Introduction to Recognition

Computer Vision

CS 543 / ECE 549

University of Illinois

Many Slides from D. Hoiem, L. Lazebnik.

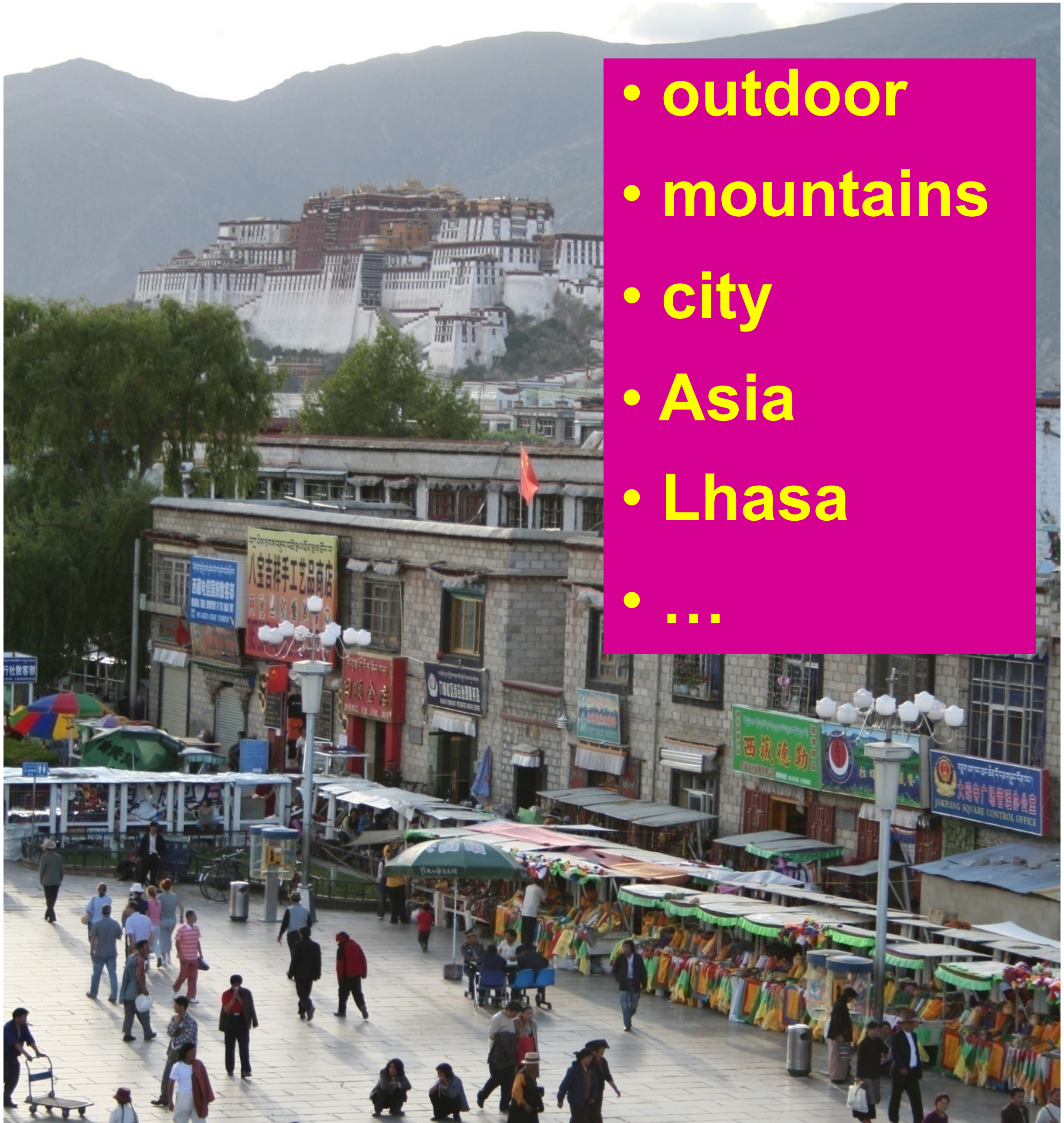
Outline

- Overview of image and region categorization
  - Task description
  - What is a category
- Example of spatial pyramids bag-of-words scene categorizer
- Key concepts: features and classification
- Deep convolutional neural networks (CNNs)

Common recognition tasks

Adapted from
Fei-Fei Li

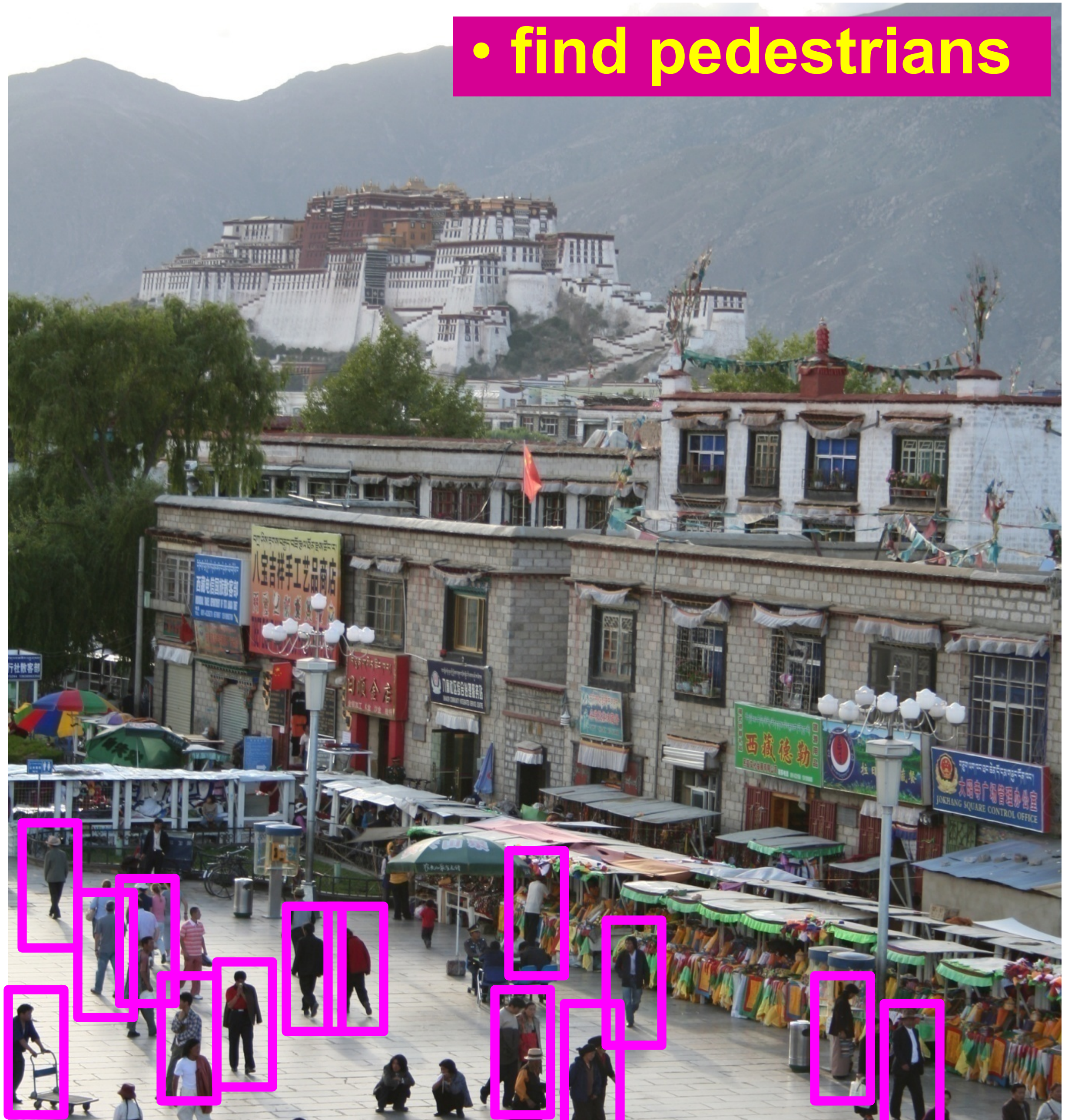
Image classification and tagging



- **outdoor**
- **mountains**
- **city**
- **Asia**
- **Lhasa**
- **…**

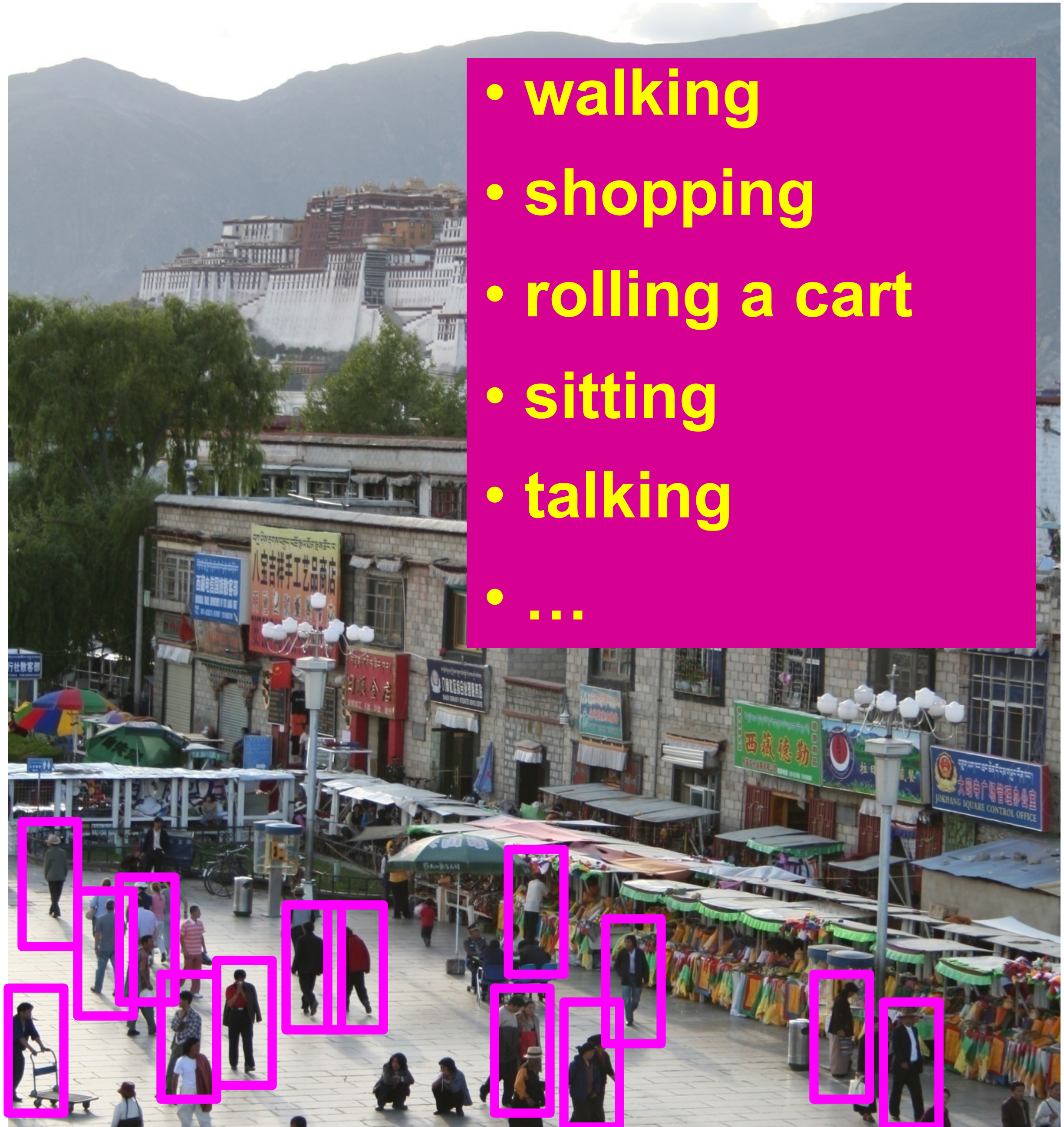
Adapted from
Fei-Fei Li

Object detection



Adapted from
Fei-Fei Li

Activity recognition



- **walking**
- **shopping**
- **rolling a cart**
- **sitting**
- **talking**
- **…**

Adapted from
Fei-Fei Li

Semantic segmentation

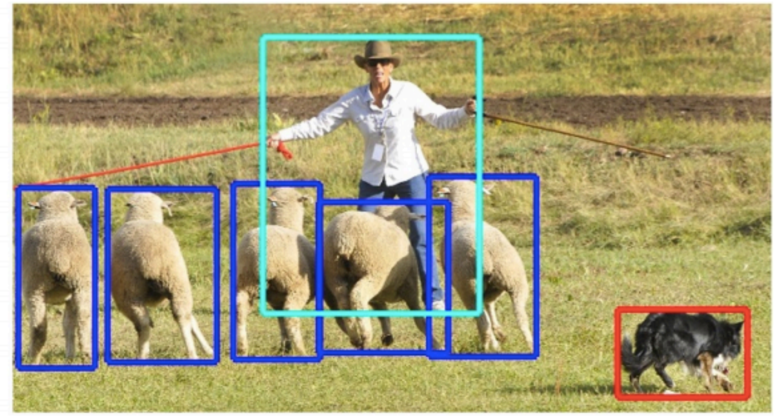
Adapted from
Fei-Fei Li

Semantic segmentation

Adapted from
Fei-Fei Li

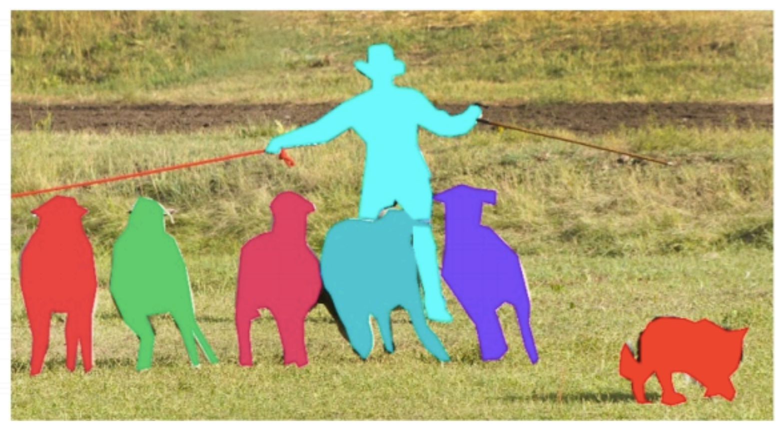
Detection, semantic segmentation, instance segmentation

image classification



object detection

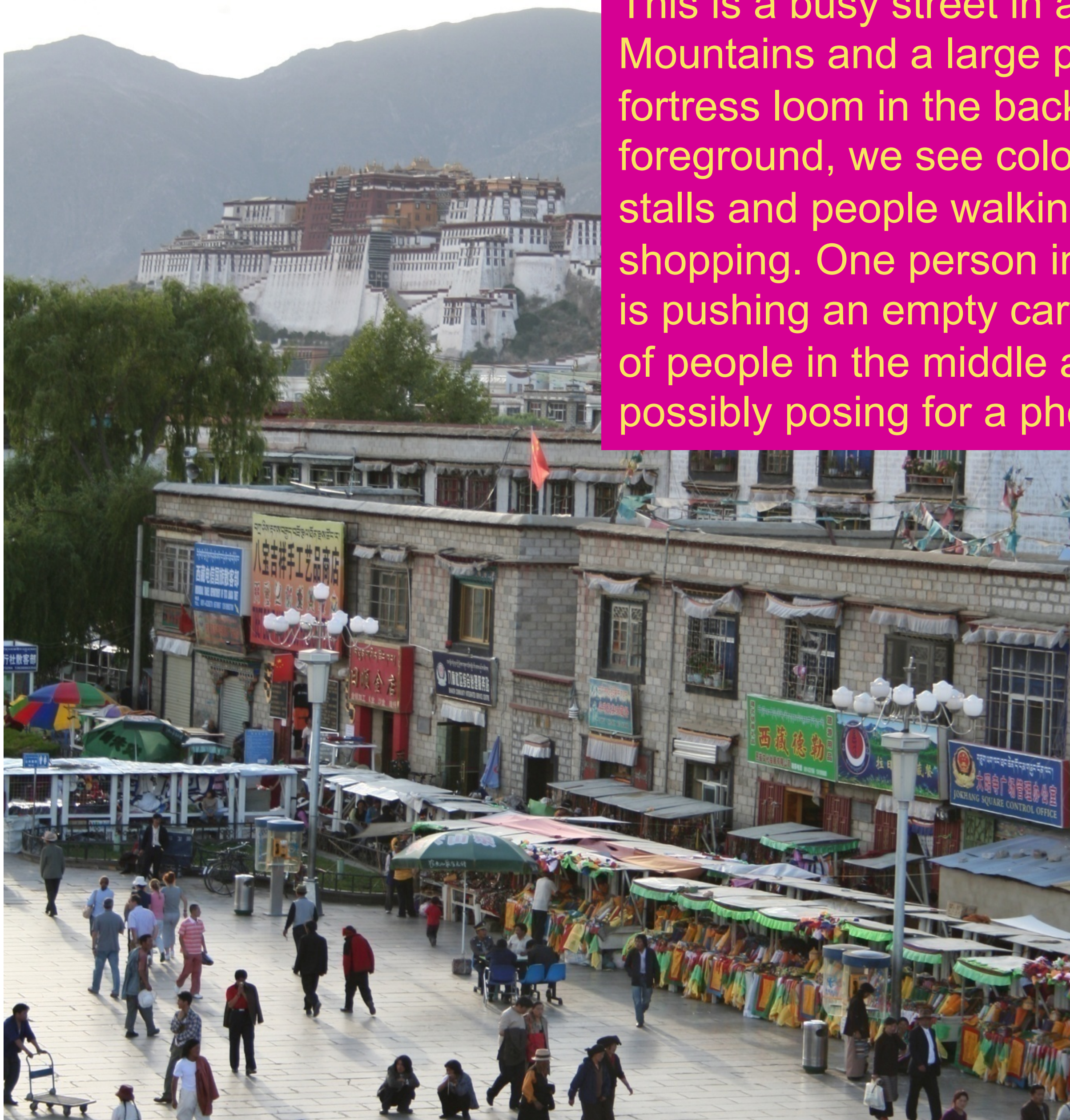
semantic segmentation



instance segmentation

Image source

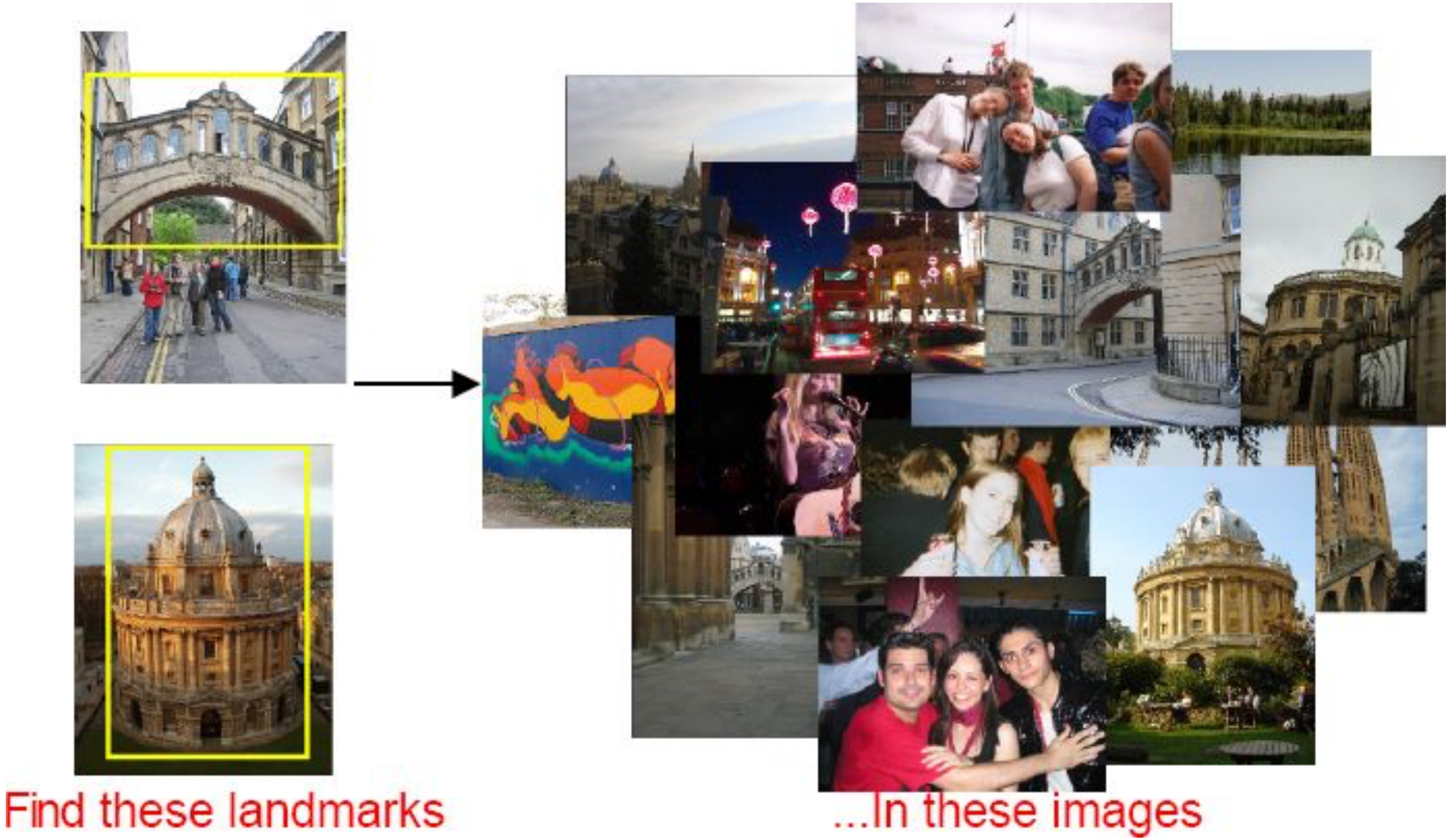
Image description



This is a busy street in an Asian city. Mountains and a large palace or fortress loom in the background. In the foreground, we see colorful souvenir stalls and people walking around and shopping. One person in the lower left is pushing an empty cart, and a couple of people in the middle are sitting, possibly posing for a photograph.

