RWTH AACHEN UNIVERSITY

Computer Vision – Lecture 10

Deep Learning

27.05.2019

Bastian Leibe

Visual Computing Institute
RWTH Aachen University
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

Course Outline

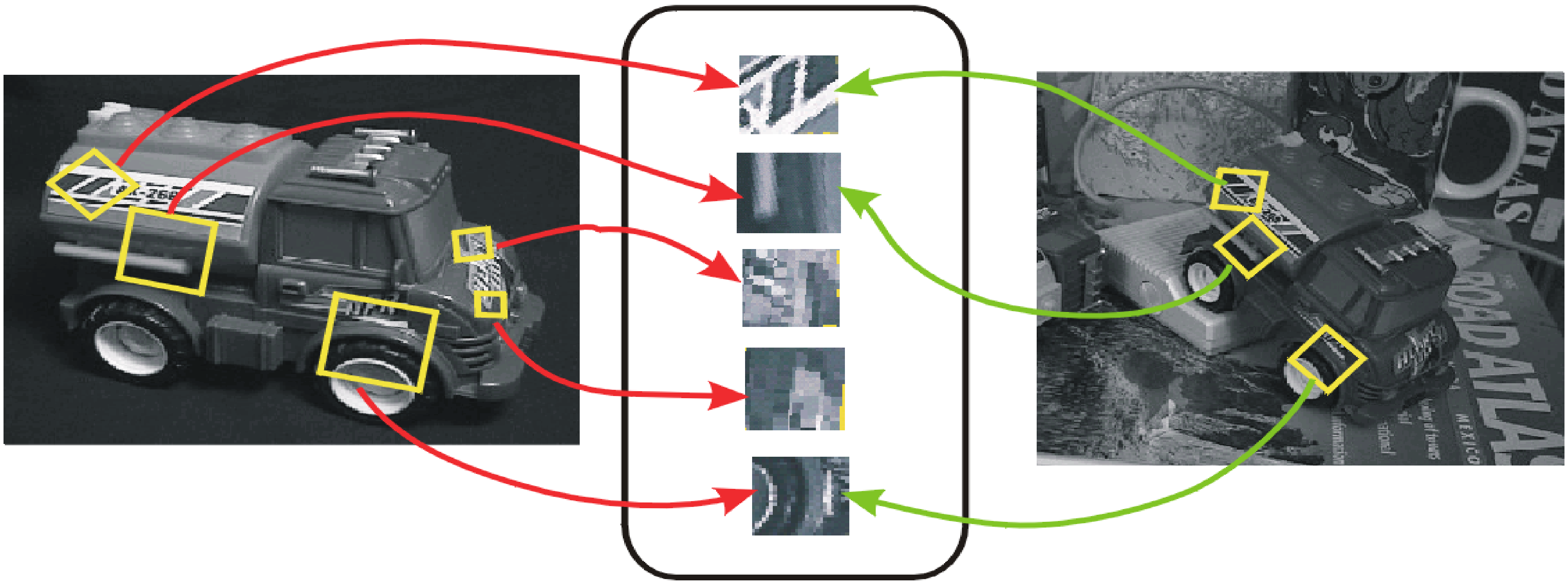
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition & Categorization
  - Sliding Window based Object Detection
- Local Features & Matching
  - Local Features – Detection and Description
  - Recognition with Local Features
- Deep Learning
- 3D Reconstruction

Topics of This Lecture

- Recap: Recognition with Local Features
- Dealing with Outliers
  - RANSAC
  - Generalized Hough Transform
- Deep Learning
  - Motivation
  - Neural Networks
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities

Recap: Recognition with Local Features

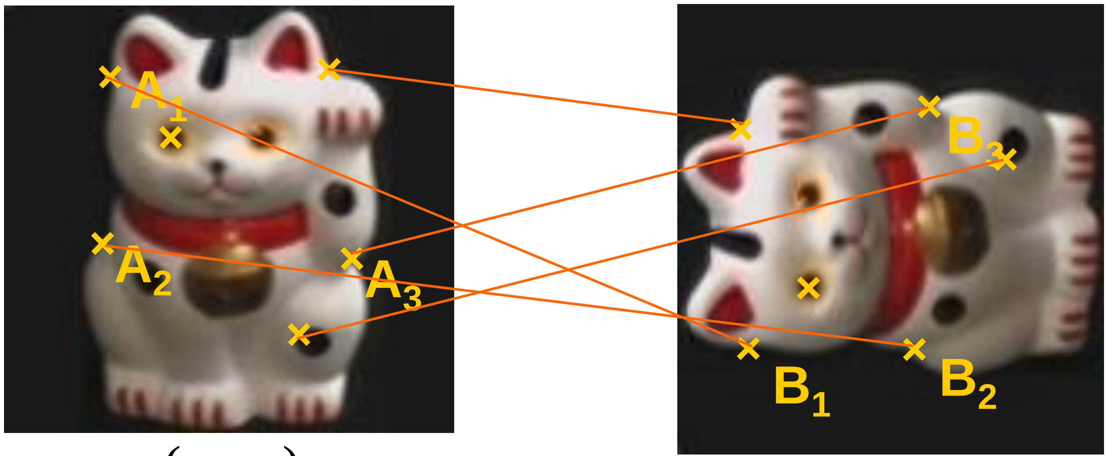
- Image content is transformed into local features that are invariant to translation, rotation, and scale
- Goal: Verify if they belong to a consistent configuration



Local Features,
e.g. SIFT

Recap: Fitting an Affine Transformation

- Assuming we know the correspondences, how do we get the transformation?



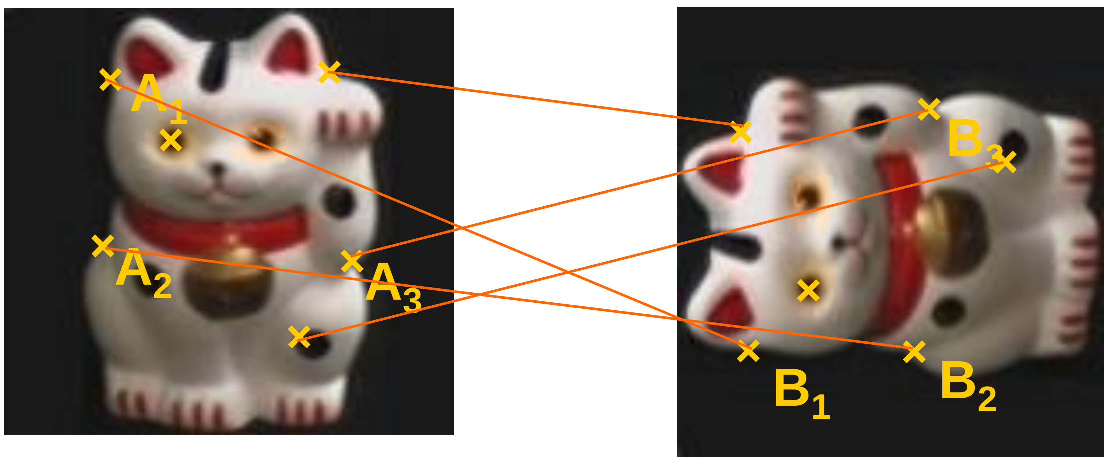
(x_i, y_i) (x_i', y_i')

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Recap: Fitting a Homography

- Estimating the transformation



Homogenous coordinates

Image coordinates

Matrix notation

$$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$$
$$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$$
$$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$$
$$\vdots$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \frac{1}{z'} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

$$x' = Hx$$

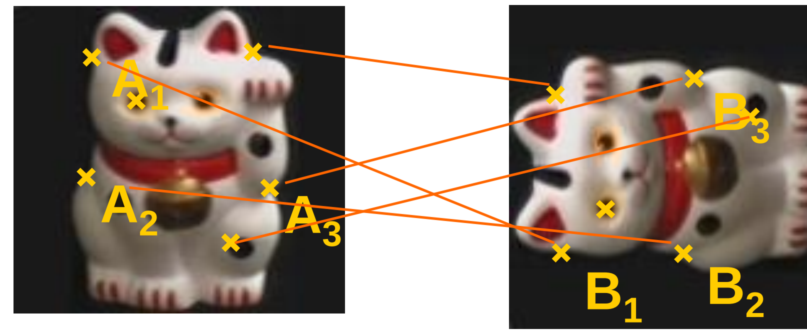
$$x'' = \frac{1}{z'} x'$$

$$x_{A_1} = \frac{h_{11}\, x_{B_1} + h_{12} y_{B_1} + h_{13}}{h_{31}\, x_{B_1} + h_{32} y_{B_1} + 1} \qquad y_{A_1} = \frac{h_{21}\, x_{B_1} + h_{22} y_{B_1} + h_{23}}{h_{31}\, x_{B_1} + h_{32} y_{B_1} + 1}$$

Slide credit: Krystian Mikolajczyk

Recap: Fitting a Homography

- Estimating the transformation



$$h_{11}\, x_{B_1} + h_{12} y_{B_1} + h_{13} - x_{A_1} h_{31}\, x_{B_1} - x_{A_1} h_{32} y_{B_1} - x_{A_1} = 0$$

$$h_{21}\, x_{B_1} + h_{22} y_{B_1} + h_{23} - y_{A_1} h_{31}\, x_{B_1} - y_{A_1} h_{32} y_{B_1} - y_{A_1} = 0$$

$$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$$
$$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$$
$$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$$
$$\vdots$$

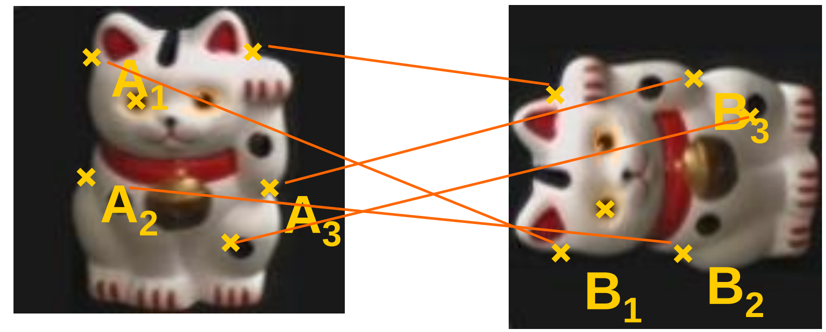
$$\begin{bmatrix} x_{B_1} & y_{B_1} & 1 & 0 & 0 & 0 & -x_{A_1} x_{B_1} & -x_{A_1} y_{B_1} & -x_{A_1} \\ 0 & 0 & 0 & x_{B_1} & y_{B_1} & 1 & -y_{A_1} x_{B_1} & -y_{A_1} y_{B_1} & -y_{A_1} \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \end{bmatrix}$$

$$Ah = 0$$

RWTH AACHEN UNIVERSITY

Recap: Fitting a Homography

- Estimating the transformation
- Solution:
  - Null-space vector of A
  - Corresponds to smallest eigenvector



$$Ah = 0$$

SVD

$$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$$
$$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$$
$$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$$
$$\vdots$$

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T = \mathbf{U}\begin{bmatrix} d_{11} & \cdots & d_{19} \\ \vdots & \ddots & \vdots \\ d_{91} & \cdots & d_{99} \end{bmatrix}\begin{bmatrix} v_{11} & \cdots & v_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \cdots & v_{99} \end{bmatrix}^T$$

$$\mathbf{h} = \frac{[v_{19}, \cdots, v_{99}]}{v_{99}}$$

Minimizes least square error

Slide credit: Krystian Mikolajczyk

Recap: A General Point

- Equations of the form

$$\mathrm{A}\mathrm{x} = 0$$

Think of this as an eigenvector equation
$\mathbf{Ax} = \lambda \mathbf{x}$
for the special case of $\lambda = 0$.

- How do we solve them? (always!)
  - Apply SVD

SVD is the generalization of the eigenvector decomposition for non-square matrices $\mathbf{A}$.

SVD

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = \boldsymbol{U}\begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{NN} \end{bmatrix}\begin{bmatrix} v_{11} & \cdots & v_{1N} \\ \vdots & \ddots & \vdots \\ v_{N1} & \cdots & v_{NN} \end{bmatrix}^T$$

Singular values   Singular vectors

  - Singular values of A = square roots of the eigenvalues of A^TA.
  - The solution of Ax=0 is the *nullspace* vector of A.
  - This corresponds to the *smallest singular vector* of A.

Recap: Object Recognition by Alignment

- Assumption
  - Known object, rigid transformation compared to model image

  ⇒ *If we can find evidence for such a transformation, we have recognized the object.*

- You learned methods for
  - Fitting an *affine transformation* from ≥ 3 correspondences
  - Fitting a *homography* from ≥ 4 correspondences

Affine: solve a system

$$At = b$$

Homography: solve a system

$$Ah = 0$$

- Correspondences may be noisy and may contain outliers

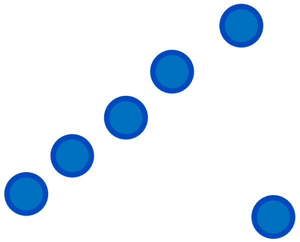
  ⇒ Need to use robust methods that can filter out outliers

Topics of This Lecture

- Recap: Recognition with Local Features
- Dealing with Outliers
  - RANSAC
  - Generalized Hough Transform
- Deep Learning
  - Motivation
  - Neural Networks
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities

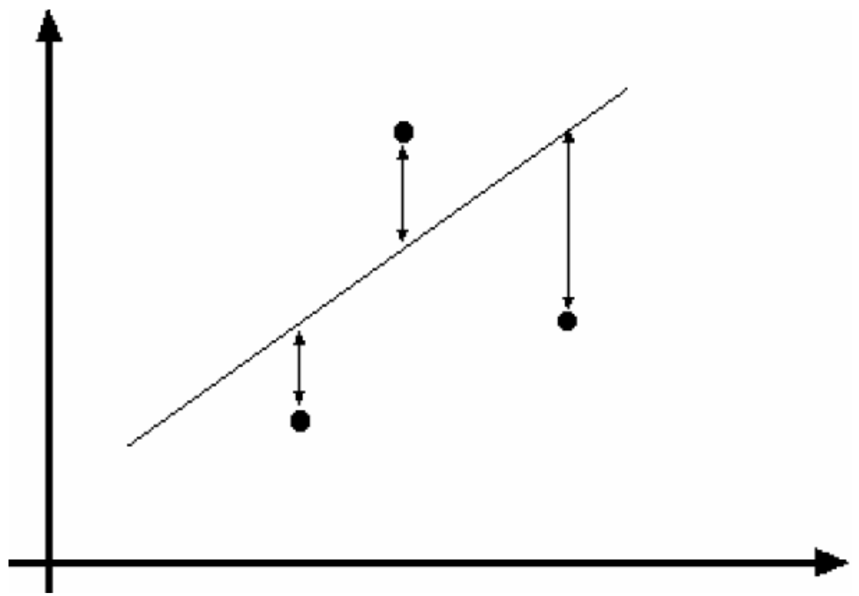
Problem: Outliers

- Outliers can hurt the quality of our parameter estimates, e.g.,
  - An erroneous pair of matching points from two images
  - A feature point that is noise or doesn't belong to the transformation we are fitting.



