algorithms exist
  - Stochastic gradient descent, dual coordinate ascent, …
  - Can learn on the fly on thousands or millions of examples

# Good encoders



x

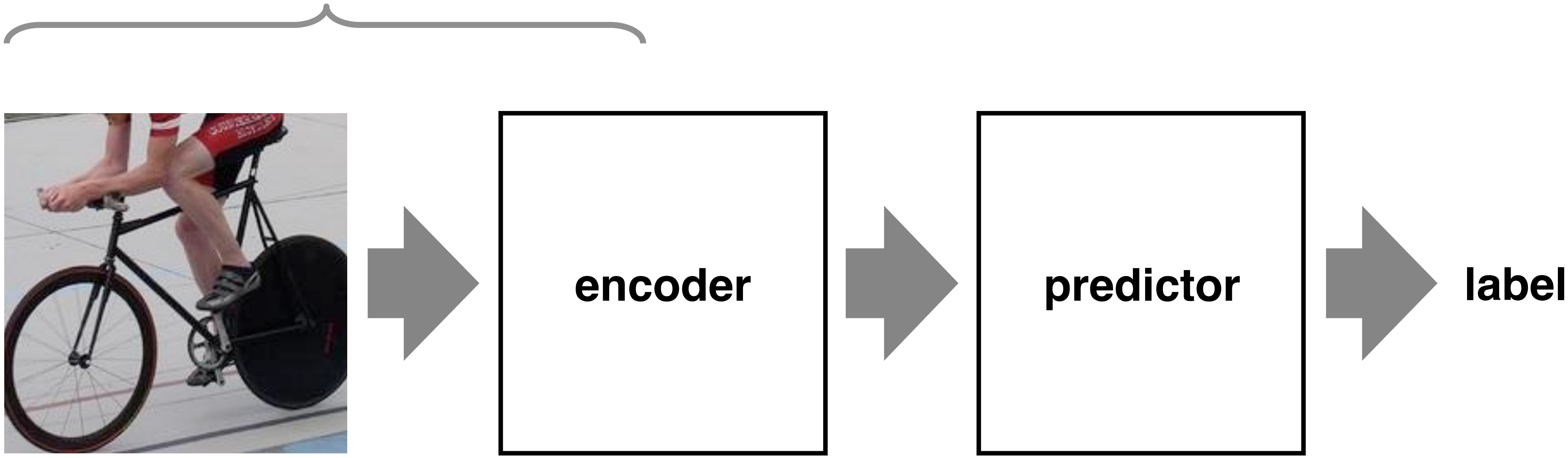


$\Phi(\mathbf{x})$

- **Main desiderata**
  - **Powerful:** meaningful similarity (accurate recognition)
  - **Cheap:** fast to evaluate (can be computed on the fly)
  - **Compact:** small code (takes little RAM, disk, IO)
- **Others**
  - Easy to learn (when applicable)
  - Easy to implement

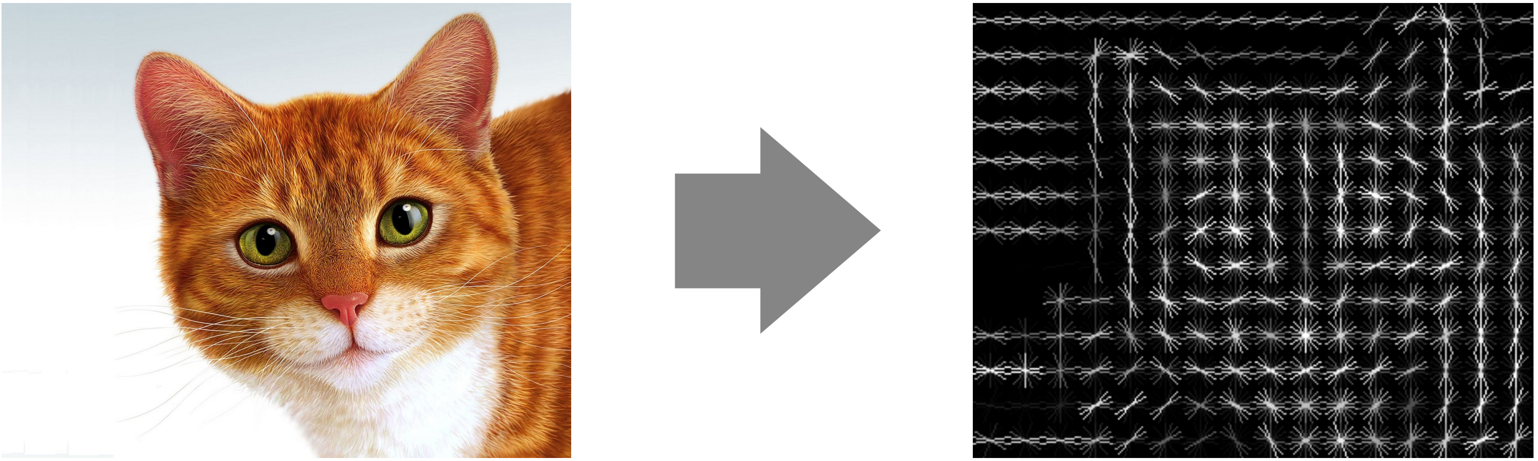
# Contents

Part 1: feature engineering

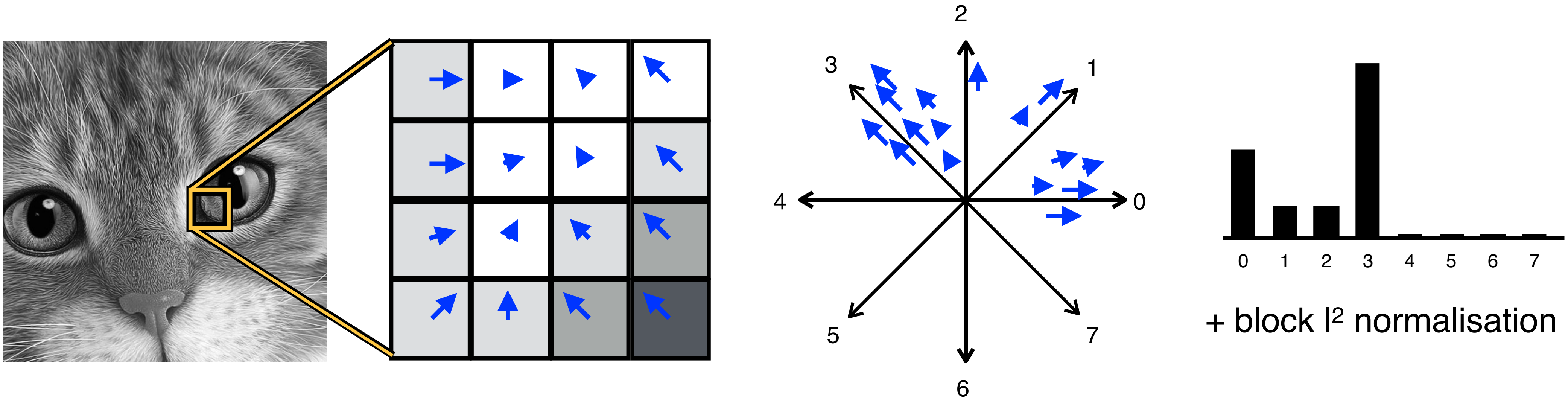


# Histogram of oriented gradients

[Lowe 1999, Dalal & Triggs 2005]



- Captures the local gradient (edge) orientations in the image

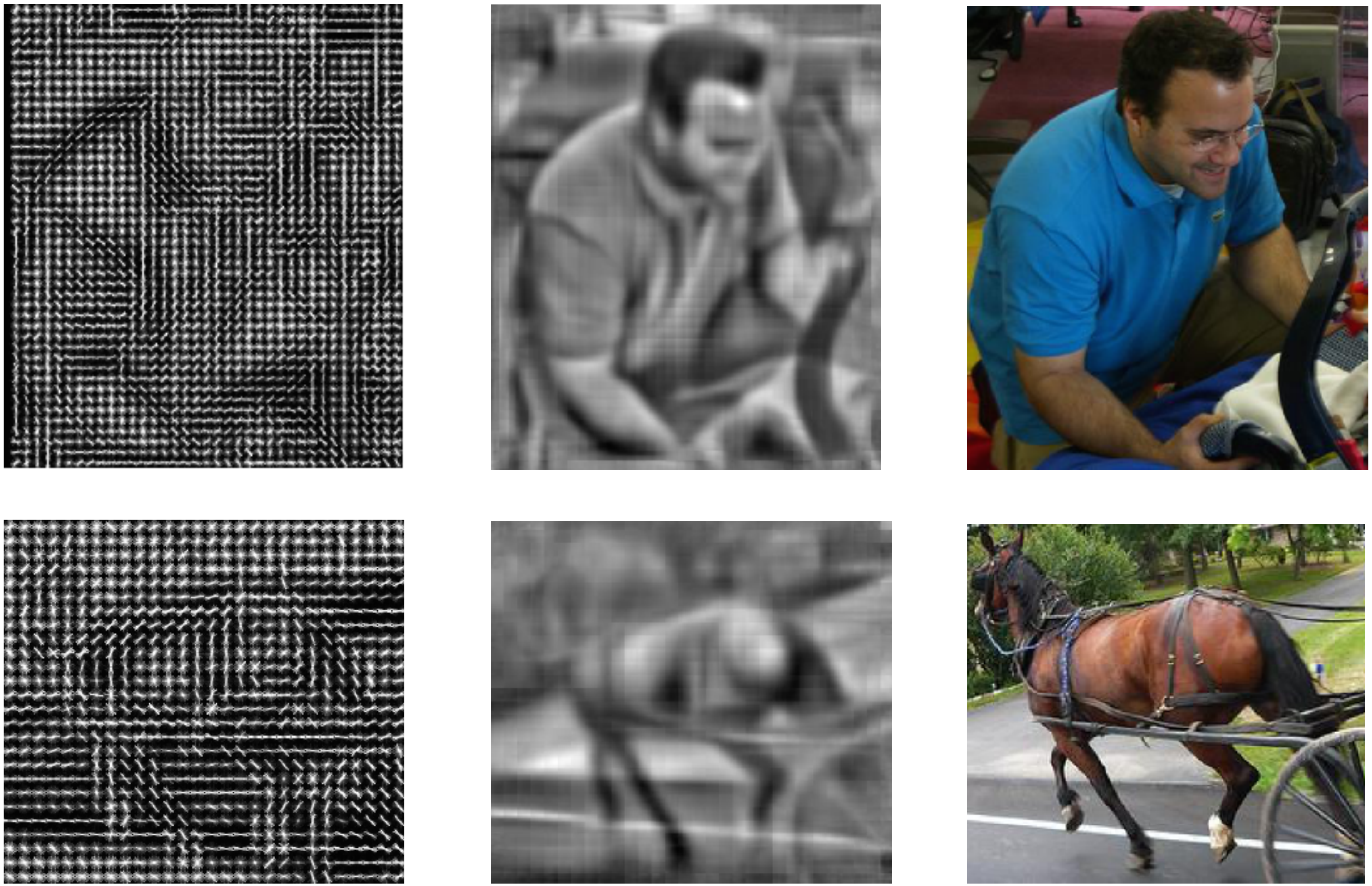



+ block $l^2$ normalisation

# HOG examples



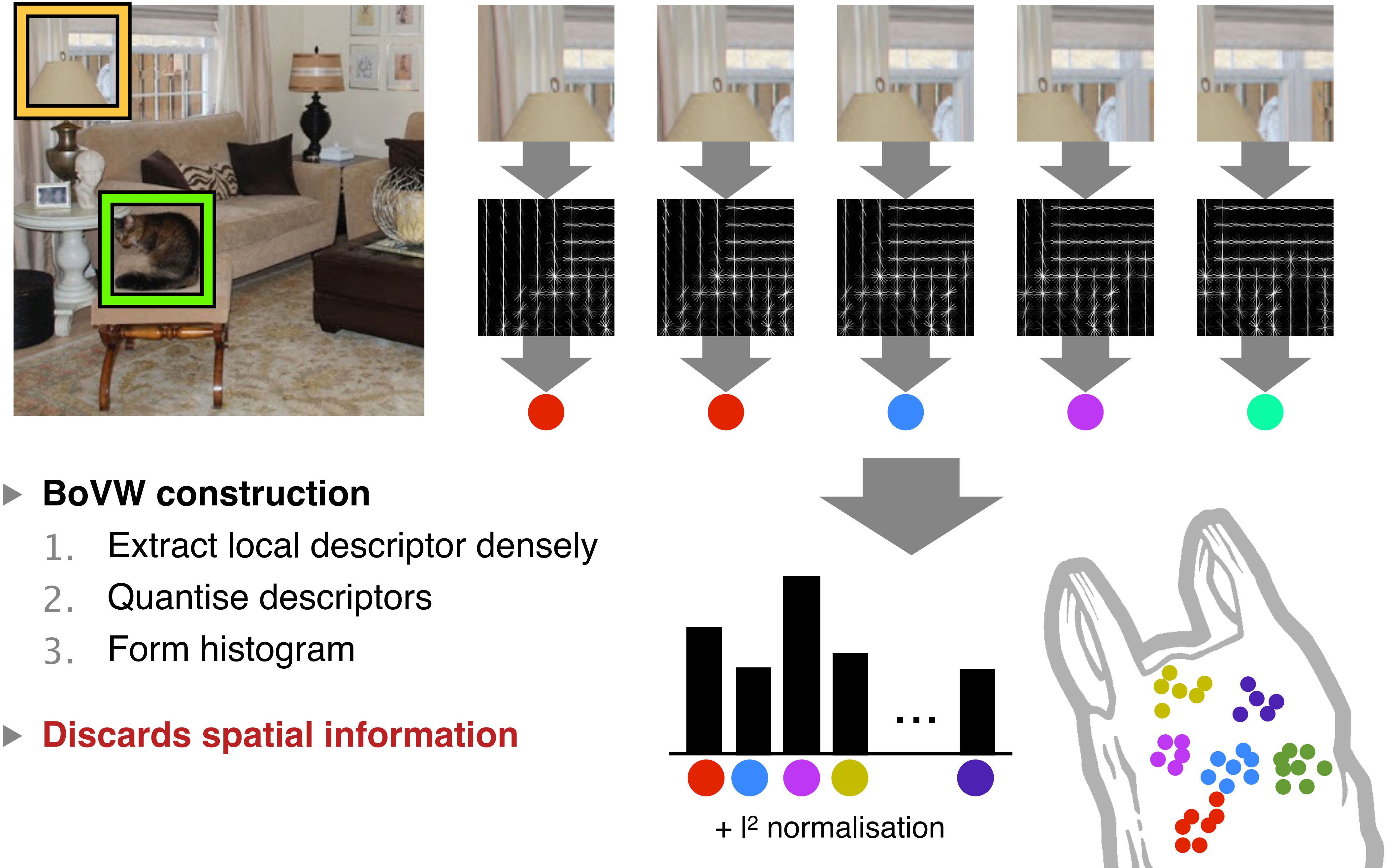
$\mathrm{HOG}(\mathbf{x})$ $\mathrm{HOG}^{-1}(\mathbf{x})$ $\mathbf{x}$



[Vondrick *et al.* 2013]

# Bag of visual words

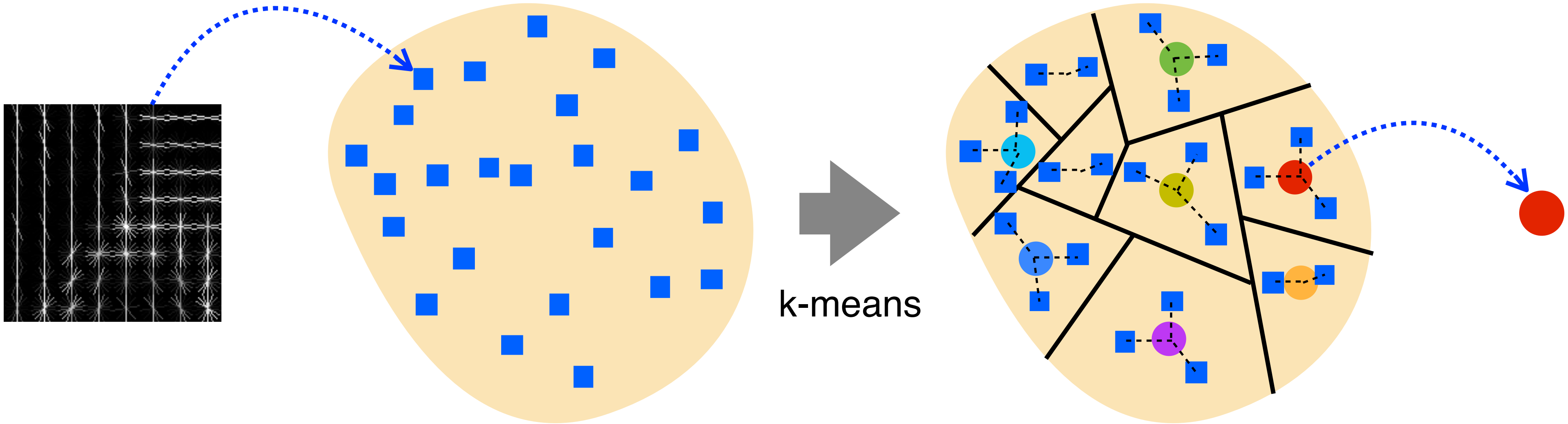
[Sivic & Zisserman 2003, Csurka *et al.* 2004, Nowak *et al.* 2006]

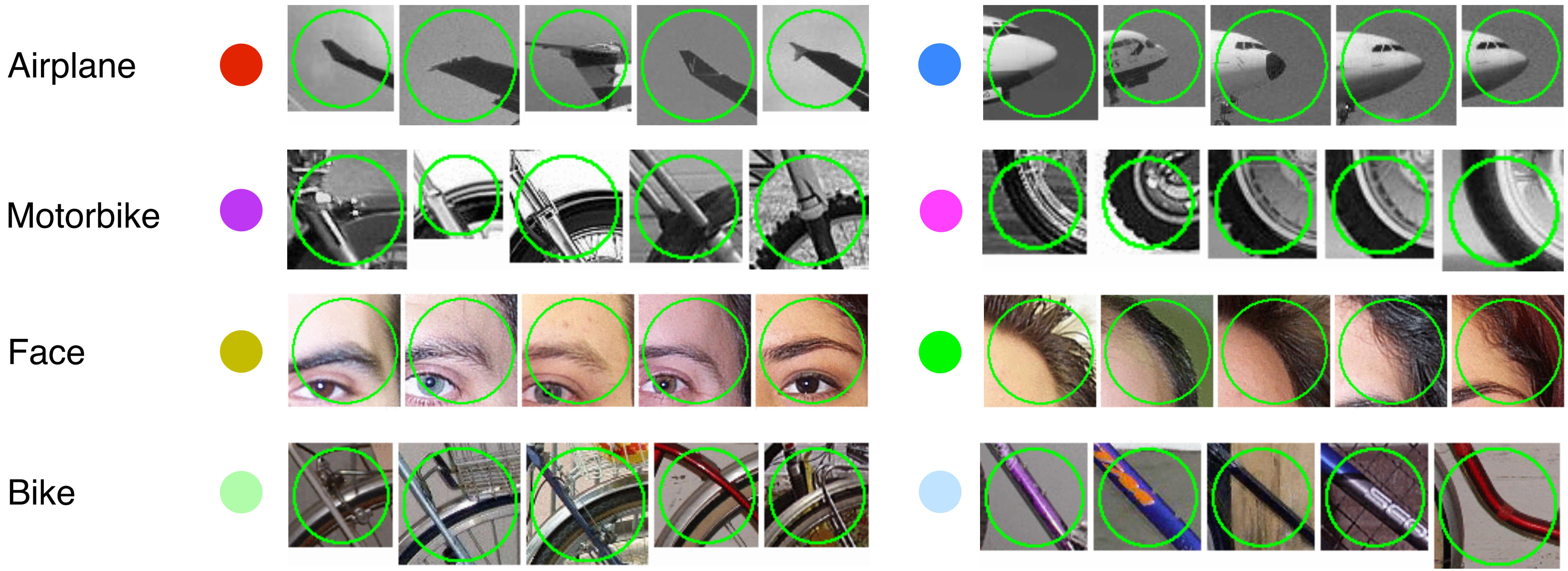



- **BoVW construction**
  1. Extract local descriptor densely
  2. Quantise descriptors
  3. Form histogram
- **Discards spatial information**