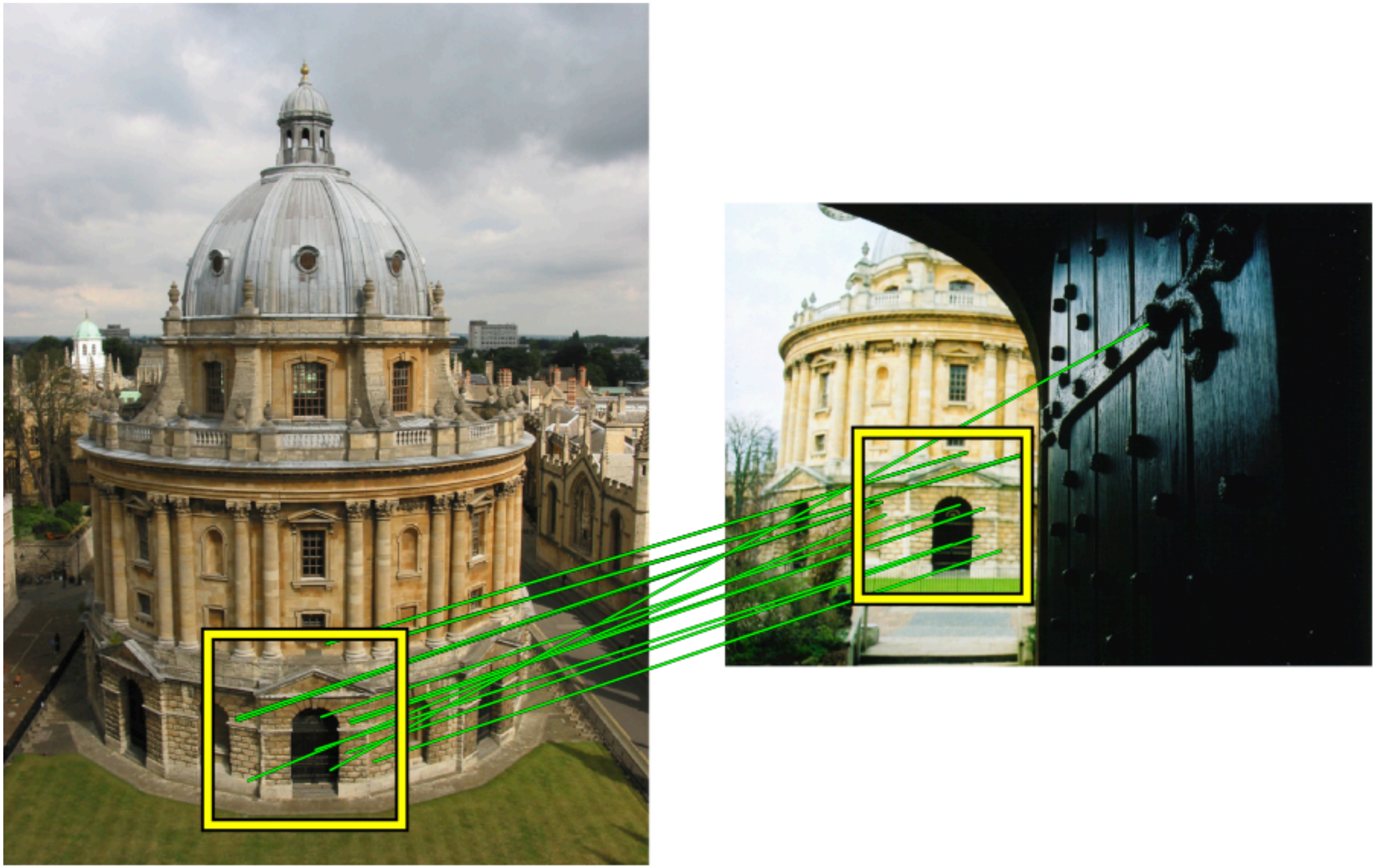
Adapted from Fei-Fei Li

Recognition as 3D Matching



http://www.robots.ox.ac.uk/~vgg/research/oxbuildings/index.html

Recognition as 3D Matching



Recognizing solid objects by alignment with an image. Huttenlocher and Ullman IJCV 1990.

Recognition as 3D Matching

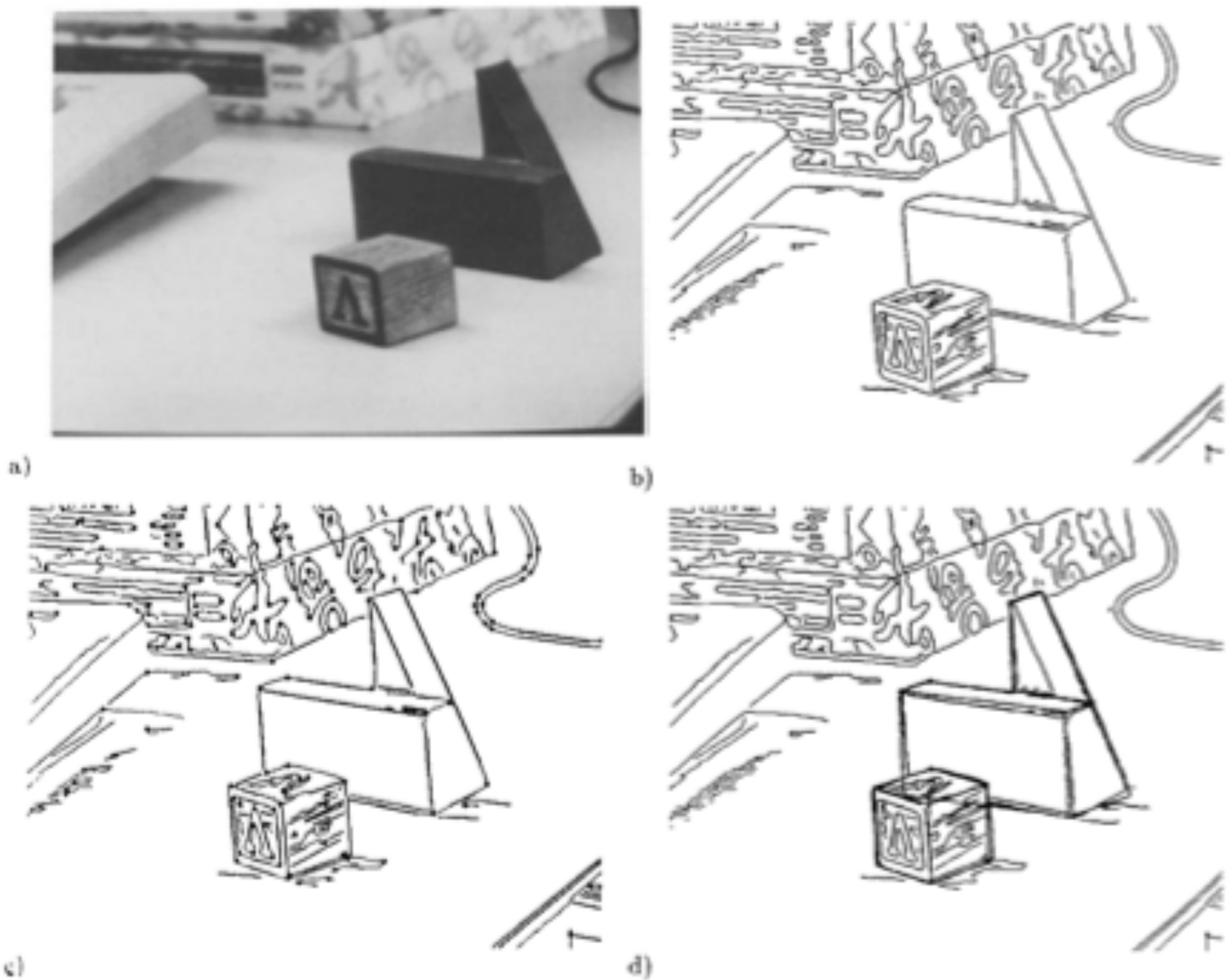


Fig. 8. The output of the recognizer: (a) grey-level image input, (b) Canny edges, (c) edge segments, (d) recovered instances.

"Instance" Recognition

"Category-level" Recognition

Recognizing solid objects by alignment with an image. Huttenlocher and Ullman IJCV 1990.

Theory of categorization

How do we determine if something is a member of a particular category?

- Definitional approach

- Prototype approach

- Exemplar approach

Definitional approach: classical view of categories

- Plato & Aristotle
  - Categories are defined by a list of properties shared by all elements in a category
  - Category membership is binary
  - Every member in the category is equal

Aristotle by Francesco Hayez

The Categories (Aristotle)

Slide Credit: A. A. Efros

Prototype Model

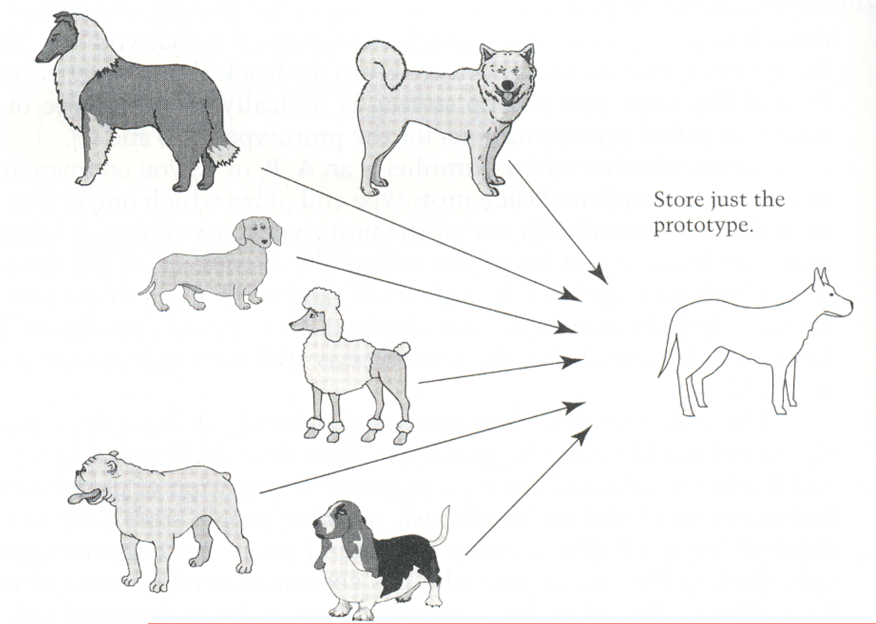


Figure 7.3. *Schematic of the prototype model. Although many exemplars are seen, only the prototype is stored. The prototype is updated continually to incorporate more experience with new exemplars.*

Category judgments are made by comparing a new exemplar to the prototype.

Exemplars Model

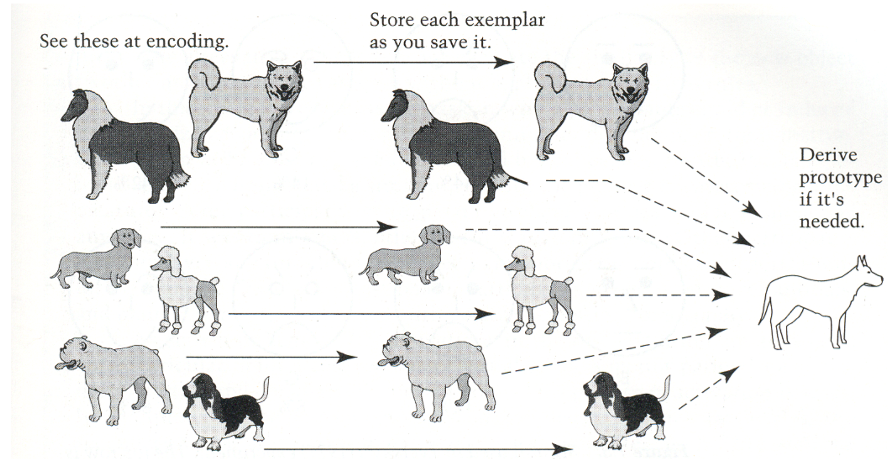


Figure 7.4. *Schematic of the exemplar model. As each exemplar is seen, it is encoded into memory. A prototype is abstracted only when it is needed, for example, when a new exemplar must be categorized.*

Category judgments are made by comparing a new exemplar to all the old exemplars of a category or to the exemplar that is the most appropriate

Slide Credit: Torralba

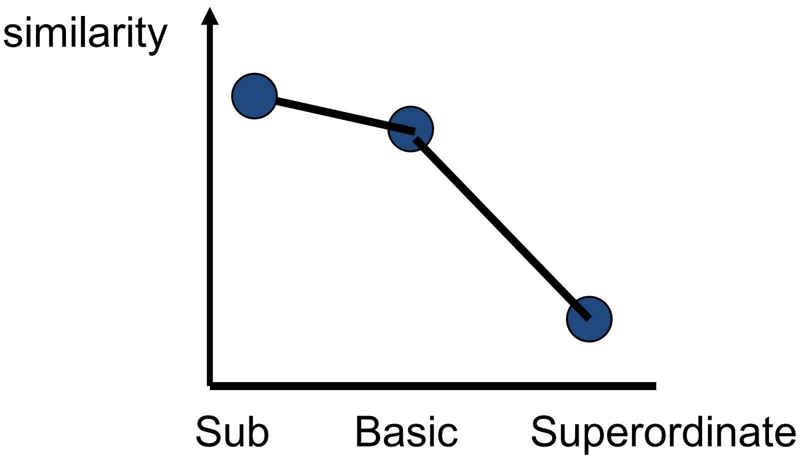
Levels of categorization [Rosch 70s]

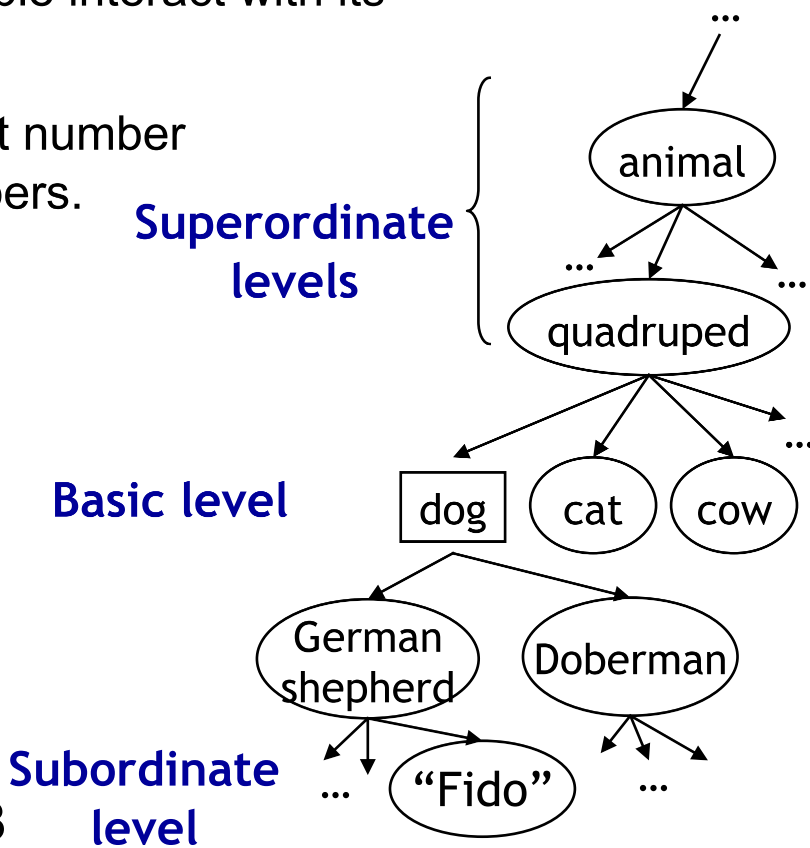
Definition of Basic Level:

• **Similar shape**: Basic level categories are the highest-level category for which their members have similar shapes.

• **Similar motor interactions**: … for which people interact with its members using similar motor sequences.

• **Common attributes**: … there are a significant number of attributes in common between pairs of members.





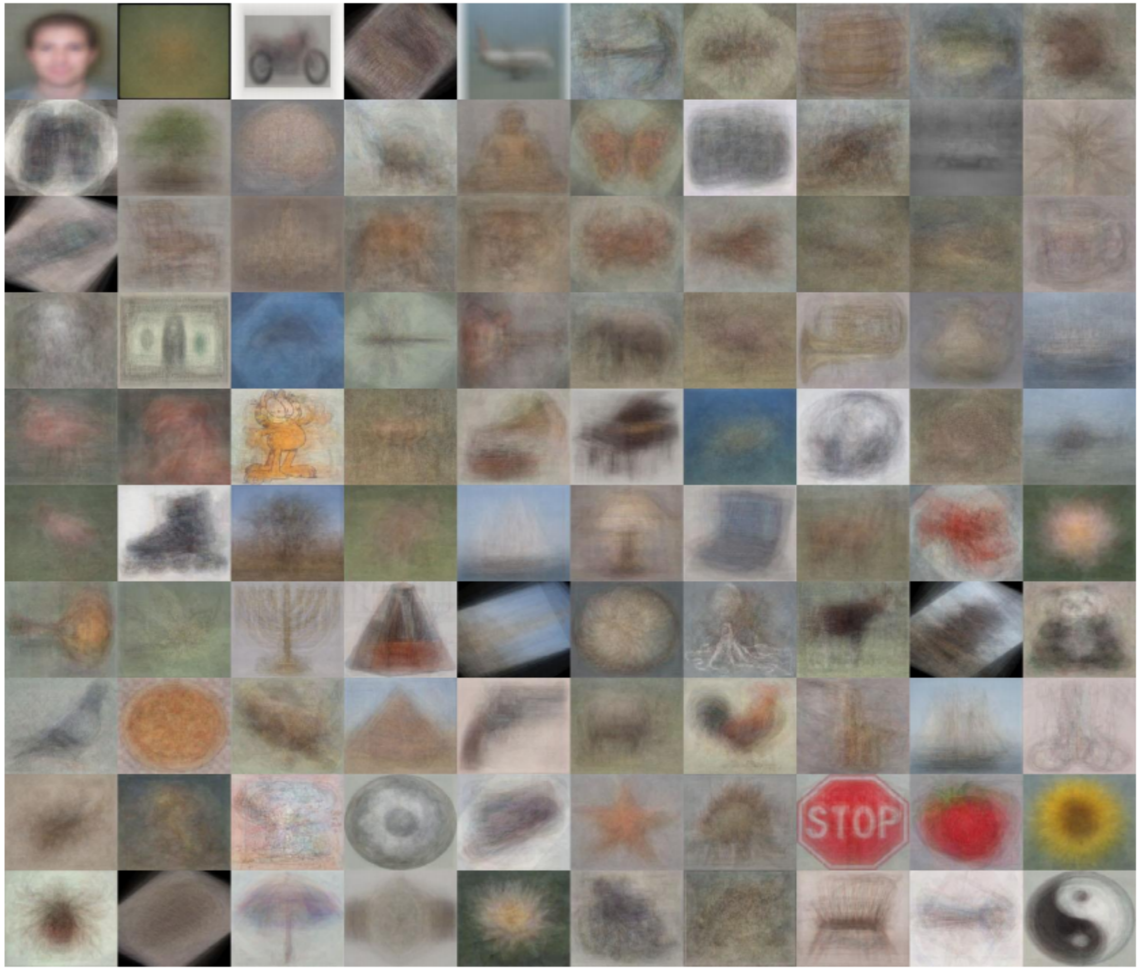
[Rosch et al. Principle of categorization](), 1978

Image categorization

- Cat vs Dog

Image categorization

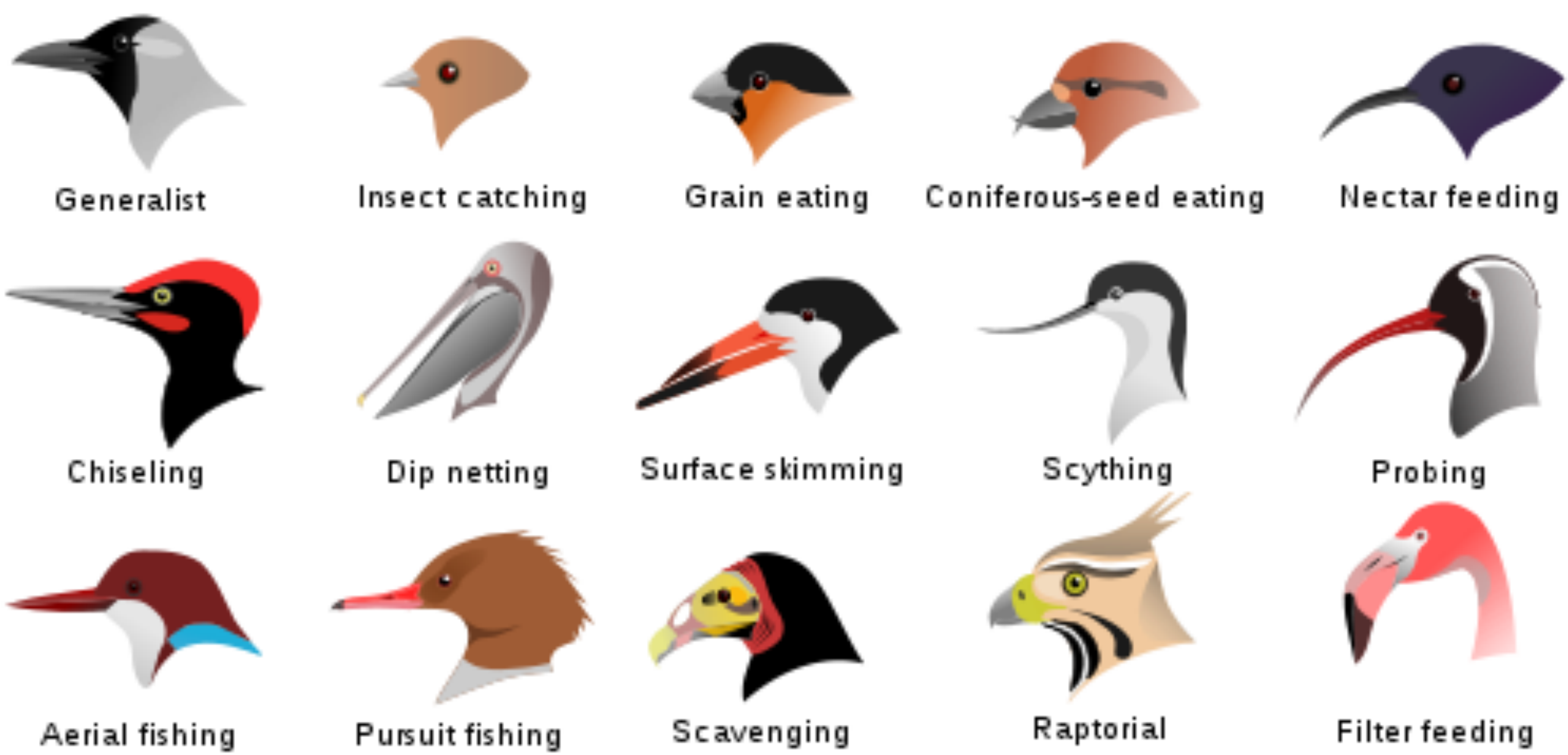
- Object recognition



Caltech 101 Average Object Images

Image categorization

- Fine-grained recognition



[Visipedia Project]

