Learning a Compact Image Code for Efficient Recognition of Novel Classes

Lorenzo Torresani

Joint work with:
- Alessandro Bergamo (Dartmouth)
- Andrew Fitzgibbon (Microsoft Research Cambridge)

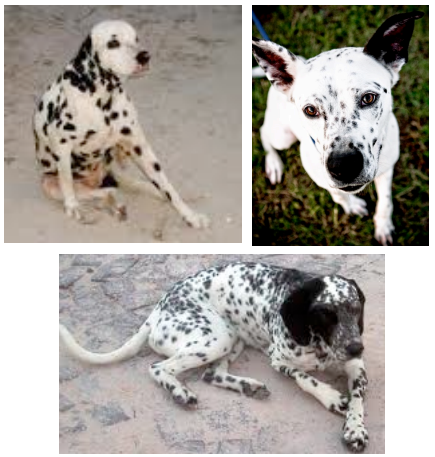
Problem statement: novel-class search

- *Given:*

large image database
(e.g., 10 million photos)

+

user-provided images
of an object class

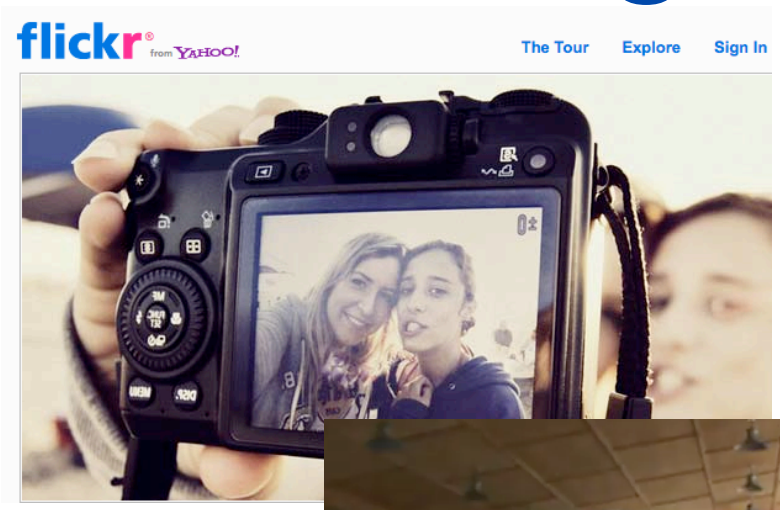


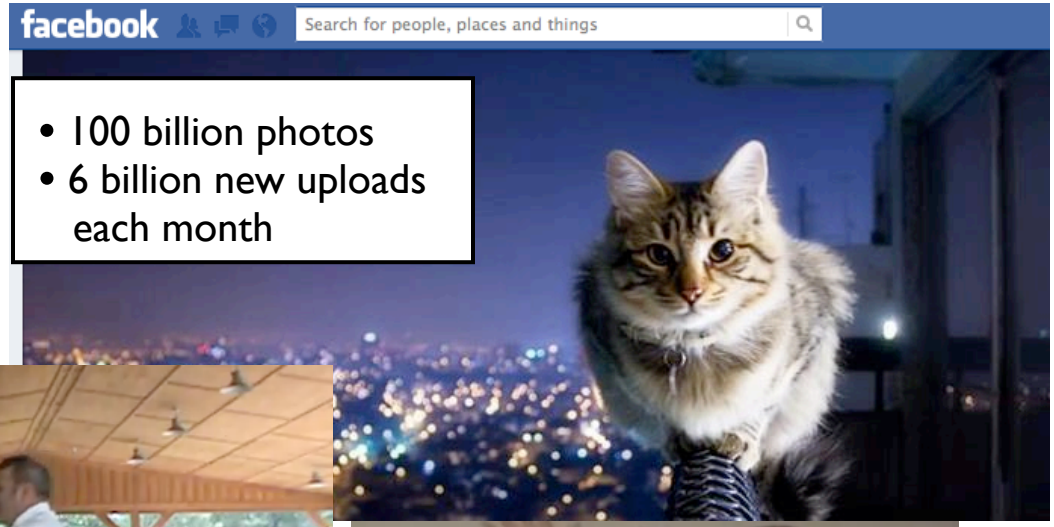
- *Want:*

database
images
of this class

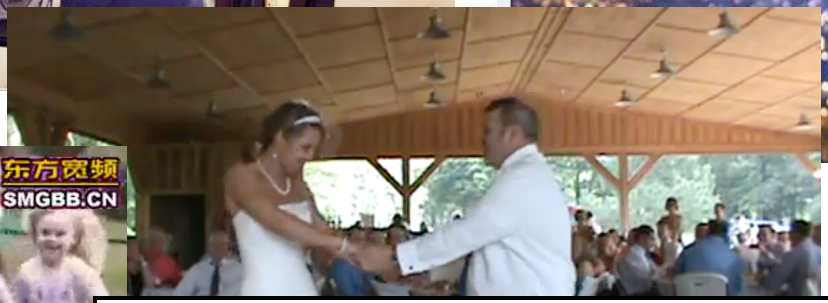
- no text/tags available
- query images may represent a **novel** class