# Quantisation

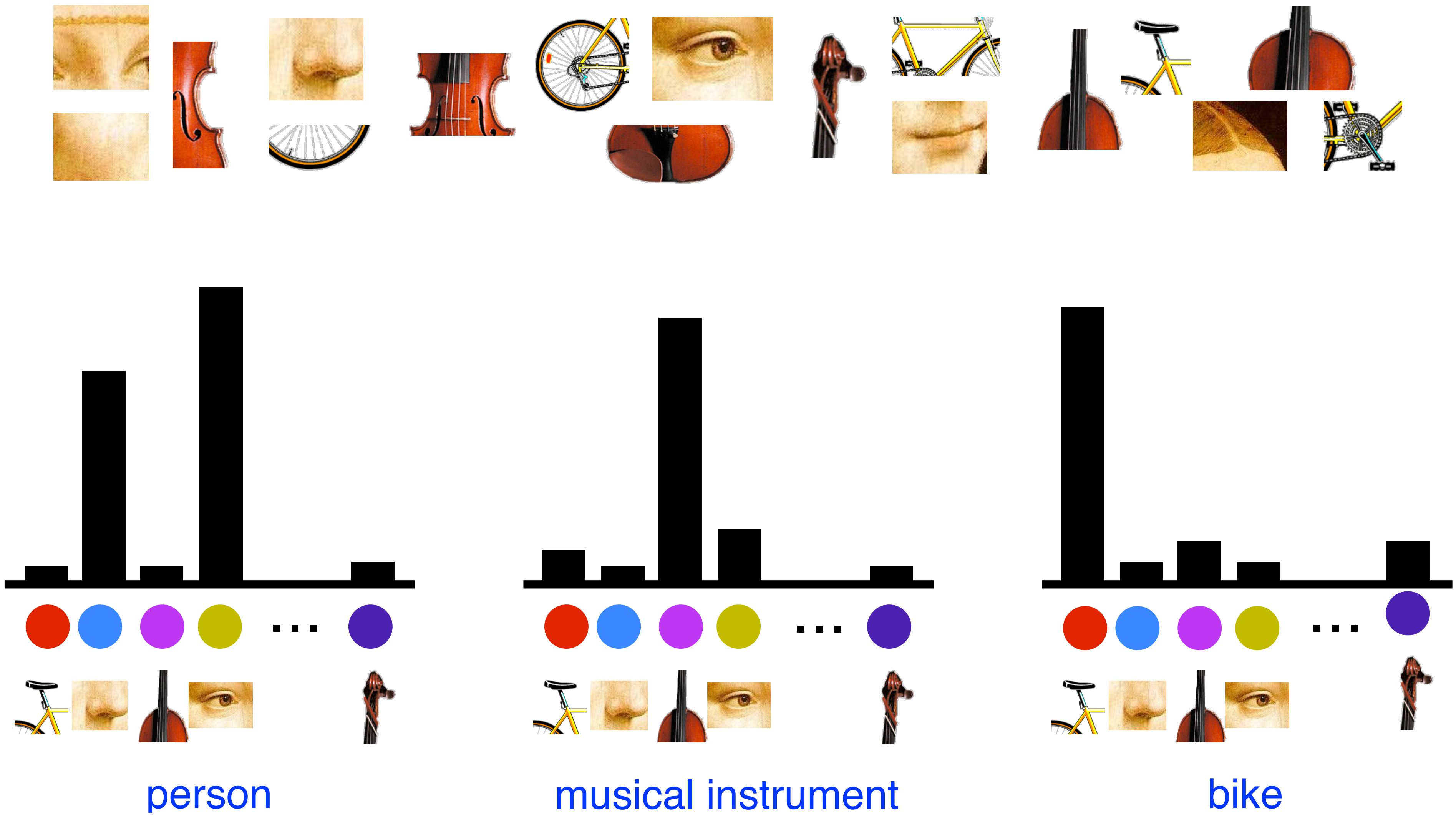




# BoVW intuition

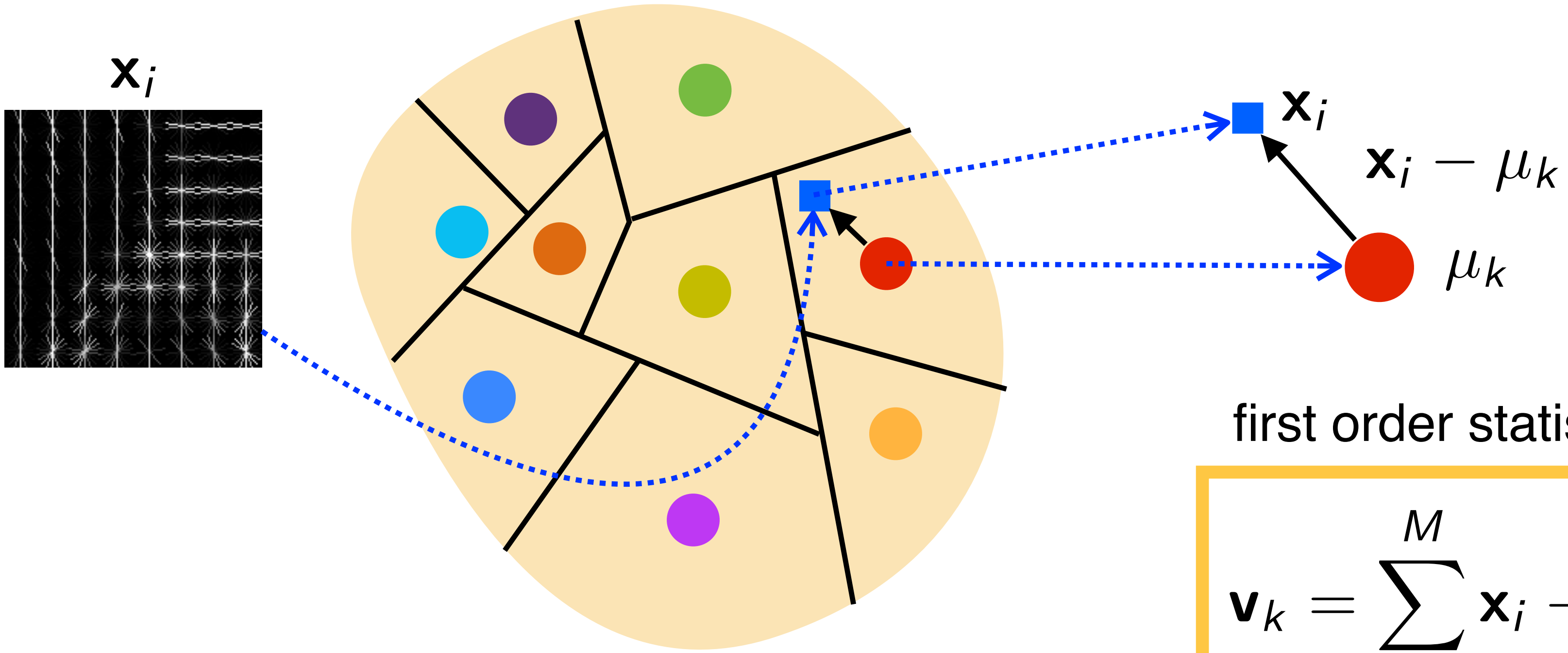
- Discarding spatial information gives lots of invariance
- Visual words represent “iconic” image fragments



# Vector of locally aggregated descriptors (VLAD)



[Jegou *et al.* 2010]



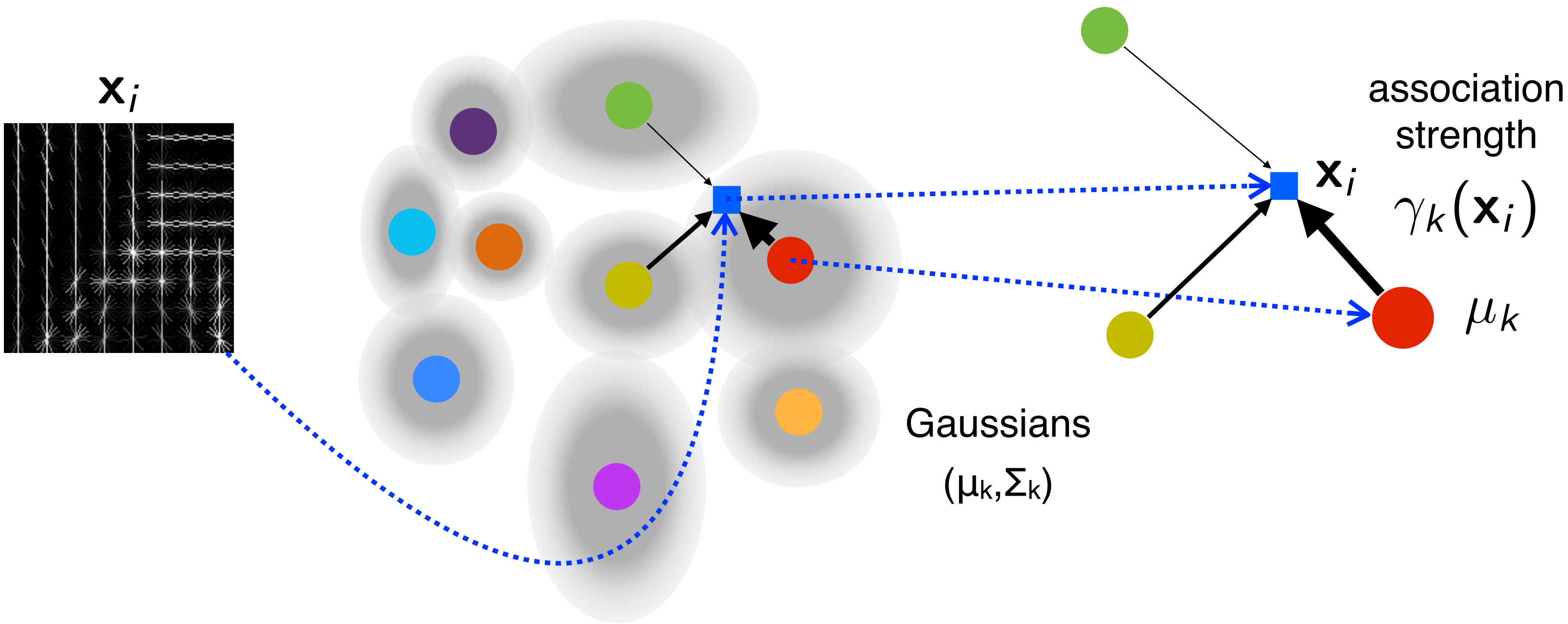

first order statistics

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

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 2012, 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-l² normalisation

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

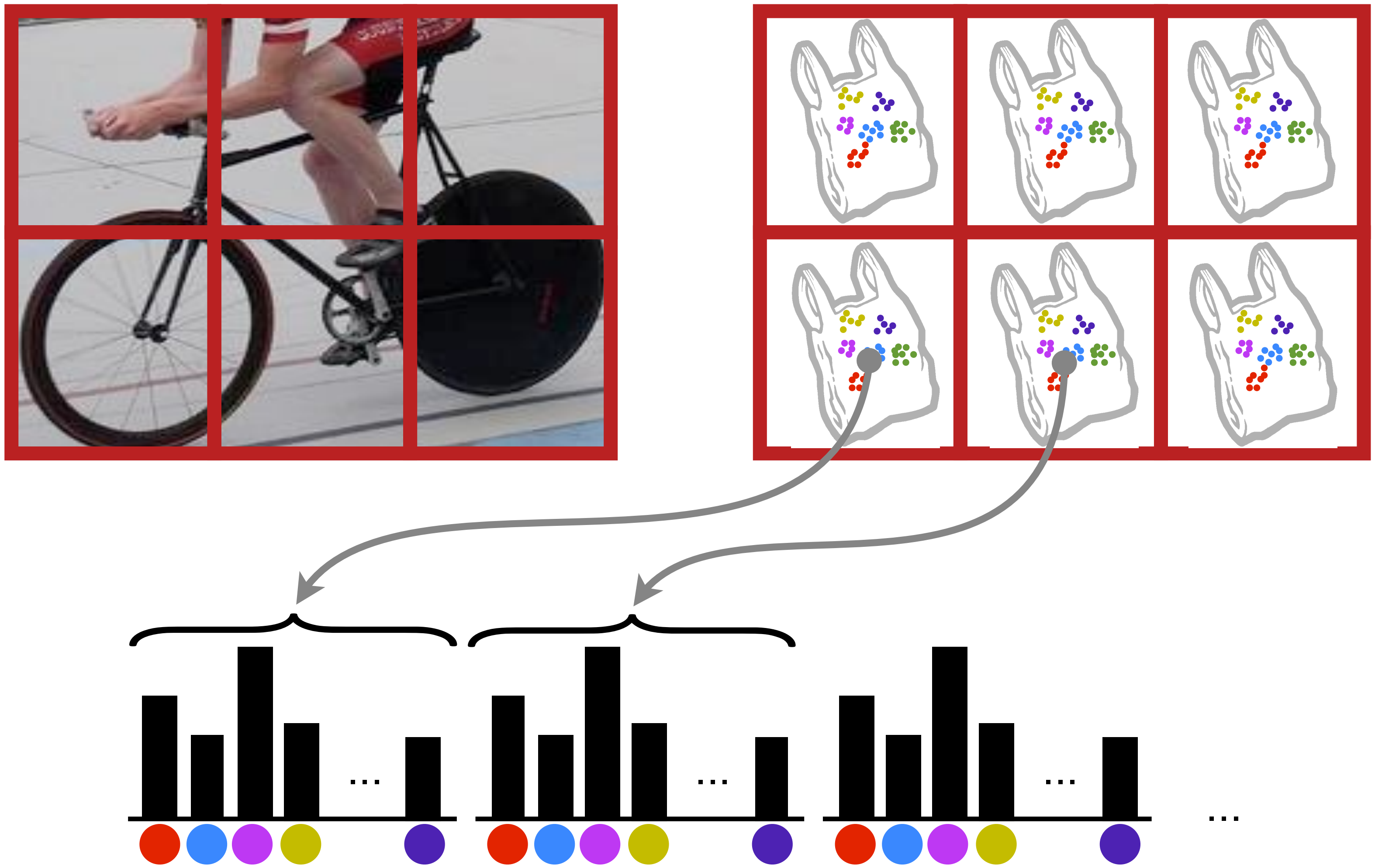
$$\mathbf{u}_k = \frac{1}{M\sqrt{2\pi_k}} \sum_{i=1}^{M} \gamma_k(\mathbf{x}_i) \left( \frac{\mathbf{x}_i - \mu_k}{\sigma_i} - 1 \right)^2$$

# Spatial histograms



[Lazebnik *et al.* 2006]

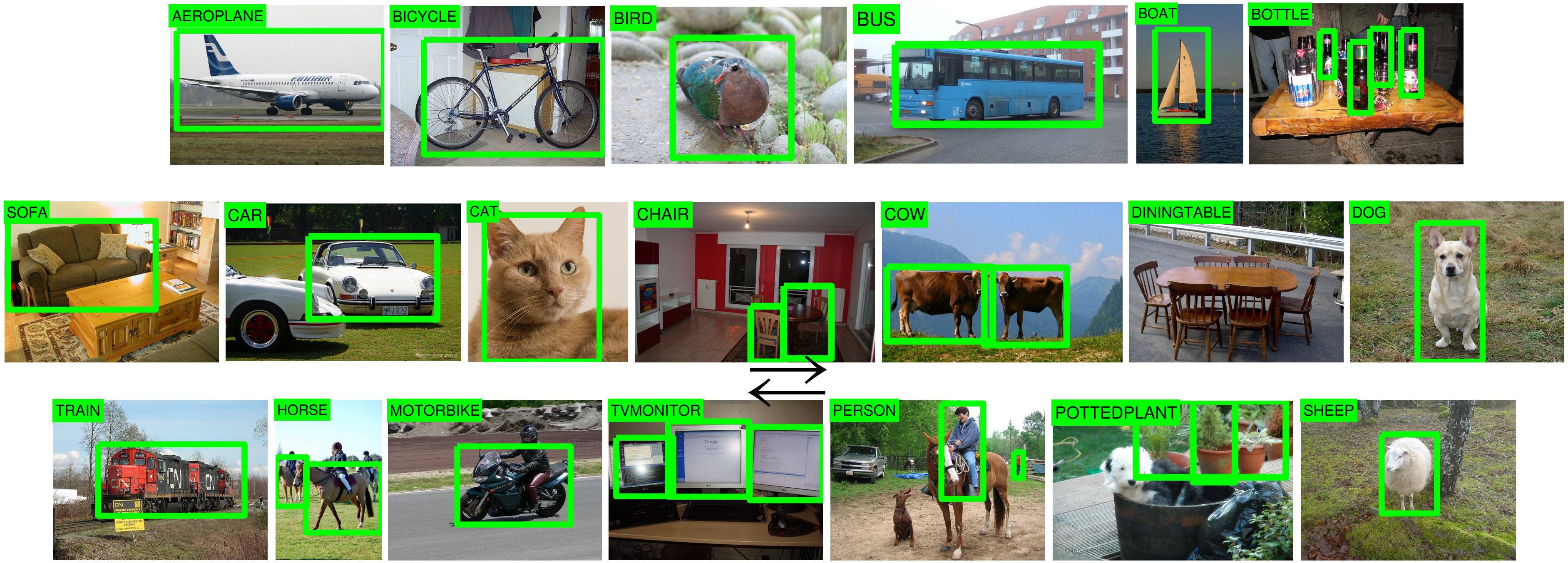
- Weak geometry: pool spatial information locally



# Reference benchmark: PASCAL VOC



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



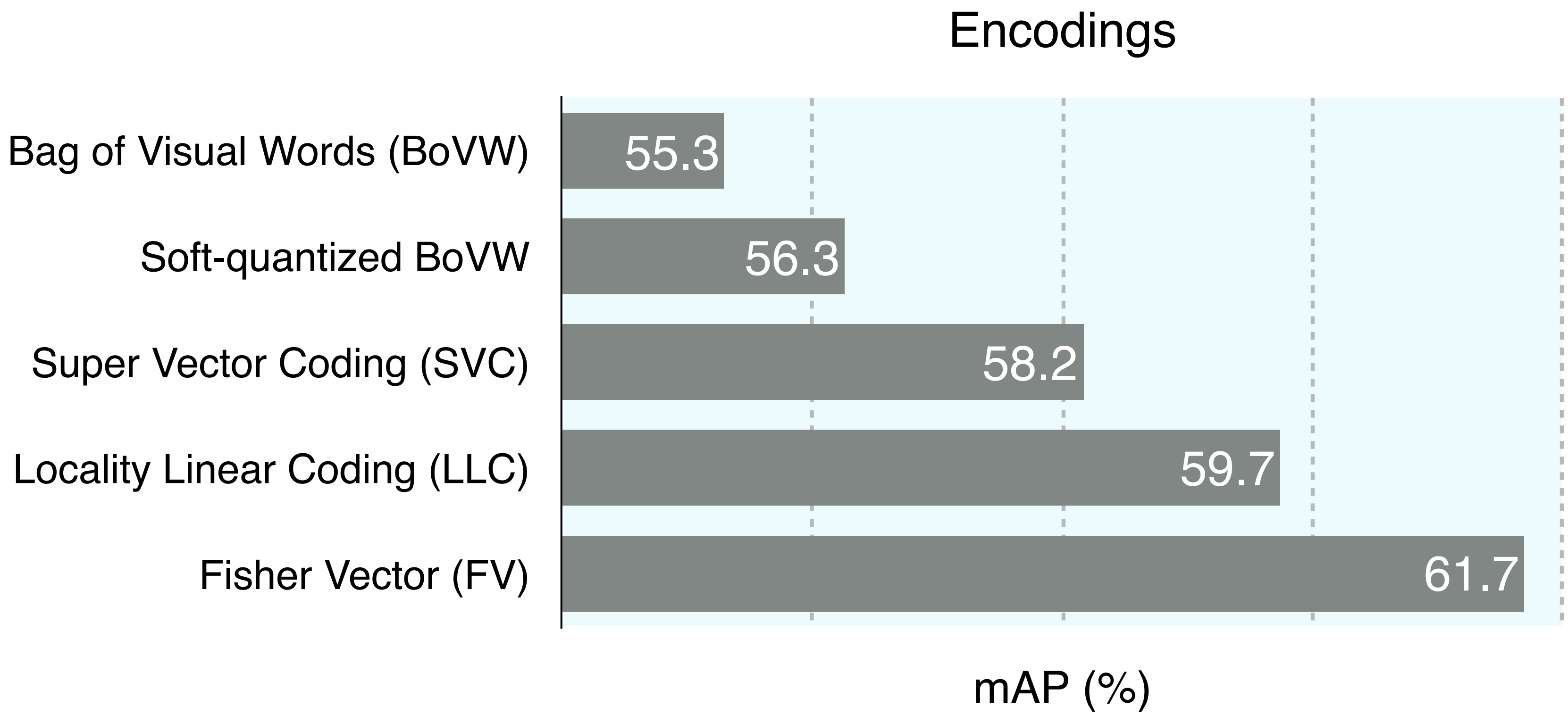

- **Performance**
  mean Average Precision (mAP)

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

# The devil is in the details



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



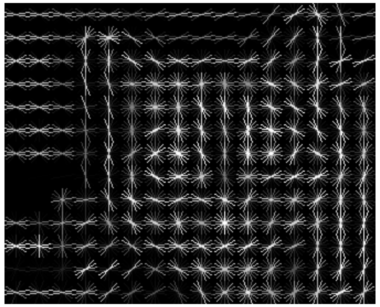

- 2005 — 2012: an industrial production of encodings
- Our evaluation compared them on an equal footing
- The (Improved) Fisher Vectors came out on top

# Some fundamental ideas



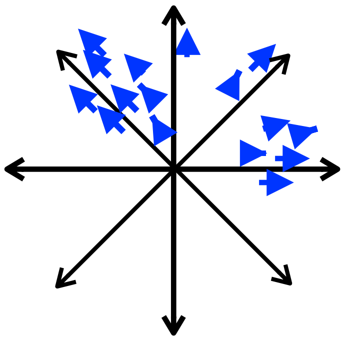
**Local and translation invariant operators**

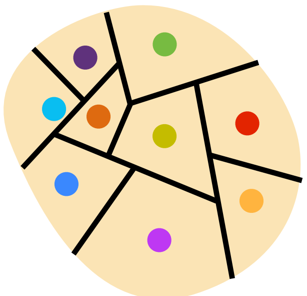
gradients, filters, visual words



**Experts**

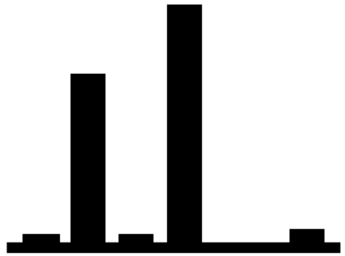
sparsity, quantisation

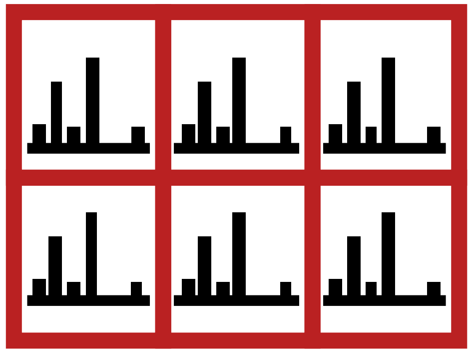




**Pooling**

max, sum, spatial pooling





Part 2: kernel methods

encoder

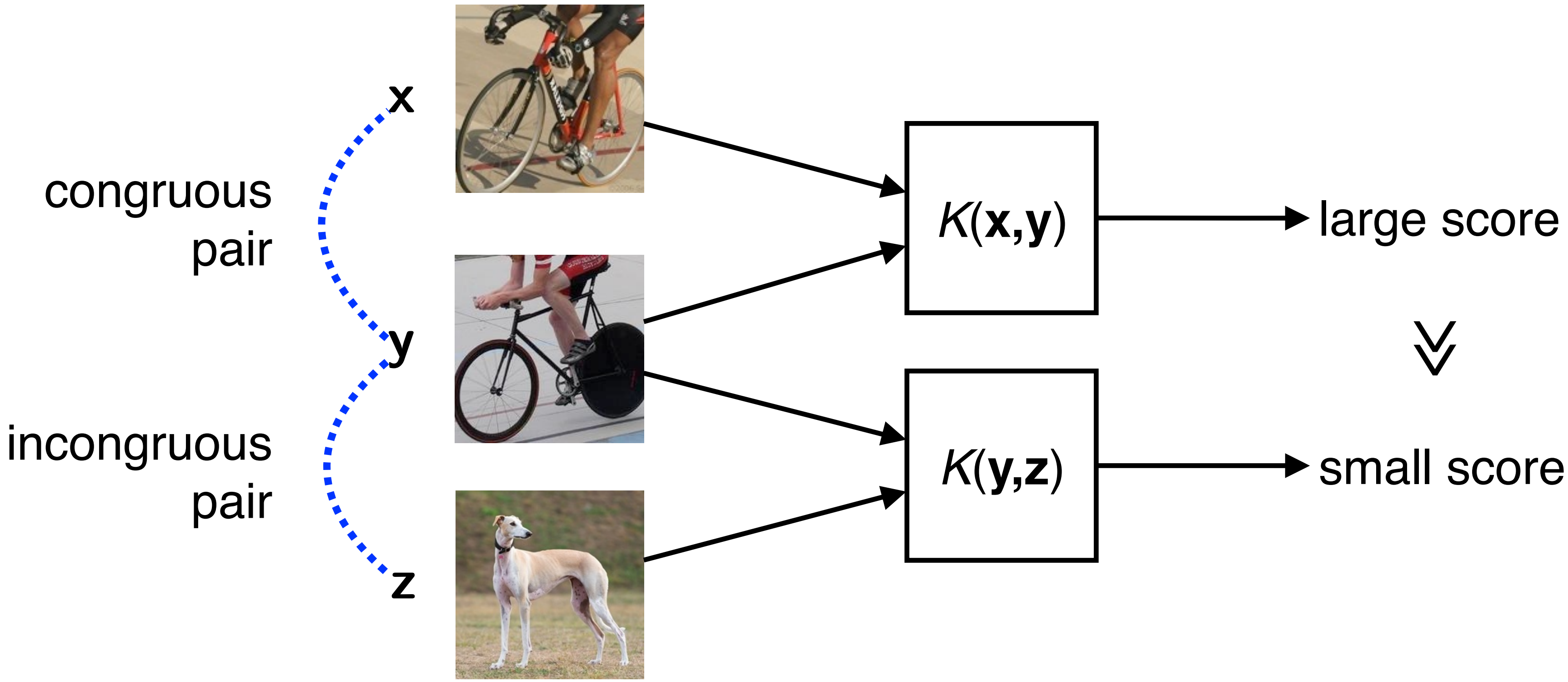
predictor

label

# Kernels

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

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

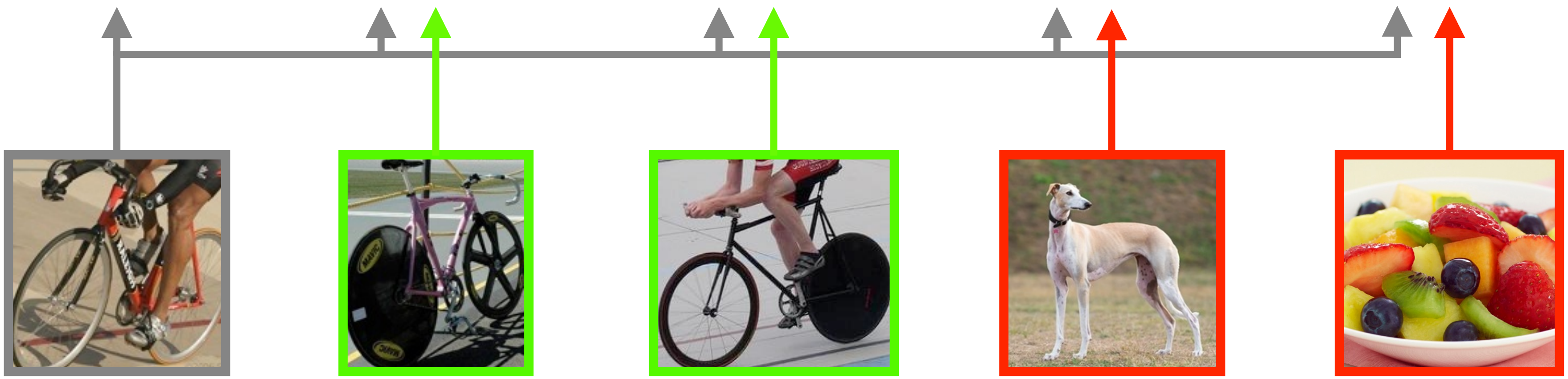


# Kernel predictor

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

- **Task**: predict the class of a datum $\mathbf{x}$
- **How**: use $K$ to compare $\mathbf{x}$ it to all training examples $\mathbf{x}_1$, $\mathbf{x}_2$, …

