Introduction to Computer Vision

Instructors: Jean Ponce and Matthew Trager
jean.ponce@inria.fr, matthew.trager@cims.nyu.edu

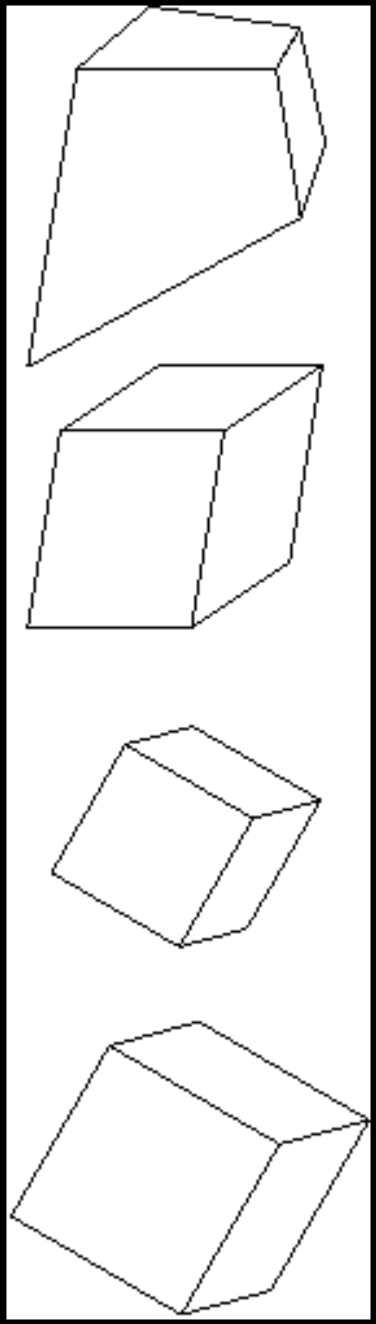
TAs: Jiachen (Jason) Zhu and Sahar Siddiqui
jiachen.zhu@nyu.edu, ss12414@nyu.edu

Outline

- Wrap-up of SfM
- Recognition, classical methods, and supervised learning
- Introduction to neural networks

Types of ambiguity

Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$	Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$	Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$	Preserves angles, ratios of length
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$	Preserves angles, lengths



- With no constraints on the camera calibration matrix or on the scene, we get a projective reconstruction
- Need additional information to upgrade the reconstruction to affine, similarity, or Euclidean

Structure from Motion

Given m pictures of n points, can we recover

- the three-dimensional configuration of these points?
- the camera configurations?

The Euclidean (perspective) Structure-from-Motion Problem

Given m (internally) calibrated perspective images of n fixed points $\boldsymbol{P}_j$ we can write

$$\begin{cases} u_{ij} = \dfrac{\boldsymbol{m}_{i1} \cdot \boldsymbol{P}_j}{\boldsymbol{m}_{i3} \cdot \boldsymbol{P}_j} \\ v_{ij} = \dfrac{\boldsymbol{m}_{i2} \cdot \boldsymbol{P}_j}{\boldsymbol{m}_{i3} \cdot \boldsymbol{P}_j} \end{cases} \quad \text{for} \quad i = 1, \ldots, m \quad \text{and} \quad j = 1, \ldots, n.$$

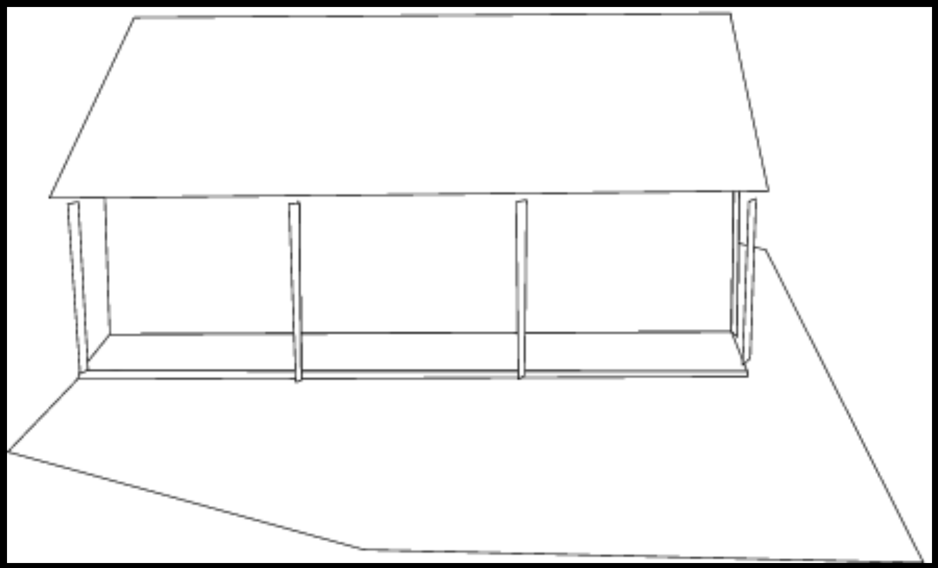
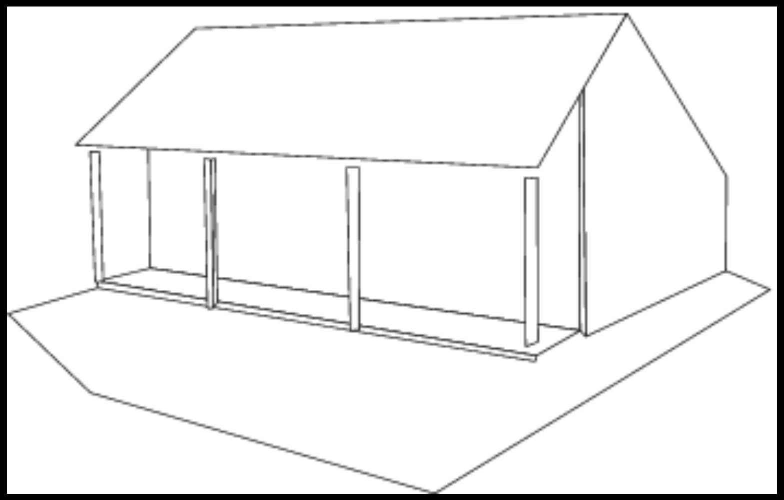
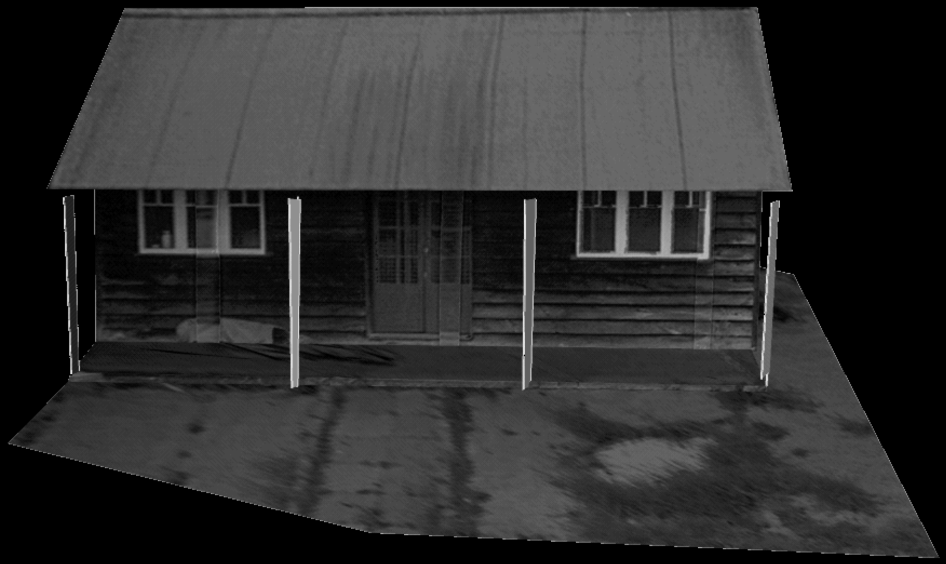
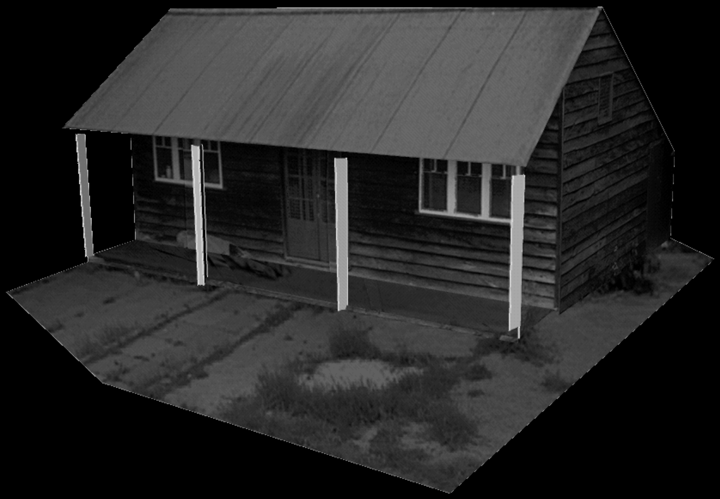
Problem: estimate the m 3x4 matrices $\mathcal{M}_i = [\mathrm{R}_i \; t_i]$ and

the n positions $\boldsymbol{P}_j$ from the mn correspondences $\boldsymbol{p}_{ij}$.

$2mn$ equations in $11m$ (or rather $5m$)+$3n$ unknowns

Overconstrained problem, that can be solved using (non-linear) least squares!

Euclidean (= similarity) ambiguity

The Projective Structure-from-Motion Problem

Given m uncalibrated perspective images of n fixed points $\boldsymbol{P}_j$ we can write

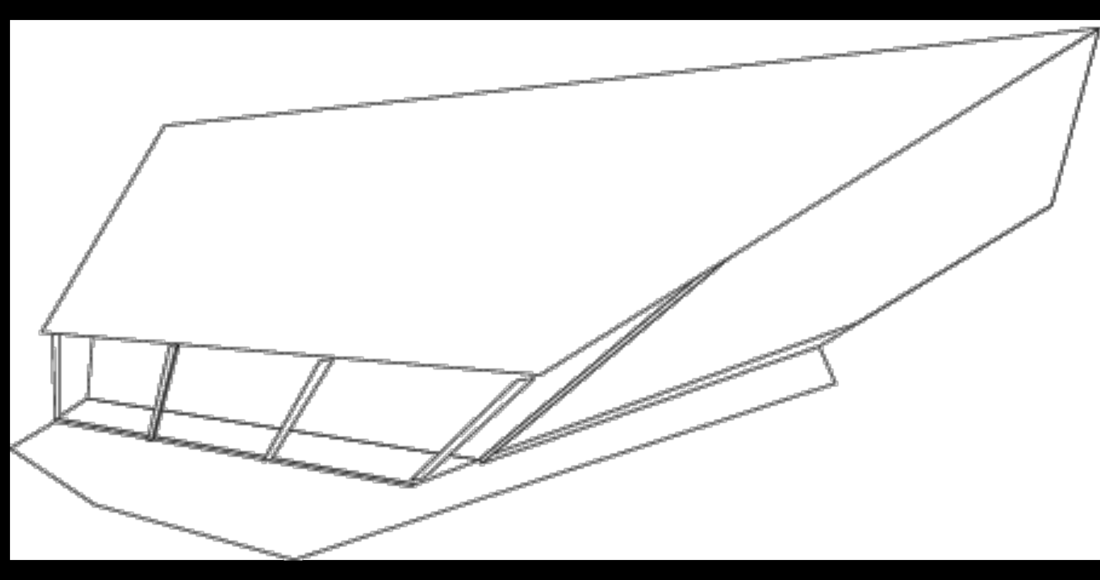
$$\begin{cases} u_{ij} = \dfrac{\boldsymbol{m}_{i1} \cdot \boldsymbol{P}_j}{\boldsymbol{m}_{i3} \cdot \boldsymbol{P}_j} \\ v_{ij} = \dfrac{\boldsymbol{m}_{i2} \cdot \boldsymbol{P}_j}{\boldsymbol{m}_{i3} \cdot \boldsymbol{P}_j} \end{cases} \quad \text{for} \quad i = 1, \ldots, m \quad \text{and} \quad j = 1, \ldots, n.$$

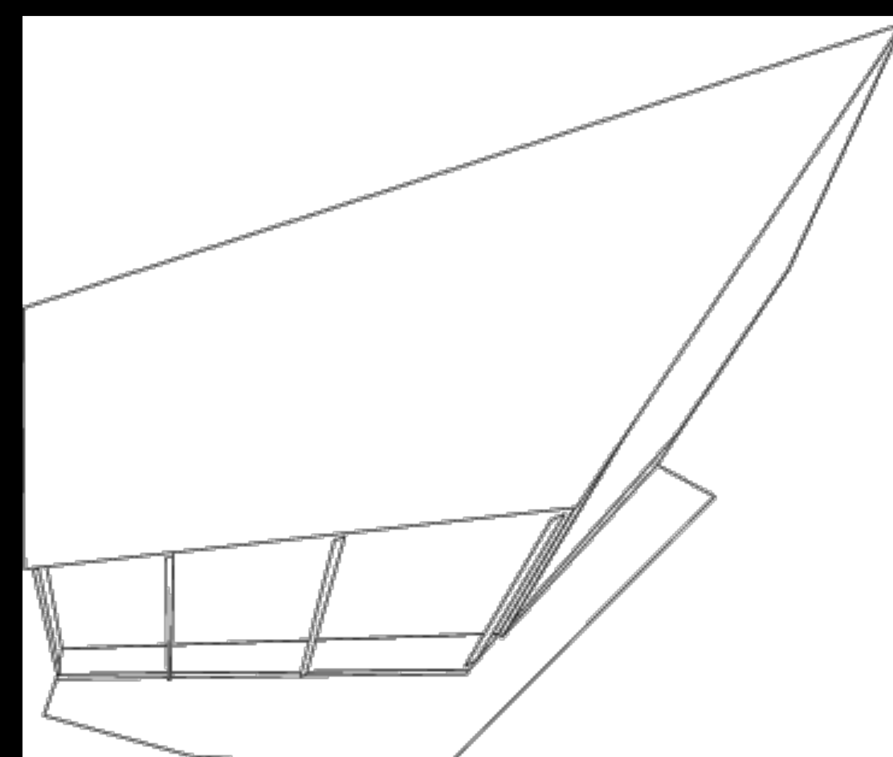
Problem: estimate the m 3x4 matrices $\mathcal{M}_i$ and

the n positions $\boldsymbol{P}_j$ from the mn correspondences $\boldsymbol{p}_{ij}$.

$2mn$ equations in $11m+3n$ unknowns

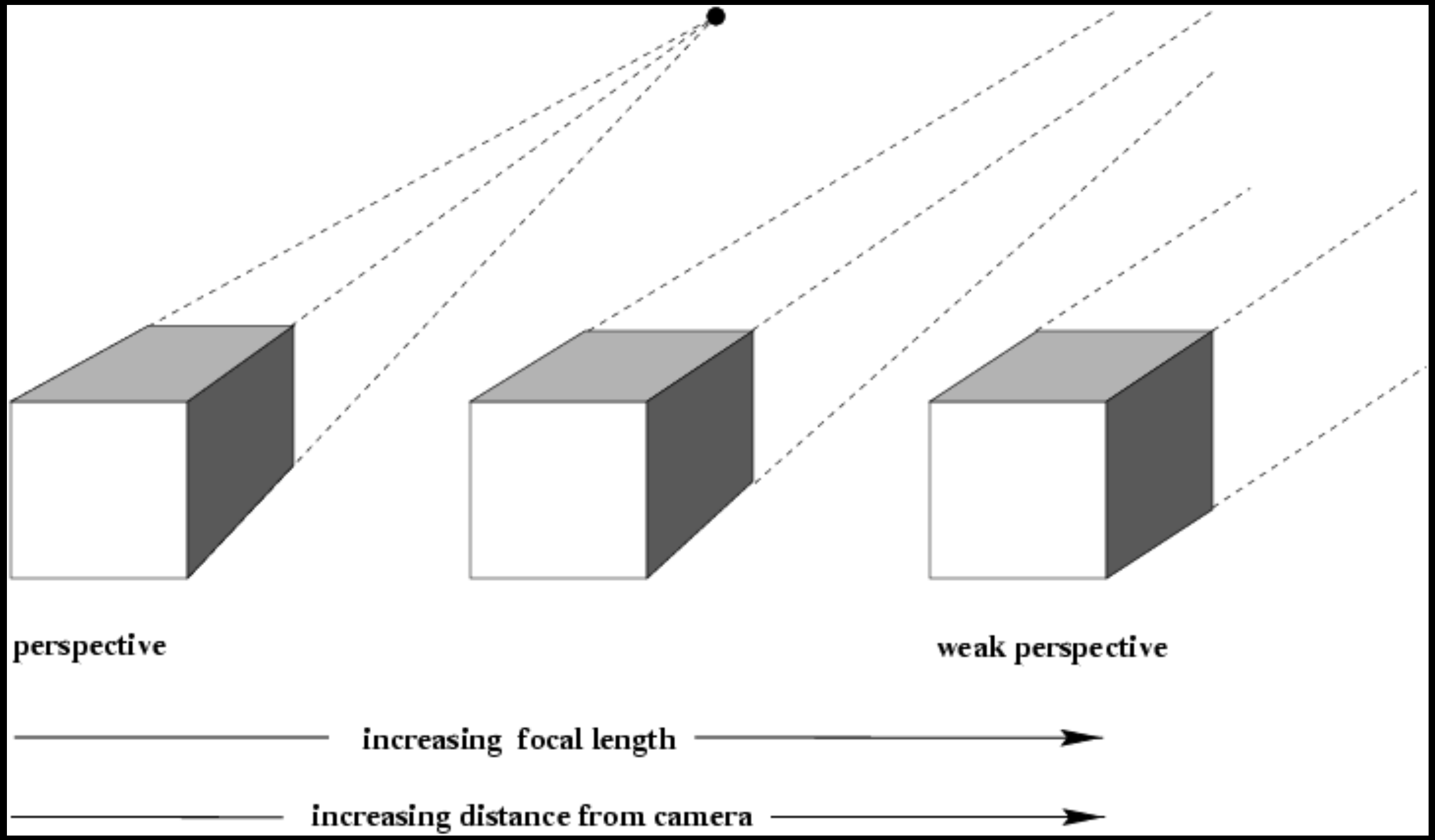
→ Overconstrained problem, that can be solved using (non-linear) least squares!





Structure from motion

- Let us now look at simpler, affine cameras



center at
infinity

The *Affine* Structure-from-Motion Problem

Given m images of n fixed points P_j we can write

$$\boldsymbol{p}_{ij} = \mathcal{M}_i \begin{pmatrix} \boldsymbol{P}_j \\ 1 \end{pmatrix} = \mathcal{A}_i \boldsymbol{P}_j + \boldsymbol{b}_i \quad \text{for} \quad i = 1, \ldots, m \quad \text{and} \quad j = 1, \ldots, n.$$

Problem: estimate the m 2x4 matrices $\mathcal{M}_i$ and the n positions P_j from the mn correspondences $\boldsymbol{p}_{ij}$.

$2mn$ equations in $8m+3n$ unknowns

Overconstrained problem, that can be solved using (non-linear) least squares!

The Affine Epipolar Constraint

$$\begin{cases} \boldsymbol{p} = \mathcal{A}\boldsymbol{P} + \boldsymbol{b} \\ \boldsymbol{p}' = \mathcal{A}'\boldsymbol{P} + \boldsymbol{b}' \end{cases}$$

$$\Rightarrow$$

$$\begin{pmatrix} \mathcal{A} & \boldsymbol{p} - \boldsymbol{b} \\ \mathcal{A}' & \boldsymbol{p}' - \boldsymbol{b}' \end{pmatrix} \begin{pmatrix} \boldsymbol{P} \\ -1 \end{pmatrix} = \boldsymbol{0}$$

$$\Downarrow$$

$$\mathrm{Det} \begin{pmatrix} \mathcal{A} & \boldsymbol{p} - \boldsymbol{b} \\ \mathcal{A}' & \boldsymbol{p}' - \boldsymbol{b}' \end{pmatrix} = 0$$

$$\Downarrow$$

$$\boxed{\alpha u + \beta v + \alpha' u' + \beta' v' + \delta = 0}$$

Note: the epipolar lines are parallel.

The Affine Fundamental Matrix

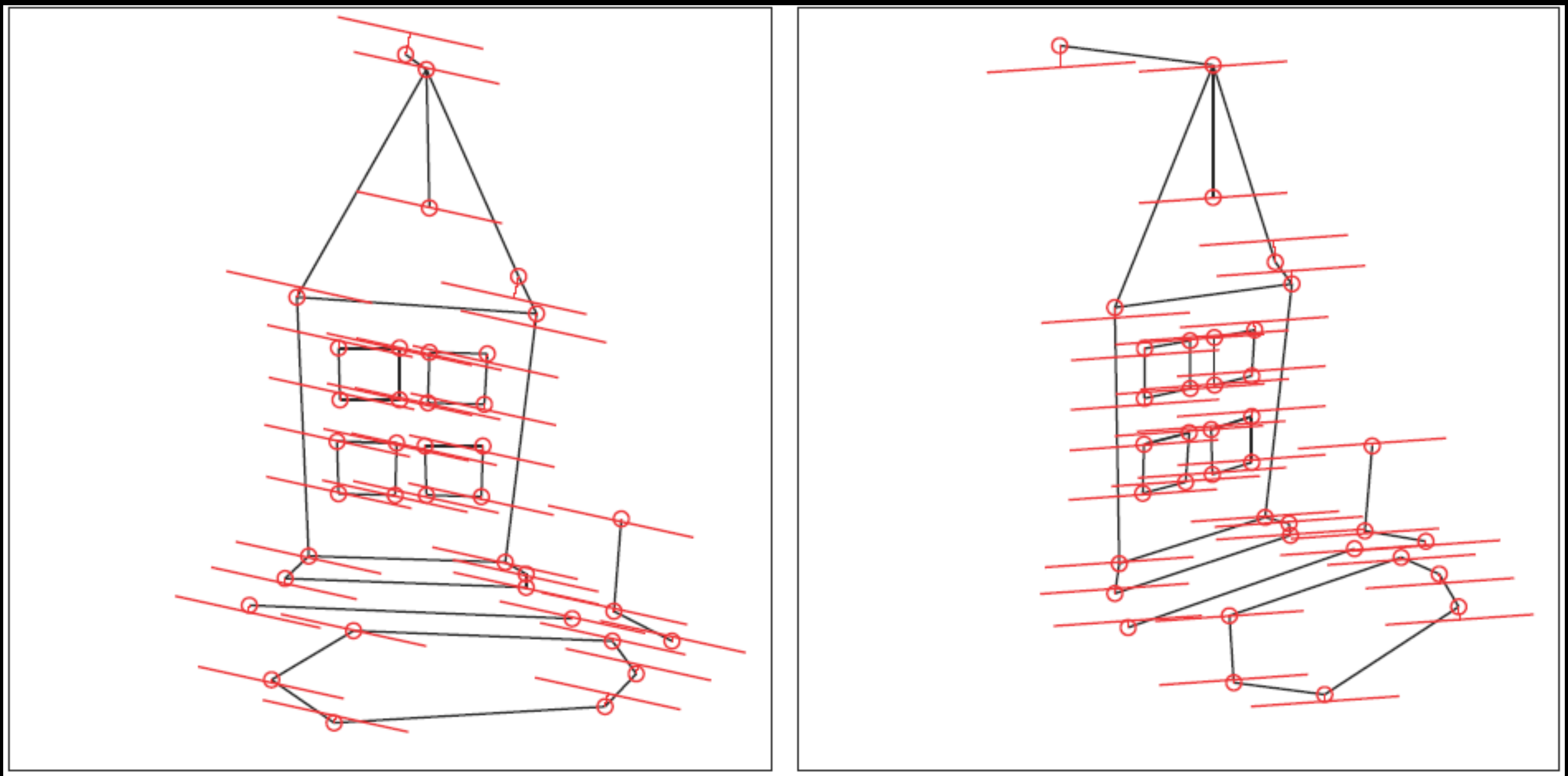
$$\alpha u + \beta v + \alpha' u' + \beta' v' + \delta = 0$$

$$(u, v, 1)\mathcal{F}\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0$$

where

$$\mathcal{F} \stackrel{\text{def}}{=} \begin{pmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \\ \alpha' & \beta' & \delta \end{pmatrix}$$

Affine case..



Mean errors: 3.24 and 3.15pixel (without normalization 160.92 and 158.54pixel).

The Affine Epipolar Constraint

$$\begin{cases} \boldsymbol{p} = \mathcal{A}\boldsymbol{P} + \boldsymbol{b} \\ \boldsymbol{p}' = \mathcal{A}'\boldsymbol{P} + \boldsymbol{b}' \end{cases}$$

$$\begin{pmatrix} \mathcal{A} & \boldsymbol{p} - \boldsymbol{b} \\ \mathcal{A}' & \boldsymbol{p}' - \boldsymbol{b}' \end{pmatrix} \begin{pmatrix} \boldsymbol{P} \\ -1 \end{pmatrix} = \boldsymbol{0}$$

$$\mathrm{Det} \begin{pmatrix} \mathcal{A} & \boldsymbol{p} - \boldsymbol{b} \\ \mathcal{A}' & \boldsymbol{p}' - \boldsymbol{b}' \end{pmatrix} = 0$$

$$\boxed{\alpha u + \beta v + \alpha' u' + \beta' v' + \delta = 0}$$

Note: the epipolar lines are parallel.

An Affine Trick.. → Algebraic Scene Reconstruction Method

$$\mathcal{M} = (\,\mathcal{A} \quad \boldsymbol{b}\,) \qquad \mathcal{M}' = (\,\mathcal{A}' \quad \boldsymbol{b}'\,) \qquad \boldsymbol{P}$$

$$\tilde{\mathcal{M}} = \mathcal{M}\mathcal{Q} \qquad \tilde{\mathcal{M}}' = \mathcal{M}'\mathcal{Q} \qquad \tilde{\boldsymbol{P}} = \mathcal{Q}^{-1}\boldsymbol{P}$$

$$\tilde{\mathcal{M}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad \tilde{\mathcal{M}}' = \begin{pmatrix} 0 & 0 & 1 & 0 \\ a & b & c & d \end{pmatrix} \qquad \tilde{\boldsymbol{P}}$$

$$\begin{pmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & u' \\ a & b & c & v'-d \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{P}} \\ -1 \end{pmatrix} = 0 \quad \Rightarrow \quad \boxed{\tilde{\boldsymbol{P}} = \begin{pmatrix} u \\ v \\ u' \end{pmatrix}}$$

Multiple affine images

Suppose we observe a static scene with m fixed cameras..

$$\boldsymbol{p}_i = \mathcal{M}_i \begin{pmatrix} \boldsymbol{P} \\ 1 \end{pmatrix} = \mathcal{A}_i \boldsymbol{P} + \boldsymbol{b}_i \quad \text{for} \quad i = 1, \ldots, m$$

$$\boldsymbol{q} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{p}_1 \\ \ldots \\ \boldsymbol{p}_m \end{pmatrix}, \quad \boldsymbol{r} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{b}_1 \\ \ldots \\ \boldsymbol{b}_m \end{pmatrix} \quad \text{and} \quad \mathcal{A} \stackrel{\text{def}}{=} \begin{pmatrix} \mathcal{A}_1 \\ \ldots \\ \mathcal{A}_m \end{pmatrix}$$

$$\mathcal{D} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{q}_1 & \ldots & \boldsymbol{q}_n \end{pmatrix} = \mathcal{A}\mathcal{P} + \boldsymbol{r} \text{ with } \mathcal{P} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{P}_1 & \ldots & \boldsymbol{P}_n \end{pmatrix}$$

Multiple affine images

Idea: pick one of the points (or their center of mass) as the origin.

$$\boldsymbol{P} \longrightarrow \boldsymbol{P} - \boldsymbol{P}_0$$

$$\boldsymbol{p} \longrightarrow \boldsymbol{p} - \boldsymbol{p}_0$$

$$\boldsymbol{p}_i = \mathcal{A}_i \boldsymbol{P} + \boldsymbol{b}_i \longrightarrow \boldsymbol{p}_i = \mathcal{A}_i \boldsymbol{P}$$

$$\boxed{\mathcal{D} \stackrel{\text{def}}{=} (\boldsymbol{q}_1 \quad \dots \quad \boldsymbol{q}_n) = \mathcal{A}\mathcal{P}, \quad \text{with} \quad \mathcal{P} \stackrel{\text{def}}{=} (\boldsymbol{P}_1 \quad \dots \quad \boldsymbol{P}_n)}$$

What if we could factorize D? (Tomasi and Kanade, 1992)

$\mathcal{A}, \mathcal{P} \rightarrow \mathcal{D}$

$\mathcal{D} \rightarrow \mathcal{A}, \mathcal{P}$

Affine SFM is solved!

$$E \stackrel{\text{def}}{=} \sum_{i,j} |\boldsymbol{p}_{ij} - \mathcal{A}_i \boldsymbol{P}_j|^2 = \sum_j |\boldsymbol{q}_j - \mathcal{A}\boldsymbol{P}_j|^2 = |\mathcal{D} - \mathcal{A}\mathcal{P}|^2$$

Singular Value Decomposition

Theorem: When $\mathcal{A}$ has a rank greater than p, $\mathcal{U}_p \mathcal{W}_p \mathcal{V}_p^T$ is the best possible rank-p approximation of $\mathcal{A}$ in the sense of the Frobenius norm.