Image categorization

- Place recognition

Places Database [Zhou et al. NIPS 2014]

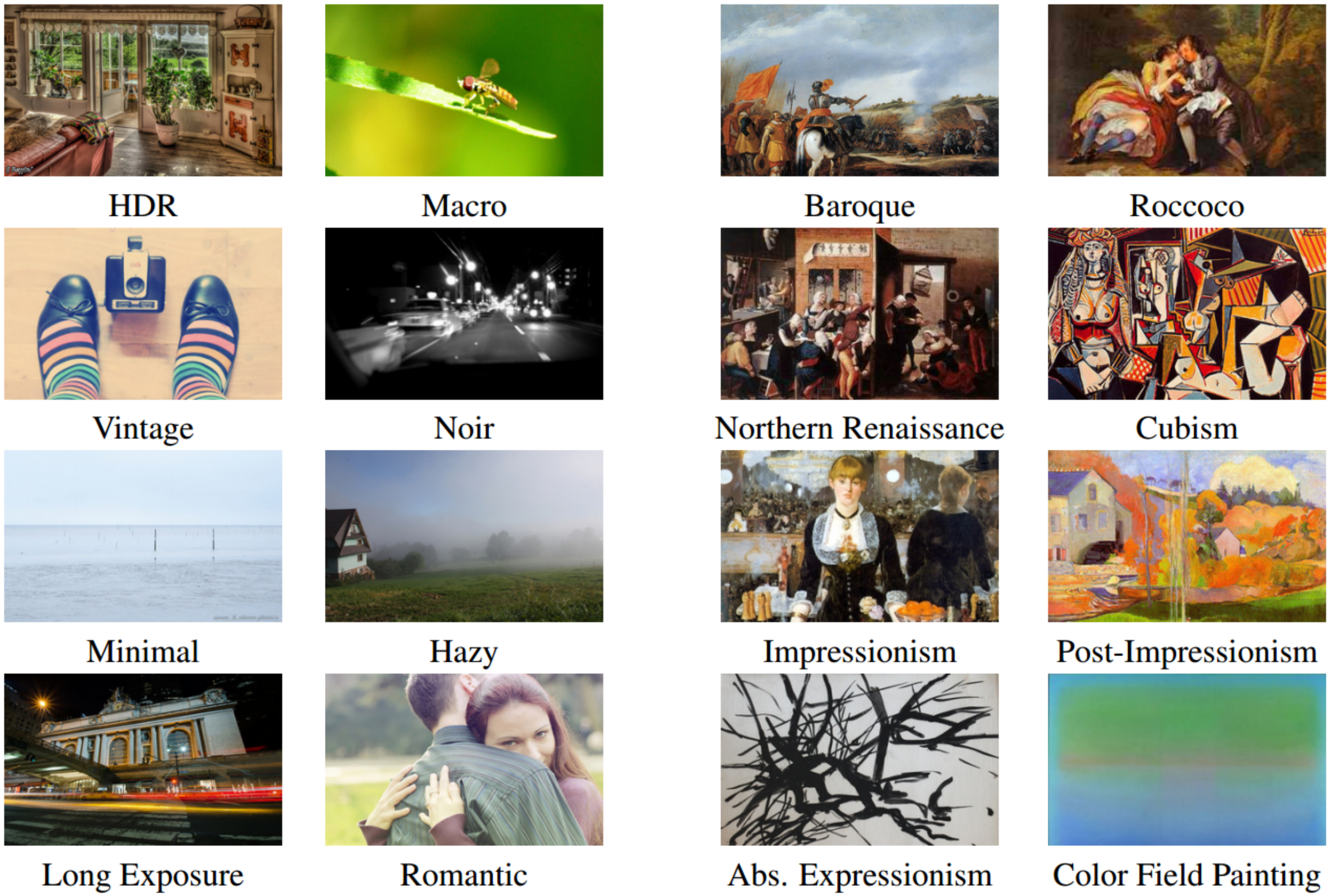
Image categorization

- Visual font recognition

[Chen et al. CVPR 2014]

Image categorization

- Image style recognition

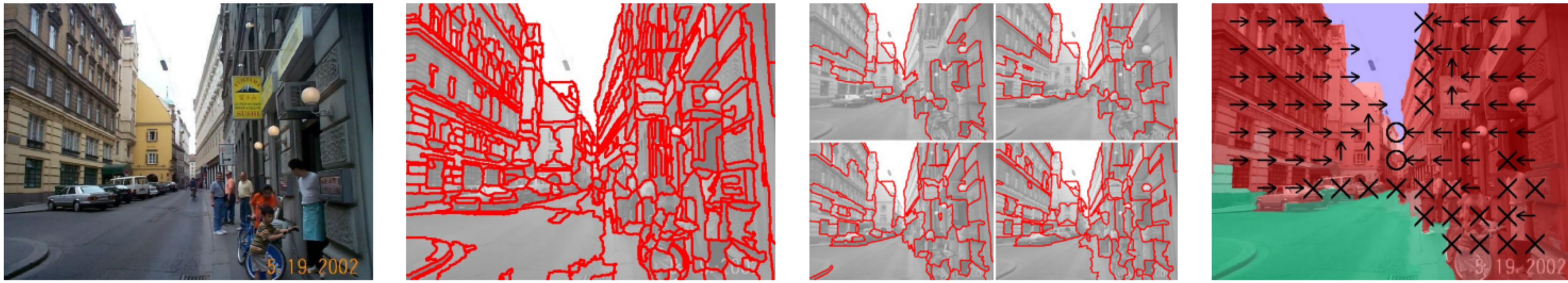


Flickr Style: 80K images covering 20 styles. Wikipaintings: 85K images for 25 art genres.

[Karayev et al. BMVC 2014]

Region categorization

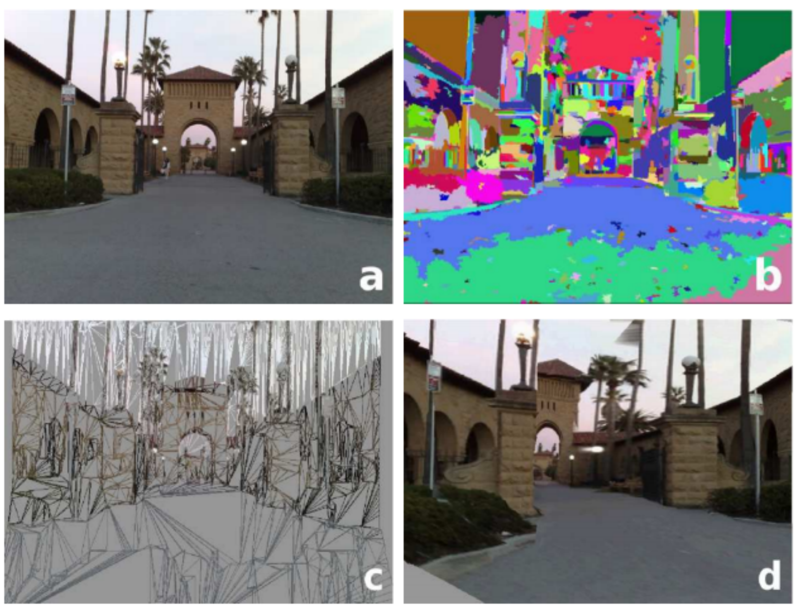
- Layout prediction



Input Superpixels Multiple Segmentations Surface Layout

Assign regions to orientation

Geometric context [Hoiem et al. IJCV 2007]

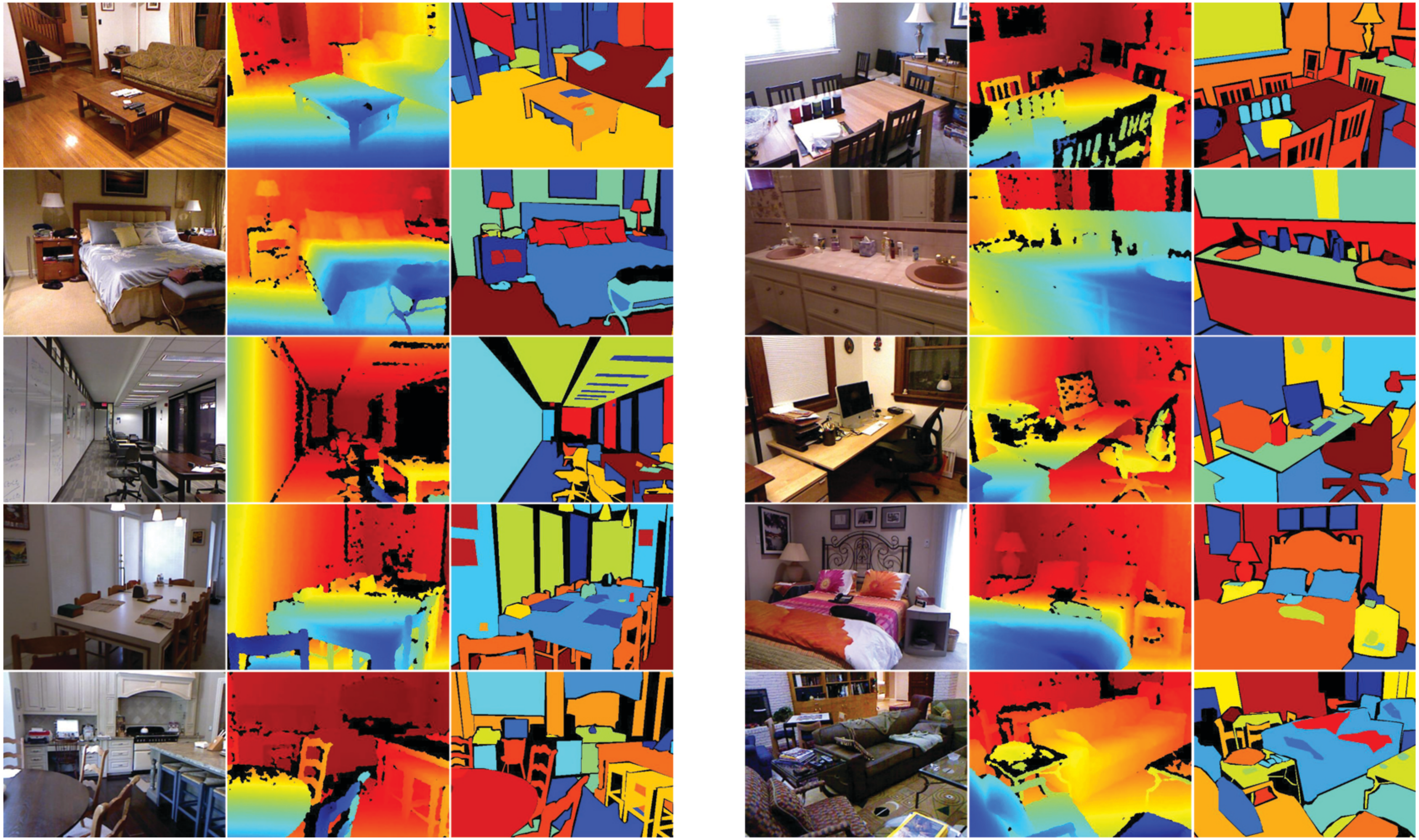


Assign regions to depth

Make3D [Saxena et al. PAMI 2008]

Region categorization

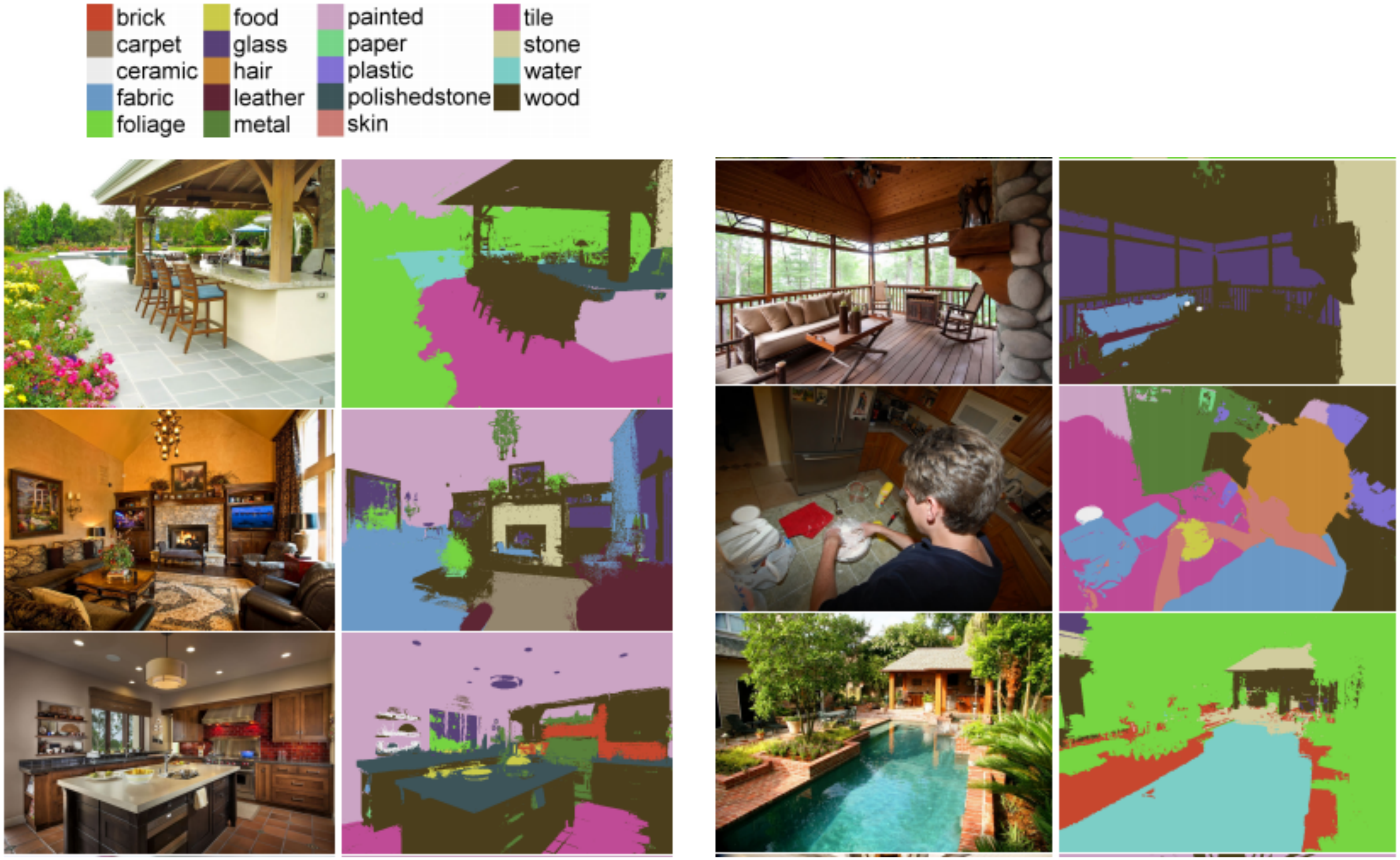
- Semantic segmentation from RGBD images



[Silberman et al. ECCV 2012]

Region categorization

- Material recognition



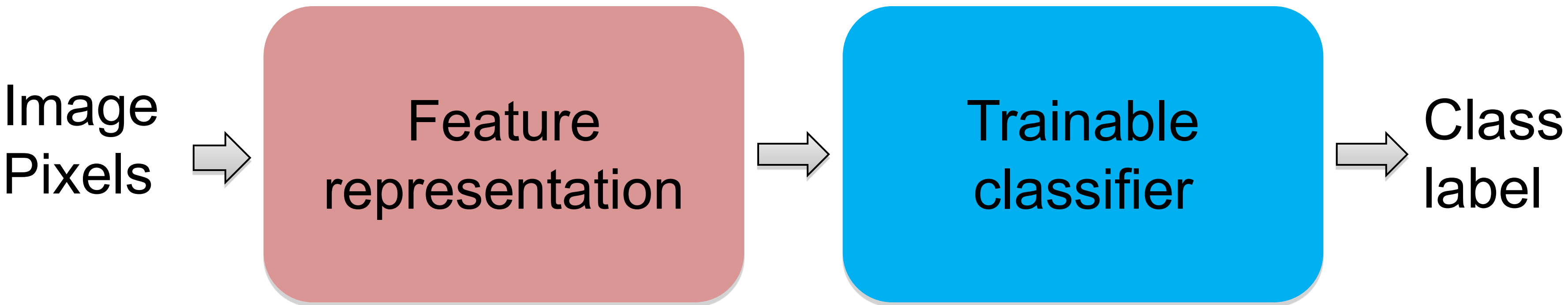
[Bell et al. CVPR 2015]

Many vision problems involve categorization

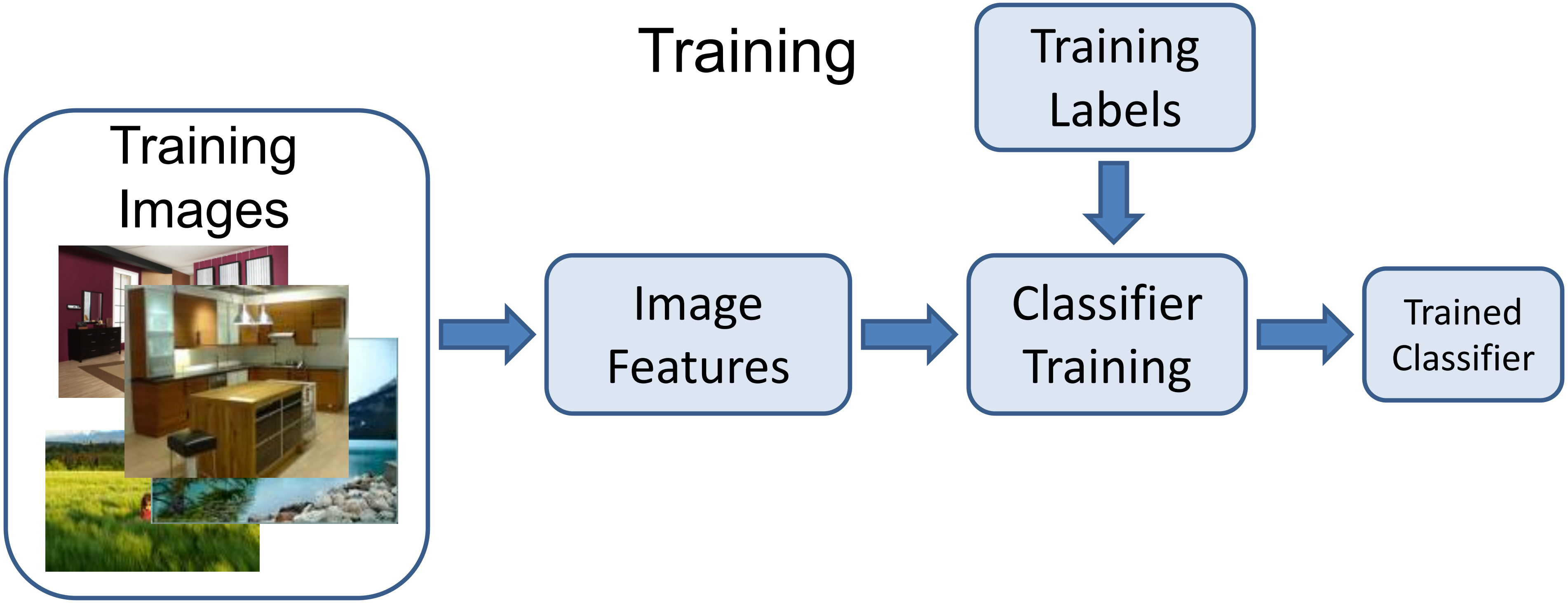
- Image: Classify as indoor/outdoor, which room, what objects are there, etc.
- Object Detection: classify location (bounding box or region) as object or non-object
- Semantic Segmentation: classify pixel into an object, material, part, etc.
- Action Recognition: classify a frame or sequence into an action type