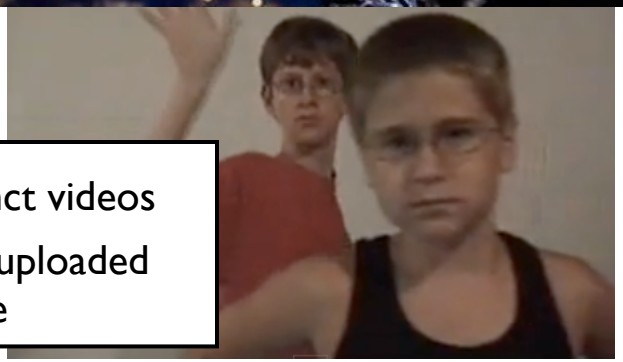
Big Image Data





- 100 billion photos
- 6 billion new uploads each month





YouTube™

- more than 120M distinct videos
- 72 hours of video are uploaded to YouTube every minute

Interactive visual search in these collections requires:

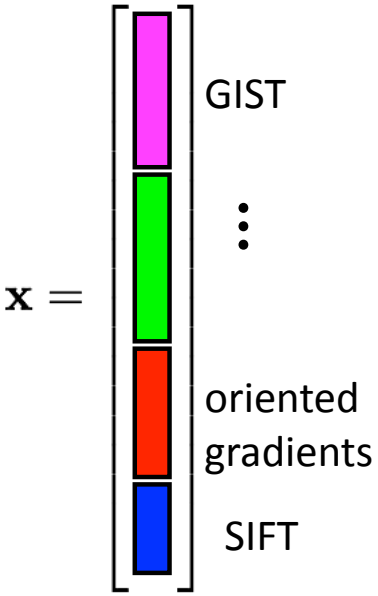
- the ability to efficiently train and test **novel** visual classes at search time

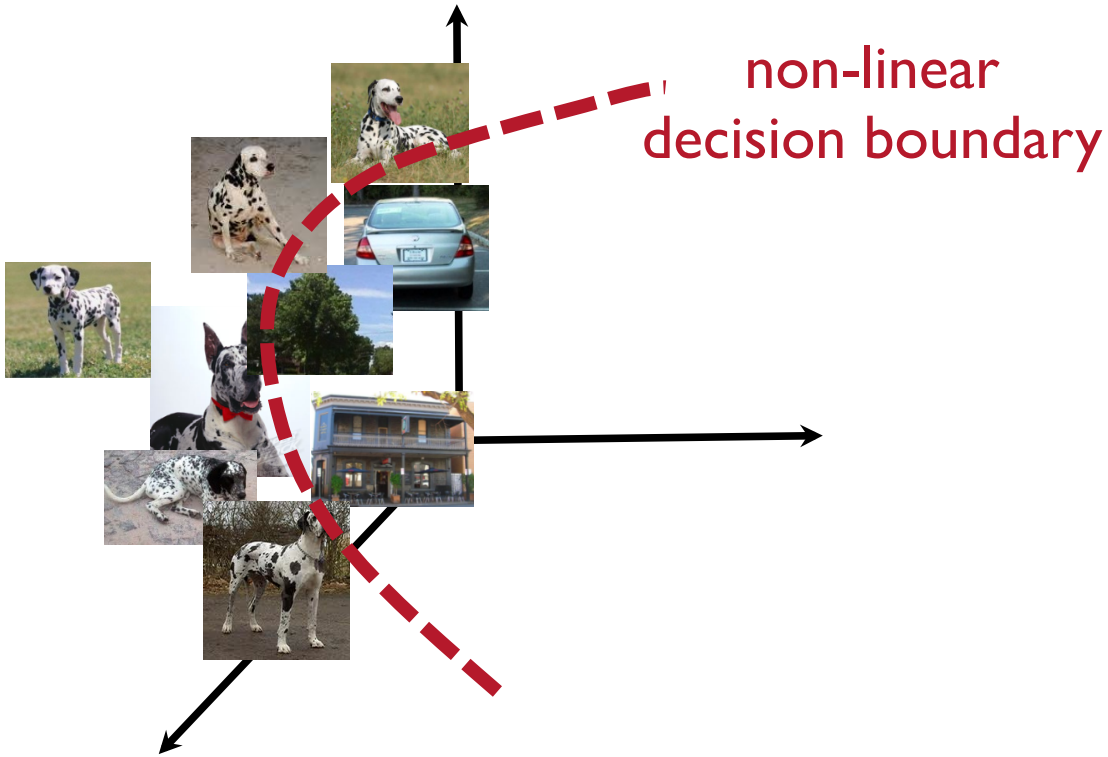
Technical requirements of novel-class search

- The object classifier must be learned on the fly from few examples
- Recognition in the database must have low computational cost
- Image descriptors must be compact to allow storage in memory

State-of-the-art in object classification

Winning recipe: many features + non-linear classifiers (e.g. [Gehler and Nowozin, CVPR'09])





Multiple kernel combiners for novel-class search?