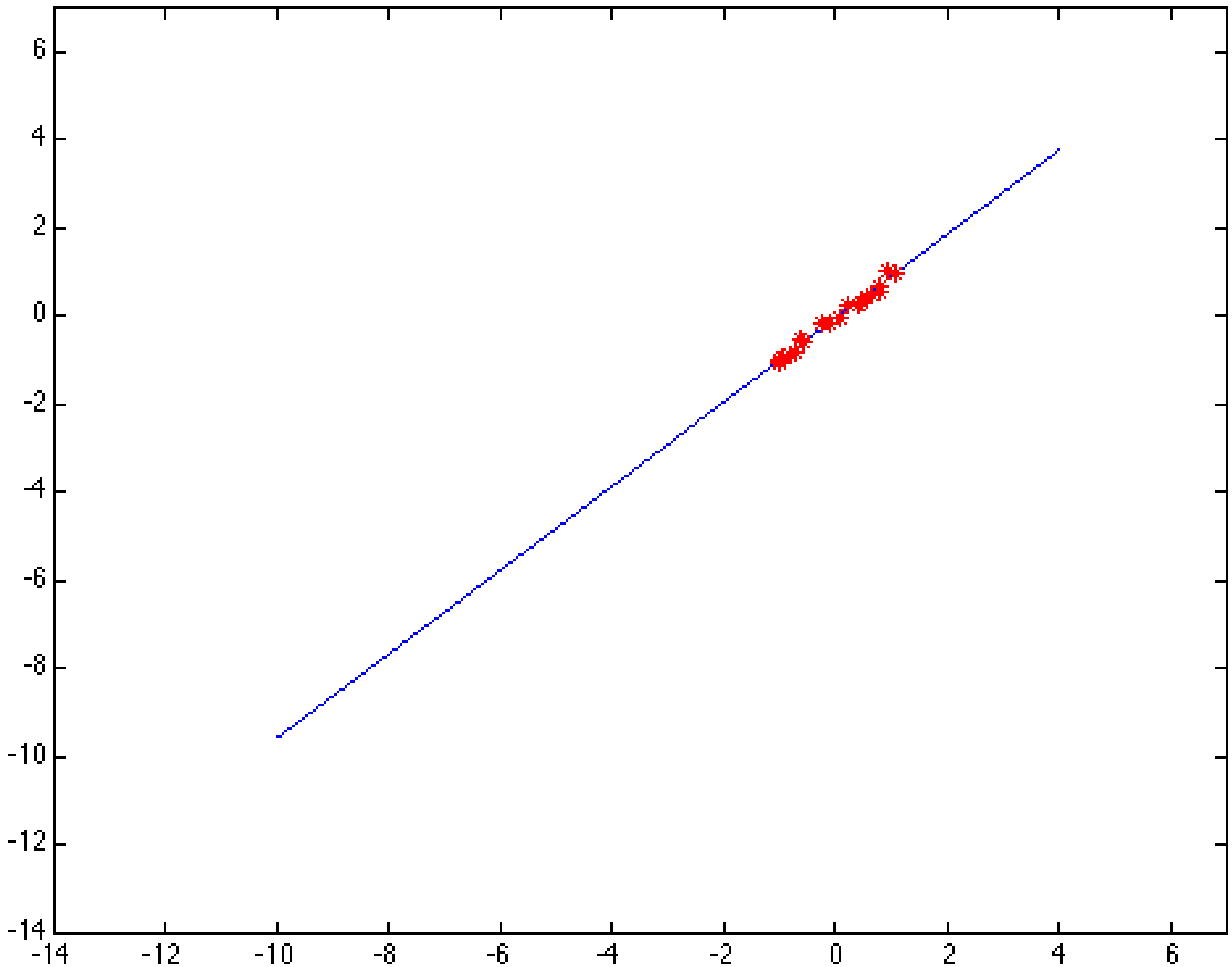
Example: Least-Squares Line Fitting

- Assuming all the points that belong to a particular line are known



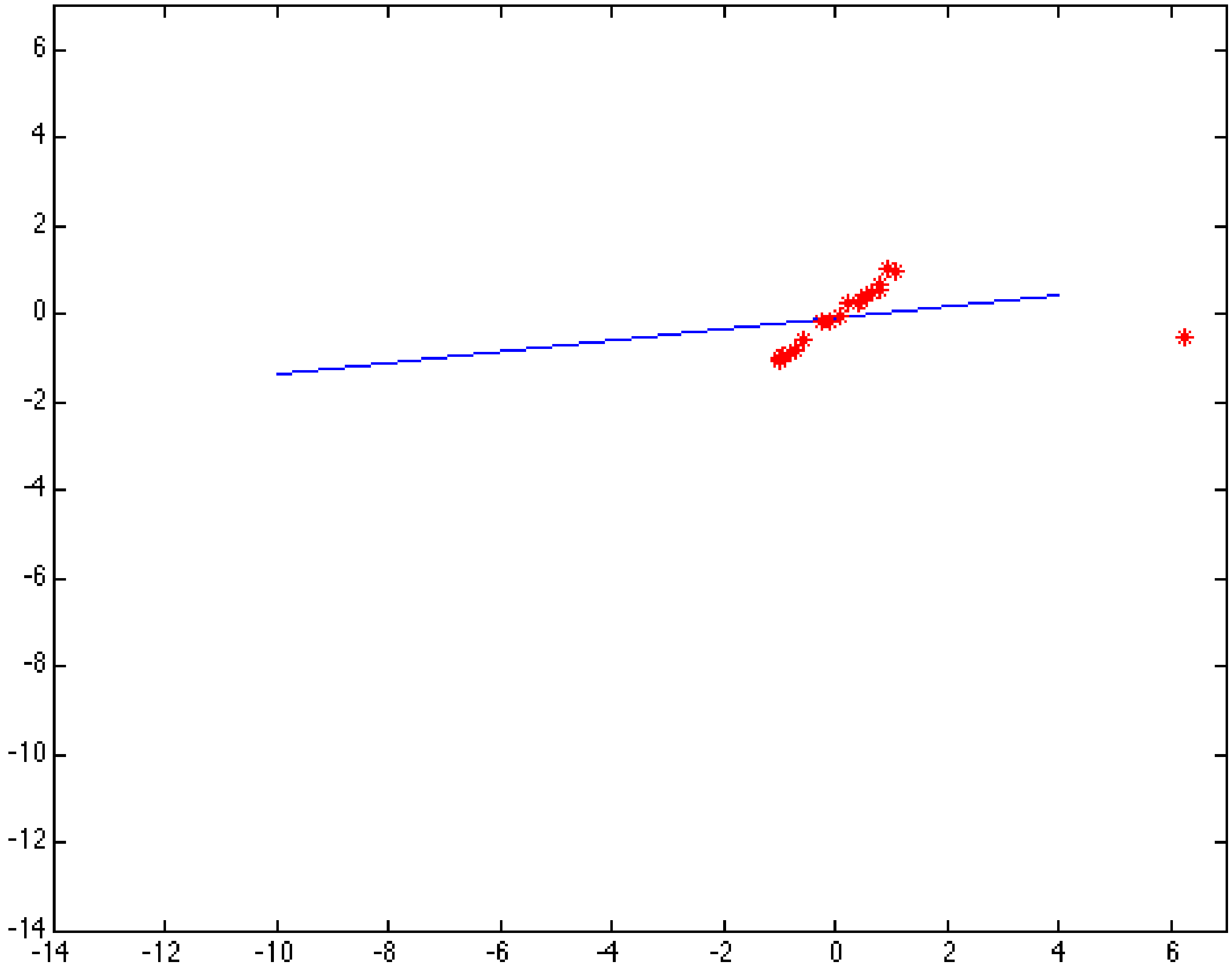
Source: Forsyth & Ponce

Outliers Affect Least-Squares Fit



Source: Forsyth & Ponce

Outliers Affect Least-Squares Fit



Source: Forsyth & Ponce

Strategy 1: RANSAC [Fischler81]

- RANdom SAmple Consensus
- Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use only those.
- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

RANSAC

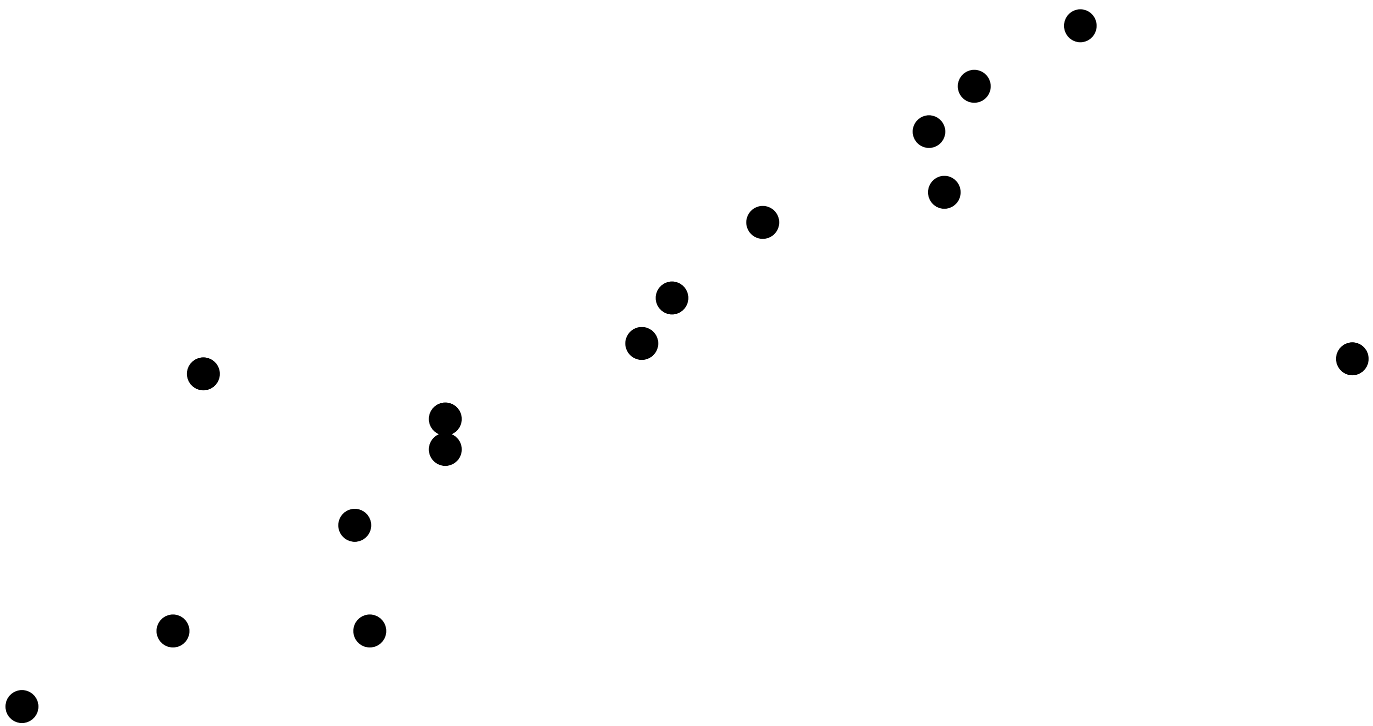
RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

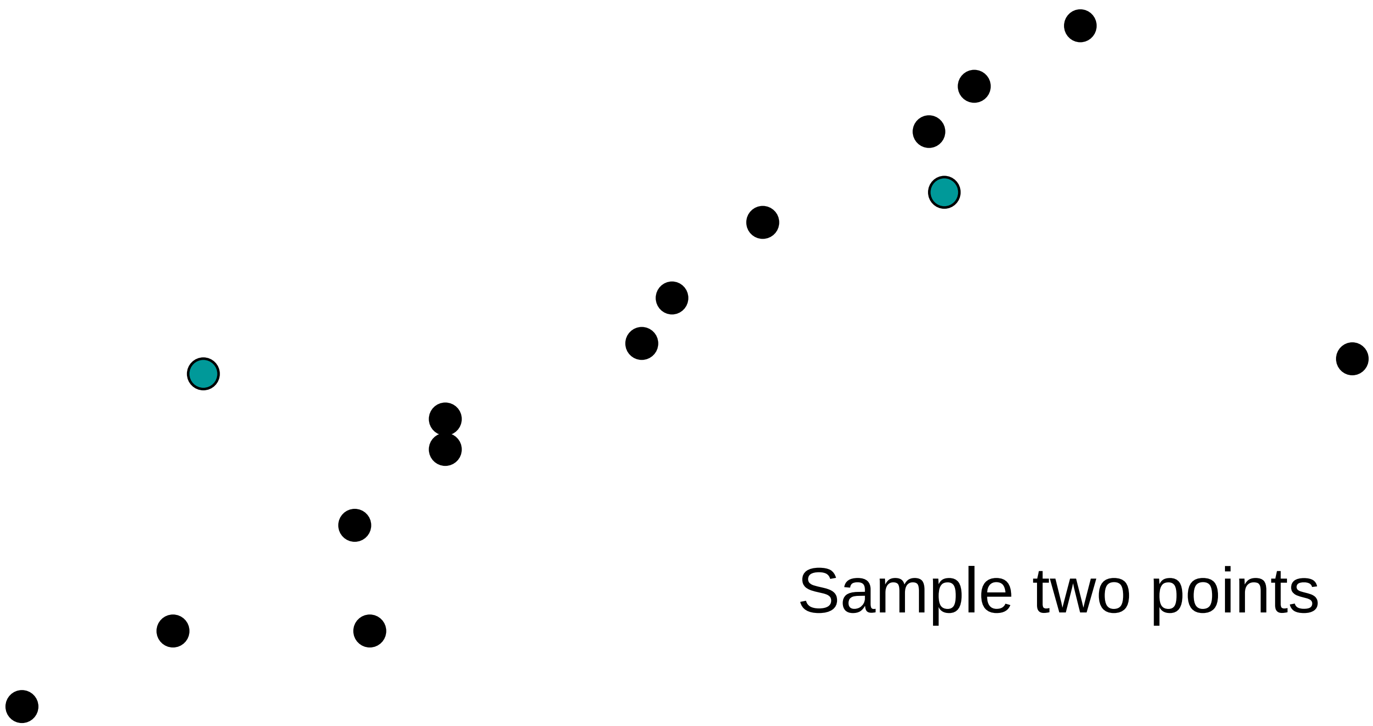
RANSAC Line Fitting Example

- Task: Estimate the best line
  - *How many points do we need to estimate the line?*



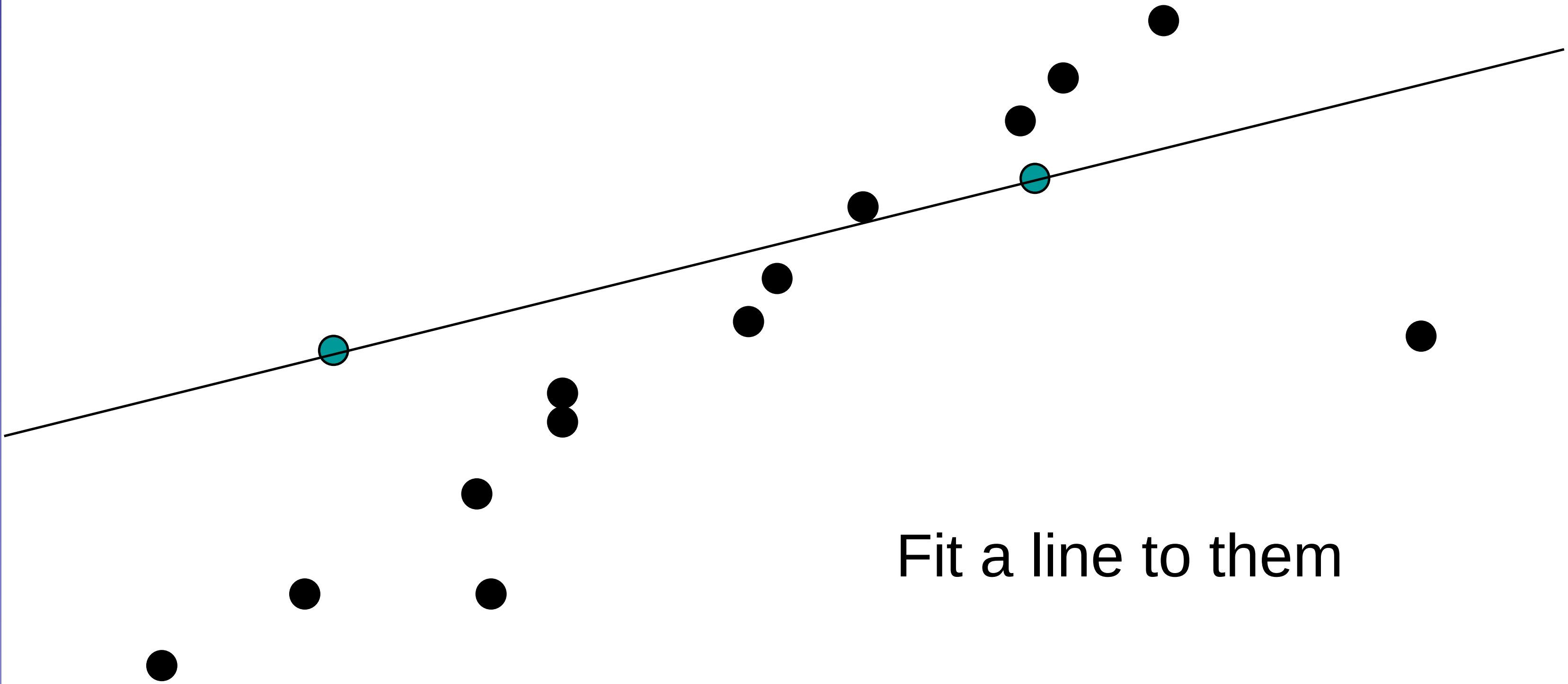
RANSAC Line Fitting Example

- Task: Estimate the best line



RANSAC Line Fitting Example

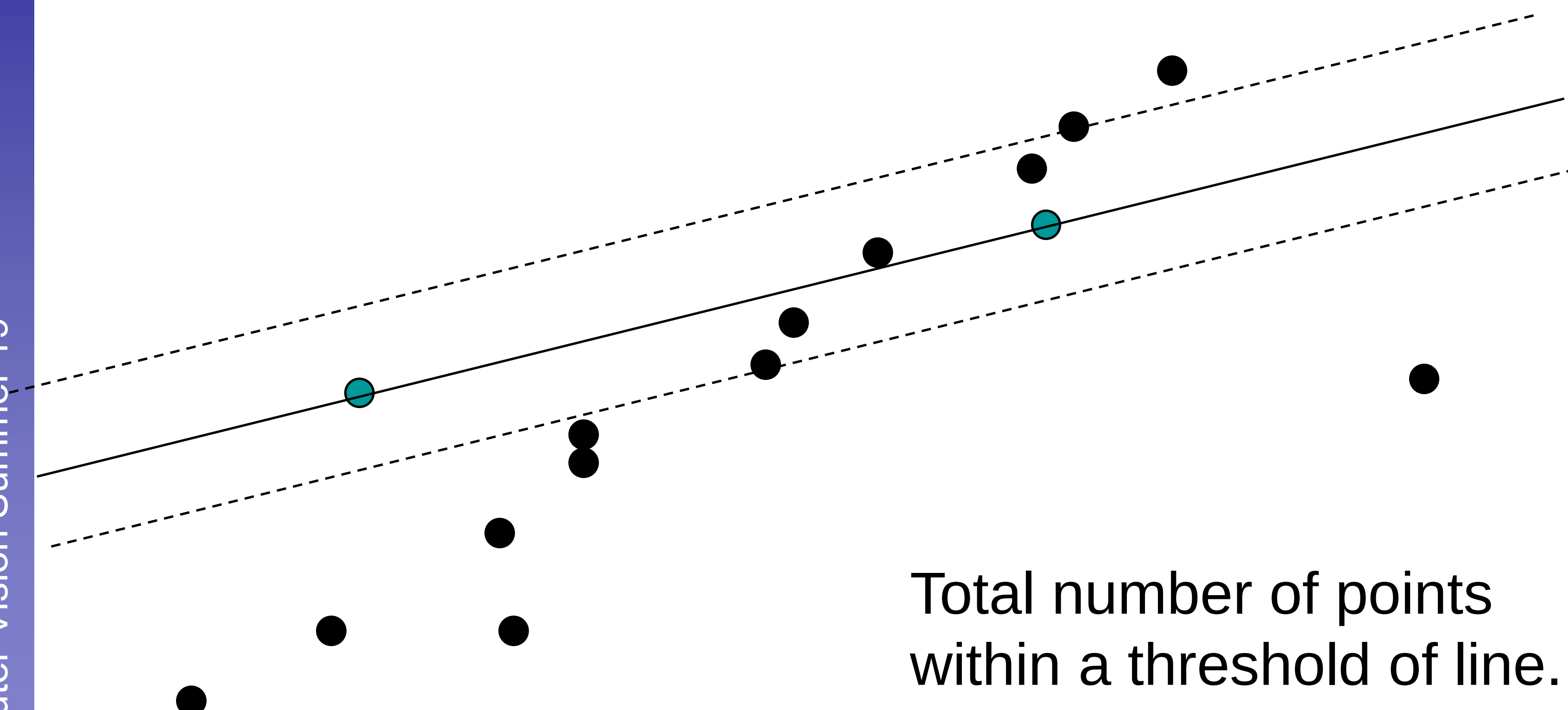
- Task: Estimate the best line



Fit a line to them

RANSAC Line Fitting Example

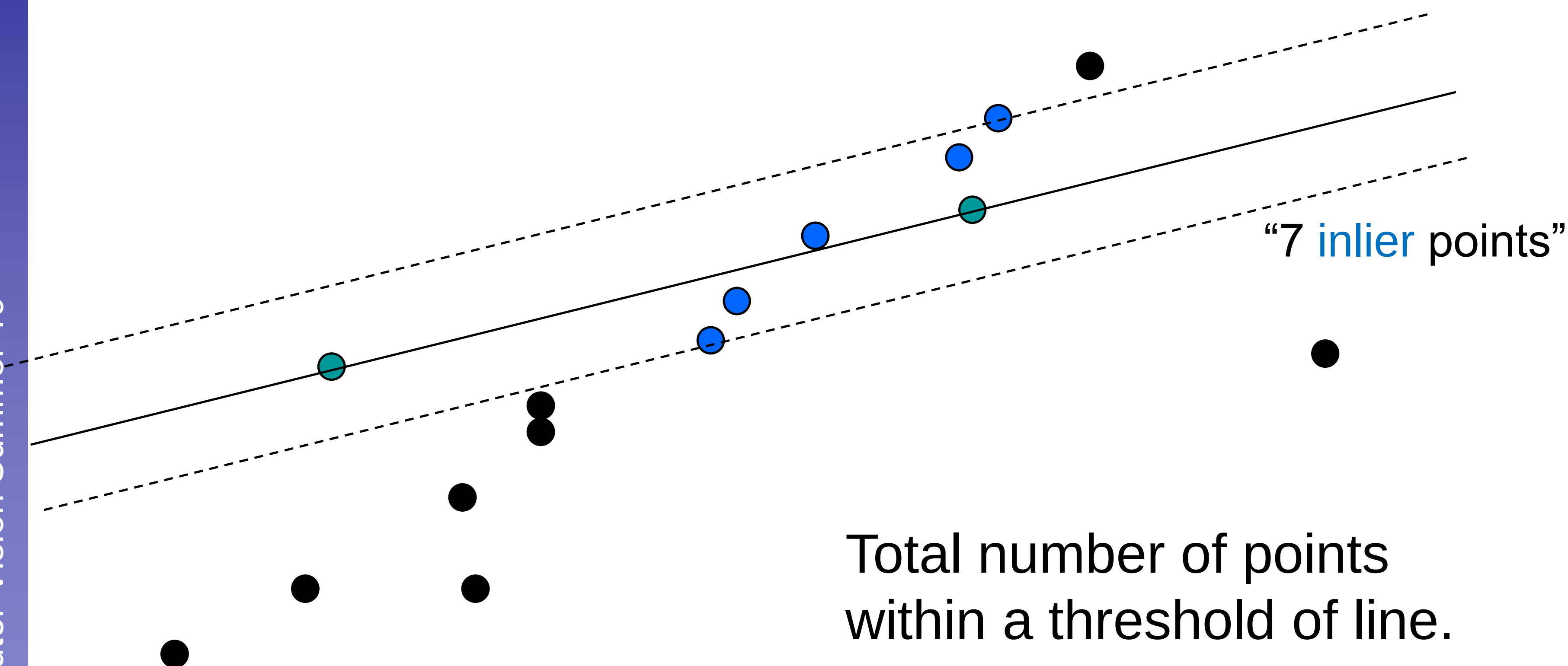
- Task: Estimate the best line



Total number of points within a threshold of line.