We can take

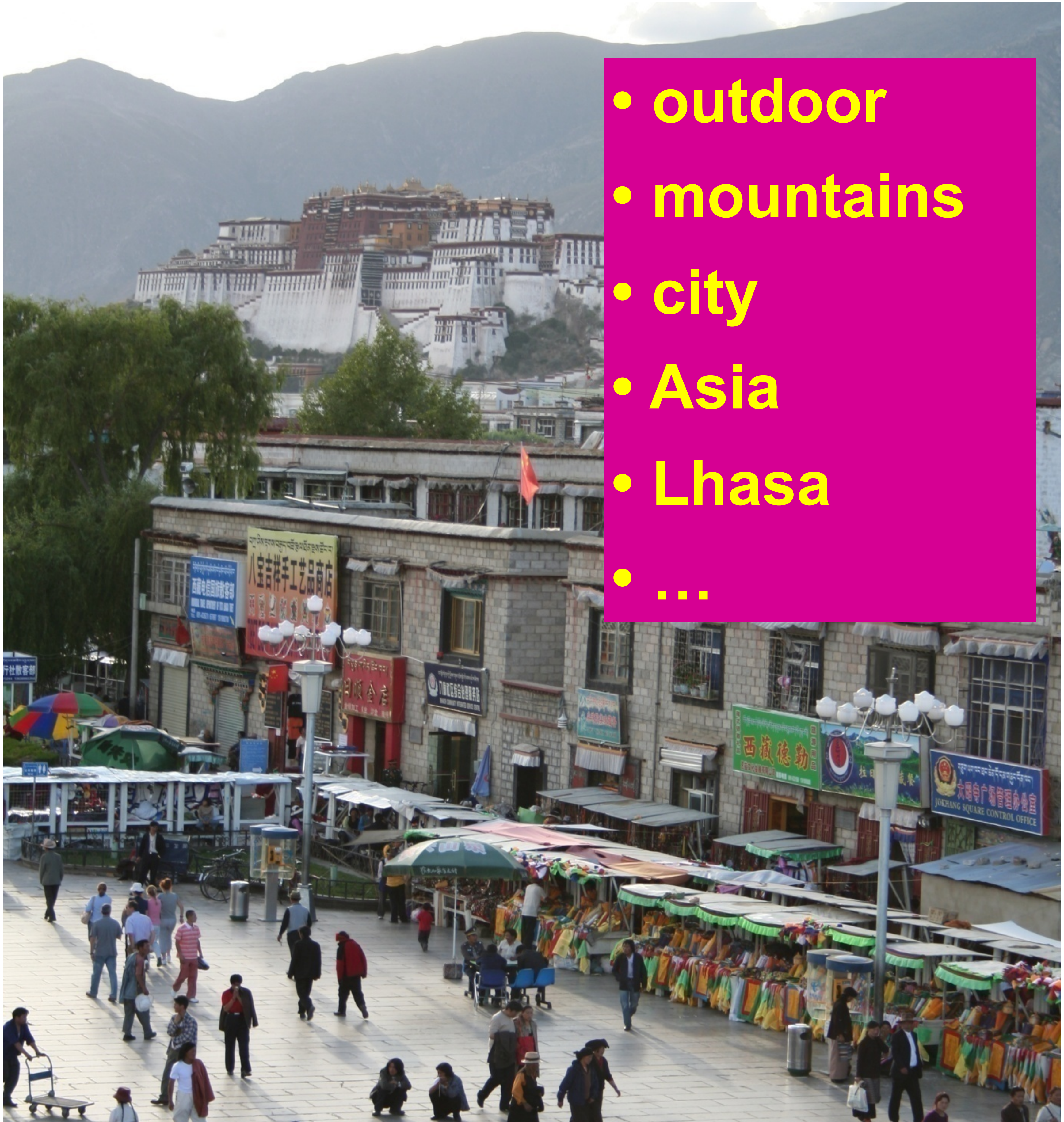
$$\begin{cases} \mathcal{A}_0 = \mathcal{U}_3 \\ \mathcal{P}_0 = \mathcal{W}_3 \mathcal{V}_3^T \end{cases}$$

Recognition, classical approaches, and ML

Common recognition tasks

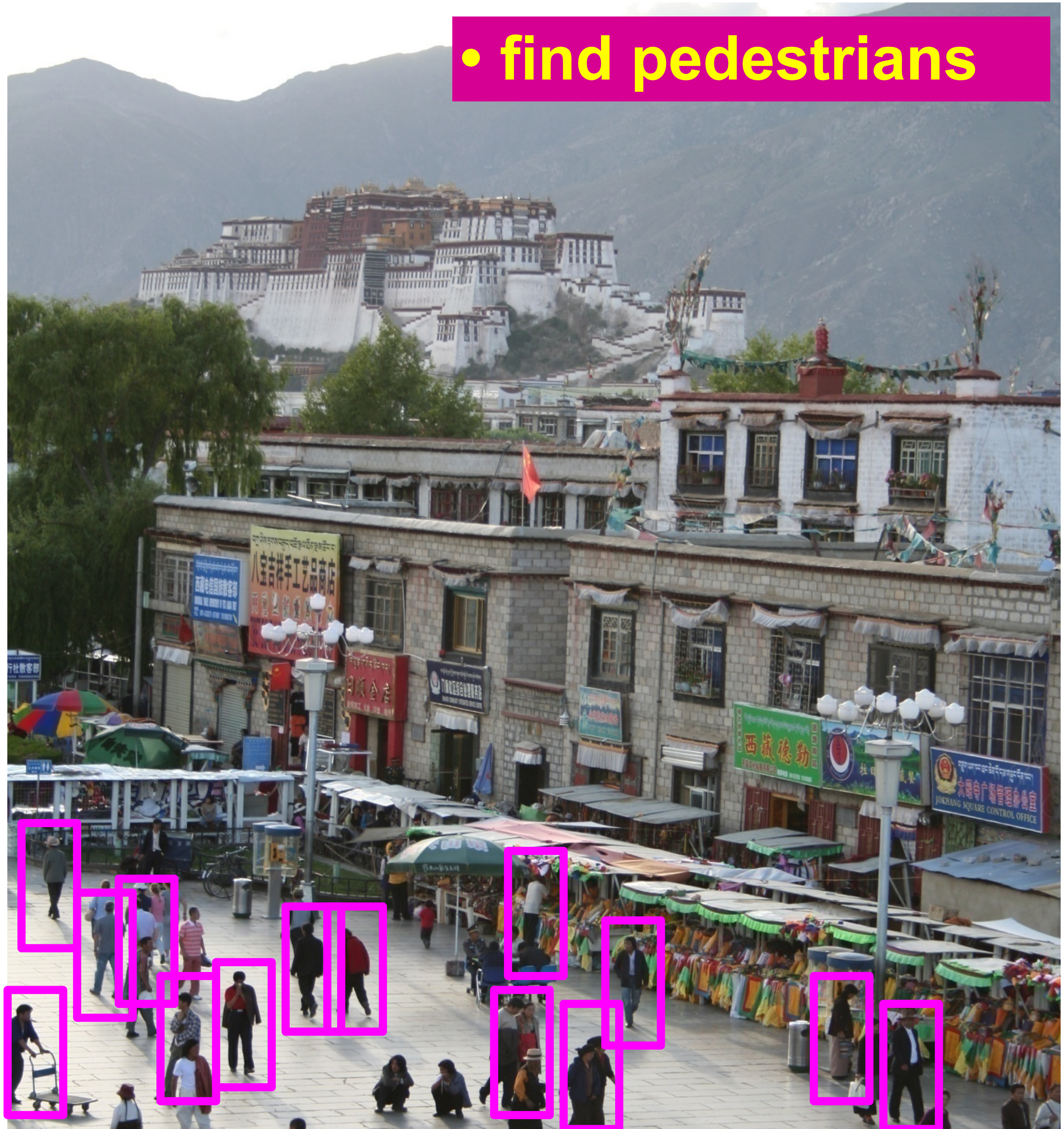
Adapted from
Fei-Fei Li

Image classification and tagging



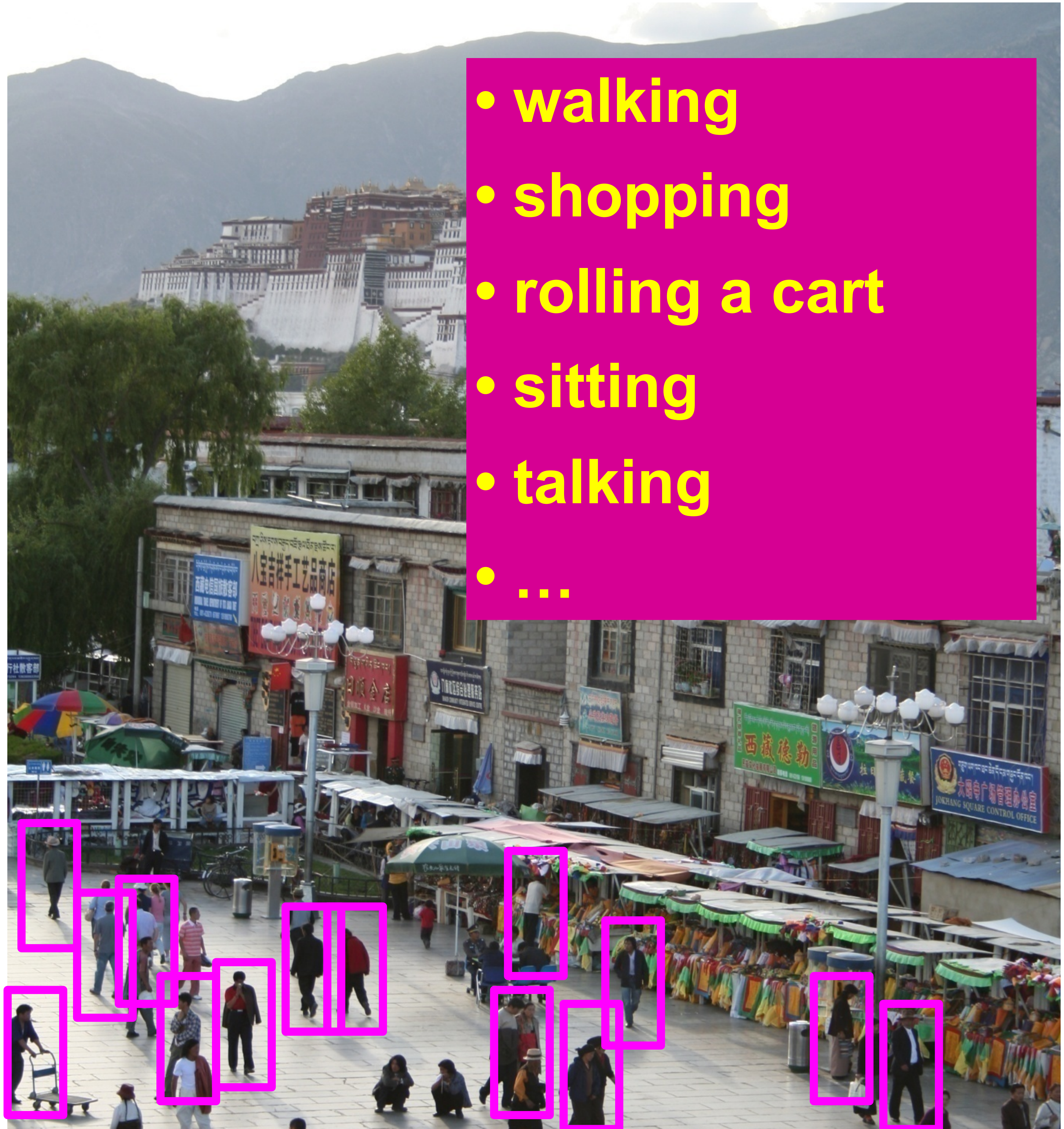
Adapted from Fei-Fei Li

Object detection



Adapted from
Fei-Fei Li

Activity recognition



Adapted from
Fei-Fei Li

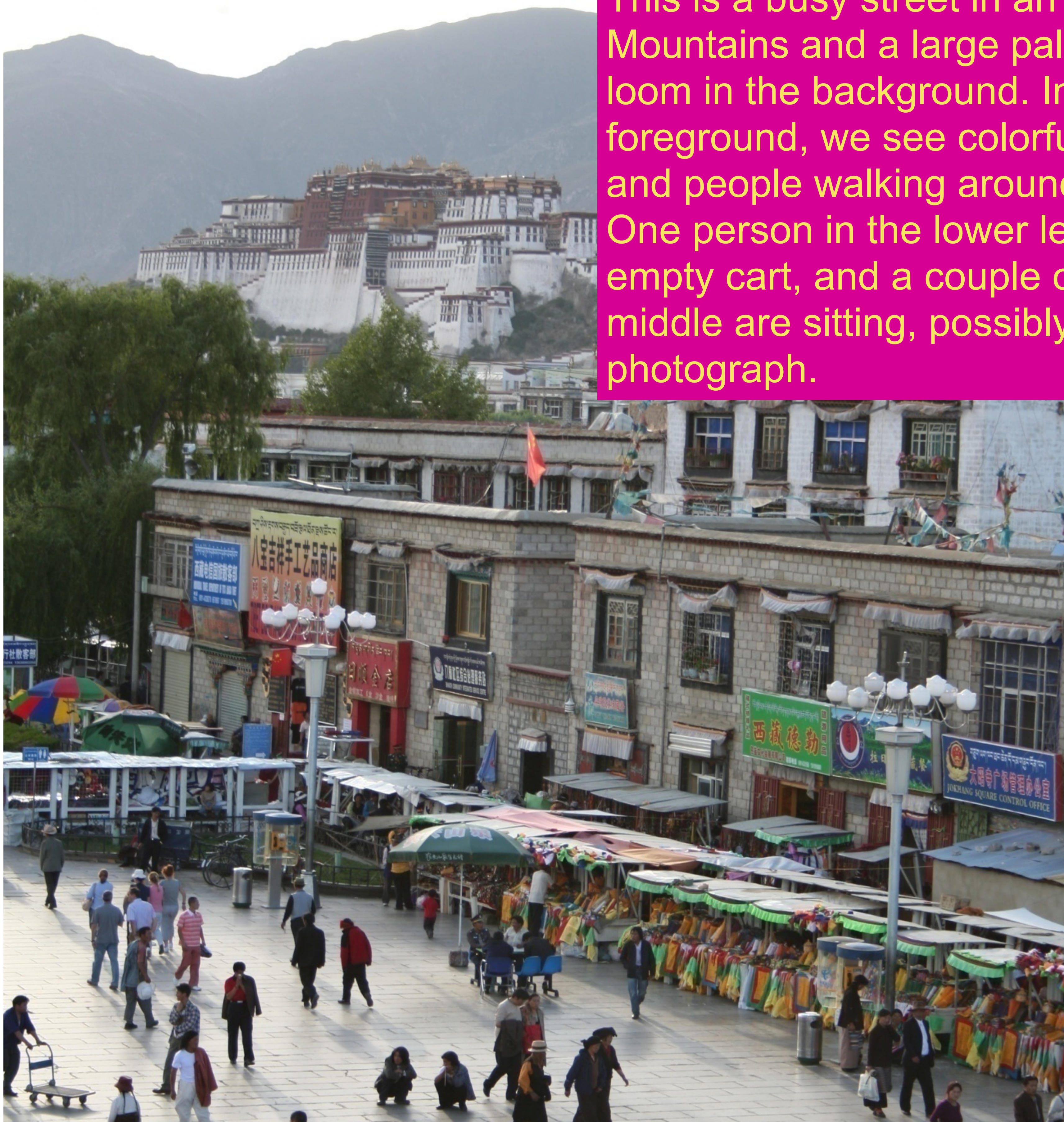
Semantic segmentation

Adapted from
Fei-Fei Li

Semantic segmentation

Adapted from
Fei-Fei Li

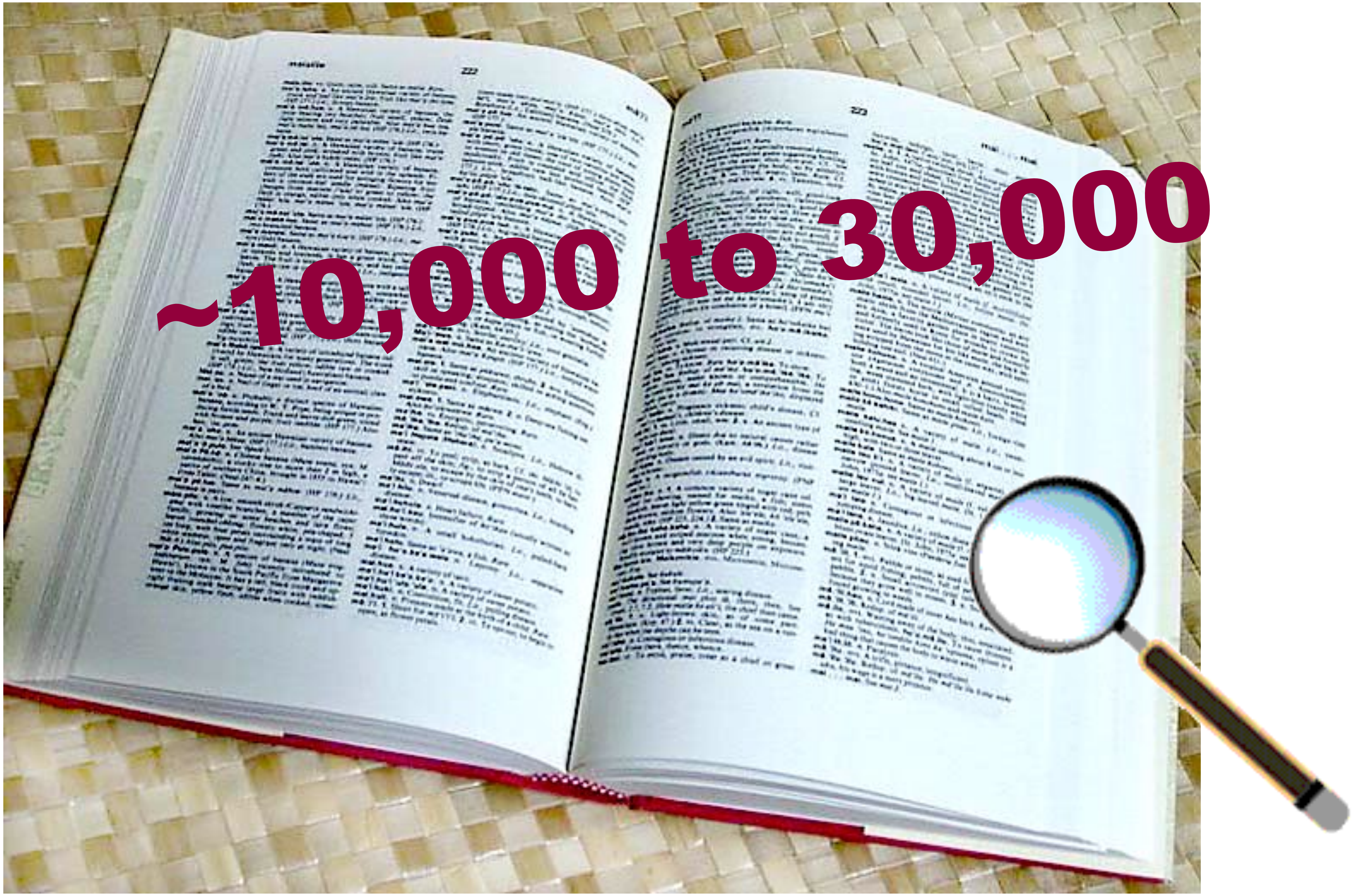
Image description



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

How many visual object categories are there?



