Convolutional Neural Networks

(CNNs, or ConvNet, DCN)

Sources:

“CS 790: Introduction to Machine Learning” at U. Wisconsin
by Jia-Bin Huang, Virginia Tech.

Ch. 6, ”Deep Learning” textbook
by Goodfellow et al.

Image Categorization

Training Images

Training

Training Labels

Image Features

Classifier Training

Trained Classifier

Testing

Test Image

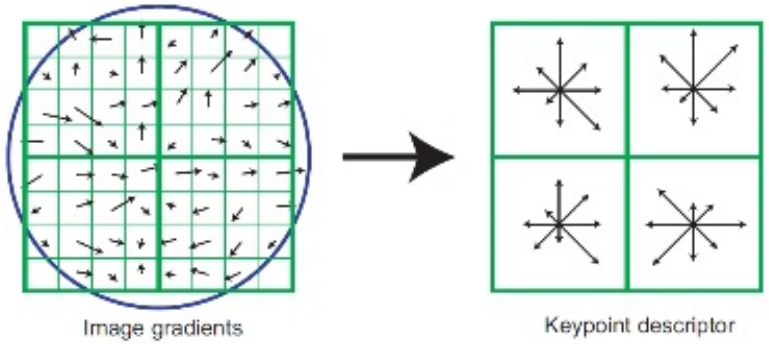
Image Features

Trained Classifier

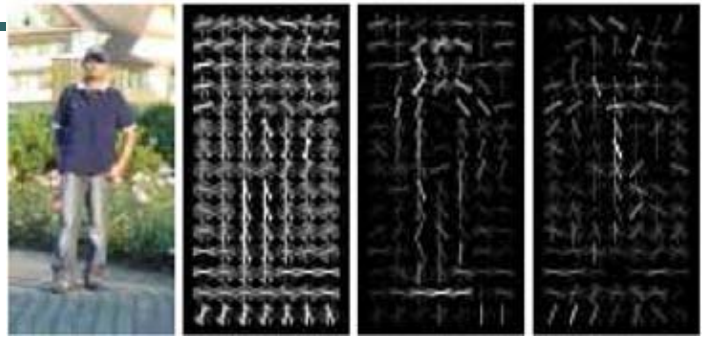
Prediction

Outdoor

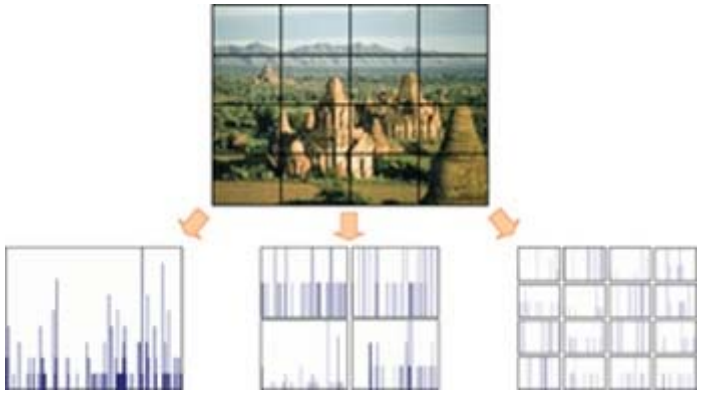
Features are the Keys



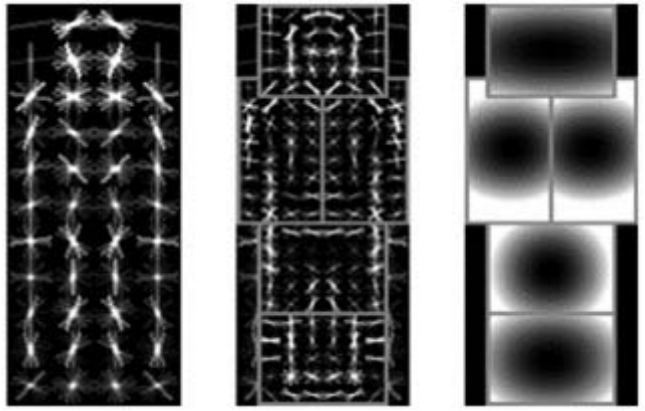
SIFT [Loewe IJCV 04]



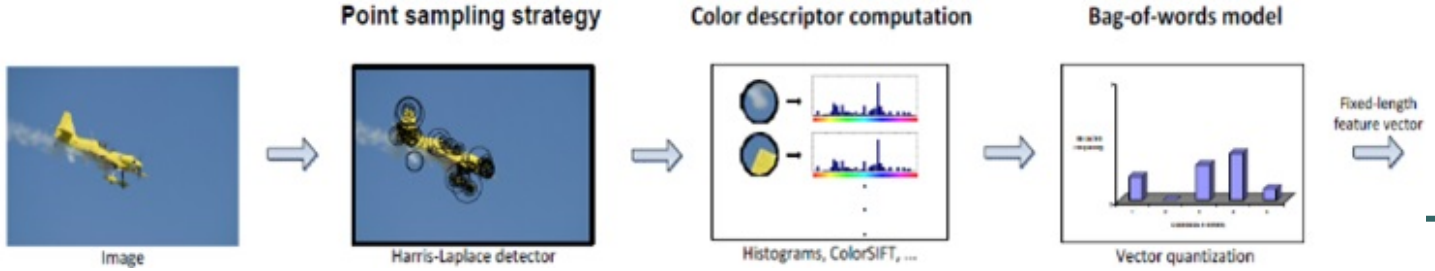
HOG [Dalal and Triggs CVPR 05]



SPM [Lazebnik et al. CVPR 06]



DPM [Felzenszwalb et al. PAMI 10]



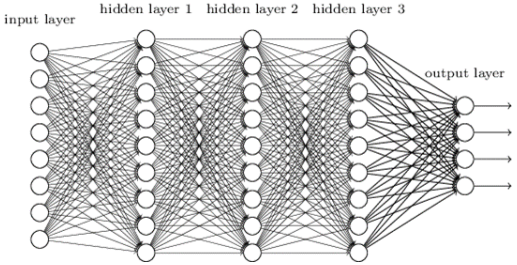
Color Descriptor [Van De Sande et al. PAMI 10]

Learning a Hierarchy of Feature Extractors

- Each layer of hierarchy extracts features from output of previous layer
- All the way from pixels → classifier
- Layers have the (nearly) same structure

Convolutional Neural Networks

- Convolutional neural networks (CNNs) use convolution in place of matrix multiplication in at least one of the layers of neural networks.
  - Everything else stays the same
    - Maximum likelihood
    - Back-propagation



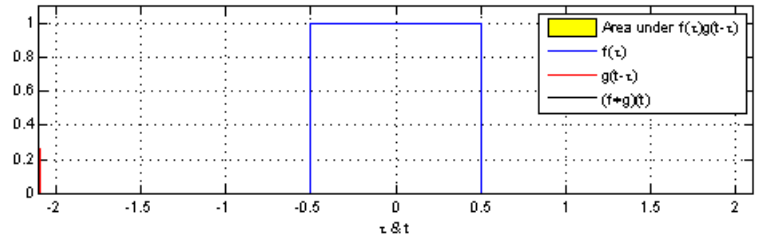
$$\mathbf{h} = g(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

  - CNNs are used to process data that has grid-like topology
    - 1-D: regularly sampled time-series data
    - 2-D: images
    - 3-D: volume data, video