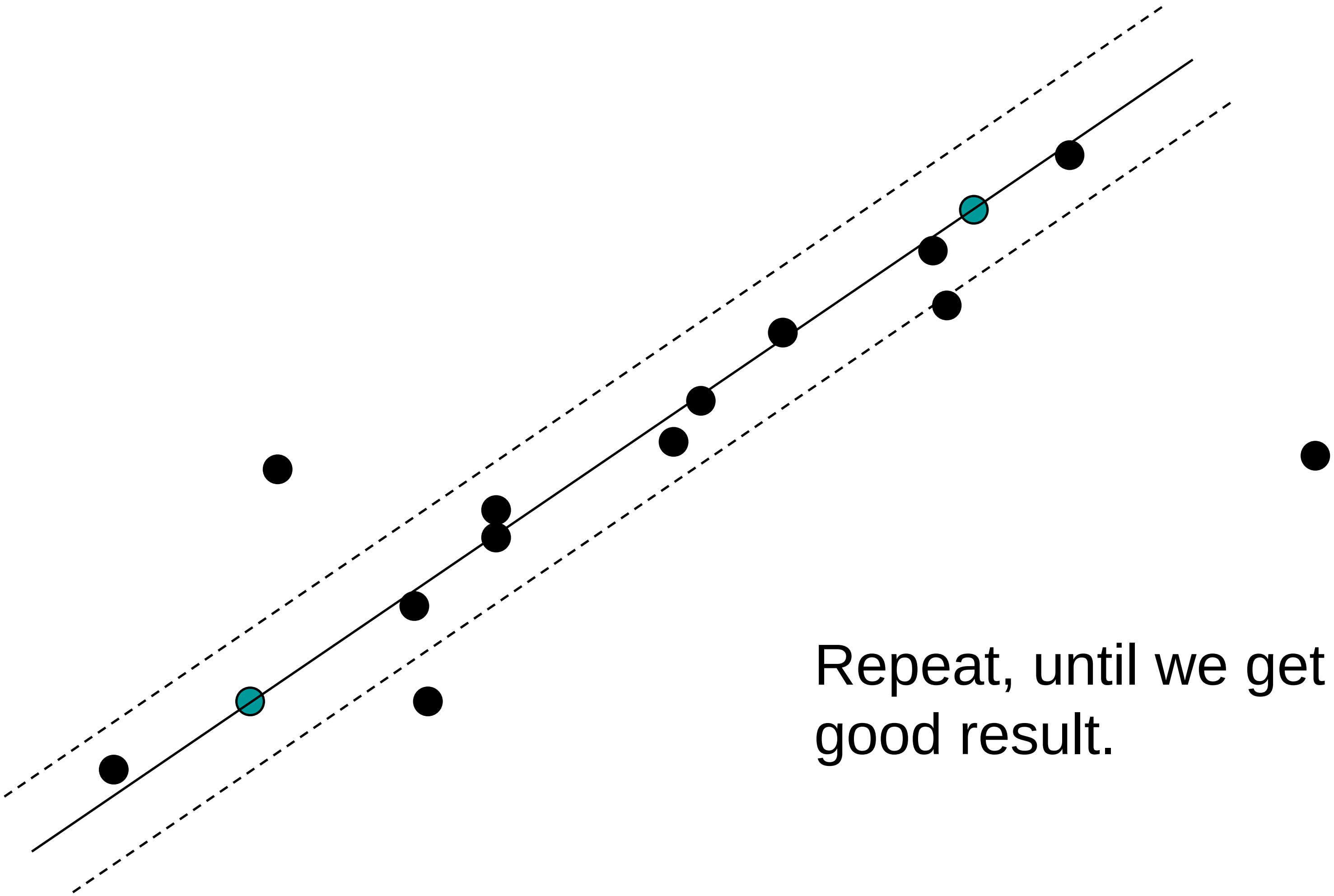
RANSAC Line Fitting Example

- Task: Estimate the best line



RANSAC Line Fitting Example

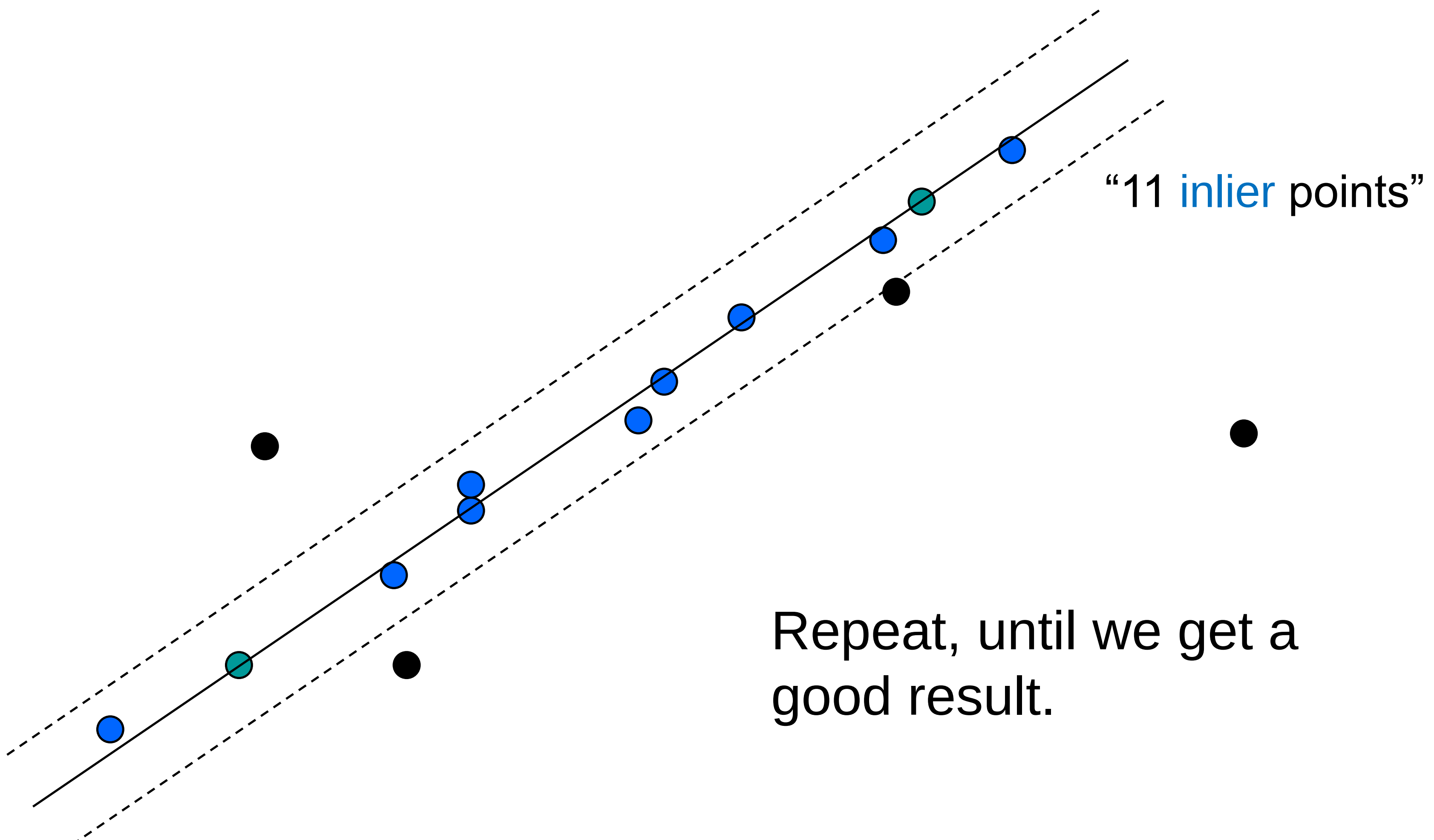
- Task: Estimate the best line



Repeat, until we get a good result.

RANSAC Line Fitting Example

- Task: Estimate the best line



Repeat, until we get a good result.

RANSAC: How many samples?

- How many samples are needed?
  - Suppose w is fraction of inliers (points from line).
  - n points needed to define hypothesis (2 for lines)
  - k samples chosen.
- Prob. that a single sample of n points is correct: w^n
- Prob. that all k samples fail is: $(1-w^n)^k$

$\Rightarrow$ Choose k high enough to keep this below the desired failure rate.

Slide credit: David Lowe

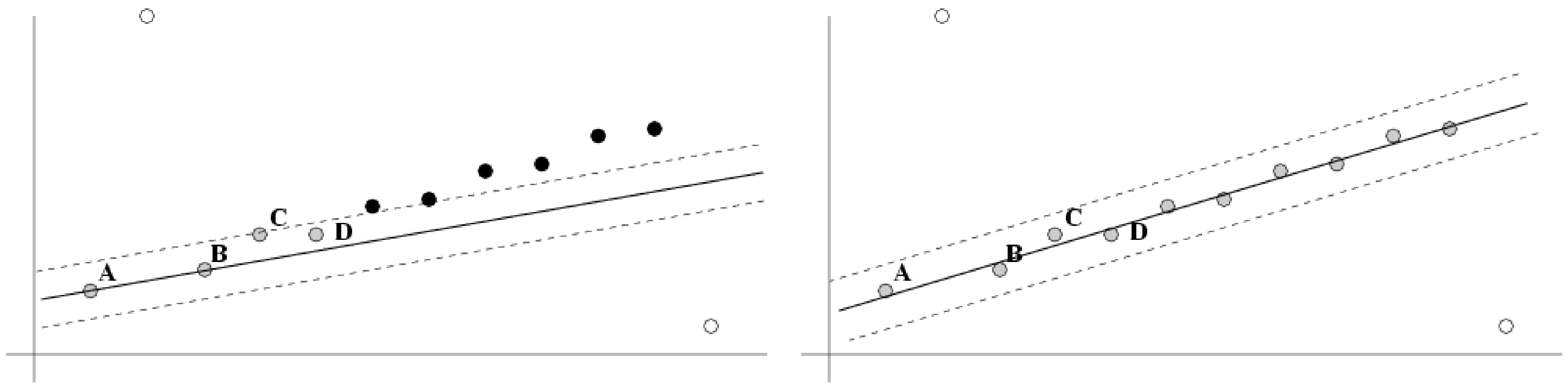
RANSAC: Computed k (p=0.99)

Sample size n	Proportion of outliers 5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Slide credit: David Lowe

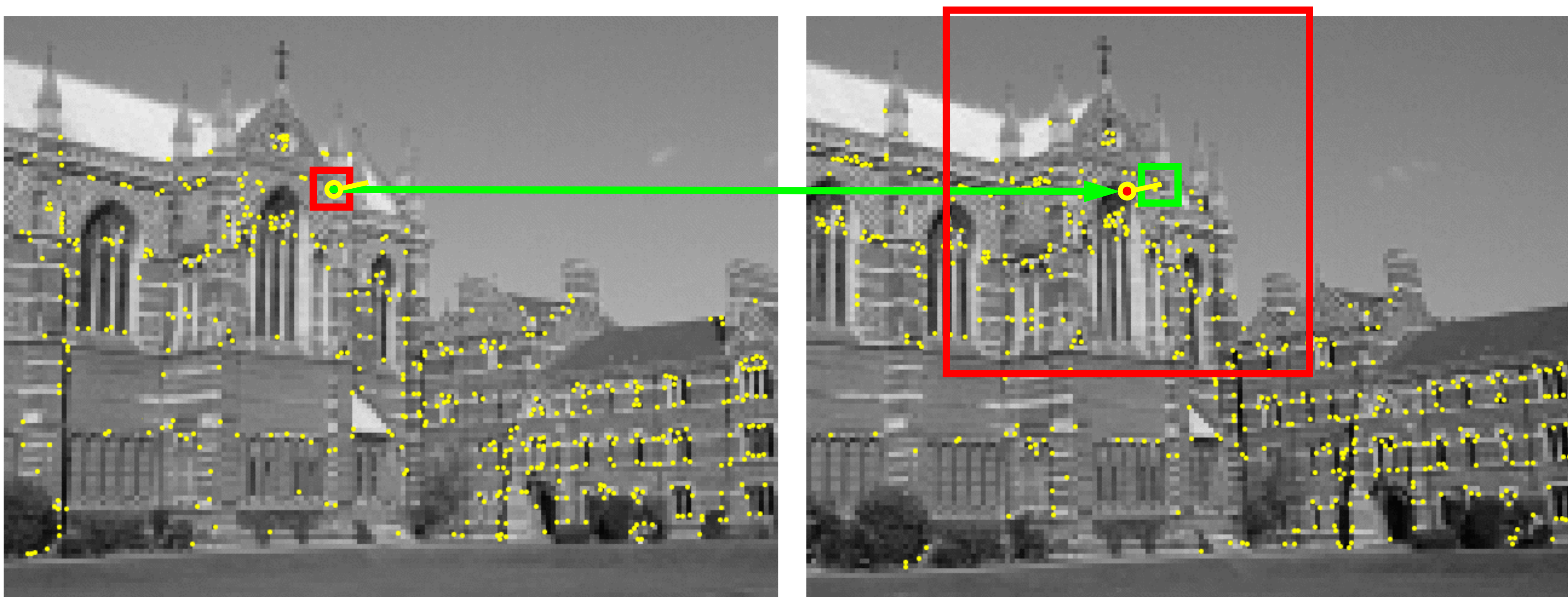
After RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.



Example: Finding Feature Matches

- Find best stereo match within a square search window (here 300 pixels^2)
- Global transformation model: epipolar geometry

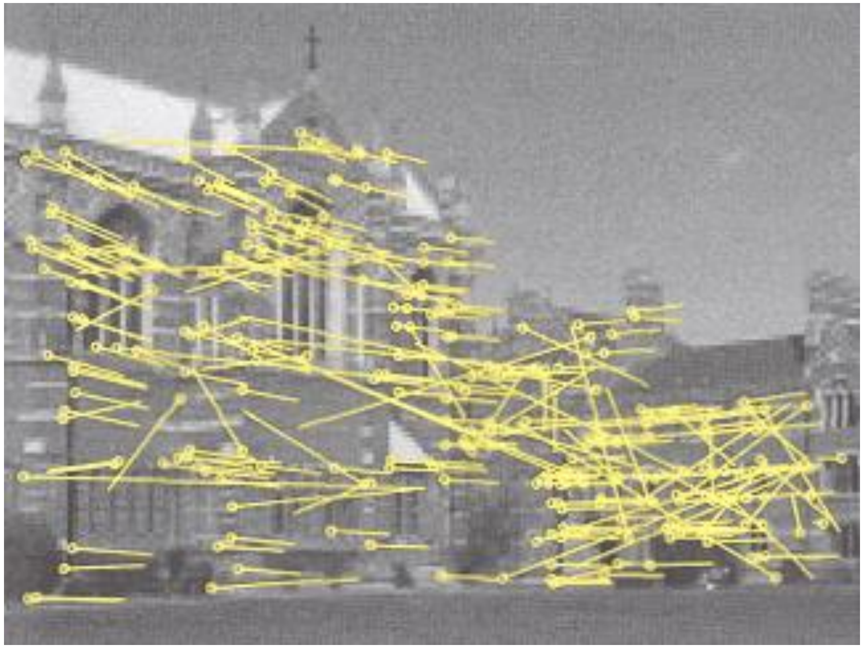


Images from Hartley & Zisserman

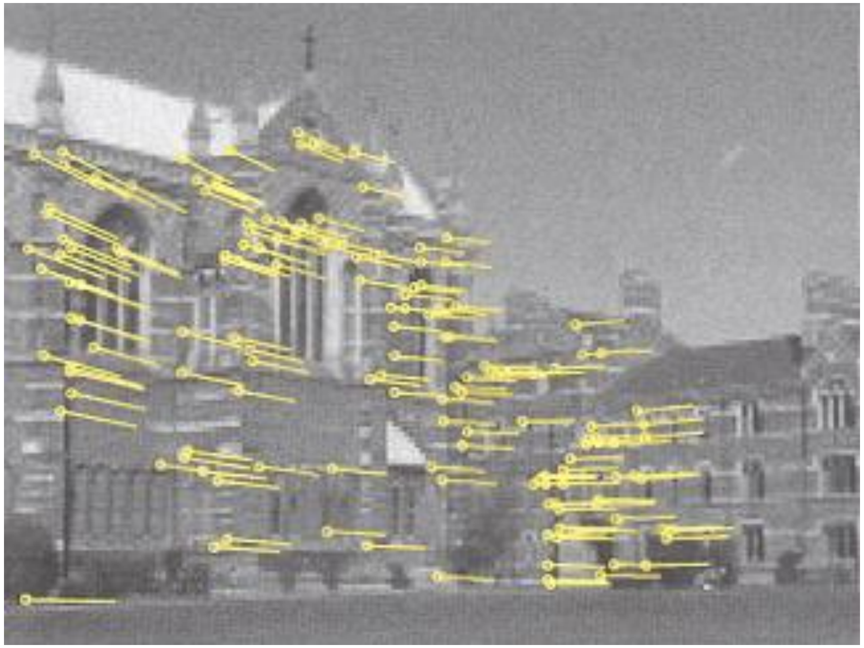
Example: Finding Feature Matches

- Find best stereo match within a square search window (here 300 pixels^2)
- Global transformation model: epipolar geometry

before RANSAC



after RANSAC



Images from Hartley & Zisserman

Slide credit: David Lowe

Problem with RANSAC

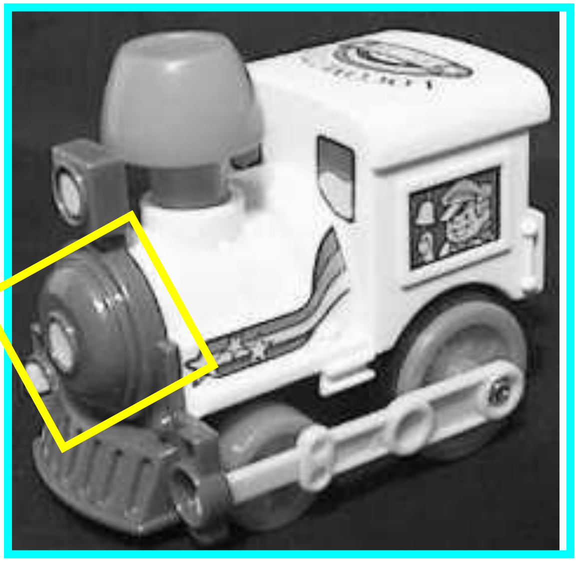
- In many practical situations, the percentage of outliers (incorrect putative matches) is often very high (90% or above).
- Alternative strategy: Generalized Hough Transform