Classification output is obtained by combining many features via non-linear kernels (e.g. the LP-β of [Gehler and Nowozin, CVPR'09]):

$$h(\mathbf{x}) = \sum_{f=1}^{F} \beta_f \sum_{n=1}^{N} k_f(\mathbf{x}, \mathbf{x}_n)\alpha_n + b$$

sum over features ← → sum over training examples

Unsuitable for our needs due to:

- large storage requirements (typically over 20K bytes/image)
- costly evaluation (requires *query-time* kernel distance computation *for each test image*)
- costly training (1+ minute for *O*(10) training examples)

Our approach [Torresani et al., 2010; Bergamo et al., 2011]

Key-idea: ***represent*** each image as the binarized output of a large number of ***predefined*** multiple kernel classifiers

image

low-level features

$\mathbf{x} =$ GIST | self-similarity descriptor | oriented gradients | SIFT

our binary descriptor

$$\Phi(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \dots \\ \phi_C(\mathbf{x}) \end{bmatrix} \in \{0,1\}^C$$

This is the only thing we store in the database

The c-th bit is the binarized output of a ***pre-learned*** LP-β for the c-th basis class:

$$\phi_c(\mathbf{x}) = \mathbf{1}\left[\sum_{f=1}^{F} \beta_f^c \sum_{n=1}^{N} k_f(\mathbf{x}, \mathbf{x}_n^c)\alpha_n^c\right]$$

- ***Compact***: only C bits per image
- A linear combination of these features is an ***efficient*** multiple kernel combiner

$$\mathbf{w}^T\Phi(\mathbf{x}) = \sum_{c=1}^{C} w_c \, \mathbf{1}\left[\underbrace{\sum_{f=1}^{F} \beta_f^c \sum_{n=1}^{N} k_f(\mathbf{x}, \mathbf{x}_n^c)\alpha_n^c}\right]$$

trained at **query-time**

LP-β trained and evaluated **before** the creation of the database

Method overview

1. *Offline learning:*
   training the basis classifiers $\phi_1, \ldots, \phi_C$
   defining the compact representation $\Phi(\mathbf{x}) \in \{0,1\}^C$

2. *Query-time learning:*
   using the binary code for recognition and retrieval

training examples of **novel** class

$\Phi(\mathbf{x_1})$ ... $\Phi(\mathbf{x_N})$

learn a linear classifier on $\Phi(\mathbf{x})$

$$g^{\texttt{duck}}(\Phi(\mathbf{x})) = \sum_{c=1}^{C} w_c^{\texttt{duck}} \phi_c(\mathbf{x})$$

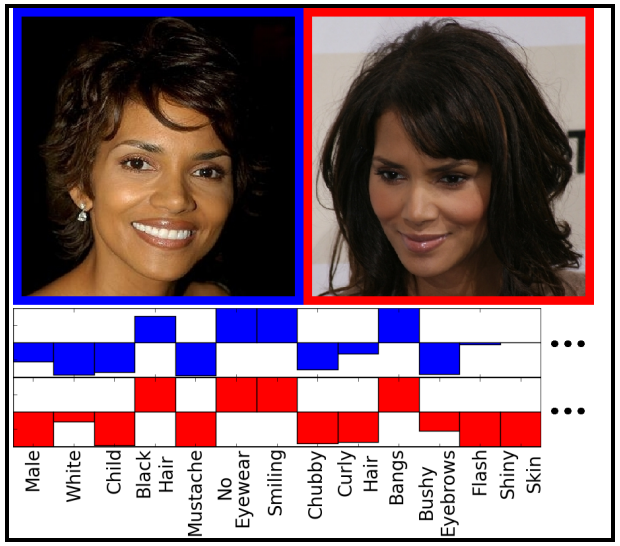
Related work

- Attribute-based recognition:

[Lampert et al., CVPR'09]

[Farhadi et al., CVPR'09]

[Kumar et al., ICCV'09]



- hand-specified visual properties correlated to the classes to recognize
- used for recognition in specific domains

How do we define the basis classifiers?

- PiCoDes (Picture Codes / Pico-descriptors) [Bergamo et al., 2011]:

we want to choose $\phi_1(\mathbf{x}), \ldots, \phi_c(\mathbf{x})$ such that the ***linear*** classification model

$$g(\Phi(\mathbf{x}); \mathbf{w}) = \sum_{c=1}^{C} w_c \phi_c(\mathbf{x})$$

enables recognition of ***many classes*** with ***good accuracy***

The descriptor-learning goal

- *Given:*
  training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N$,
  each belonging to one of K classes (where K is large).