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



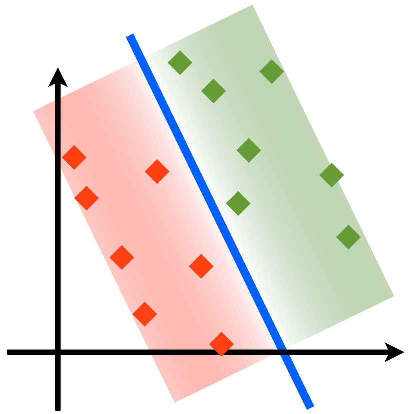
# Non-linear kernels



**Linear SVM**

✔ fast

✘ restrictive

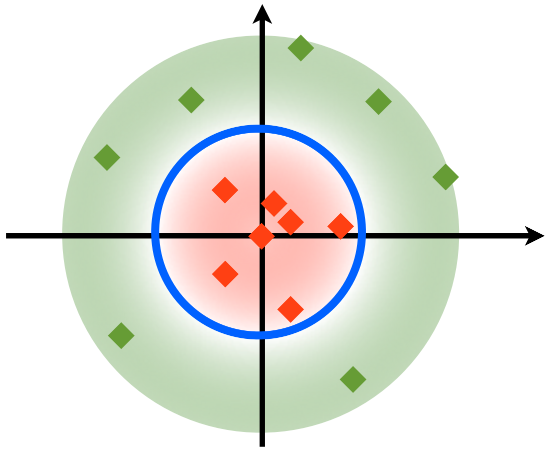


$$F(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$$

**Non-linear SVM**

✘ much slower

✔ powerful



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

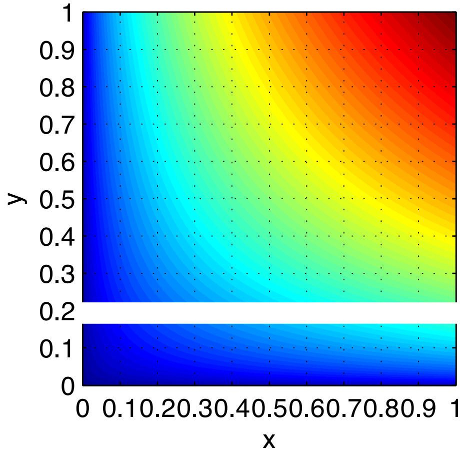
# Additive homogeneous kernels

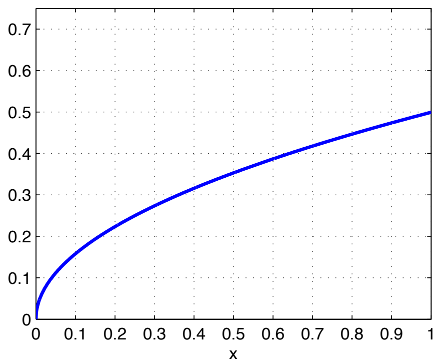


$$\boxed{K(\mathbf{x}, \mathbf{y}) = \sum_{l=1}^{d} k(x_l, y_l)}$$

## Hellinger

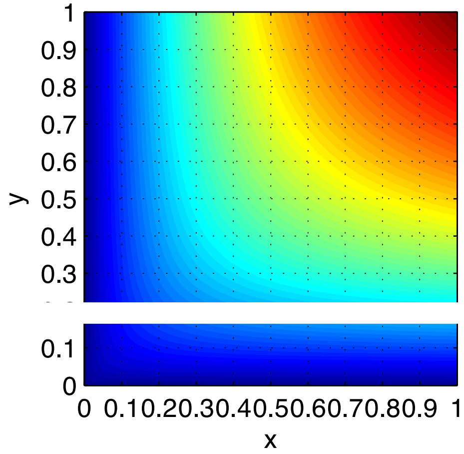
$$\sqrt{xy}$$

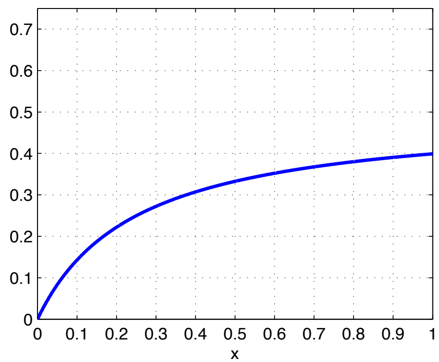



## $\chi^2$

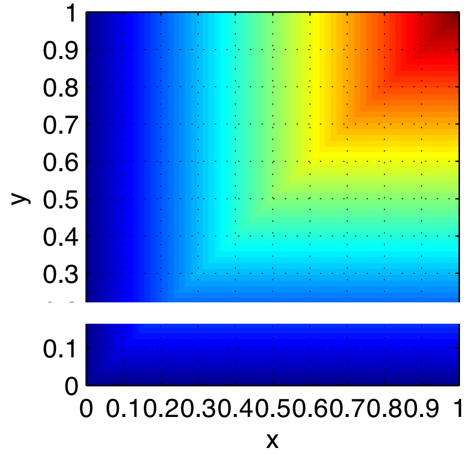
$$\frac{2xy}{x+y}$$

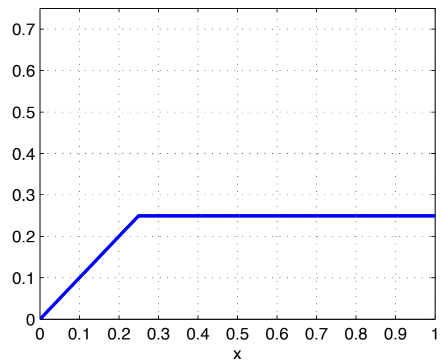


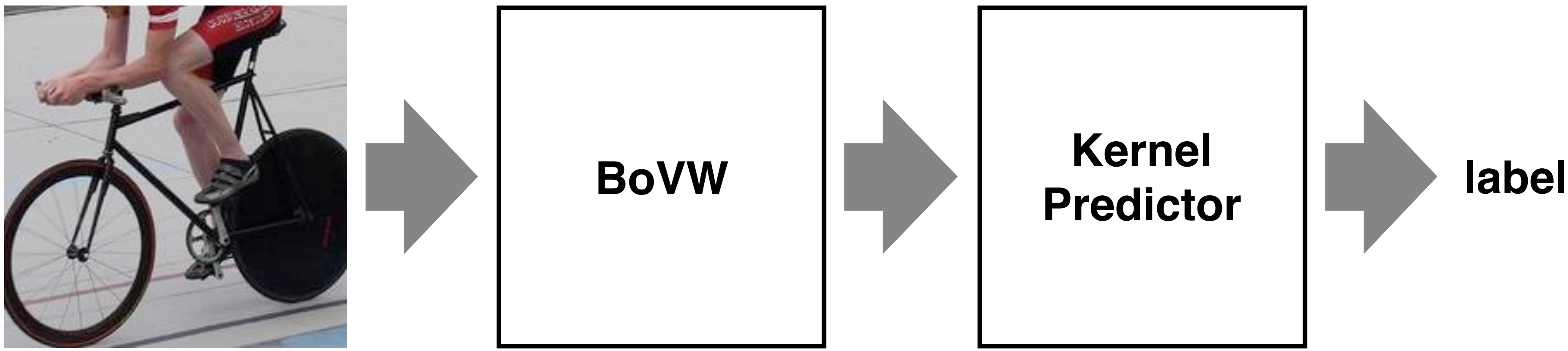

## intersection

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

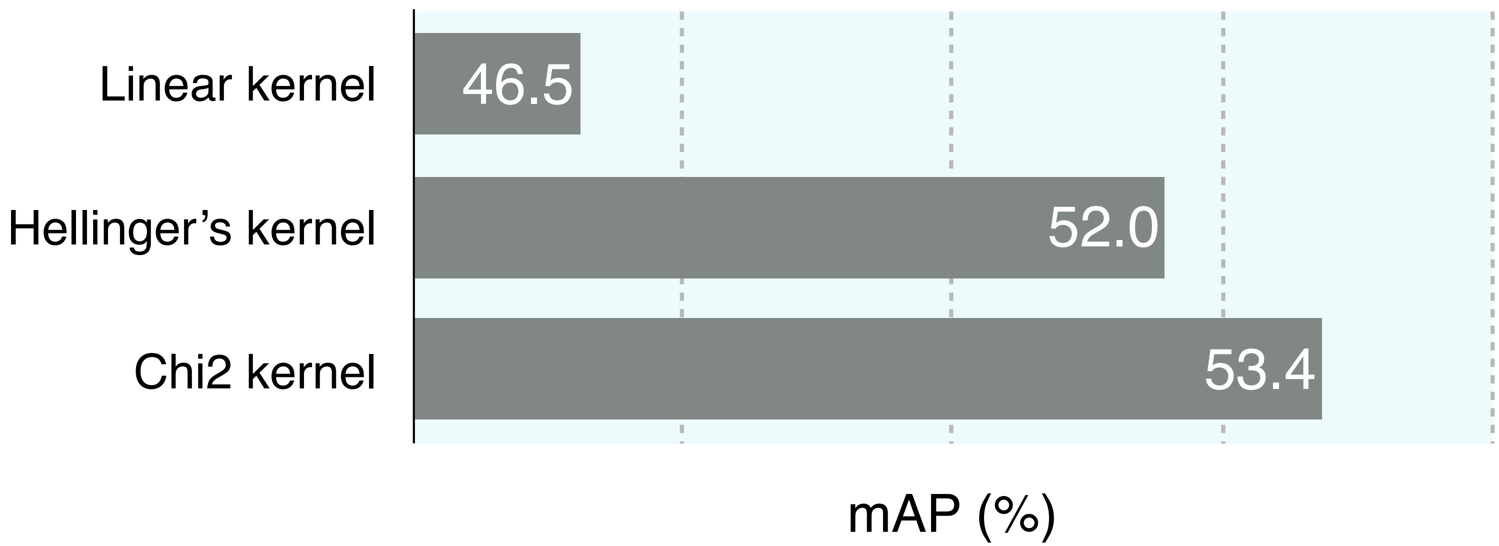


# Additive kernels example




Bag of Visual Word on PASCAL VOC 07



# Non-linear kernels are expensive

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

thousand bicycles

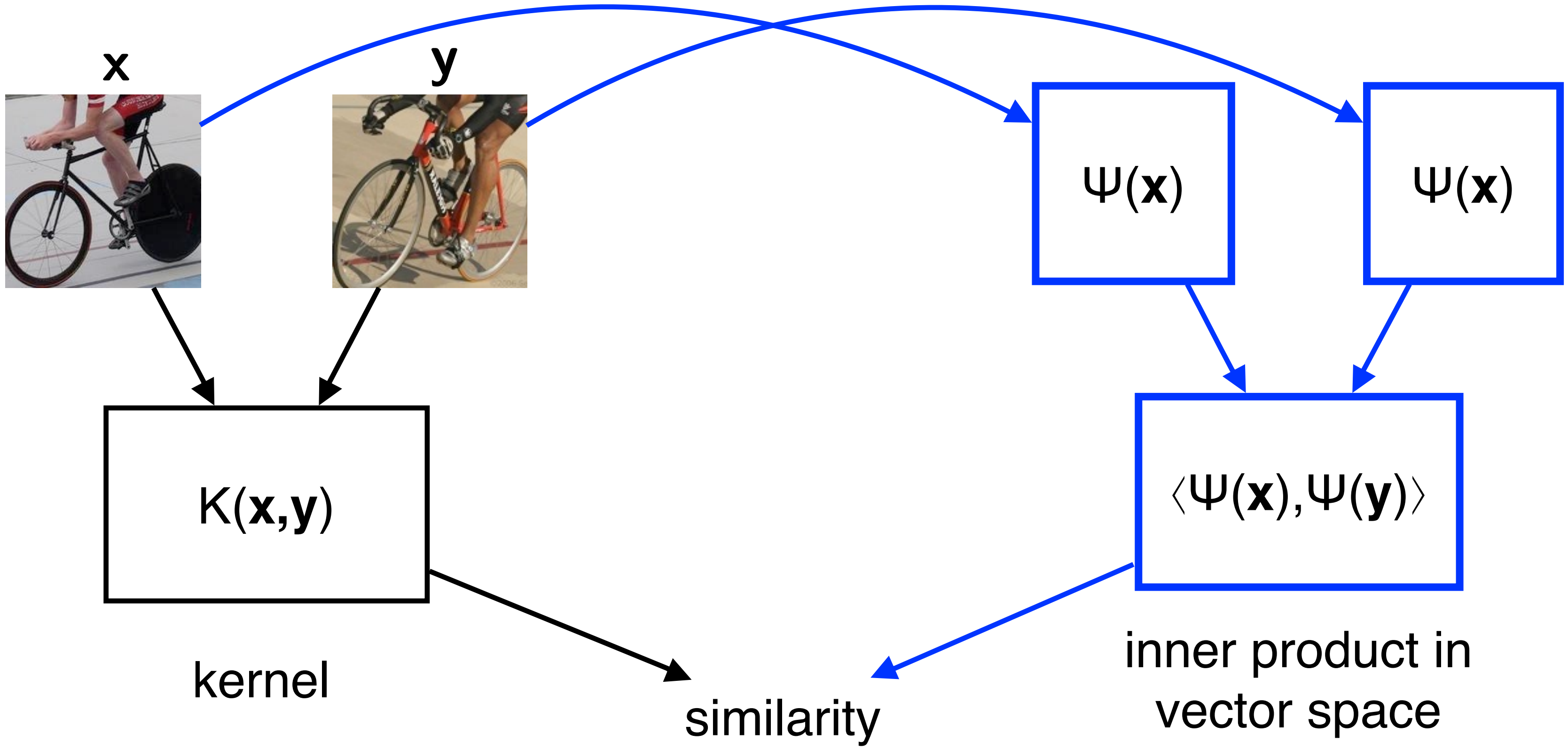
many more non-bicycle

# Kernel maps

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

representation

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



# Explicit kernel maps

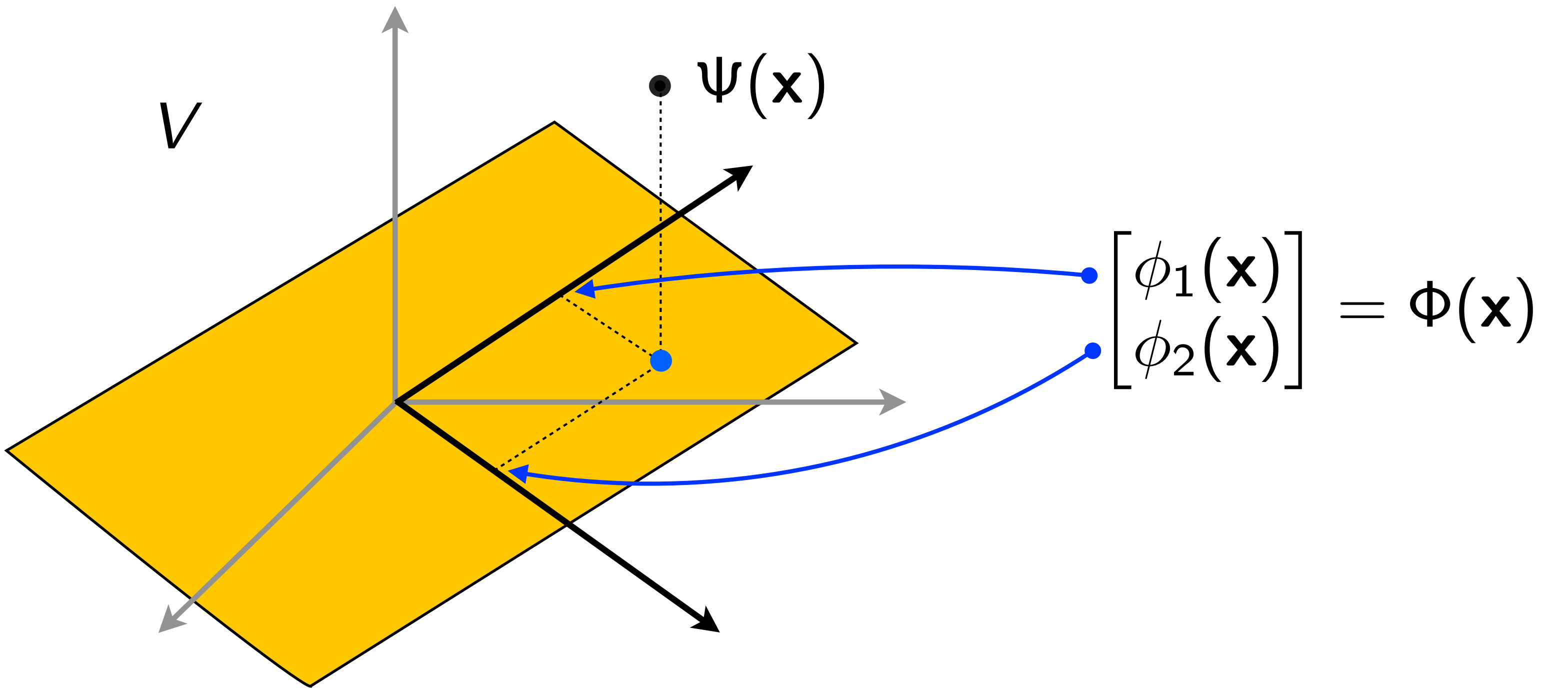


- **Kernel maps**
  - often infinite dimensional
  - used implicit (kernel trick)
  - theoretical

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

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

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



# Explicit maps are efficient



a kernel predictor …

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

$K(\mathbf{x}, \mathbf{y}) \approx \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$

… reduces to a linear predictor

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

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i)$$

a **single vector** summarises
the entire training set

- **The catch**
  - $\Phi$ could be expensive to compute
  - $\Phi(\mathbf{x})$ could be very high-dimensional

# 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)

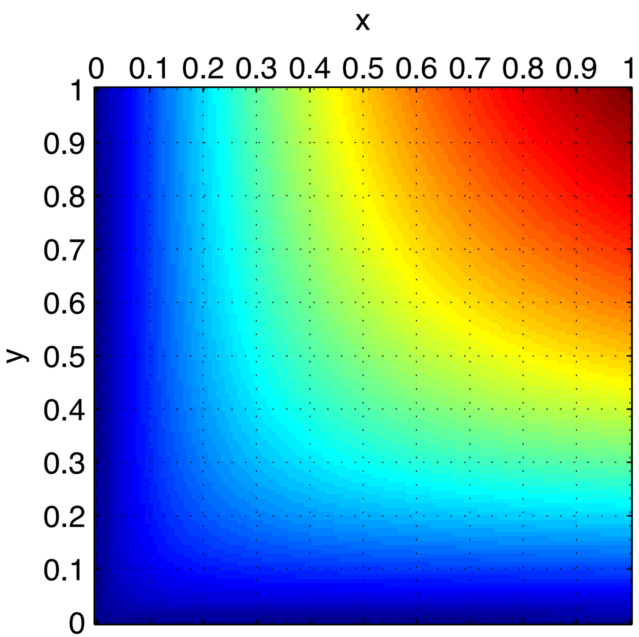
# Empirical explicit maps



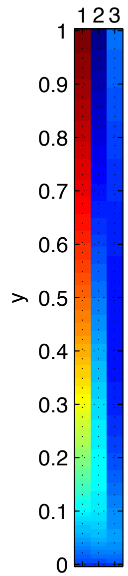
- **Empirical Nyström approximation**
  - Form empirical kernel matrix $K$
  - Find square root $K = V^\top V$ using eigenvectors
  - Keep top $d$ eigenvectors only

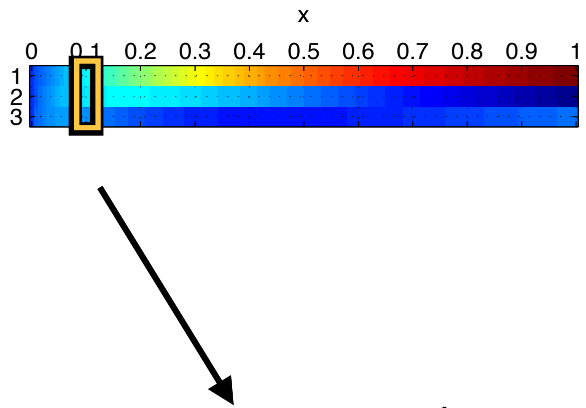
$$K(\mathbf{x}, \mathbf{y}) \approx \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$$

$$K \approx \Phi^\top \Phi$$




$\approx$




$\Phi(\mathbf{x}) \in \mathbb{R}^d$

Each column is an **explicit feature**

# 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

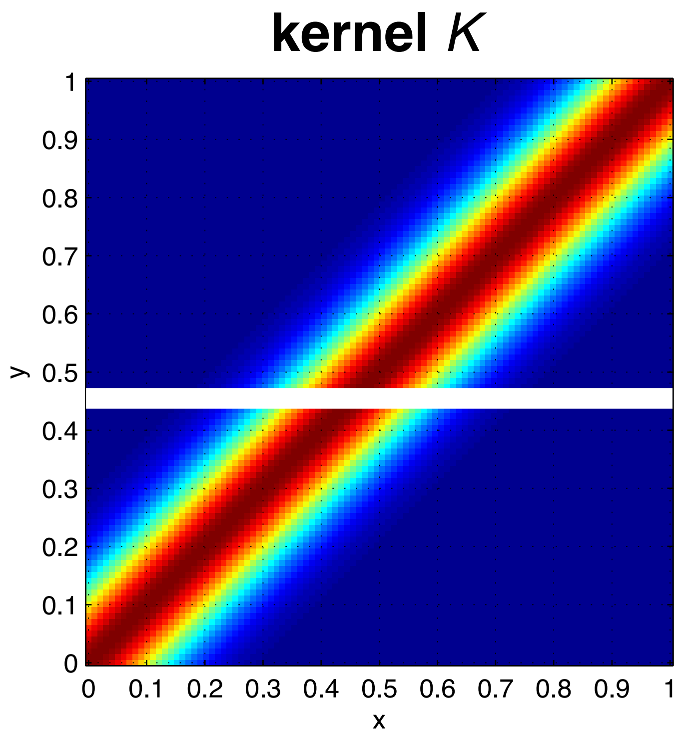
- A few kernels have trivial maps

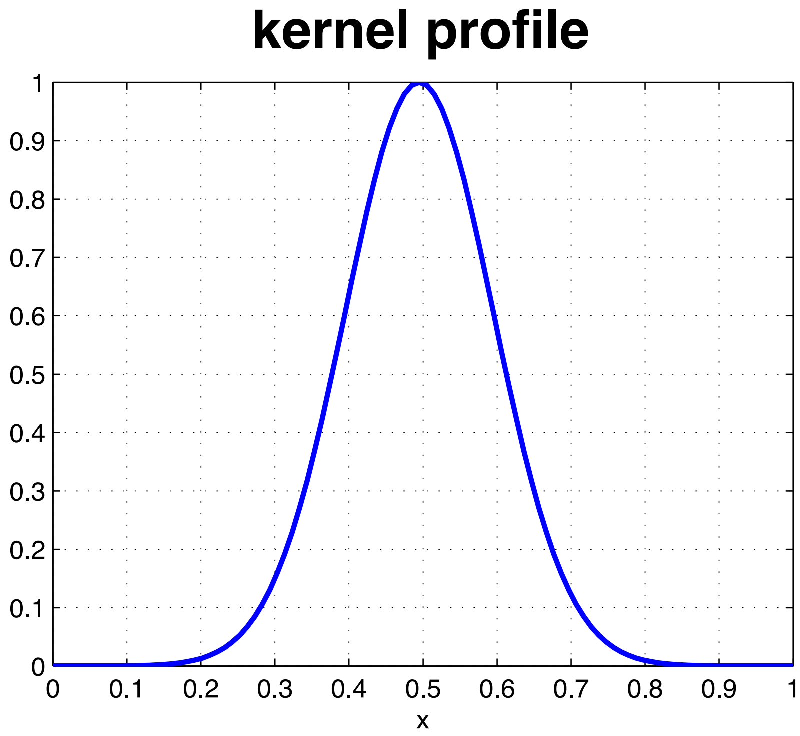
| | | |
|---|---|---|
| linear | $K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ | $\Phi(\mathbf{x}) = \mathbf{x}$ |
| Hellinger's | $K(x, y) = \sqrt{xy}$ | $\Phi(x) = \sqrt{x}$ |

Which other kernels have analytical maps?