Source: J. Hays

Biederman 1987


~10,000 to 30,000
Source: J. Hays

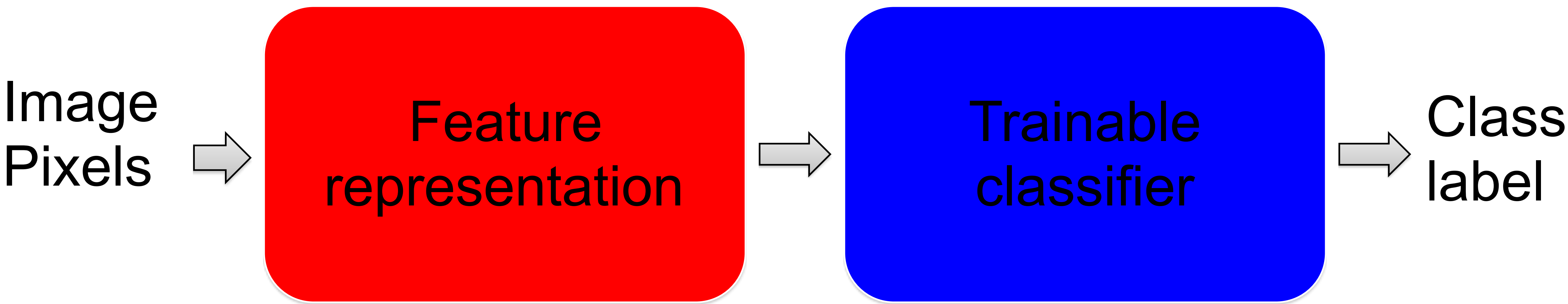
Within-class variations






Svetlana Lazebnik

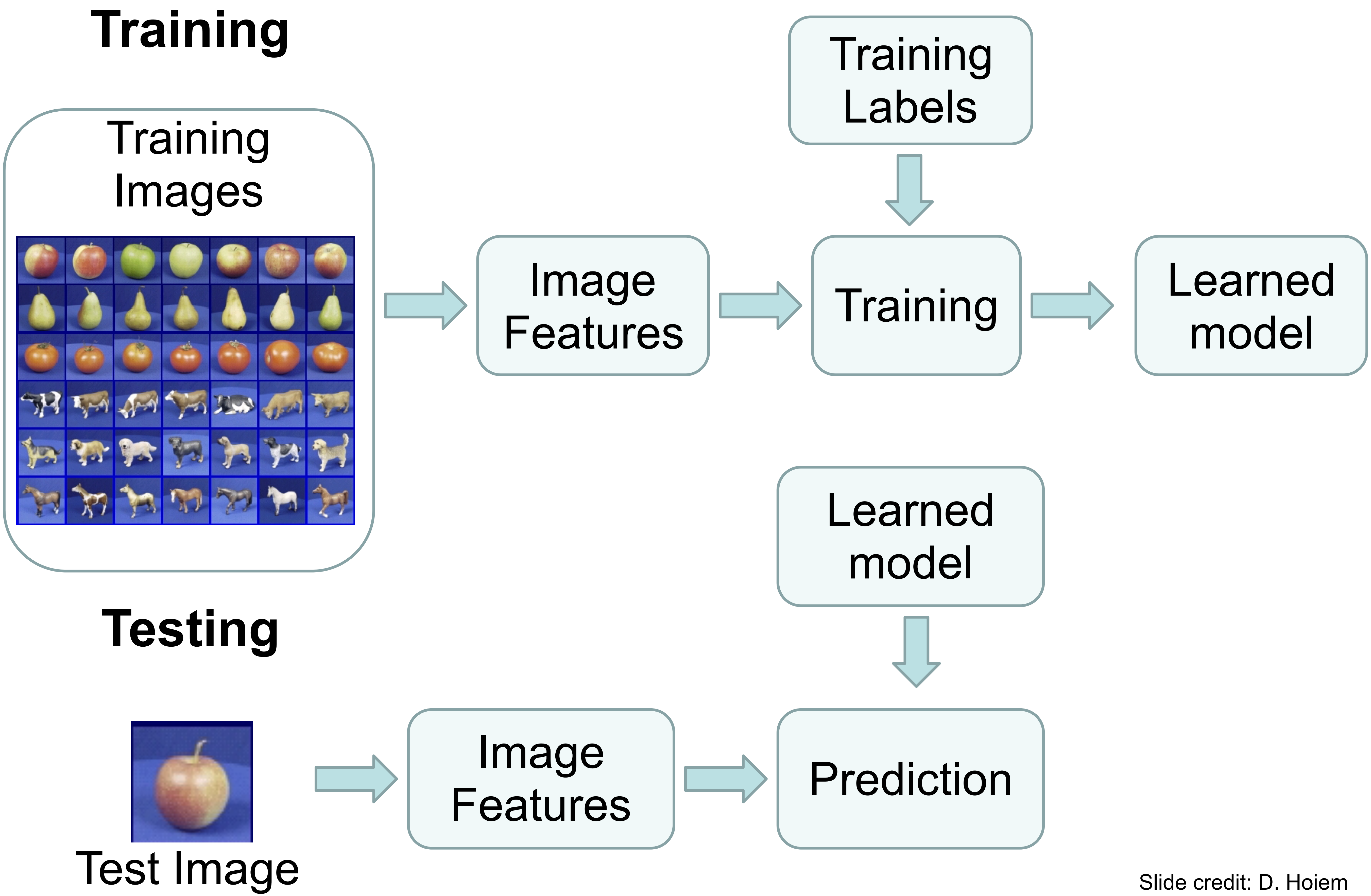
“Classic” recognition pipeline



- Hand-crafted feature representation
- Off-the-shelf trainable classifier

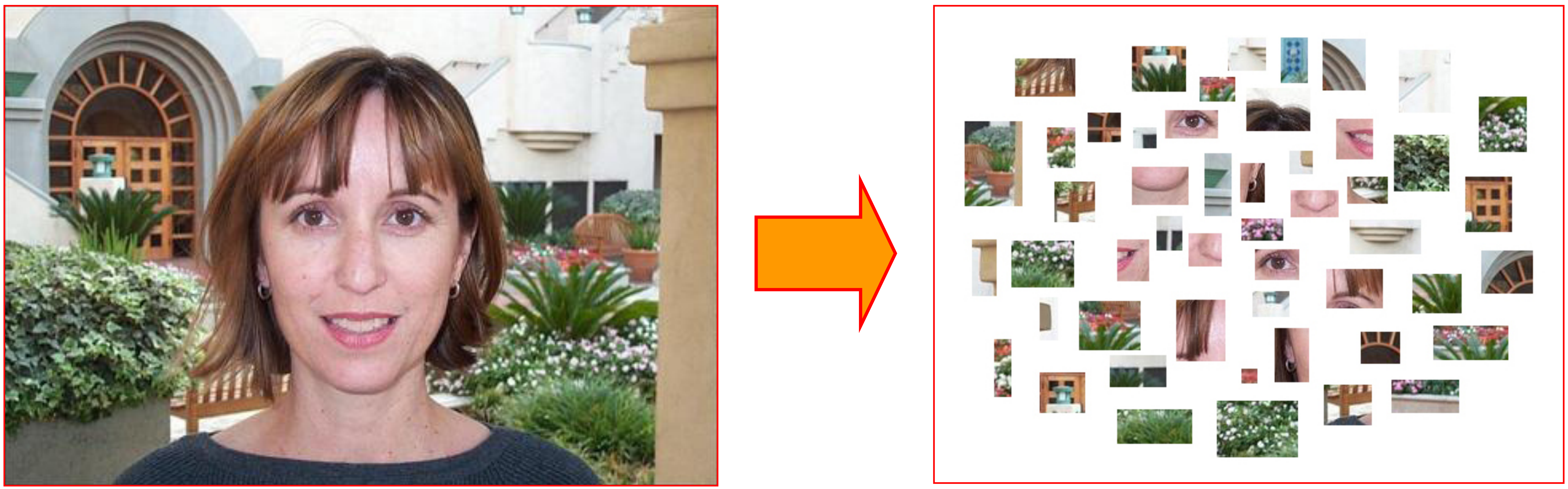
Source: S. Lazebnik

Steps



Slide credit: D. Hoiem

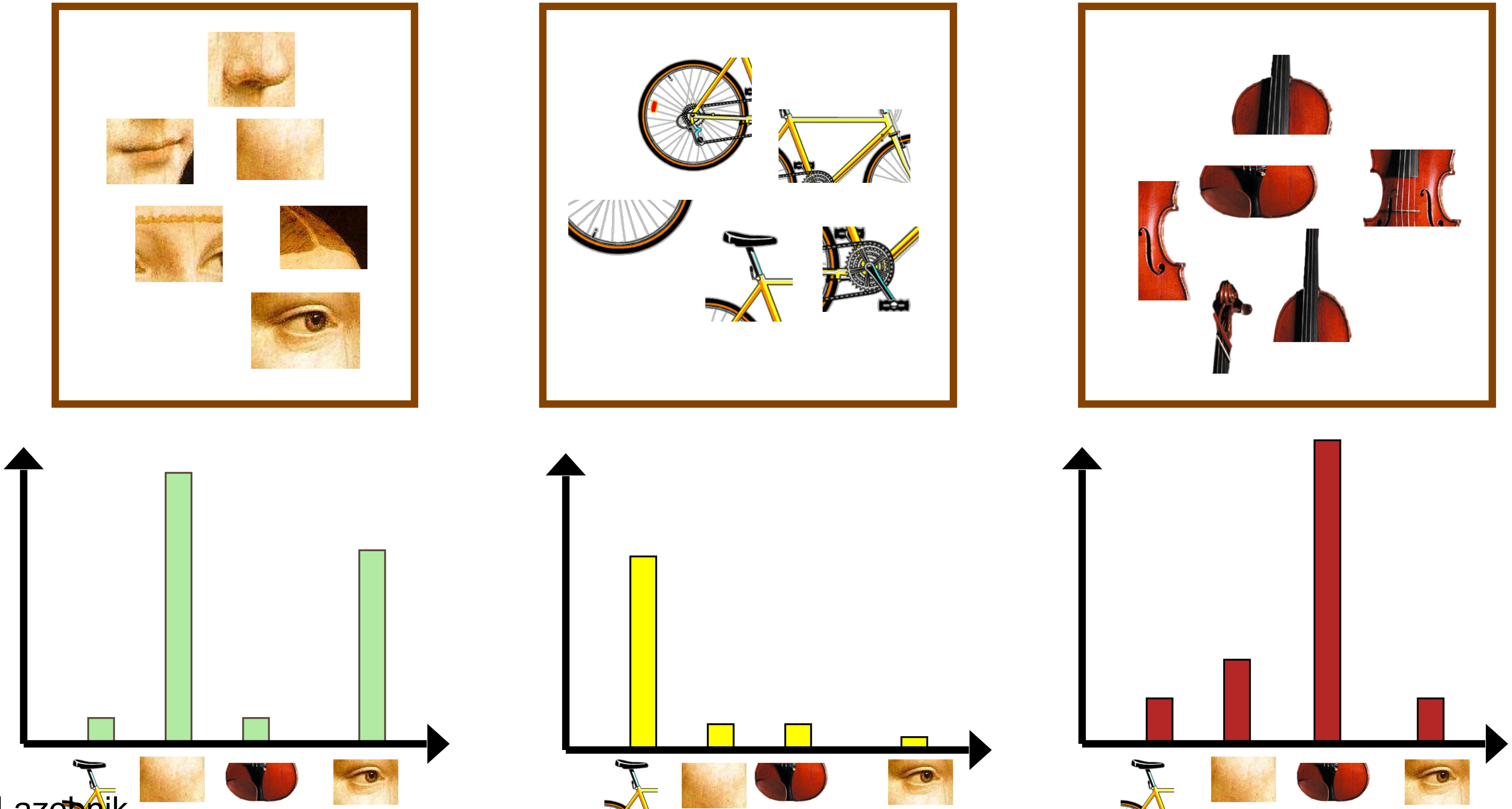
“Classic” representation: Bag of features



Source: S. Lazebnik

Bag of features: Outline

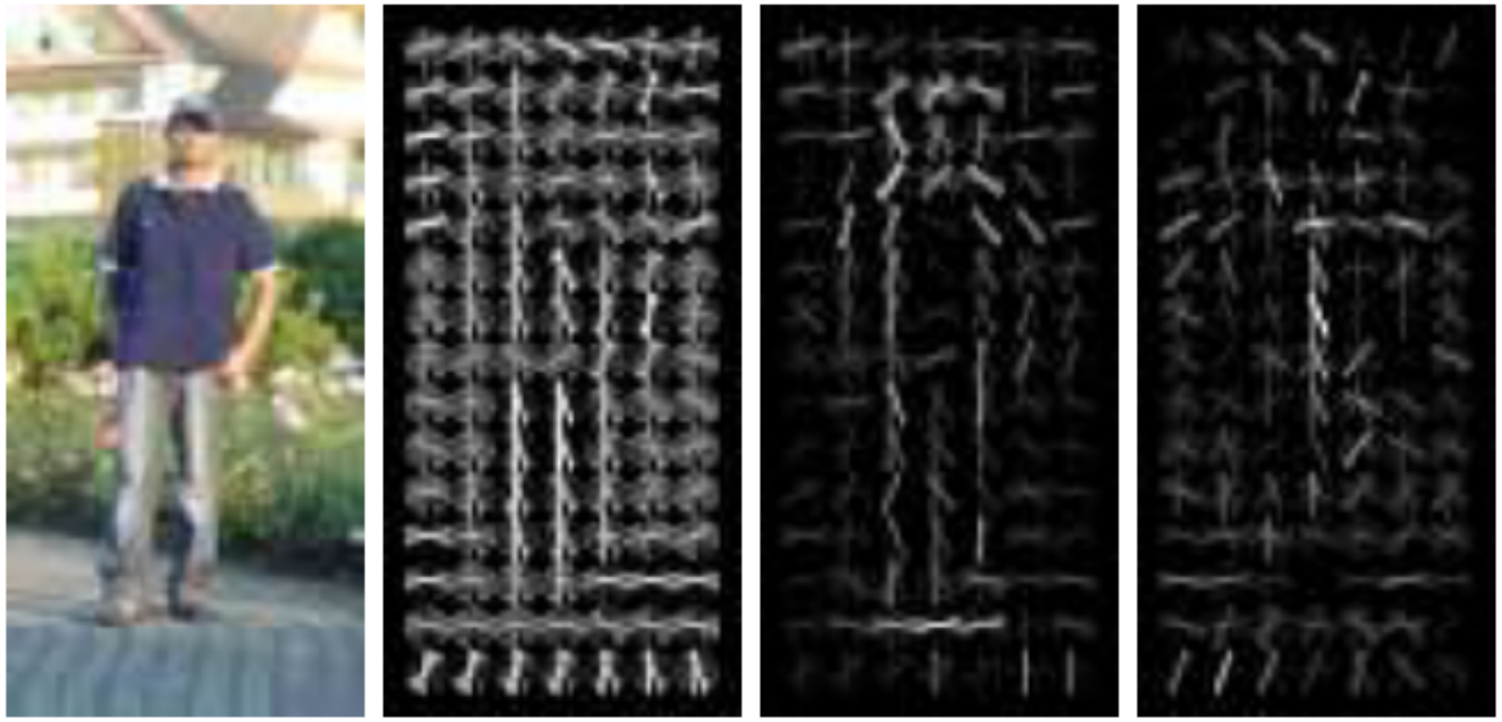
1. Extract local features
2. Learn “visual vocabulary”
3. Quantize local features using visual vocabulary
4. Represent images by frequencies of “visual words”



Source: S. Lazebnik

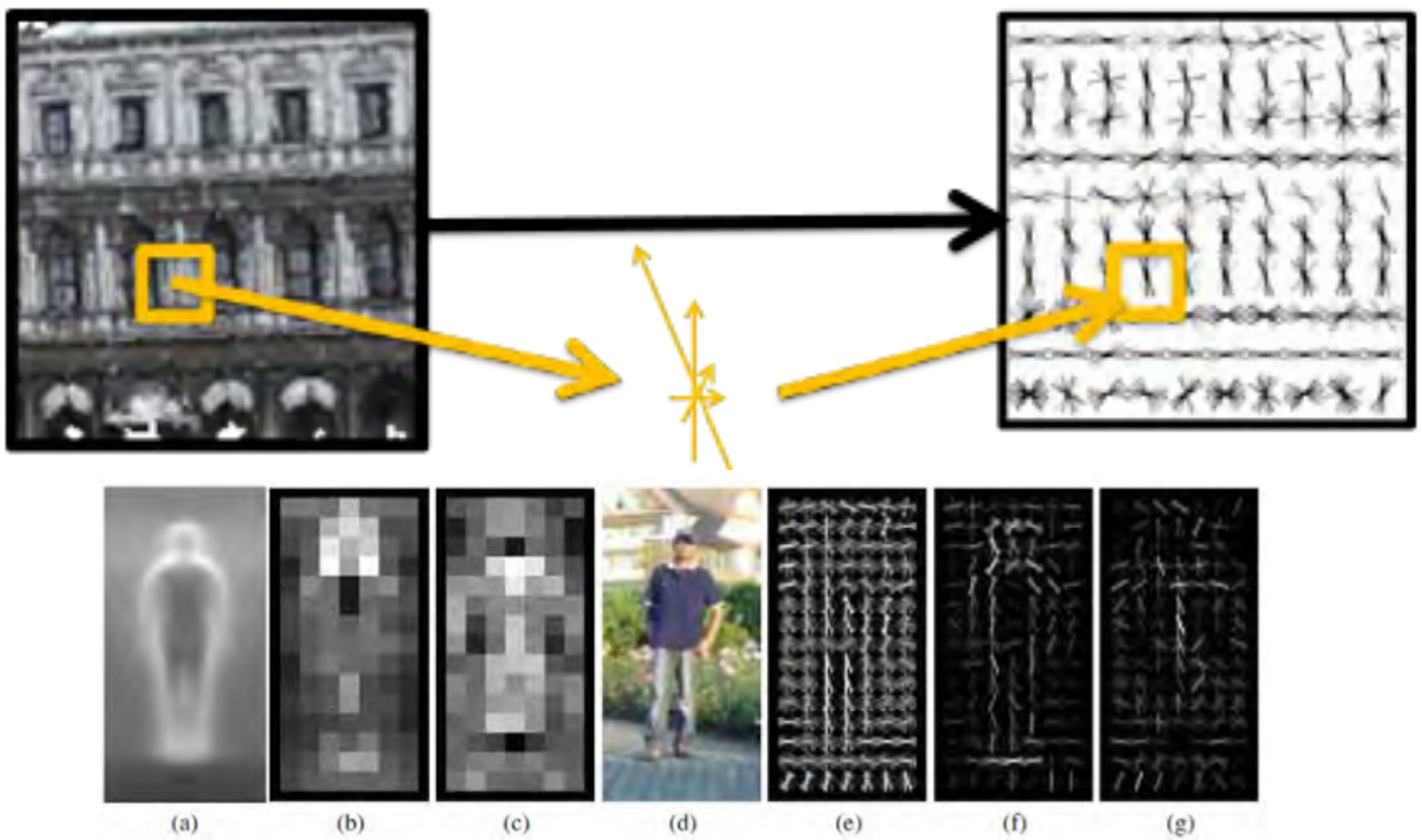
Contour based classification

- HOG+SVM+sliding window, Dalal and Trigs [2005]



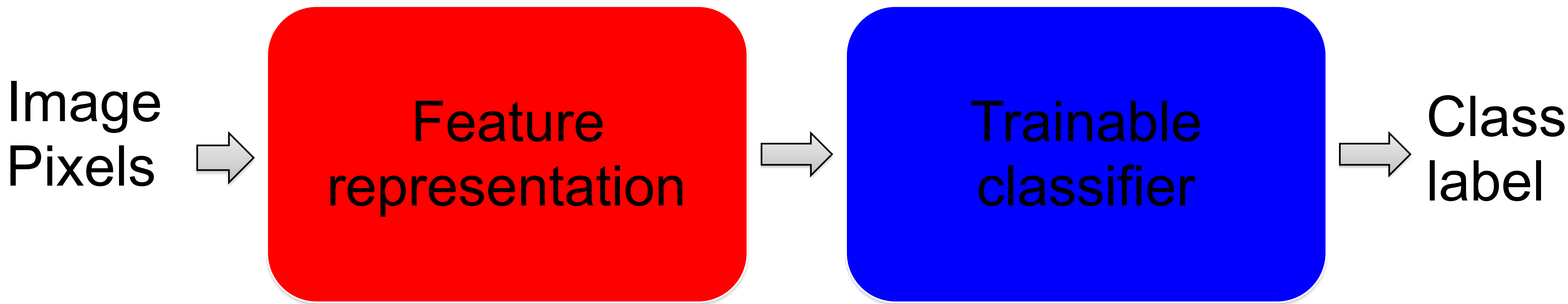
Histograms of Oriented Gradients

- Use Histograms



Dalal, N., & Triggs, B. Histograms of oriented gradients for human detection. *CVPR 2005*.

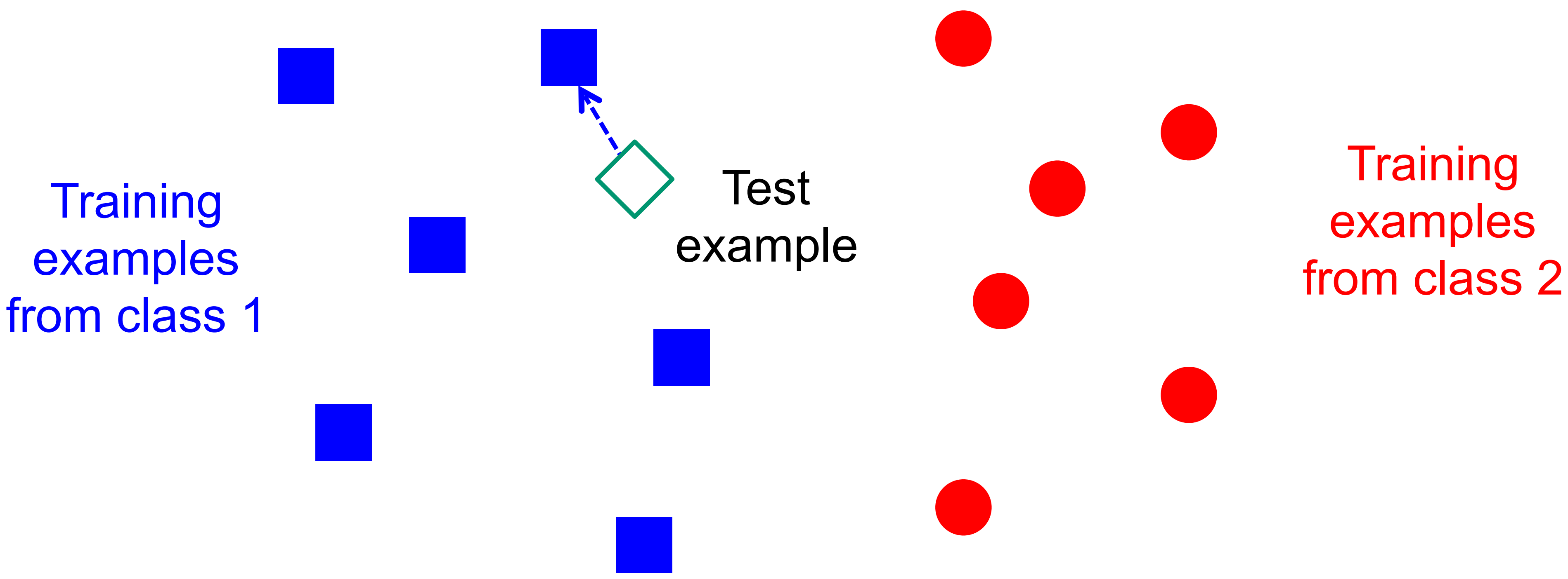
“Classic” recognition pipeline



- Hand-crafted feature representation
- Off-the-shelf trainable classifier

Source: S. Lazebnik

Non-parametric learning: nearest neighbor



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

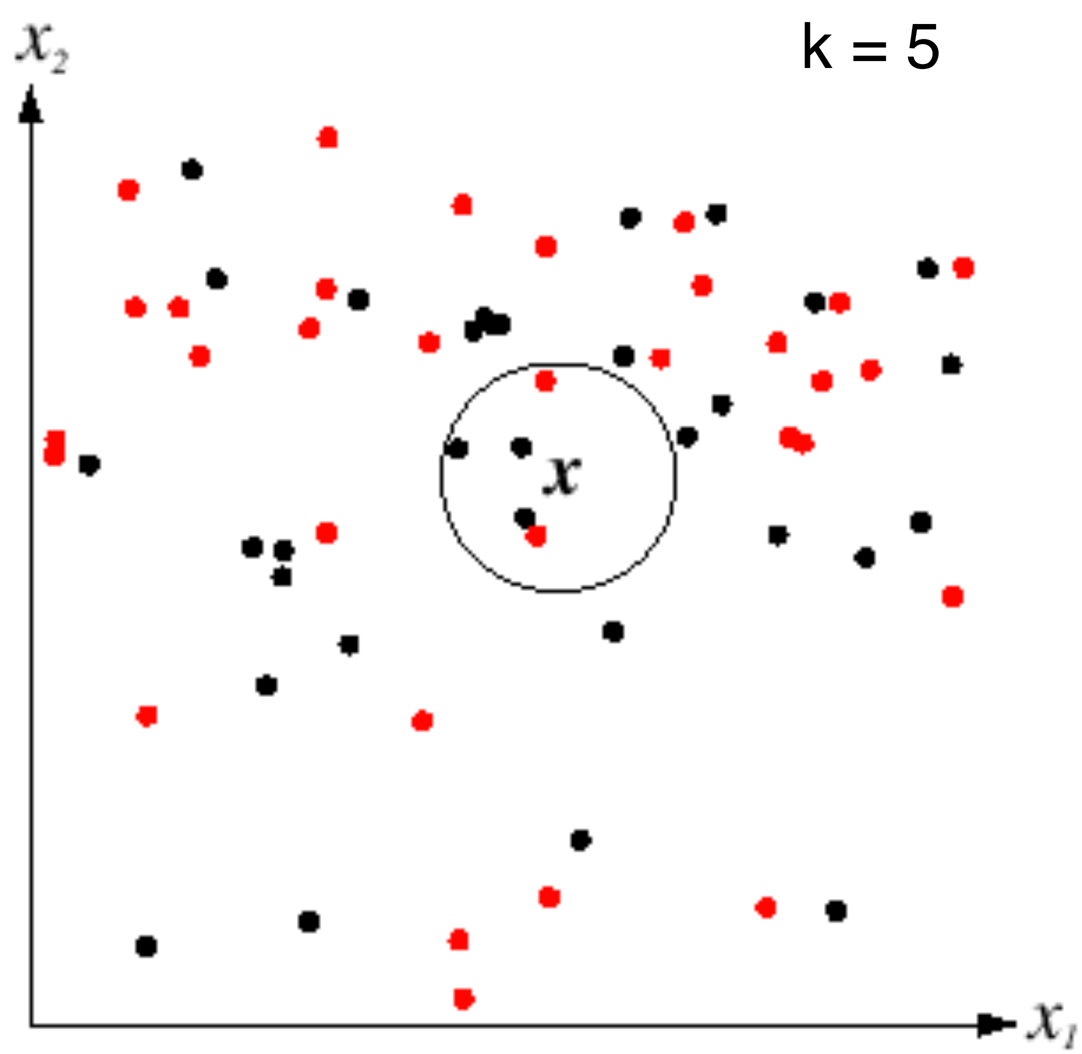
All we need is a distance or similarity function for our inputs

No training required!

Source: S. Lazebnik

K-nearest neighbor classifier

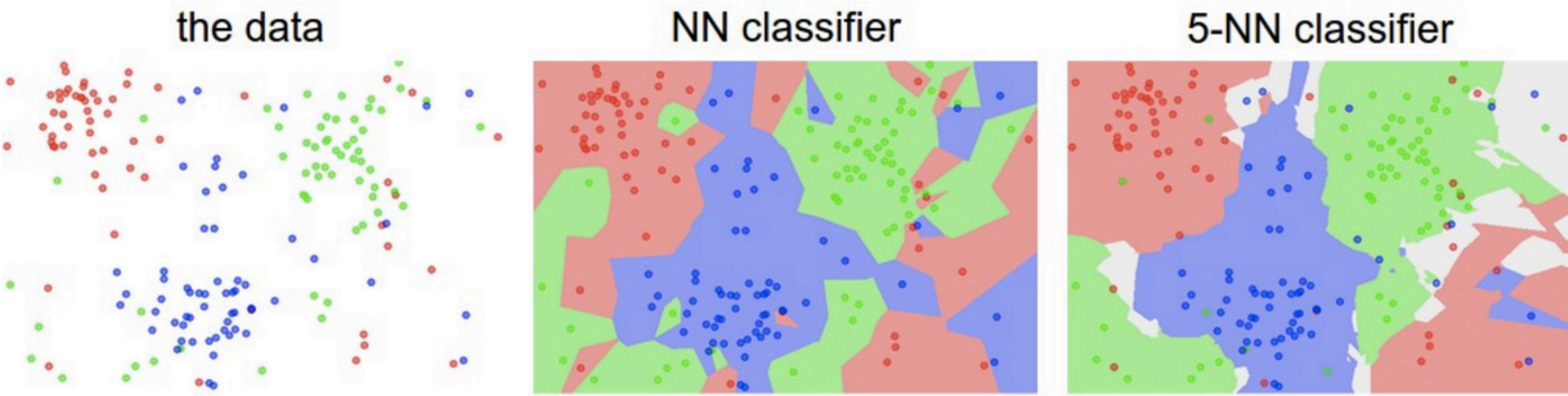
- For a new point, find the k closest points from training data
- Vote for class label with labels of the k points



Source: S. Lazebnik

K-nearest neighbor classifier

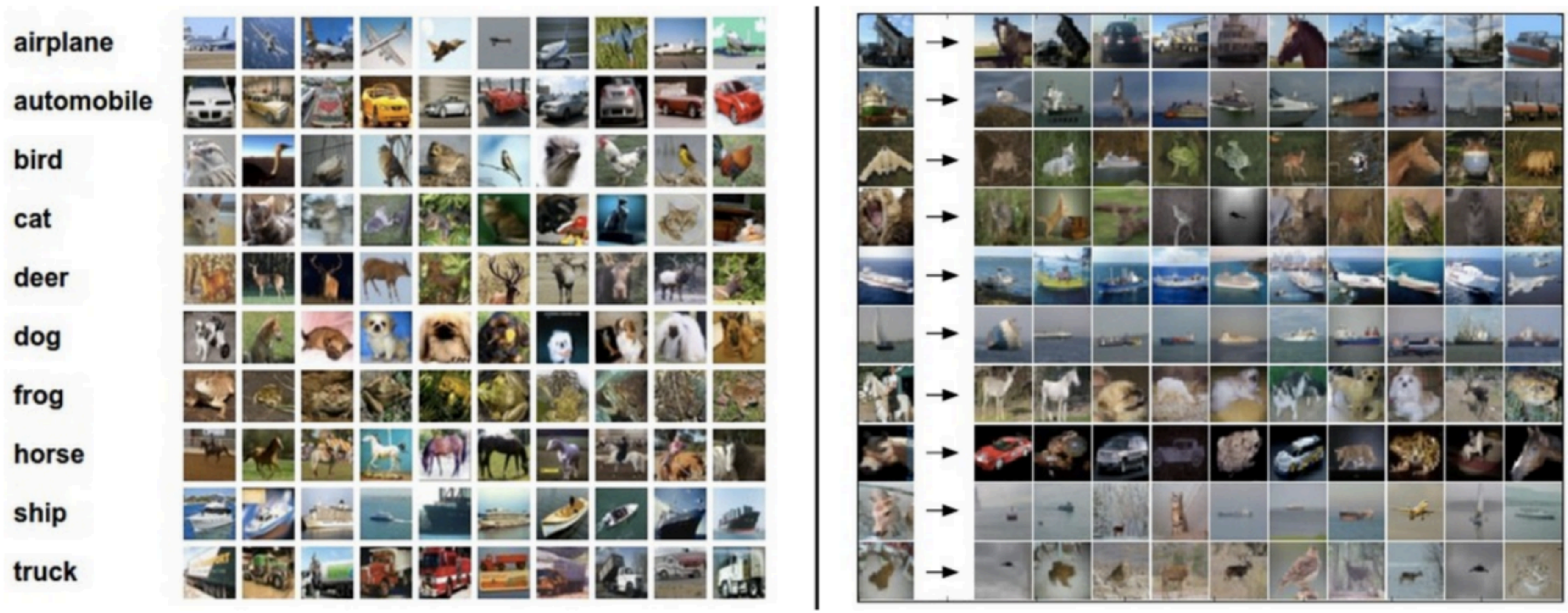
Which classifier is more robust to *outliers*?