Convolution Operation

- Convolution of two 1-D signals:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$



- Convolution of a 2-D image with a 2-D kernel:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n), \text{ or}$$

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

- Convolution of a 2-D image with a 2-D kernel:

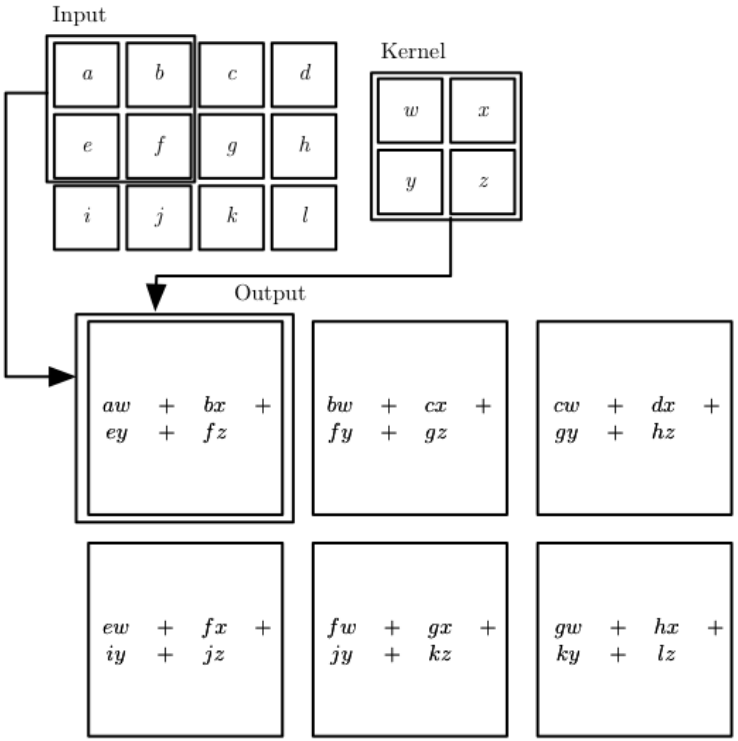
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

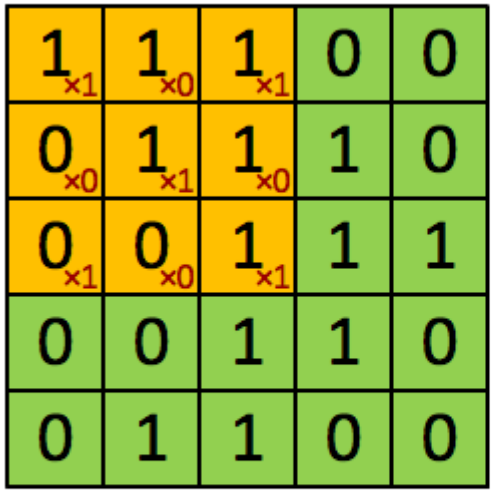
Convolution Operation

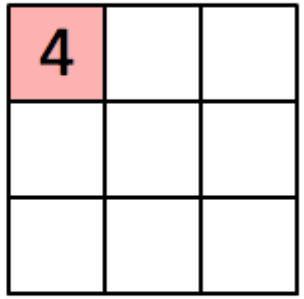
- Convolution of a 2-D image with a 2-D kernel:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