# Translation-invariant kernels






Fourier $^{-1}$

$$\boxed{\Phi_\omega(\mathbf{x}) = \kappa_\omega e^{-\mathbf{i}\langle \omega, \mathbf{x} \rangle}}$$

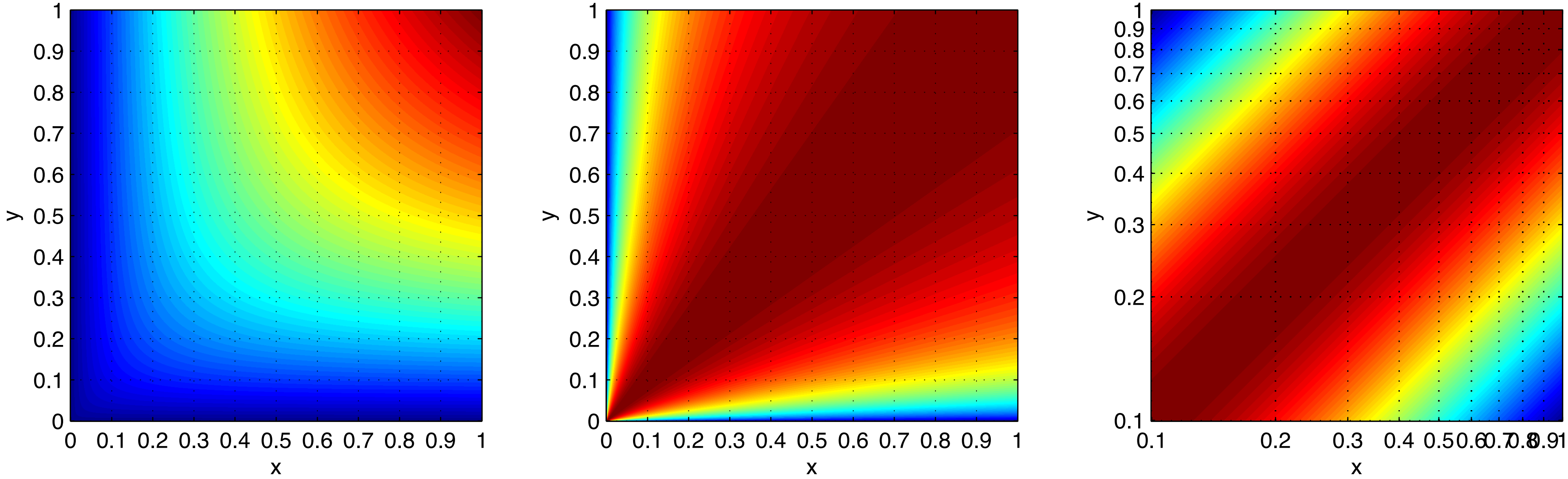
- Because of **translation invariance**
  - Profile = a kernel slice
  - Eigenvectors = sinusoids
  - Eigenvalues = Fourier transform of profile

- Feature map obtained from Fourier tf, often in closed-form

# Homogeneous kernels

$$\boxed{k(cx, cy) = ck(x, y)}$$

$k(x, y)$ $\qquad$ $\dfrac{k(x, y)}{\sqrt{xy}}$ $\qquad$ $x \leftarrow \log x$




$$\boxed{\Phi_\omega(x) = \kappa_\omega \sqrt{x}\, e^{-\mathbf{i}\langle \omega, \log x \rangle}}$$

[Vedaldi Zisserman 2010, 11]

# Homogeneous kernel map: examples



| | | |
|---|---|---|
| linear | $K(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y} \rangle$ | $\Phi(\mathbf{x}) = \mathbf{x}$ |
| Hellinger's | $K(x,y) = \sqrt{xy}$ | $\Phi(x) = \sqrt{x}$ |
| Chi2 | $K(x,y) = \dfrac{2xy}{x+y}$ | $\Phi_\omega(x) = \sqrt{\dfrac{2x}{\pi(1+4\omega^2)}}\, e^{-\mathbf{i}\,\omega \log x}$ |
| Intersection | $K(x,y) = \min\{x,y\}$ | $\Phi_\omega(x) = \sqrt{x \operatorname{sech}(\pi\omega)}\, e^{-\mathbf{i}\,\omega \log x}$ |

[Vedaldi Zisserman 2010, 11]

# Example: $\mathrm{Chi}^2$ map



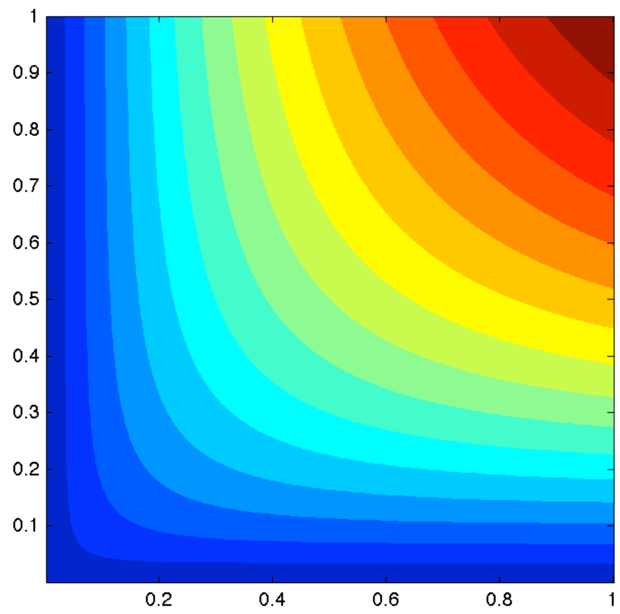
**MATLAB code for Chi2 kernel**

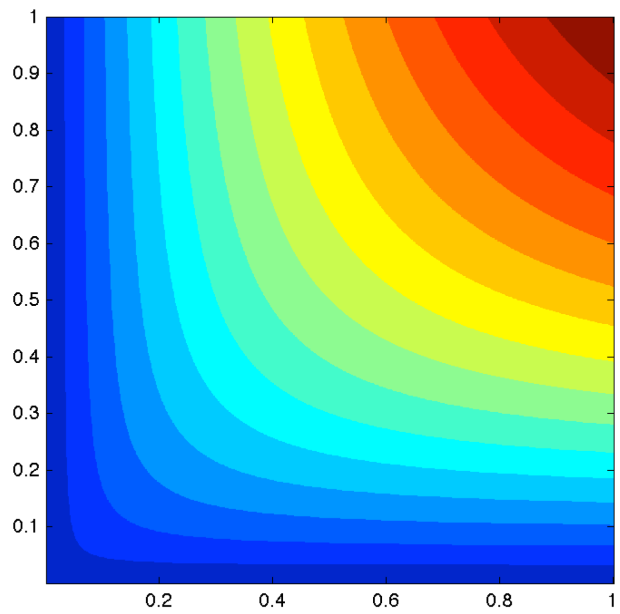
```
x = .01:.01:1 ;
for i = 1:100
  for j = 1:100
    K(i,j) = ...
      2*x(i)*x(j)/(x(i)+x(j));
  end
end
```

**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: Chi$^2$ 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

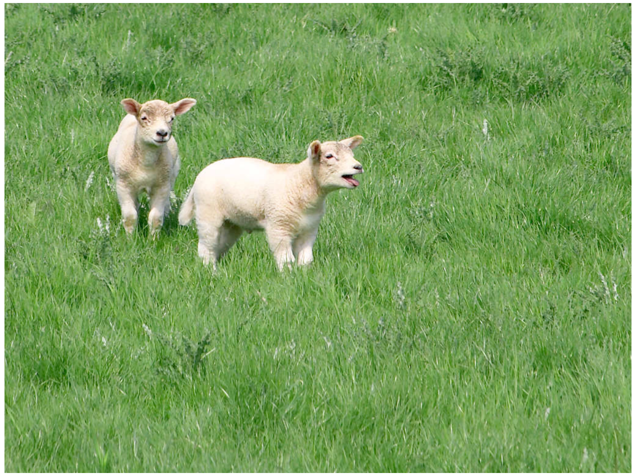
> 1 h → 22.6 s

**160× speedup**

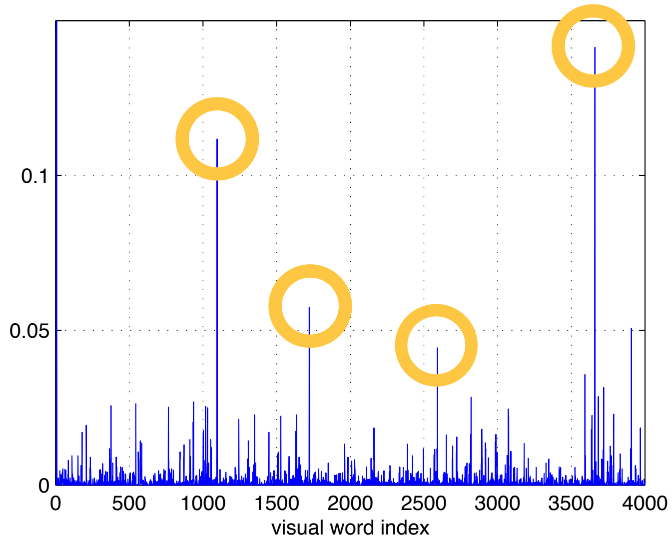
# The square-root map and burstiness



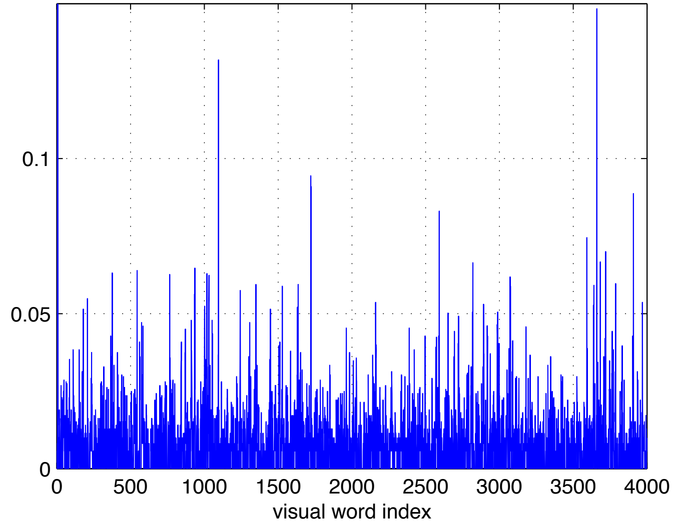
dominated by “grass”



$\mathbf{x}$




$\Phi(\mathbf{x}) = \sqrt{\mathbf{x}}$




- **Burstiness**
  - histograms are often dominated by **bursts of identical words**
- **Hellinger’s kernel**
  - compensates by taking the square root
- **Simple and broadly applicable**
  - E.g. RootSIFT

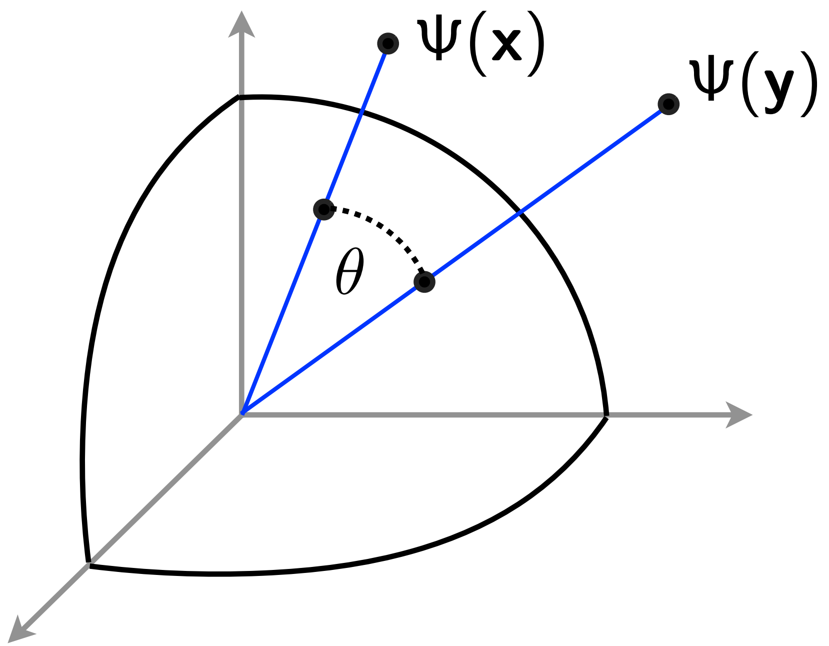
# Normalization



- Recall: a kernel should encode a useful notion of similarity
- Assumption: **any object should be most similar to itself**

$$K(\mathbf{x}, \mathbf{x}) \geq K(\mathbf{x}, \mathbf{y})$$

- Easy fix in feature space: measure angles by l²-normalising vectors




$$\cos\theta = \left\langle \frac{\Psi(\mathbf{x})}{\|\Psi(\mathbf{x})\|}, \frac{\Psi(\mathbf{y})}{\|\Psi(\mathbf{y})\|} \right\rangle$$

$$K'(\mathbf{x}, \mathbf{y}) = \frac{K(\mathbf{x}, \mathbf{y})}{\sqrt{K(\mathbf{x}, \mathbf{x})}\sqrt{K(\mathbf{y}, \mathbf{y})}}$$

Part 3: learning the embedding

encoder

predictor

label

# Learning to compare



For a thorough review: [Weinberger Saul JMLR 2009]

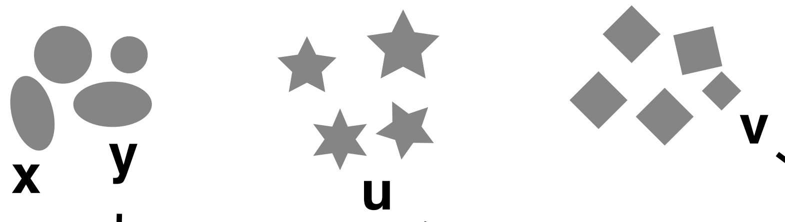
- **Goal**
  - compare (rather than classify) objects **x**, **y**
  - formally, learn a distance $d^2(\mathbf{x},\mathbf{y})$
- **Desiderata**
  - if **x** and **y** are *congruous* $\implies$ small distance
  - if **x** and **y** are *incongruous* $\implies$ large distance
- **Parametrisation of the distance**

Euclidean distance + linear projection $W$

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

# Classification-like constraints






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

congruous pairs — incongruous pairs

- For all object pairs **x**, **y**
  - congruous $\Longrightarrow$ distance **smaller** than threshold - margin
  - incongruous $\Longrightarrow$ distance **larger** than threshold + margin

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

# 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

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

$+$

linear projection

$$\mathbf{x} \in \mathbf{R}^n \xrightarrow{W \in \mathbf{R}^{m \times n}} \bar{\mathbf{x}} = W\mathbf{x} \in \mathbf{R}^m$$

- *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



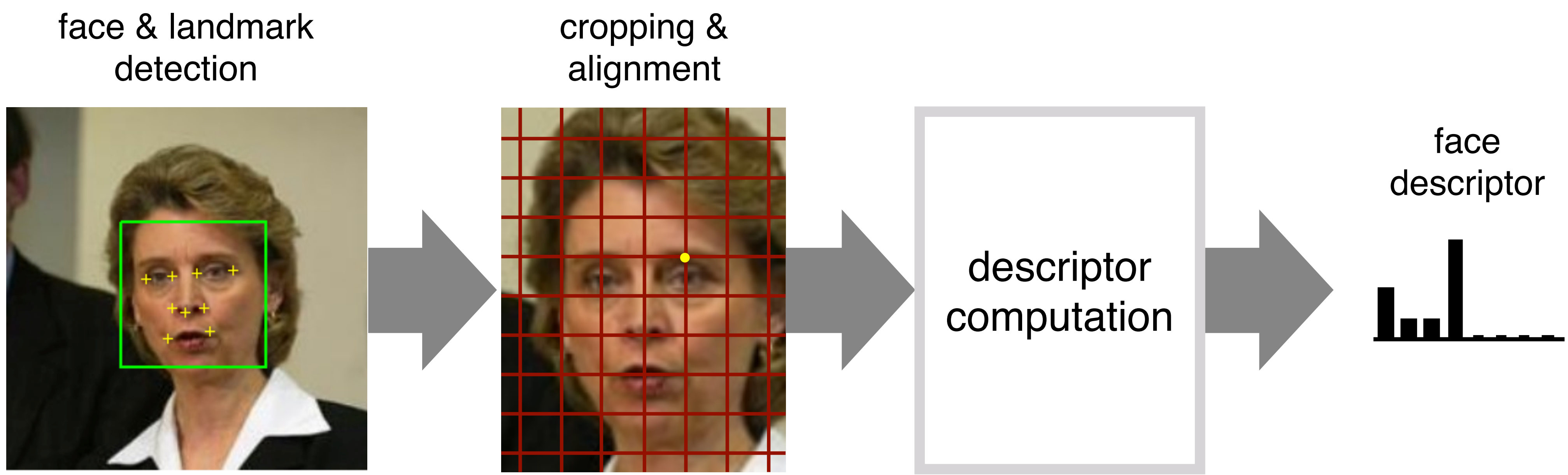
[Simonyan *et al.* BMVC 2013]


- **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]

# Face verification / recognition



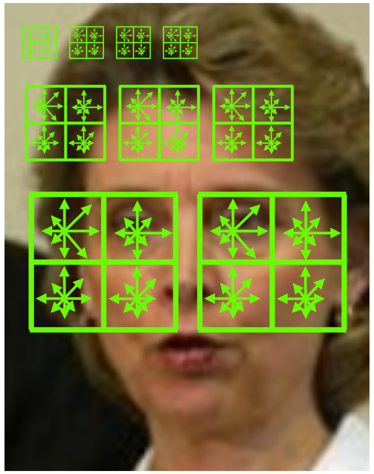


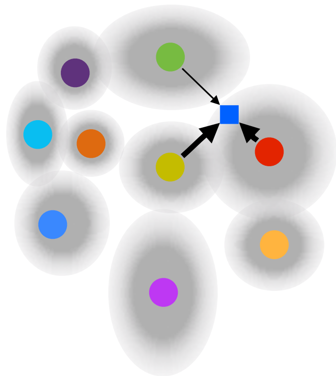

- **Typical face identification pipeline**
  1. Face detection
  2. Face registration (may use detected landmarks)
  3. Descriptor computation (may use detected landmarks)
  4. Decision (classification, distance learning, dim. reduction, …)

# Fisher Vector Faces (FVF)

[Simonyan *et al.* 2012]

descriptor computation = Dense SIFT + Fisher Vector + Metric learning



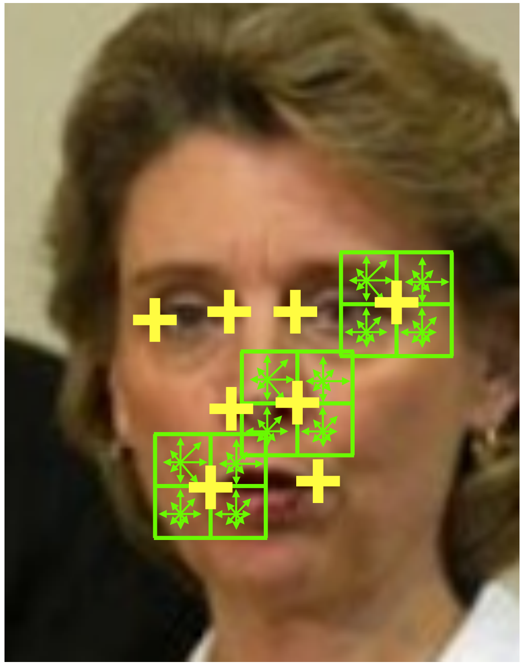


$$d_W^2(\mathbf{x},\mathbf{y}) = \|W\mathbf{x} - W\mathbf{y}\|^2$$

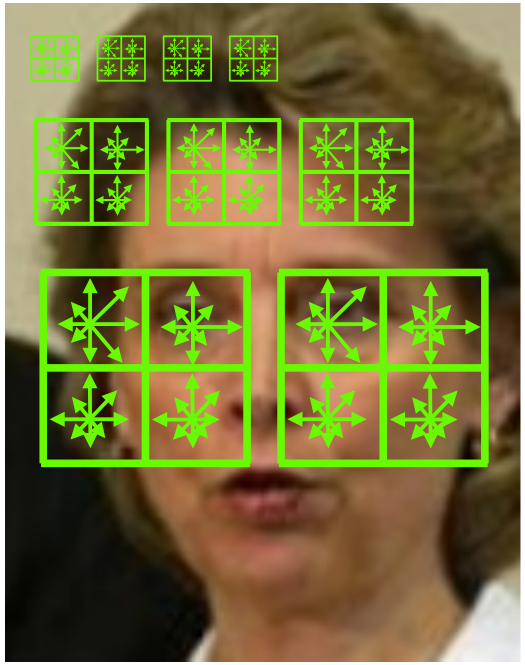
- **FVF descriptor**
  - A. Features: *densely sampled*, *spatially augmented* SIFT features
  - B. Encoding: Fisher Vectors
  - C. Post-processing: metric learning & dimensionality reduction
  - D. Optional post-processing: binarization