- *Want:*
  learn C multiple kernel combiners $\phi_1(\mathbf{x}), \ldots, \phi_c(\mathbf{x})$
  s.t. there exist K linear classifiers $\underbrace{(\mathbf{w}_1, b_1), \ldots, (\mathbf{w}_K, b_K)}$
  satisfying

parameters of linear classifiers for the K classes

$$\mathbf{w}_k^\top \Phi(\mathbf{x}_i) + b_k > 0 \qquad \text{if } \mathbf{x}_i \text{ belongs to class } k$$

$$\mathbf{w}_k^\top \Phi(\mathbf{x}_i) + b_k < 0 \qquad \text{otherwise}$$

A large margin formulation

- *Training set:*

  $\mathbf{x}_i$: image i

  $y_{i,k} \in \{-1, 1\}$: label $y_{i,k} = 1$ iff image i belongs to class k

- *Learning objective:*

descriptor produced for the *i*-th image

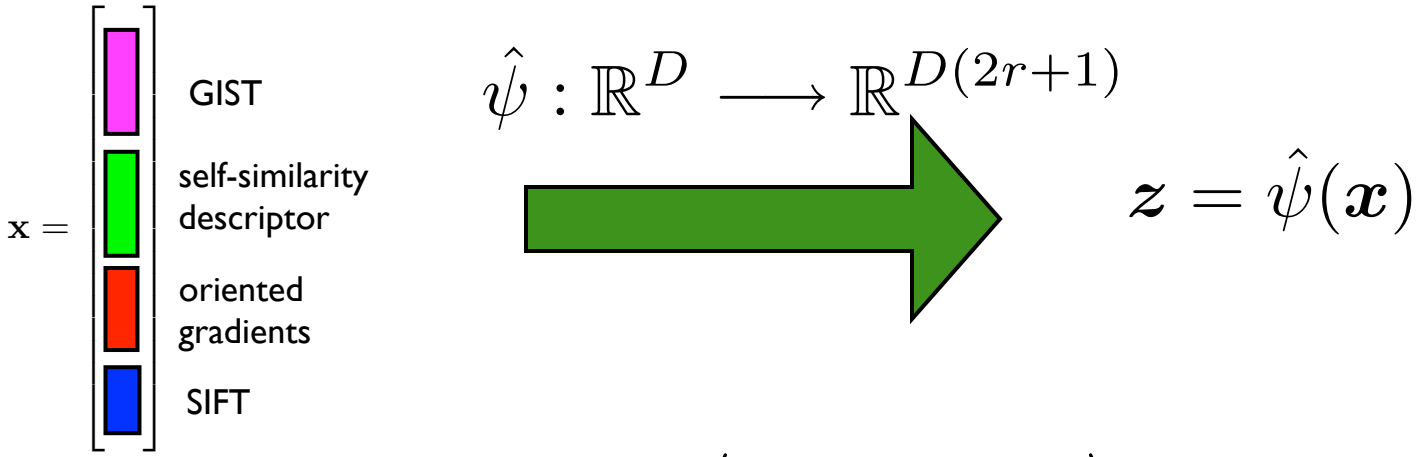
$$E(\Phi, \mathbf{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2}\|\mathbf{w}_k\|^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \ell\left[y_{i,k}(\mathbf{w}_k^\top \Phi(\mathbf{x}_i) + b_k)\right] \right\}$$

tradeoff between large margin and misclassification over the K training classes

$\ell[.]$: hinge function

Linearization of a multiple kernel combiner

- "Sidestepping" the kernel trick [Vedaldi and Zisserman, 2010]: for the family of additive kernels there exists an *explicit feature map* $\hat{\psi}$



such that $K(\mathbf{x}, \mathbf{x}') \approx \left\langle \hat{\psi}(\mathbf{x}), \hat{\psi}(\mathbf{x}') \right\rangle$ for small r (we use $r = 1$).

- Each basis classifier ϕ_c can be approximated as a binarized linear projection in the $3D$-dimensional space:

$$\phi_c(\mathbf{z}) = \mathbf{1}[\mathbf{a}_c^T \mathbf{z}]$$

The final learning objective

$$E(\mathbf{a}_{1..C}, \mathbf{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2}\|\mathbf{w}_k\|^2 + \right.$$

$$\left. \frac{\lambda}{N} \sum_{i=1}^{N} \ell \left[y_{i,k} \left(\sum_{c=1}^{C} w_{k,c} \boxed{\mathbf{1}[\mathbf{a}_c^T \mathbf{z}_i]} + b_k \right) \right] \right\}$$