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

Source: S. Lazebnik

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/

Source: S. Lazebnik

Parametric supervised learning

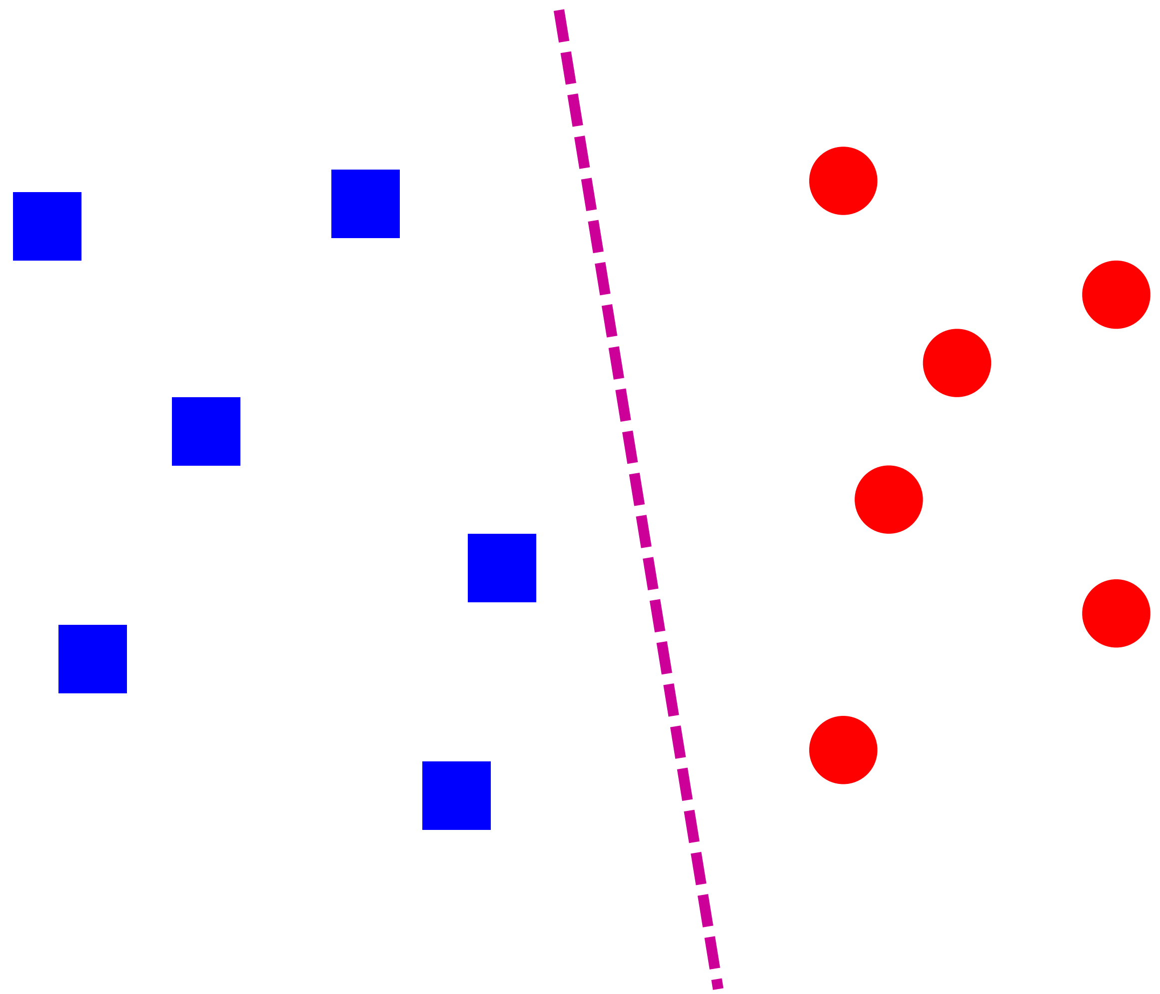
- Data $\{(x_1,y_1),\ldots(x_n,y_n)\}$
- A function class $\mathcal{F} = \{f_\theta \,|\, \theta \in \mathbb{R}^d\}$
- A loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- Minimize the empirical risk

$$\hat{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)$$

- Hope that this “generalizes”: if (X, Y) is a r.v., we would like to minimize

$$L(\theta) = \mathbb{E}[\ell(f_\theta(X), Y)]$$

Linear classifiers



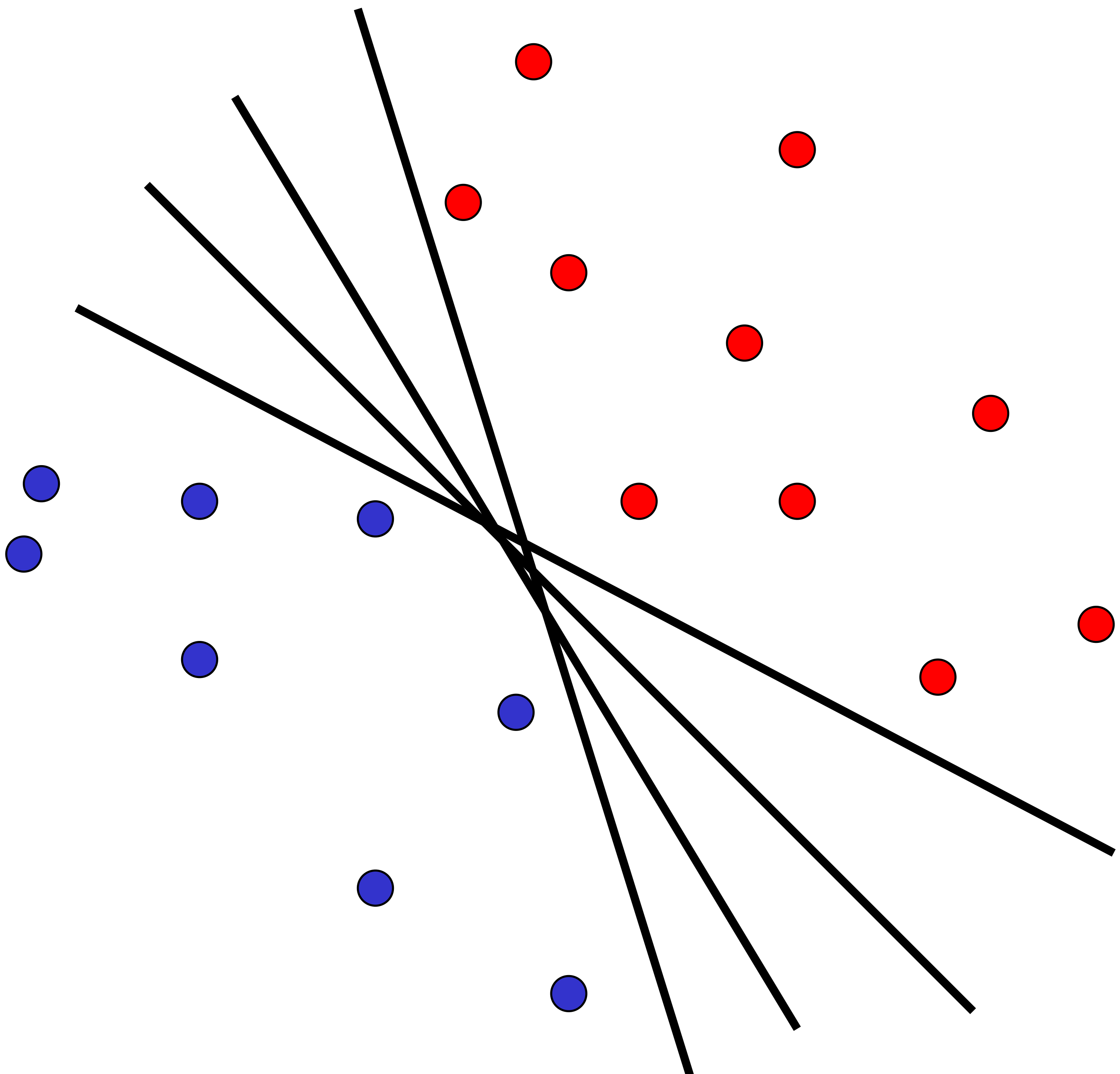
Find a *linear function* to separate the classes:

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

Source: S. Lazebnik

Linear classifiers

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

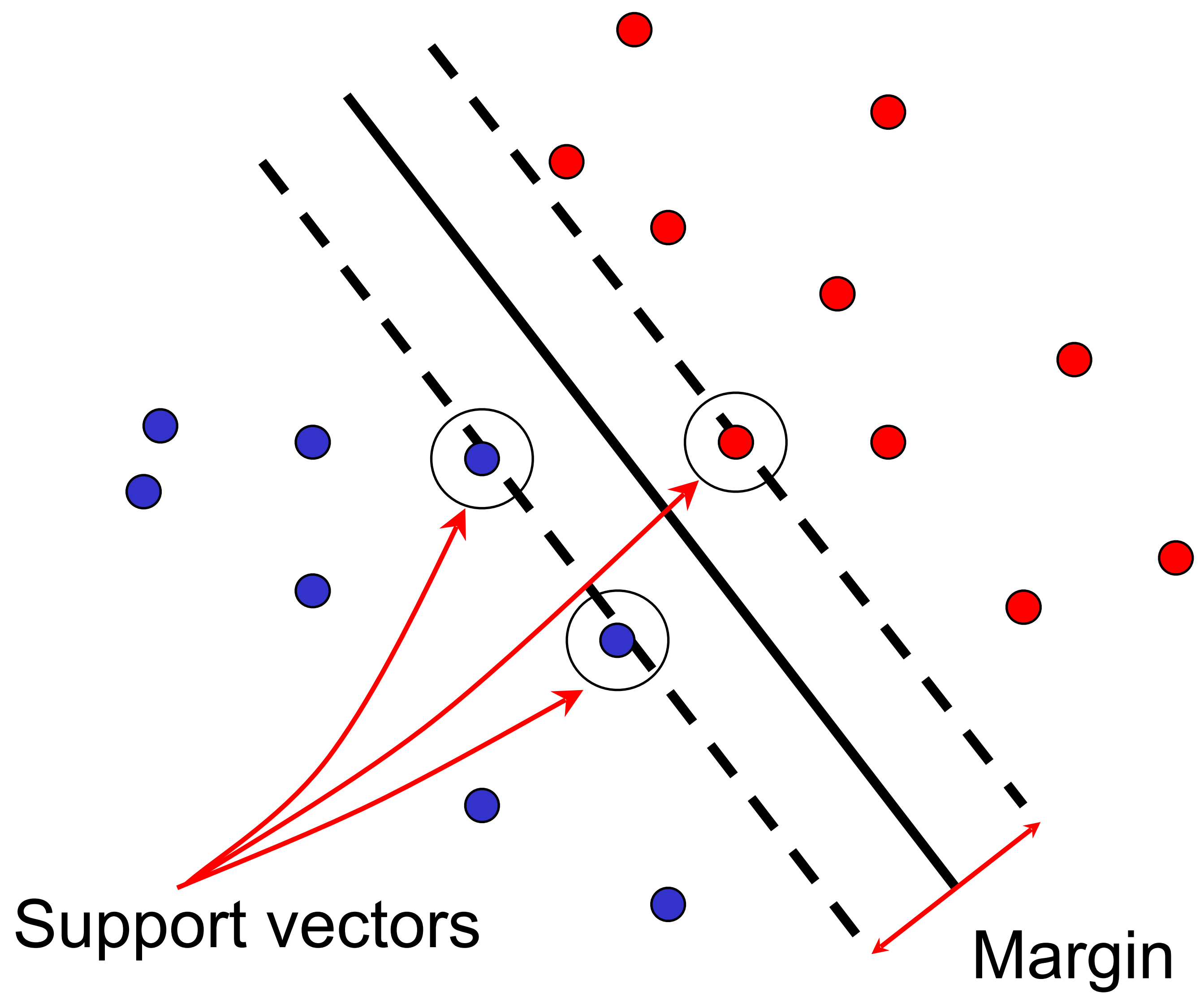


Which separator is best?

Source: S. Lazebnik

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

Source: S. Lazebnik

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:

$$\boxed{\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1}$$

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

Source: S. Lazebnik

SVM parameter learning

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

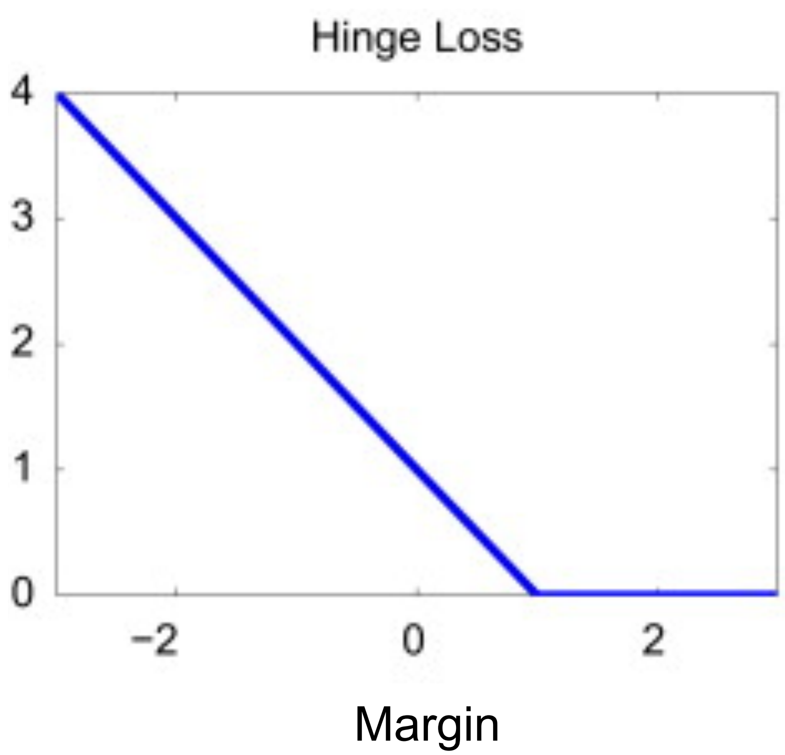
- Non-separable data:

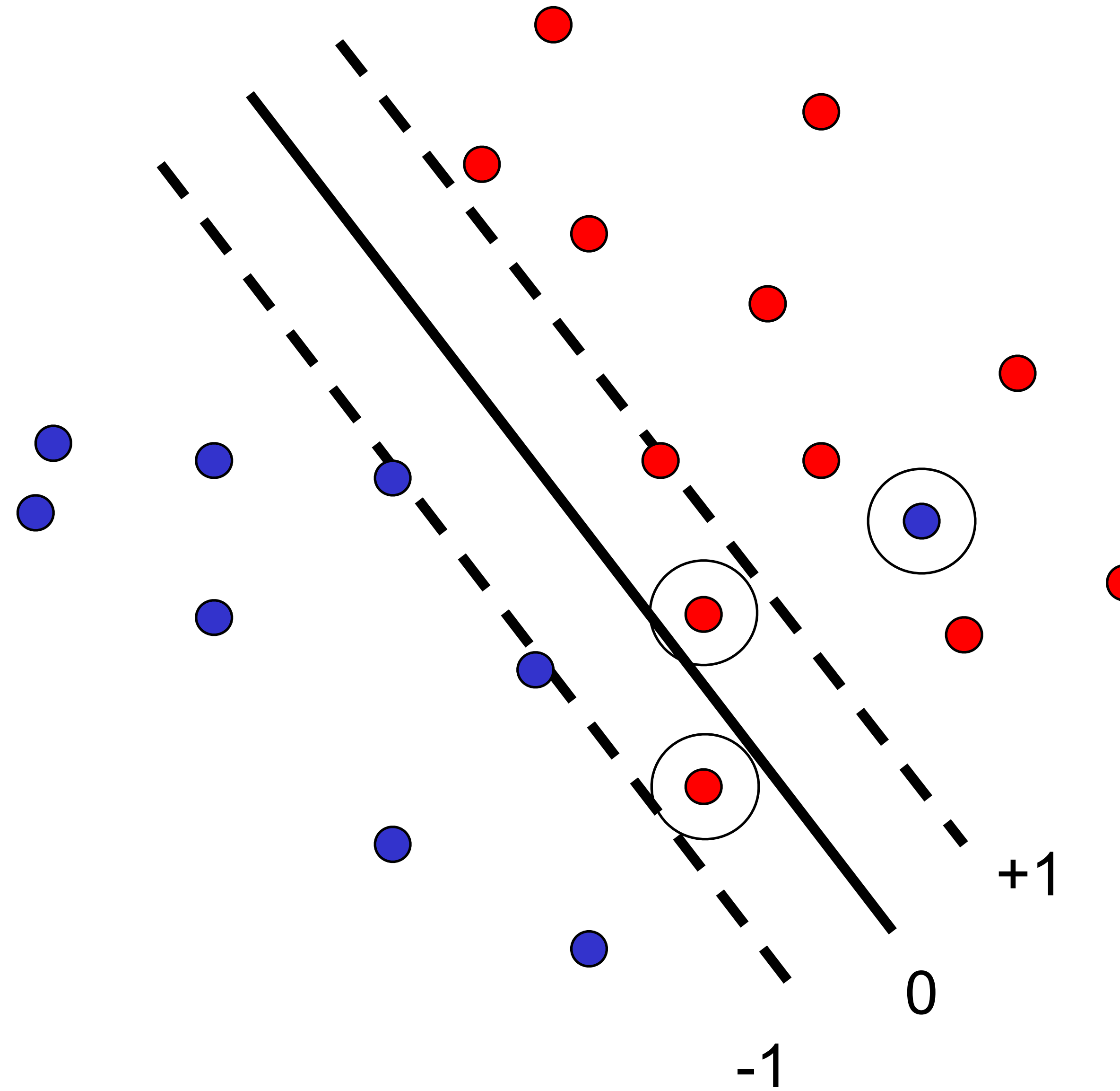
$$\min_{\mathbf{w},b} \quad \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{Maximize margin}} + \underbrace{C\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\right)}_{\text{Minimize classification mistakes}}$$

Source: S. Lazebnik

SVM parameter learning

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





Source: S. Lazebnik

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 set is separable

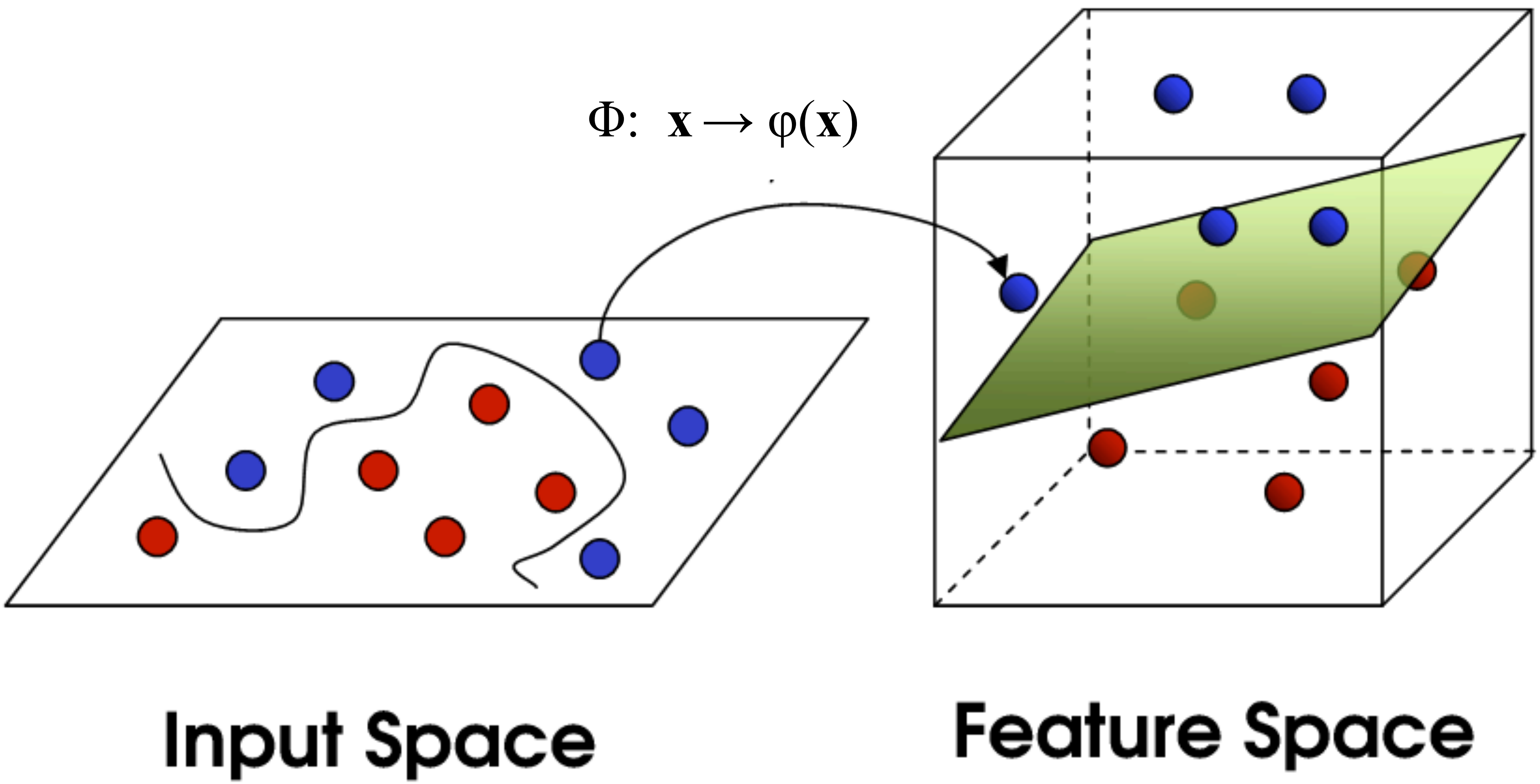
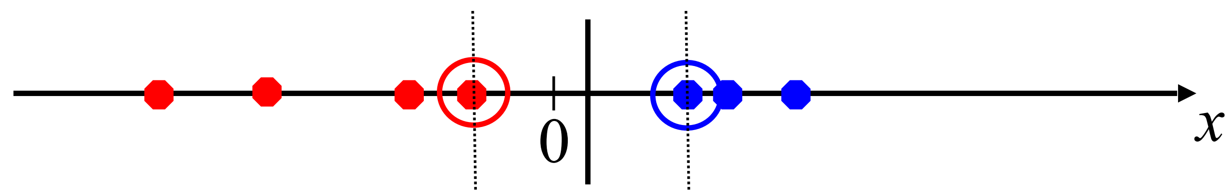


Image source

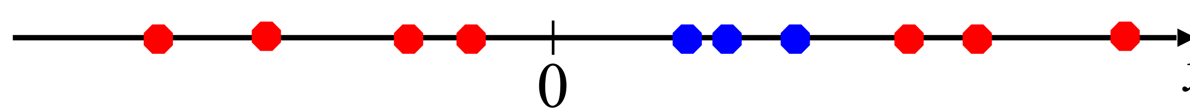
Source: S. Lazebnik

Nonlinear SVMs

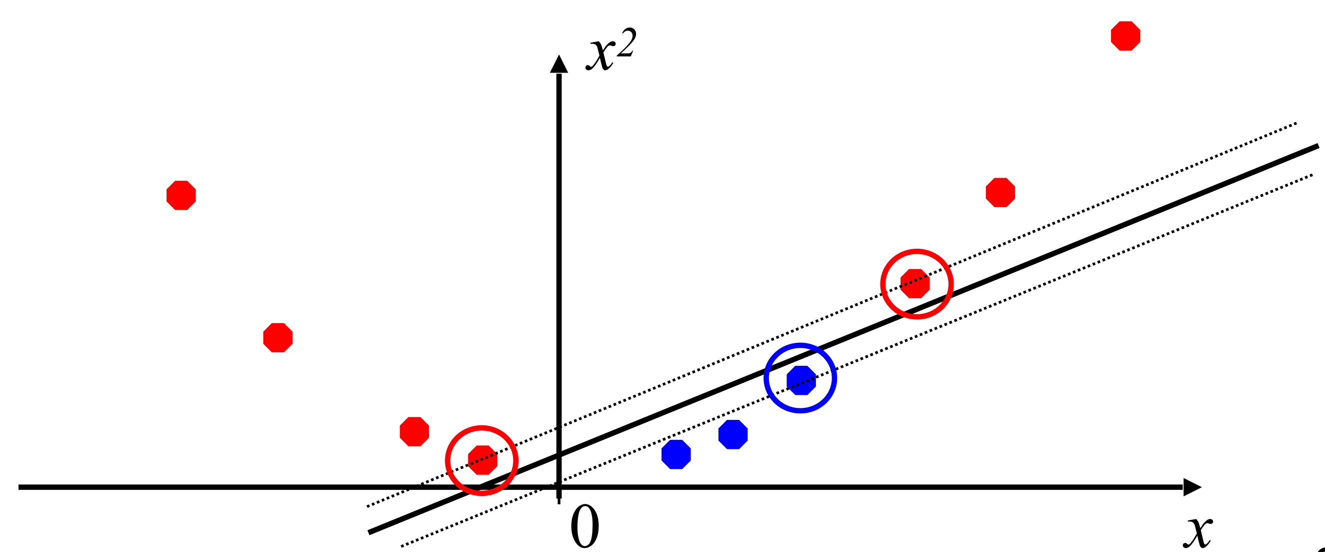
- Linearly separable dataset in 1D:



- 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*)

Source: S. Lazebnik

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

Source: S. Lazebnik

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

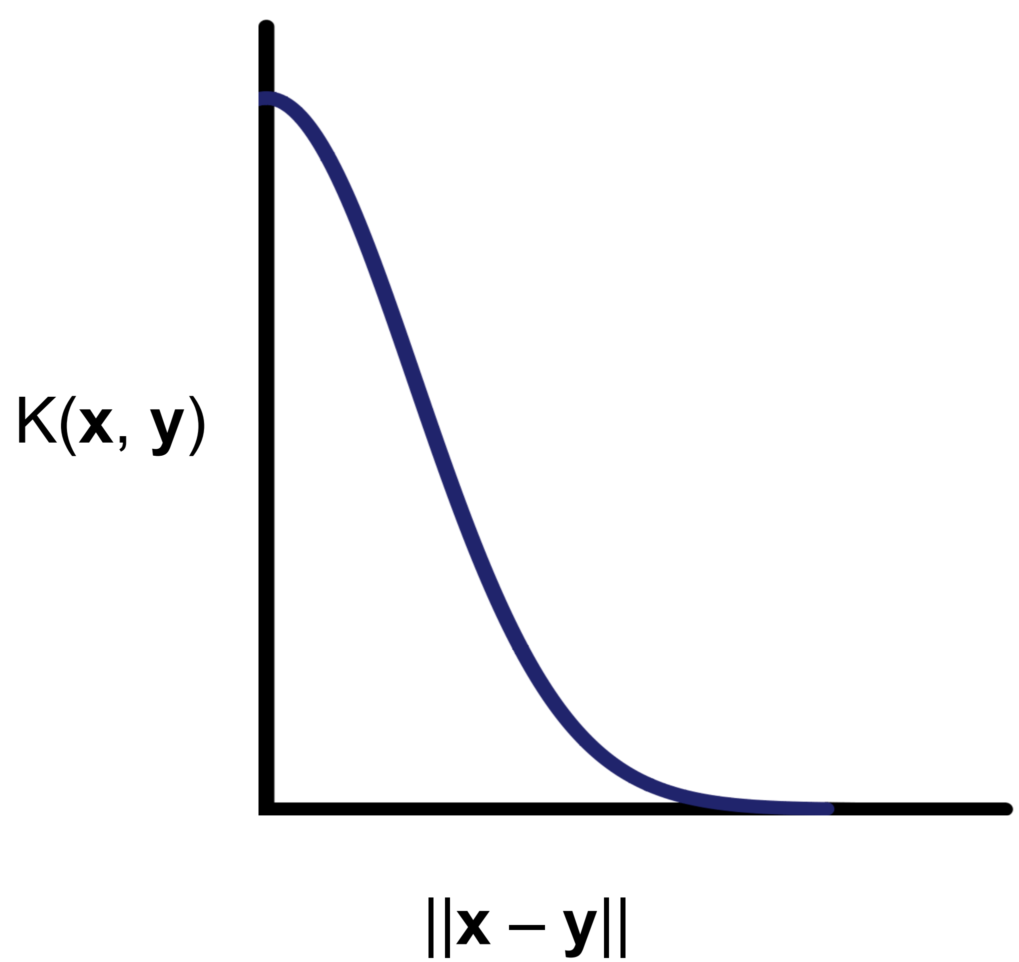
C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Source: S. Lazebnik