# Landmarks or not?

landmarks

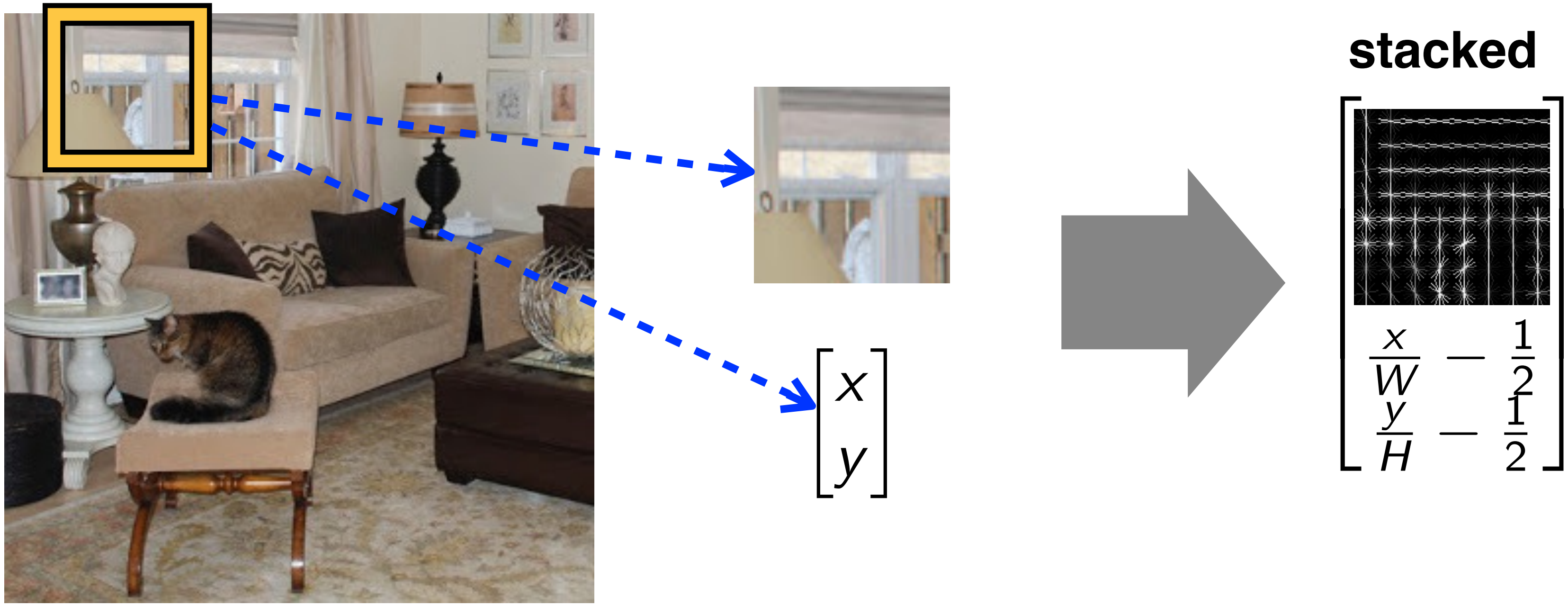


FVF



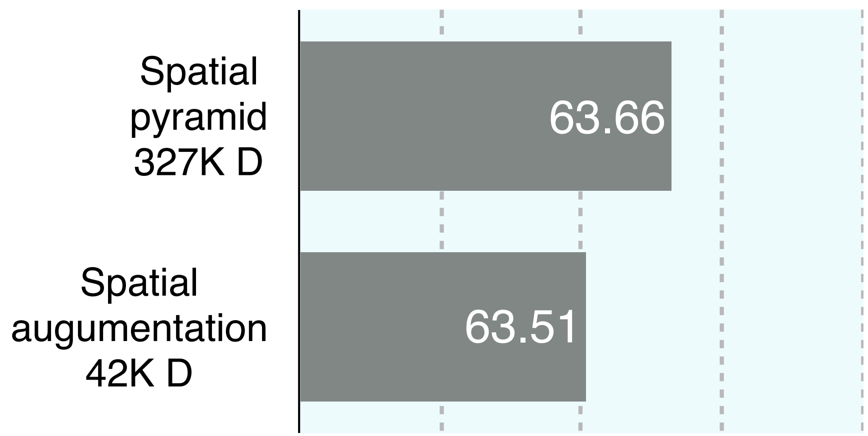
- **Landmarks**
  - sample patches at landmarks
  - good: alignment
  - bad: expensive, brittle
- **Dense sampling**
  - sample patches uniformly
  - good: simple, robust
  - bad: no alignment

# 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

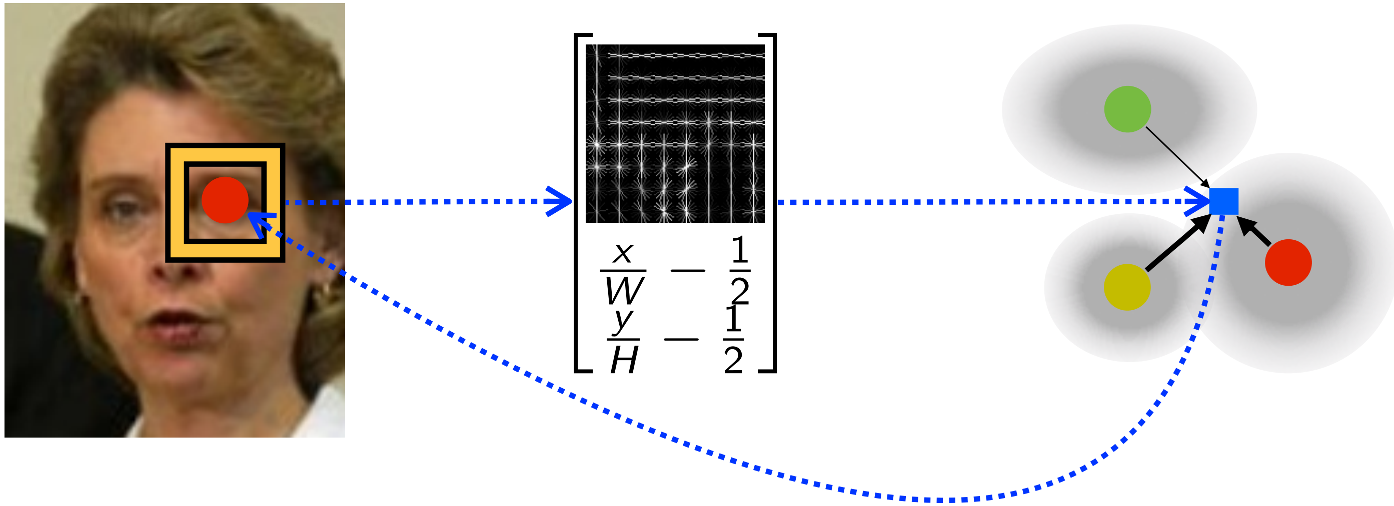
PASCAL VOC + FV

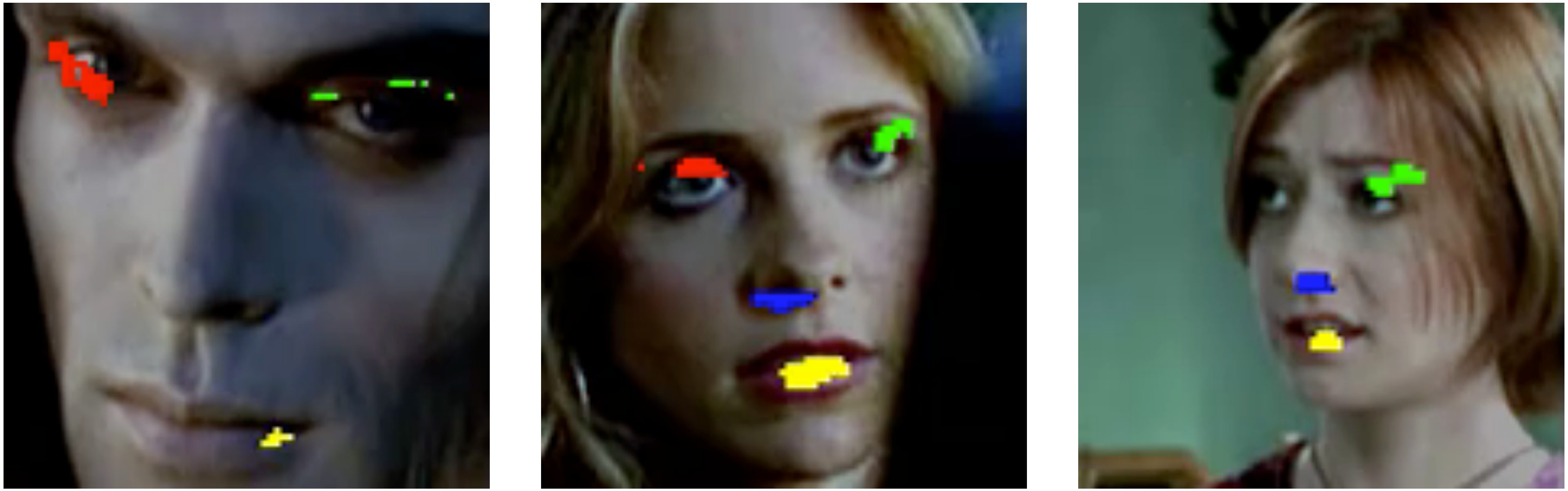



mAP (%)

[Chatfield *et al.* 2014]

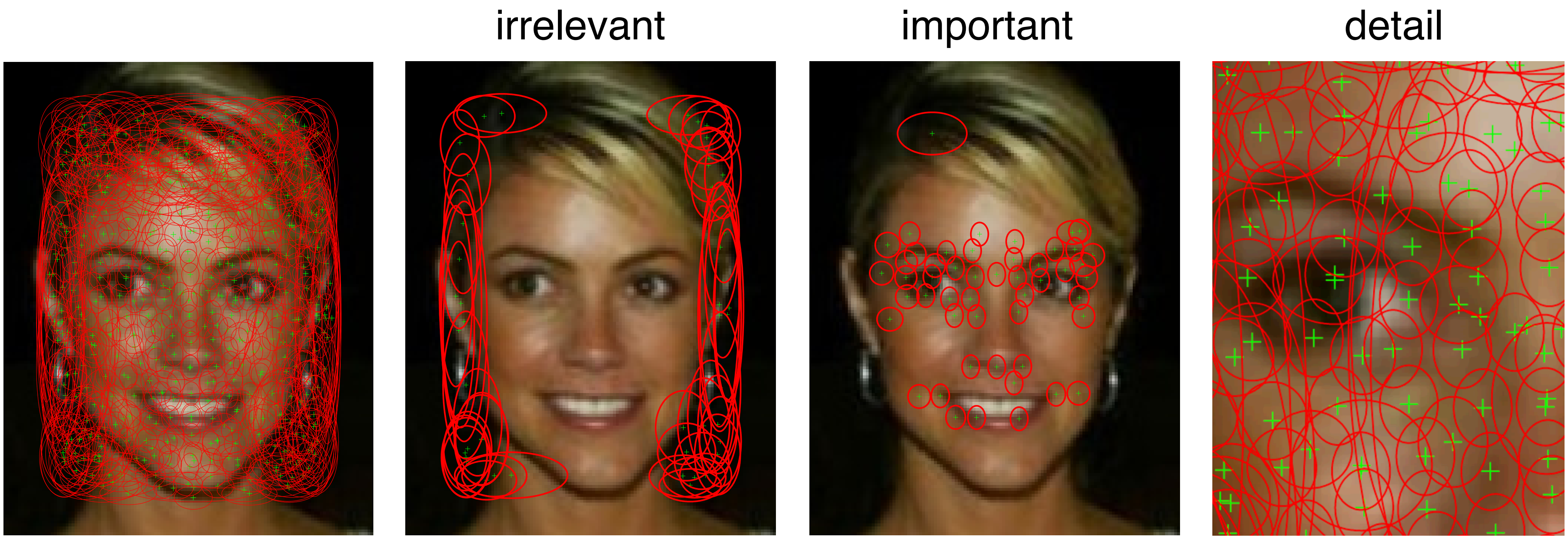
# Fisher Vectors as part-based models

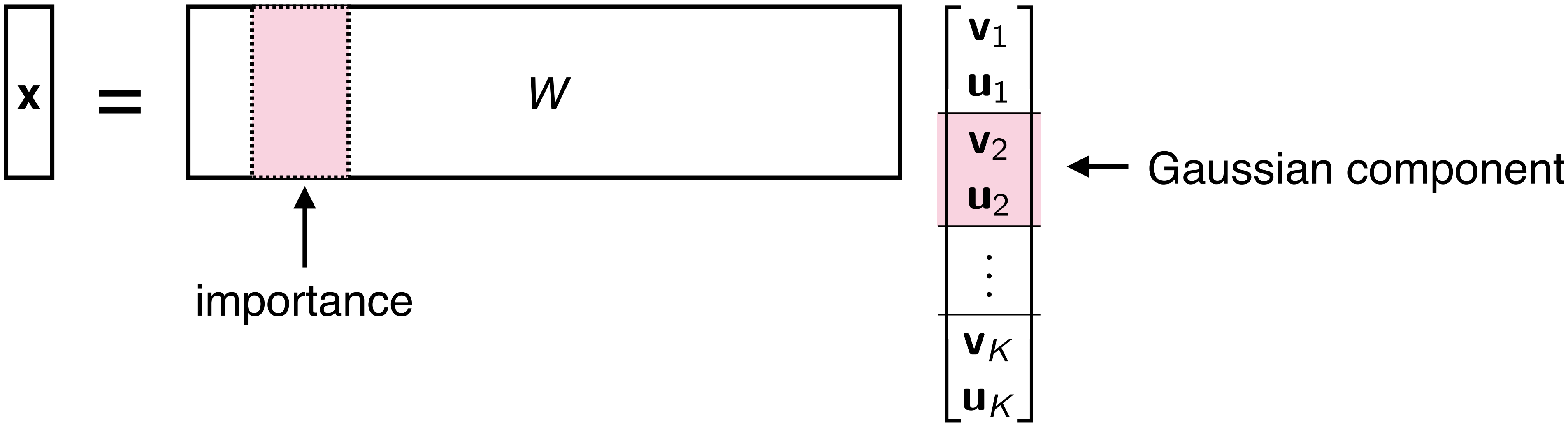


# Fisher Vectors as part-based models

## Distinctive face elements



# Video Fisher Vector Faces (VF$^2$)



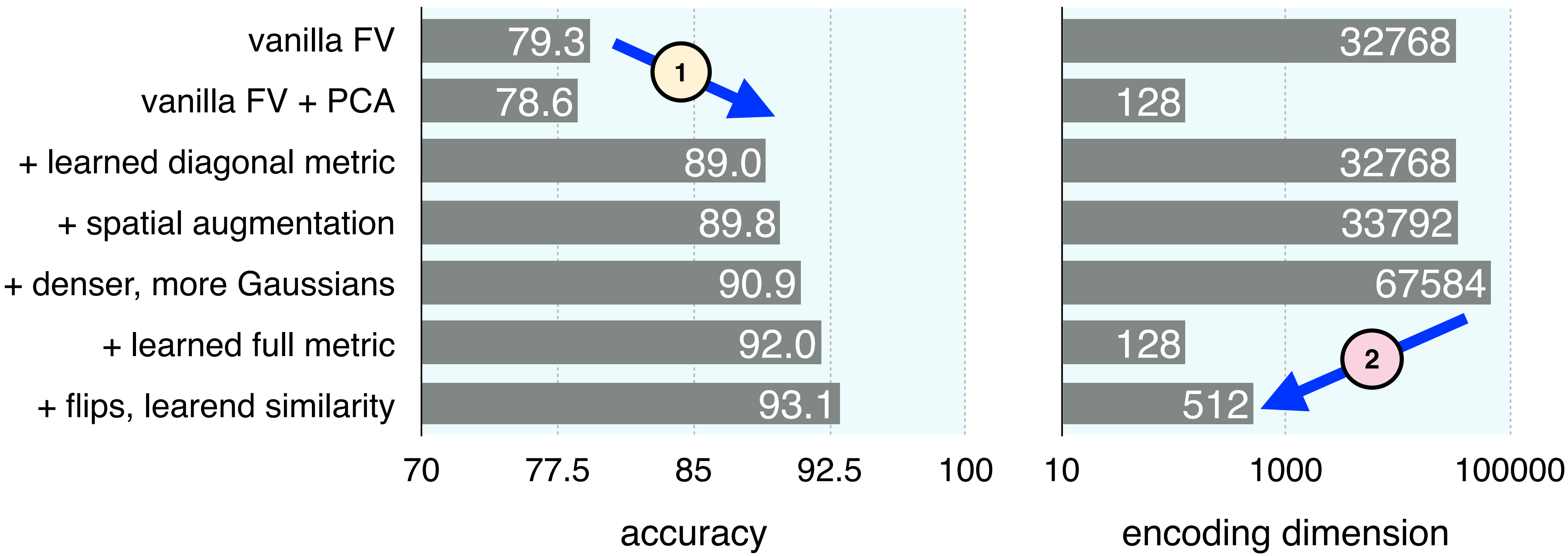
[Parkhi *et al.* CVPR 2014]

- **From still images to videos**
  - Hard-assignment FV
  - RootSIFT
  - Image, video, and jittered pooling
- **Dimensionality reduction**
  - Metric learning
  - Joint metric and distance learning
  - Binarization

# FVF design choices



Benchmark: Labelled Faces in the Wild (LFW)




1

**Metric learning** dramatically boosts **performance**

2

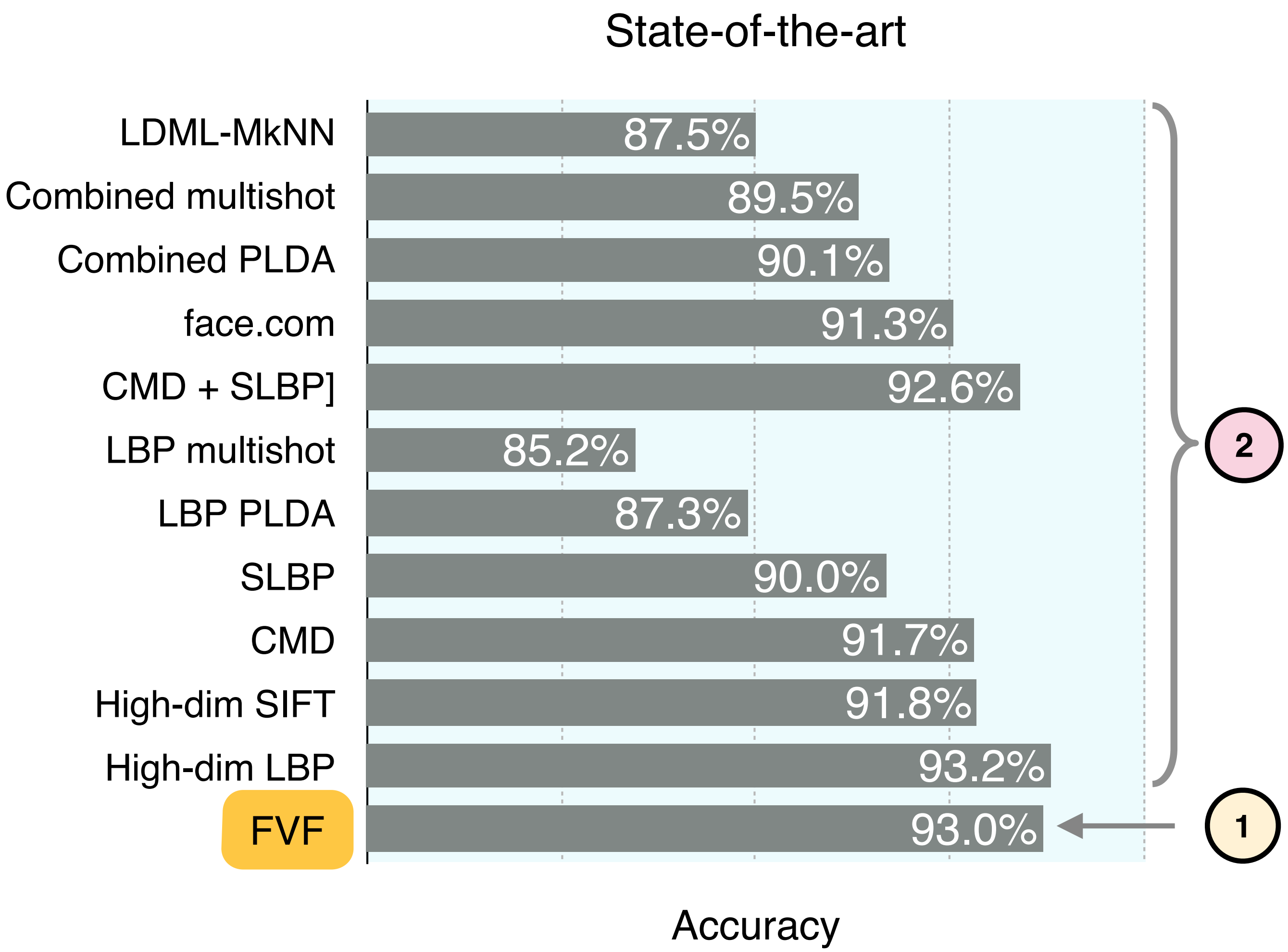
**Full metric** allows for a tremendous **compression**

3

**Simple**
(no complex alignment / landmarks)

# FVF still image performance

Benchmark: Labelled Faces in the Wild




1

Accurate

Fast

Small

2

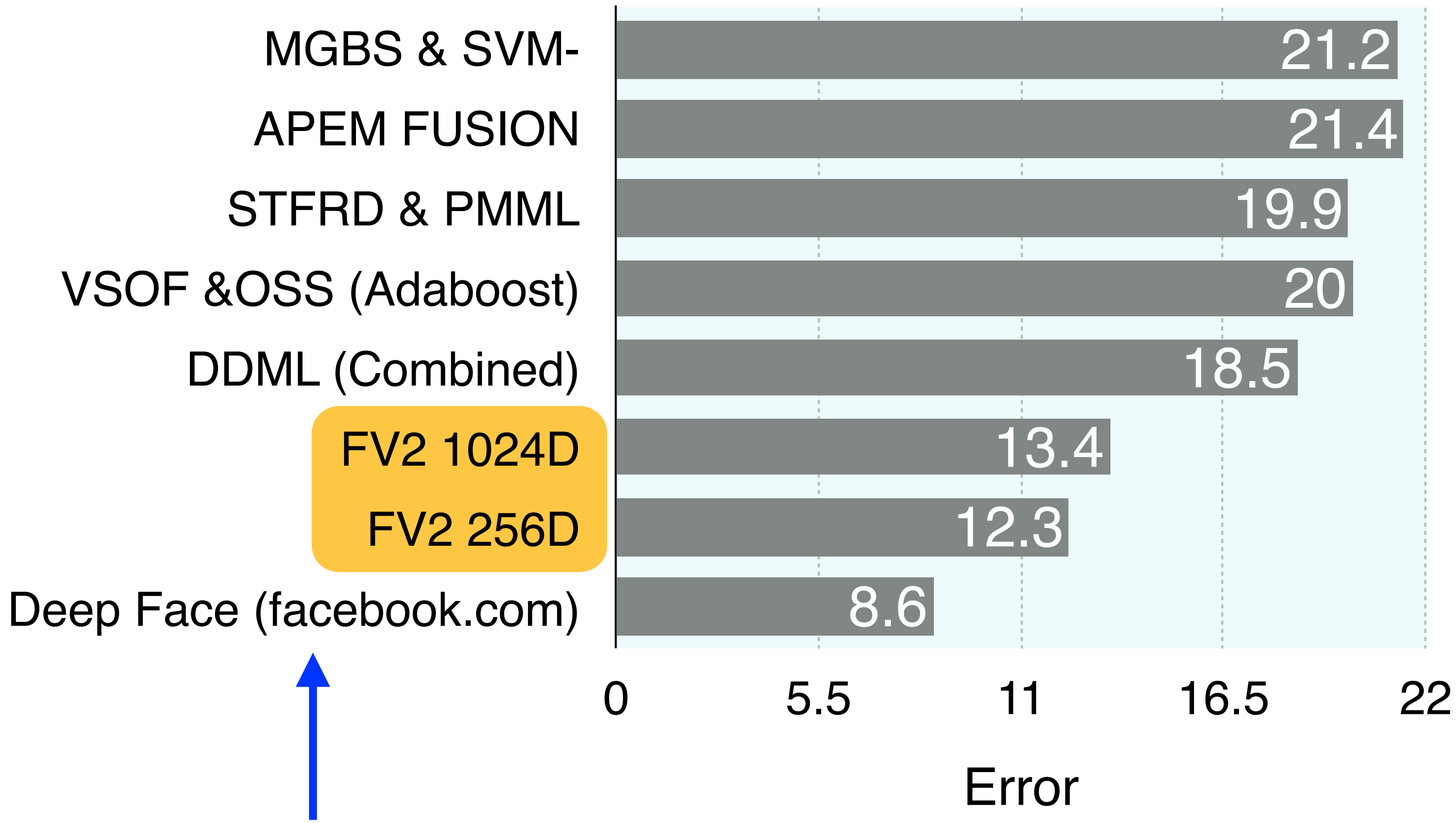
Simpler

(no complex alignment / landmarks)