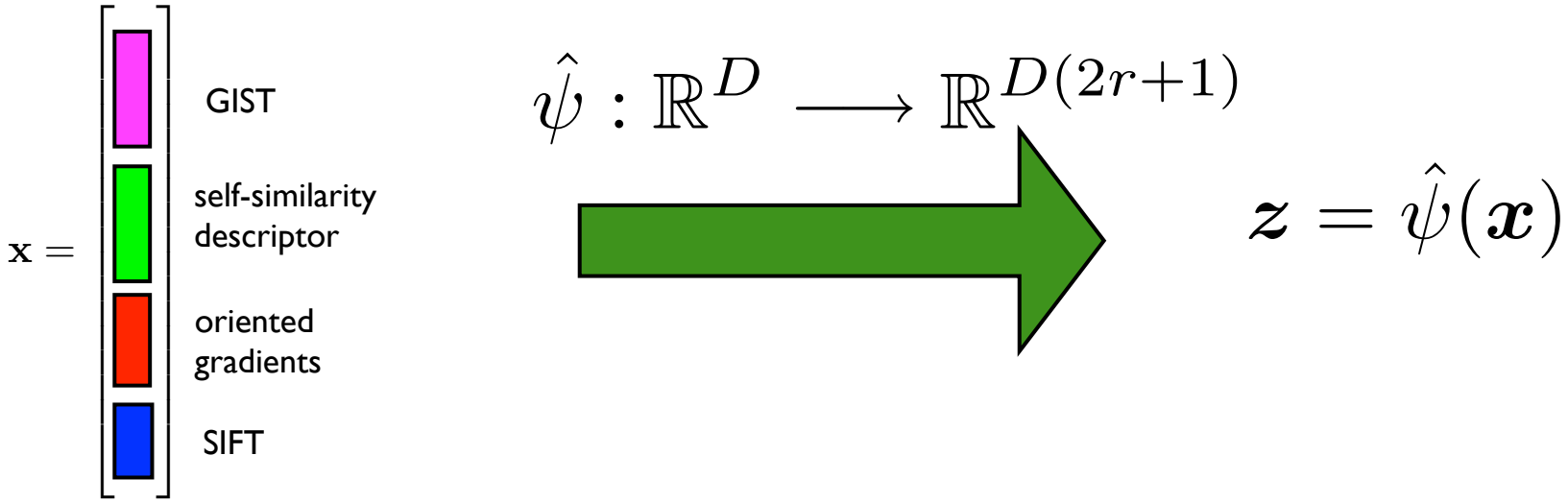
the c-th bit
for image *i*

Optimization via alternation:

- ***learn linear classifiers*** $(\mathbf{w}_k, b_k)$ (while keeping $\mathbf{a}_{1..C}$ fixed):
  traditional linear SVM learning

- ***learn PiCoDes projections*** $\mathbf{a}_{1..C}$ (while keeping $(\mathbf{w}_k, b_k)$ fixed):
  we optimize a convex upper bound of the objective that can be formulated as a linear program

Implementation details



- We use spatial pyramid histograms of 4 low-level features, yielding a total of 13 histograms
- We choose the mapping $\hat{\psi}(\mathbf{x})$ that approximates the histogram intersection kernel
- D=17K but we reduce the dimensionality of the learning space to d=6K via PCA

Prior work on compact image codes

[Andoni and Indyk, 2006]; [Salakhutdinov and Hinton, 2009]; [Torralba et al., 2008]; [Ranzato et al., 2007]; [Weiss et al., 2008]; [Jegou et al., 2010]; [Perronnin and Sanchez, 2011], [Gong and Lazebnik, 2011] ...

- **Given:**
  image descriptor $x \in \mathcal{R}^D$ (e.g., GIST)

- **Learn:**
  compact code $y \in \{0,1\}^{D'}$

  such that y_i is "near" $y_j \Leftrightarrow x_i$ is "near" x_j

Experimental setup

Offline training set
(PiCoDes learning):

- ImageNet (a subset)
  - 2625 classes
  - 30 examples / class

IMAGENET

Evaluation database
(PiCoDes testing):

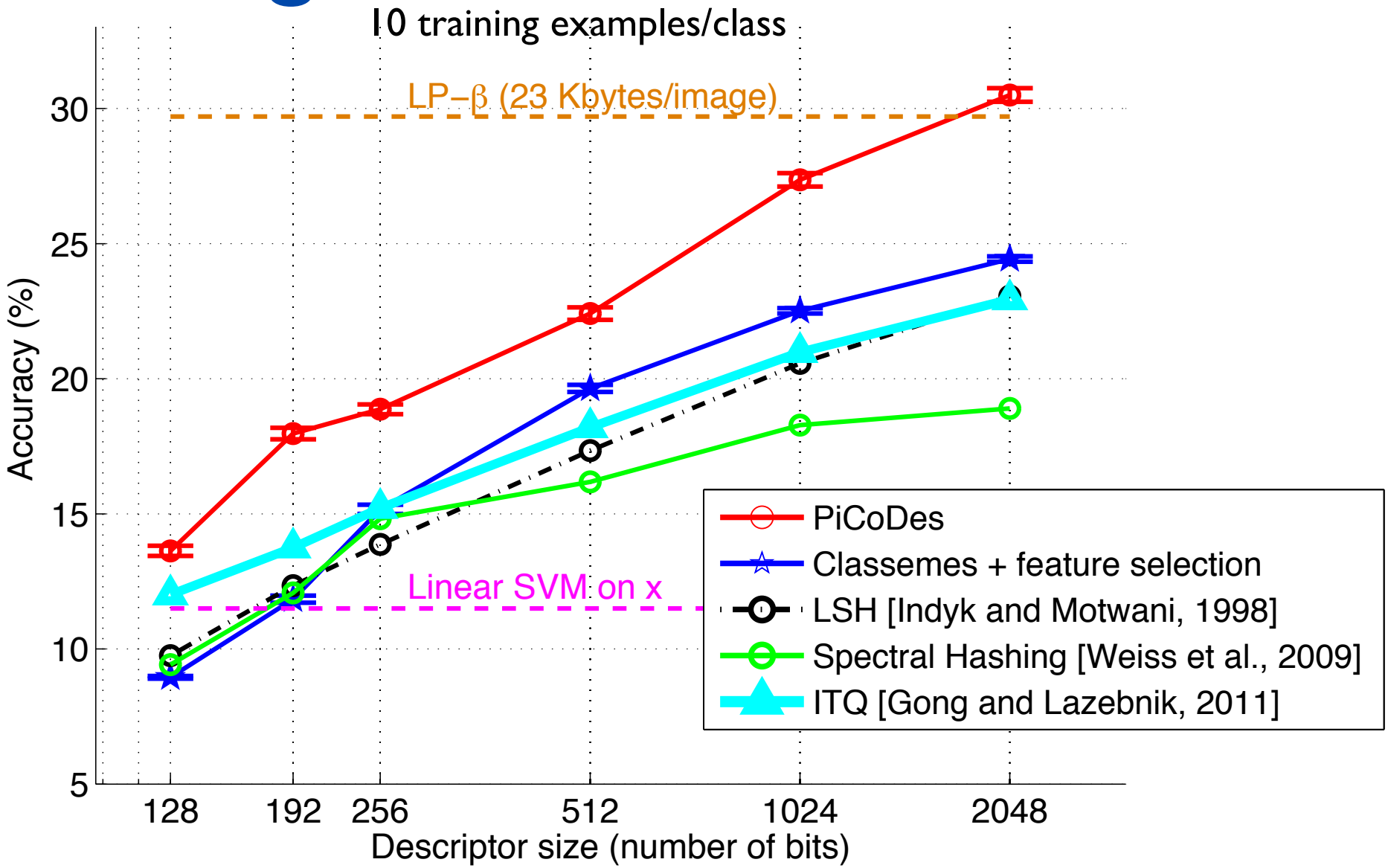
- Caltech 256
  - 256 classes
  - 10 training ex / class
  - 25 test examples / class
- ImageNet ILSVRC 2010
  - 1000 classes
  - varying # tr ex / class
  - 150 test examples / class