feature map = input * kernel

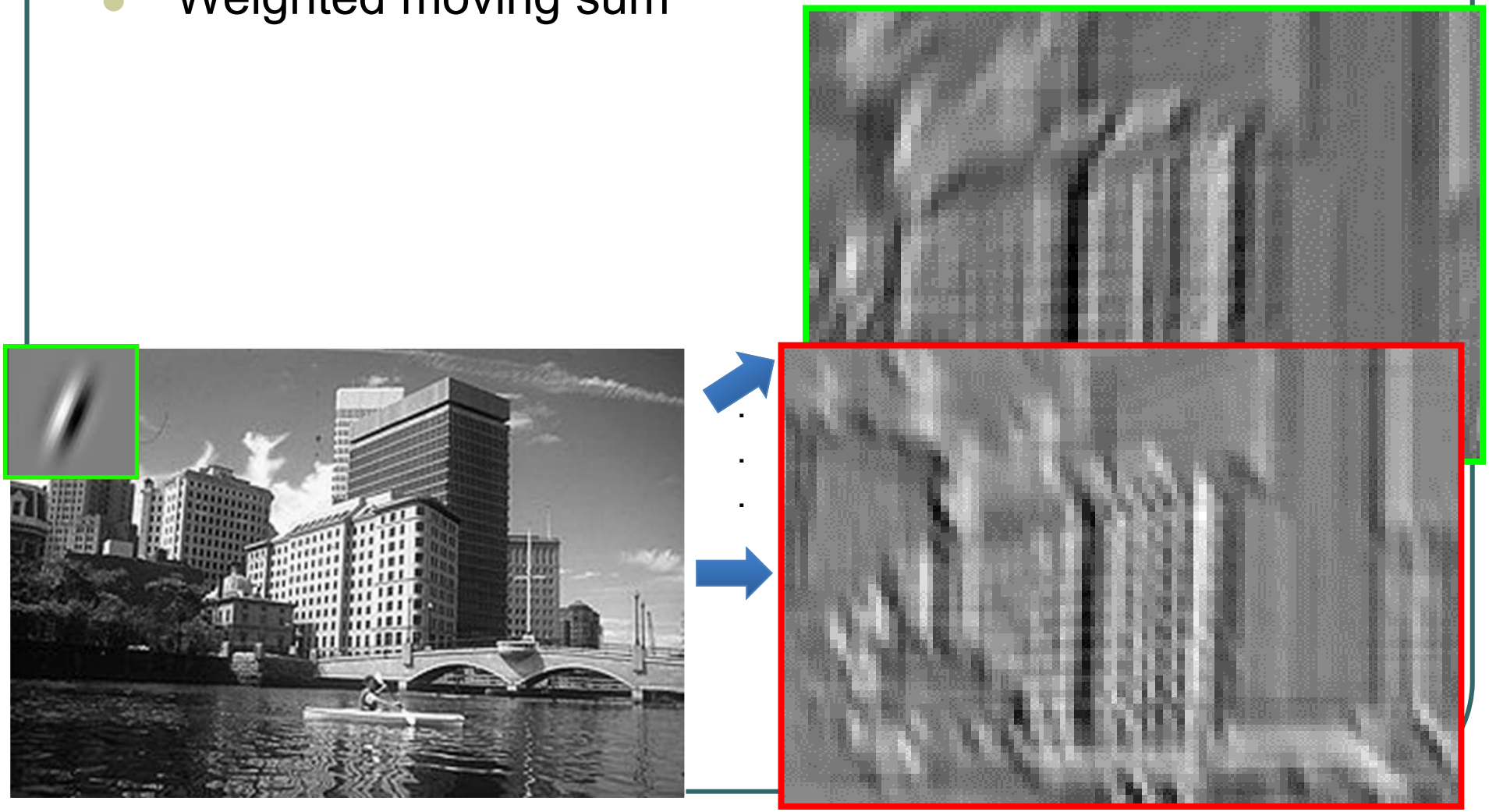






Convolution Operation

- Weighted moving sum



Input

Feature Activation Map

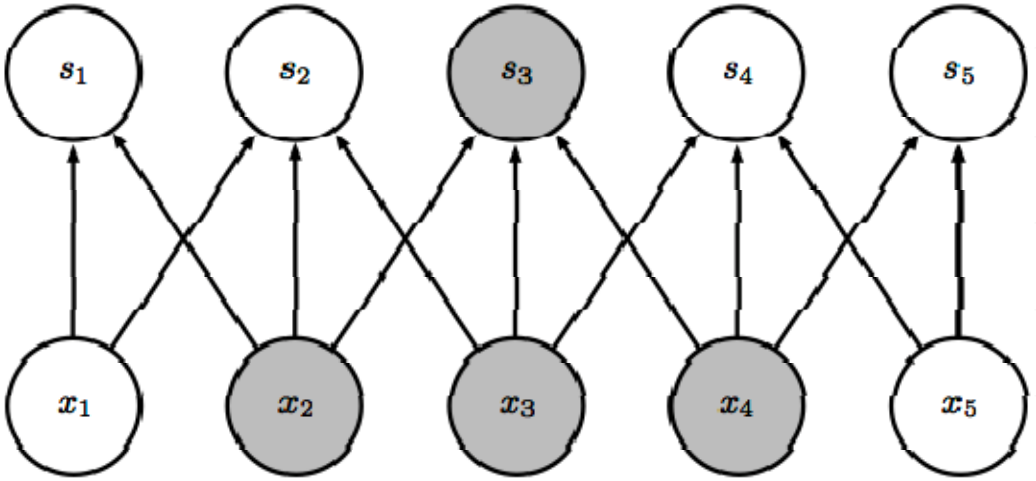
slide credit: S. Lazebnik

Motivation of CNN

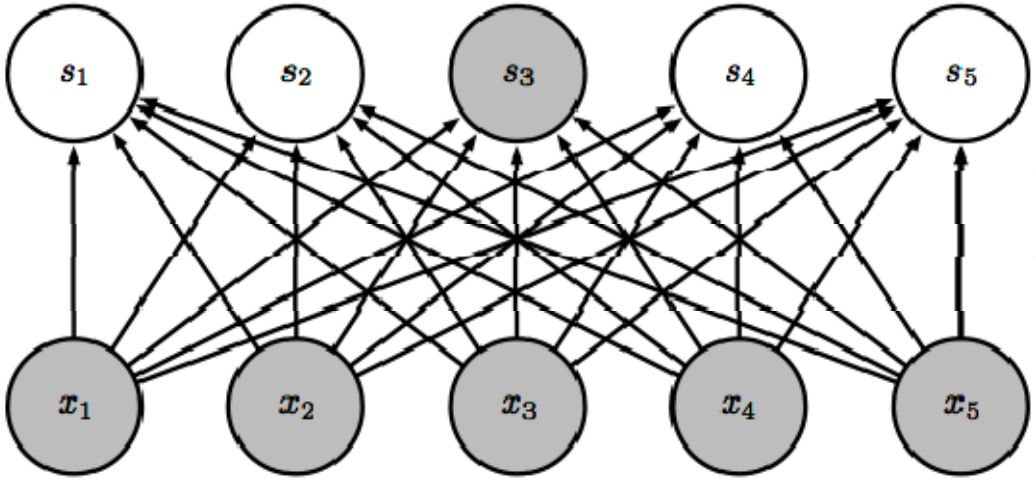
- Scale up neural networks to process very large signals / images / video sequences
- Three essential ideas:
  - Sparse and local connectivity
    - Kernel is usually much smaller than input
    - Less memory and computation required
  - Parameter (weight) sharing
    - Use the same set of parameters for more than one function in a model
  - Equivalent representation
    - Automatically generalize across spatial translations of inputs

CNN: Local Connectivity

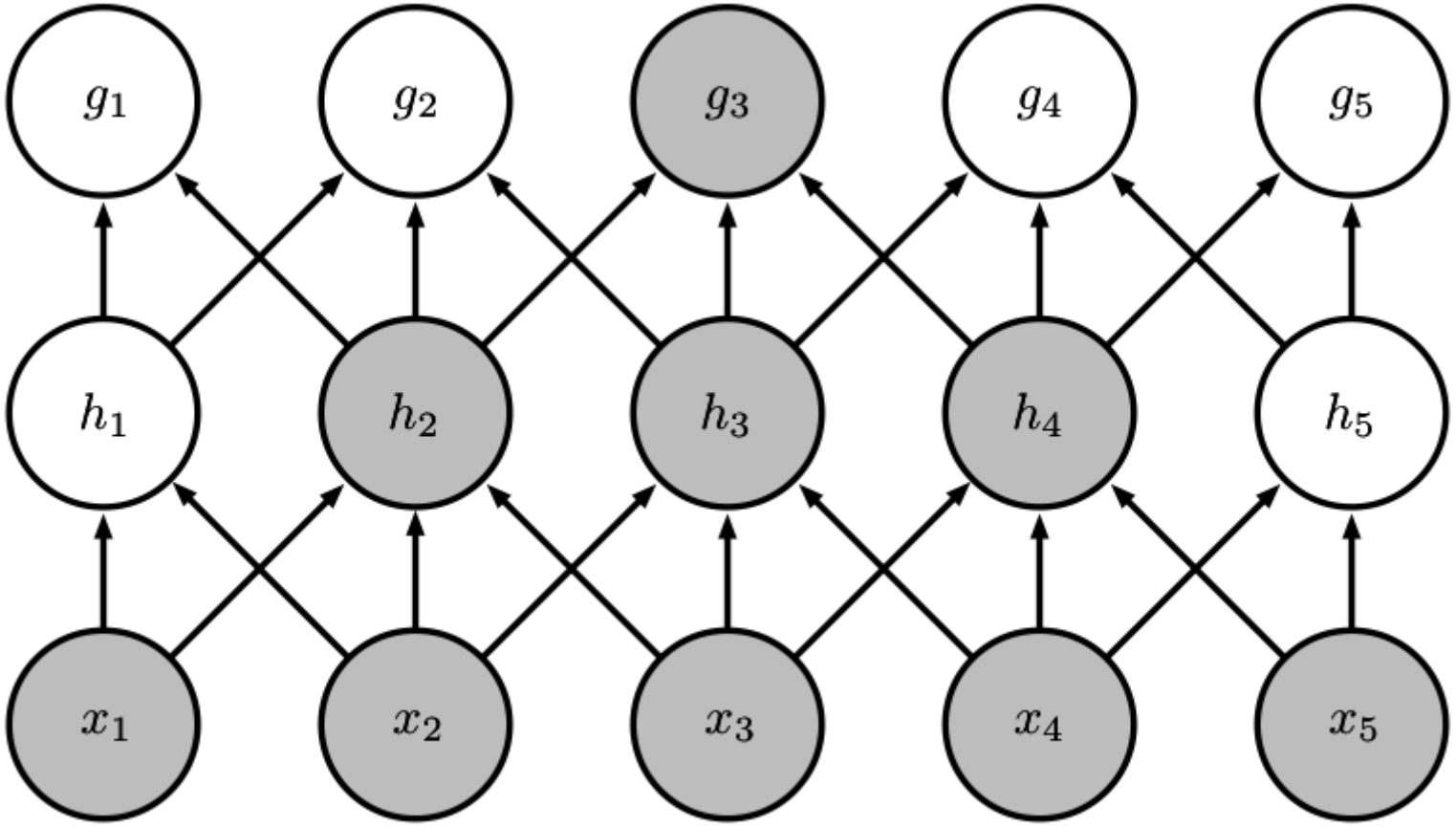
Sparse connections due to small convolution kernel