Gaussian kernel

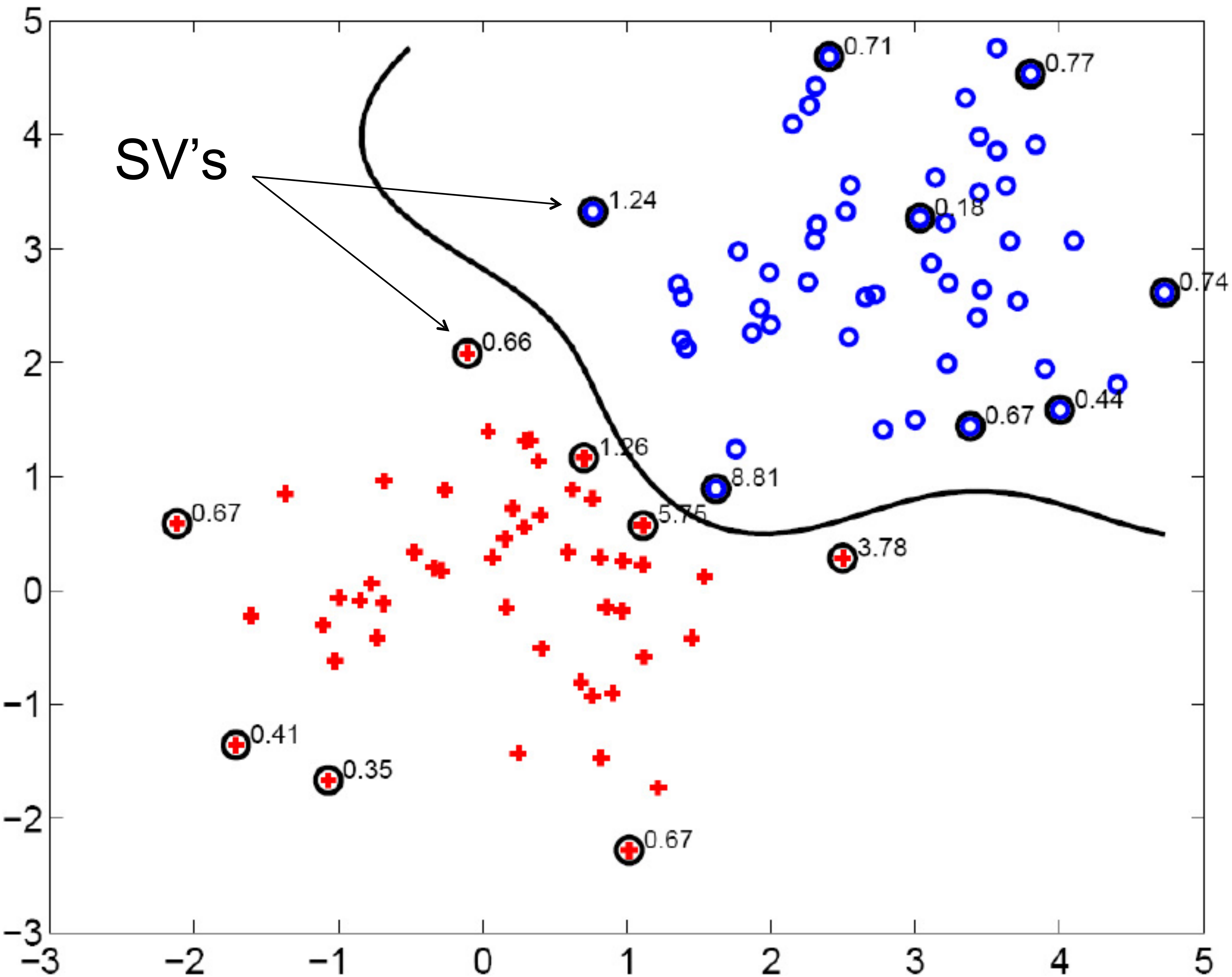
- Also known as the radial basis function (RBF) kernel:

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



Source: S. Lazebnik

Gaussian kernel



Source: S. Lazebnik

SVMs: Pros and cons

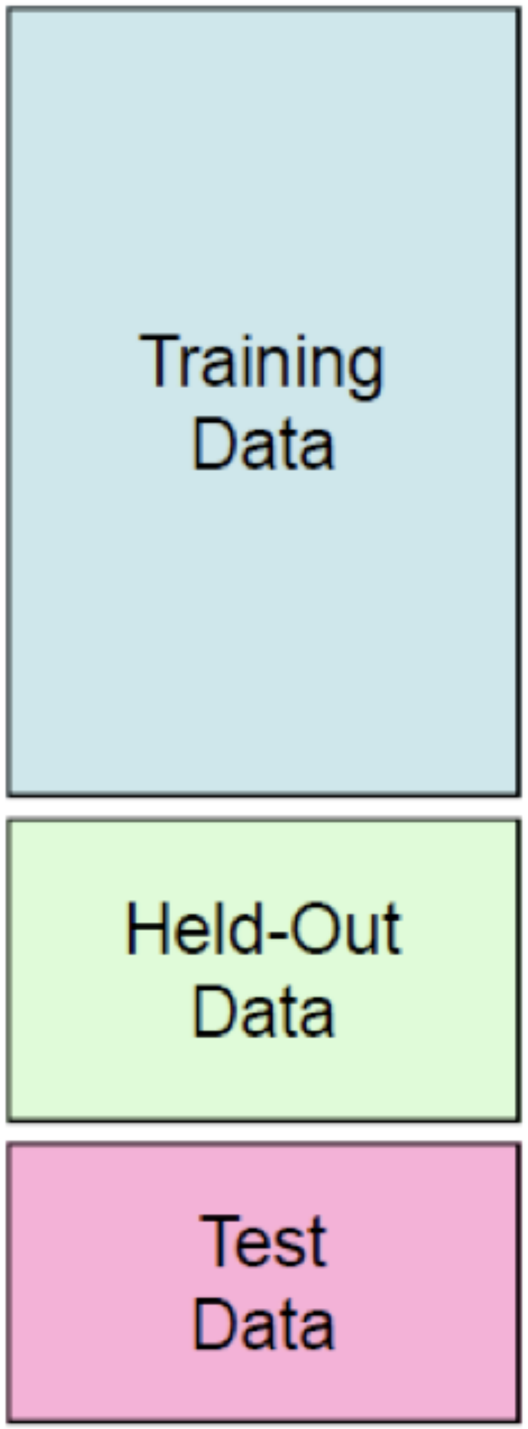
- Pros
  - Kernel-based framework is very powerful, flexible
  - Training is convex optimization, globally optimal solution can be found
  - Amenable to theoretical analysis
  - SVMs work very well in practice, even with very small training sample sizes
- Cons
  - No “direct” multi-class SVM, must combine two-class SVMs (e.g., with one-vs-others)
  - Computation, memory (esp. for nonlinear SVMs)

Source: S. Lazebnik

Best practices for training classifiers

- Goal: obtain a classifier with **good generalization** or performance on never before seen data

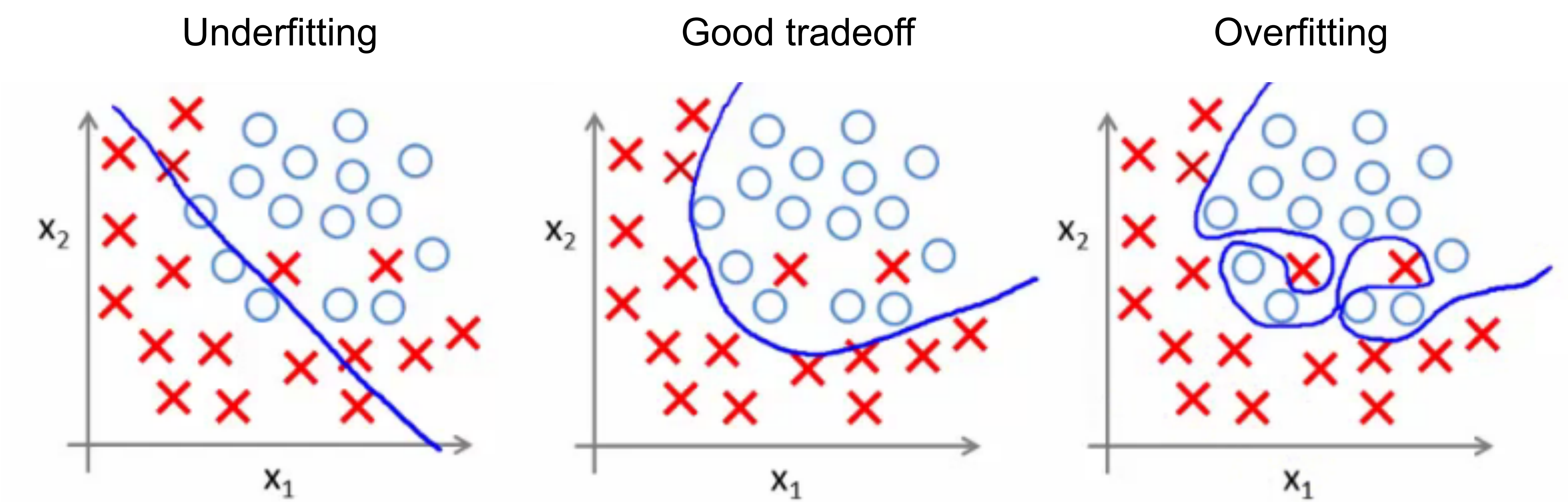
1. Learn *parameters* on the ***training set***
2. Tune *hyperparameters* (implementation choices) on the ***held out*** ***validation set***
3. Evaluate performance on the ***test set***
   – Crucial: do not peek at the test set when iterating steps 1 and 2!



Source: S. Lazebnik

Underfitting and overfitting

- **Underfitting:** training and test error are both *high*
  – Model does an equally poor job on the training and the test set
  – The model is too "simple" to represent the data or the model is not trained well
- **Overfitting:** Training error is *low* but test error is *high*
  - Model fits irrelevant characteristics (noise) in the training data
  – Model is too complex or amount of training data is insufficient



Source: S. Lazebnik

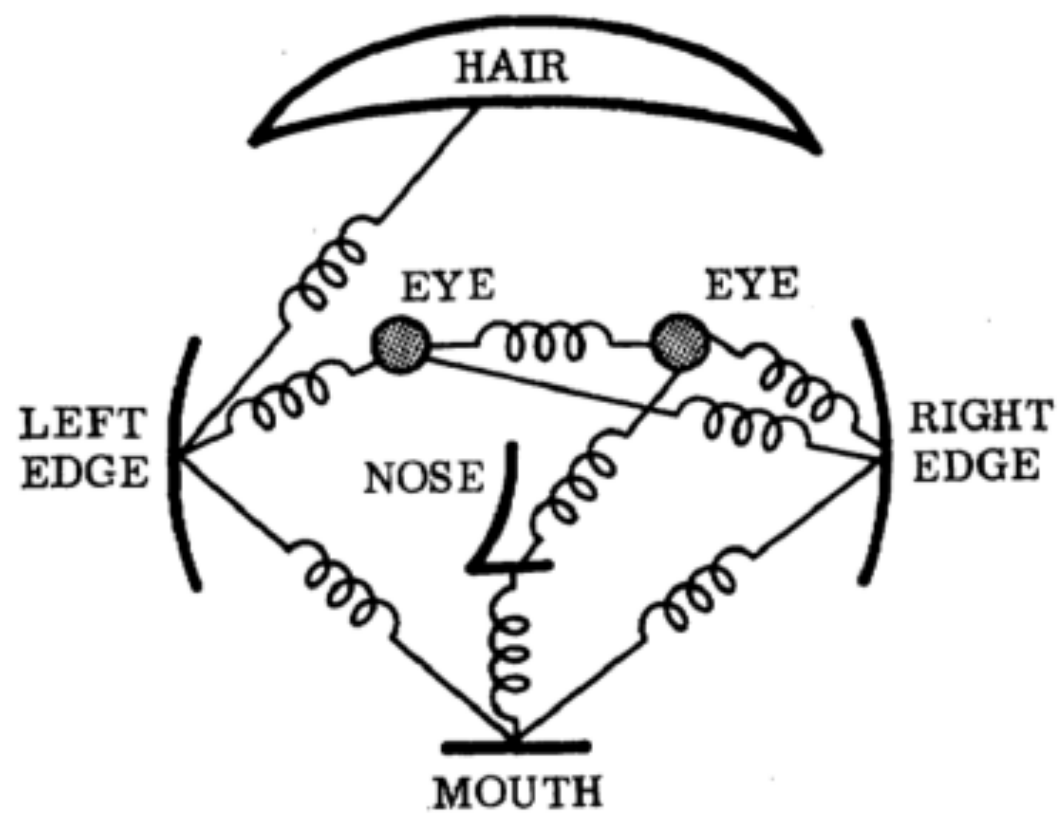
Figure source

Summary: classical detection pipeline

1. Sample a test region (e.g. densely)
2. Compute a descriptor (e.g. BoW, Histogram of Gradients)
3. Apply a simple classifier (e.g. Linear)

Constellation approach

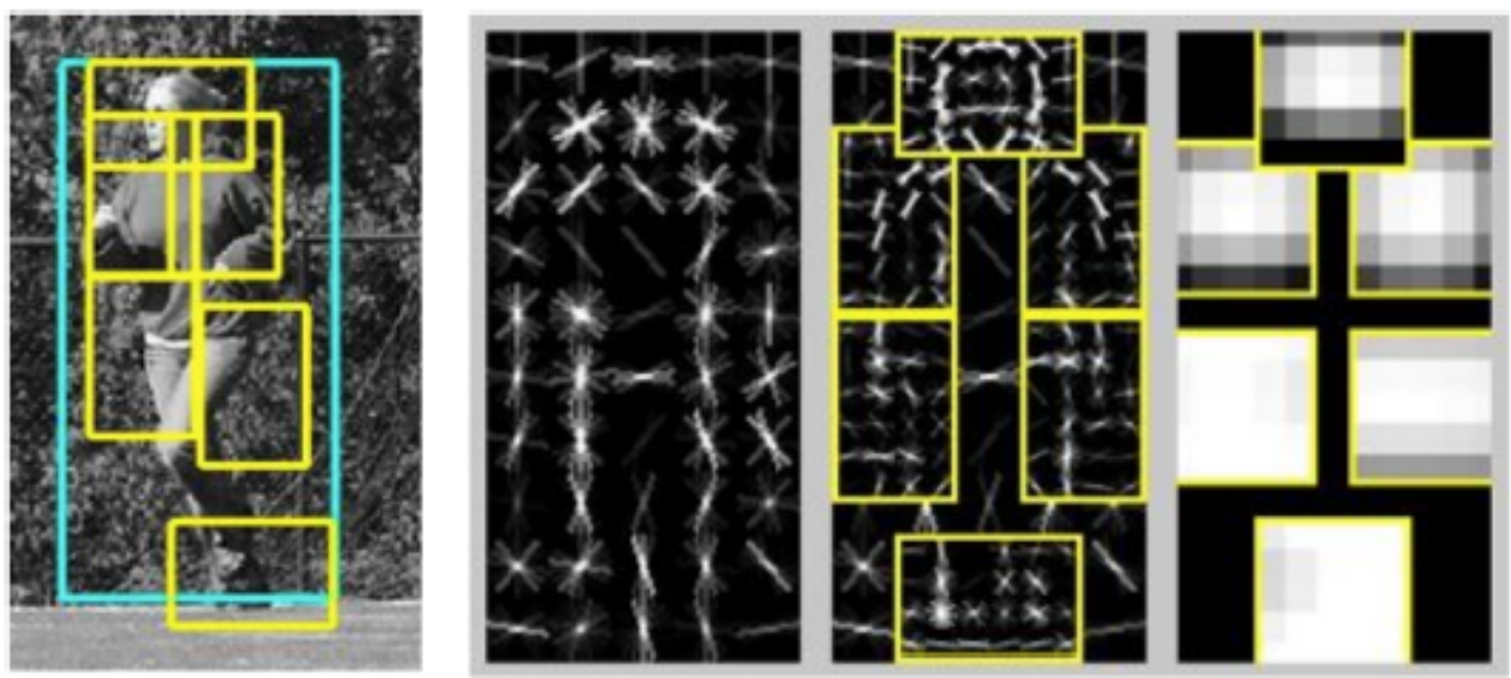
- Fischler and Elschlager [1973]



Deformable parts model

- DPM, Felzenszwalb et al. [2010]

Reference for classification until 2014.

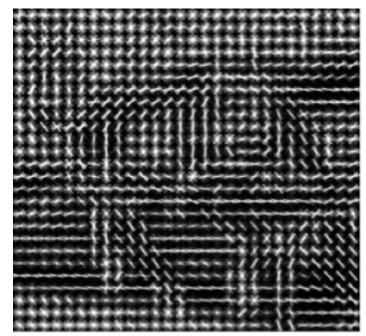


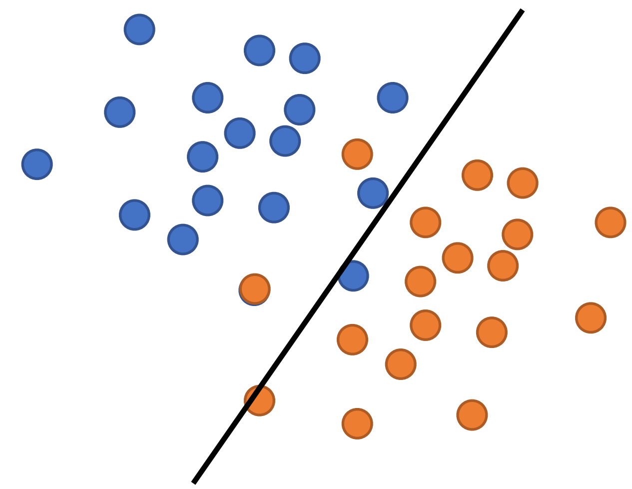
Introduction to Neural Networks

Many slides from M. Aubry

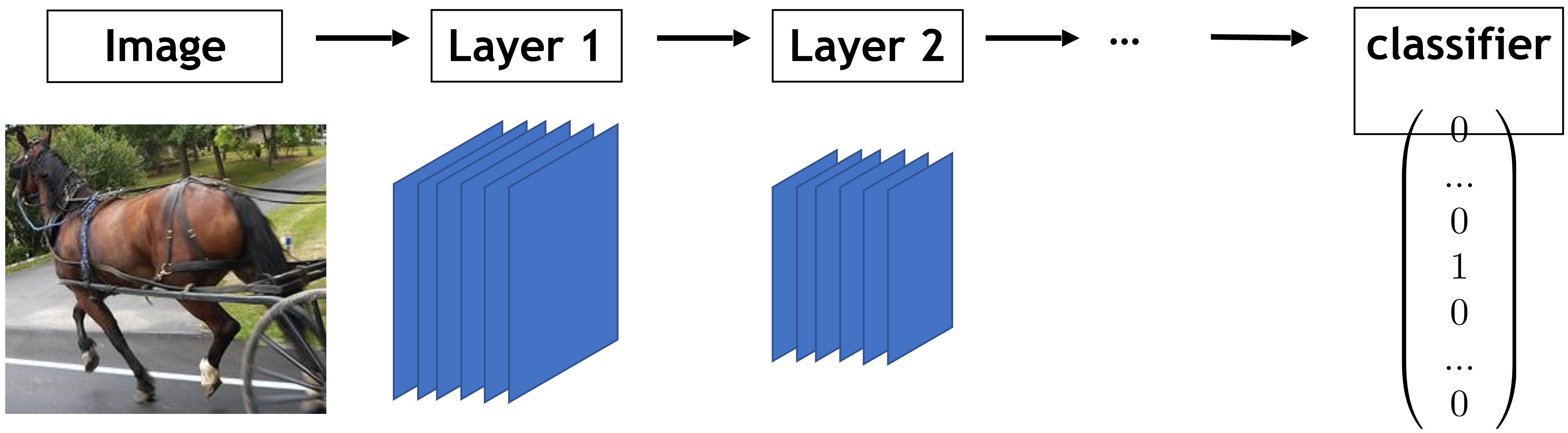
Example: classical vision

Image → **Hand crafted decriptor** Ex: HOG, bag of features → **Simple classifier:** Ex: SVM





Deep Learning



- Idea:
  1. Learn intermediate representation
  2. Compose intermediate representations

Implicit hypothesis: this compositionality is useful for the data we have

Deep representation learning

- Simple idea: learn ϕ (with a simple form)
- Combine more than two layers, learn $f \circ \phi_1 \circ \phi_2 \circ \phi_3 ...$
  = hierarchical representation, multilayer perceptron

Relationship/difference with kernels:

- The mapping is explicit and learned (often implicit and hand designed in kernel methods)
- The result of the mapping is relatively low dimensional
- Not a convex problem -> no guarantees

Relation to Kernel idea

Supervised learning:

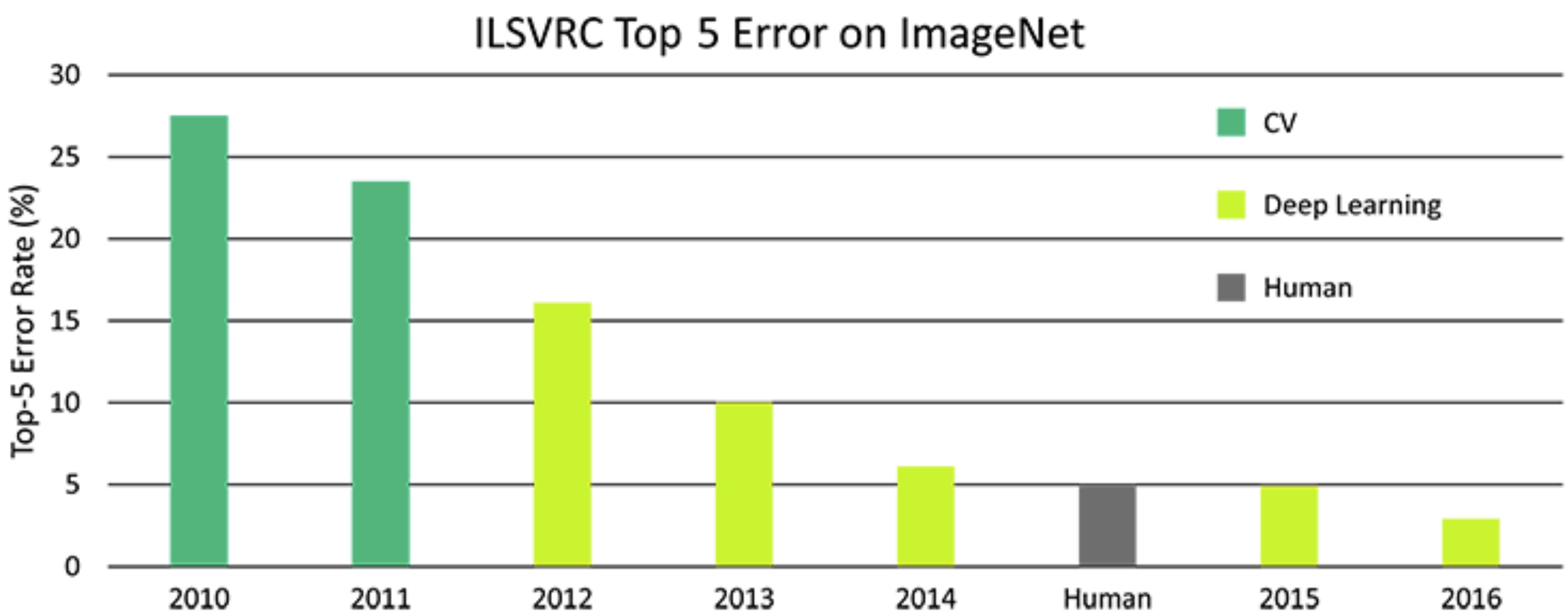
- n training data pairs $(x_1, y_1), ..., (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$
- Learn a linear predictor/decision function $\hat{f} : \mathcal{X} \to \mathcal{A}$

(Logistic regression, SVM…)

Kernel:

- Replace the dot product $< x|y >$ by a kernel $K(x, y) =< \phi(x)|\phi(y) >$
- Can be interpreted as learning a classifier $\hat{f} \circ \phi$
- More powerful, but you have to design the kernel

Results



https://www.dsiac.org

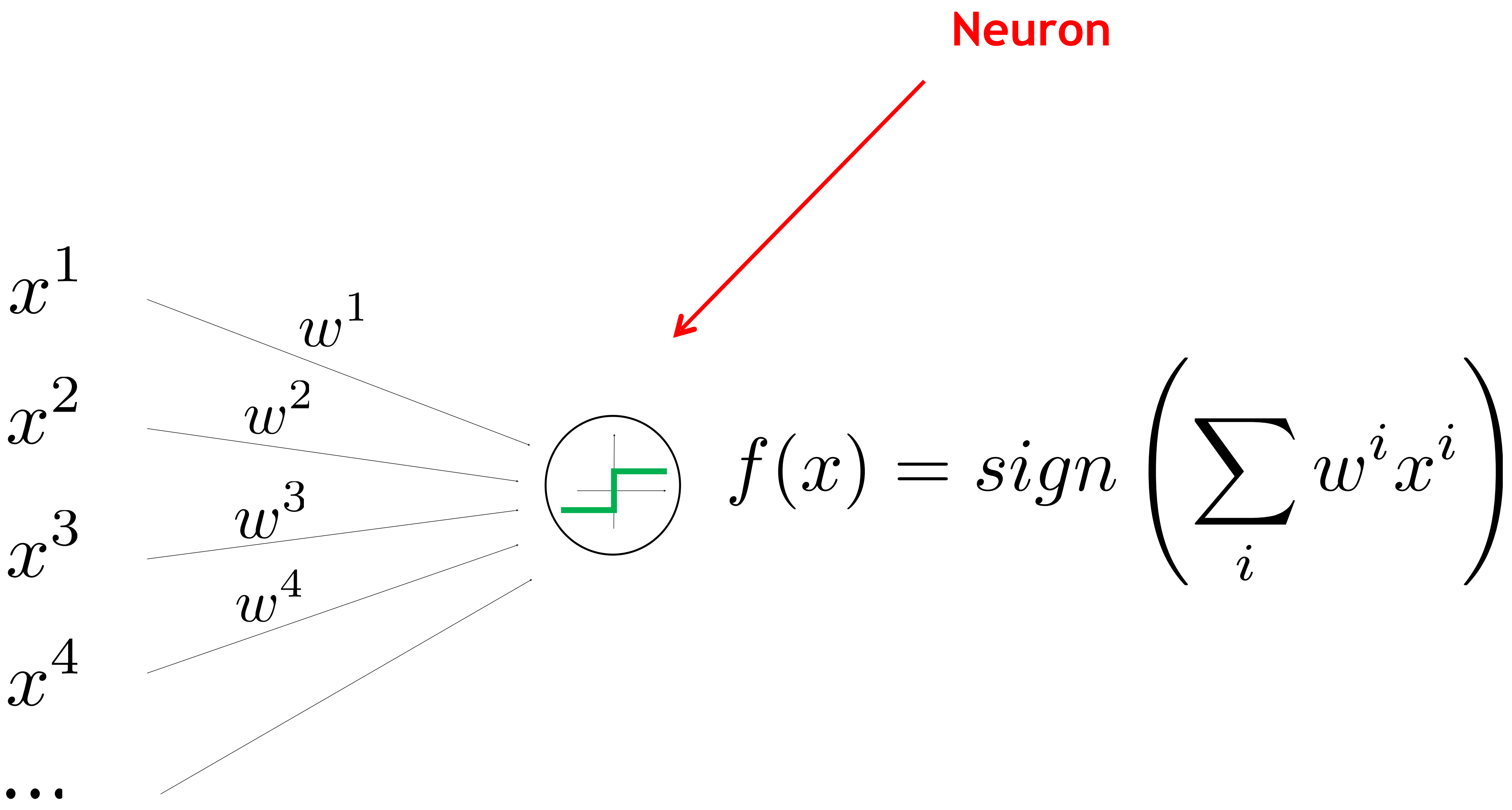
Perceptron

- Frank Rosenblatt, 1957

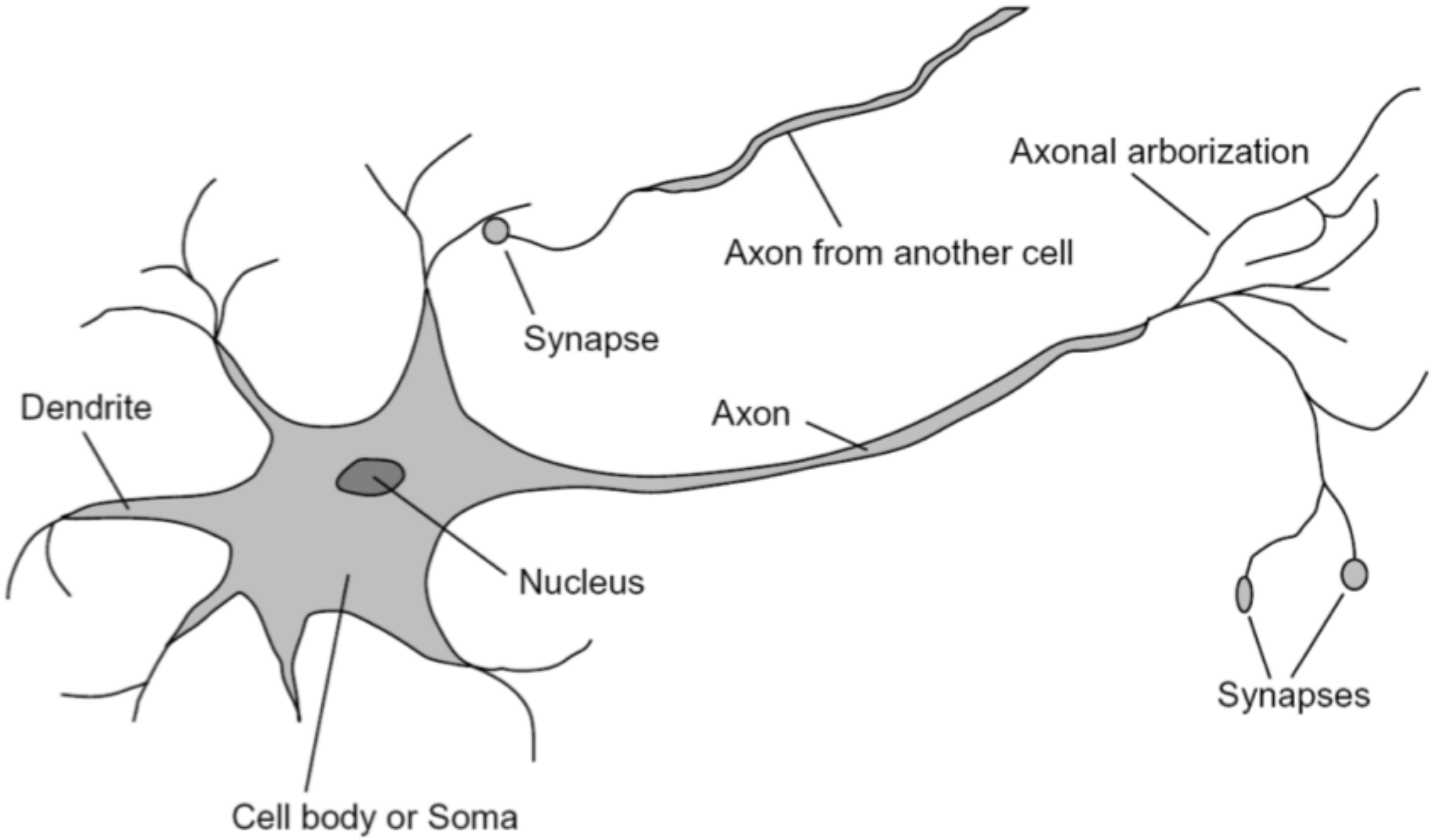
$$f(x) = sign\left(\sum_i w^i x^i\right)$$