…

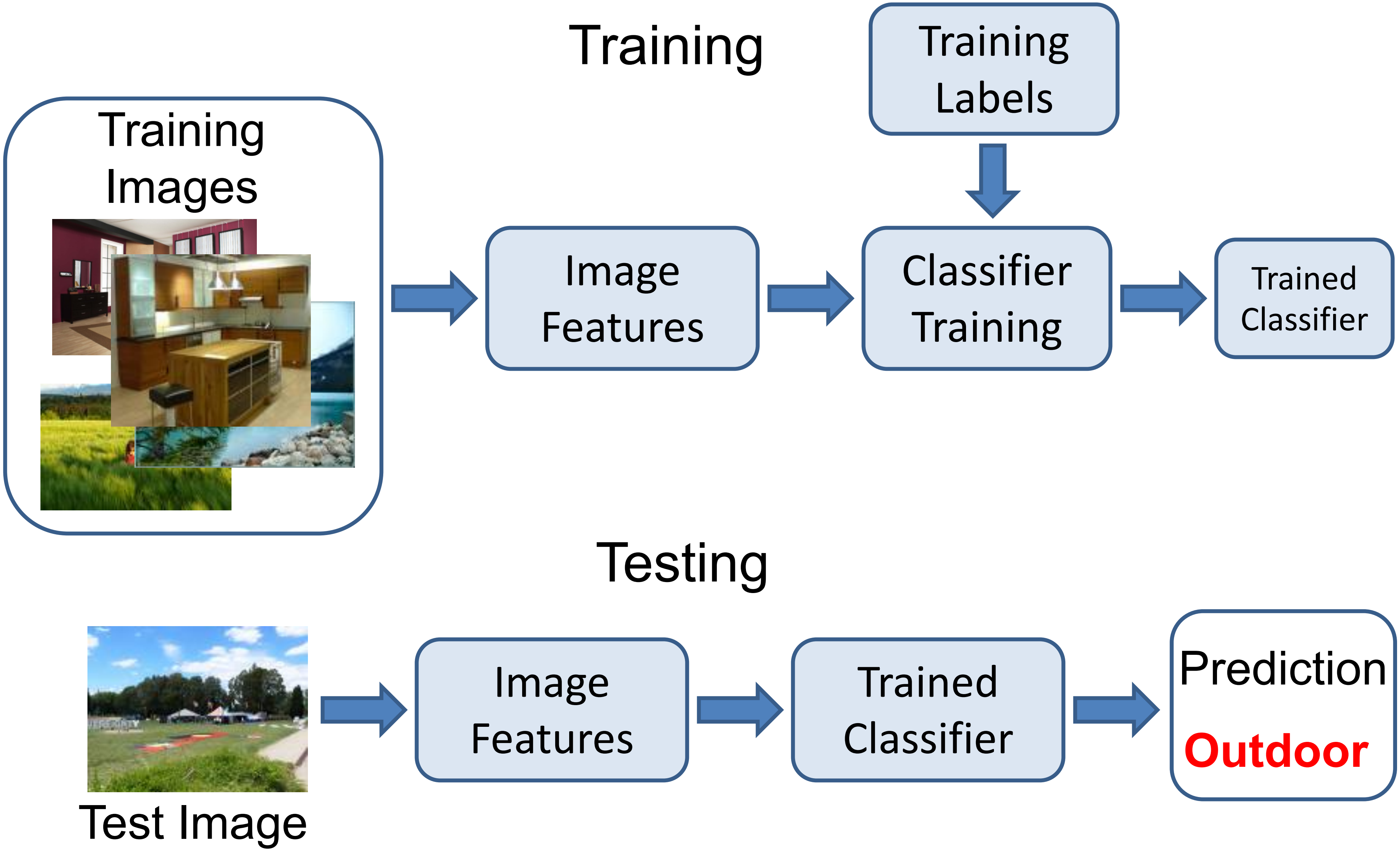
“Classic” recognition pipeline



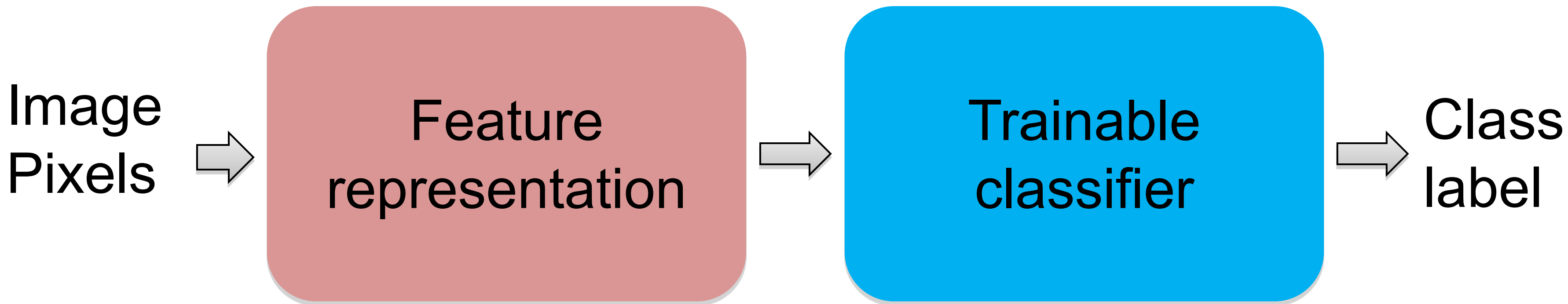
Training phase



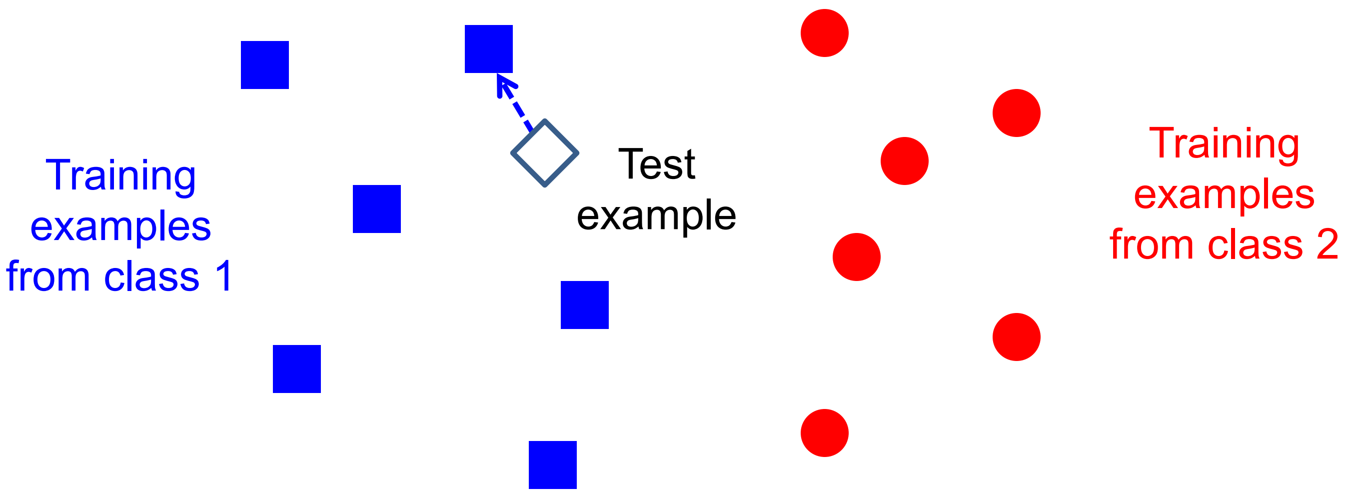
Testing phase



“Classic” recognition pipeline



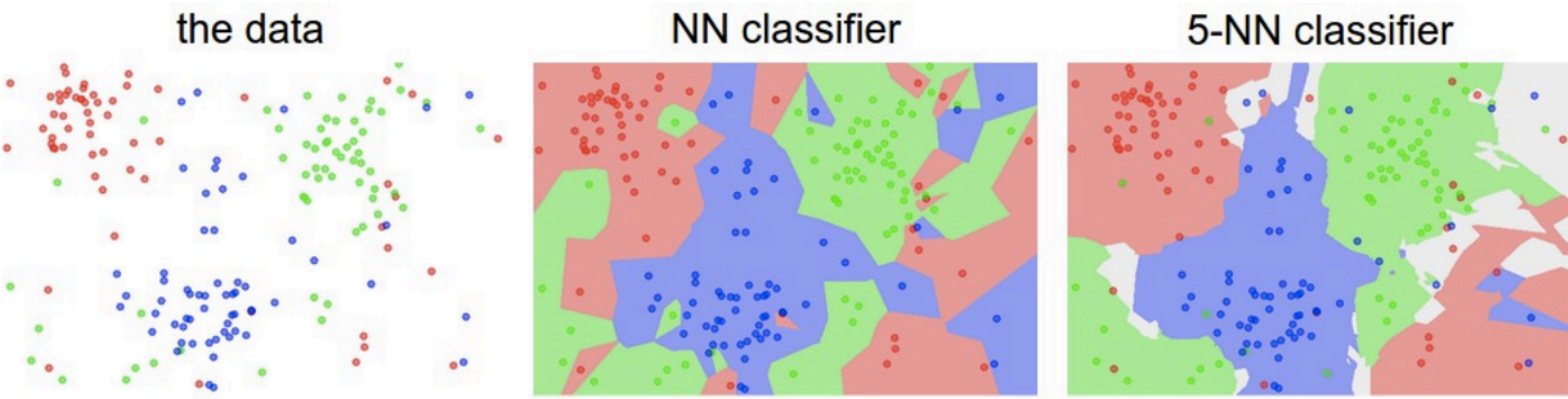
Classifiers: Nearest neighbor



f(**x**) = label of the training example nearest to **x**

- All we need is a distance or similarity function for our inputs
- No training required!

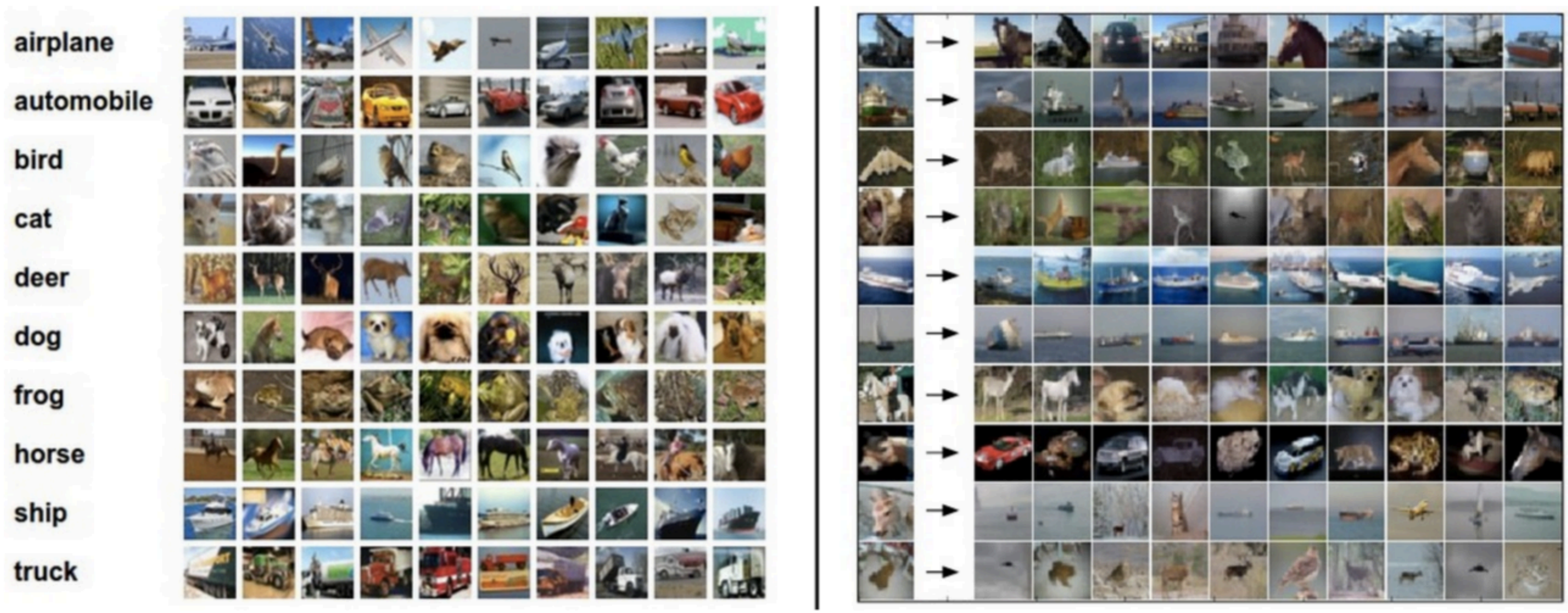
K-nearest neighbor classifier



- Which classifier is more robust to *outliers*?

Credit: Andrej Karpathy, http://cs231n.github.io/classification/

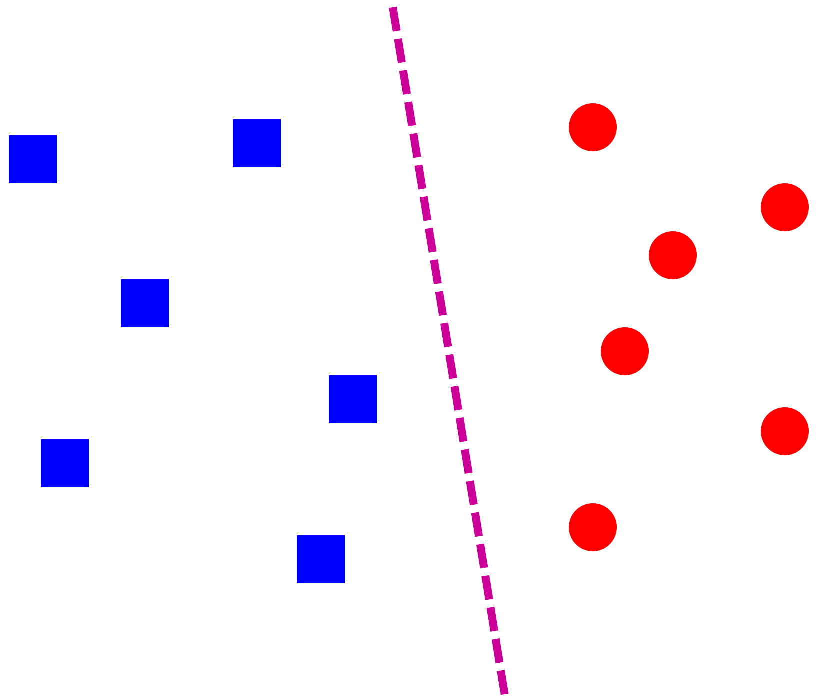
K-nearest neighbor classifier



Left: Example images from the CIFAR-10 dataset. Right: first column shows a few test images and next to each we show the top 10 nearest neighbors in the training set according to pixel-wise difference.

Credit: Andrej Karpathy, http://cs231n.github.io/classification/

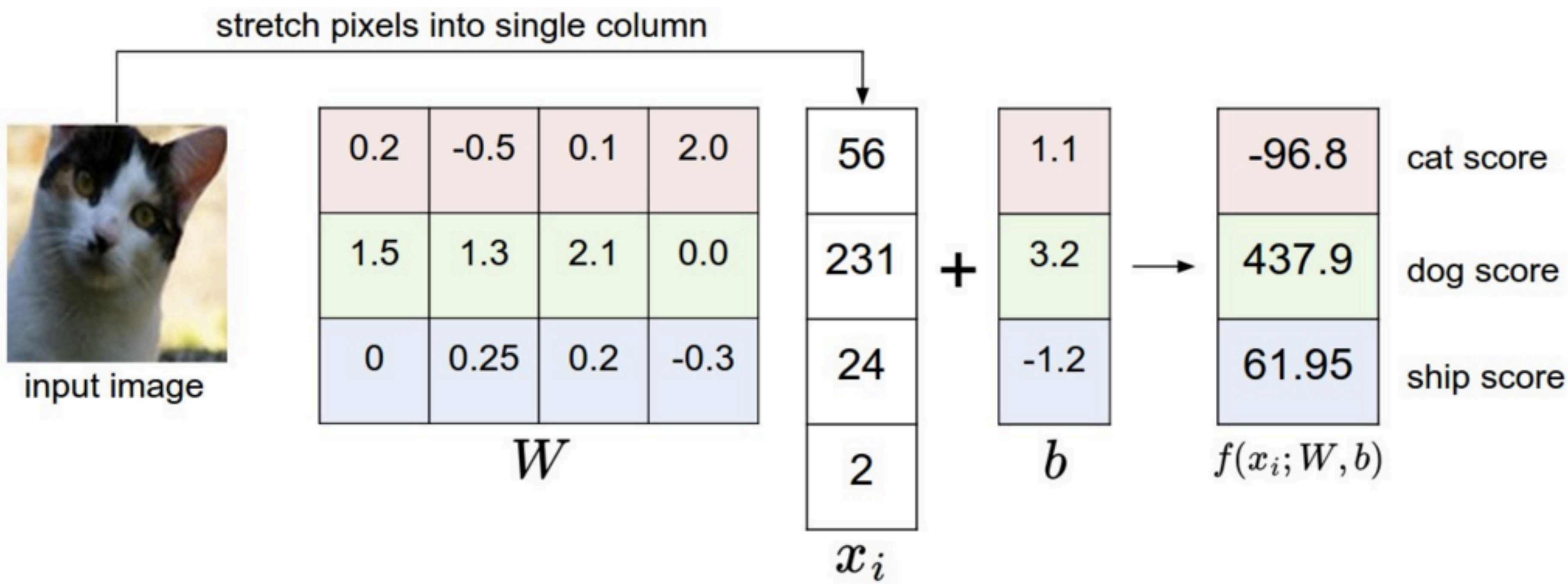
Linear classifiers



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Visualizing linear classifiers



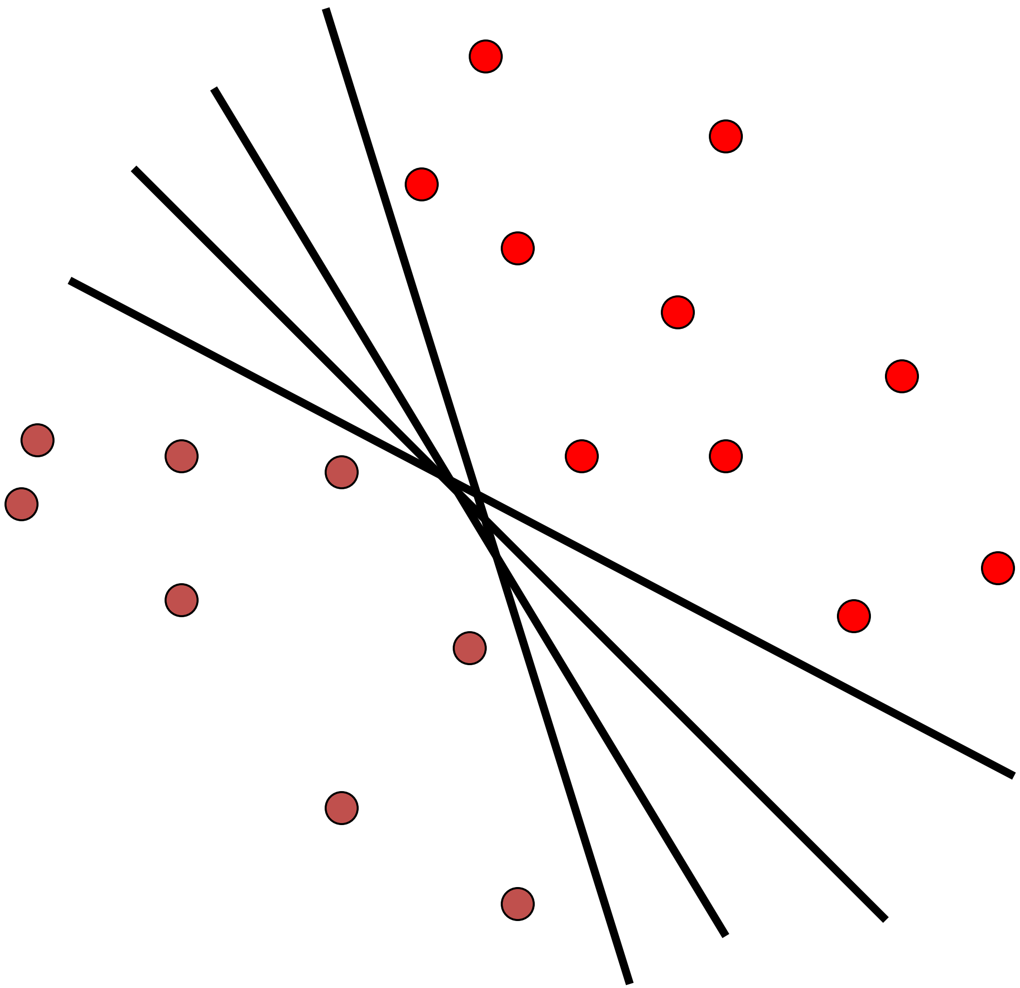
Source: Andrej Karpathy, http://cs231n.github.io/linear-classify/

Nearest neighbor vs. linear classifiers

- NN pros:
  - Simple to implement
  - Decision boundaries not necessarily linear
  - Works for any number of classes
  - *Nonparametric* method
- NN cons:
  - Need good distance function
  - Slow at test time
- Linear pros:
  - Low-dimensional *parametric* representation
  - Very fast at test time
- Linear cons:
  - Works for two classes
  - How to train the linear function?
  - What if data is not linearly separable?

Linear classifiers

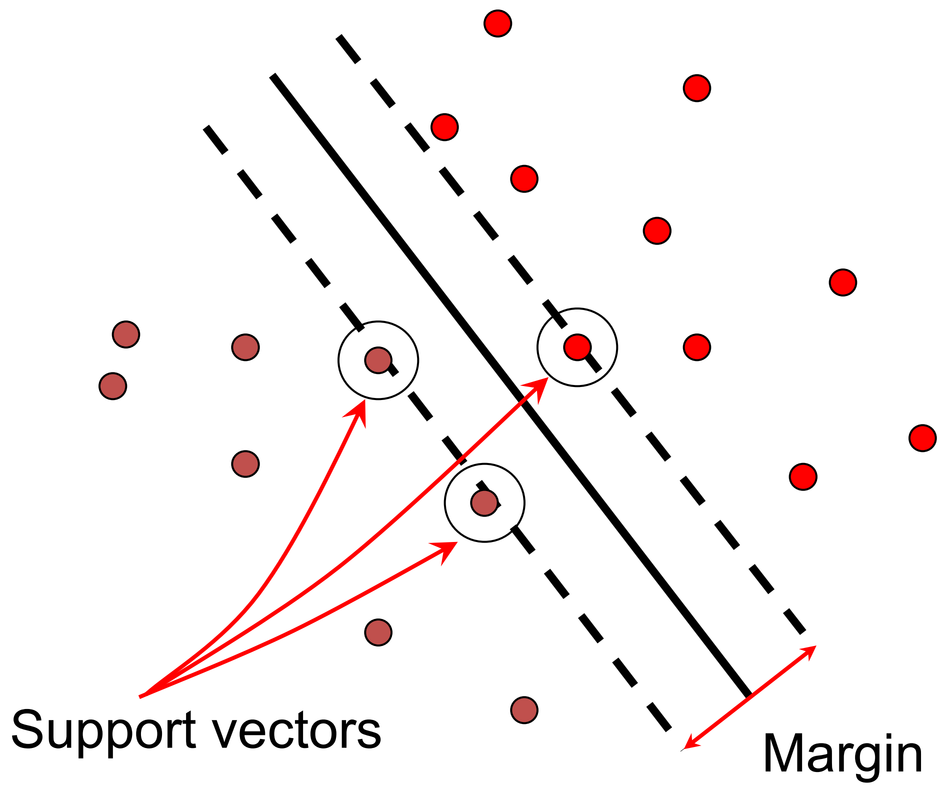
- When the data is linearly separable, there may be more than one separator (hyperplane)



Which separator is best?