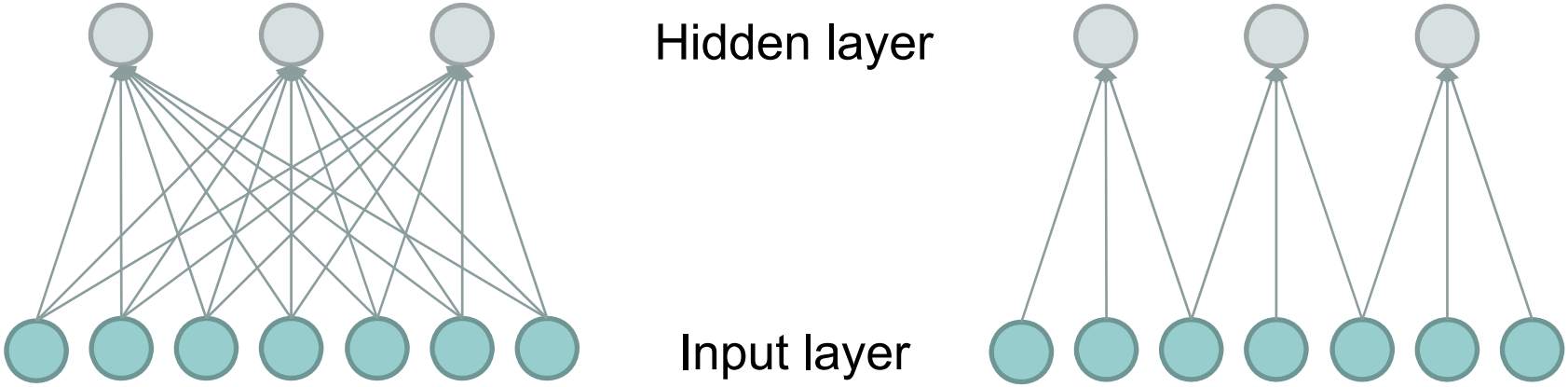
Dense connections



CNN: Growing Receptive Fields



CNN: Local Connectivity

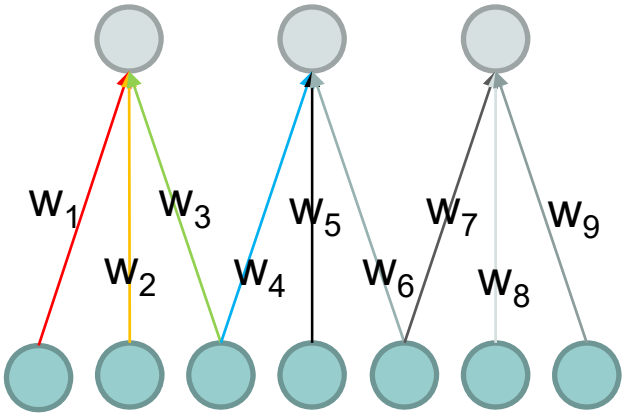


Global connectivity

Local connectivity

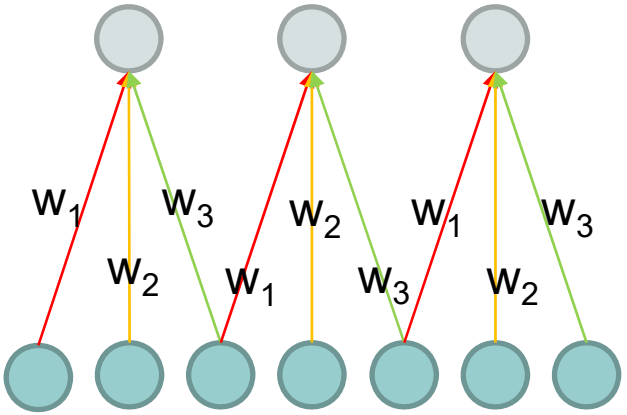
- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Global connectivity: 3 x 7 = 21
  - Local connectivity: 3 x 3 = 9

CNN: Weight Sharing



Hidden layer

Input layer



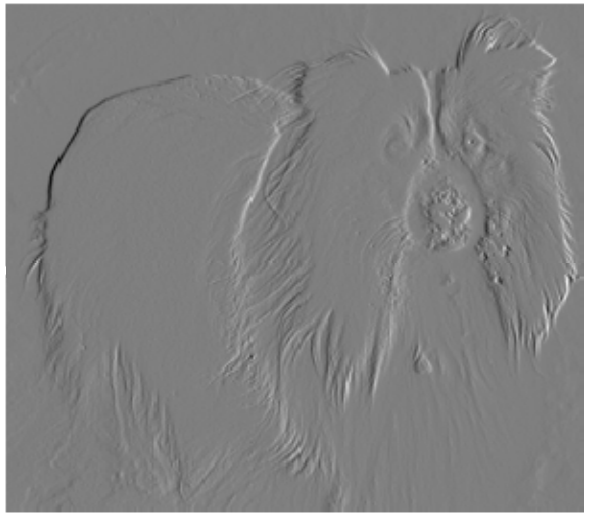
Without weight sharing

With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Without weight sharing: 3 x 3 = 9
  - With weight sharing : 3 x 1 = 3