Perceptron

- Frank Rosenblatt, 1957



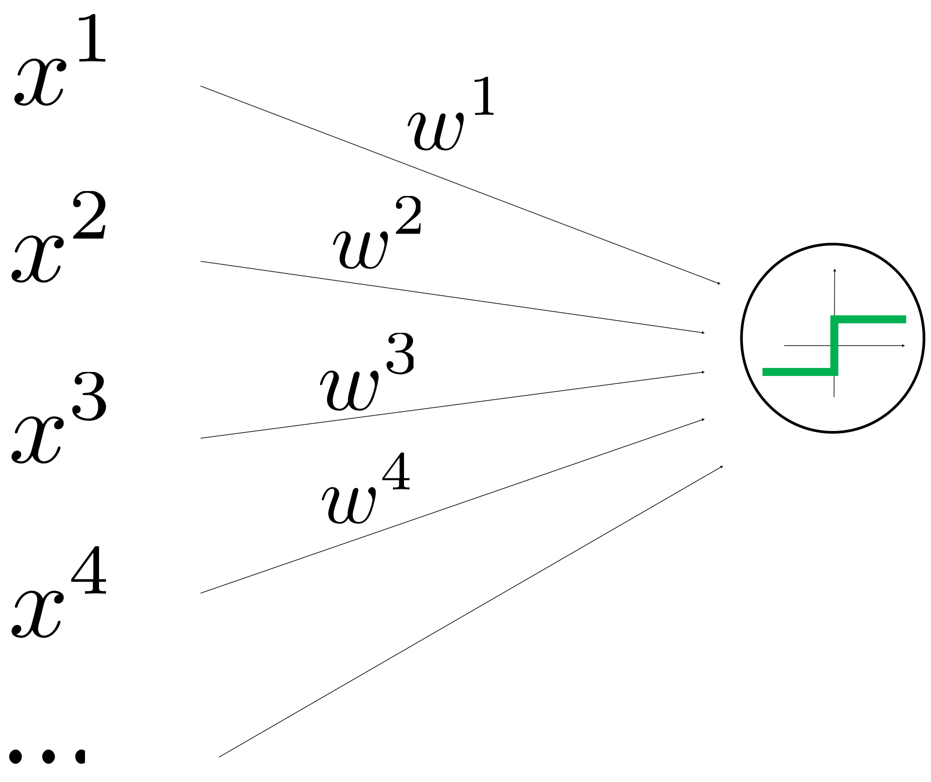
Biological neuron



Slide L. Lazebnik

Perceptron

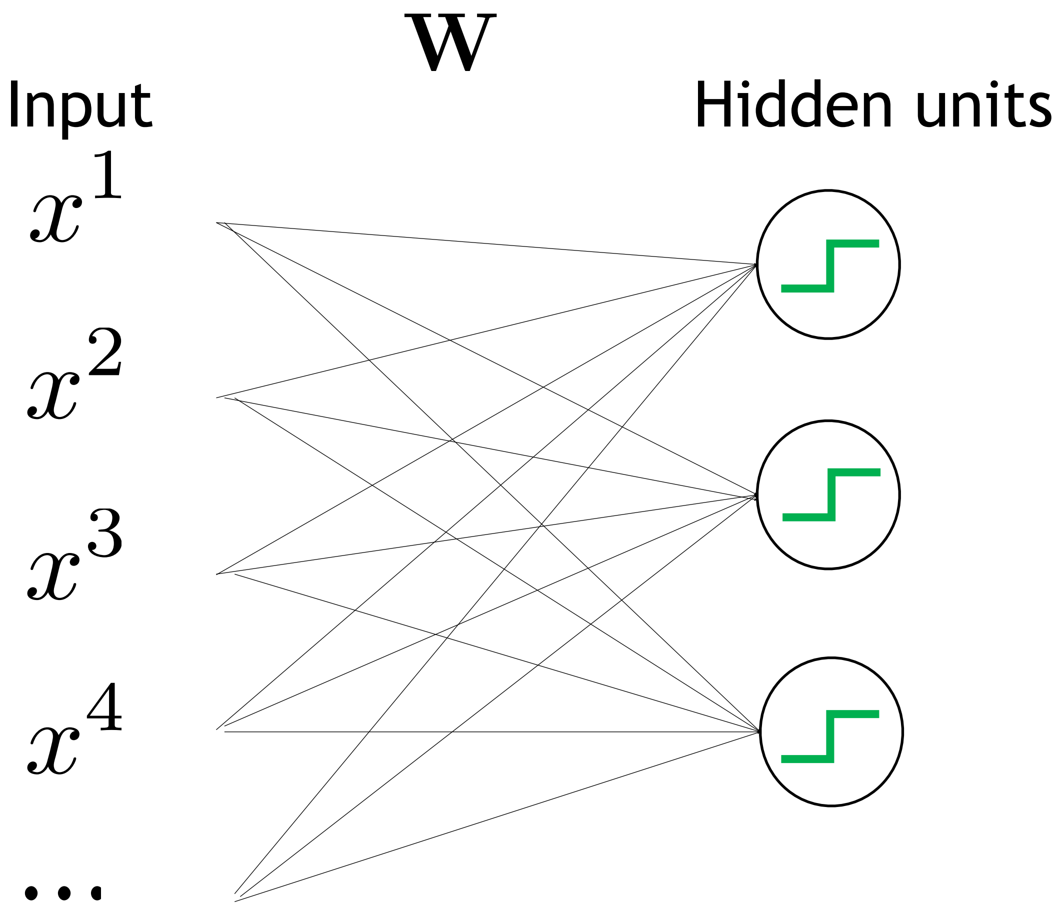
- Frank Rosenblatt, 1957



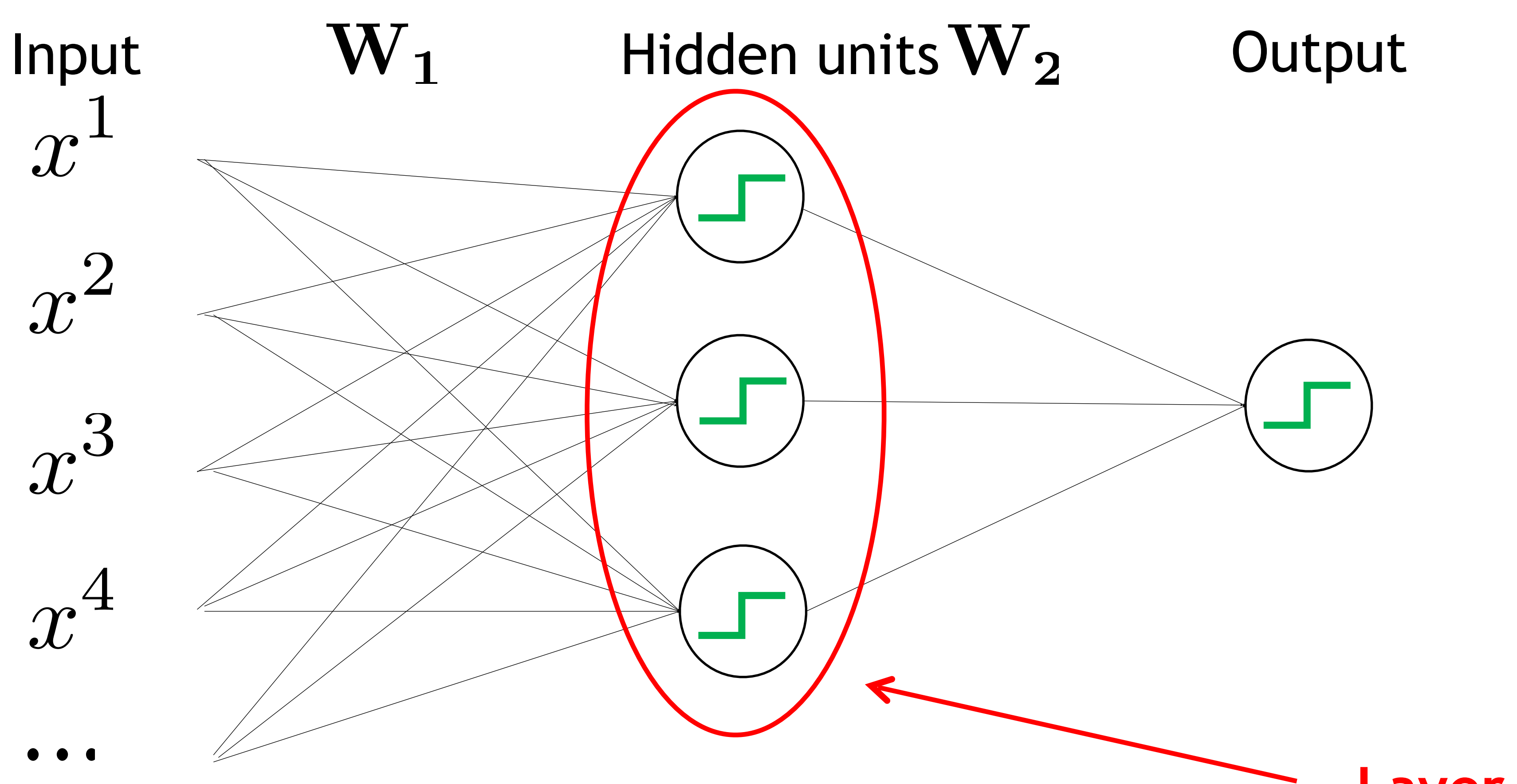
$$f(x) = sign(w^T x)$$

Issue: incapable of performing XOR (Minsky and Papert 1969)

Perceptron



2 layers perceptron



Abstraction

Input → Layer 1 (linear + non-linearity) → Layer 2 (linear) → Output

Non linearities

- Sign, sigmoid, tanh, ReLu, “leaky” ReLu ($max(x, \epsilon x)$)

- In practice, some can make the networks harder to train.

- Lots of success with ReLu
  - Avoids extremely small derivatives (e.g. of a sigmoid)
  - Leads to sparse outputs
  - Very simple derivative

- Why non linearities?

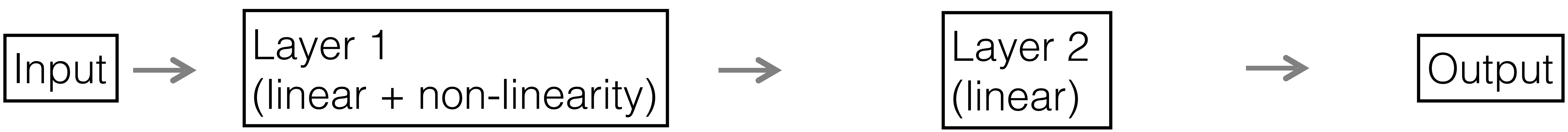
Universal approximation theorem

- A 2 layer MLP with increasing continuous and bounded non linearity can approximate any continuous function on a compact given enough hidden neurons (Cybenko 1989)

- Alternative view: the set of parametric functions defined by 2-layers MLPs is dense.

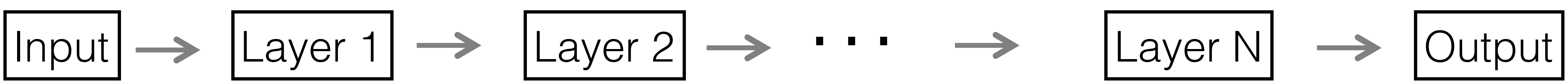
- Limitation: doesn't say anything about the number of hidden neurons required -> more layers, deeper networks could be more efficient (e.g. Bengio et al '07, Montufar et al '14)

Abstraction



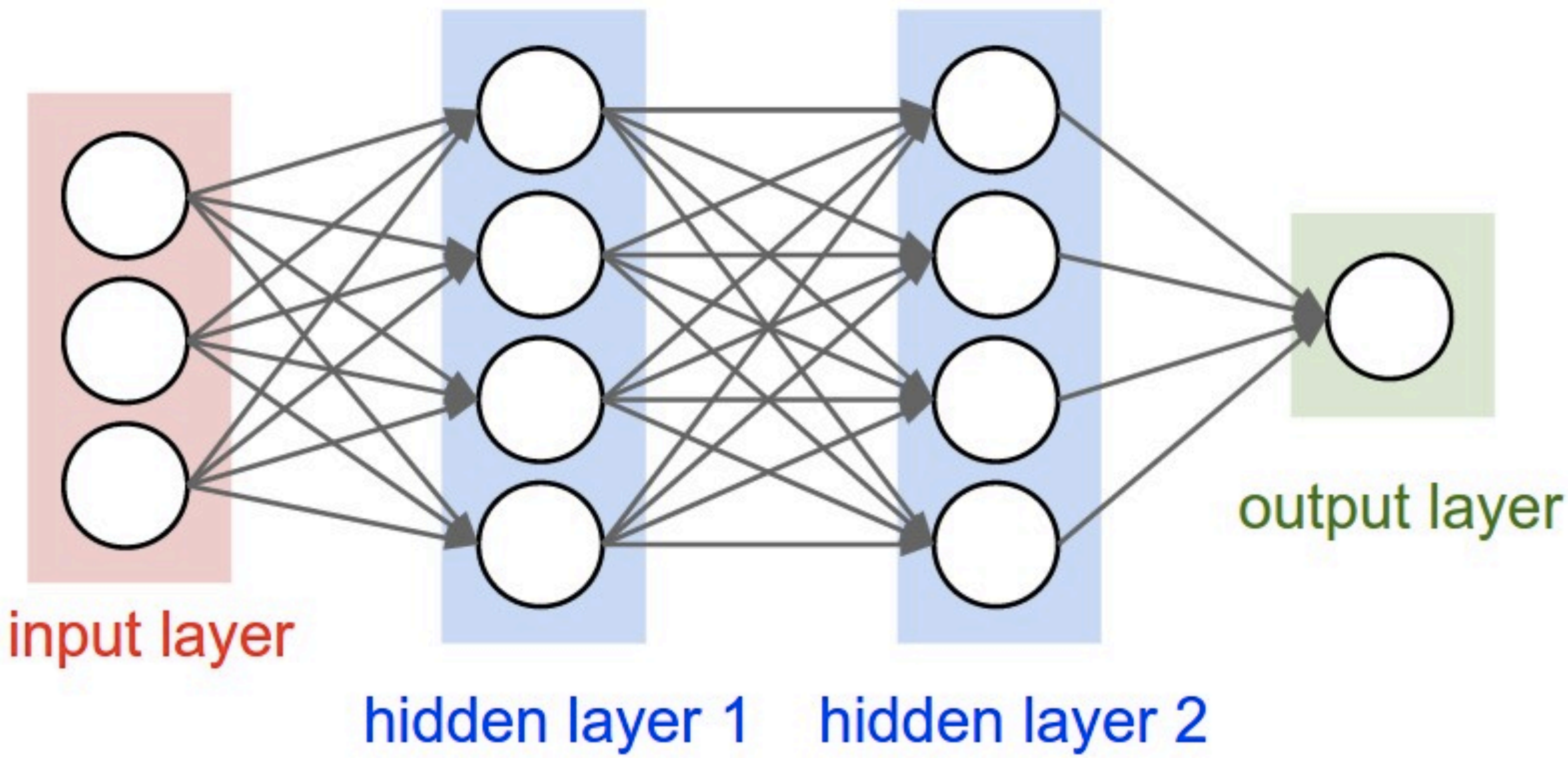
Abstraction

Feed-forward NN



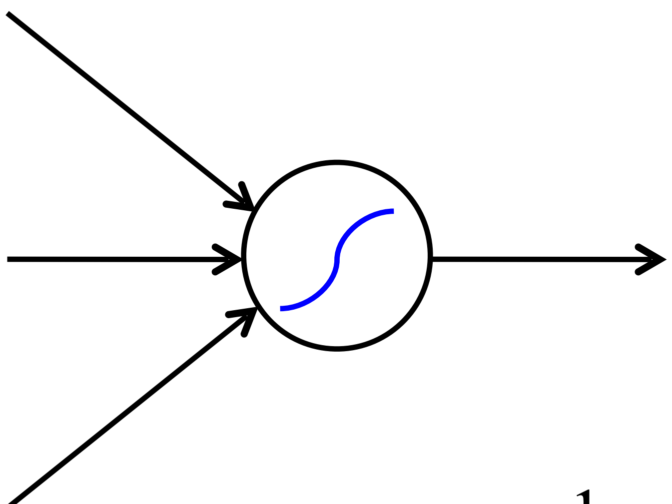
Multi-layer perceptron: all layers except the last one are Linear+NL and the last one is linear

Multi-layer perceptrons

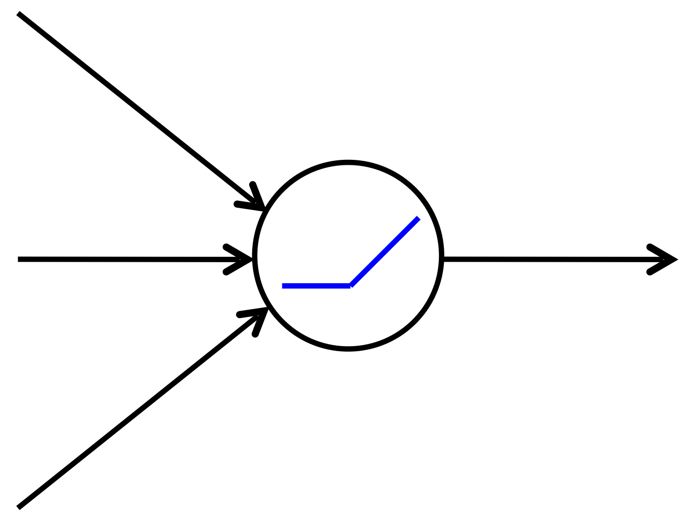


Multi-layer perceptrons

- Each perceptron to has a nonlinearity
- To be trainable, the nonlinearity should be *differentiable*



Sigmoid: $g(t) = \frac{1}{1+e^{-t}}$

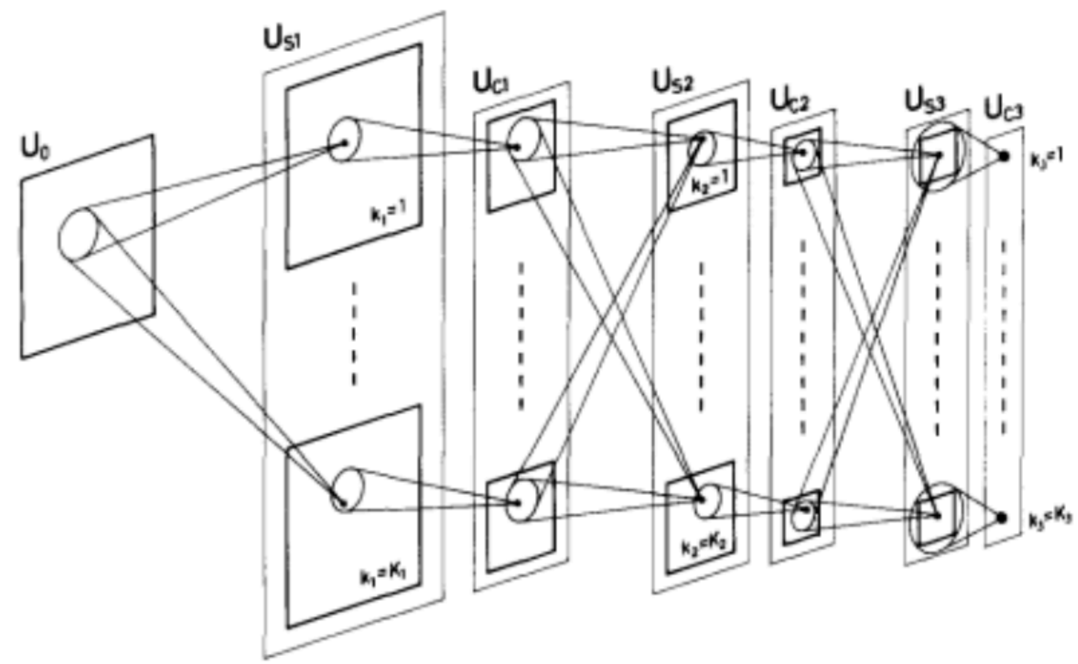


Rectified linear unit (ReLU): $g(t) = \max(0,t)$

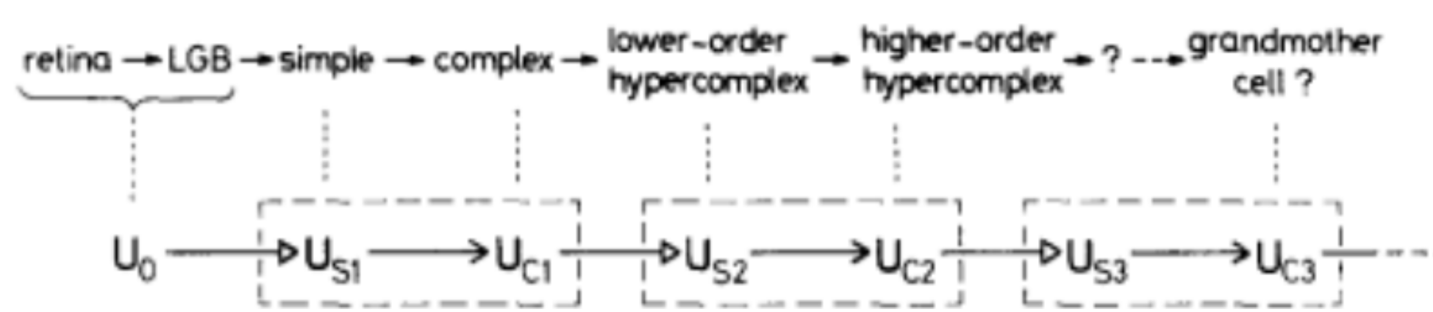
Source: S. Lazebnik

Neocognitron

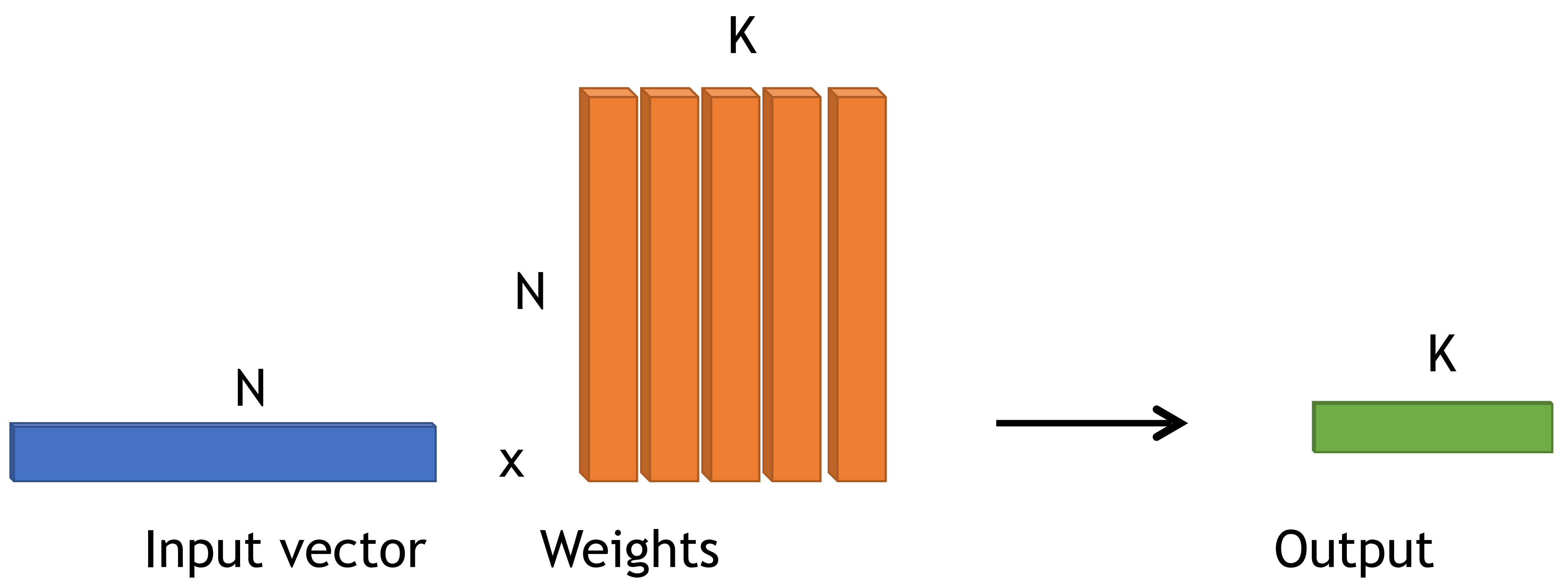
- Fukushima 1980



- Biological inspiration: Hubel and Wiesel 1962: simple and complex cells in the visual cortex