Slide credit: Svetlana Lazebnik

Strategy 2: Generalized Hough Transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation).

model

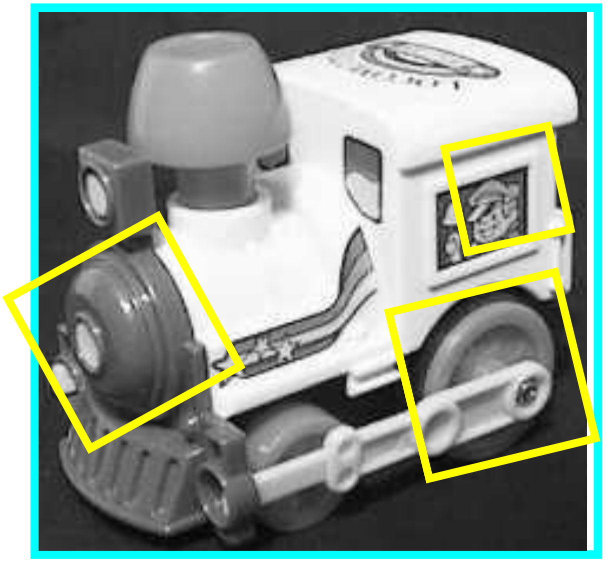


Slide credit: Svetlana Lazebnik

Strategy 2: Generalized Hough Transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation).
  - Of course, a hypothesis from a single match is unreliable.
  - Solution: let each match vote for its hypothesis in a Hough space with very coarse bins.

model

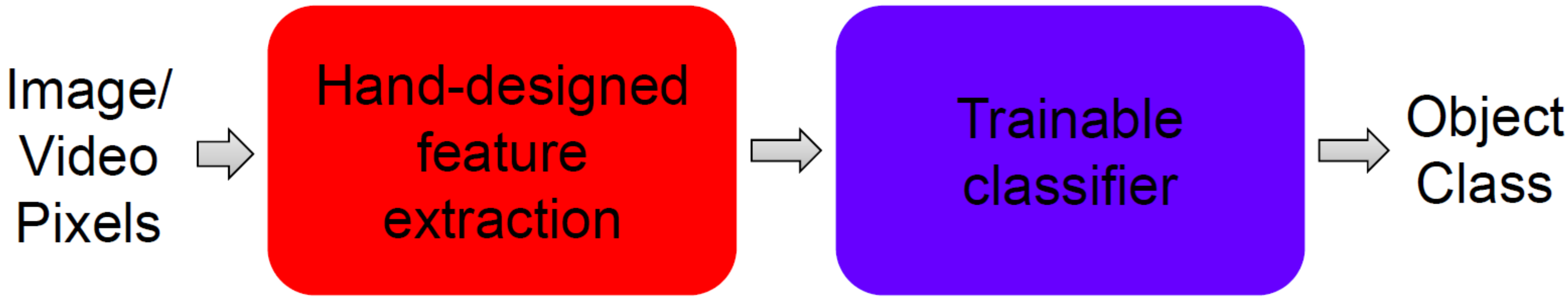


Topics of This Lecture

- Recap: Recognition with Local Features
- Dealing with Outliers
  - RANSAC
  - Generalized Hough Transform
- Deep Learning
  - Motivation
  - Neural Networks
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities

We've finally got there!

Traditional Recognition Approach



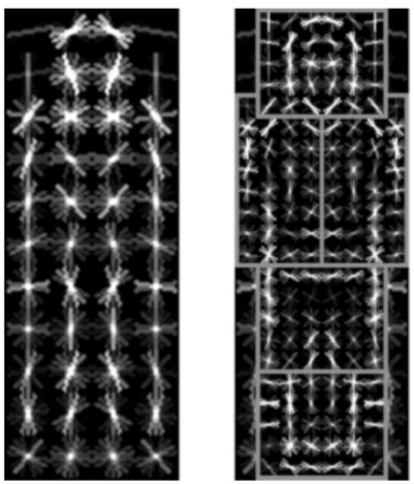
- Characteristics
  - Features are not learned, but engineered
  - Trainable classifier is often generic (e.g., SVM)

  $\Rightarrow$ Many successes in 2000-2010.

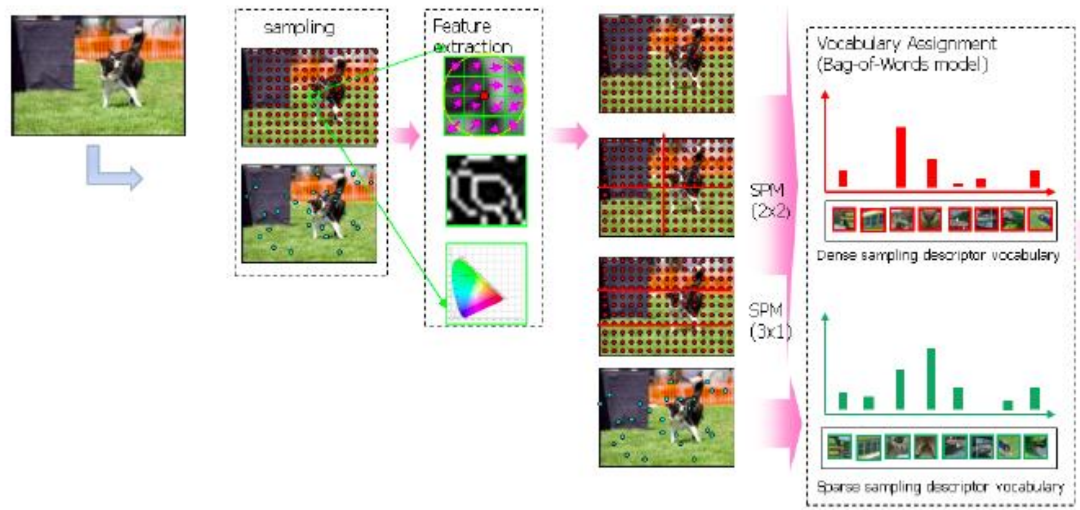
Traditional Recognition Approach

- Features are key to recent progress in recognition
  - Multitude of hand-designed features currently in use
  - SIFT, HOG, …………

  ⇒ *Where next? Better classifiers? Or keep building more features?*



DPM
[Felzenszwalb
et al., PAMI'07]



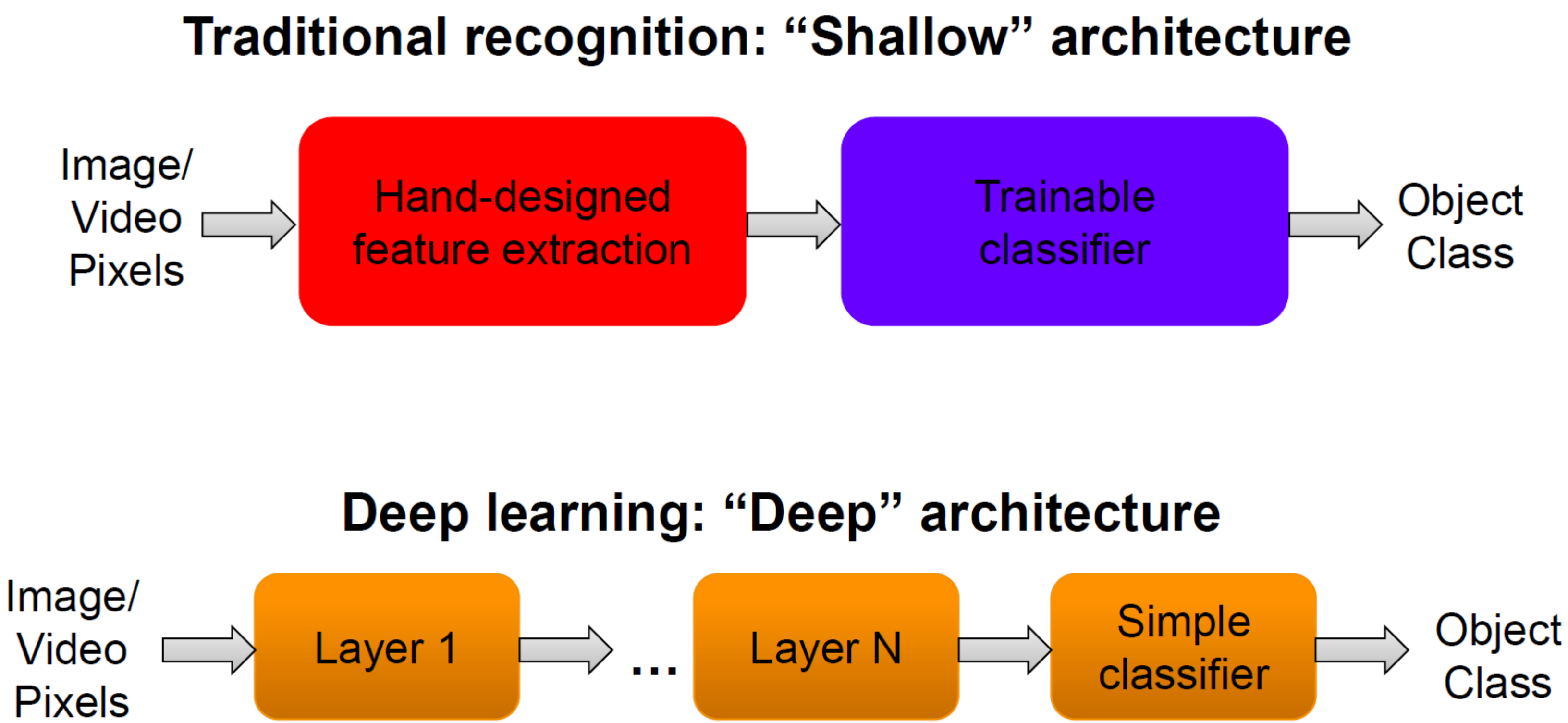
Dense SIFT+LBP+HOG → BOW → Classifier
[Yan & Huan '10]
(Winner of PASCAL 2010 Challenge)

Slide credit: Svetlana Lazebnik

What About Learning the Features?

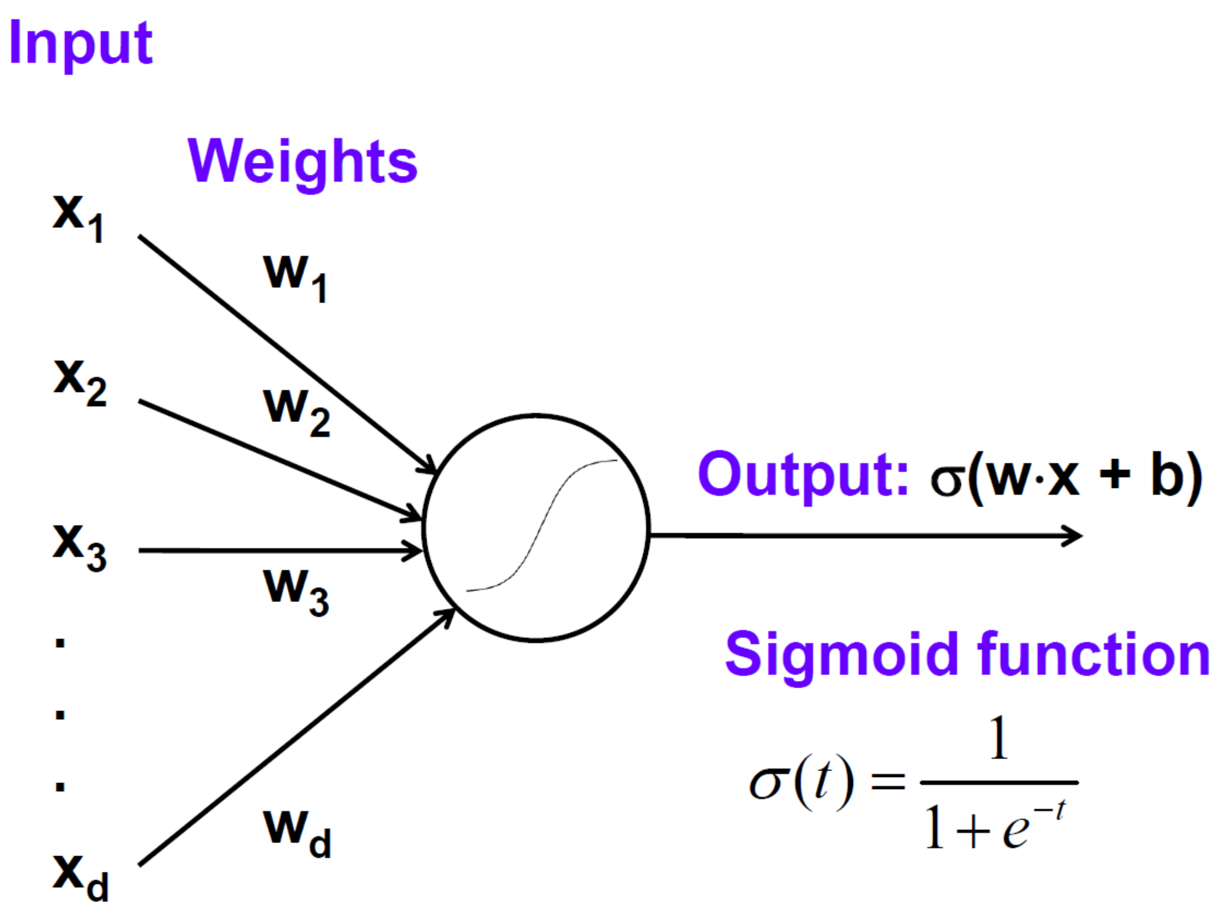
- Learn a *feature hierarchy* all the way from pixels to classifier
  - Each layer extracts features from the output of previous layer
  - Train all layers jointly

“Shallow” vs. “Deep” Architectures



Slide credit: Svetlana Lazebnik

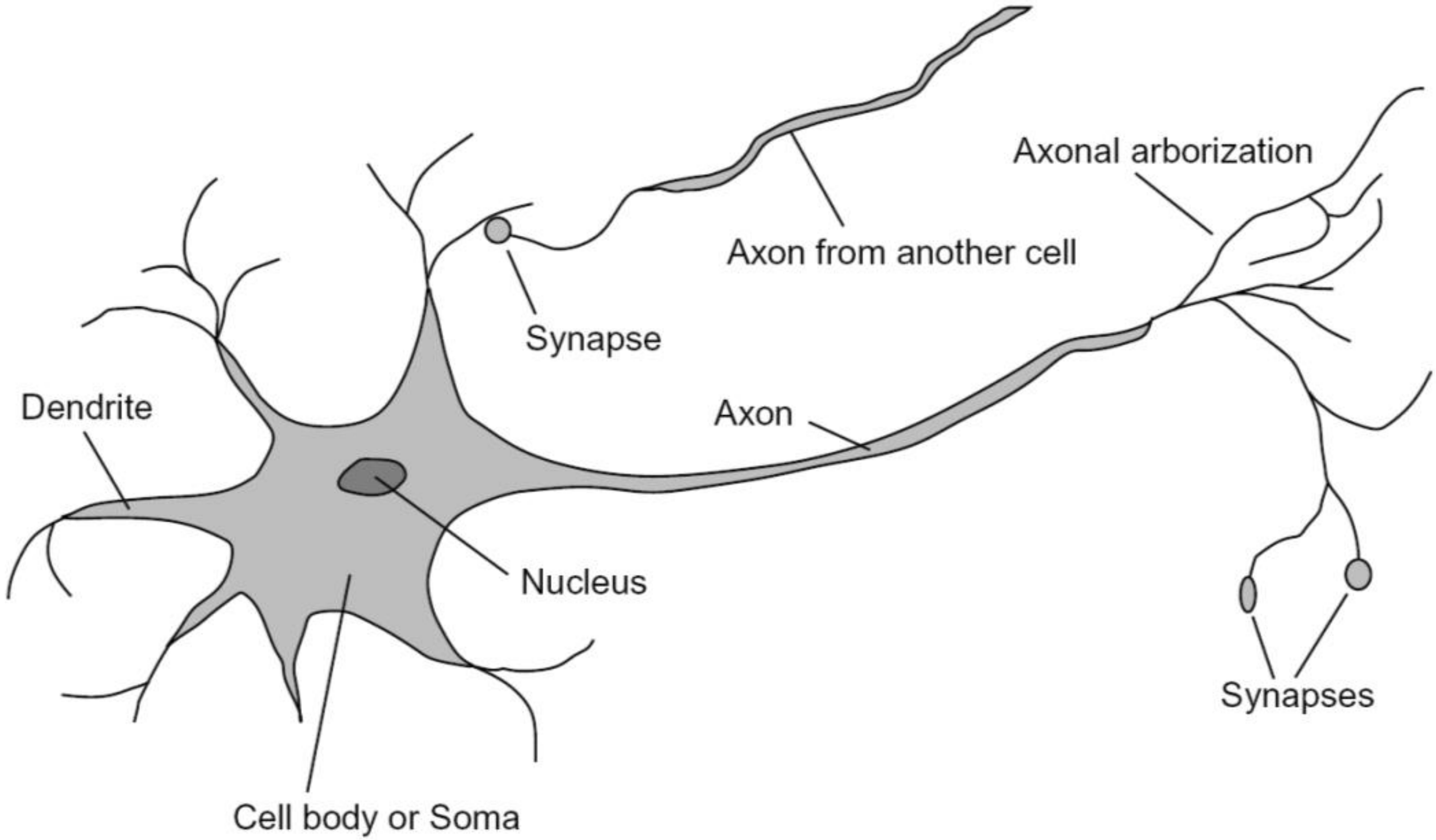
Background: Perceptrons



$$\sigma(t) = \frac{1}{1+e^{-t}}$$

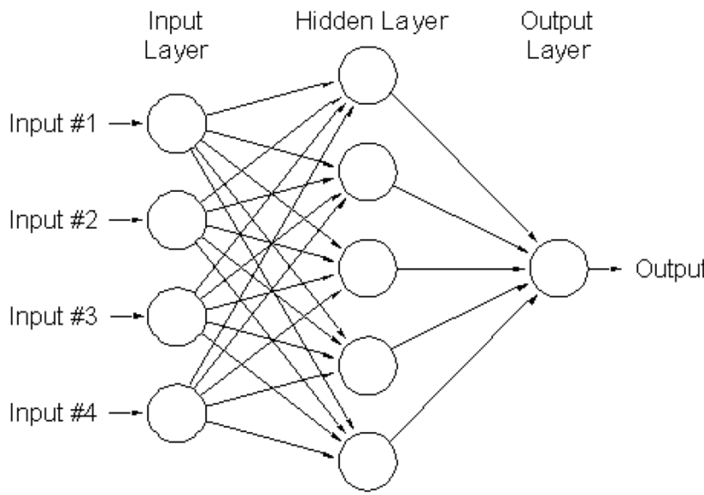
Slide credit: Svetlana Lazebnik

Inspiration: Neuron Cells



Slide credit: Svetlana Lazebnik, Rob Fergus

Background: Multi-Layer Neural Networks



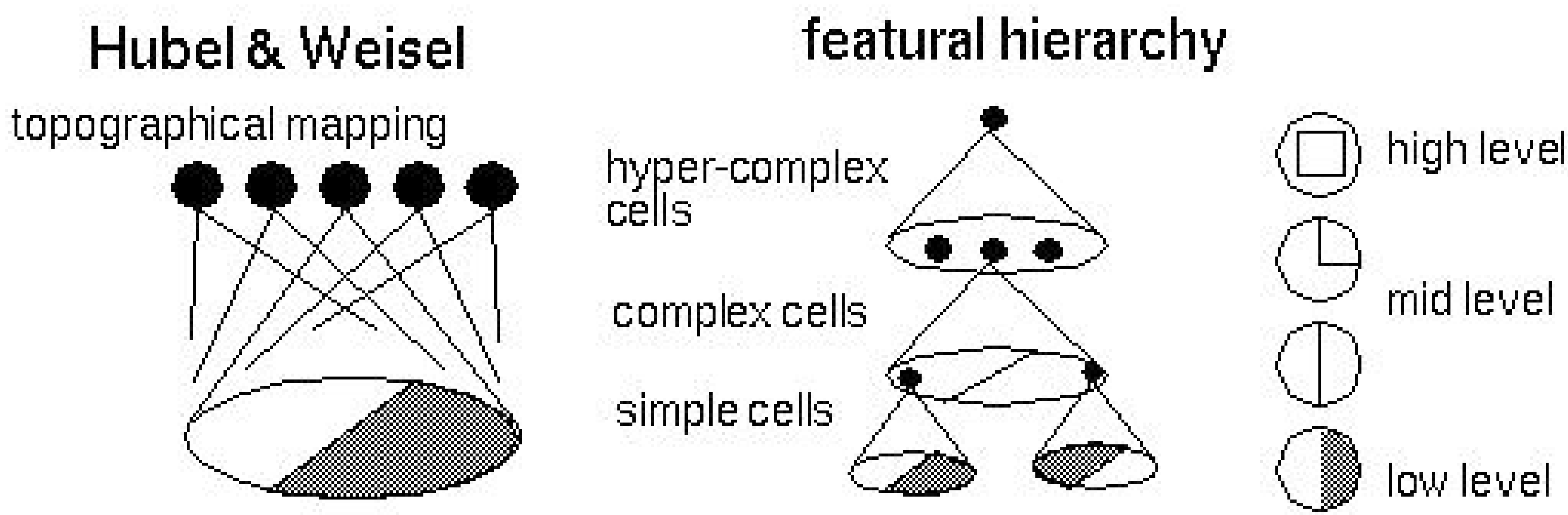
- Nonlinear classifier
  - Training: find network weights $\mathbf{w}$ to minimize the error between true training labels t_n and estimated labels $f_{\mathbf{w}}(x_n)$:

$$E(\mathbf{W}) = \sum_n L(t_n, f(\mathbf{x}_n; \mathbf{W}))$$

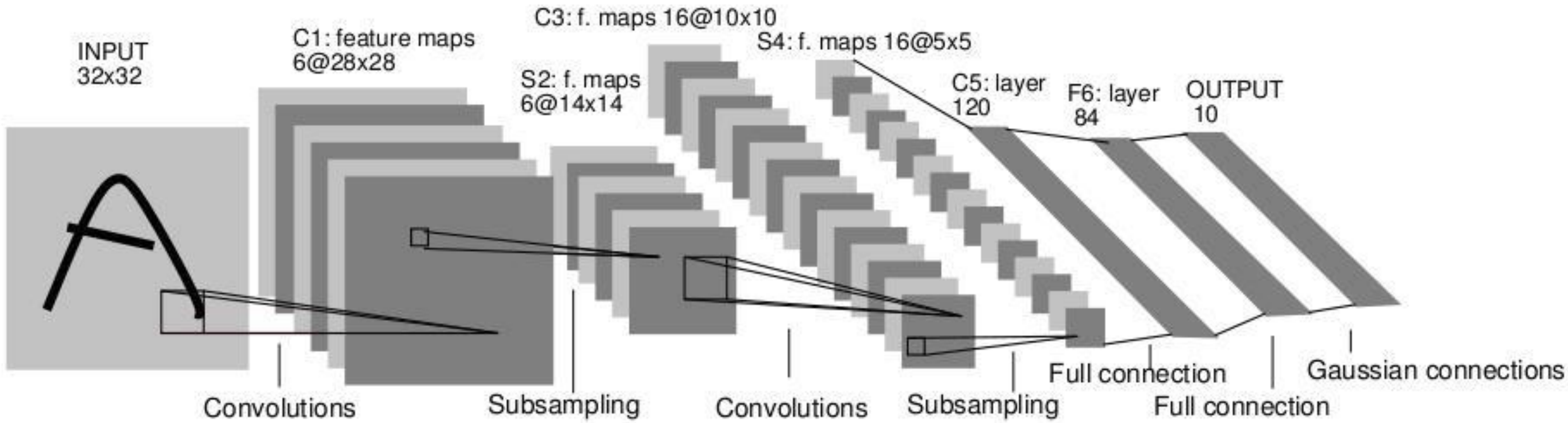
  - Minimization can be done by gradient descent, provided f is differentiable
    - Training method: Error backpropagation.

Hubel/Wiesel Architecture

- D. Hubel, T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells



Convolutional Neural Networks (CNN, ConvNet)



- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end

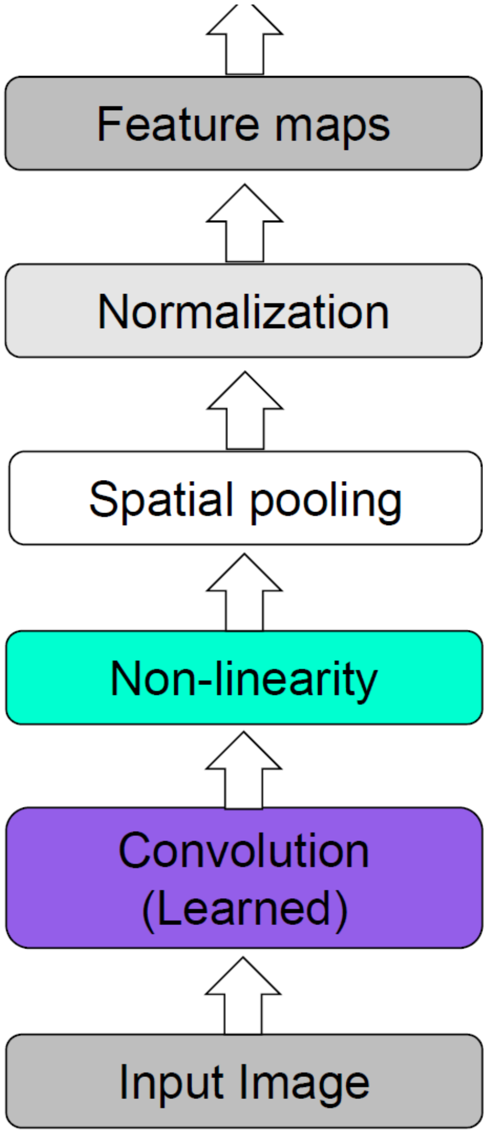
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Topics of This Lecture

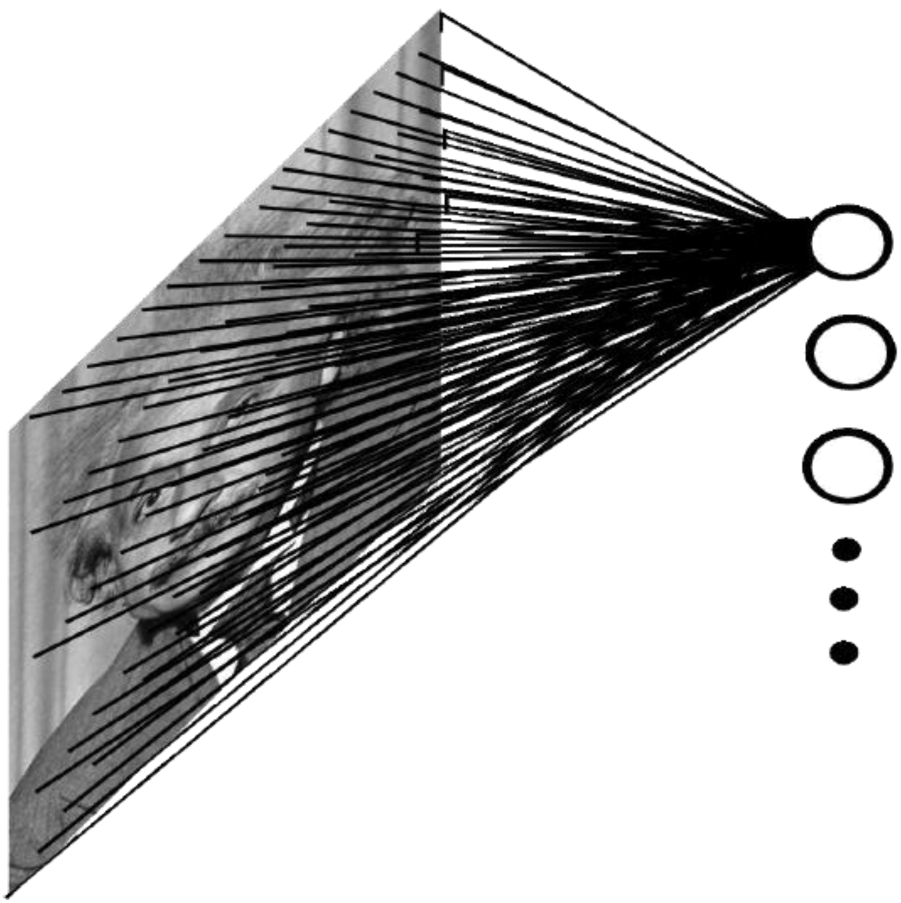
- Recap: Recognition with Local Features
- Dealing with Outliers
  - RANSAC
  - Generalized Hough Transform
- Deep Learning
  - Motivation
  - Neural Networks
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities

Convolutional Networks: Structure

- Feed-forward feature extraction
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. (Normalization)
- Supervised training of convolutional filters by back-propagating classification error



Convolutional Networks: Intuition



- Fully connected network
  - E.g. 1000×1000 image
    1M hidden units

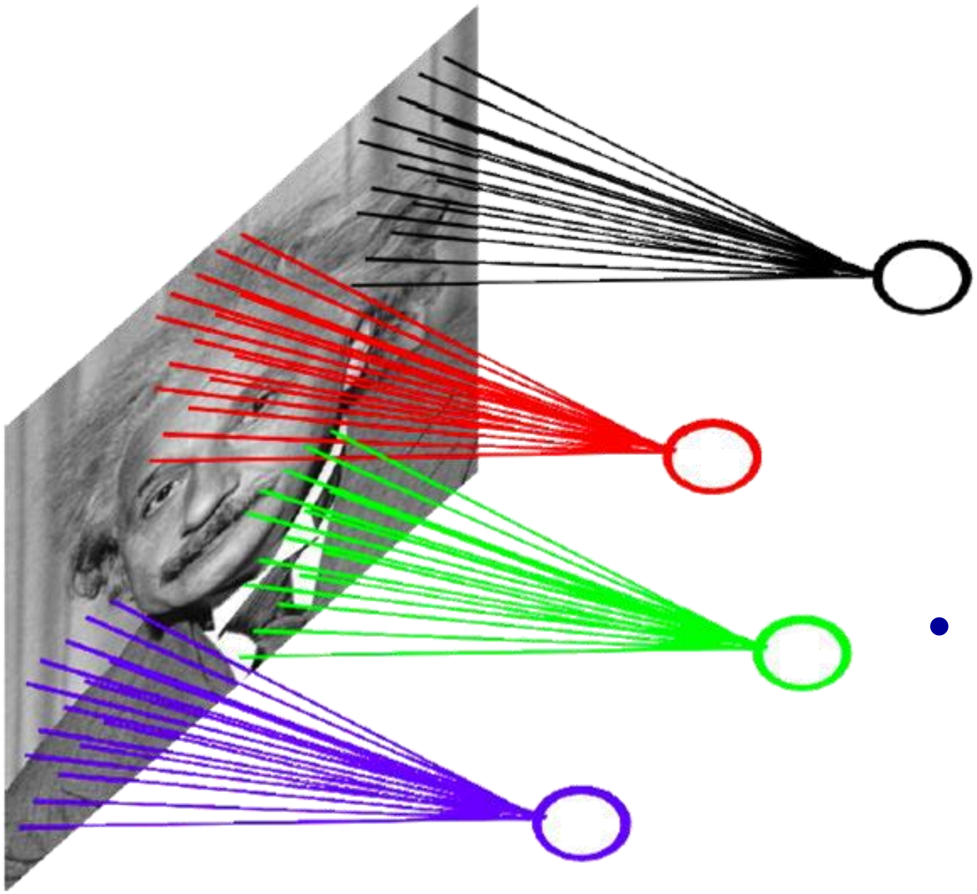
  ⇒ 1T parameters!

- Ideas to improve this
  - Spatial correlation is local

Slide adapted from Marc'Aurelio Ranzato

Image source: Yann LeCun

Convolutional Networks: Intuition



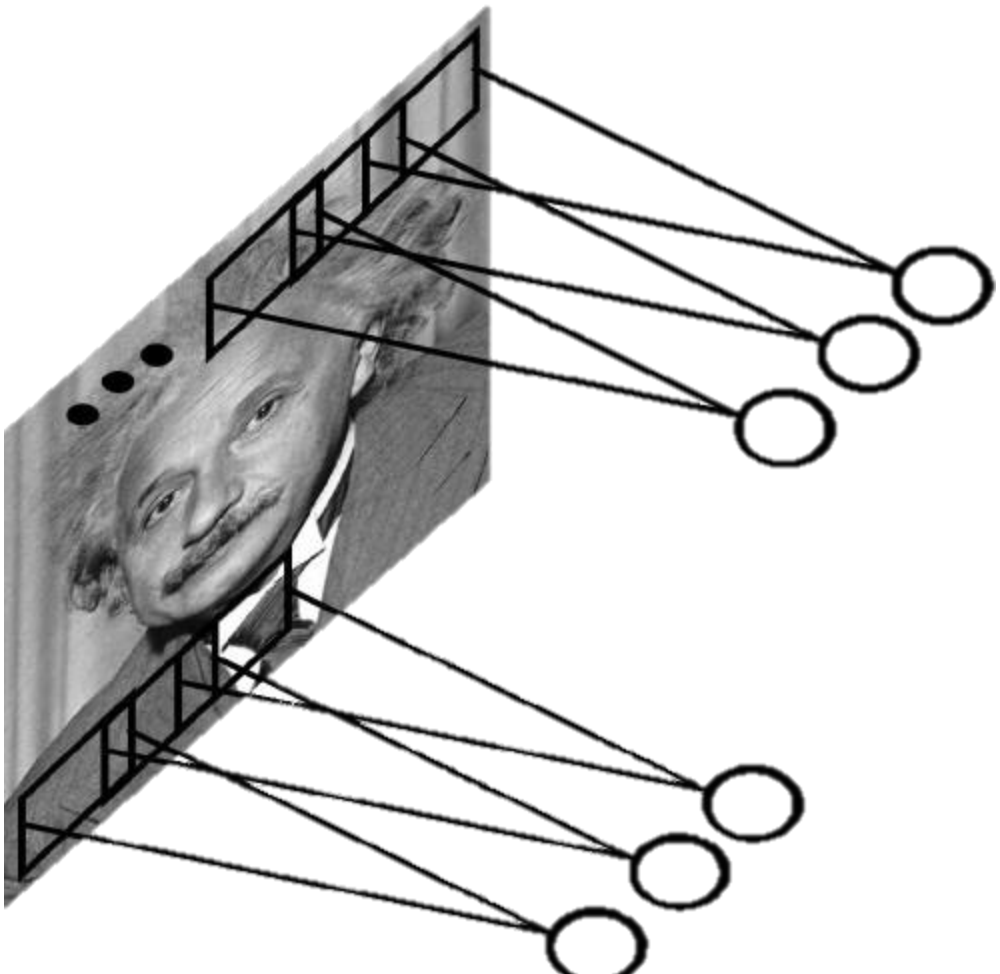
- Locally connected net
  - E.g. 1000×1000 image
    1M hidden units
    10×10 receptive fields

  ⇒ 100M parameters!

- Ideas to improve this
  - Spatial correlation is local
  - Want translation invariance

Image source: Yann LeCun

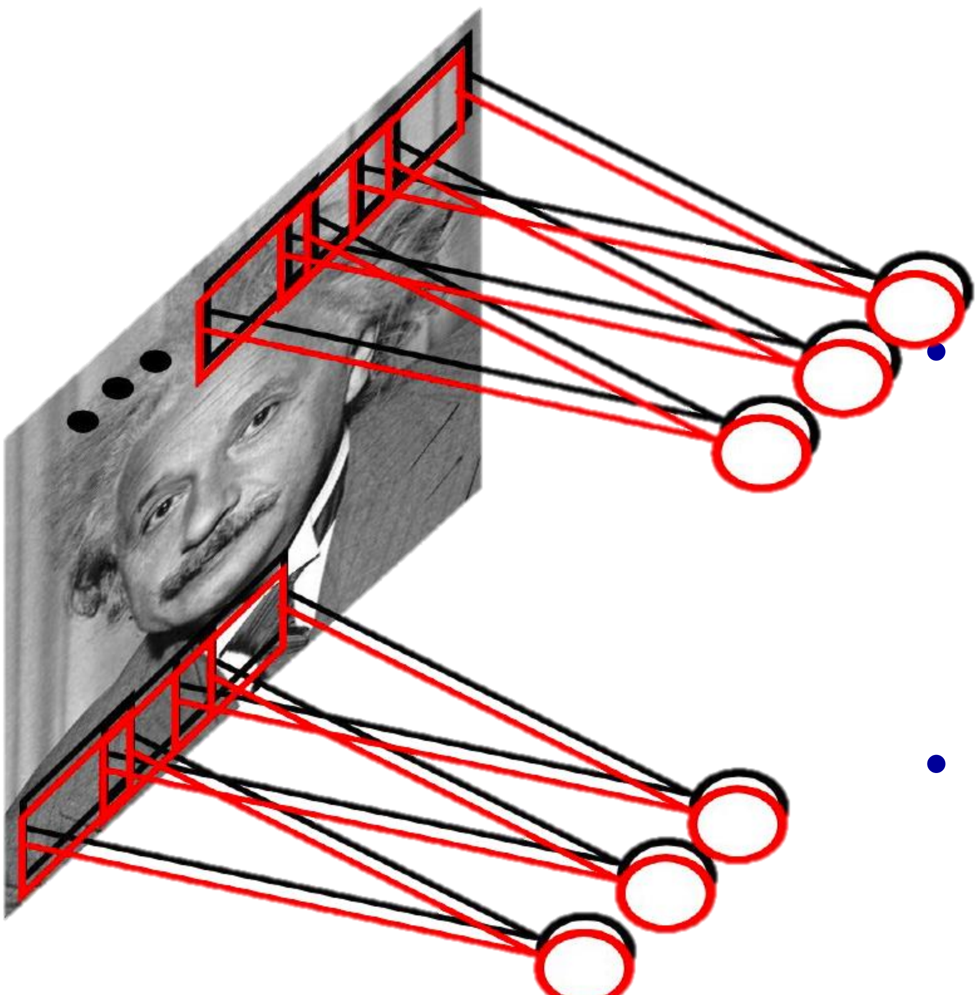
Convolutional Networks: Intuition



- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels

Image source: Yann LeCun

Convolutional Networks: Intuition



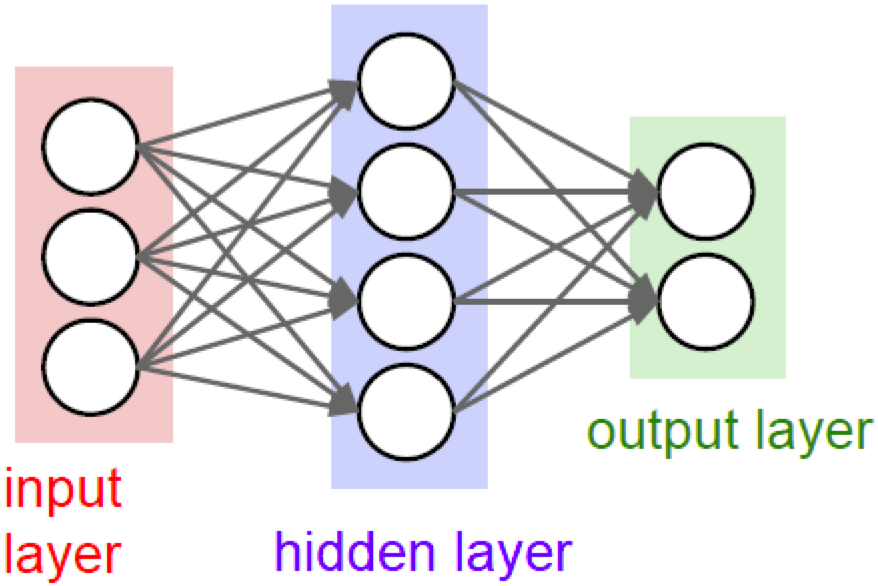
- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels
- Learn *multiple* filters
  - E.g. 1000×1000 image
    100 filters
    10×10 filter size

  ⇒ 10k parameters
- Result: Response map
  - size: 1000×1000×100
  - Only memory, not params!