Edge Detection by Convolution

Input



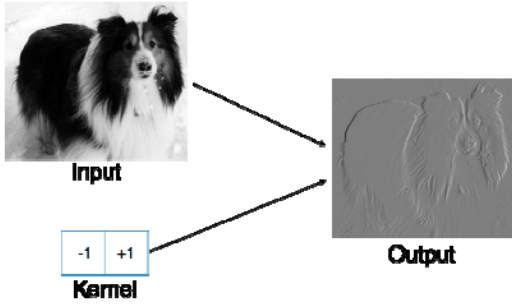
Output

-1	+1

Kernel

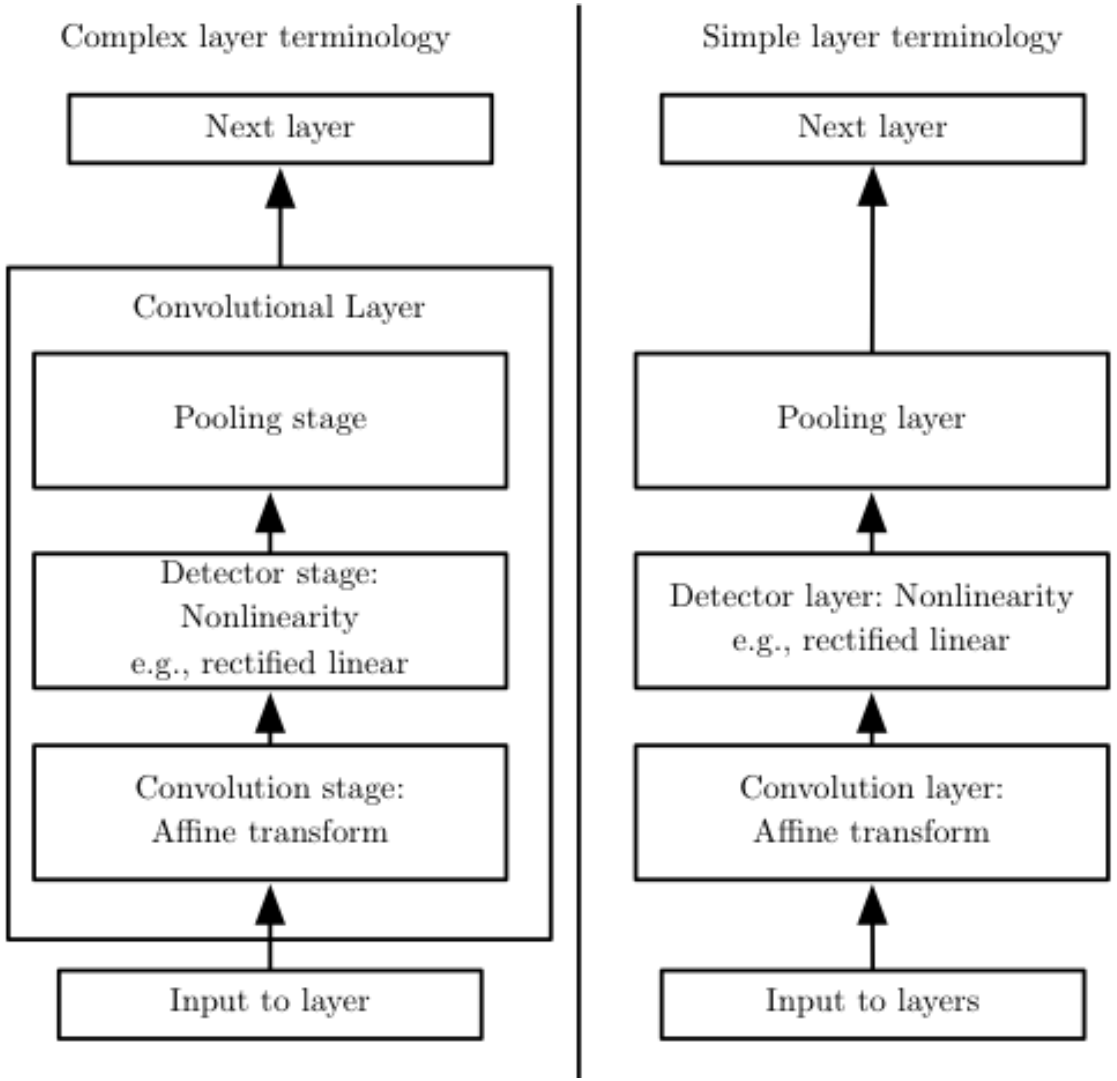
Efficiency of Convolution

Input size: 320 by 280
Kernel size: 2 by 1
Output size: 319 by 280



	Convolution	Dense matrix	Sparse matrix
Stored floats	2	319*280*320*280 > 8e9	2*319*280 = 178,640
Float muls or adds	319*280*3 = 267,960	> 16e9	Same as convolution (267,960)

Components of CNN

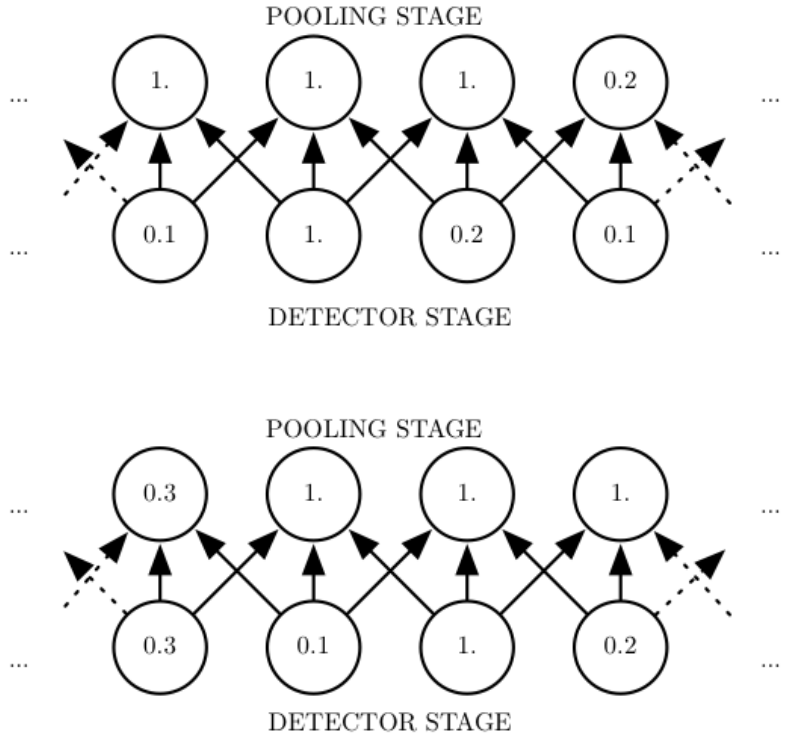


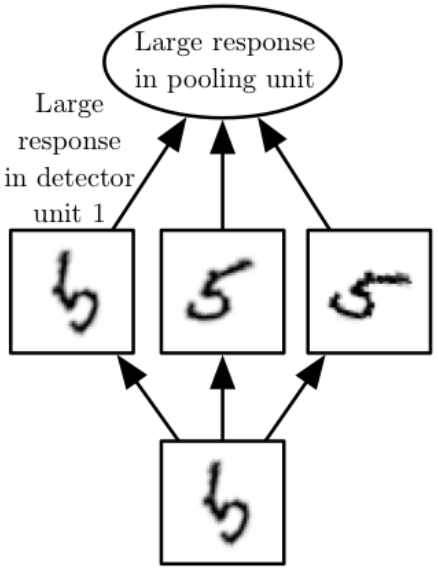
Pooling

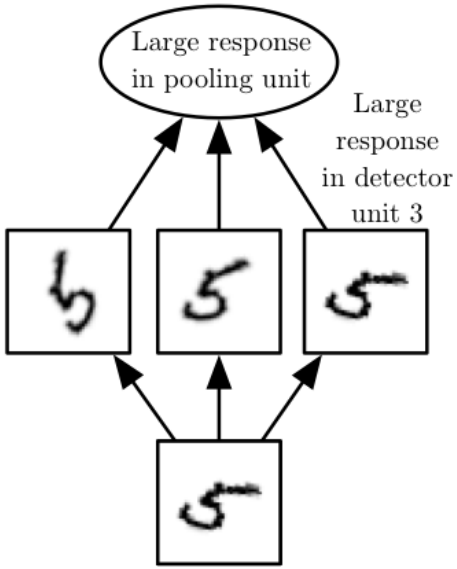
- A pooling function replaces the output of the net with a summary statistic of the nearby outputs.
- Examples of pooing functions:
  - Maximum within a rectangular neighborhood
  - Average of a rectangular neighborhood
  - L^2 norm of a rectangular neighborhood
  - Weighted average based on the distance from the central pixel
- Can use fewer pooling units than detector units
- Can help to handle inputs of varying size