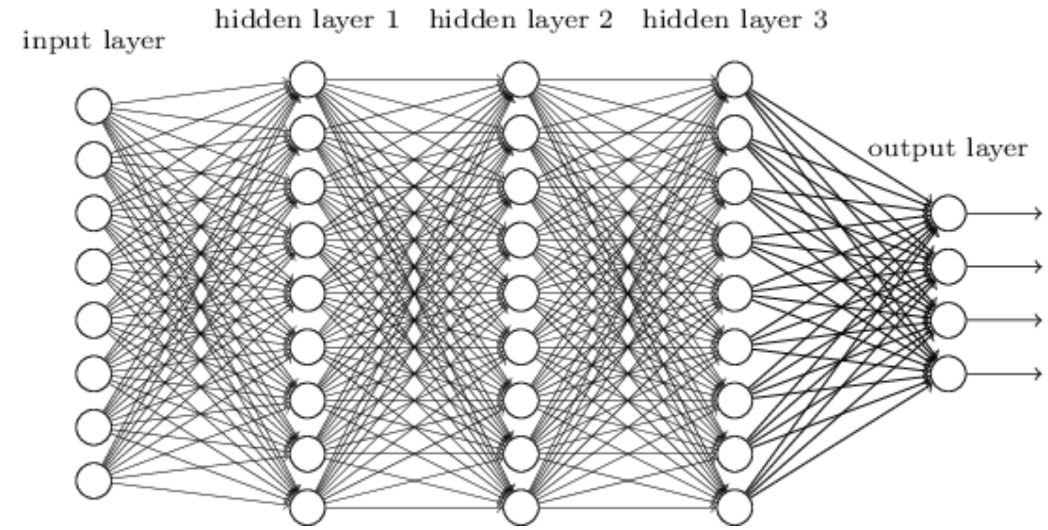


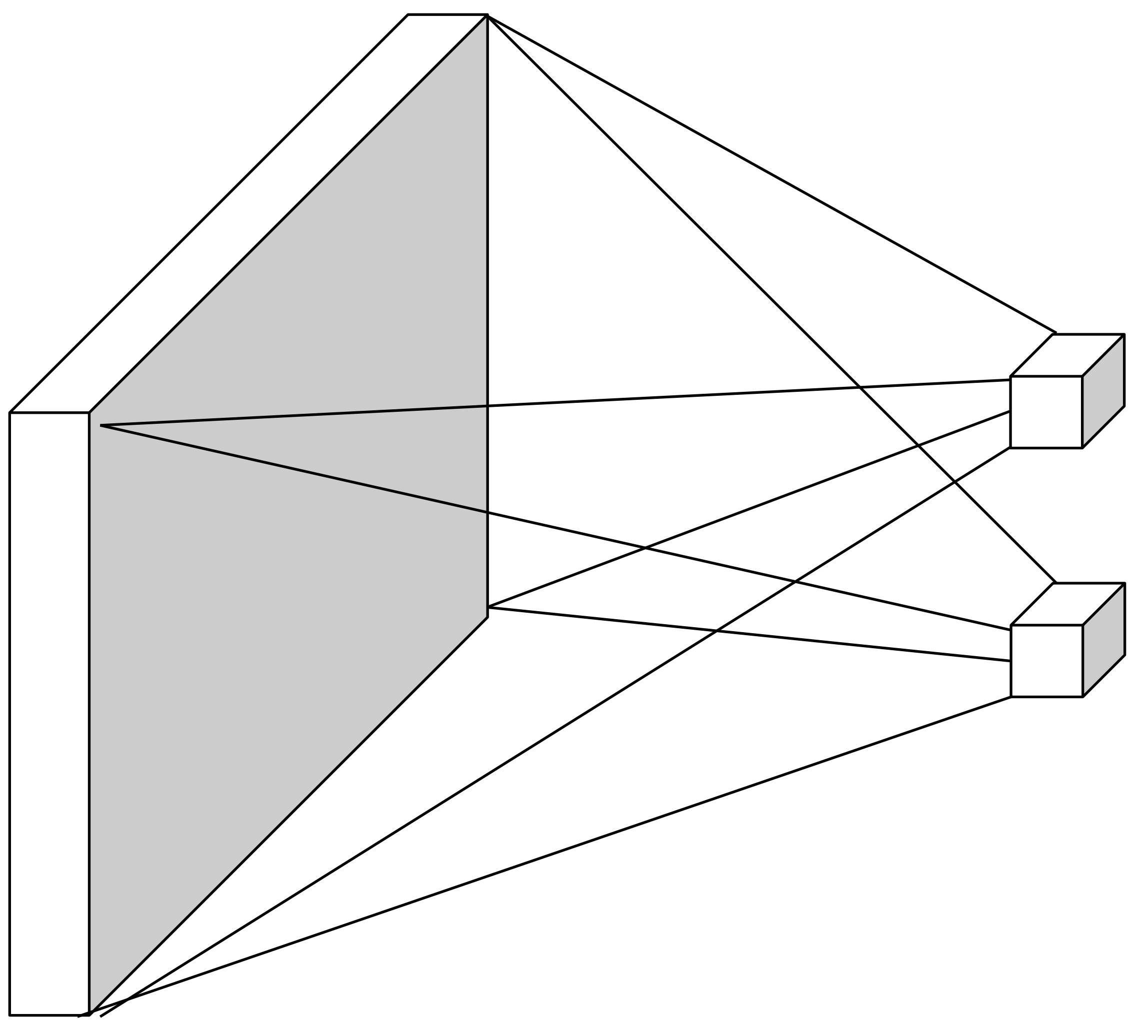
Linear



- Issue: lots of parameters

Neural networks for images



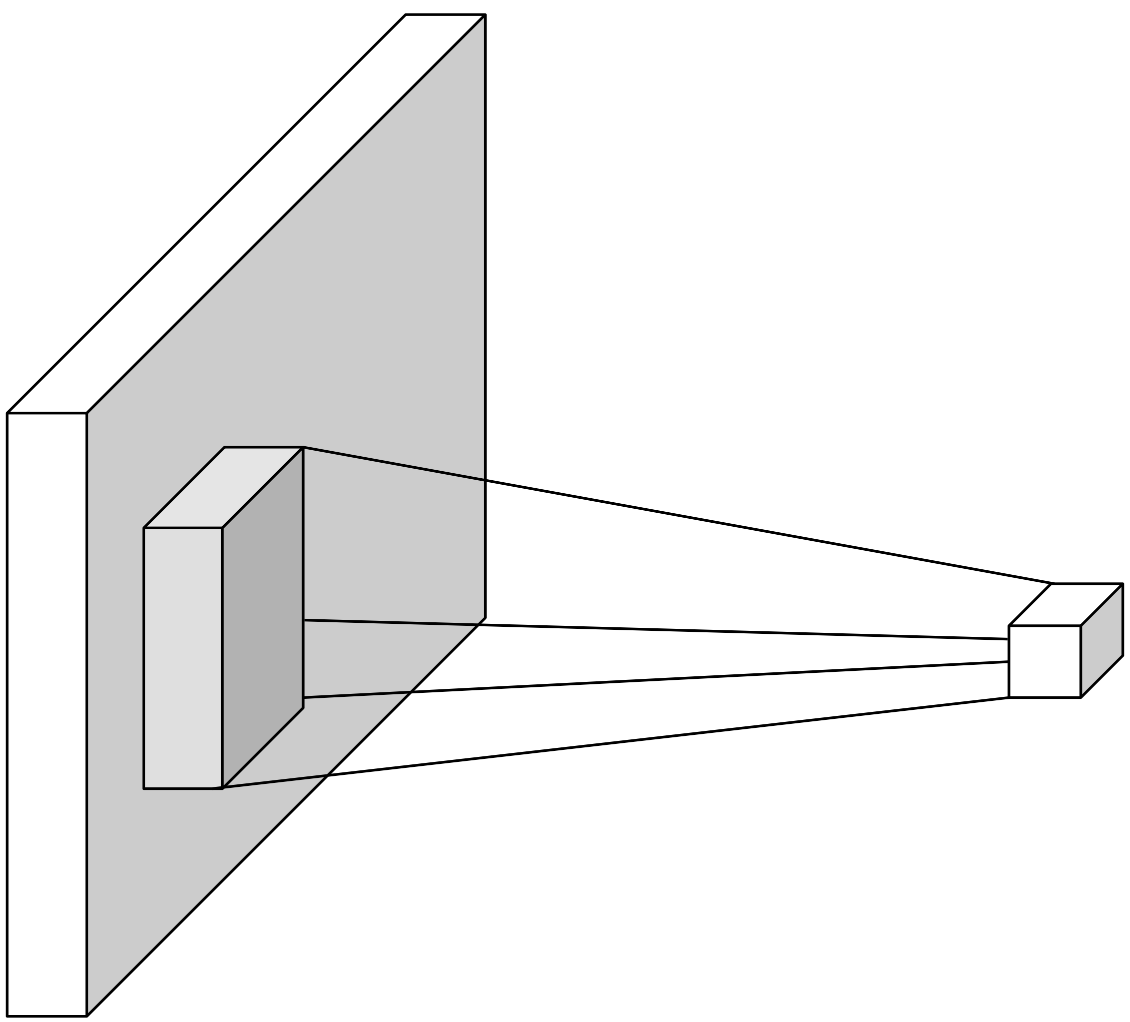


image

Fully connected layer

Source: S. Lazebnik

Neural networks for images

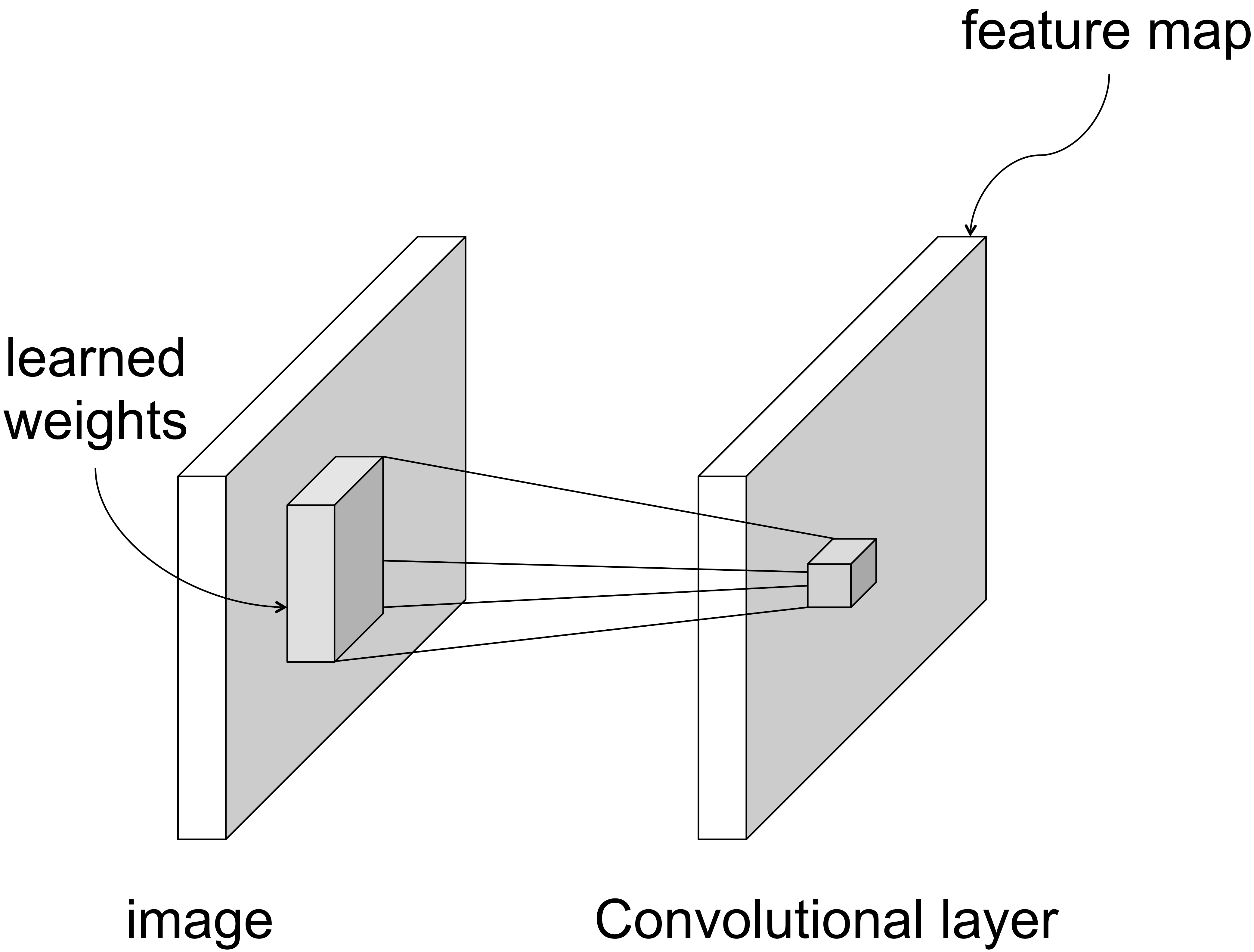


image

Convolutional layer

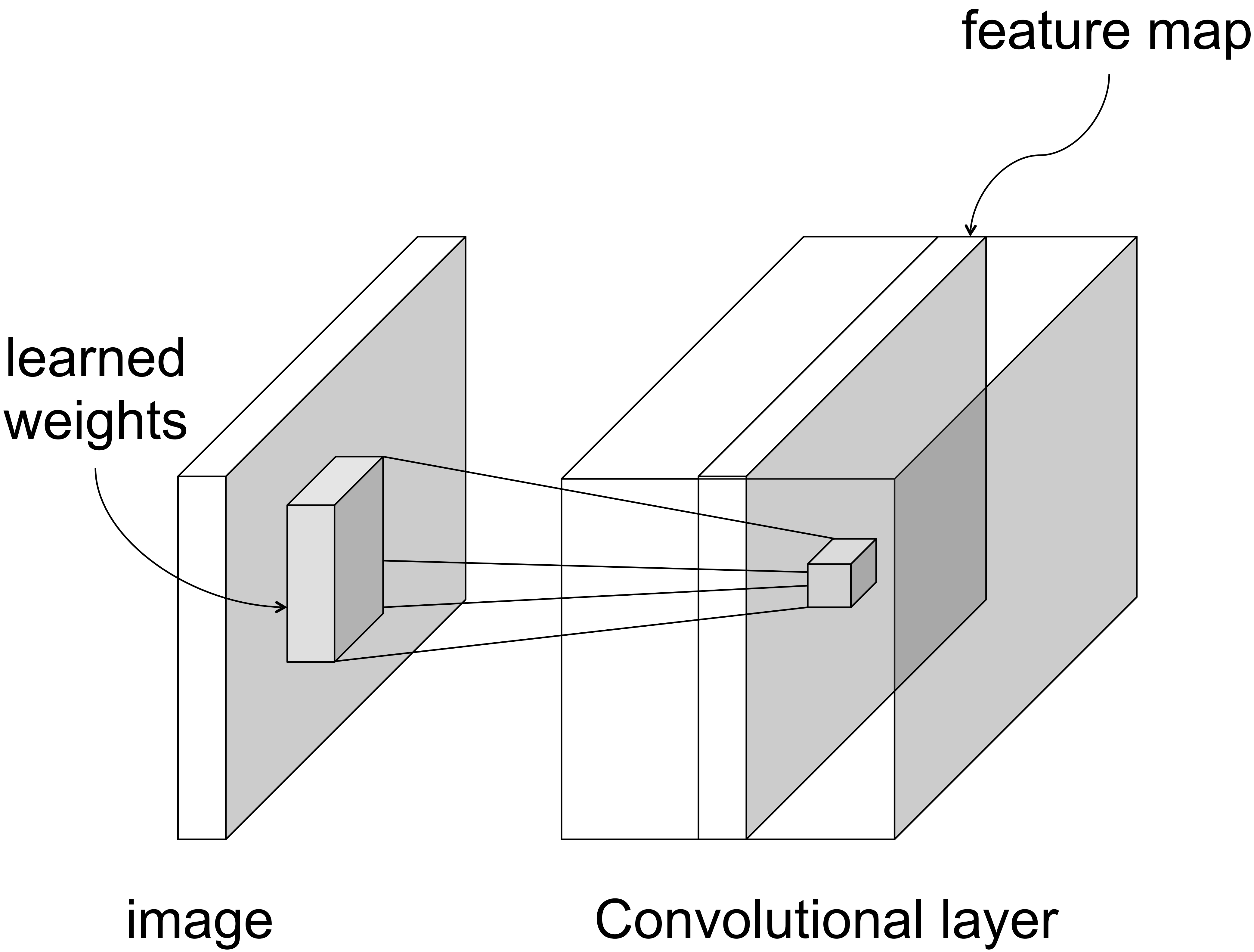
Source: S. Lazebnik

Neural networks for images



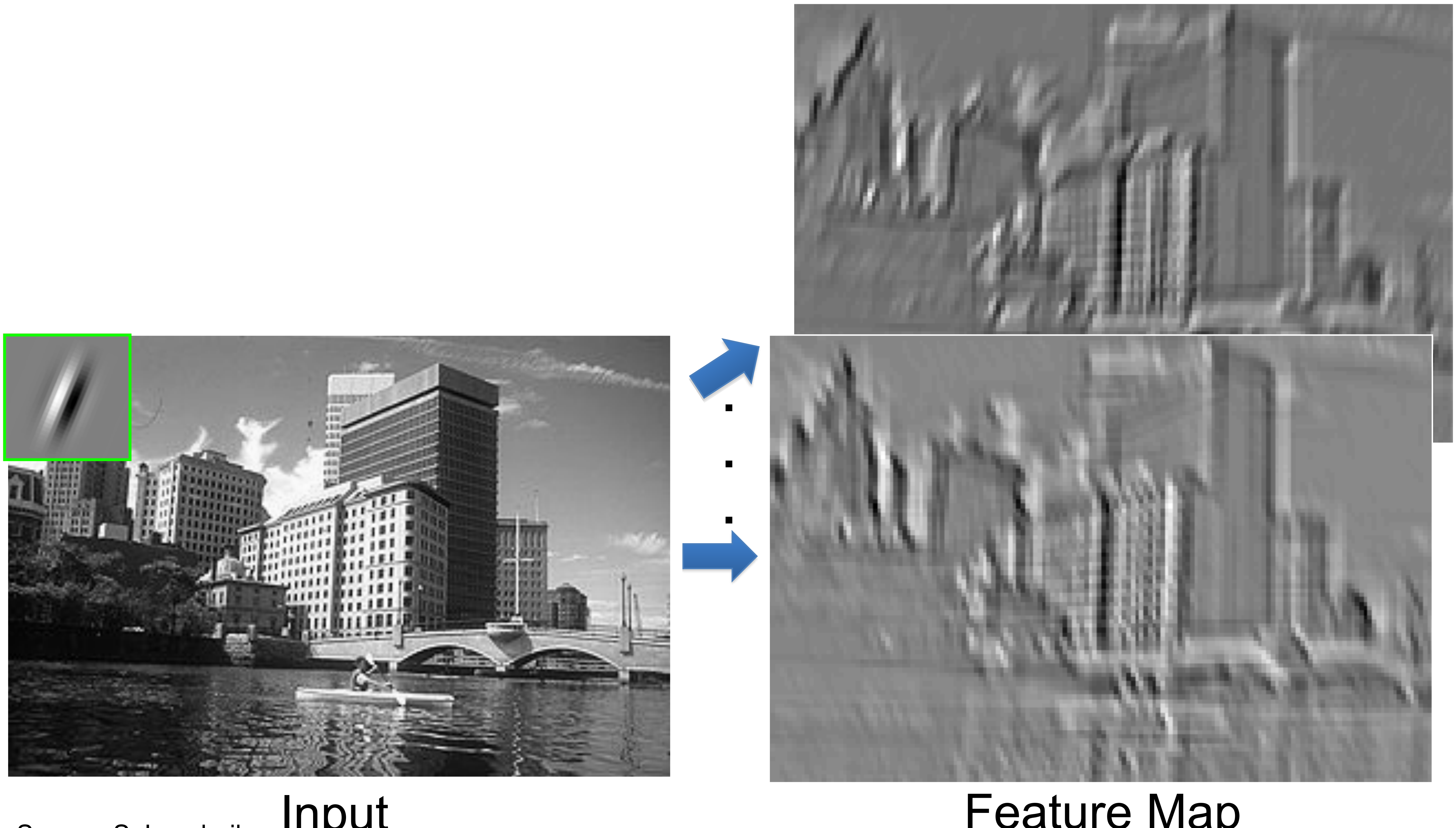
Source: S. Lazebnik

Neural networks for images



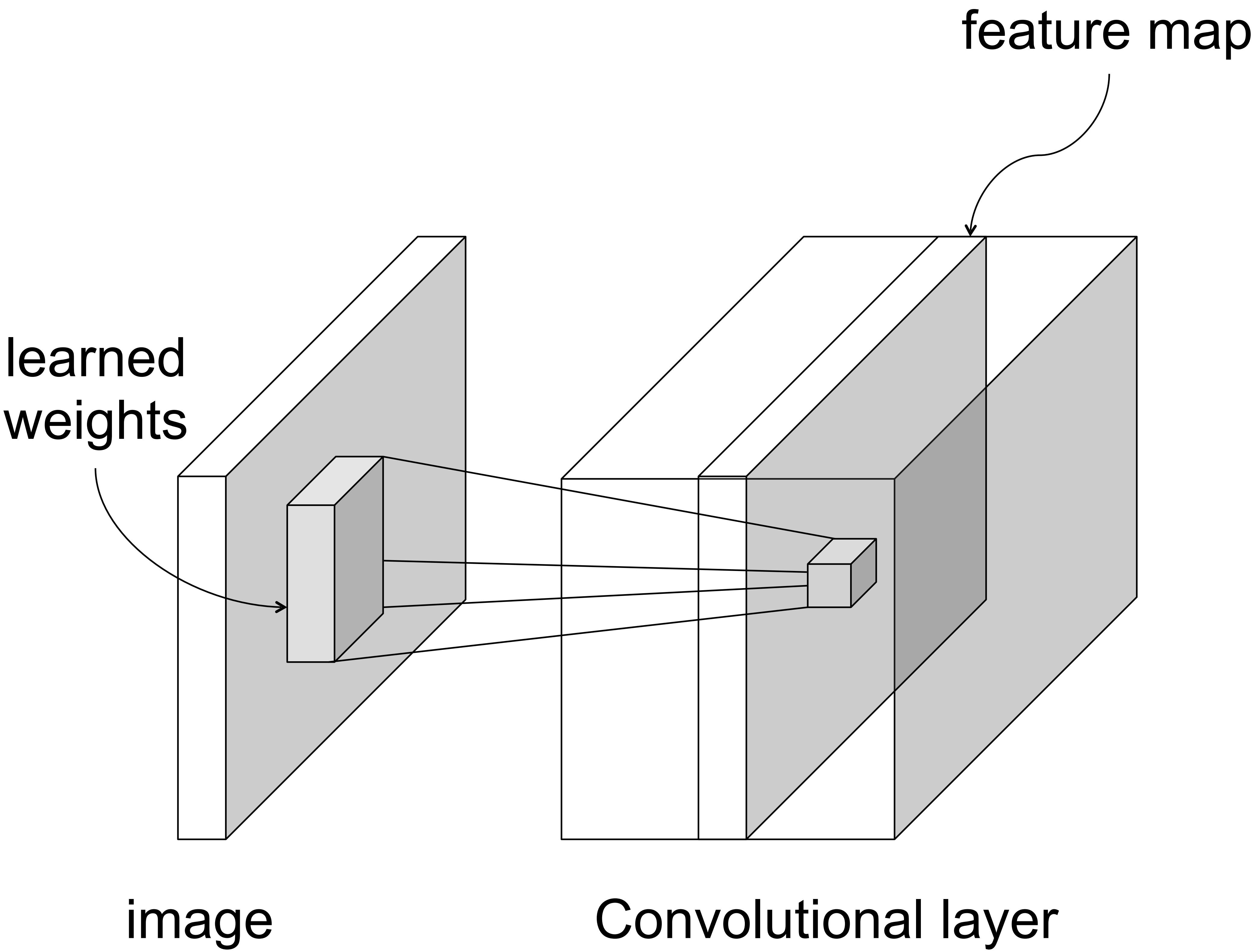
Source: S. Lazebnik

Convolution as feature extraction



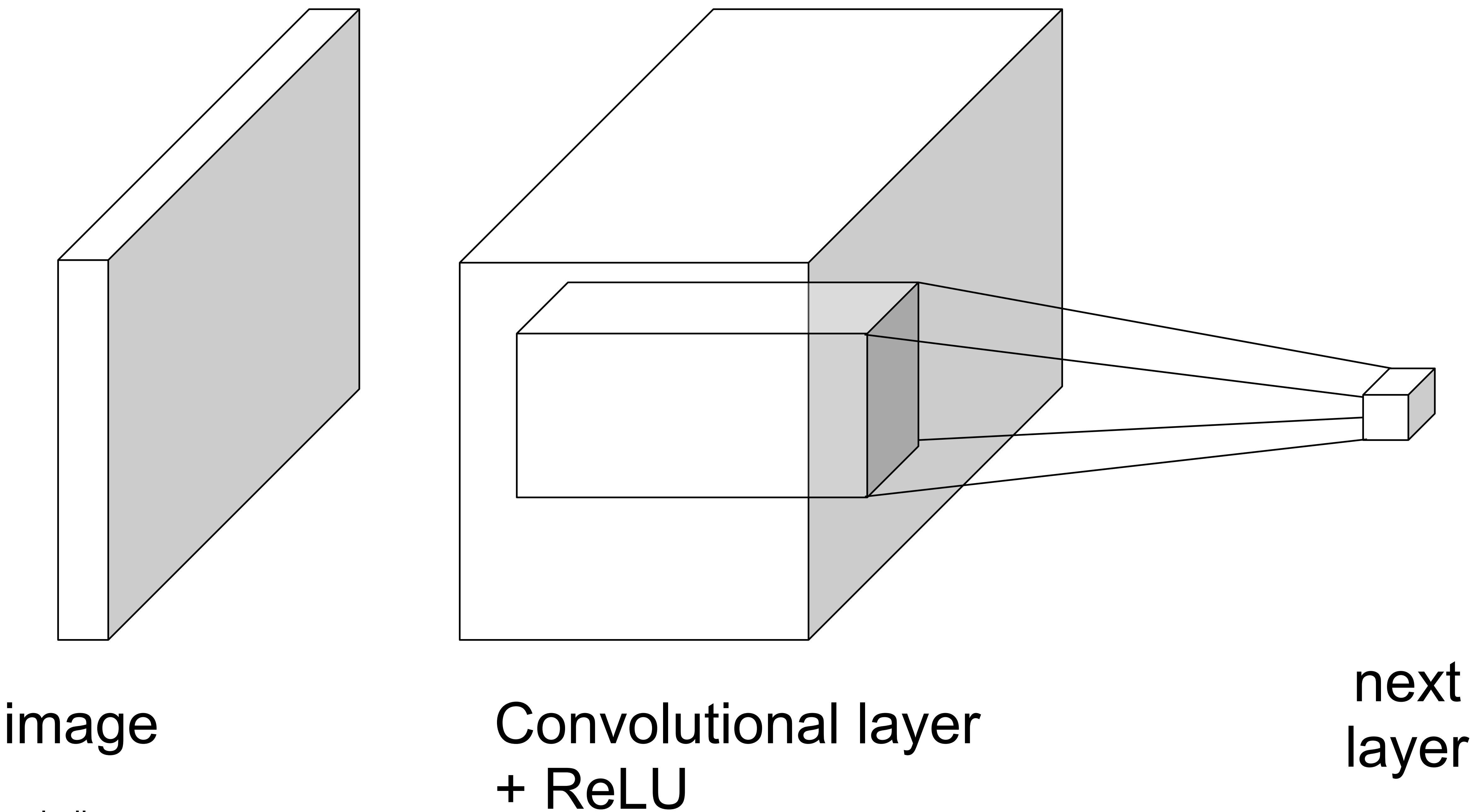
Source: S. Lazebnik

Neural networks for images



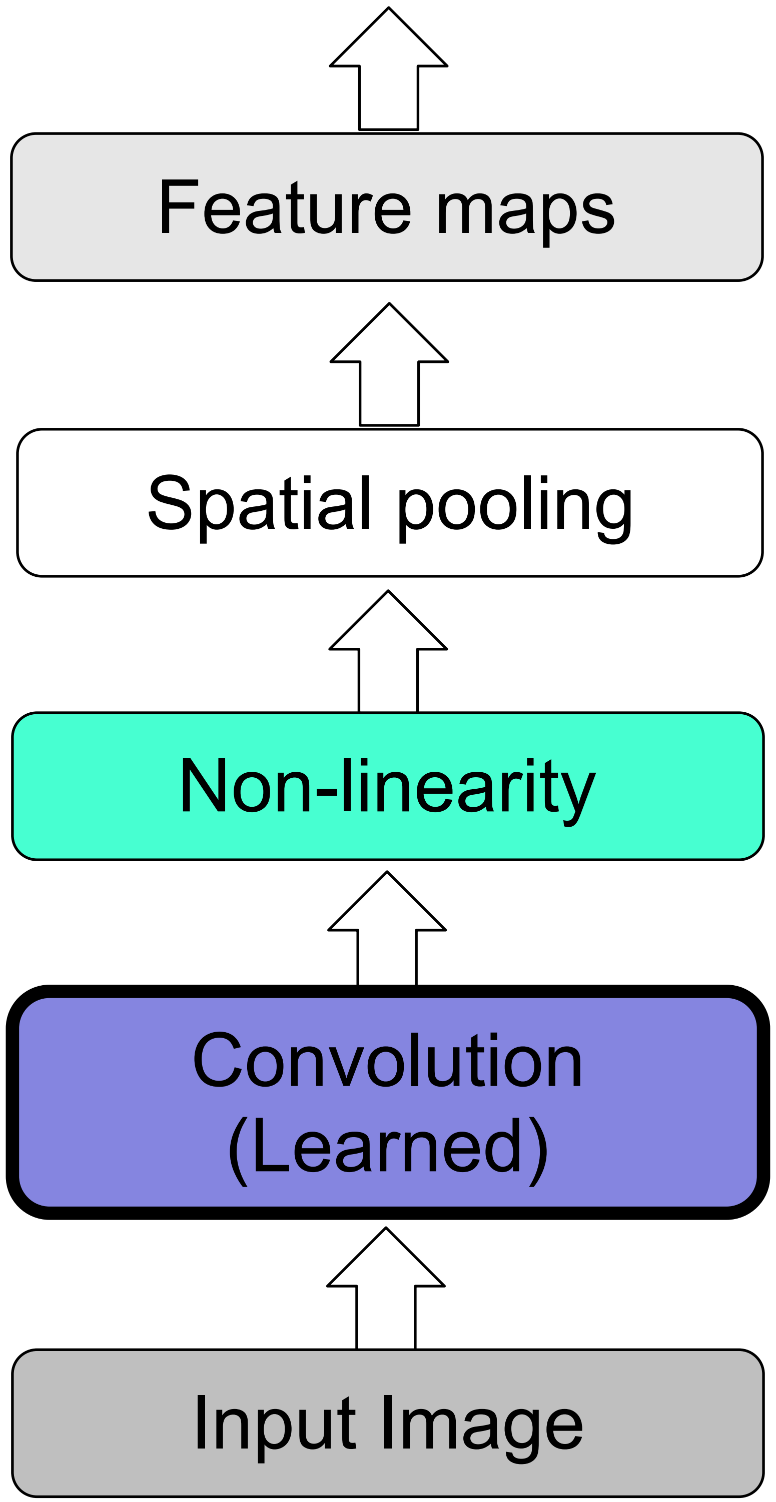
Source: S. Lazebnik

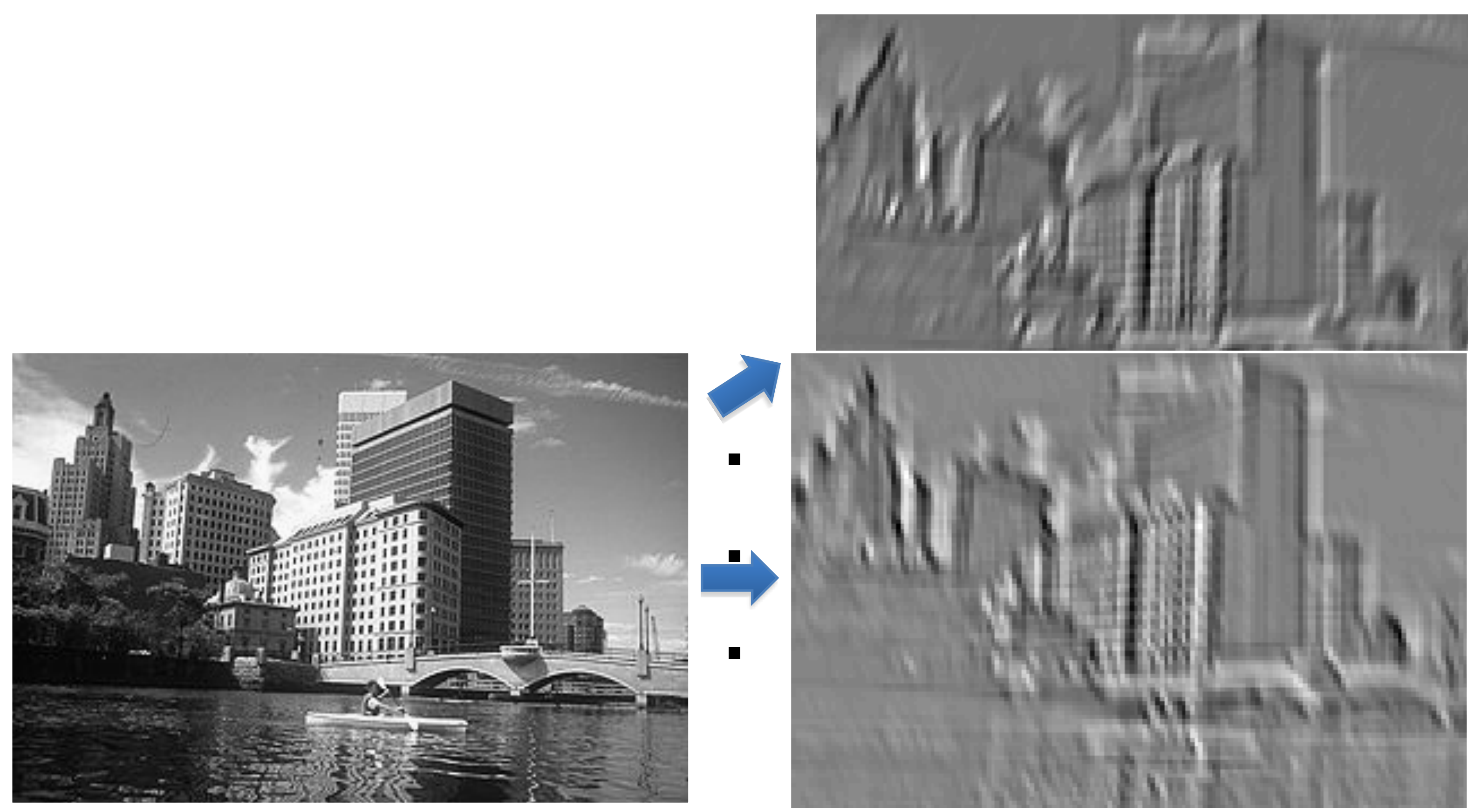
Neural networks for images



Source: S. Lazebnik

Key operations in a CNN