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

1. Maximize margin $2 / \|\mathbf{w}\|$
2. Correctly classify all training data:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- *Quadratic optimization problem*:
- $$\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

SVM parameter learning

- Separable data:

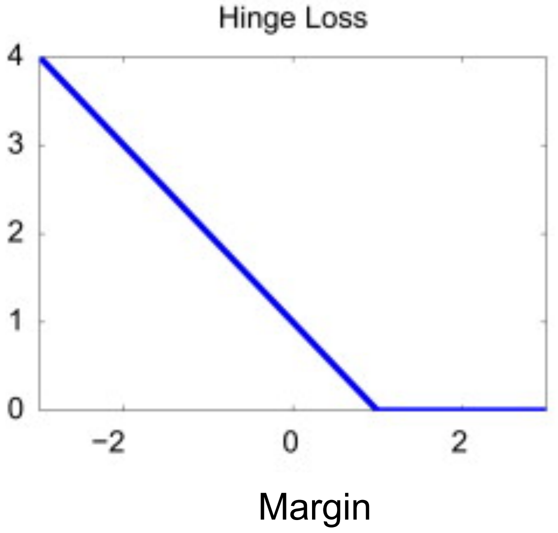
$$\underbrace{\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2}_{\boxed{\text{Maximize margin}}} \quad \underbrace{\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1}_{\boxed{\text{Classify training data correctly}}}$$

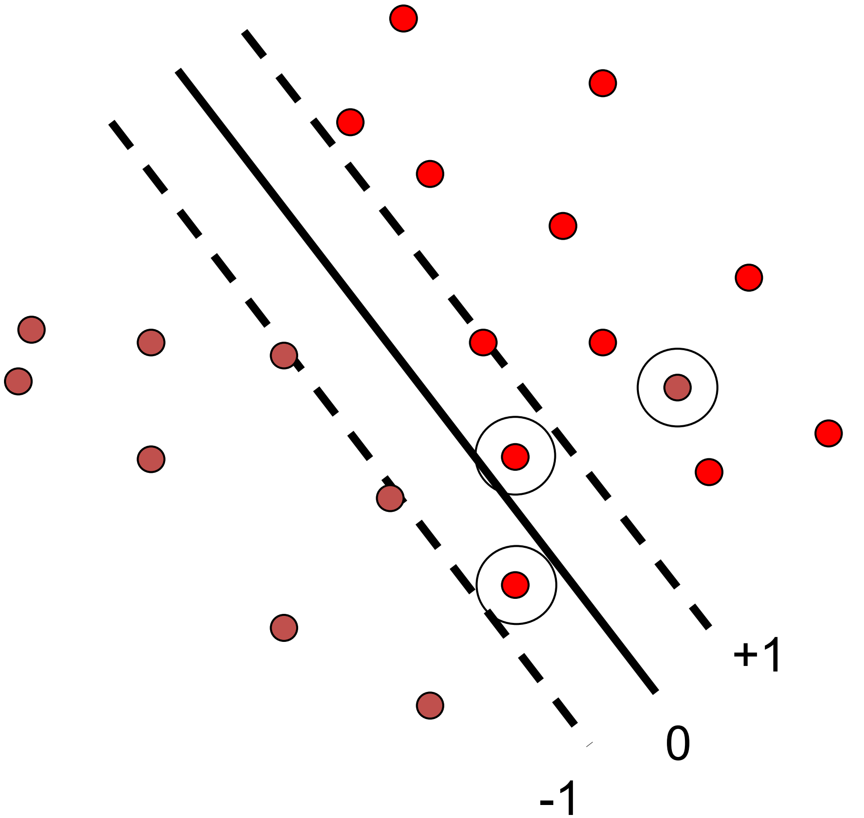
- Non-separable data:

$$\min_{\mathbf{w},b} \quad \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\boxed{\text{Maximize margin}}} + C \underbrace{\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\right)}_{\boxed{\text{Minimize classification mistakes}}}$$

SVM parameter learning

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\right)$$





- Demo: http://cs.stanford.edu/people/karpathy/svmjs/demo

Nonlinear SVMs

- **General idea:** the original input space can always be mapped to some higher-dimensional feature space where the training

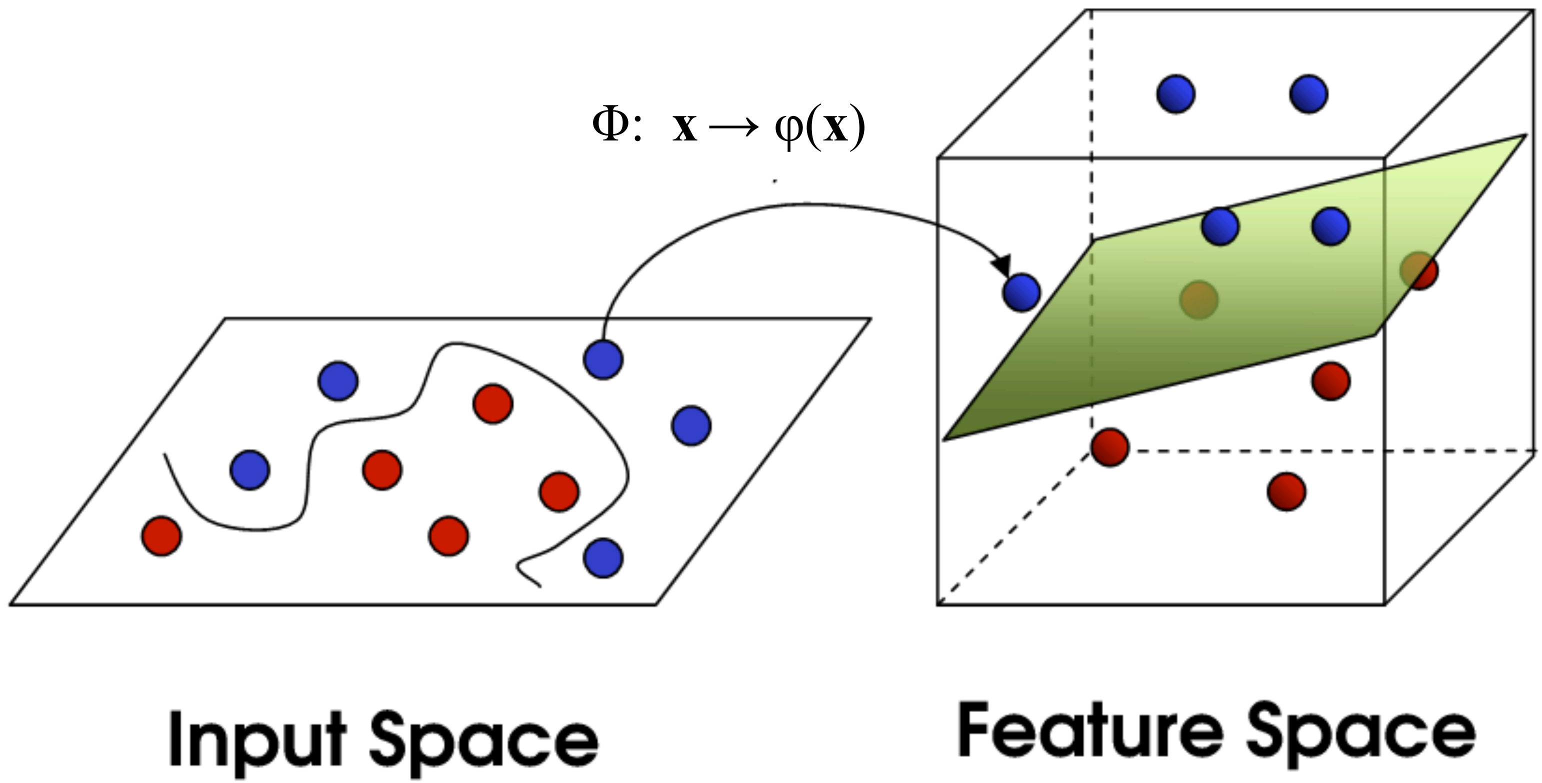
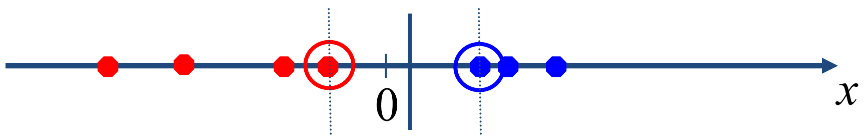


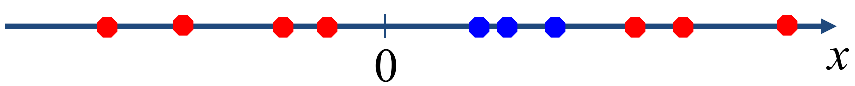
Image source

Nonlinear SVMs

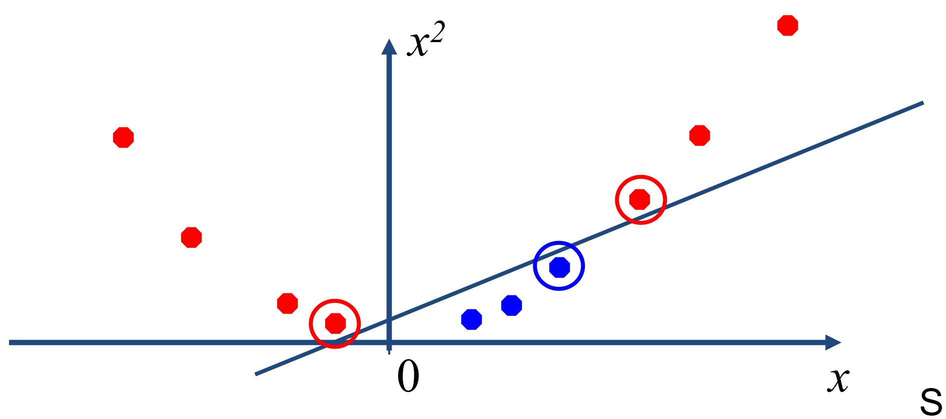
- Linearly separable dataset in 1D:



- Non-separable dataset in 1D:



- We can map the data to a *higher-dimensional space*:



Slide credit: Andrew Moore

The kernel trick

- **General idea:** the original input space can always be mapped to some higher-dimensional feature space where the training set is separable

- **The kernel trick:** instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}, \mathbf{y}) = \boldsymbol{\varphi}(\mathbf{x}) \cdot \boldsymbol{\varphi}(\mathbf{y})$$

- (to be valid, the kernel function must satisfy *Mercer's condition*)

The kernel trick

- Linear SVM decision function:

$$\mathbf{w}\cdot\mathbf{x}+b=\sum_i \underbrace{\alpha_i}_{\text{learned weight}} y_i \underbrace{\mathbf{x}_i}_{\text{Support vector}}\cdot\mathbf{x}+b$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

The kernel trick

- Linear SVM decision function:

$$\mathbf{w}\cdot\mathbf{x}+b=\sum_i \alpha_i y_i \mathbf{x}_i\cdot\mathbf{x}\ +b$$

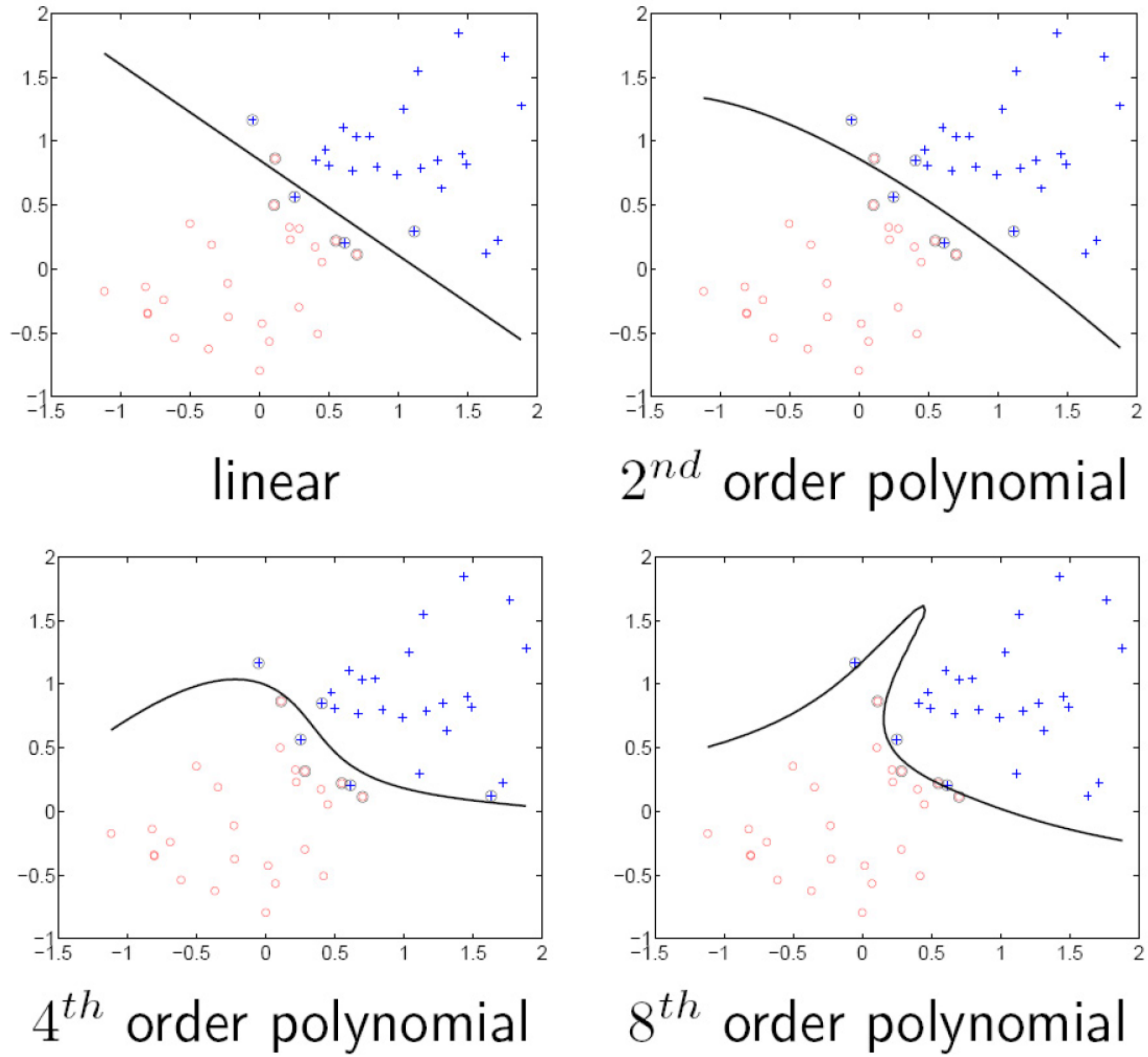
- Kernel SVM decision function:

$$\sum_i \alpha_i y_i \varphi(\mathbf{x}_i)\cdot\varphi(\mathbf{x})\ +b=\sum_i \alpha_i y_i K(\mathbf{x}_i,\mathbf{x})\ +b$$

- This gives a nonlinear decision boundary in the original feature space

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

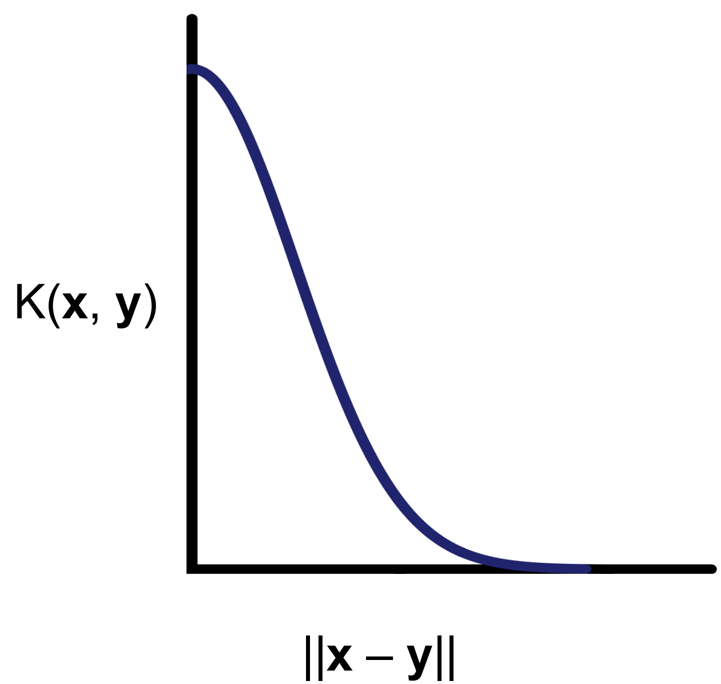
Polynomial kernel: $K(\mathbf{x},\mathbf{y}) = (c + \mathbf{x}\cdot\mathbf{y})^d$



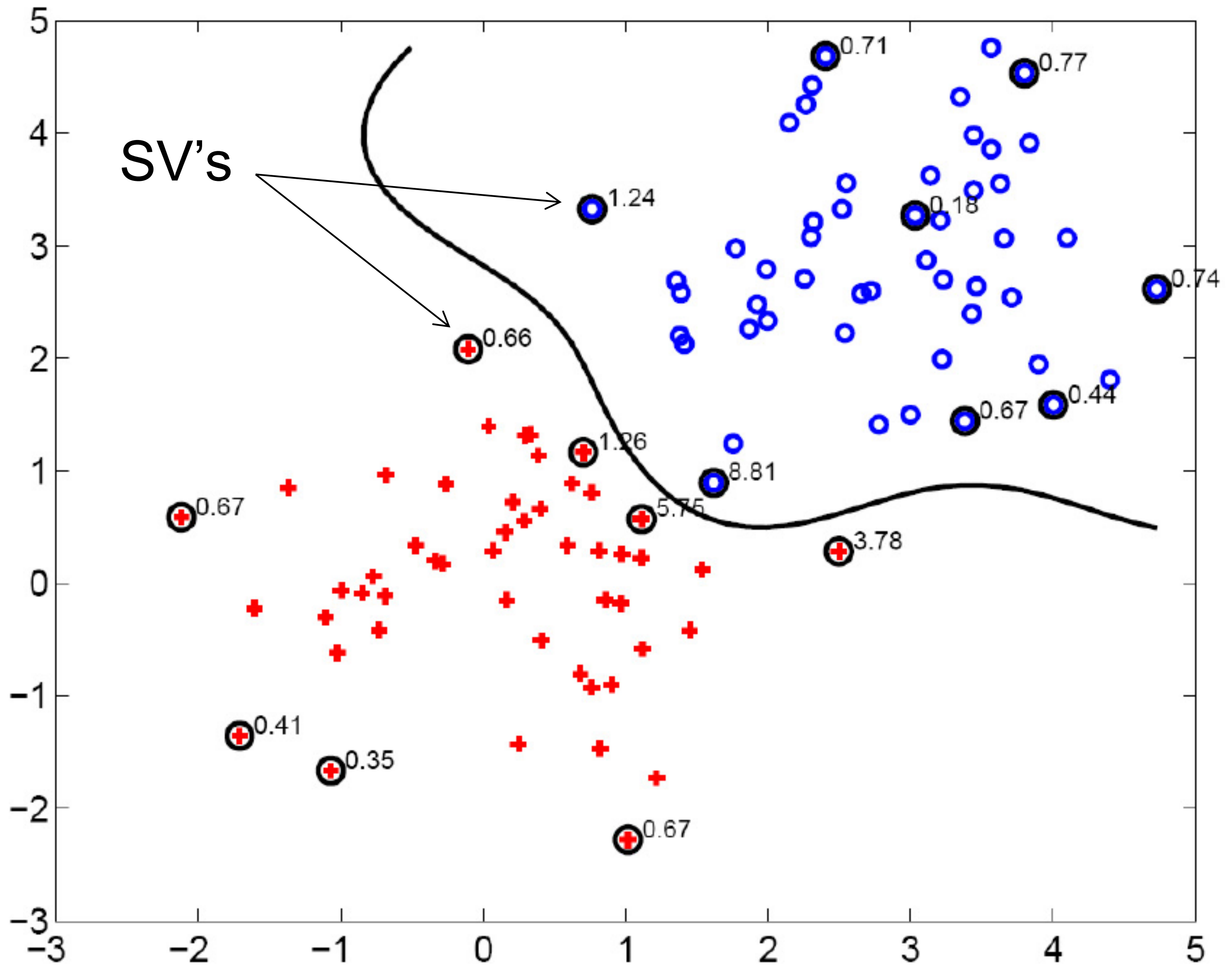
Gaussian kernel

- Also known as the radial basis function (RBF) kernel:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{\sigma^2} \|\mathbf{x} - \mathbf{y}\|^2 \right)$$