# FV² video performance



Benchmark: YouTube Faces

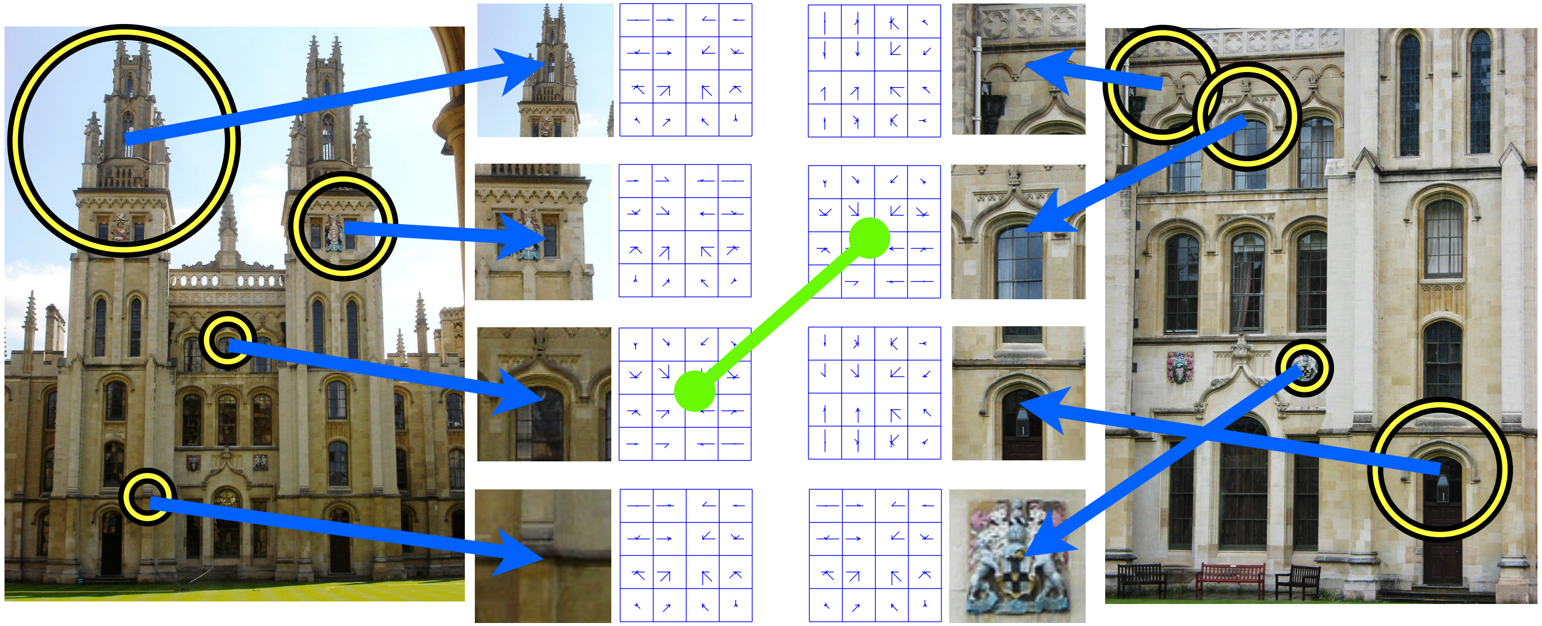



requires fairly sophisticated alignment
and a lot more training data

# Other applications: local descriptor learning

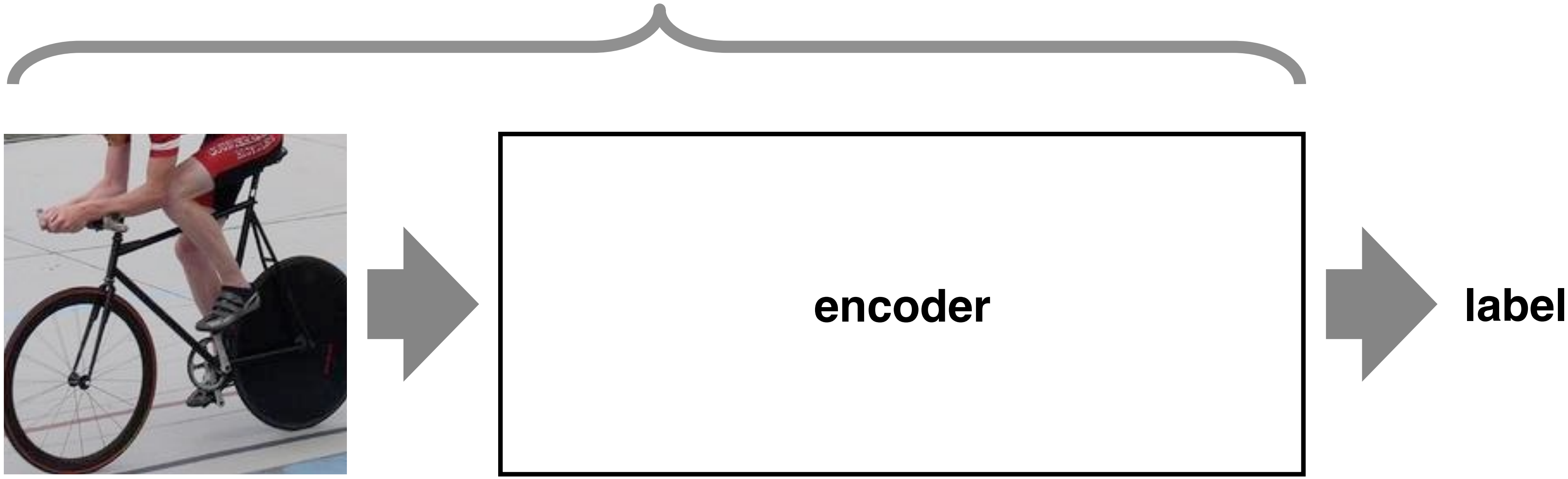


[Simonyan *et al.* 2011]

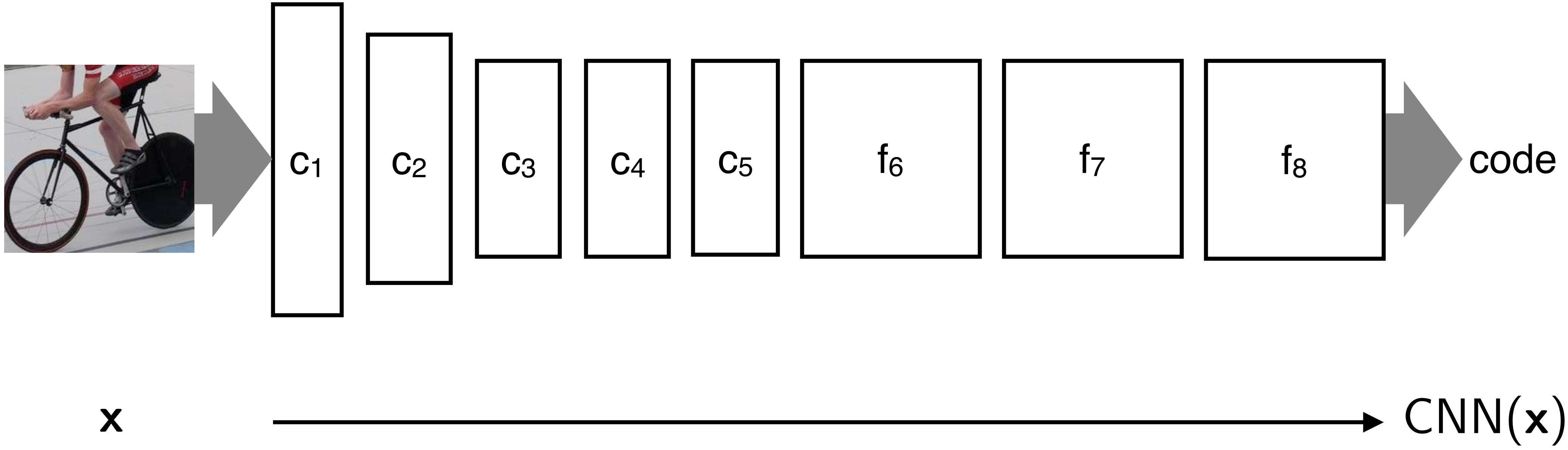


- Learning to compare & compress works beyond faces
- State-of-the-art **local descriptors** and **instance search**
- http://www.robots.ox.ac.uk/~vgg/research/learn_desc/

Part 4: deep learning

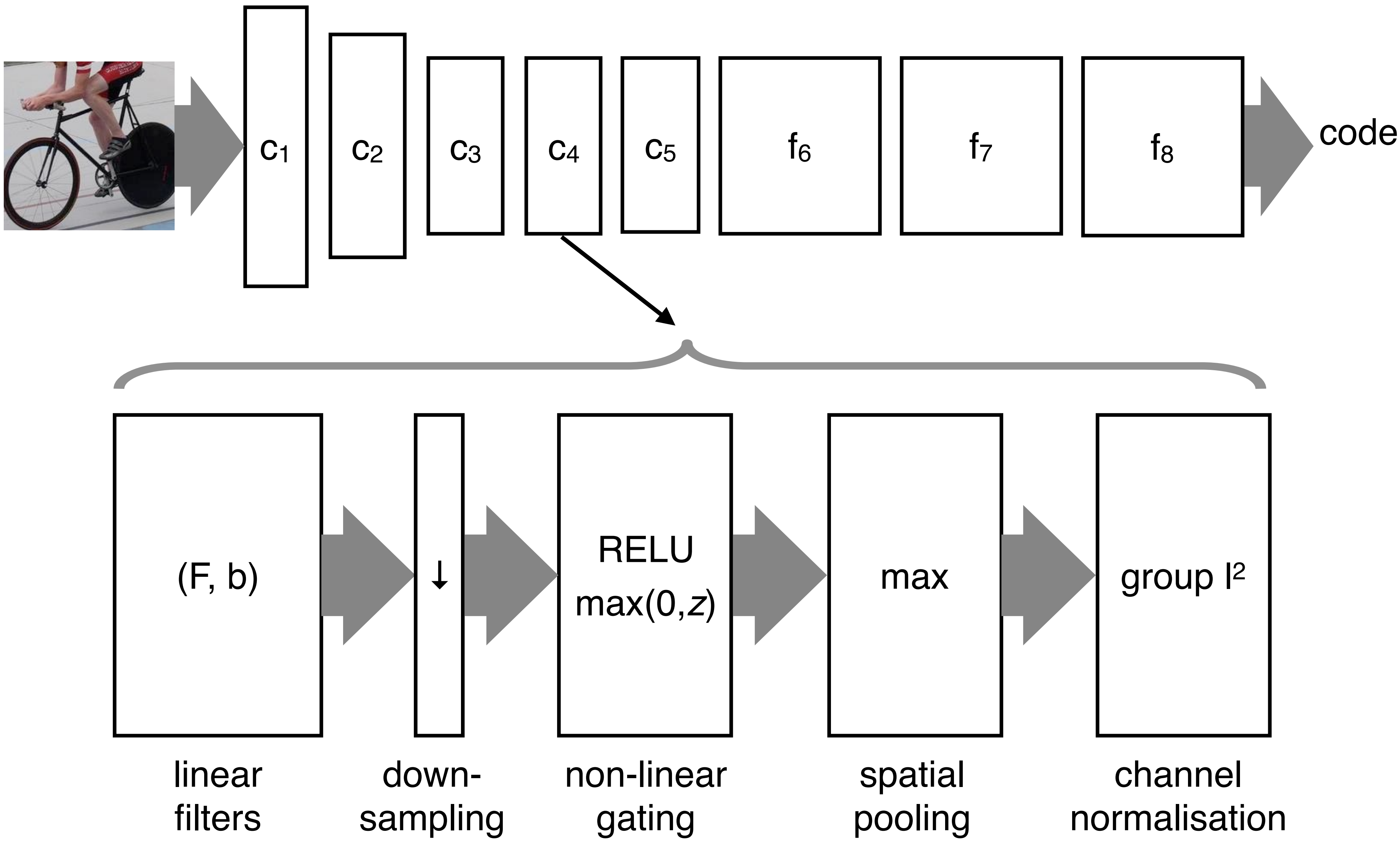


# Convolutional neural network (CNN)




- **From left to right**
  - decreasing spatial resolution
  - increasing feature dimensionality
- **Fully-connected layers**
  - same as convolutional, but with $1 \times 1$ spatial resolution
  - contain most of the parameters

# Convolutional layers



# 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

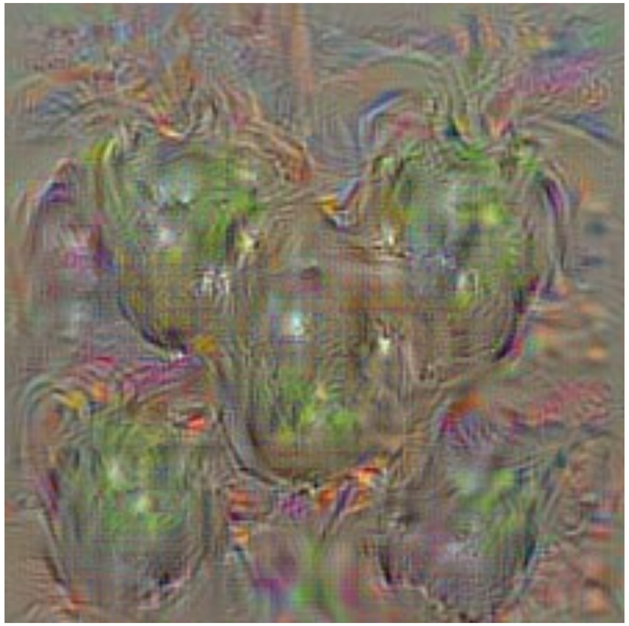
What do CNNs learn?

# Deep dreams

[Simonyan et al. 14]

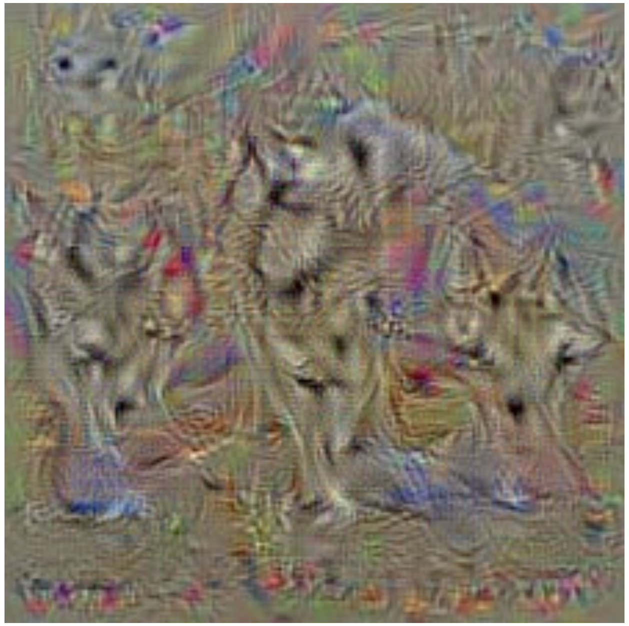
- Invert a CNN by finding the image that maximises the output of a class

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



bell pepper

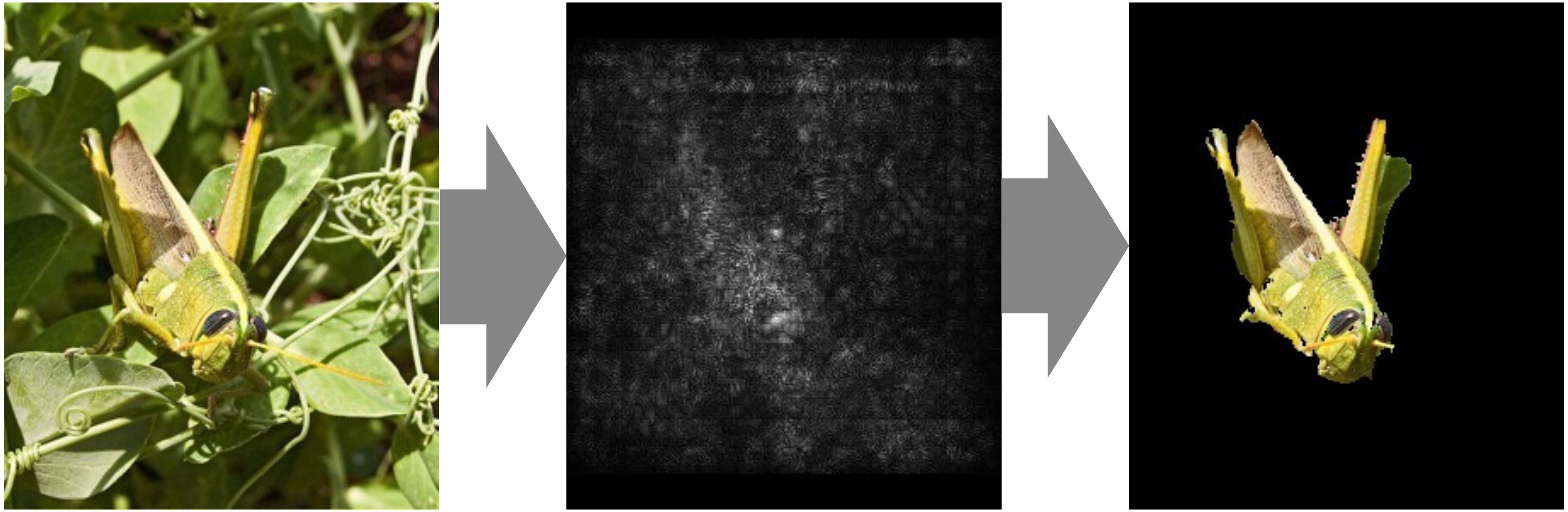
ostrich



husky

# Weakly-supervised learning

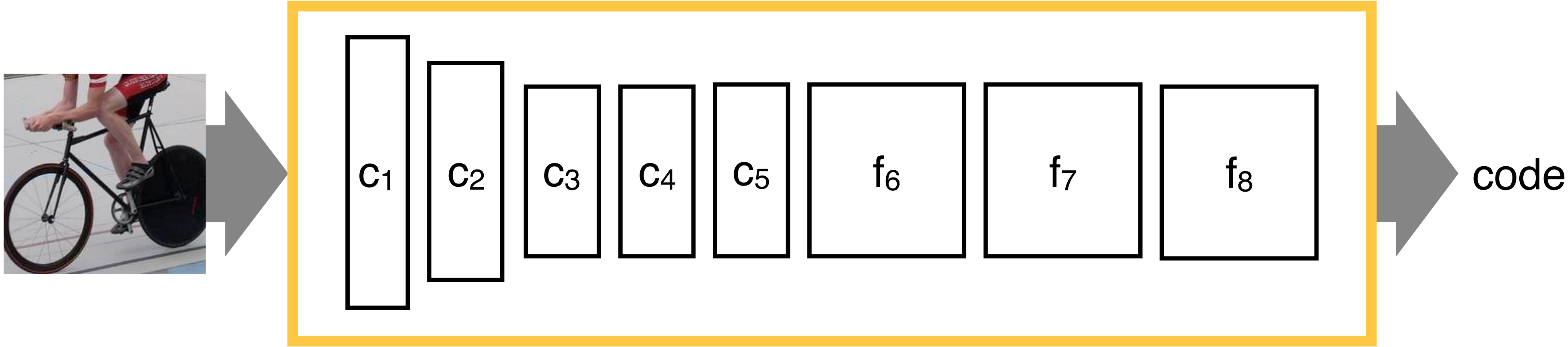
- This can be used to segment objects
- Remarkably, no object segmentation or bounding box is given during training



input image input saliency grabcut

# CNNs as general purpose encoders



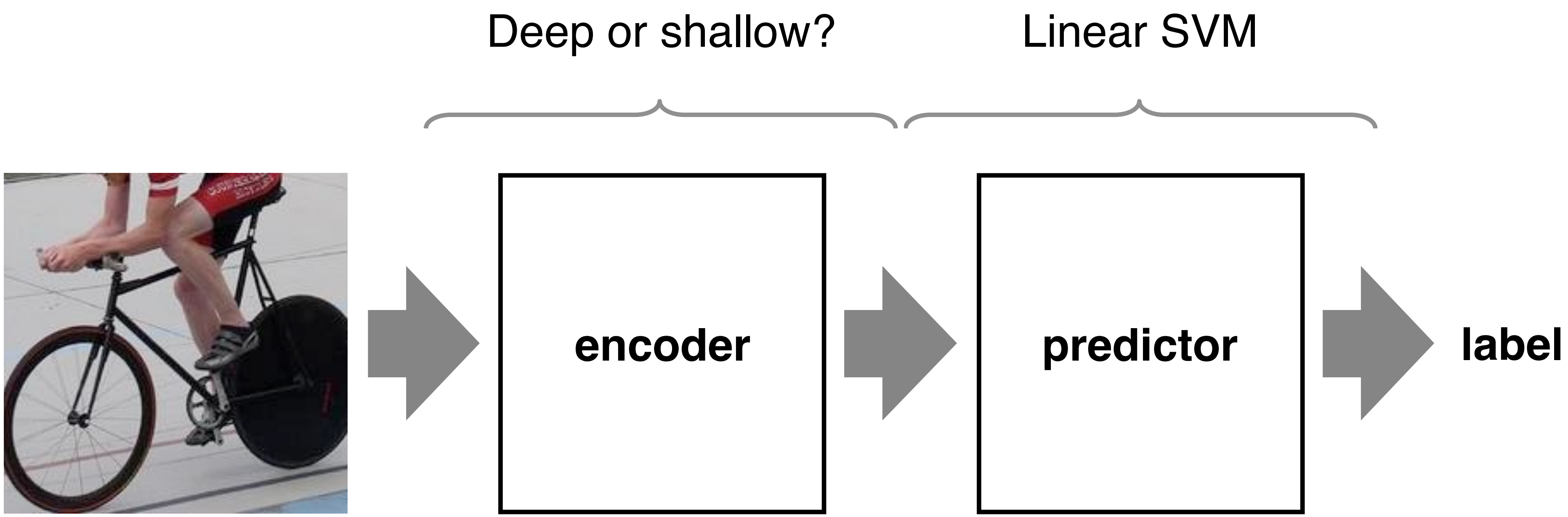



- **Pre-trained CNN encoders**
  - Architecture trained on ~ 1M ImageNet images
  - Last softmax layer chopped off
  - Output used as image encoding
- **Used as general-purpose features**
  - Applied to PASCAL VOC, Caltech, UCSD Birds, MIT Scene 67, …
  - [Zeiler & Fergus, DeCAF, Caffe, …]