Pooling

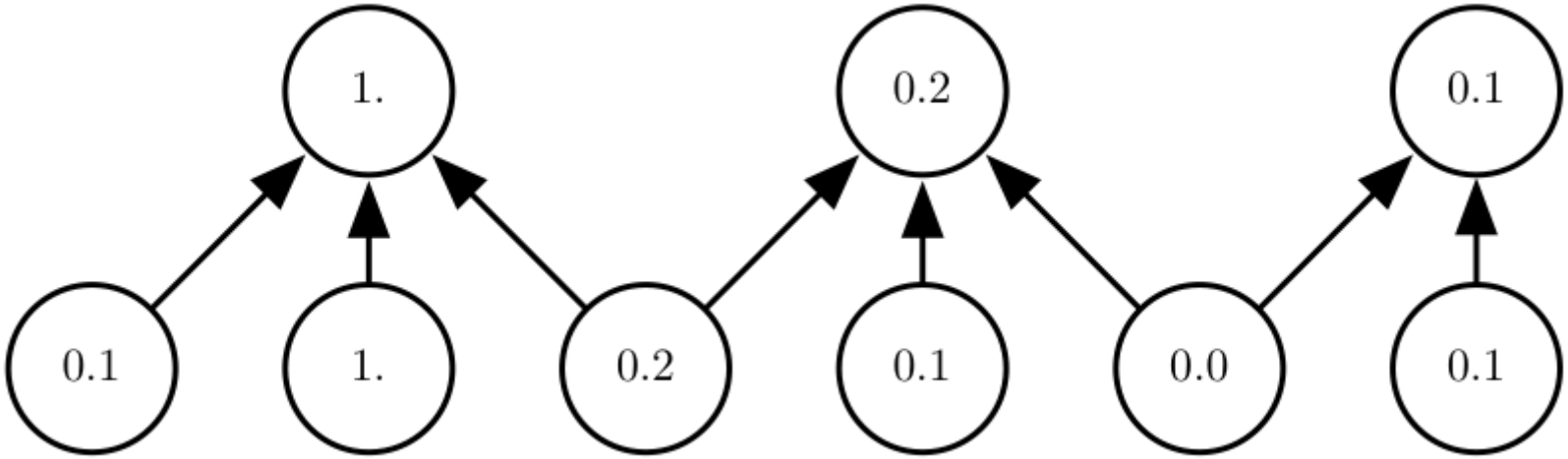
- Pooling helps to make the representation become approximately invariant to small translation and rotation.





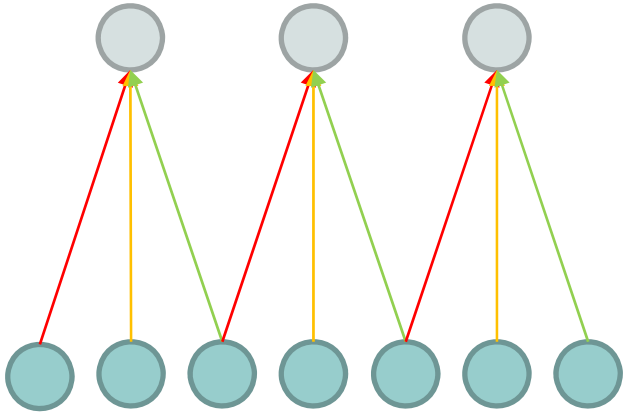


Pooling with Downsampling



- Max-pooling
  - Pooling width: 3
  - Stride between pools: 2

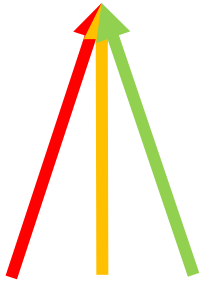
CNN with Multiple Input Channels



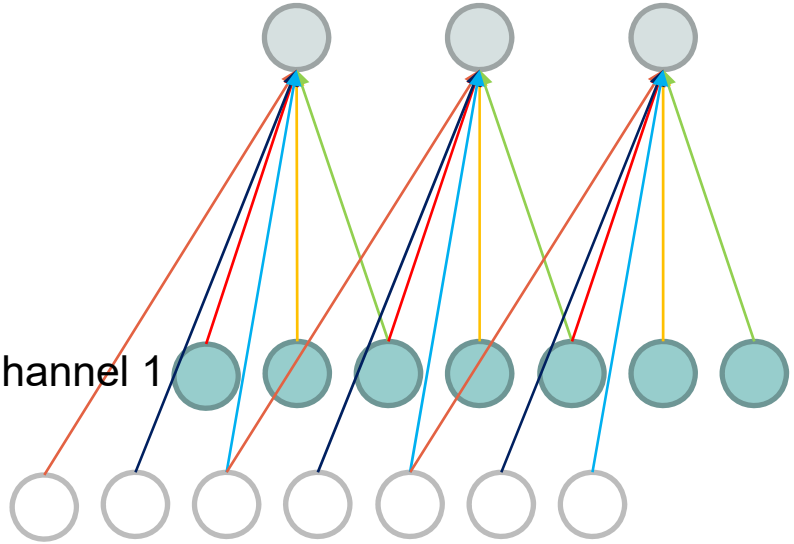
Hidden layer

Input layer

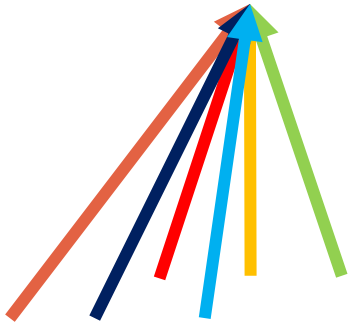
Single input channel



Filter weights

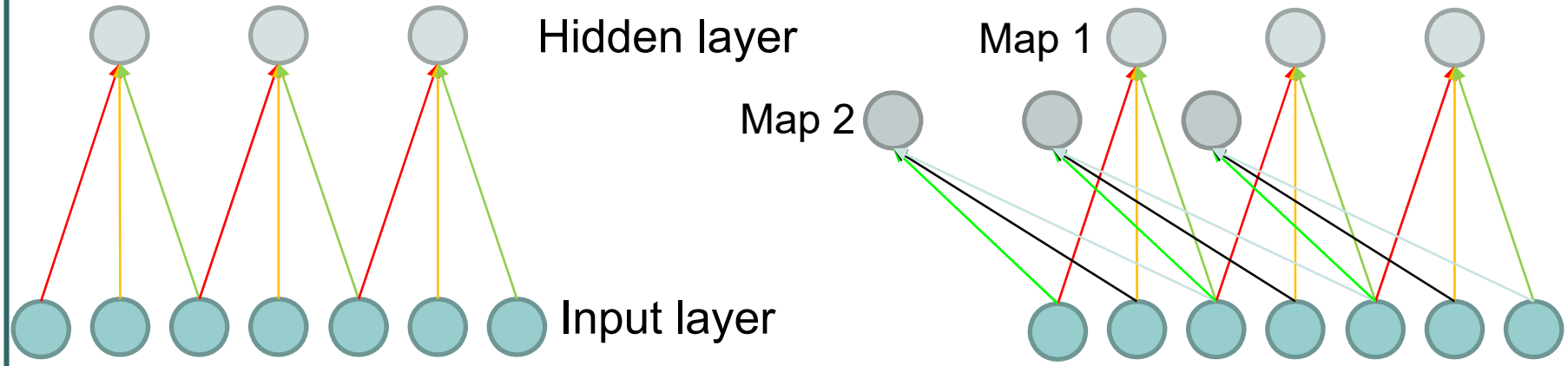


Multiple input channels



Filter weights

CNN with Multiple Output Maps



Single output map

Multiple output maps

Multi-channel Convolution

- For color images, input/output of convolution are 3-D tensors and kernels are 4-D tensors
- Input ***V***: $V_{l,j,k}$, channel *l* at row *j*, column *k*
- Output ***Z***: $Z_{i,j,k}$, channel *i* at row *j*, column *k*
- *Kernel* ***K***: $K_{i,l,m,n}$, channel *i* (output), channel *l* (input)