no classes in common

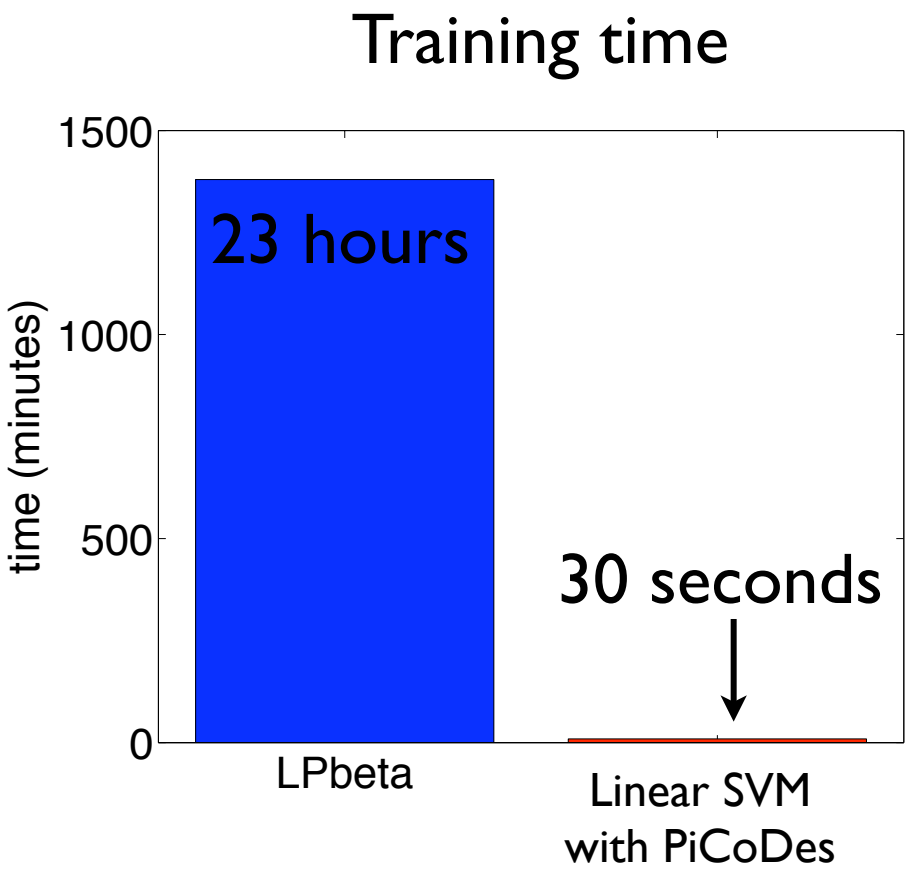
Experiments:

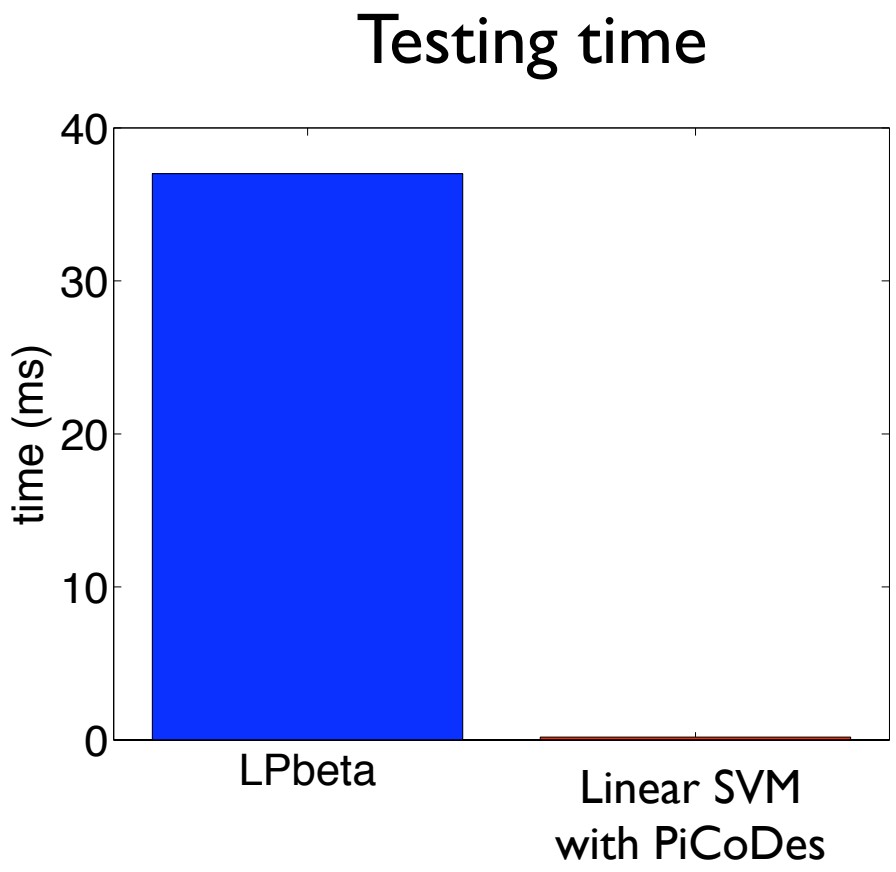
- Multiclass recognition and object-class search
  - we use a linear SVM as classification model