# Return of the devil



## Evaluating shallow and deep encoders



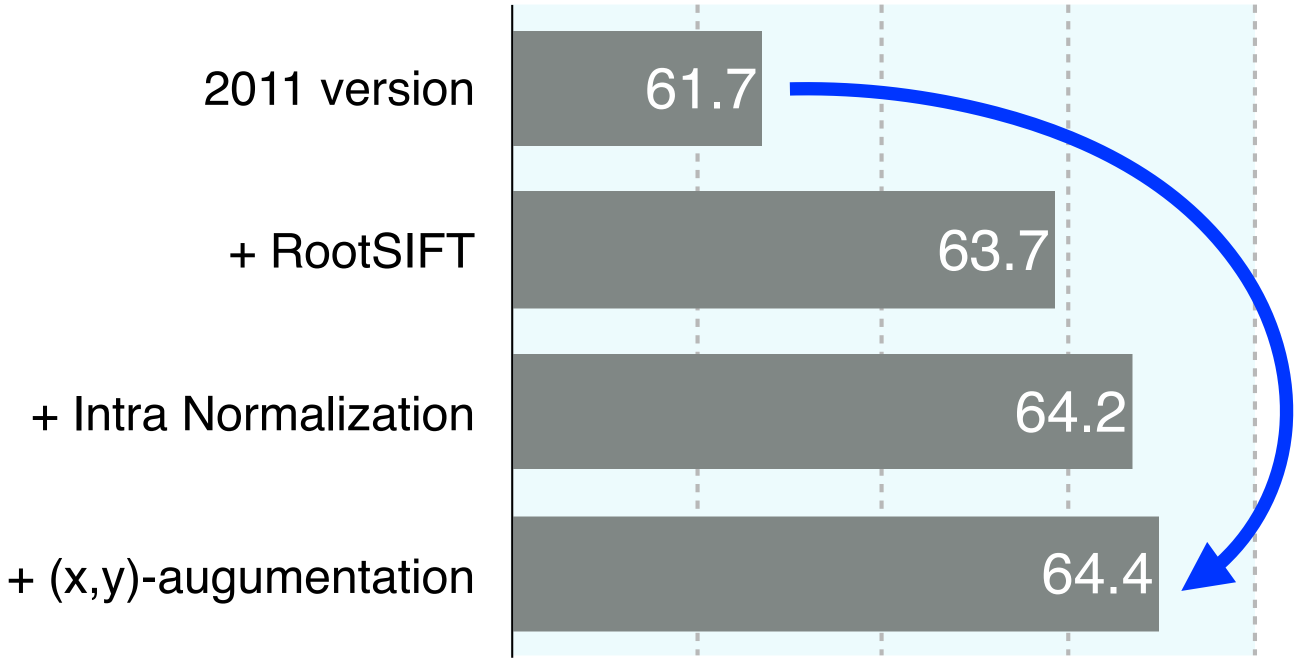

- **Shallow encoder**
  - Further Improved Fisher Vector
- **Deep encoders**
  - CNN Fast (CNN-F)
  - CNN Medium (CNN-M)
  - CNN Slow (CNN-S)

[Chatfield *et al.* 2014 - under revision]

# Shallow visual encoding

**Pumping Fisher Vectors**

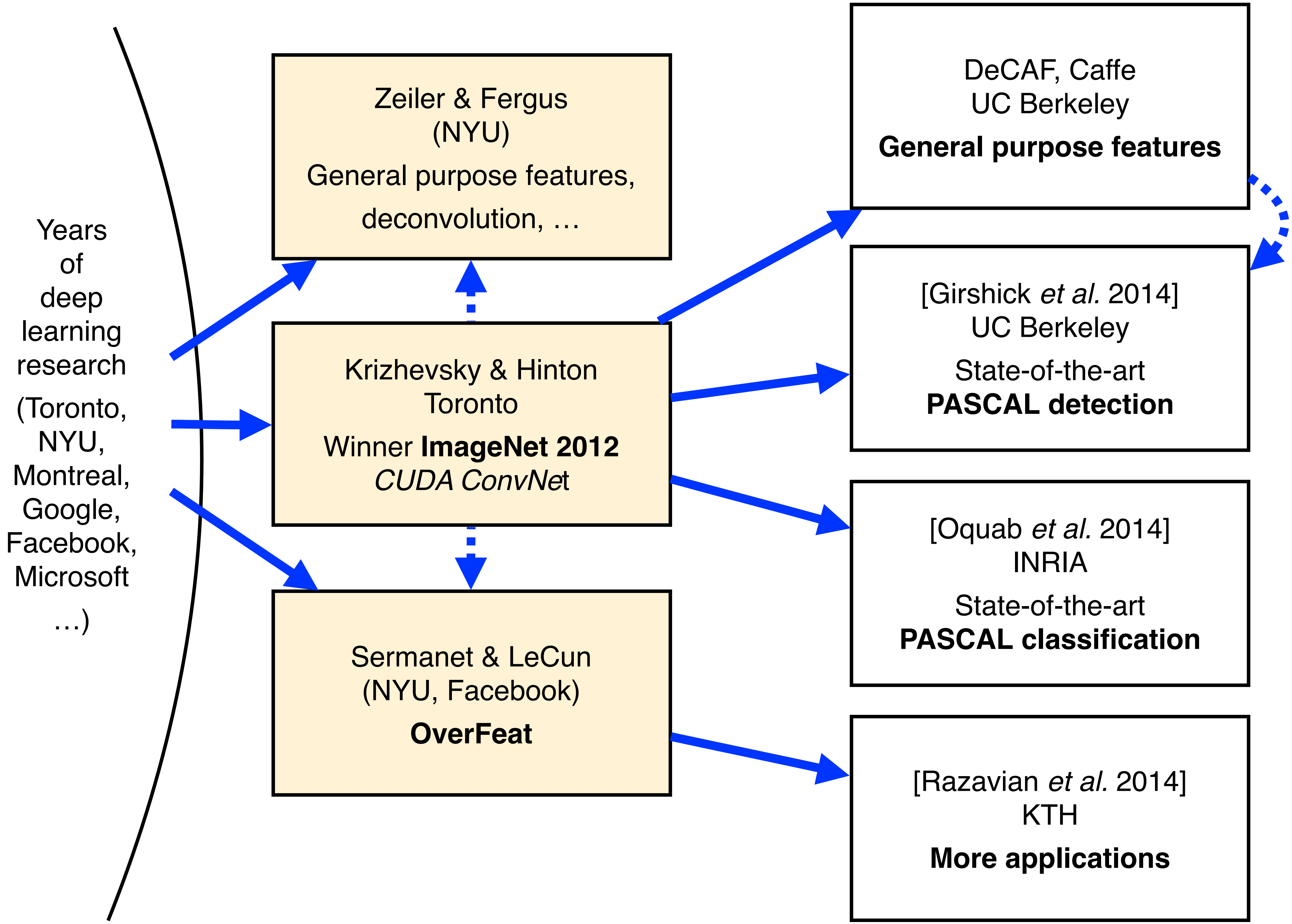
PASCAL VOC 2017




mAP (%)

A significant improvement compared to the old baseline

# Deep visual encodings



# Reference implementations

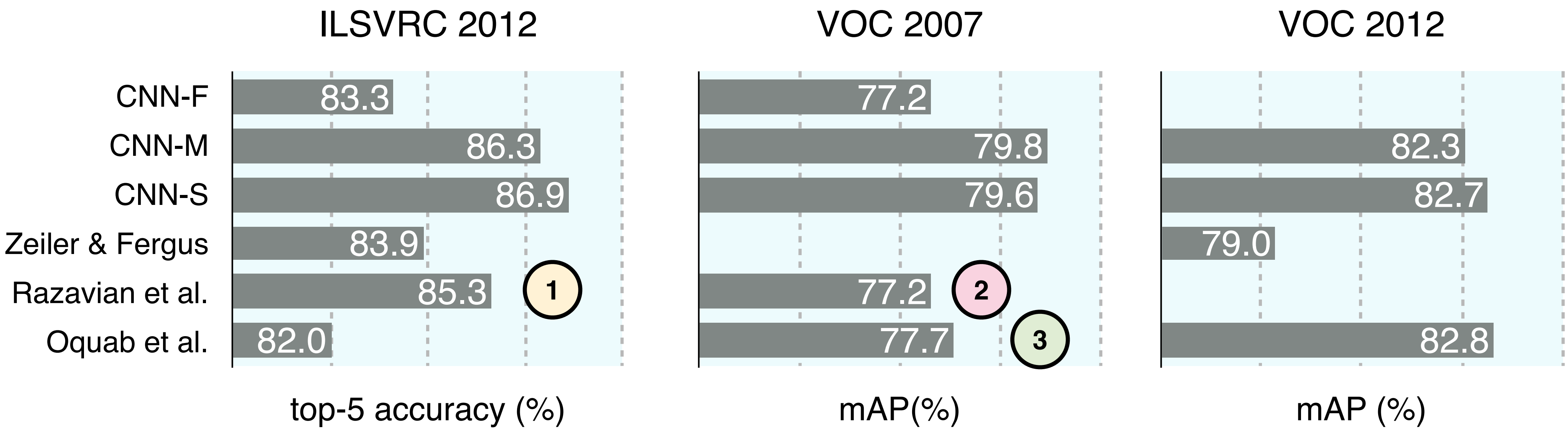


| Name | Speed | s/image | Similar to |
|---|---|---|---|
| CNN-S | Slow | 1.82 | OverFeat |
| CNN-M | Medium | 1.33 | Zeiler & Fergus |
| CNN-F | Fast | 0.6 | Krizhevsky & Hinton |

[Karen Simonyan]

- **Types**
  - Inspired by existing implementations
  - Trained in-house using one uniform setup
- **Main differences**
  - Number of filters
  - downsampling factors

# Reference implementations performance




**1**

Excellent performance[1] on the ImageNet challenge data (~ state-of-the-art).

**2**

CNN-F,M,S use a modified Caffe

Yet better than other using DeCAF, Caffe, OverFeat

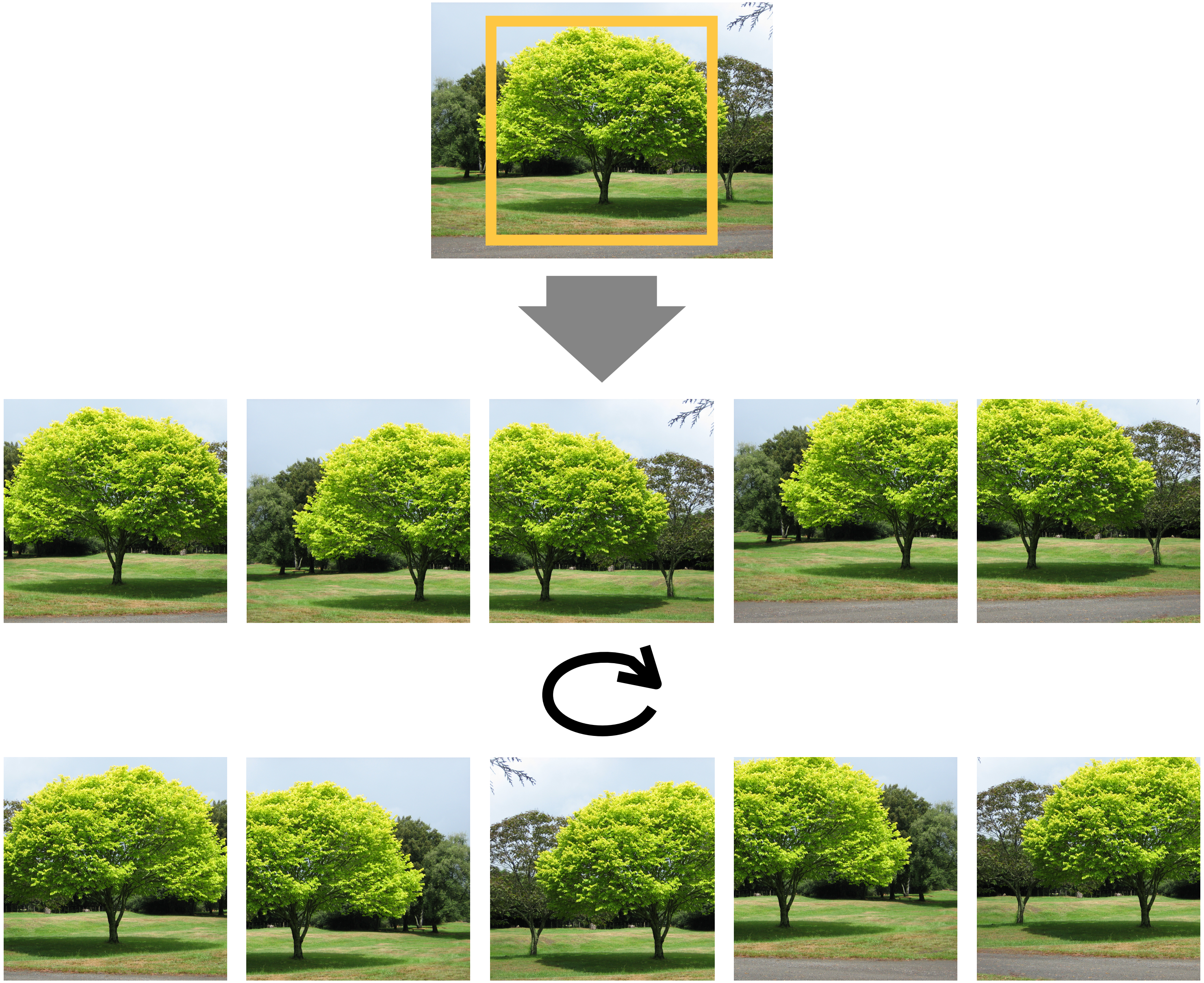
**3**

**Simpler** and yet **better** or **equal** than alternative ways of using the encoders.

[1] A bit better than OverFeat, probably due to slightly different data augmentation (crops from the whole image & test set augmentation)

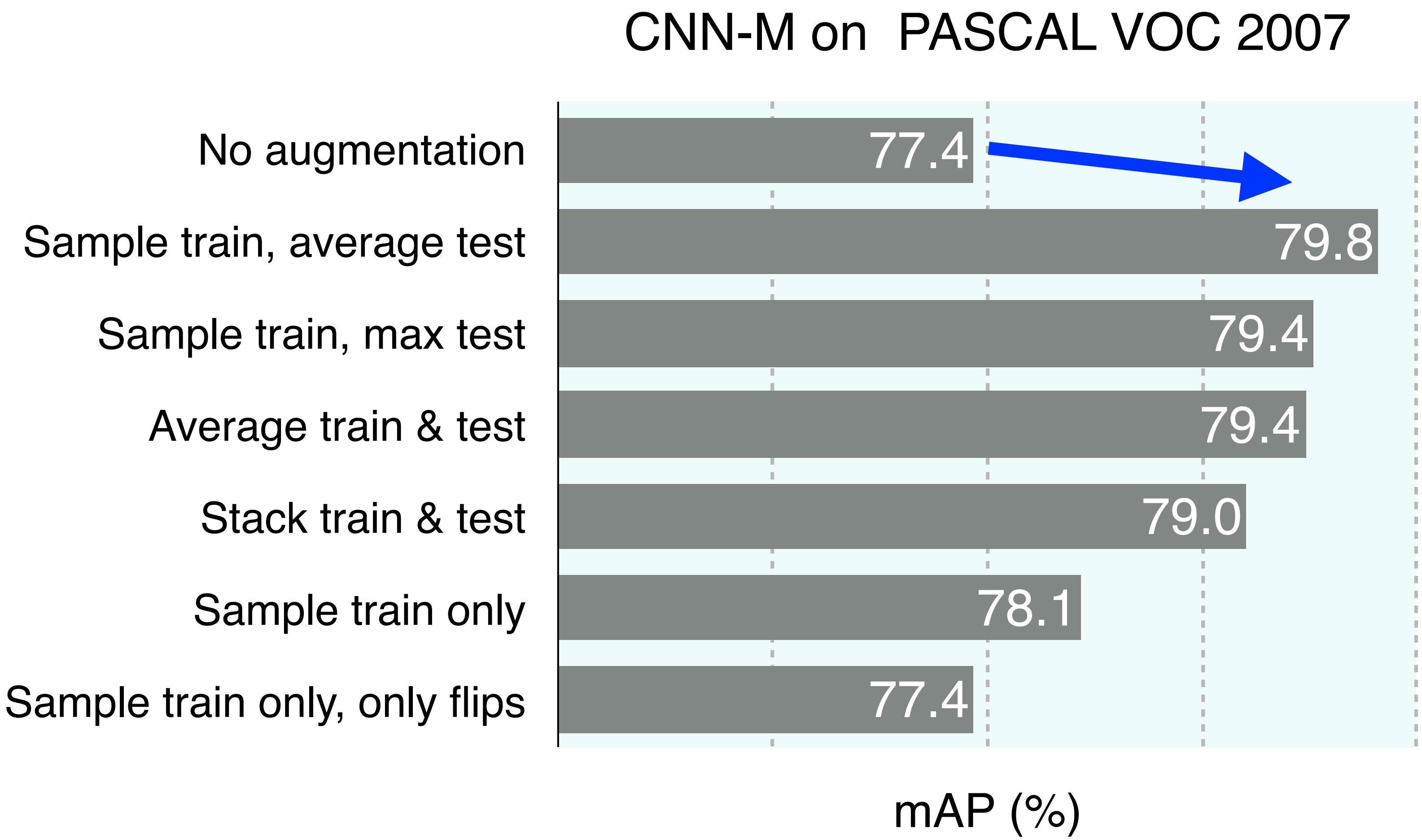
# Data augmentation

- Augment the training data by adding jittered versons of each image



# Data augmentation: CNNs

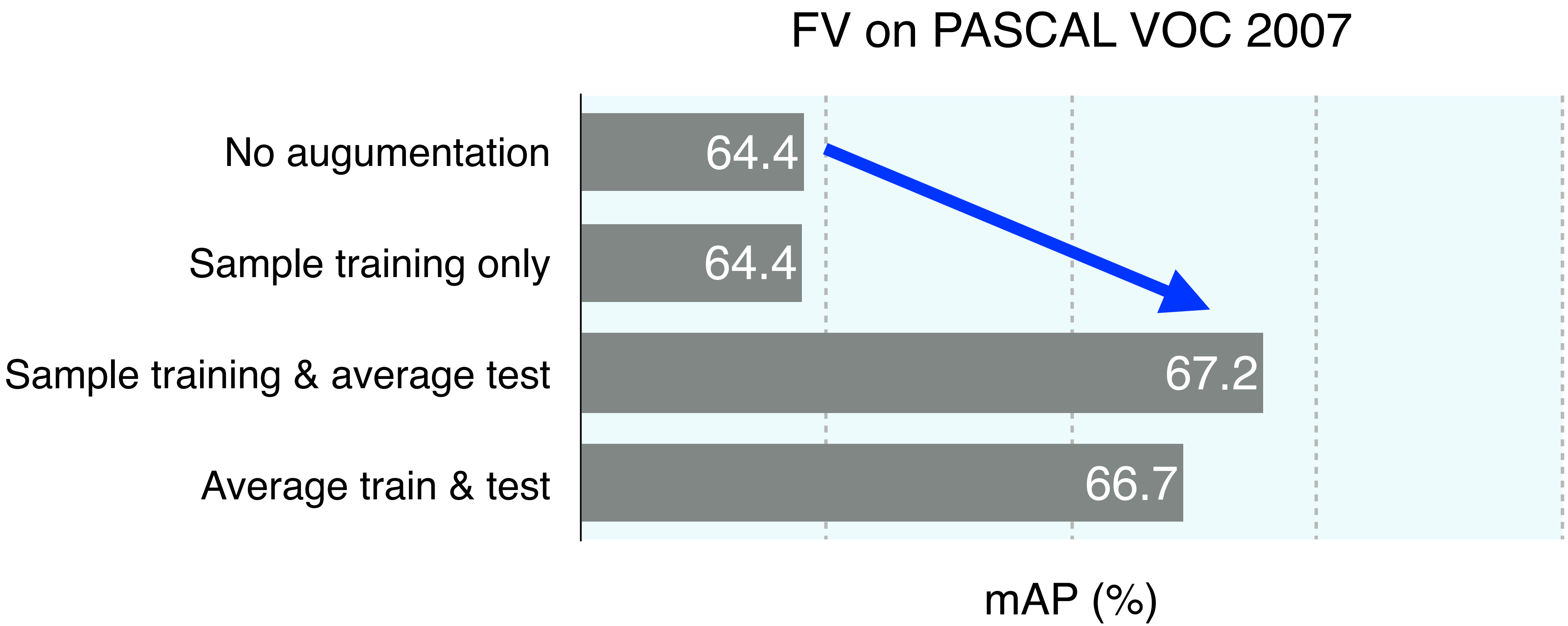





- **Best practices**
  - **Sample** *training* and **average** *test*
  - Only flipping is insufficient
  - Further augmentation has diminishing returns

# Data augmentation: Fisher Vectors



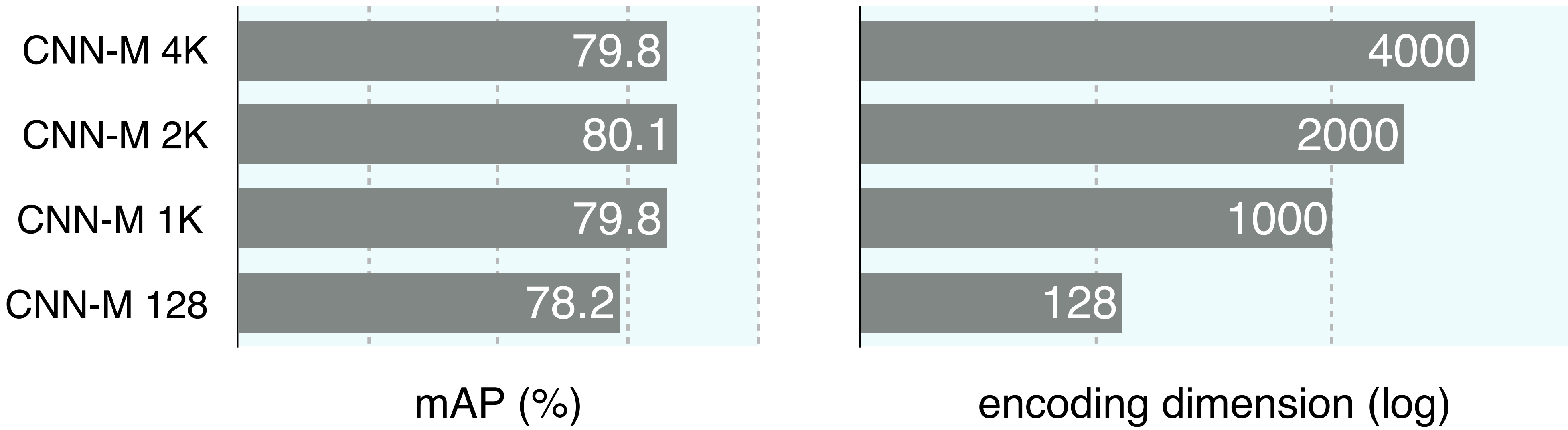



- **Porting augmentation from CNNs to FV**
  - Similar benefits observed
  - Augmenting test data is essential
  - See also [Paulin *et al.* CVPR 2014]