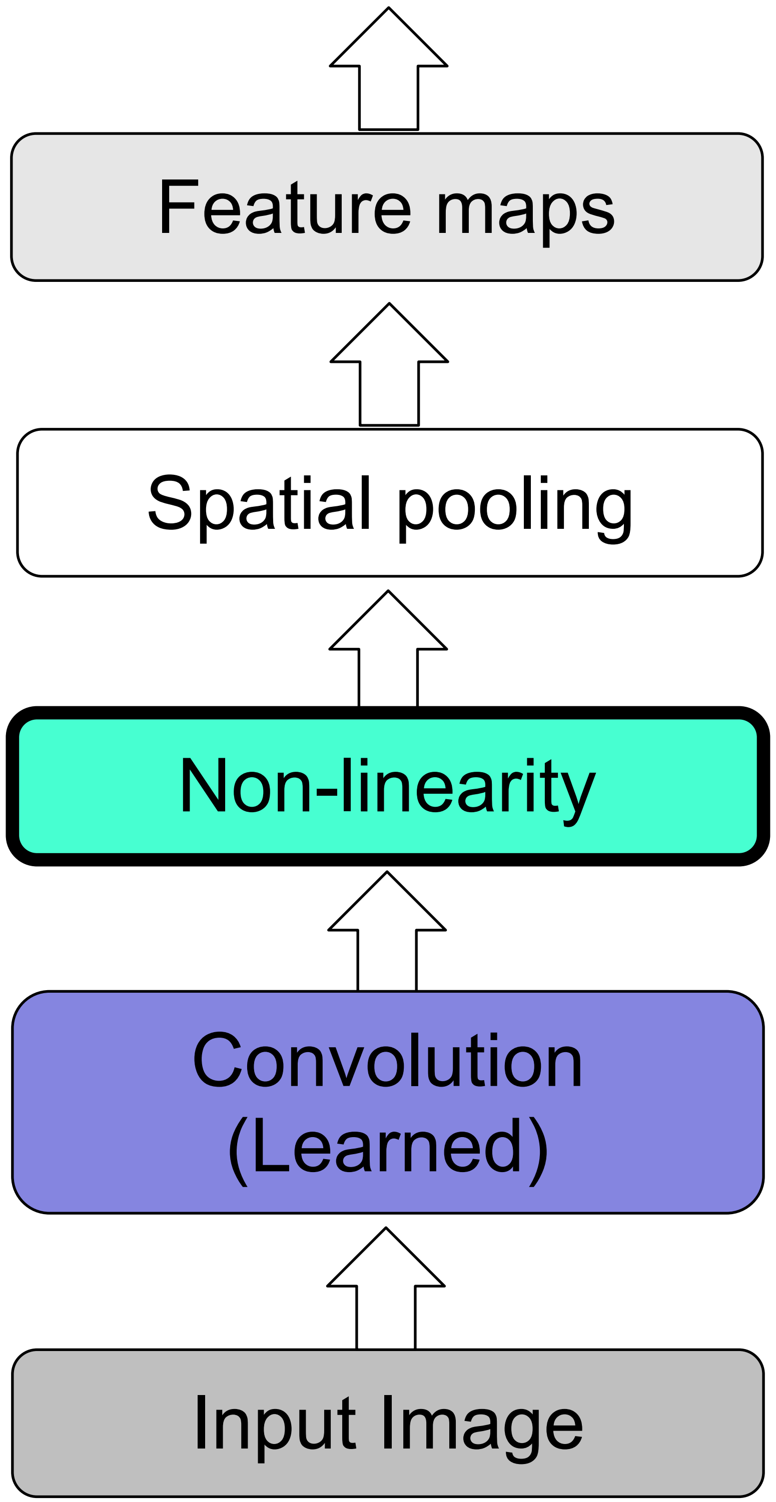


Input

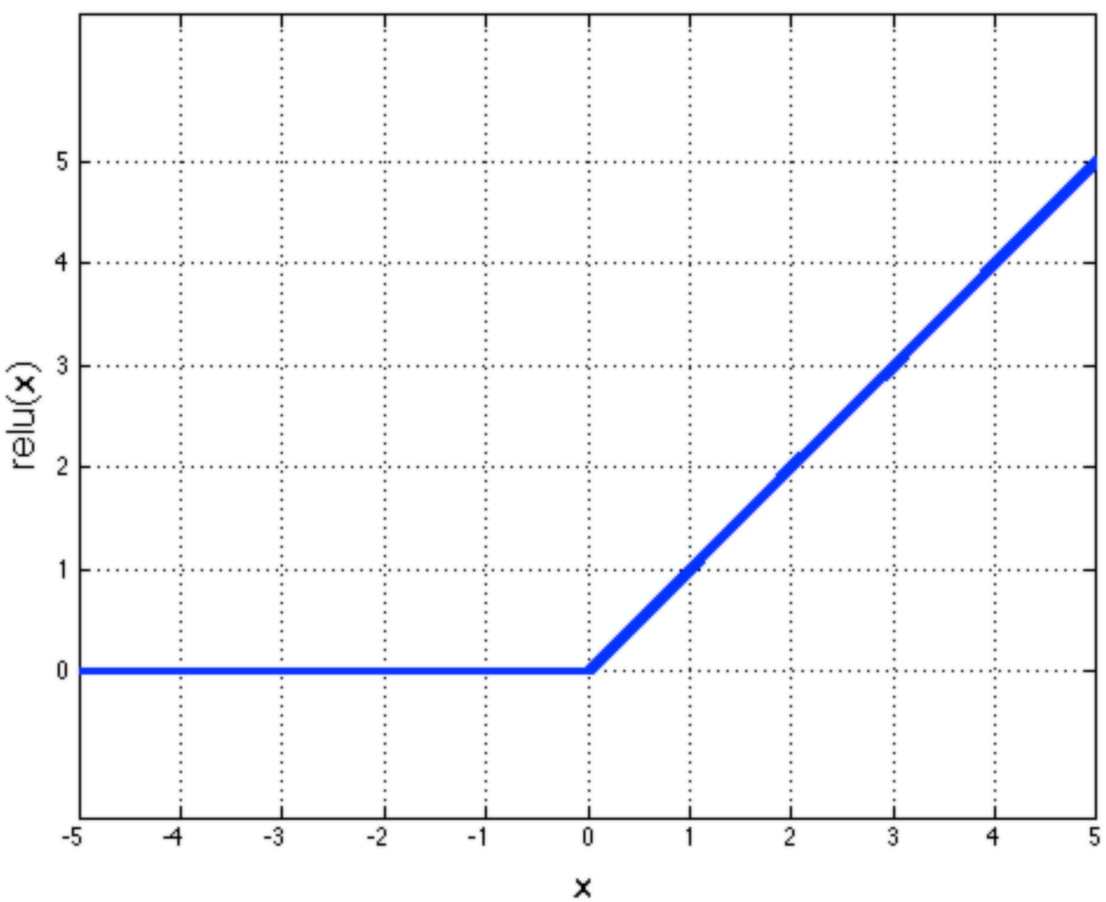
Feature Map

Source: R. Fergus, Y. LeCun

Key operations in a CNN

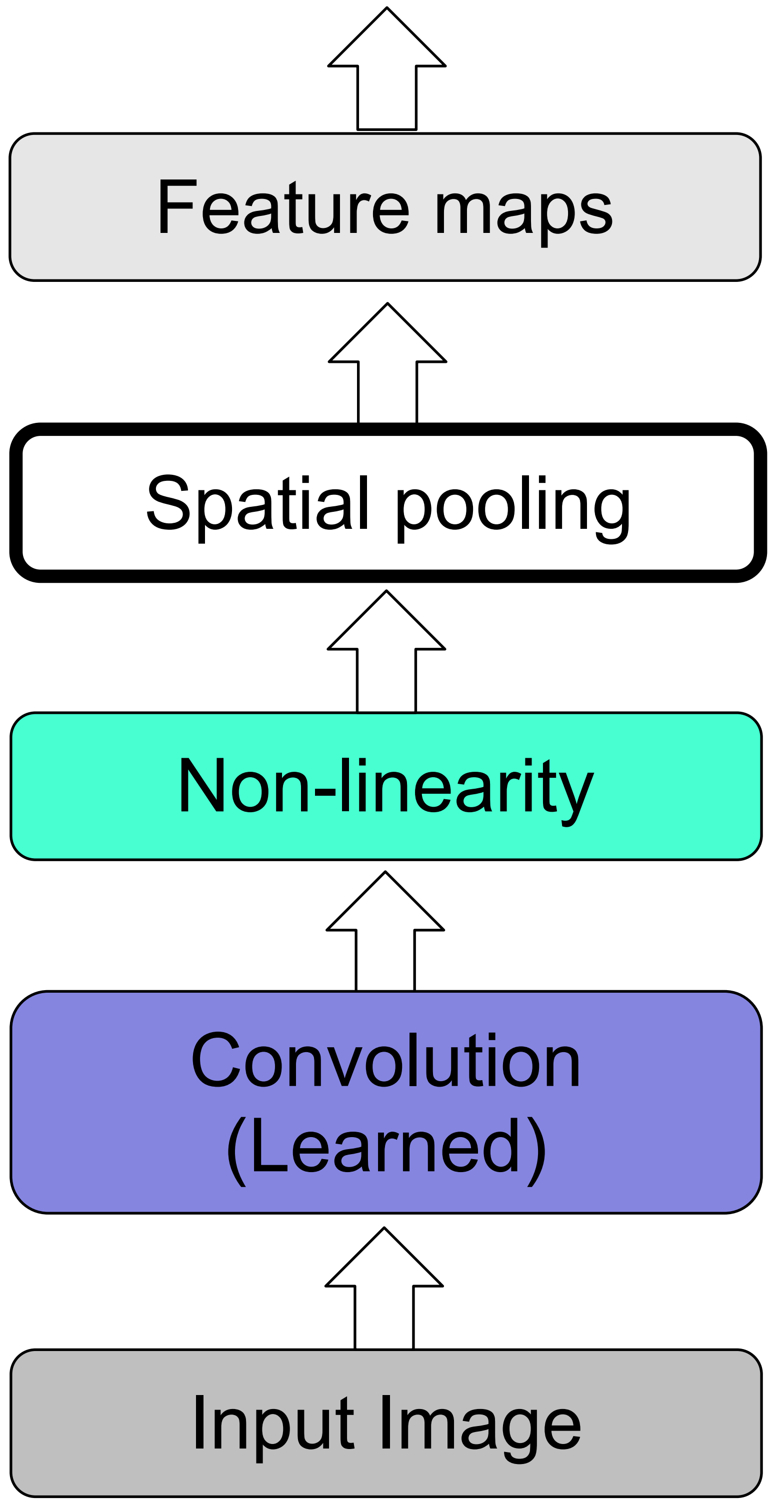


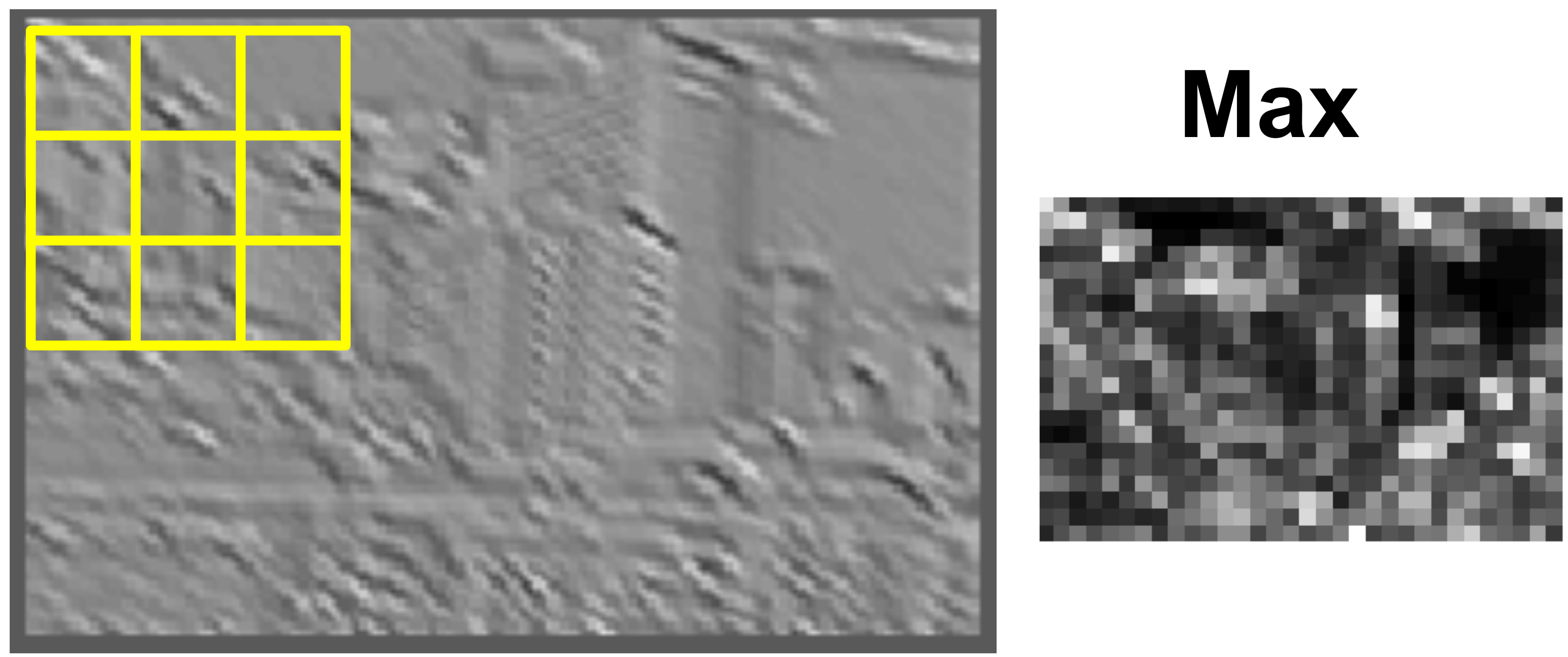
Rectified Linear Unit (ReLU)



Source: R. Fergus, Y. LeCun

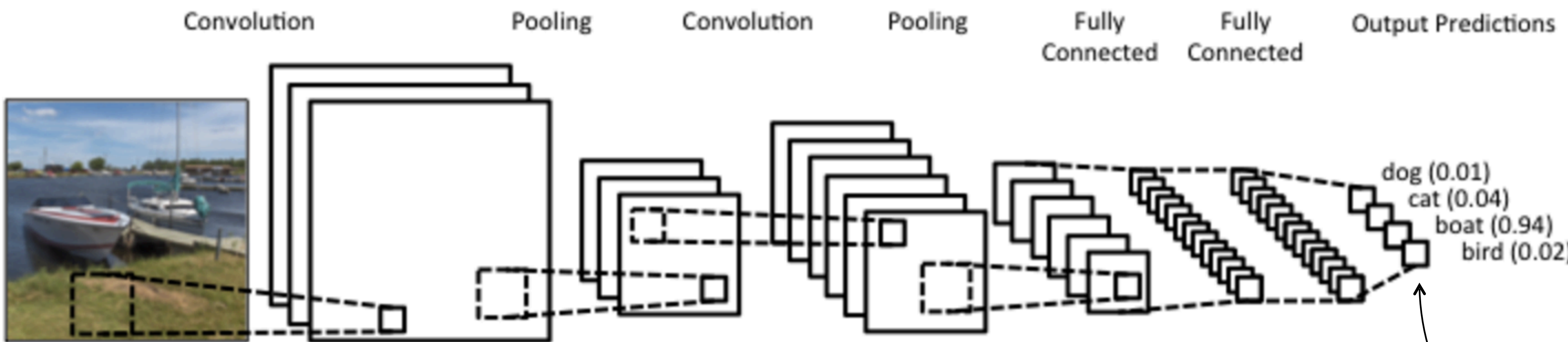
Key operations in a CNN





Source: R. Fergus, Y. LeCun

Key operations in a CNN

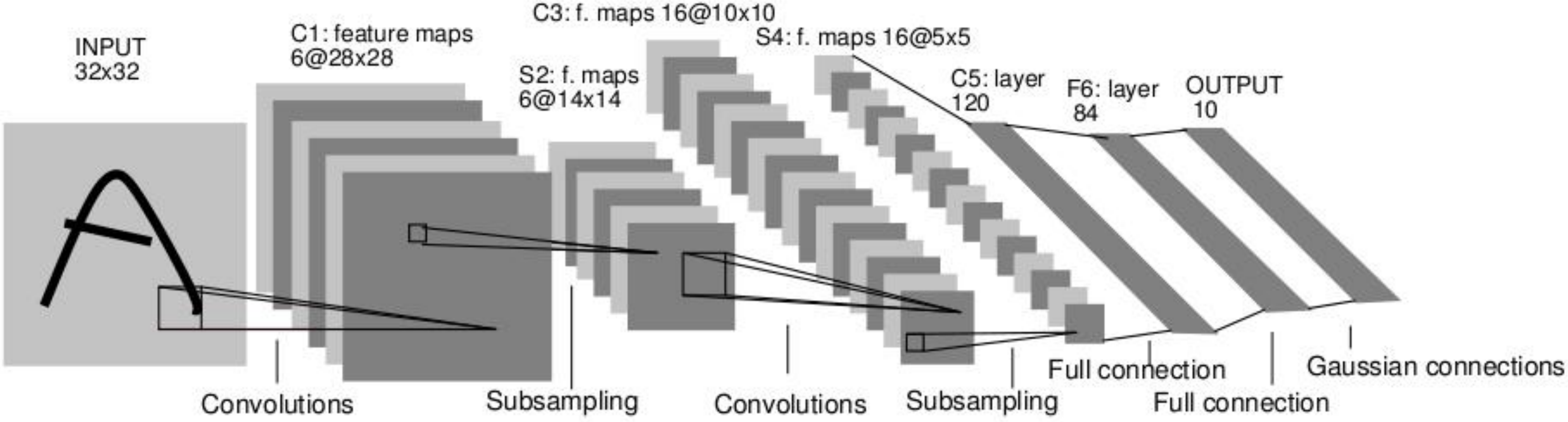


Softmax layer:

$$P(c \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_c \cdot \mathbf{x})}{\sum_{k=1}^{C} \exp(\mathbf{w}_k \cdot \mathbf{x})}$$

Source: S. Lazebnik

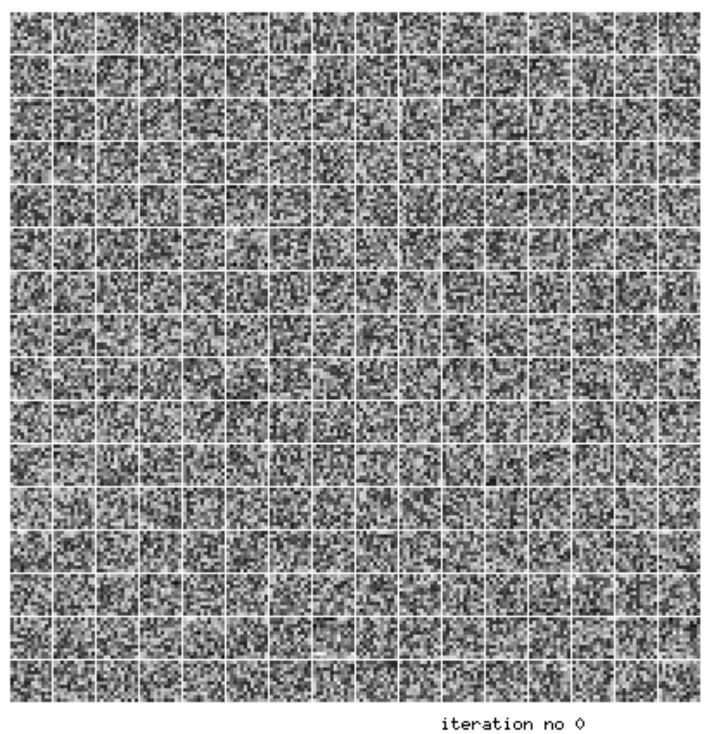
LeNet-5



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

LeNet: First layer

- Directly interpretable. E.g. LeNet 5 during training



Gif from Y. LeCun

Questions

- How to define the loss?
- How to minimize the loss?