Gaussian kernel

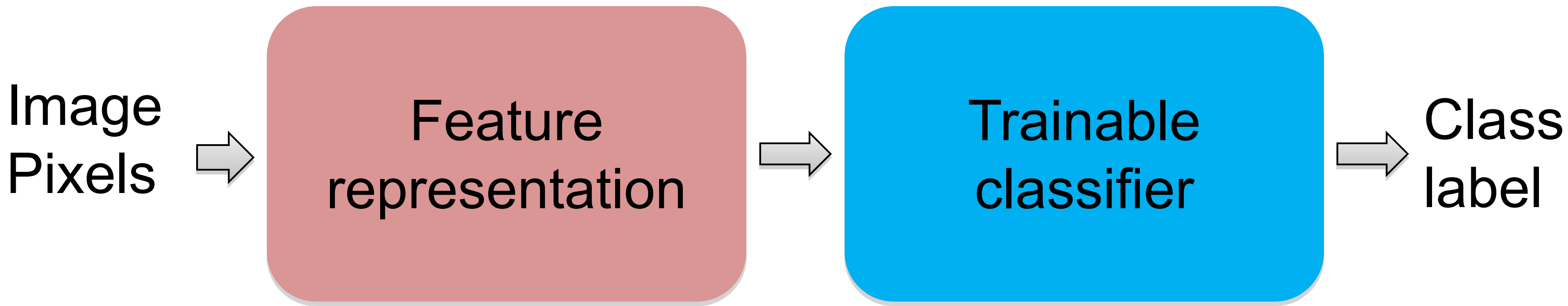


SVMs: Pros and cons

- Pros
  - Kernel-based framework is very powerful, flexible
  - Training is convex optimization, globally optimal solution can be found
  - Amenable to theoretical analysis
  - SVMs work very well in practice, even with very small training sample sizes

- Cons
  - No “direct” multi-class SVM, must combine two-class SVMs (e.g., with one-vs-others)
  - Computation, memory (esp. for nonlinear SVMs)

“Classic” recognition pipeline



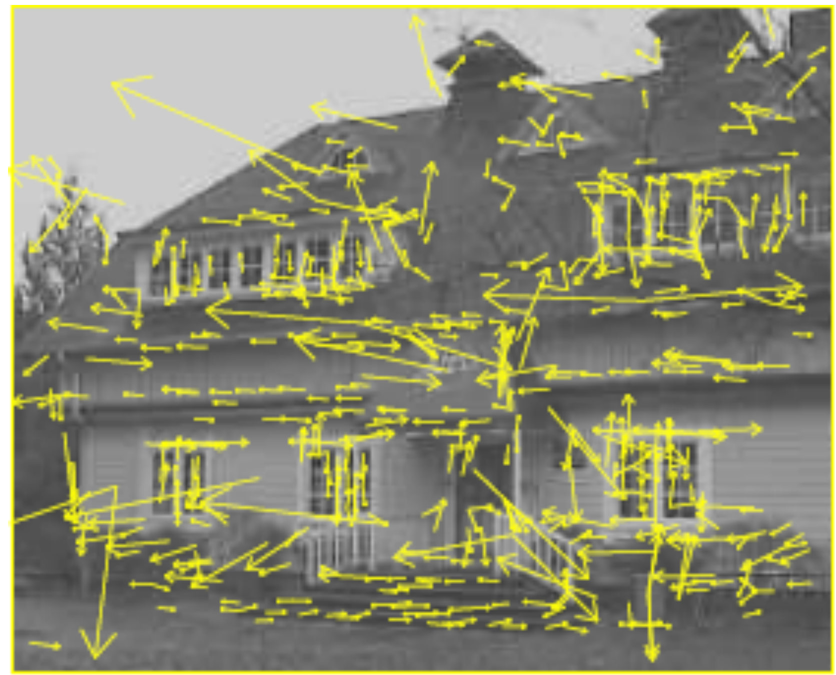
Building an Image Classifier

Training Data

Test Data

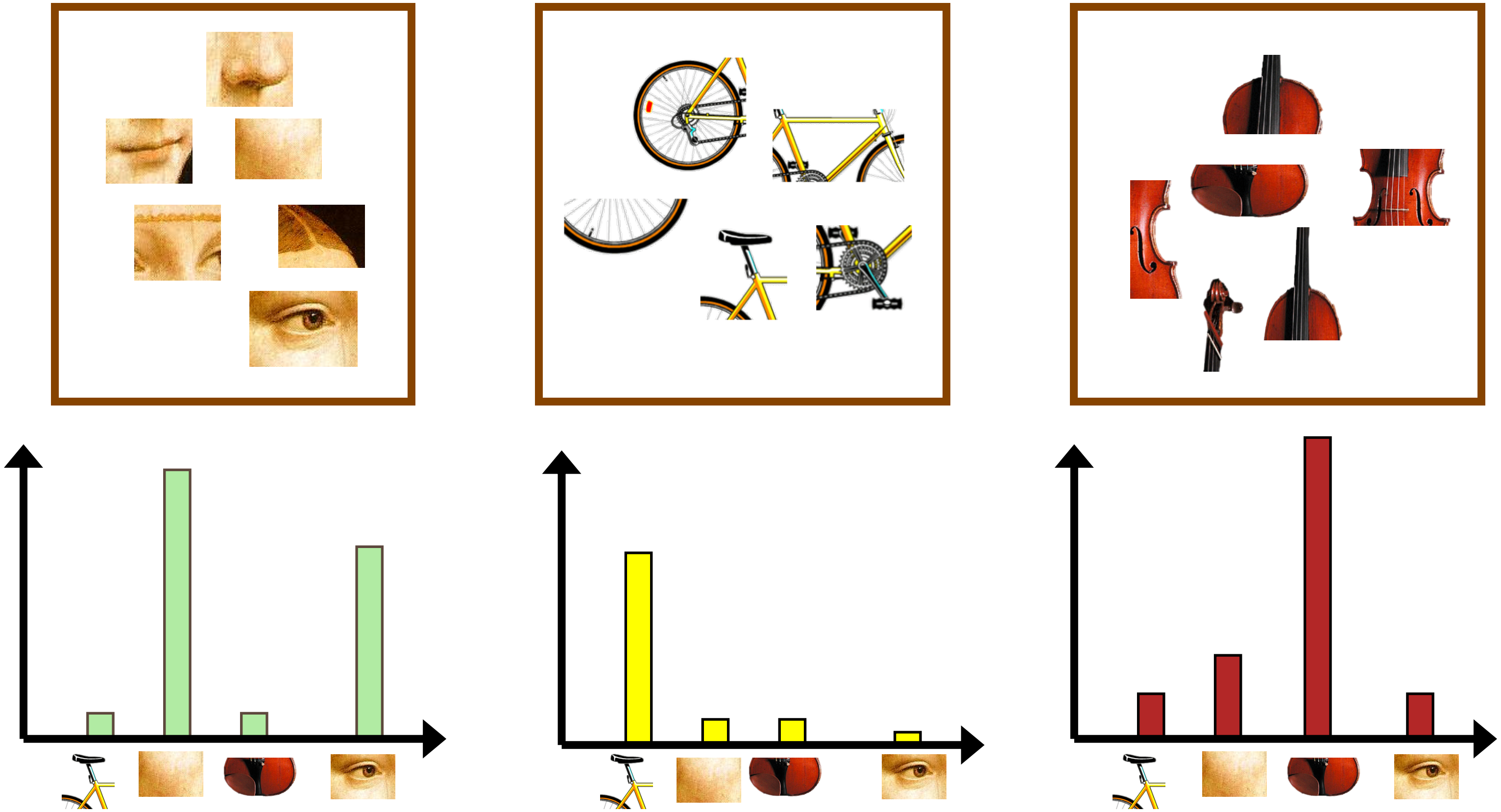
Building an Image Classifier

- Raw Pixels?
- SIFT features at keypoints?



Bag of features: Outline

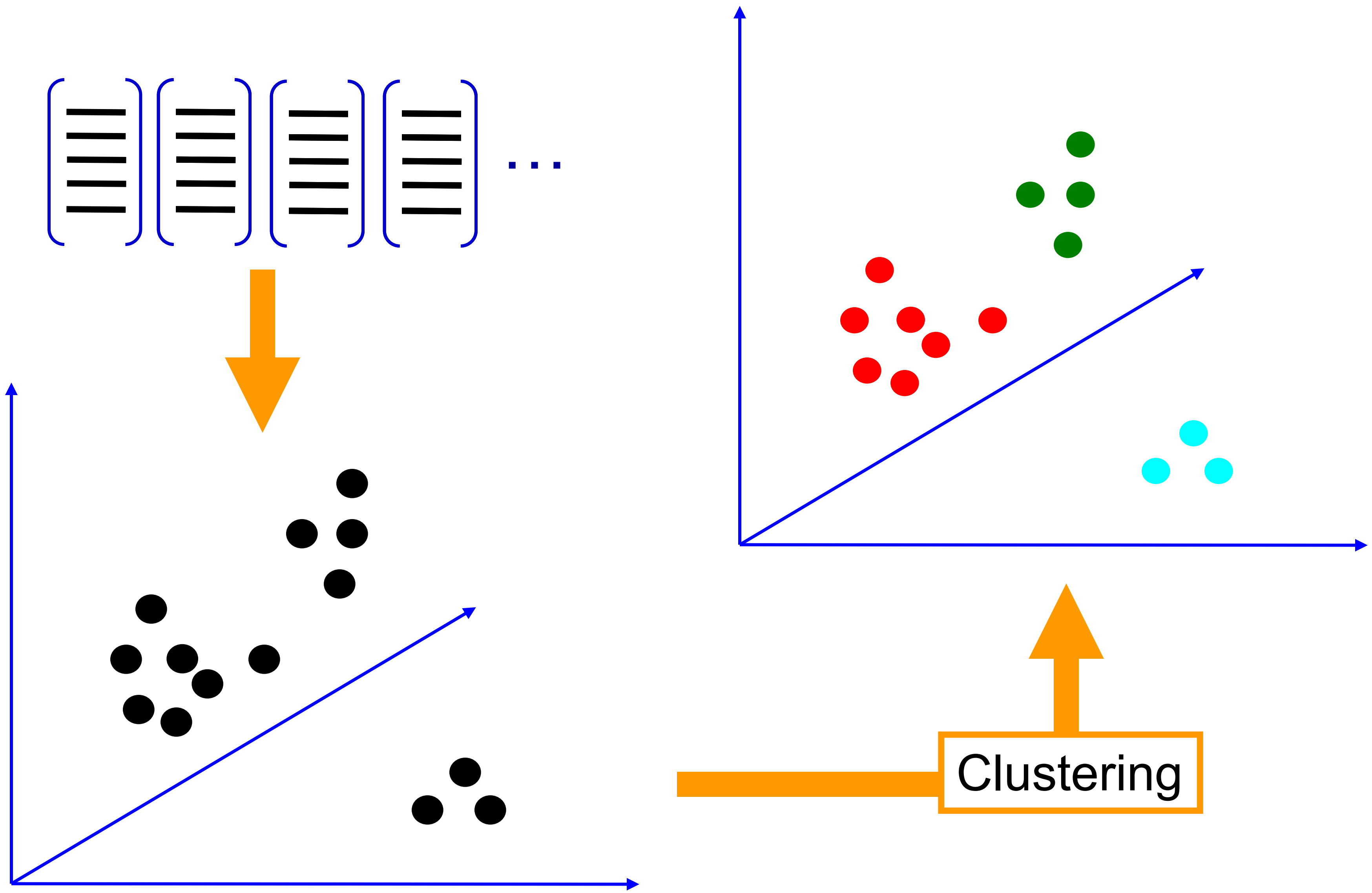
1. Extract local features
2. Learn "visual vocabulary"
3. Quantize local features using visual vocabulary
4. Represent images by frequencies of "visual words"



1. Local feature extraction

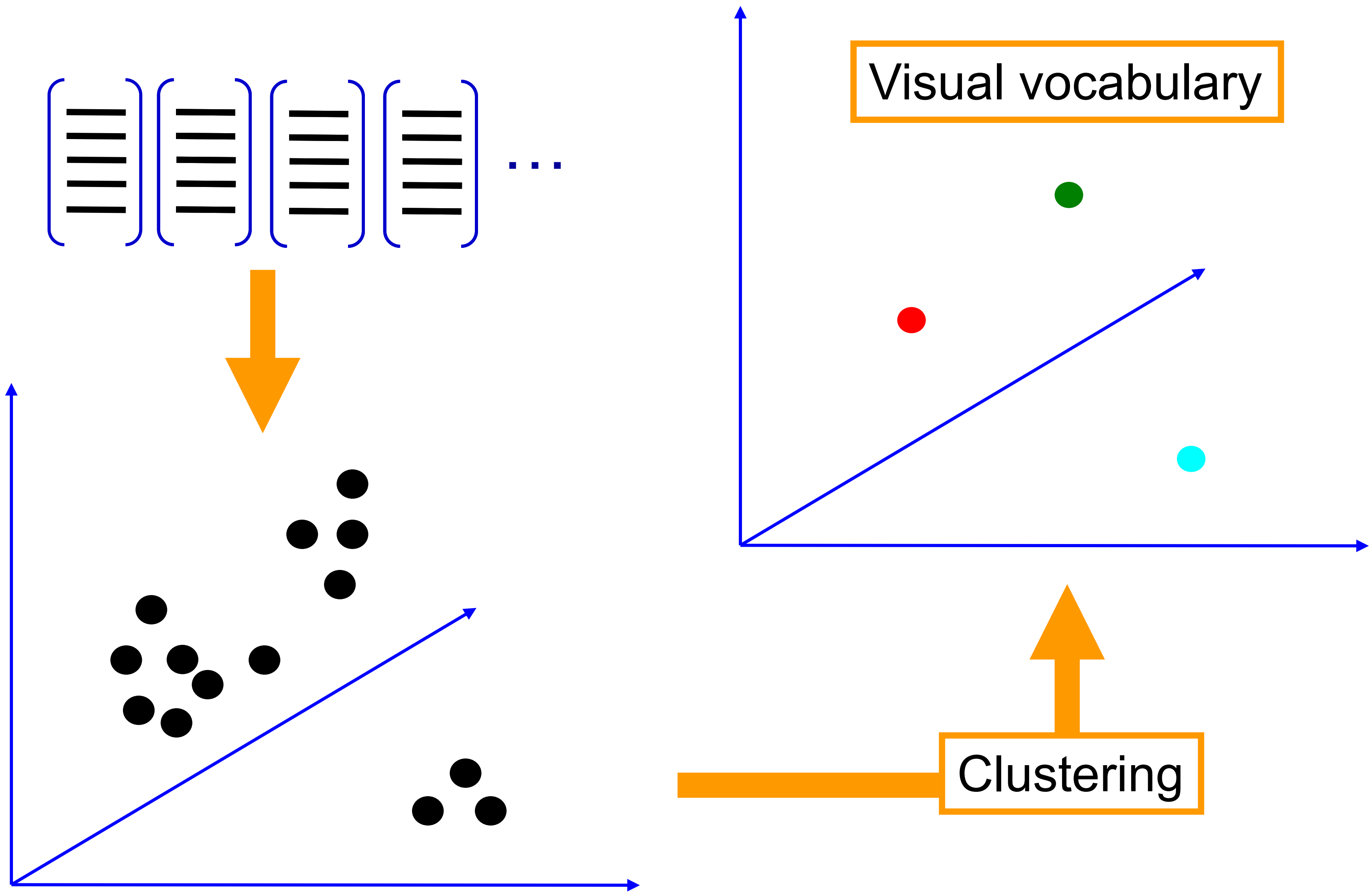
- Sample patches and extract descriptors

2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

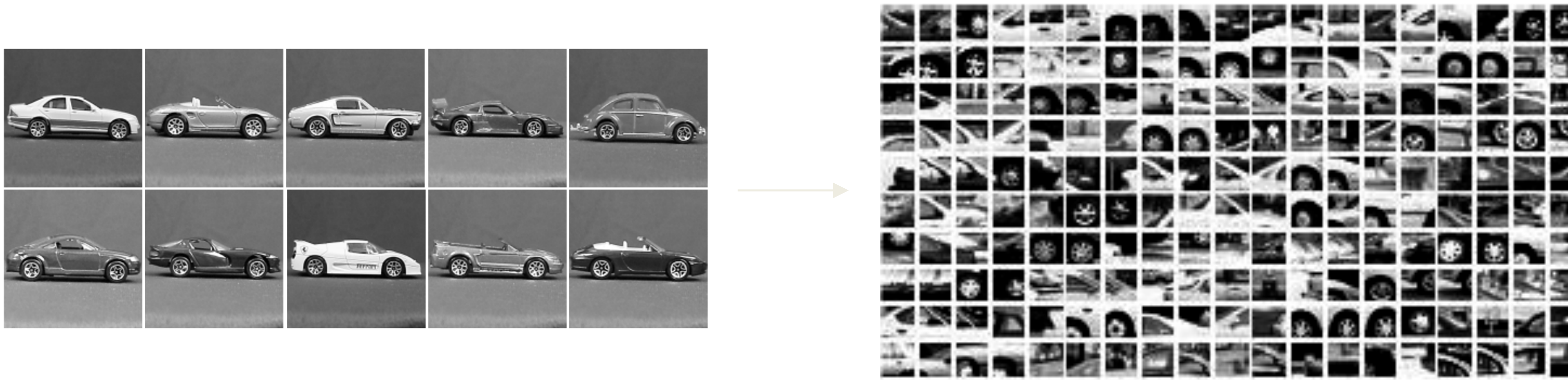
K-means clustering

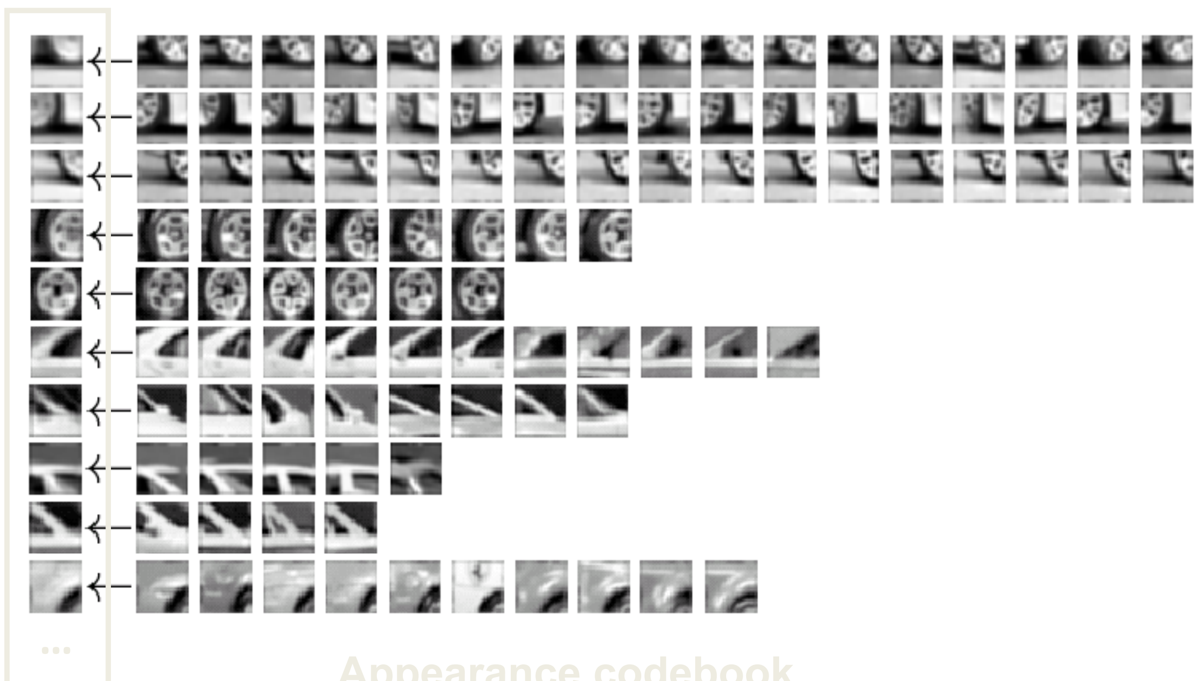
- Want to minimize sum of squared Euclidean distances between features $\mathbf{x}_i$ and their nearest cluster centers $\mathbf{m}_k$

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (\mathbf{x}_i - \mathbf{m}_k)^2$$

- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each feature to the nearest center
  - Recompute each cluster center as the mean of all features assigned to it