# Dimensionality reduction



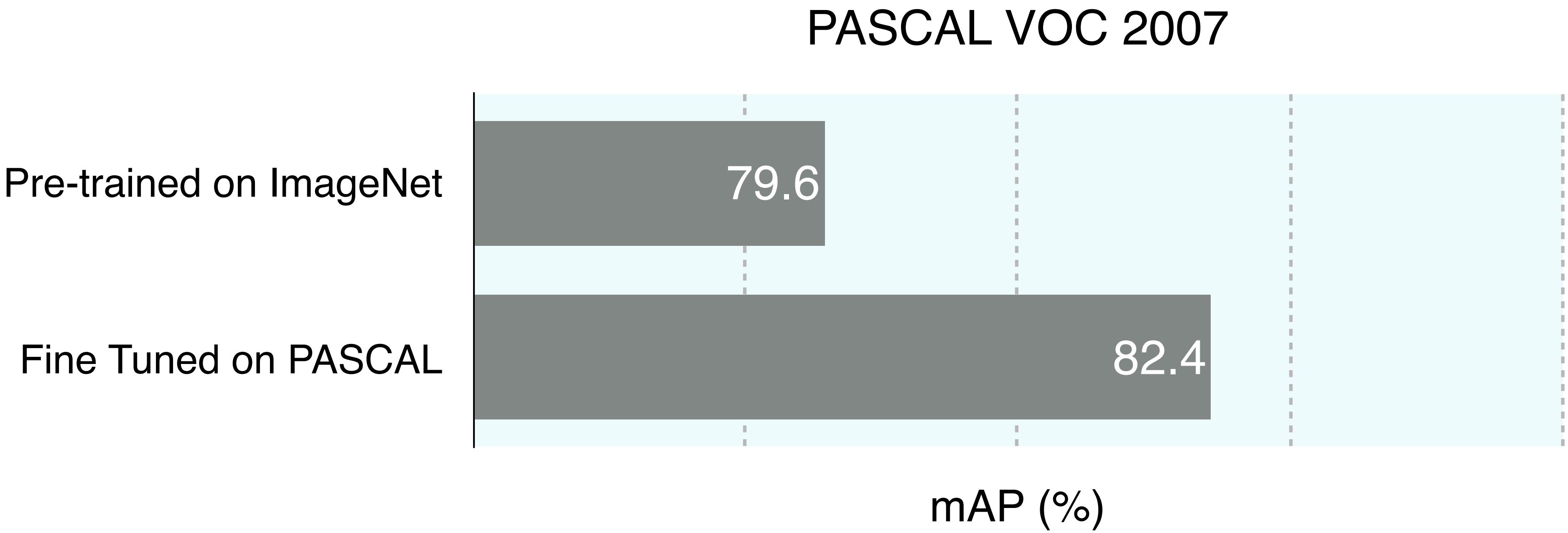
Tested on PASCAL VOC 2007




- Encodings are often **highly redundant**
- **CNN**
  - **reduce dimension 31 times**, ~ same performance
  - (re-learn last layer using a multi-class loss and PASCAL VOC)
- **FV dimensionally reduction**
  - similar compression possible
  - (use e.g. WSABIE [Weston *et al.* 2011])

# CNN fine-tuning

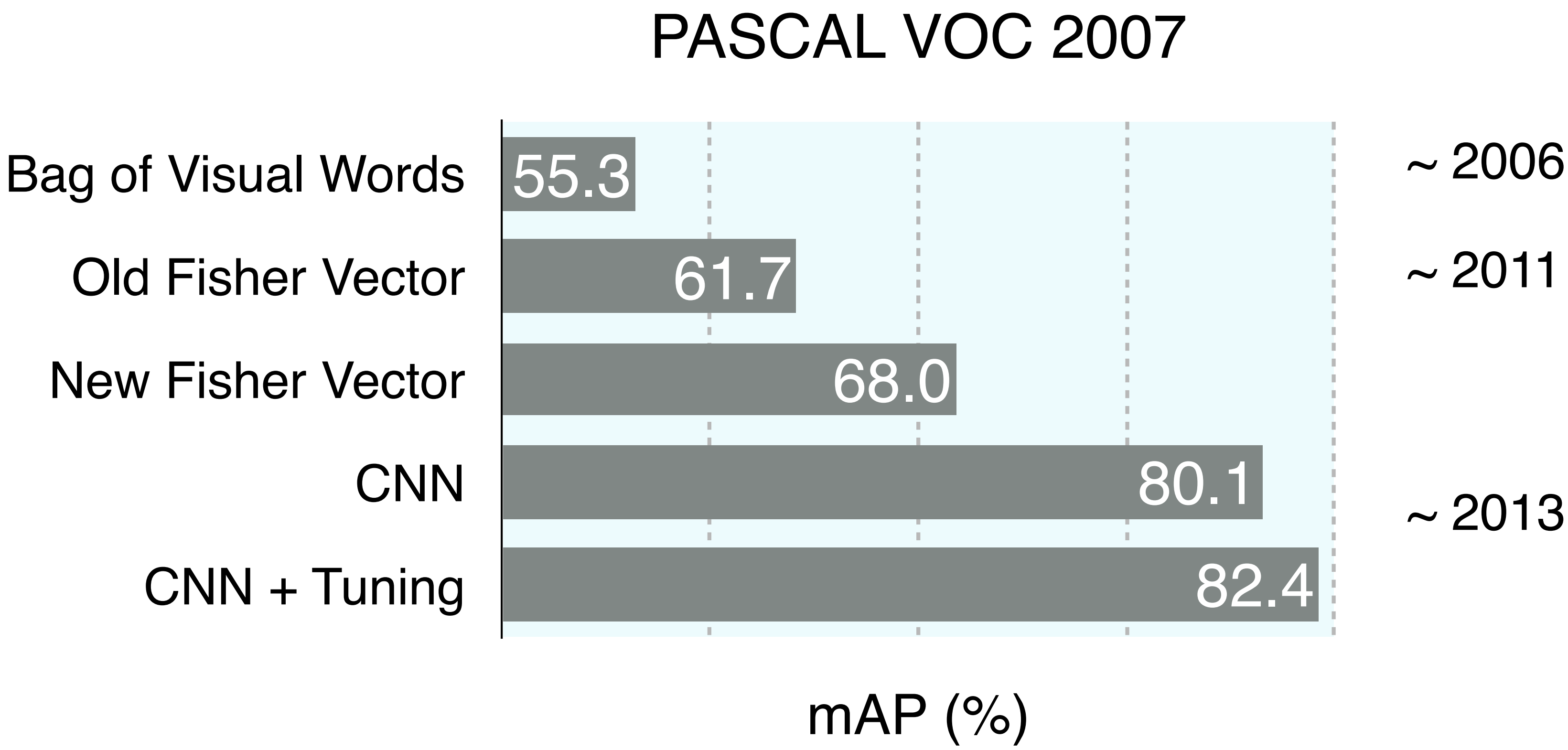





- Pre-trained CNNs can be **tuned on target dataset**
  - Use target data to provide more training images
  - Remark: tuning in PASCAL requires a multi-class loss
- Often (but not always) yields a nice improvement

# Deep *vs.* shallow





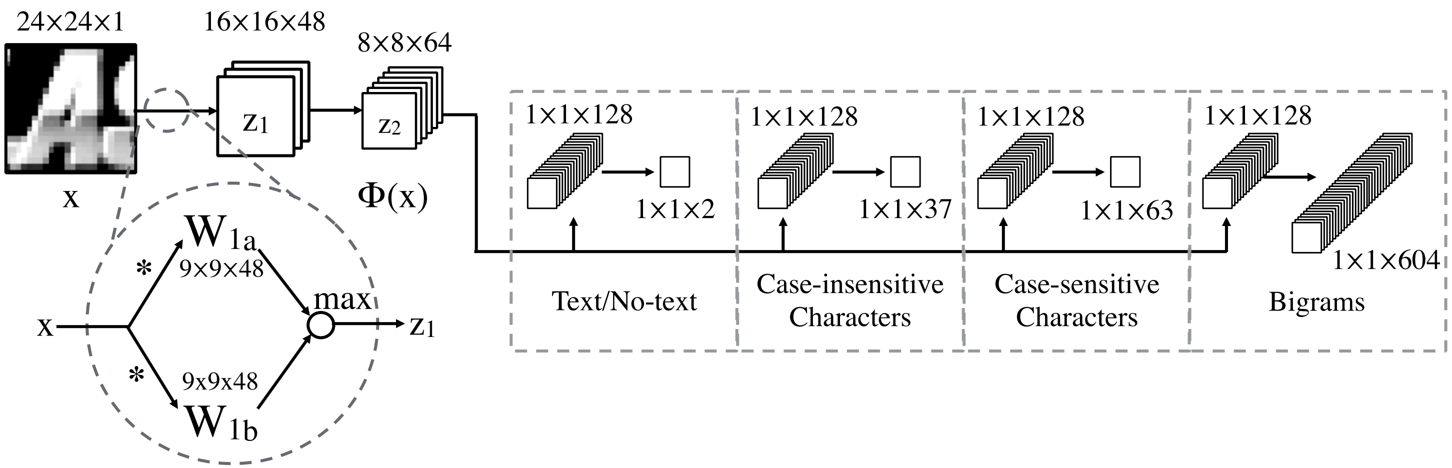

- **CNNs**
  - Best shallow encodings
  - Are expensive to train, but fast to evaluate
  - Do provide low-dimensional, general-purpose codes
  - Will definitely get much better

# CNNs are versatile



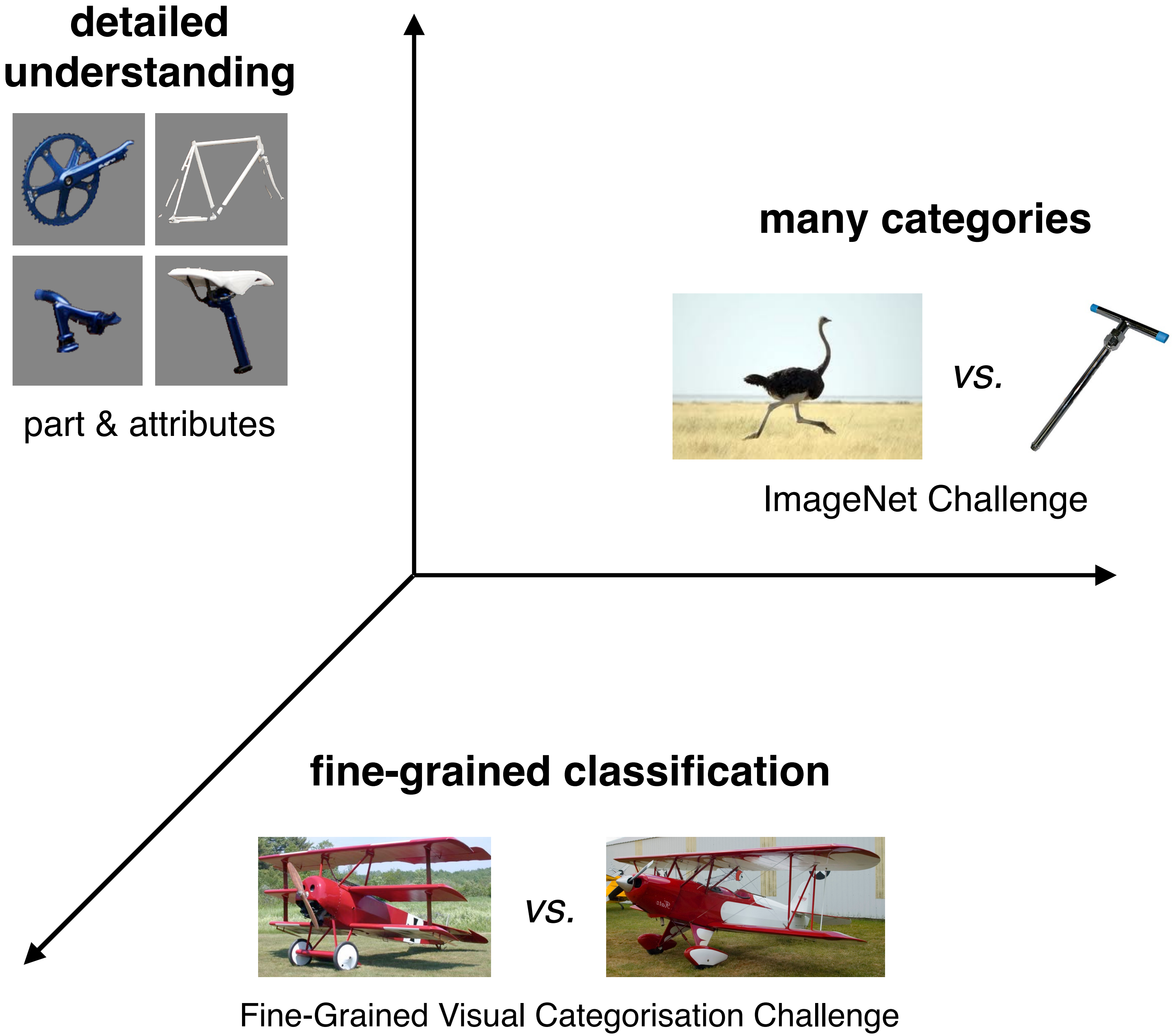
## Deep text spotting






[Jadreberg *et al.* 2014 (under revision)]

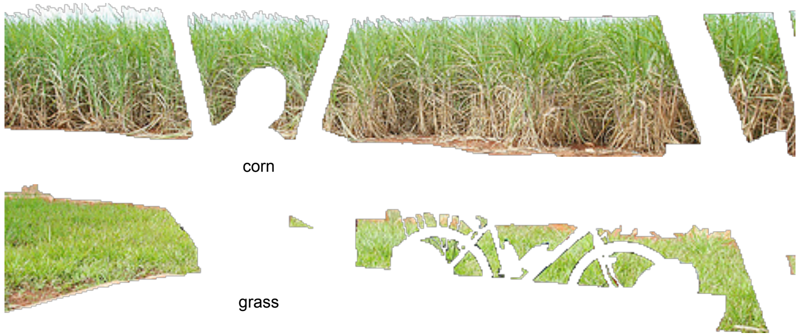
# Beyond image-based modelling


detailed understanding
part & attributes
many categories
vs.
ImageNet Challenge
fine-grained classification
vs.
Fine-Grained Visual Categorisation Challenge

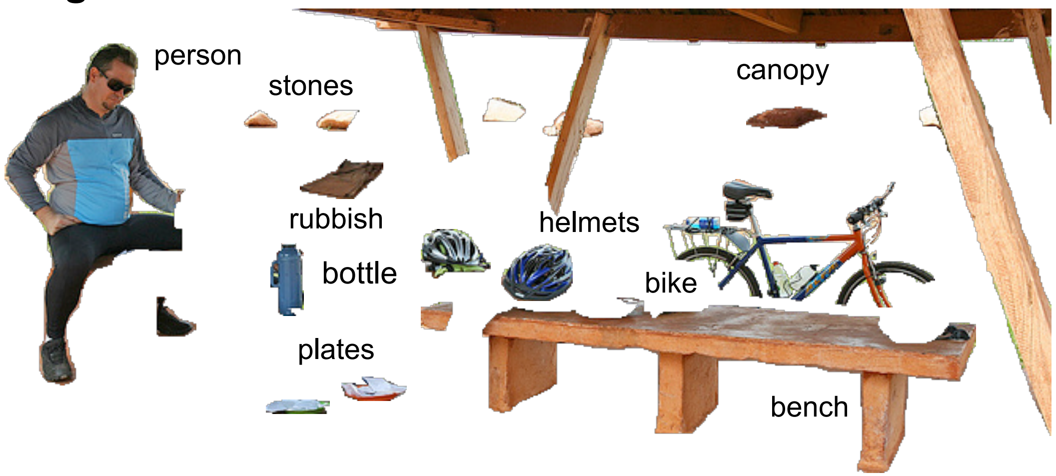
# Detailed image understanding

- **Breadth**
  - large visual vocabulary
  - completeness
- **Depth**
  - compositionally
  - parts and attributes
- **Abstraction**
  - surfaces, objects
  - categories, subcategories

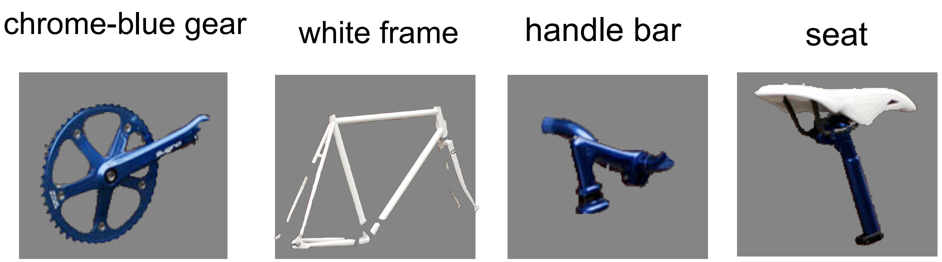
**stuff**



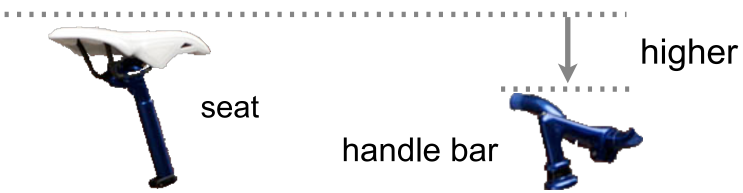

**things**




**parts, materials, colours, ...**



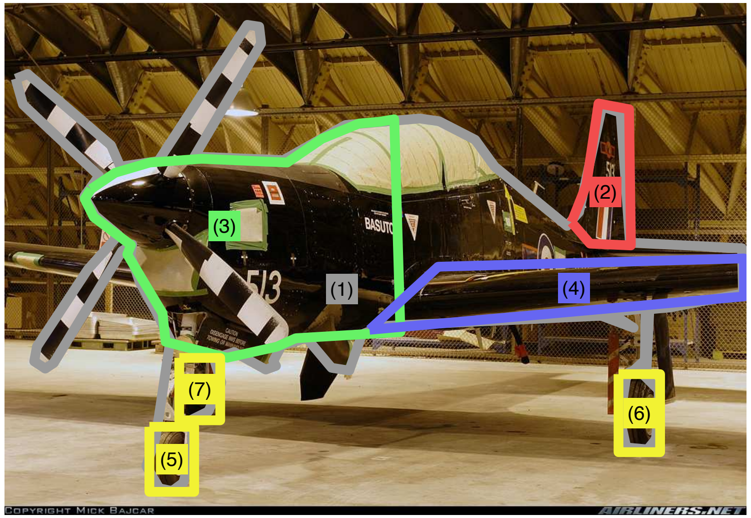

**relationships**



# Objects in detail



[Vedaldi *et al.* 2014]




**1 aeroplane** facing-direction: SW; is-airliner: no; is-cargo-plane: no; is-glider: no; is-military-plane: yes; is-propellor-plane: yes; is-seaplane: no; plane-location: on ground/water; plane-size: medium plane; wing-type: single wing plane; undercarriage-arrangement: one-front-two-back; airline: UK–Air Force; model: Short S-312 Tucano T1 2 **2 vertical stabilizer** tail-has-engine: no-engine **3 nose** has-engine-or-sensor: has-engine **4 wing** wing-has-engine: no-engine **5 undercarriage** cover-type: retractable; group-type: 1-wheel-1-axle; location: front-middle **5 undercarriage** cover-type: retractable; group-type: 1-wheel-1-axle; location: back-left **5 undercarriage** cover-type: retractable; group-type: 1-wheel-1-axle; location: back-right.

- **Describing objects**: beyond object recognition and detection
- Requires data annotated with detailed object properties
  - parts & attributes
  - category, instance, and time-dependent properties

# OID Aircraft

Detailed and Fine-Grained and Understanding

7400K aircraft images with detailed annotations

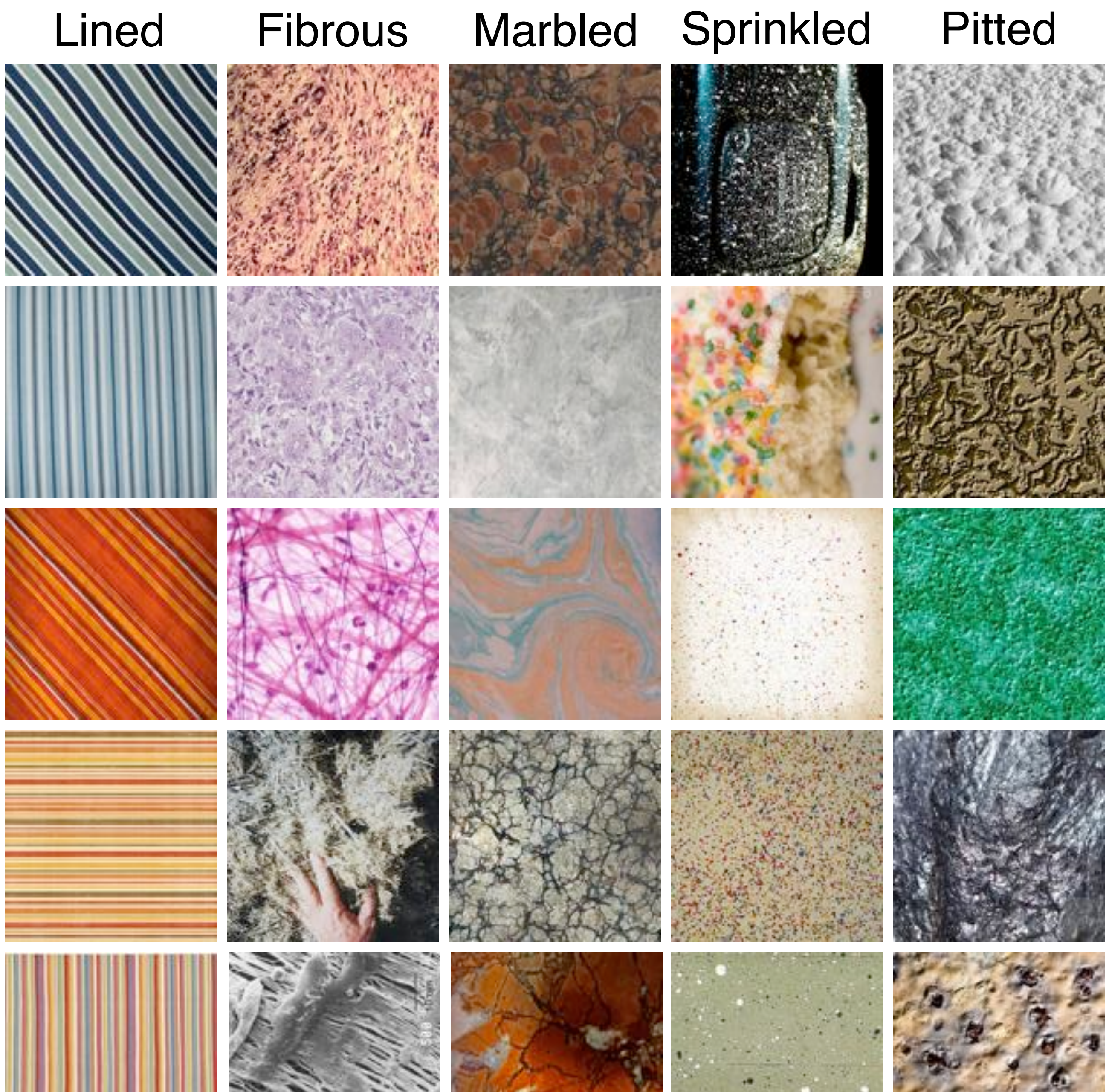
# Spin-off: FGVC Competition 2013



# Describable Texture Dataset



[Cimpoi *et al.* 2014]




**Describable Textures**

47 texture words

5,000 texture images

Each texture described by a combination of words

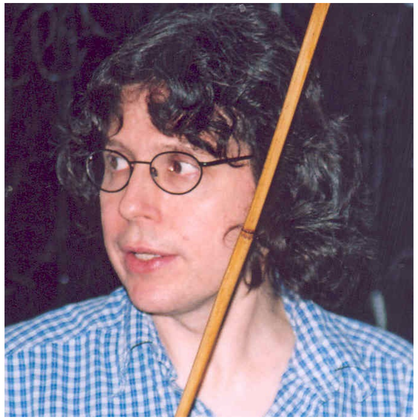
Byproduct: **state-of-the-art material recognition**

# Credits

Karen Simonyan

Ken Chatfield

Omkar Parkhi



Andrew Zisserman

We are seeking a postdoctoral researcher
on image understanding and deep learning

# 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 understanding 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.