Experiment 1: multiclass recognition on Caltech256

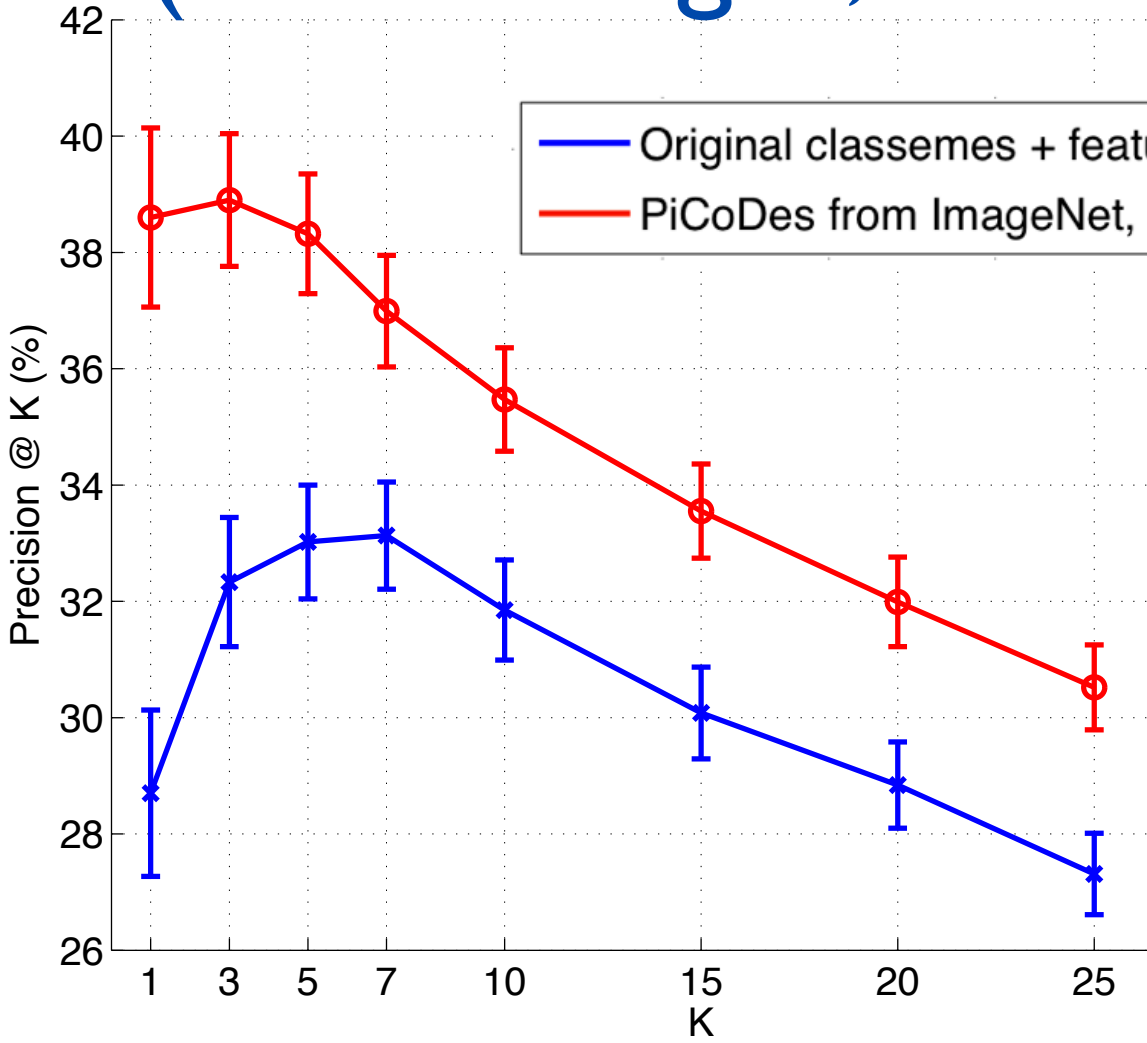


Computational cost comparison





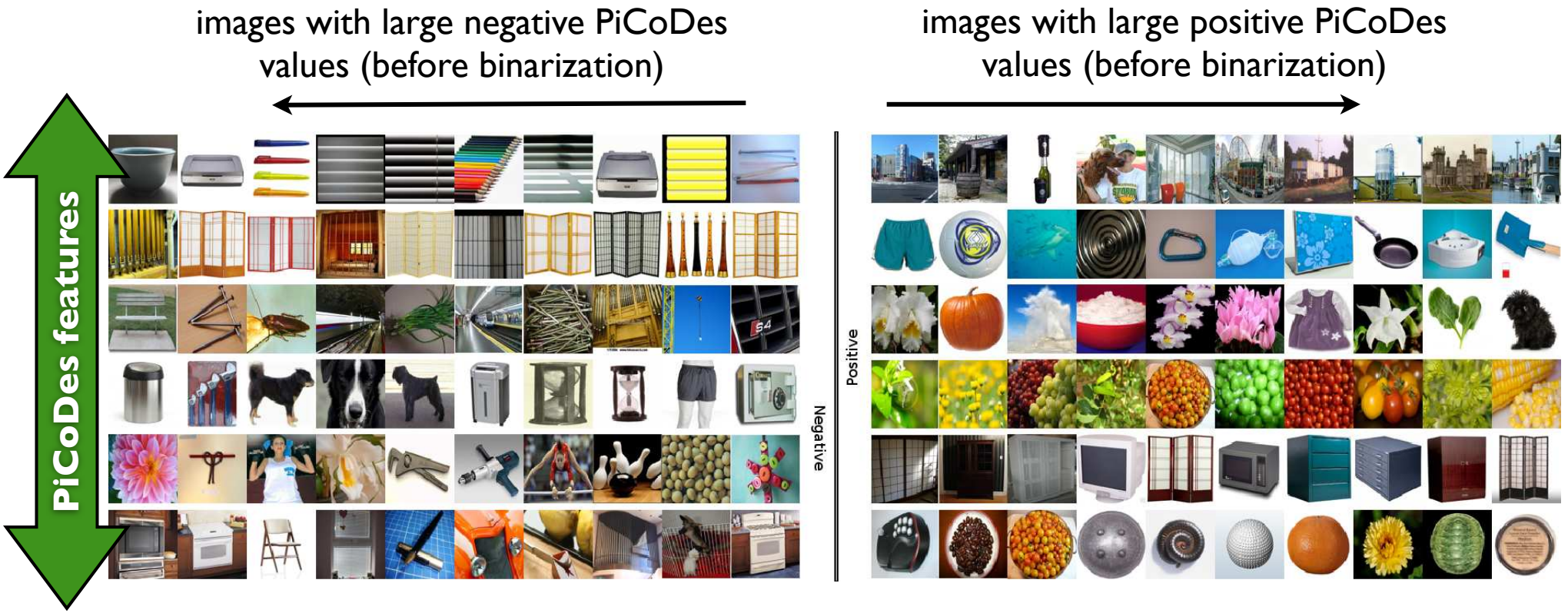
Experiment 2: novel-class search in ImageNet ILSVRC2010 (150K images, 1000 classes)



- for each query class the database contains 150 true positives, ~150K distractors
- random performance is about 0.1%
- training + search takes less than 1 second

Visualization of PiCoDes

- What kind of information is encoded in PiCoDes?



Conclusions

- PiCoDes:
  - binary features explicitly optimized for linear classification
  - can be trained for any desired descriptor size

- Even when reduced to about 200 bytes/image, recognition accuracy is similar to the best known MKL at a tiny fraction of the cost

- Future work:
  - features optimized for sparse/conjunctive classifiers
  - descriptors for subwindow recognition ([Li et al. NIPS10])

- *Software for (fast!) extraction of PiCoDes is available at:* http://vlg.cs.dartmouth.edu/