Visual vocabularies

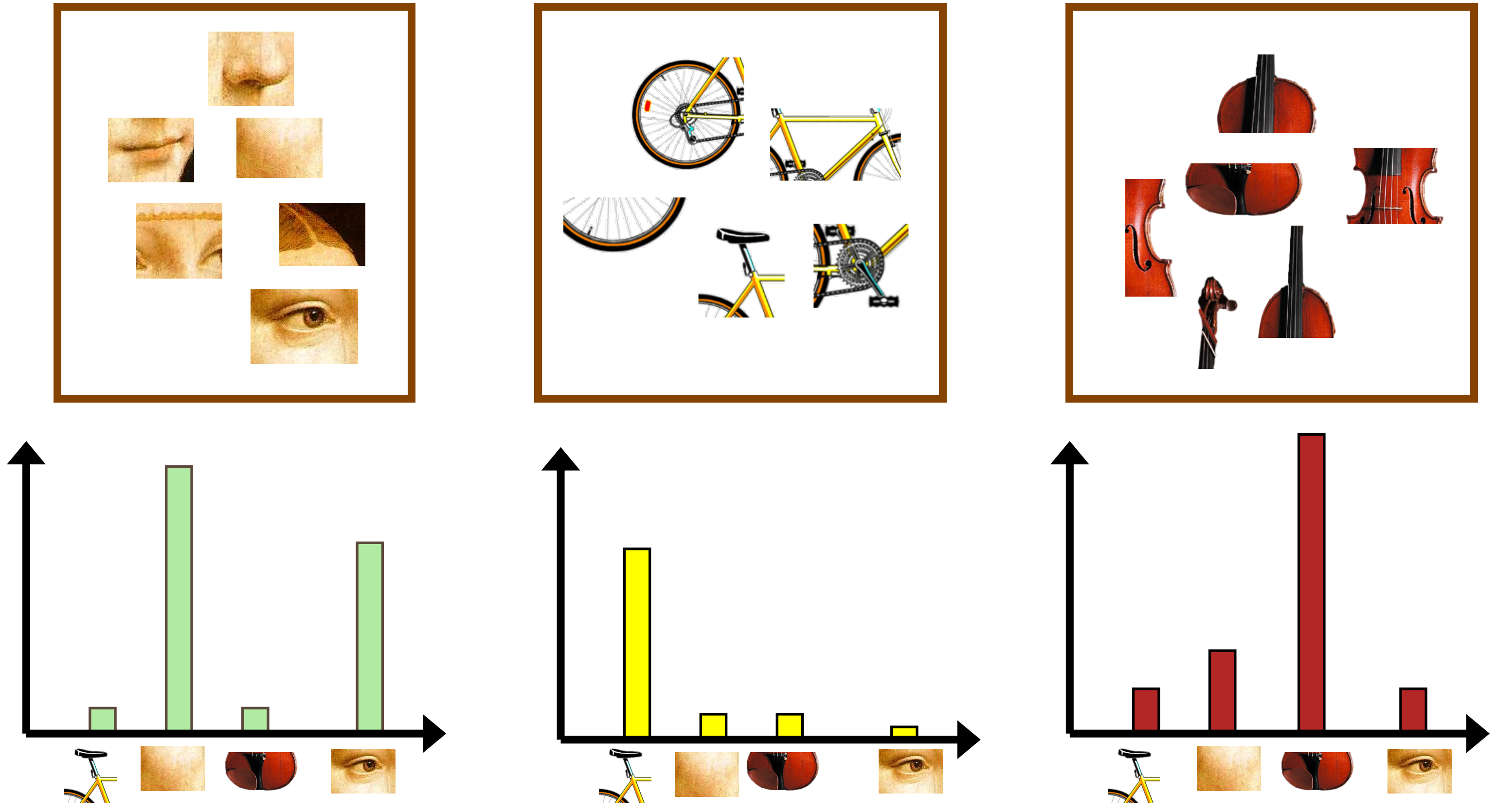




Source: B. Leibe

Bag of features: Outline

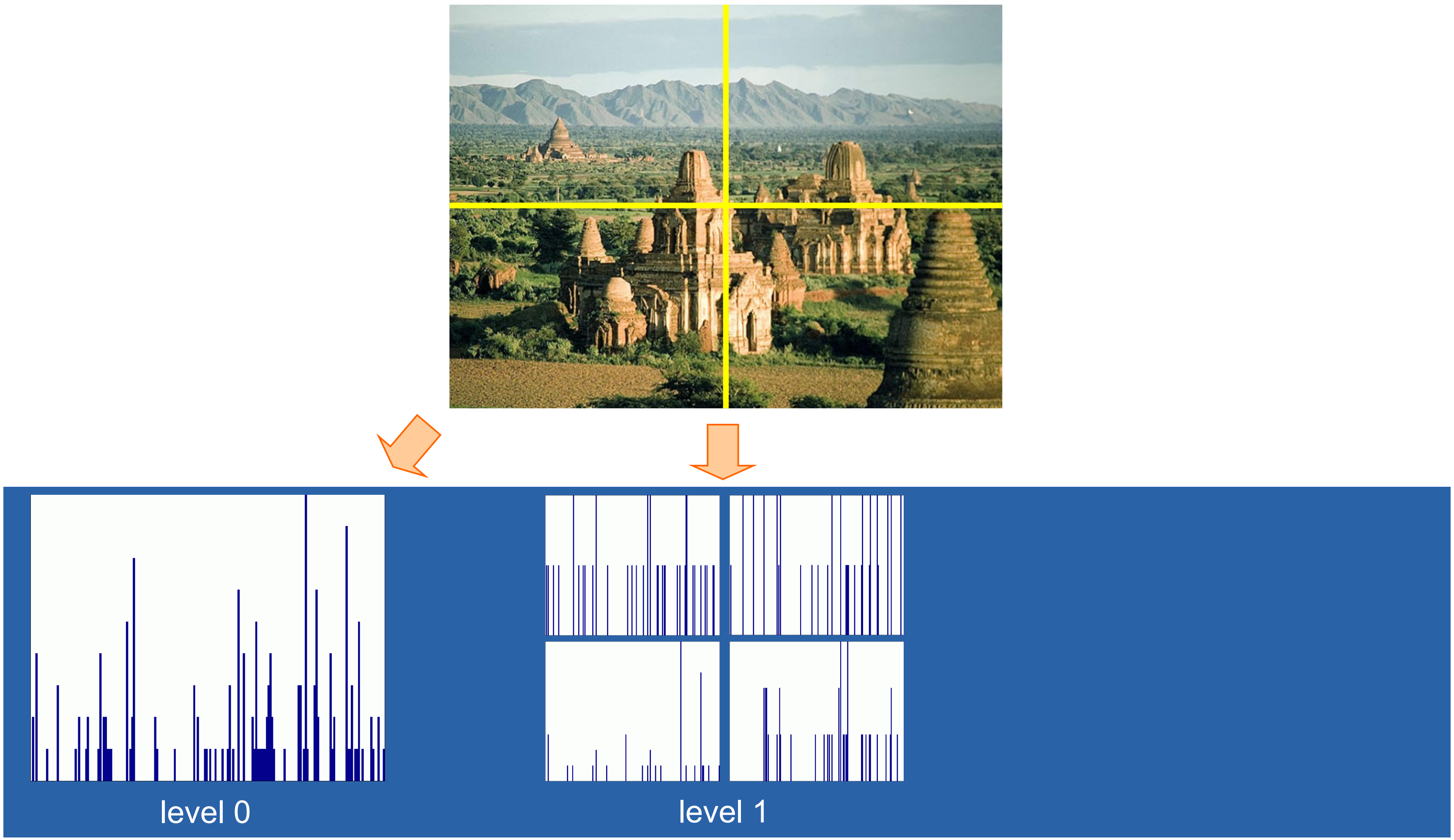
1. Extract local features
2. Learn "visual vocabulary"
3. **Quantize local features using visual vocabulary**
4. **Represent images by frequencies of "visual words"**



Spatial pyramids

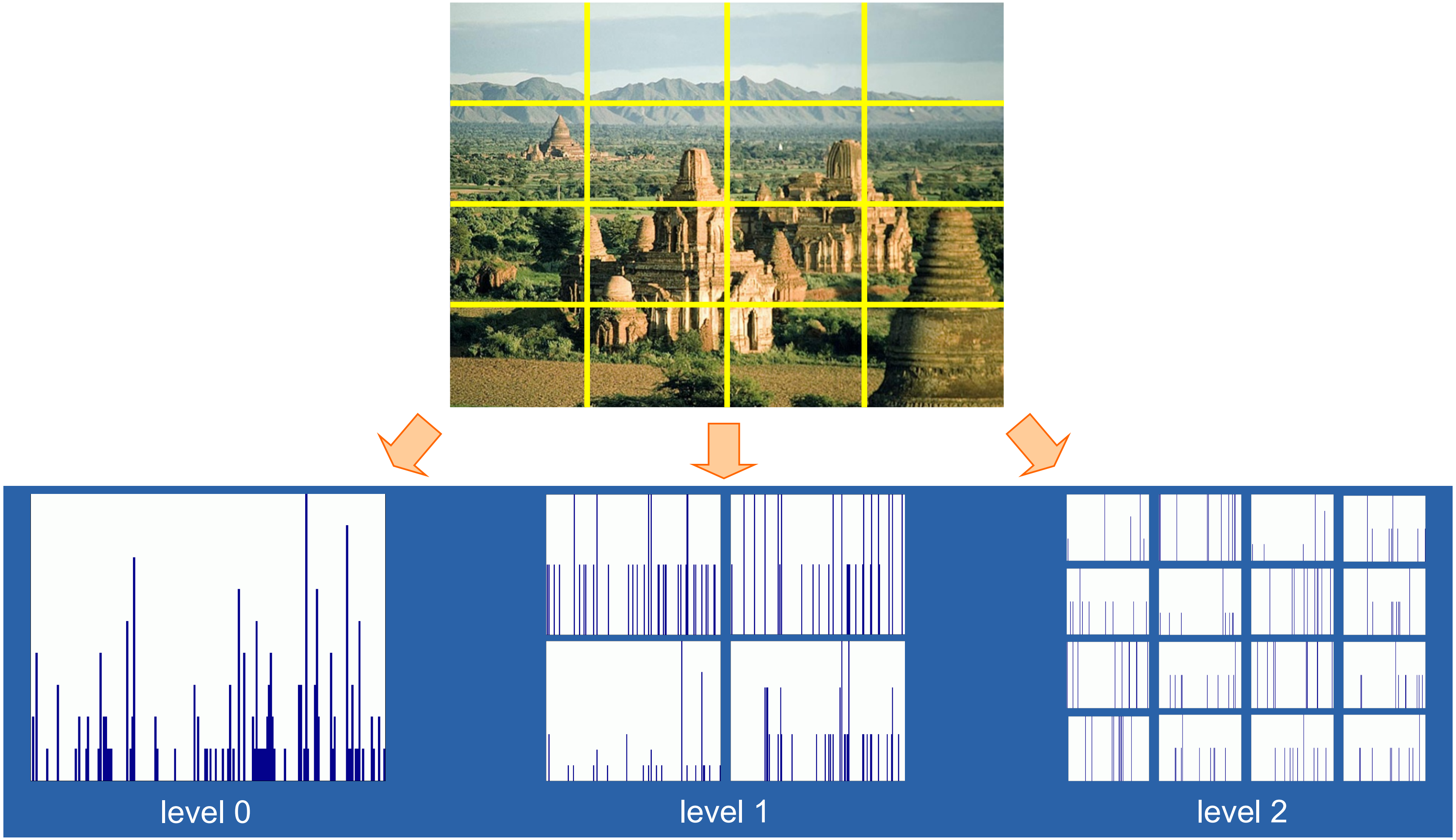
Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids



Lazebnik, Schmid & Ponce (CVPR 2006)

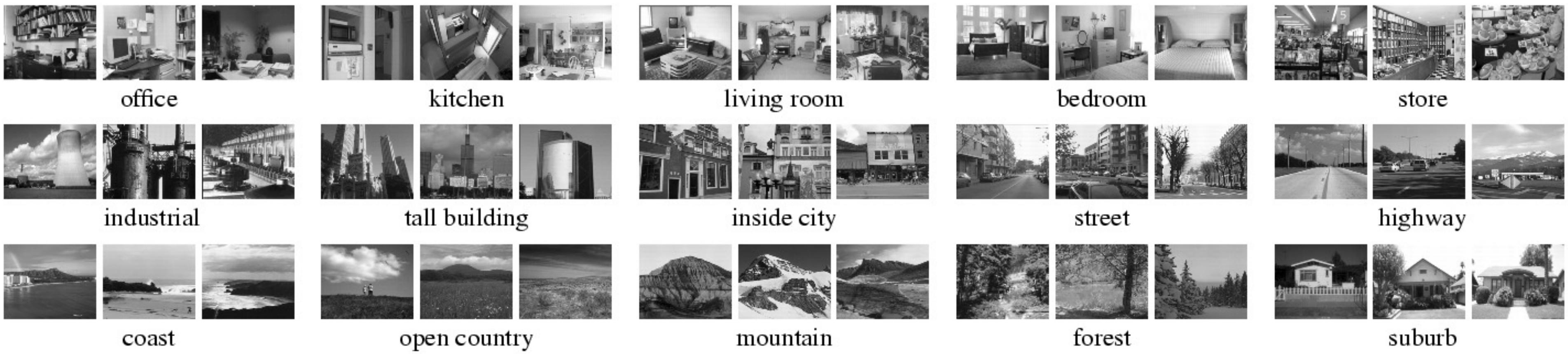
Spatial pyramids



Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids

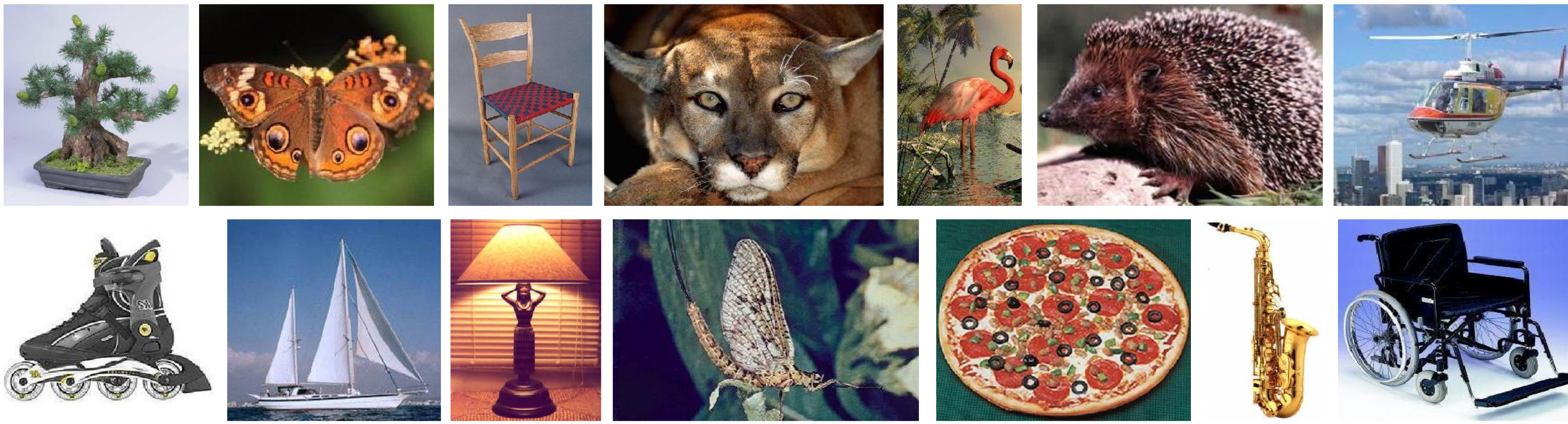
- Scene classification results



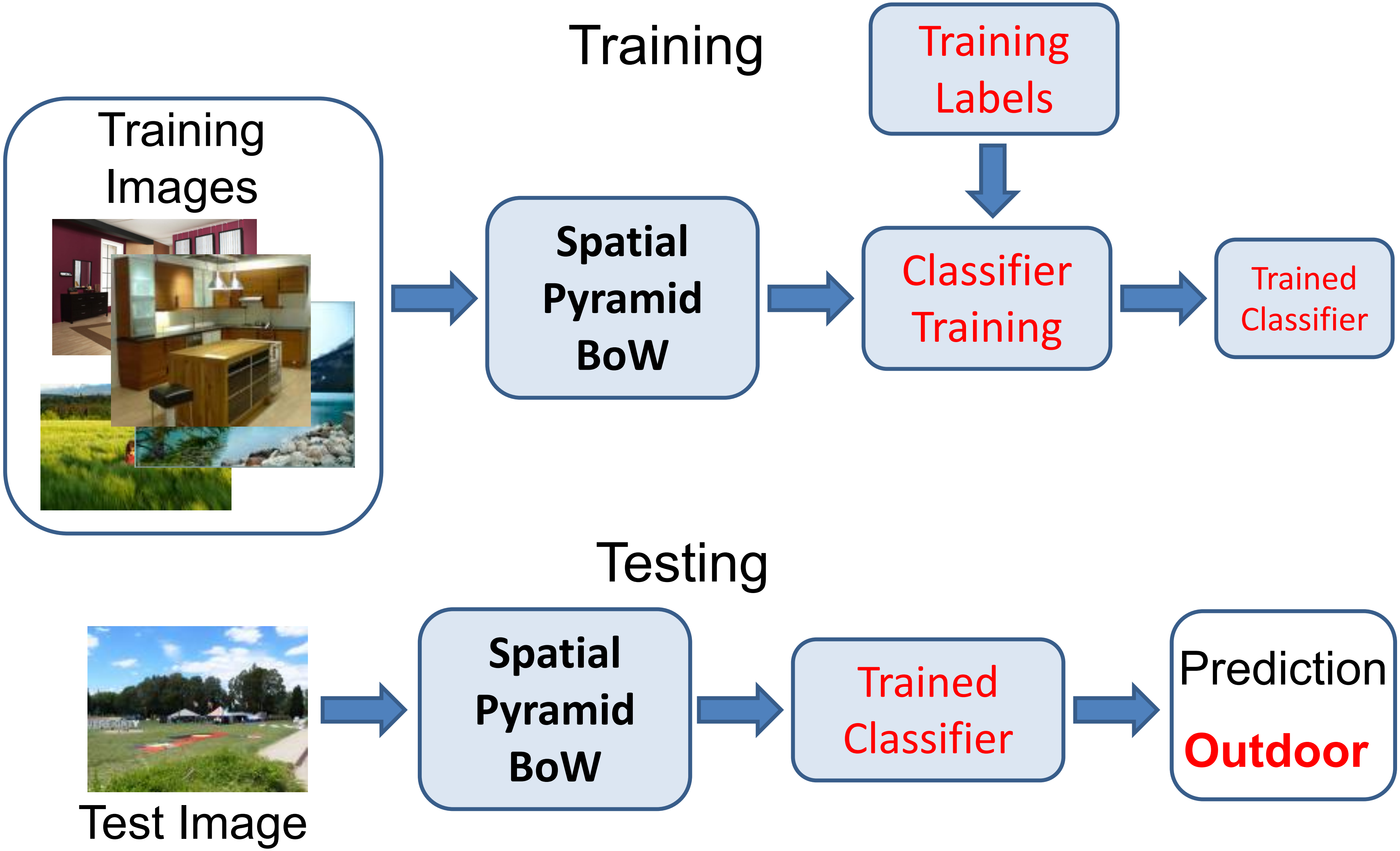
	Weak features (vocabulary size: 16)		Strong features (vocabulary size: 200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0 (1×1)	45.3 ±0.5		72.2 ±0.6	
1 (2×2)	53.6 ±0.3	56.2 ±0.6	77.9 ±0.6	79.0 ±0.5
2 (4×4)	61.7 ±0.6	64.7 ±0.7	79.4 ±0.3	**81.1** ±0.3
3 (8×8)	63.3 ±0.8	**66.8** ±0.6	77.2 ±0.4	80.7 ±0.3

Spatial pyramids

- Caltech101 classification results



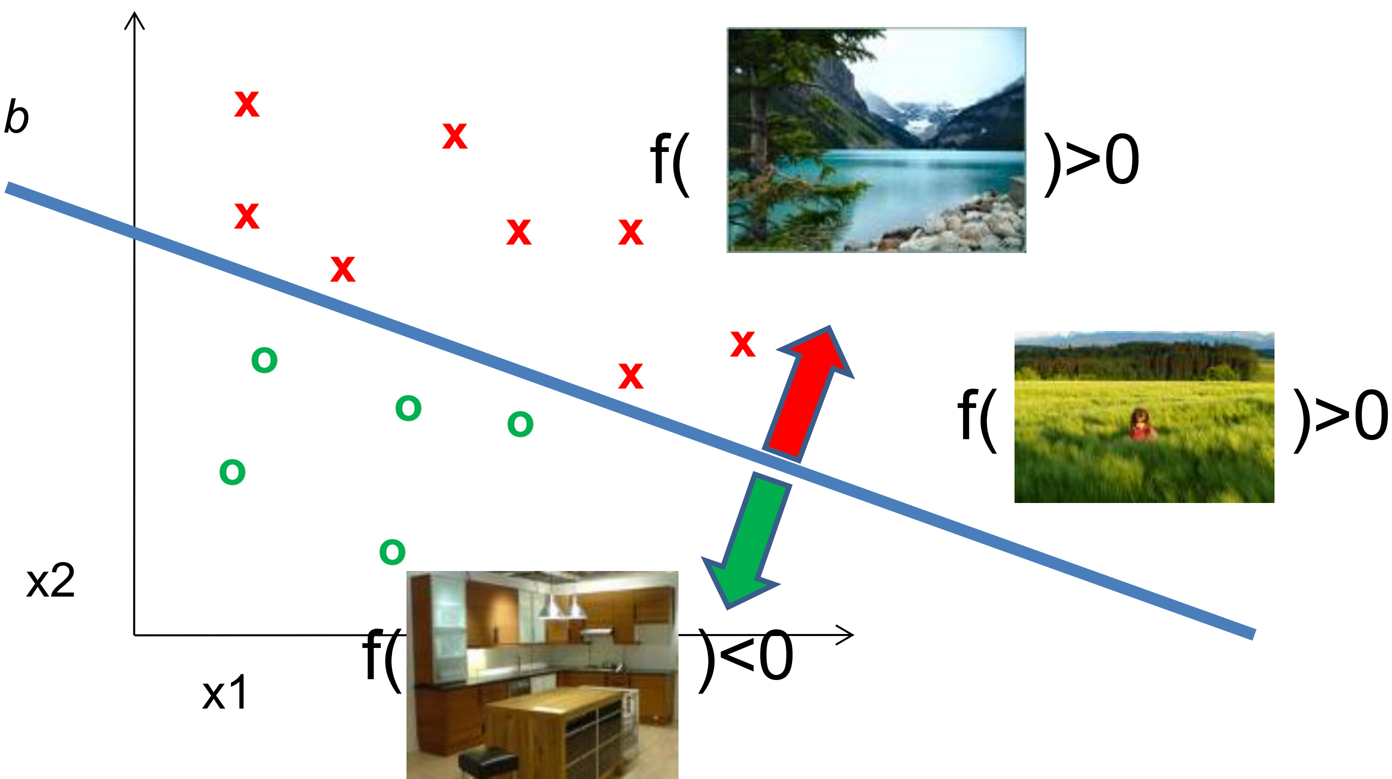
	Weak features (16)		Strong features (200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0	15.5 ±0.9		41.2 ±1.2	
1	31.4 ±1.2	32.8 ±1.3	55.9 ±0.9	57.0 ±0.8
2	47.2 ±1.1	49.3 ±1.4	63.6 ±0.9	**64.6** ±0.8
3	52.2 ±0.8	**54.0** ±1.1	60.3 ±0.9	64.6 ±0.7


Training
Training Labels
Training Images
Spatial Pyramid BoW
Classifier Training
Trained Classifier
Testing
Spatial Pyramid BoW
Trained Classifier
Prediction
Outdoor
Test Image