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

Multi-channel Convolution

- Local connectivity
- Weight sharing
- Handling multiple input channels
- Handling multiple output maps

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

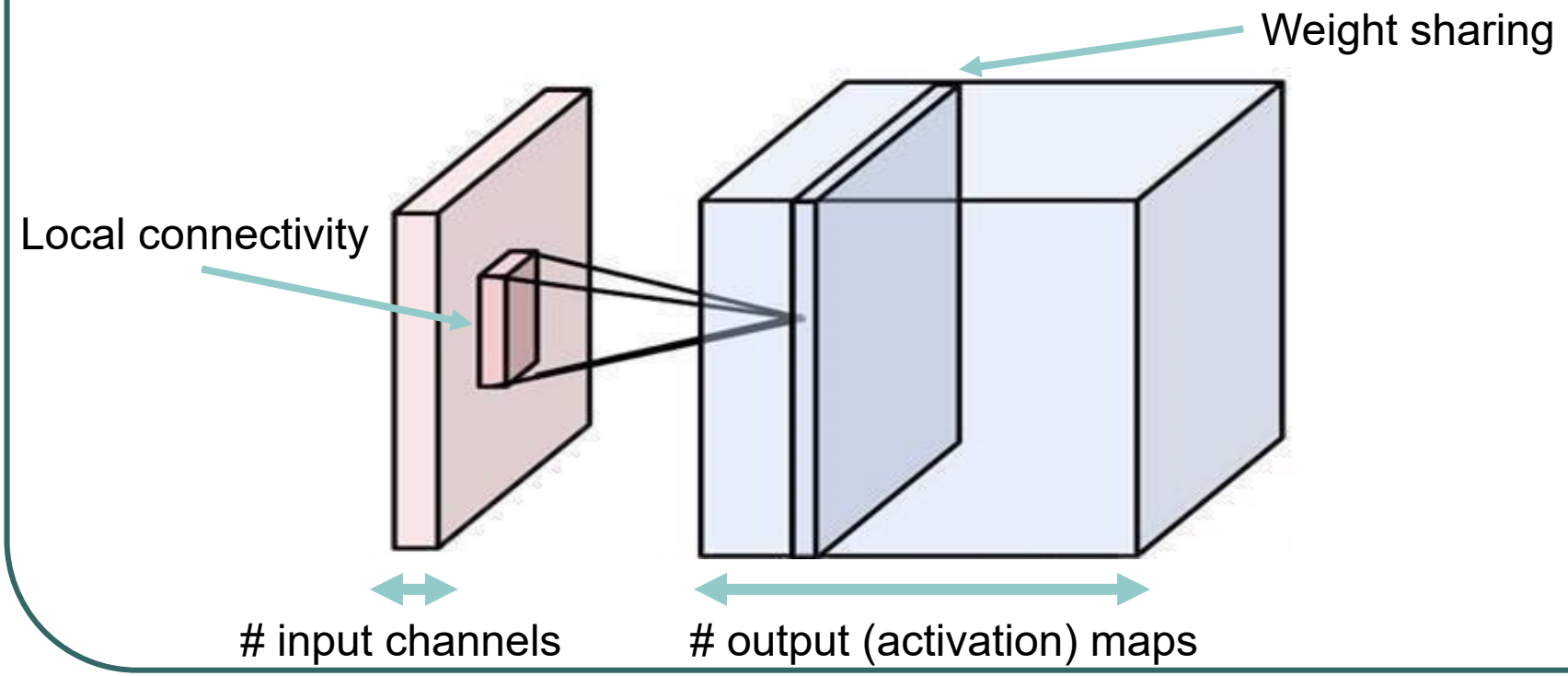
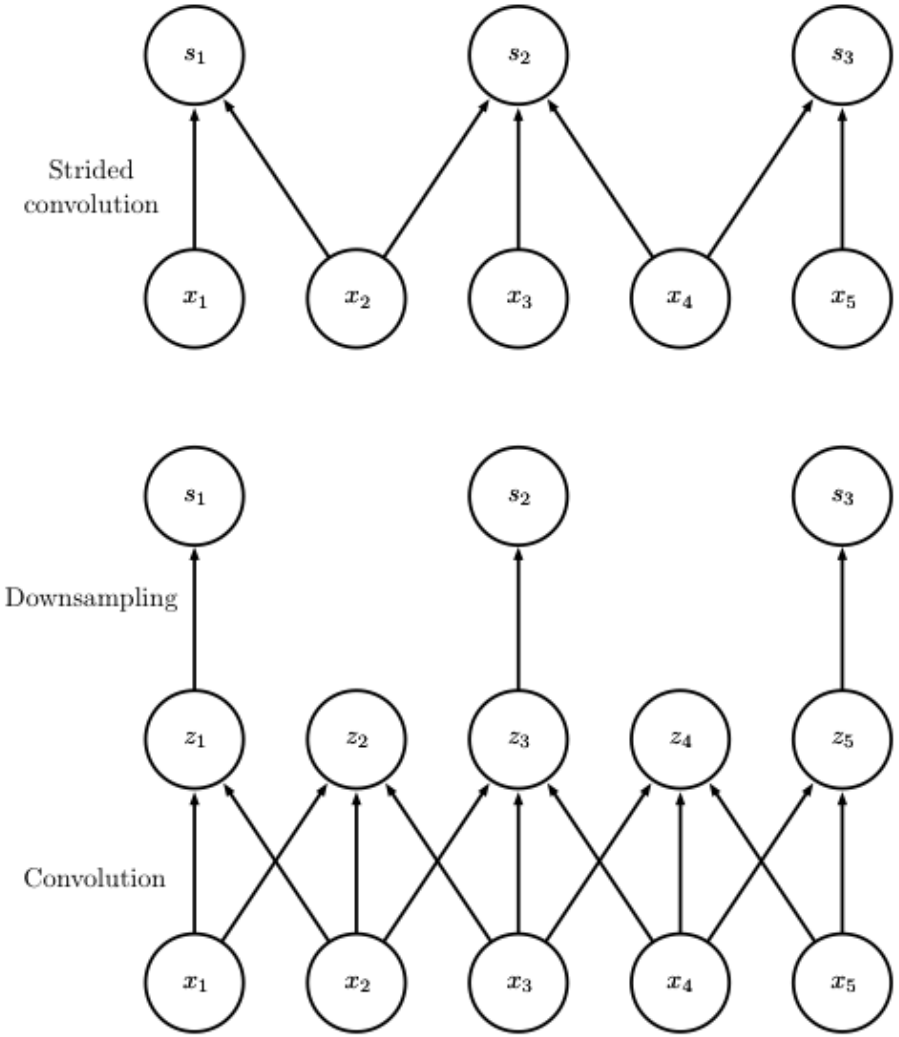