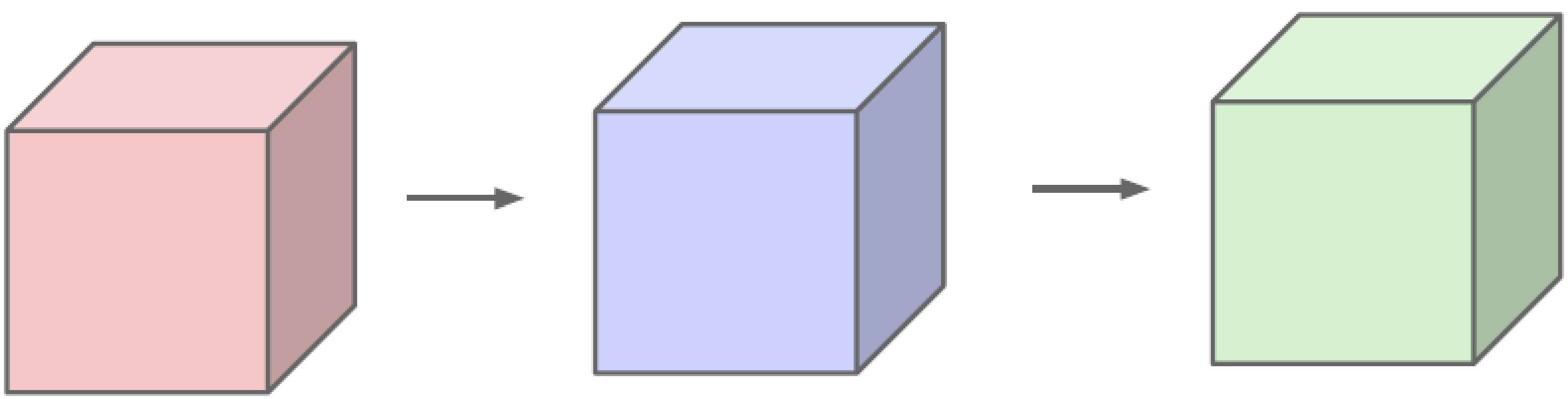
Image source: Yann LeCun

Important Conceptual Shift

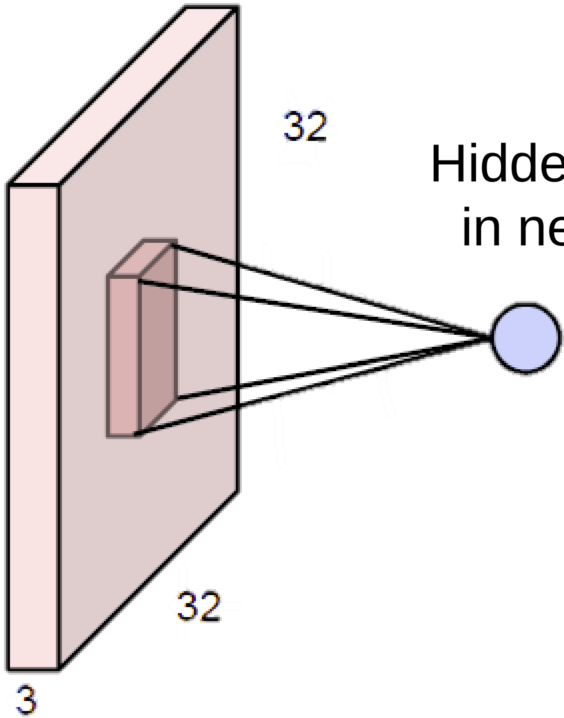
- Before



- Now:



Convolution Layers



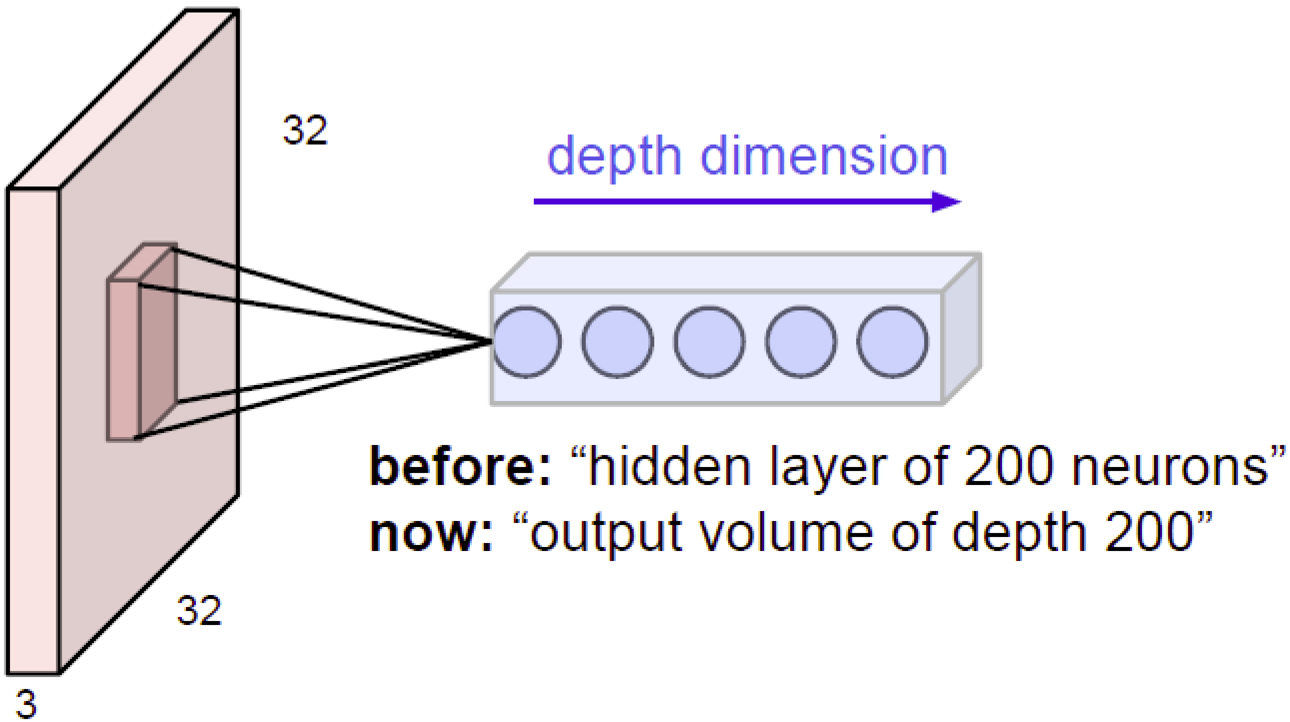
Example
image: $32 \times 32 \times 3$ volume

Before: Full connectivity
$32 \times 32 \times 3$ weights

Now: Local connectivity
One neuron connects to, e.g., $5 \times 5 \times 3$ region.
$\Rightarrow$ Only $5 \times 5 \times 3$ shared weights.

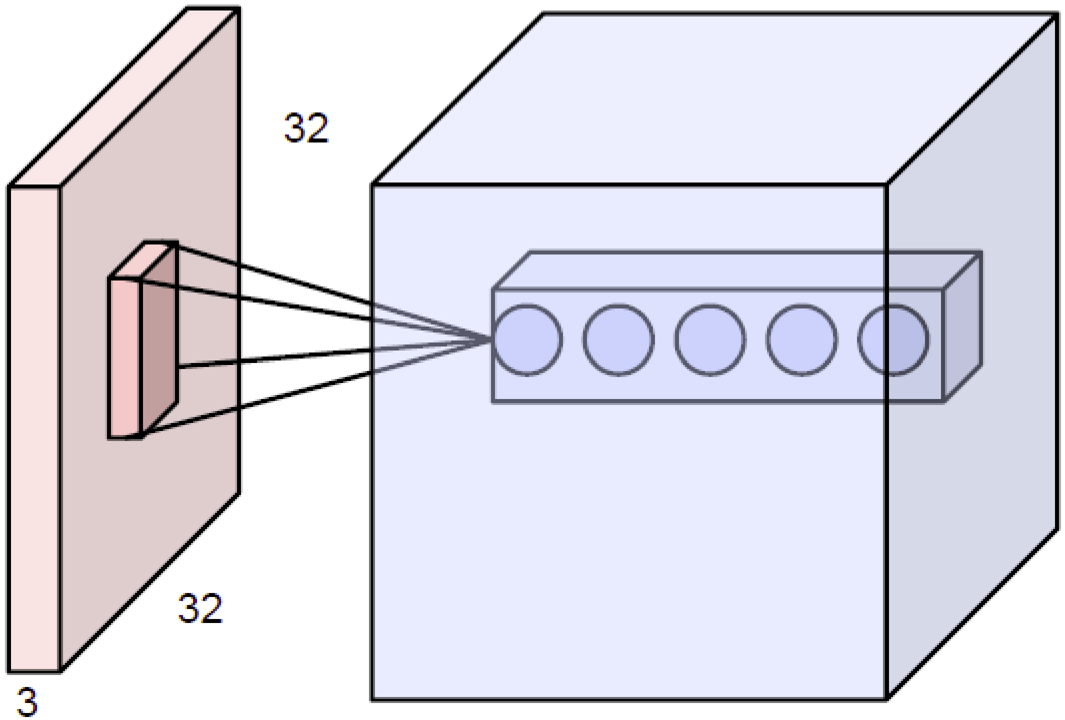
- Note: Connectivity is
  - Local in space (5×5 inside 32×32)
  - But full in depth (all 3 depth channels)

Convolution Layers

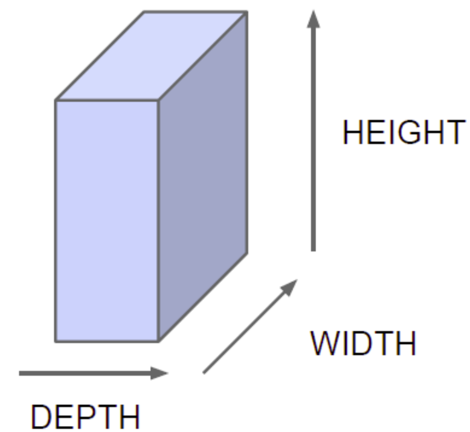


- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth

Convolution Layers

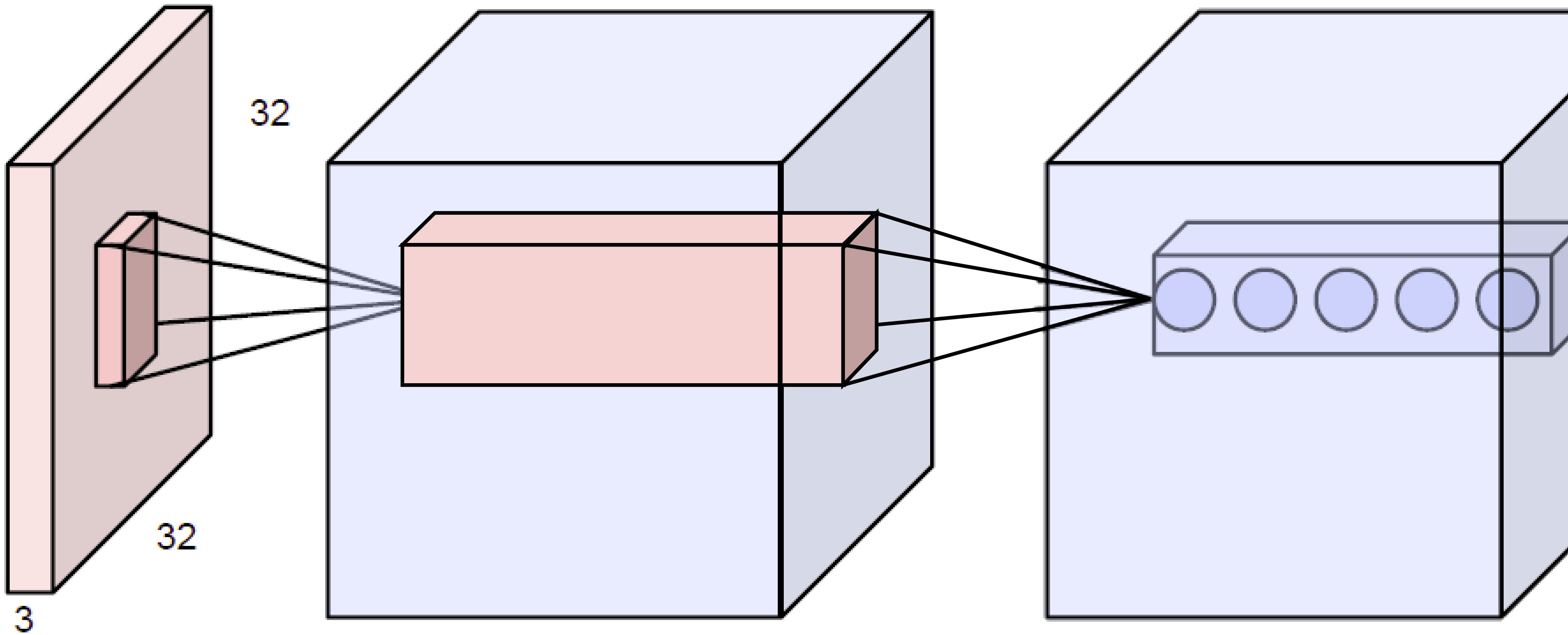


Naming convention:



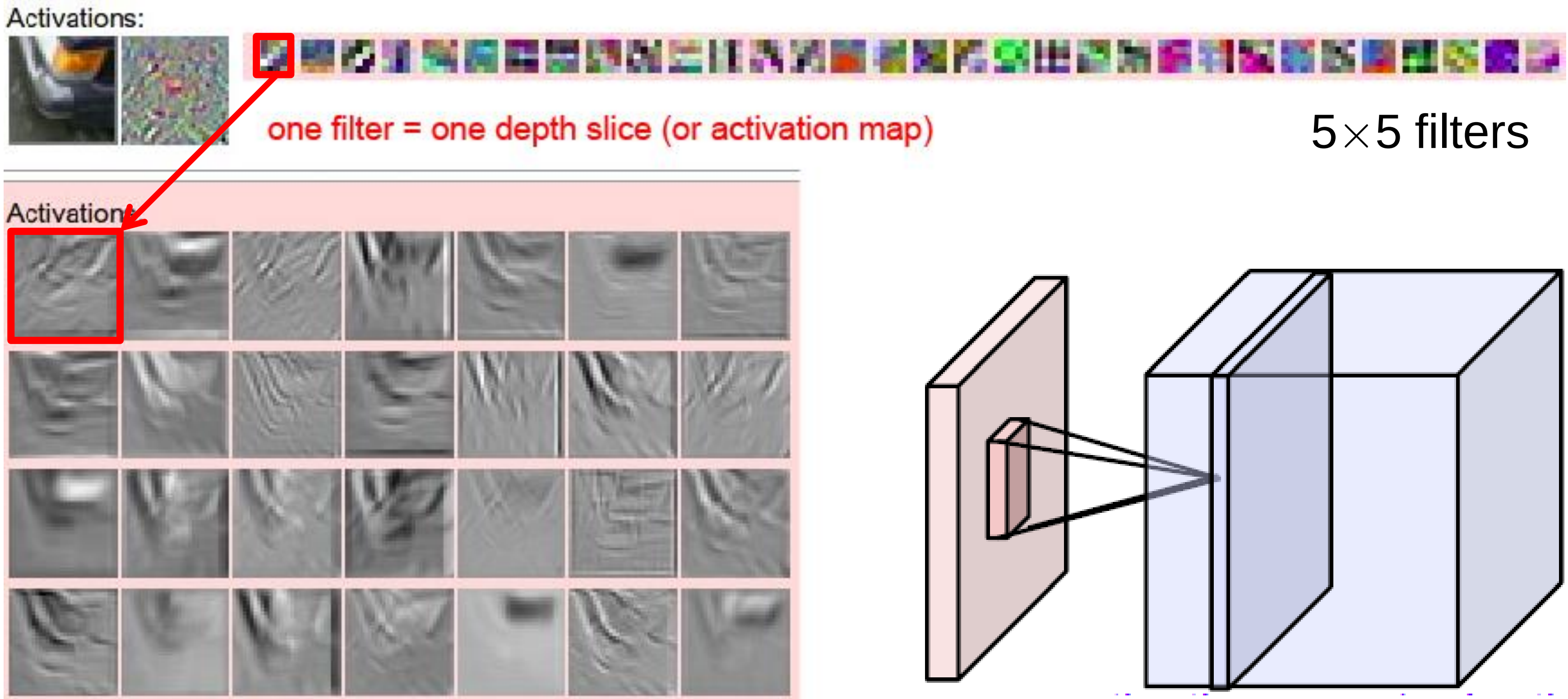
- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single [1×1×depth] depth column in output volume.

Convolution Layers



- All Neural Net activations arranged in 3 dimensions
  - Convolution layers can be stacked
  - The filters of the next layer then operate on the full activation volume.
  - Filters are local in (x,y), but densely connected in depth.