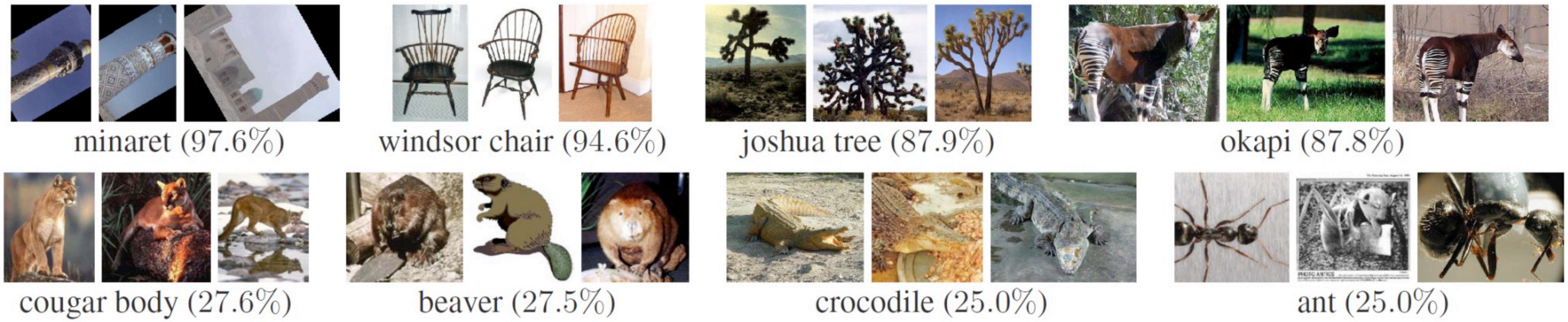
Linear SVM classifier

Find the hyperplane that separate examples of different categories

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$



Spatial Pyramids Results

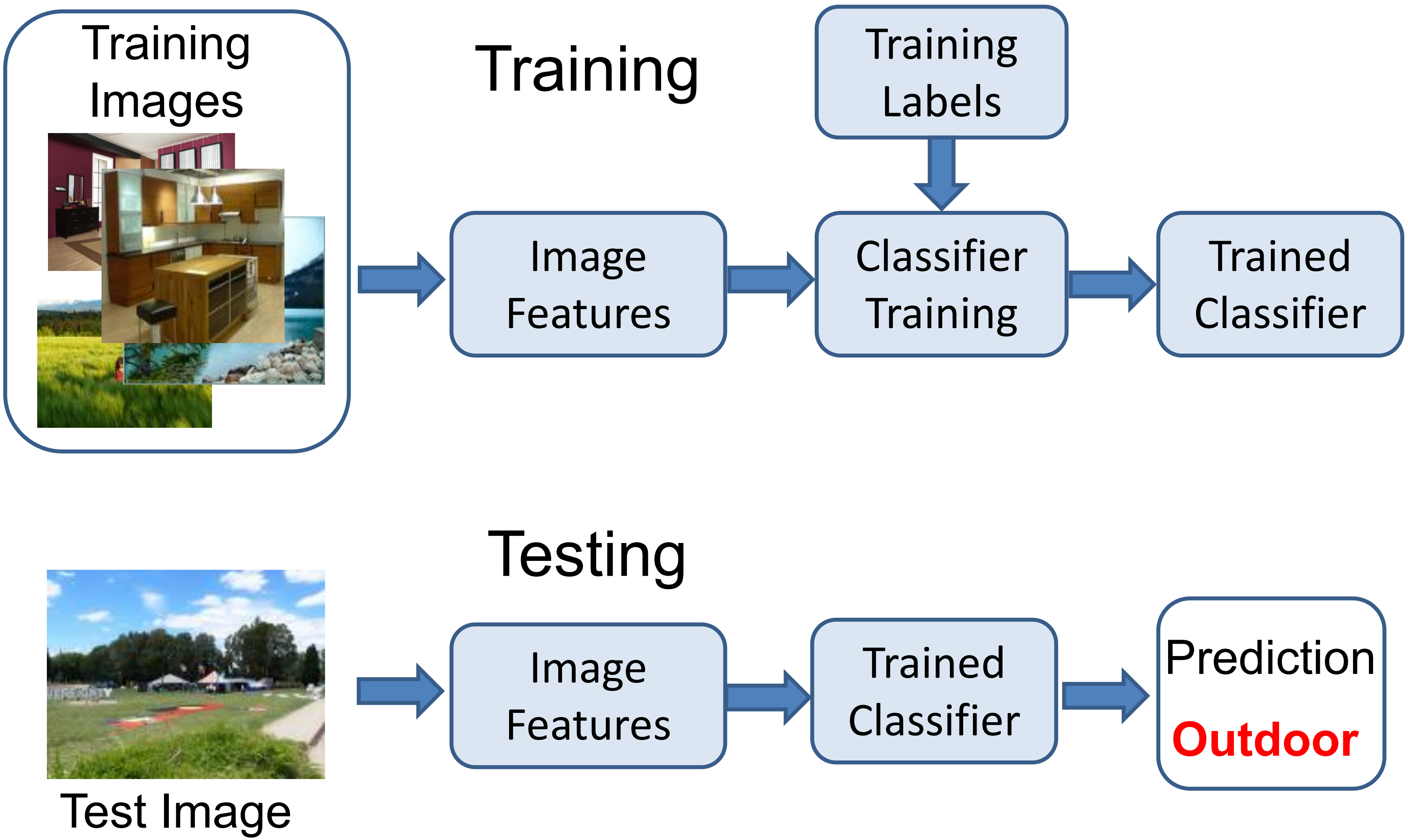


	Weak features		Strong features (200)	
L	Single-level	Pyramid	Single-level	Pyramid
0	15.5 ±0.9		41.2 ±1.2	
1	31.4 ±1.2	32.8 ±1.3	55.9 ±0.9	57.0 ±0.8
2	47.2 ±1.1	49.3 ±1.4	63.6 ±0.9	**64.6** ±0.8
3	52.2 ±0.8	**54.0** ±1.1	60.3 ±0.9	64.6 ±0.7

Table 2. Classification results for the Caltech-101 database.

Now let’s go back to the core concepts of image categorization in general

Categorization involves **features** and a classifier



Q: What are good features for…

- recognizing a beach?






Q: What are good features for…

- recognizing cloth fabric?






Q: What are good features for…

- recognizing a mug?






What are the right features?

Depend on what you want to know!

- Object: shape
  - Local shape info, shading, shadows, texture
- Scene : geometric layout
  - linear perspective, gradients, line segments
- Material properties: albedo, feel, hardness
  - Color, texture
- Action: motion
  - Optical flow, tracked points

General principles of features

- Coverage
  - Ensure that all relevant info is captured
- Concision
  - Minimize number of features without sacrificing coverage
- Directness
  - Ideal features are independently useful for prediction

It's hard to design good features for a specific problem

Many machine learning algorithms solve for both feature and classifier parameters, such as

- Decision trees
- Random forests
- Multilayer neural networks (including CNNs)

Classifiers

Goal: From labeled training samples, learn parameters of a scoring/decision function that is likely to predict the correct label on test samples

Typical assumptions:

- Training and test samples drawn from same distribution (i.i.d.)
- Training labels are correct

Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Deep networks
- Etc.

Which is the best one?

Classifiers: three main options

- **Nearest neighbor**: take a vote from K closest neighbors
  - Can learn features or distance measure

- **Linear**: score is linear combination of features
  - SVM, perceptron, naïve bayes, logistic regression
  - Others learn features and then apply linear classifier (e.g. deep network, random forest)

- **Structured prediction**: score an interdependent set of labels
  - E.g. label body part positions
  - Structured SVM, CNN, graphical model algorithms

Generalization Theory

It’s not enough to do well on the training set: also should make good predictions for new examples

No Free Lunch Theorem

Bias-Variance Trade-off

$$E(MSE) = noise^2 + bias^2 + variance$$

Unavoidable error

Error due to incorrect assumptions

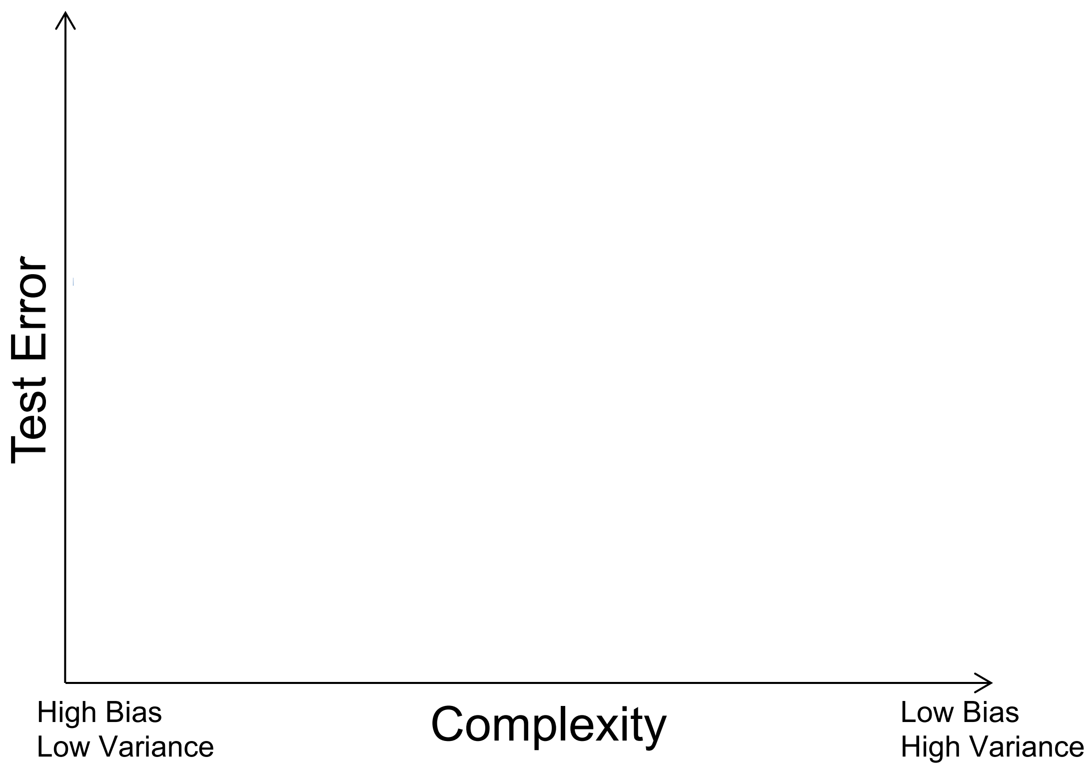
Error due to variance parameter estimates from training samples

See the following for explanation of bias-variance (also Bishop's "Neural Networks" book):

- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

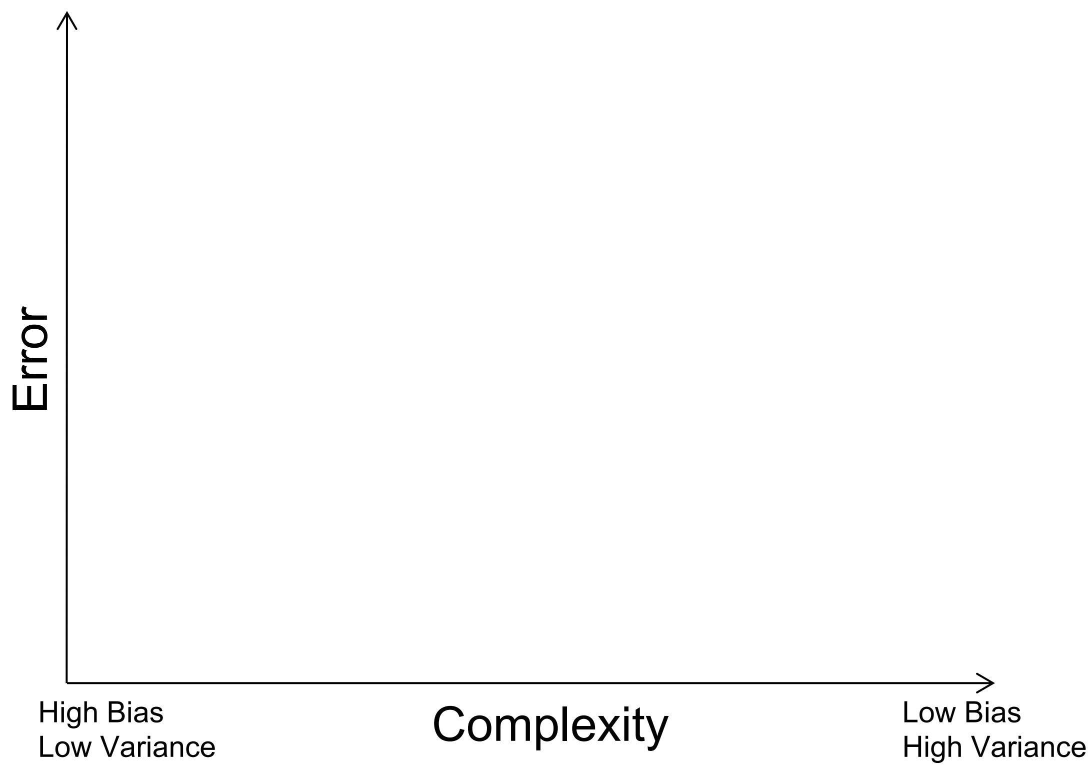
Bias and Variance

$$\text{Error} = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

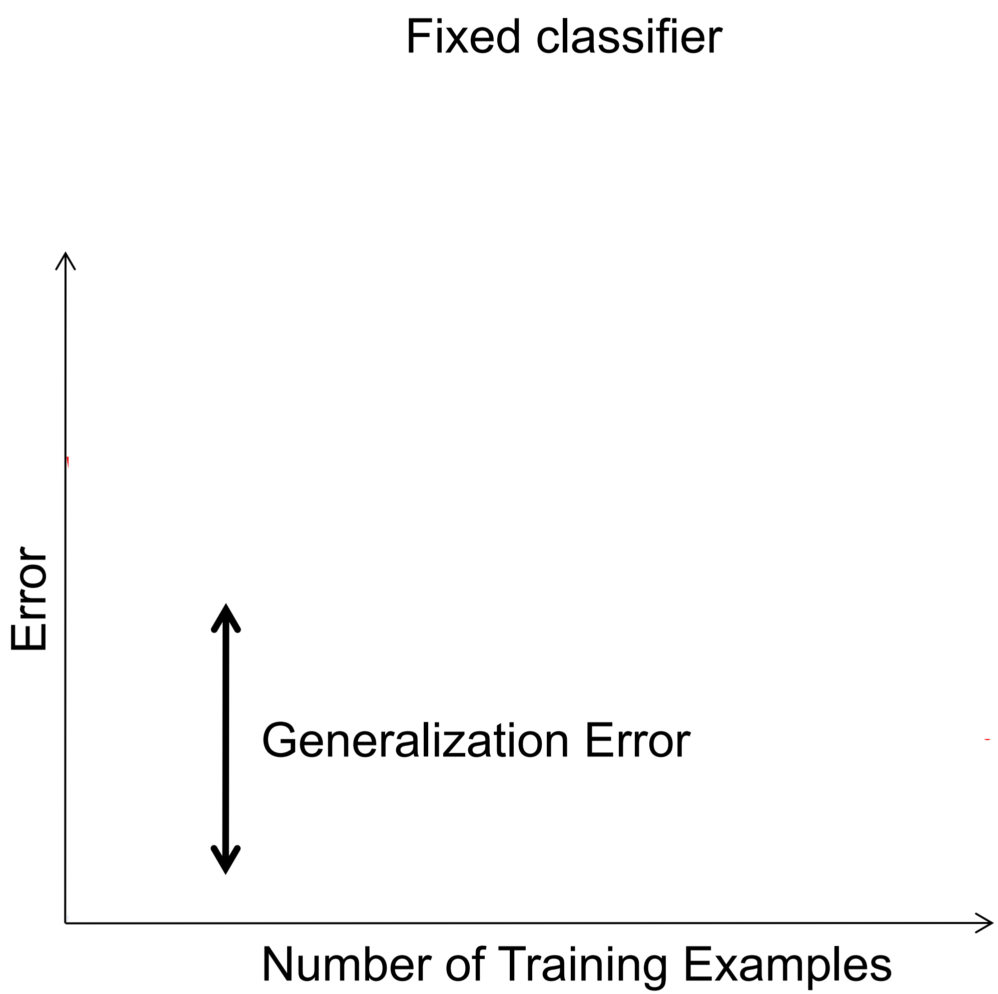


Choosing the trade-off

- Need validation set
- Validation set is separate from the test set



Effect of Training Size



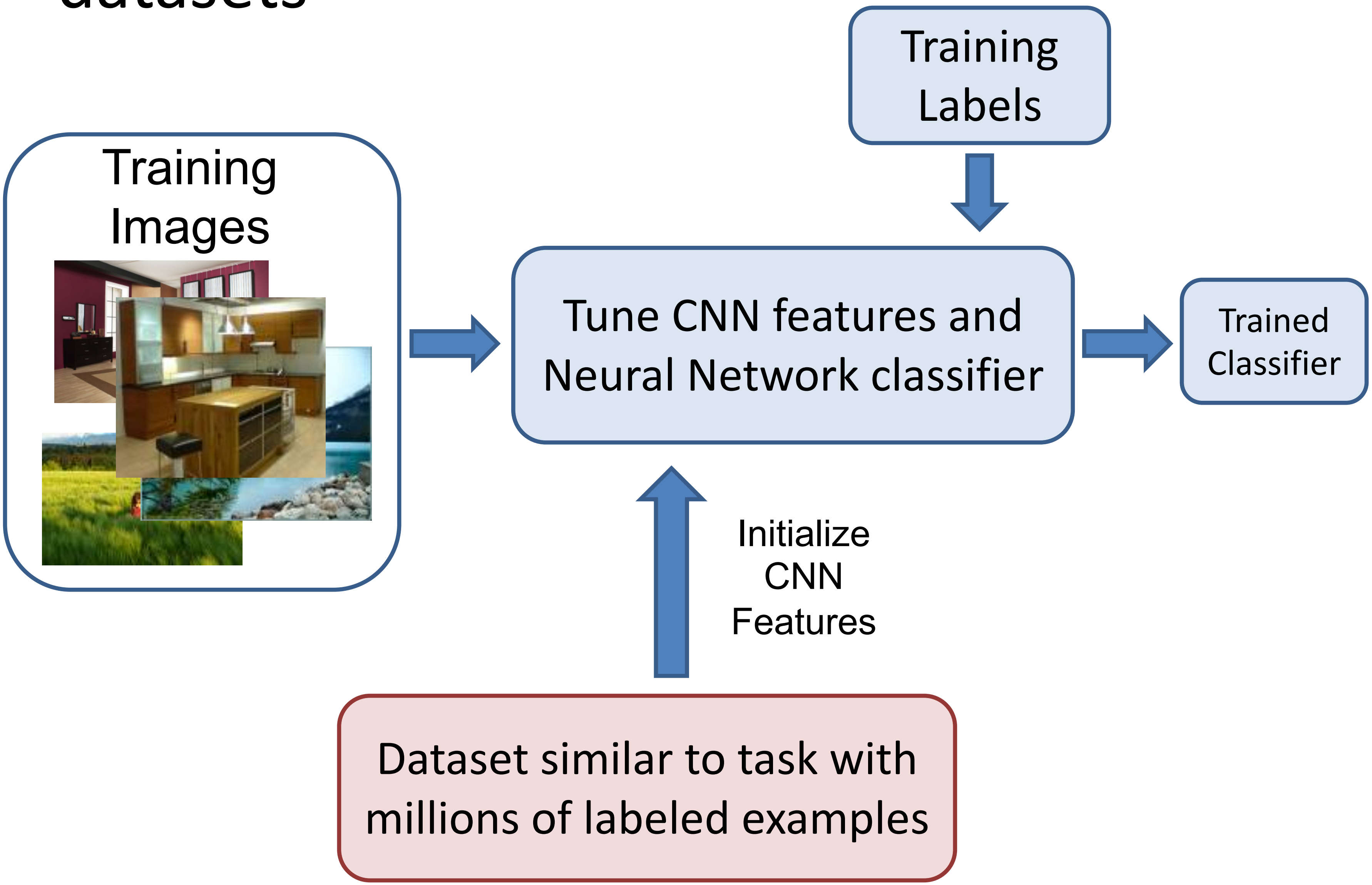
Characteristics of vision learning problems

- Lots of continuous features
  - E.g., HOG template may have 1000 features
  - Spatial pyramid may have ~15,000 features
- Imbalanced classes
  - often limited positive examples, practically infinite negative examples
- Difficult prediction tasks

When a massive training set is available

- Relatively new phenomenon
  - MNIST (handwritten letters) in 1990s, LabelMe in 2000s, ImageNet (object images) in 2009, …
- Want classifiers with low bias (high variance ok) and reasonably efficient training
- Very complex classifiers with simple features are often effective
  - Random forests
  - Deep convolutional networks

New training setup with moderate sized datasets



Practical tips

- Preparing features for linear classifiers
  - Often helps to make zero-mean, unit-dev
  - For non-ordinal features, convert to a set of binary features

- Selecting classifier meta-parameters (e.g., regularization weight)
  - Cross-validation: split data into subsets; train on all but one subset, test on remaining; repeat holding out each subset
    - Leave-one-out, 5-fold, etc.

- Most popular classifiers in vision
  - *SVM*: linear for when fast training/classification is needed; performs well with lots of weak features
  - *Logistic Regression*: outputs a probability; easy to train and apply
  - *Nearest neighbor*: hard to beat if there is tons of data (e.g., character recognition)
  - *Boosted stumps or decision trees*: applies to flexible features, incorporates feature selection, powerful classifiers
  - *Random forests*: outputs probability; good for simple features, tons of data
  - *Deep networks / CNNs*: flexible output; learns features; adapt existing network (which is trained with tons of data) or train new with tons of data

- Always try at least two types of classifiers

Next class

Deep convolutional neural networks (CNNs)

- Architecture (Alexnet, VGG, Inception, Resnet)
- Optimizer (stochastic gradient descent, Adam)
- Transfer
- Why it works