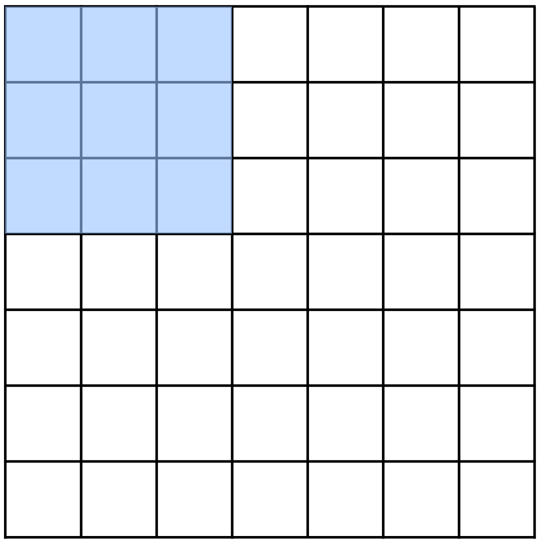
Activation Maps of Convolutional Filters



Each activation map is a depth slice through the output volume.

Activation maps

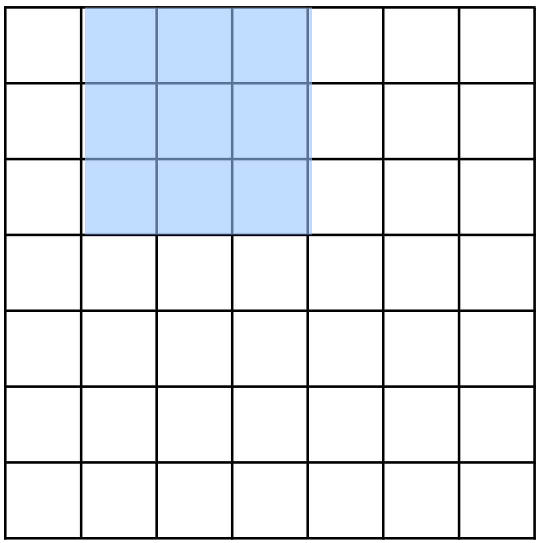
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1

- Replicate this column of hidden neurons across space, with some stride.

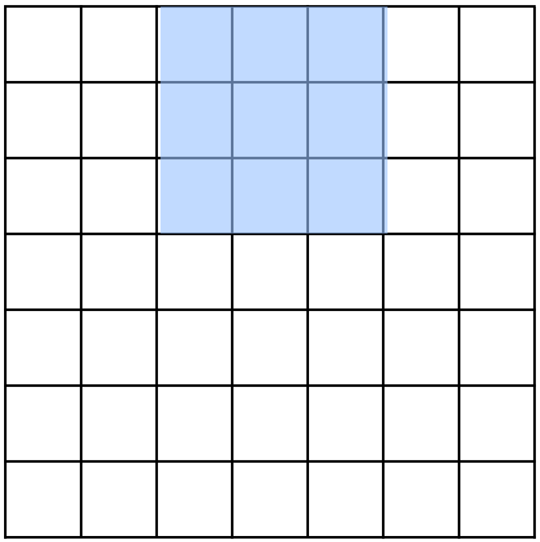
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1

- Replicate this column of hidden neurons across space, with some stride.

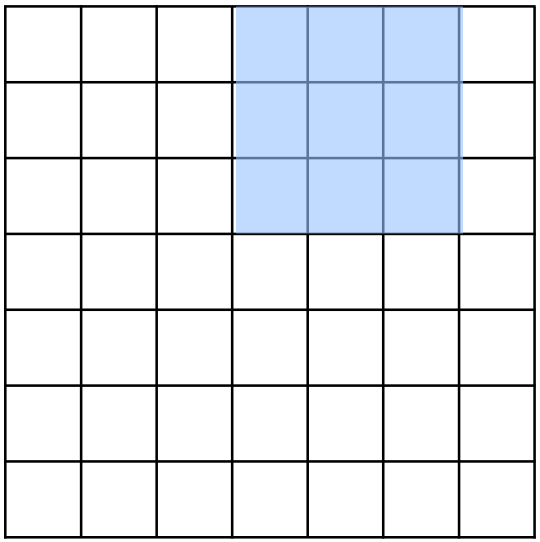
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1

- Replicate this column of hidden neurons across space, with some stride.

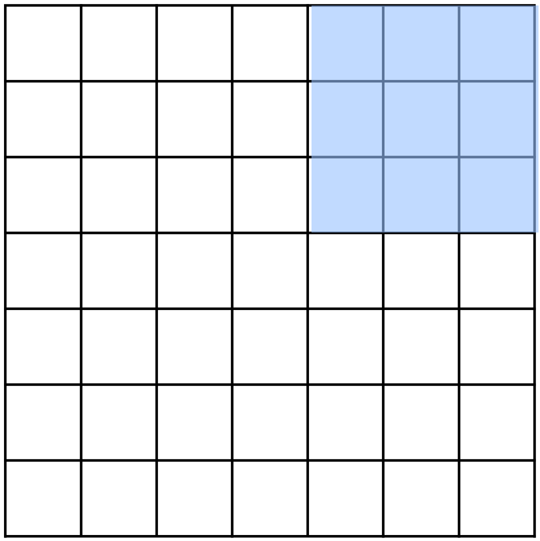
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1

- Replicate this column of hidden neurons across space, with some stride.

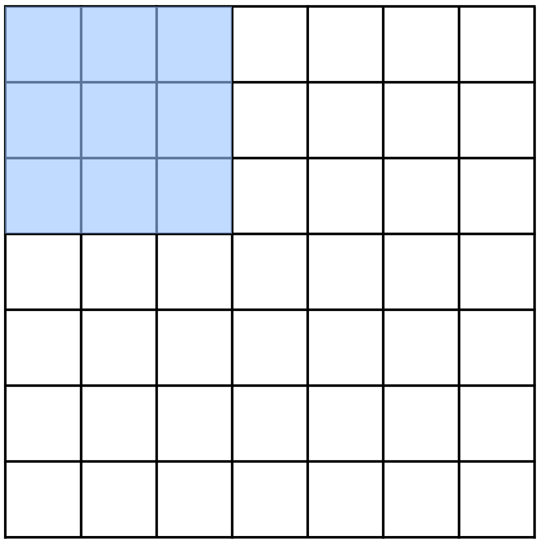
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1
$\Rightarrow 5 \times 5$ output

- Replicate this column of hidden neurons across space, with some stride.

Convolution Layers

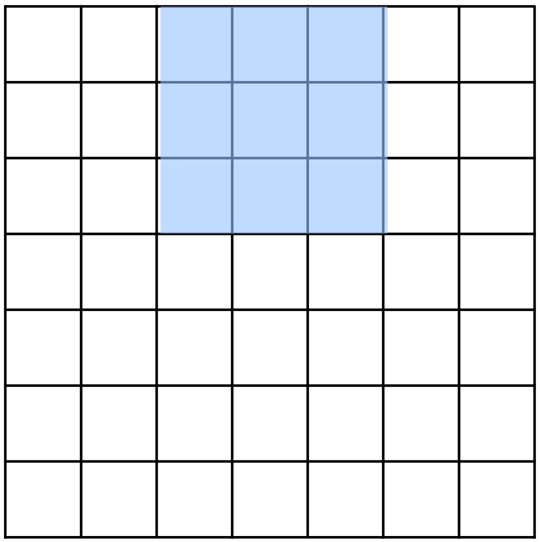


Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

What about stride 2?

- Replicate this column of hidden neurons across space, with some stride.

Convolution Layers

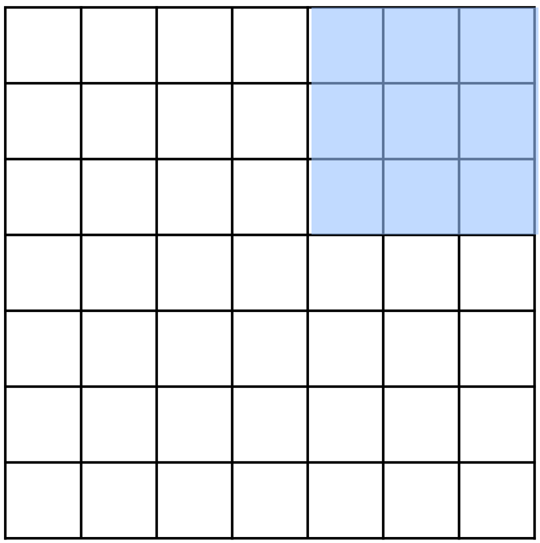


Example:
7×7 input
assume 3×3 connectivity
stride 1
$\Rightarrow$ 5×5 output

What about stride 2?

- Replicate this column of hidden neurons across space, with some stride.

Convolution Layers

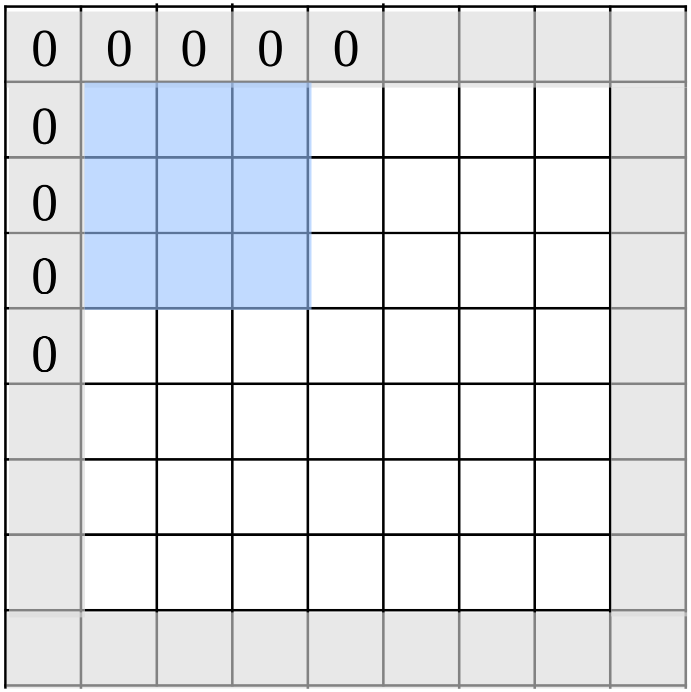


Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

What about stride 2?
⇒ 3×3 output

- Replicate this column of hidden neurons across space, with some stride.

Convolution Layers

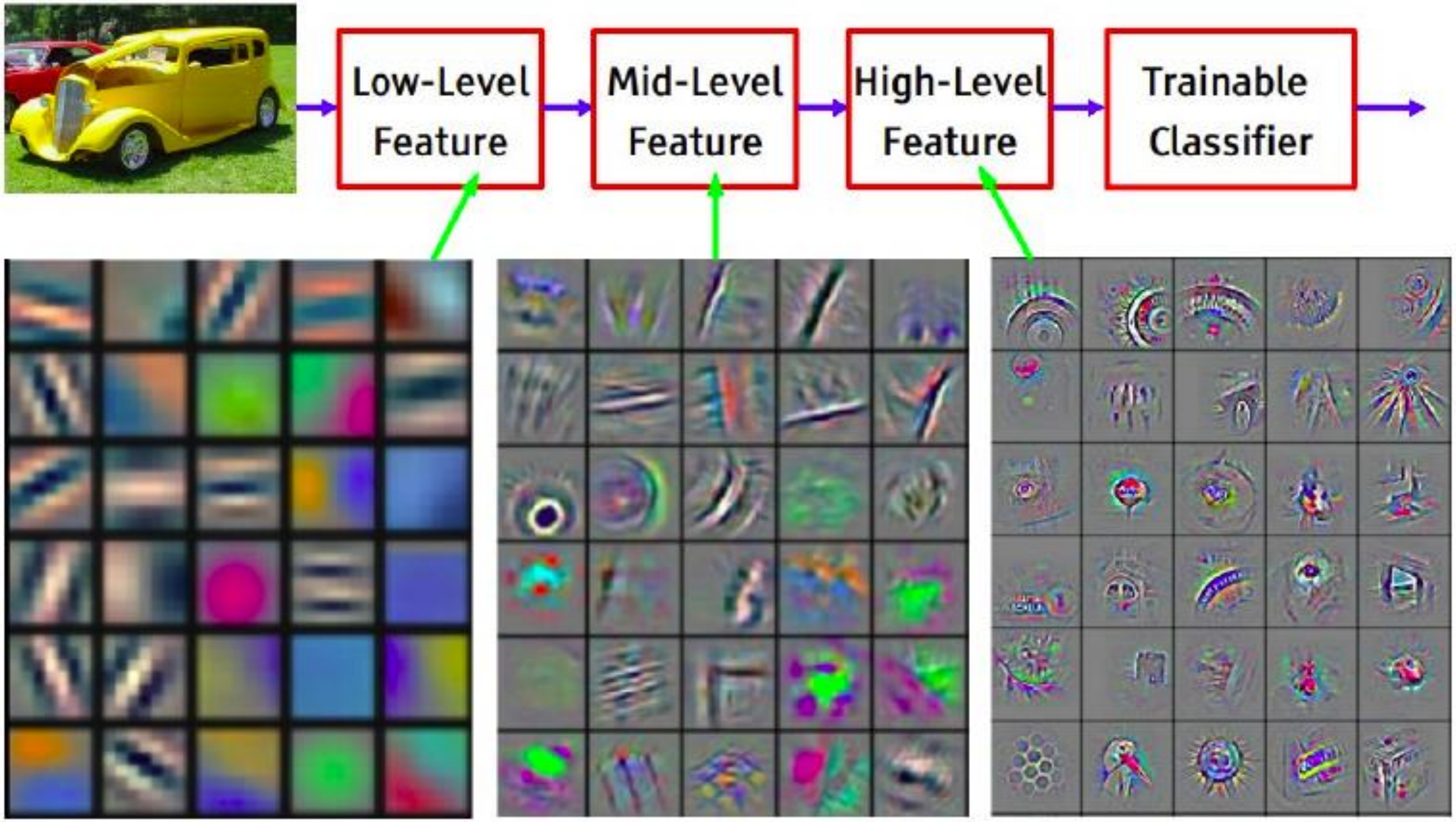


Example:
7×7 input
assume 3×3 connectivity
stride 1
$\Rightarrow 5 \times 5$ output

What about stride 2?
$\Rightarrow 3 \times 3$ output

- Replicate this column of hidden neurons across space, with some stride.
- In practice, common to zero-pad the border.
  - Preserves the size of the input spatially.

Effect of Multiple Convolution Layers

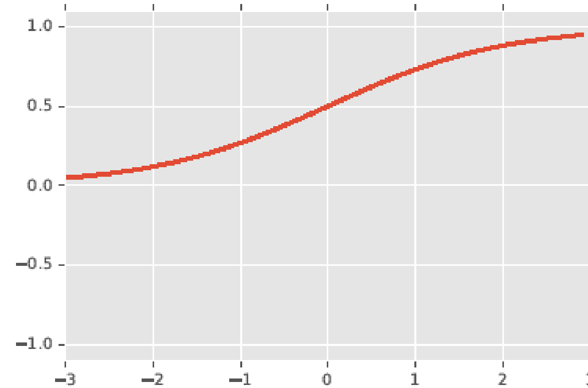


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Commonly Used Nonlinearities

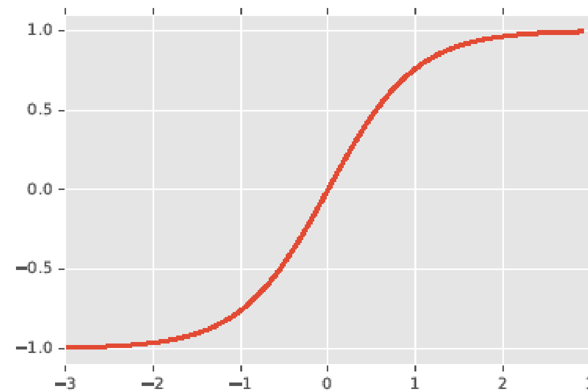
- Sigmoid

$$\begin{aligned} g(a) &= \sigma(a) \\ &= \frac{1}{1+\exp\{-a\}} \end{aligned}$$



- Hyperbolic tangent

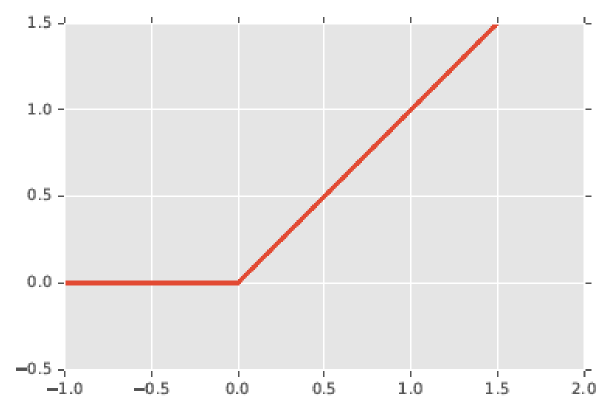
$$\begin{aligned} g(a) &= tanh(a) \\ &= 2\sigma(2a) - 1 \end{aligned}$$



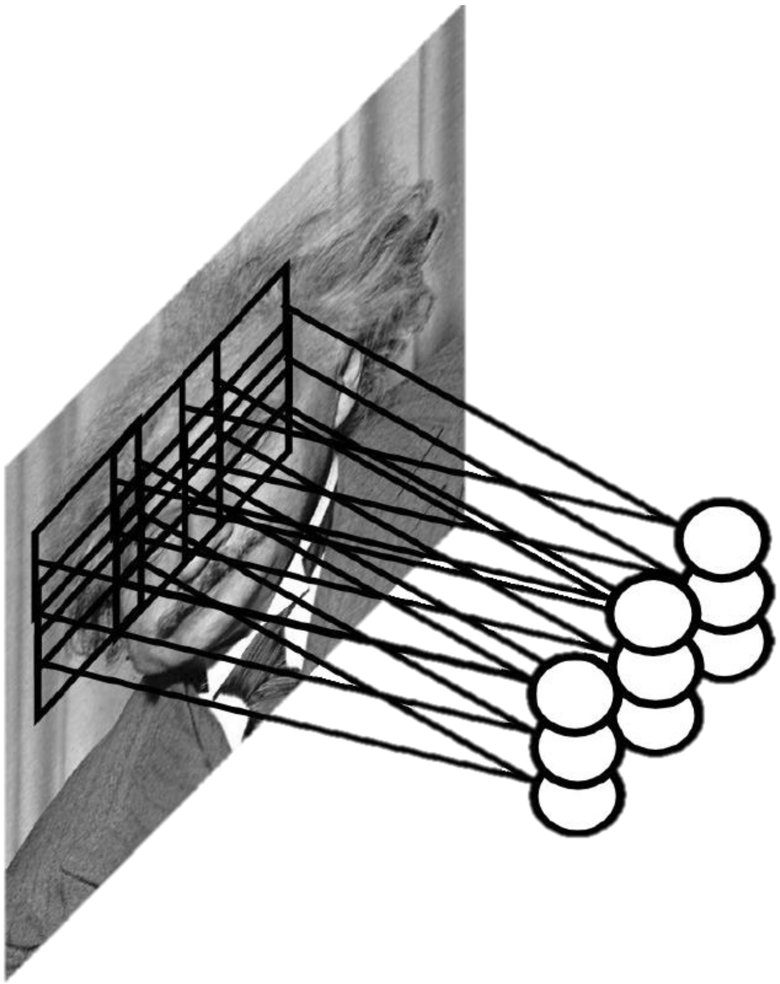
- Rectified linear unit (ReLU)

$$g(a) = \max\{0, a\}$$

Preferred option for deep networks

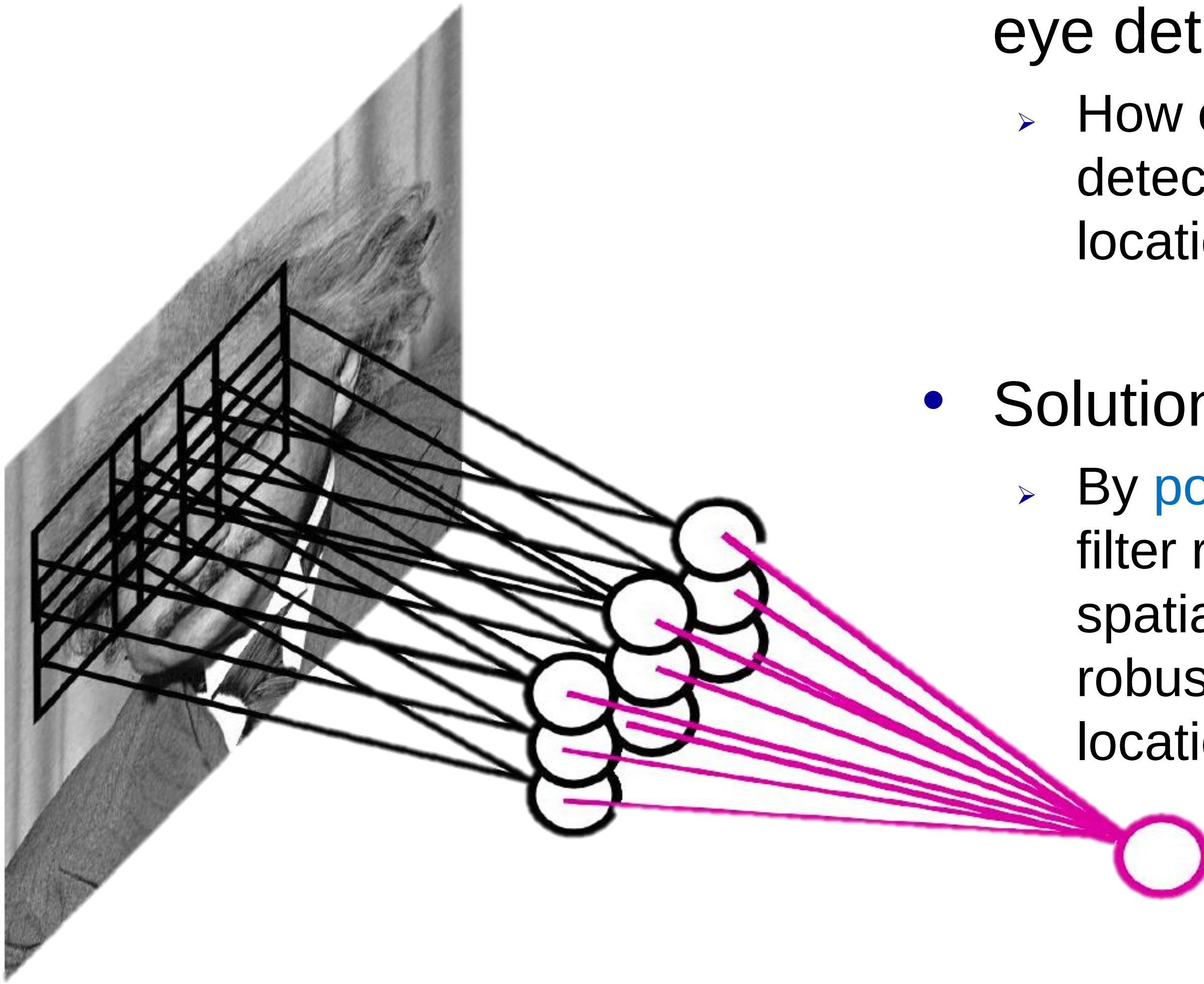


Convolutional Networks: Intuition



- Let's assume the filter is an eye detector
  - How can we make the detection robust to the exact location of the eye?

Convolutional Networks: Intuition

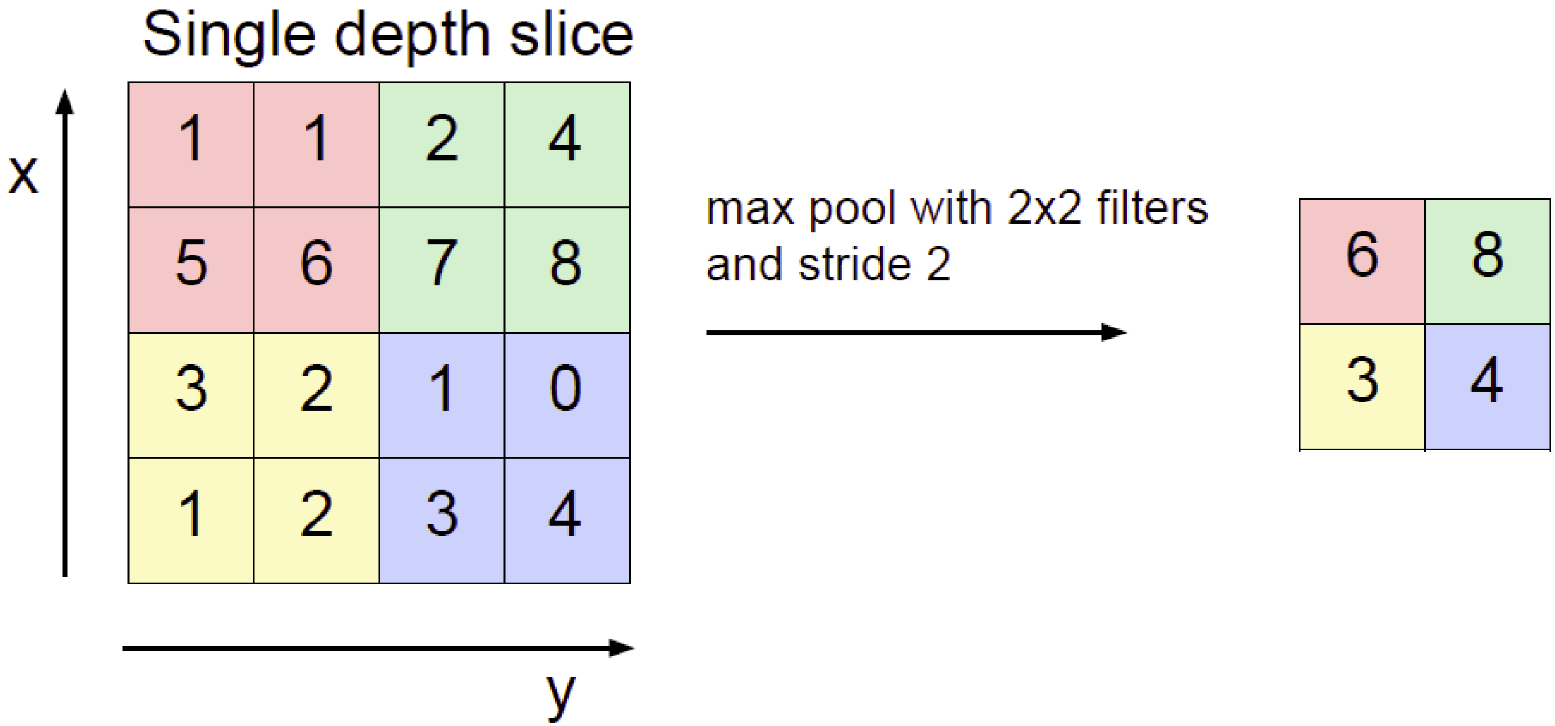


- Let's assume the filter is an eye detector
  - How can we make the detection robust to the exact location of the eye?

- Solution:
  - By pooling (e.g., max or avg) filter responses at different spatial locations, we gain robustness to the exact spatial location of features.

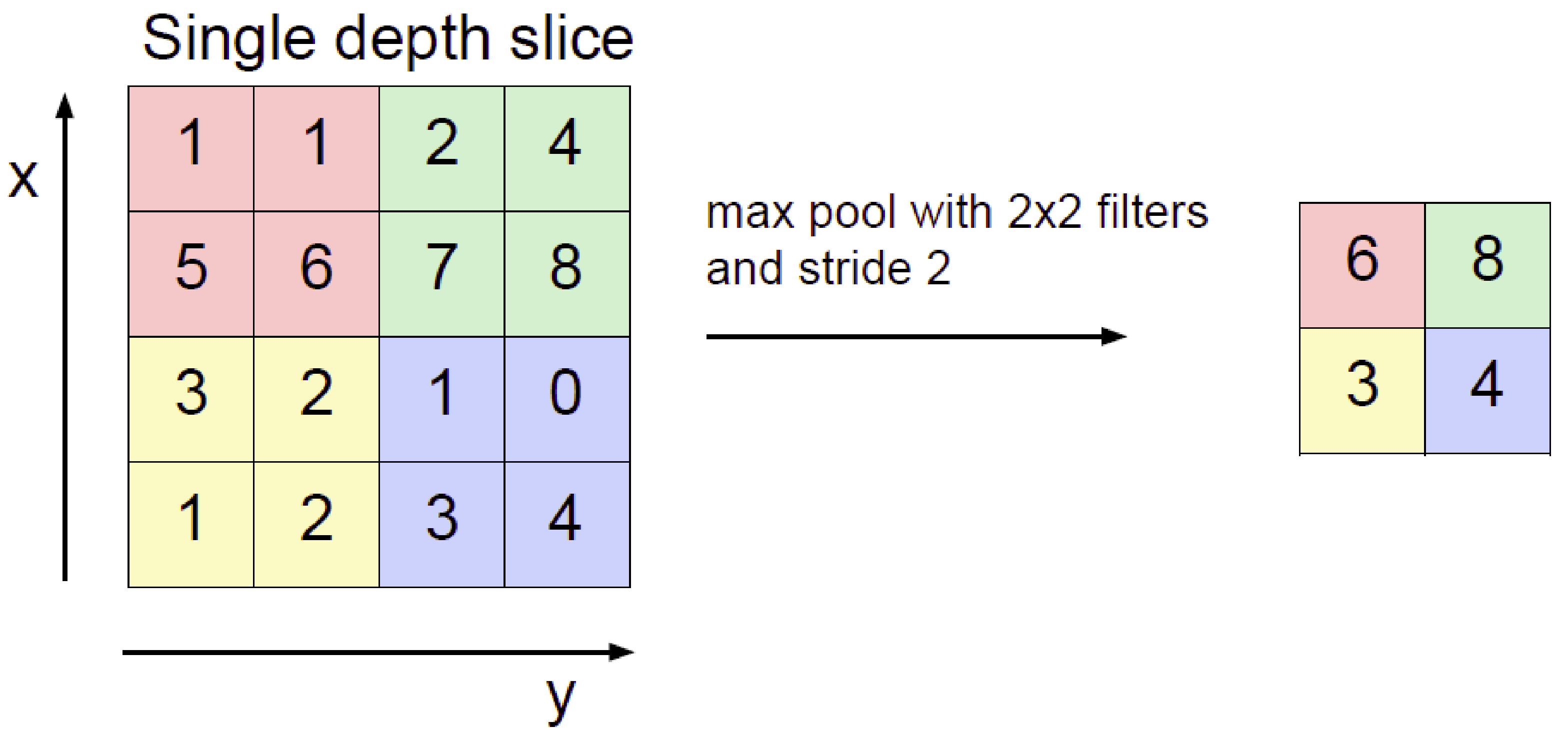
Image source: Yann LeCun

Max Pooling



- Effect:
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations

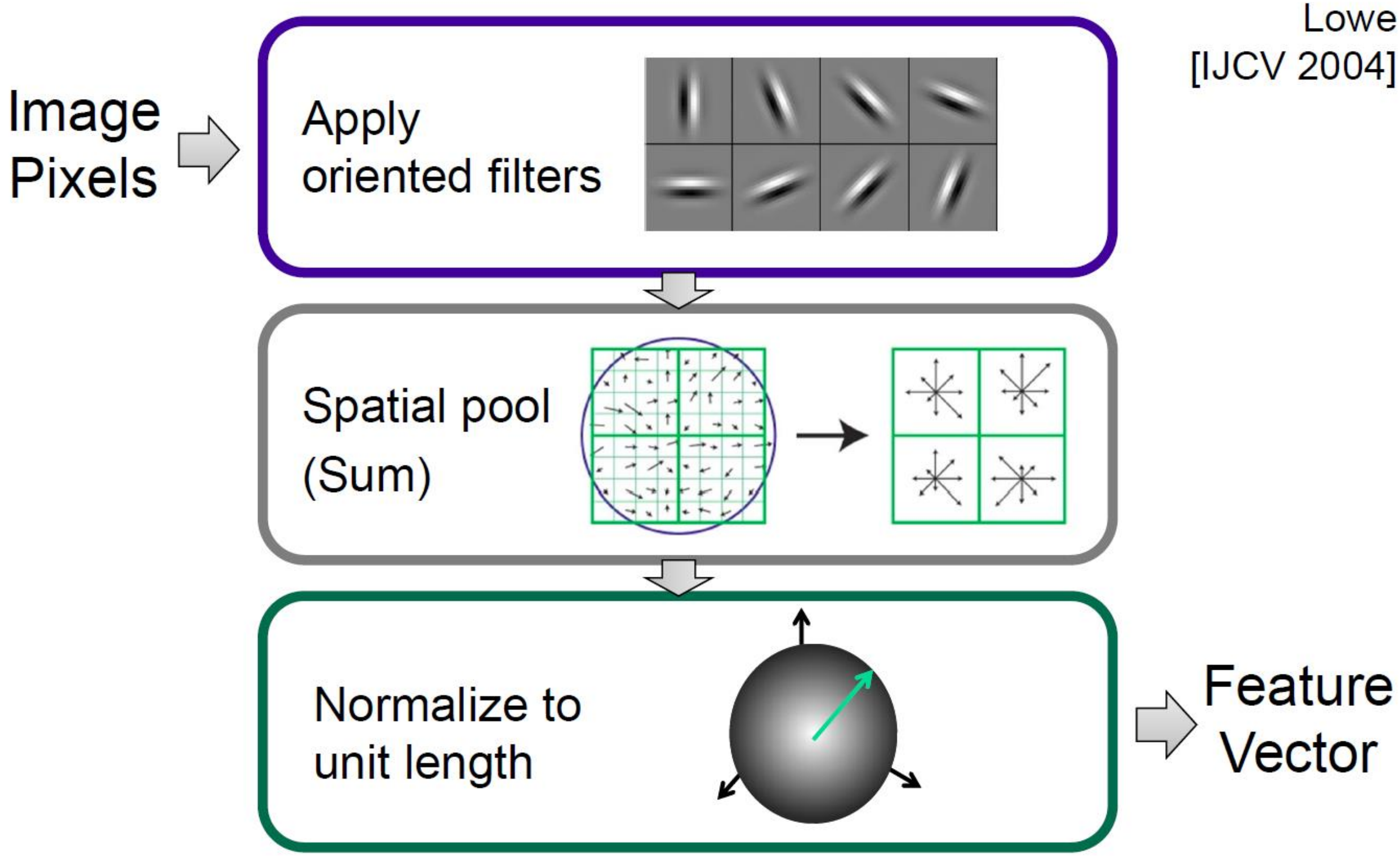
Max Pooling



- Note
  - Pooling happens independently across each slice, preserving the number of slices.

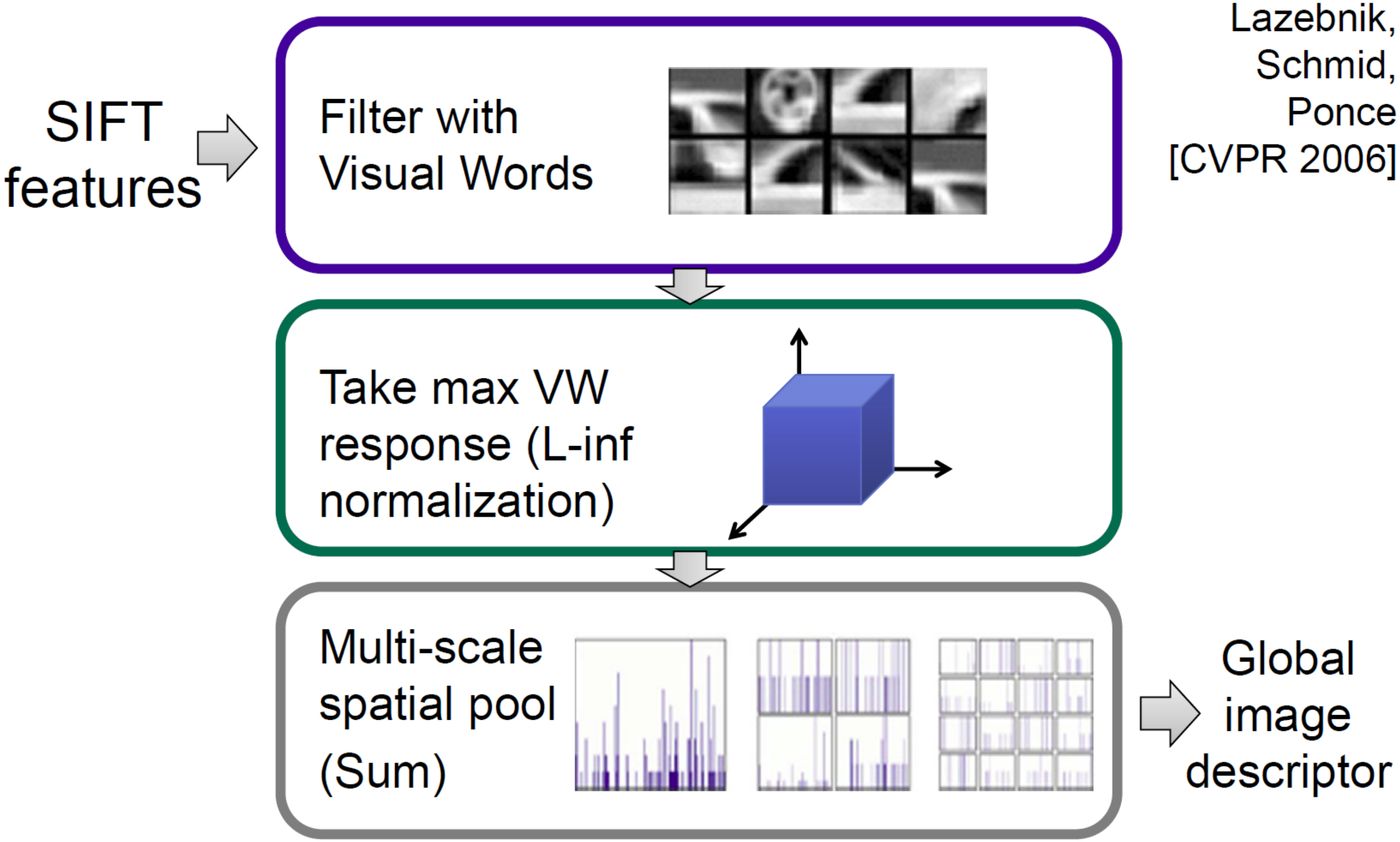
Compare: SIFT Descriptor

Lowe
[IJCV 2004]



Compare: Spatial Pyramid Matching

Lazebnik, Schmid, Ponce [CVPR 2006]



Slide credit: Svetlana Lazebnik

References and Further Reading

- More information on Deep Learning and CNNs can be found in Chapters 6 and 9 of the Goodfellow & Bengio book

I. Goodfellow, Y. Bengio, A. Courville
Deep Learning
MIT Press, 2016
http://www.deeplearningbook.org/