Image credit: A. Karpathy

Convolution with Stride



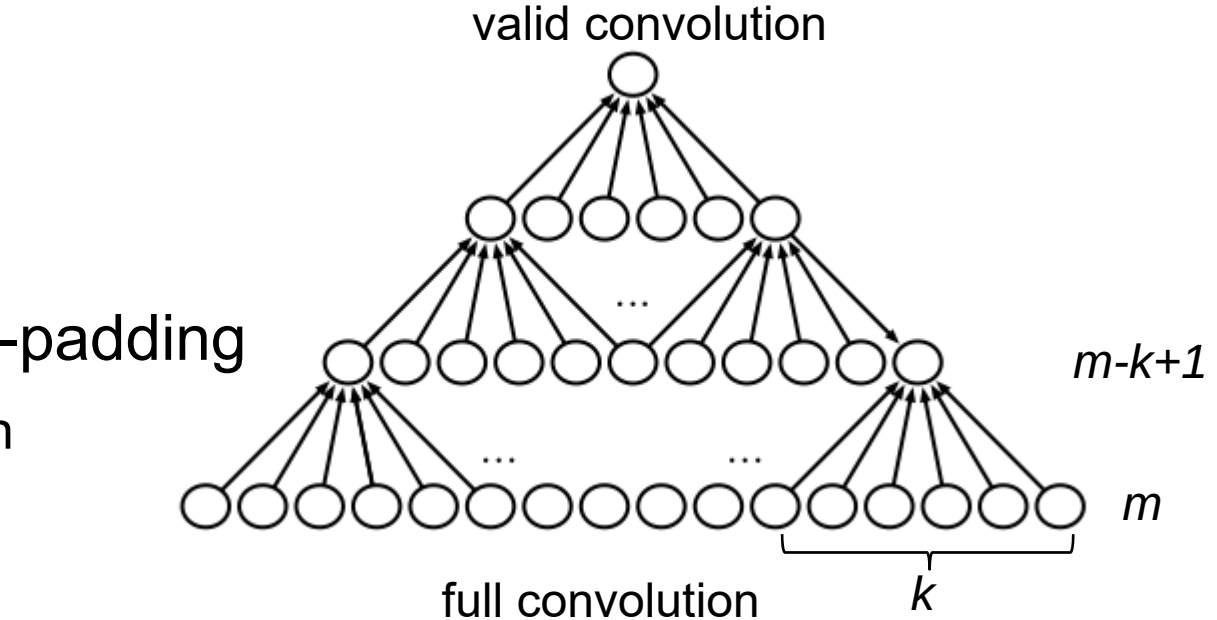
$$Z_{i,j,k} = c(\mathbf{K}, \mathbf{V}, s)_{i,j,k}$$

$$= \sum_{l,m,n} V_{l,(j-1)s+m,(k-1)s+n} K_{i,l,m,n}$$

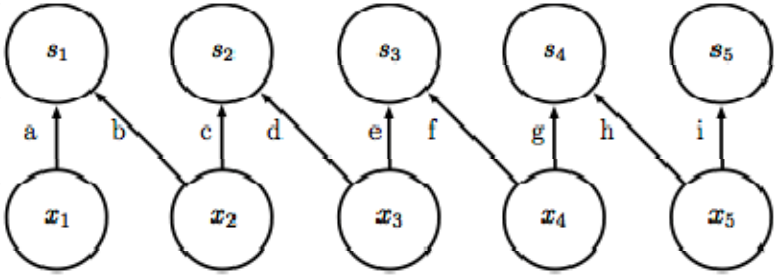
s: stride

Zero-padding Controls Size

- No zero-padding
  - Valid convolution
  - Shrinking size
- Just enough zero-padding
  - Same convolution
  - Keep the size
- Padding with k-1 zeros
  - Full convolution



Local Connections

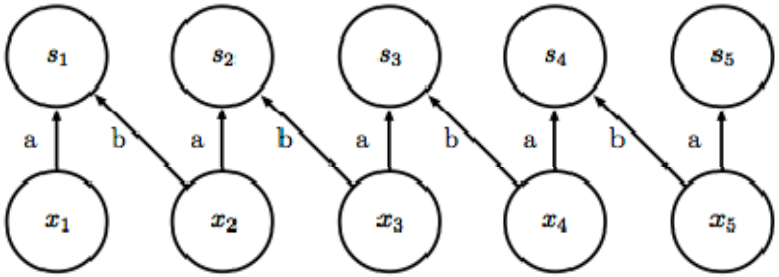


Local connection: like convolution, but no sharing

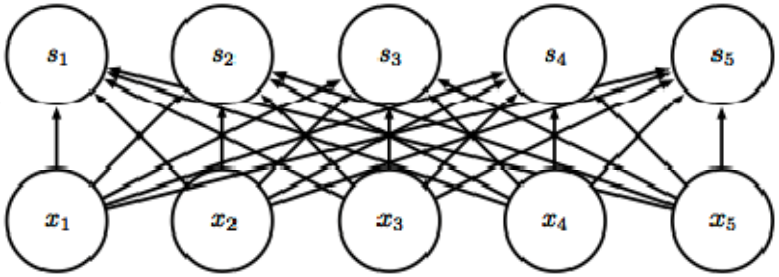
unshared convolution

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,j,k,l,m,n}$$

K: 6D tensor

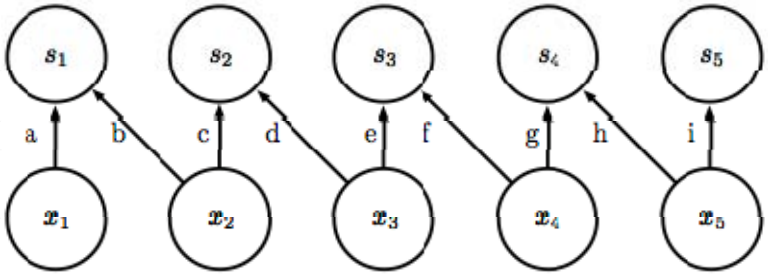


Convolution

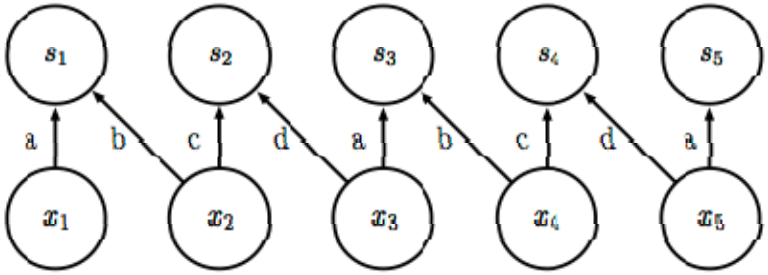


Fully connected

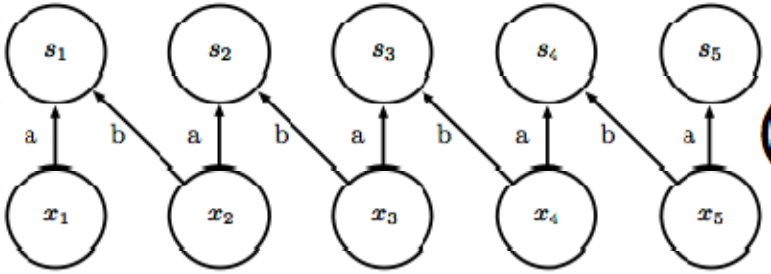
Tiled Convolution



Local connection (no sharing)



Tiled convolution (cycle between groups of shared parameters)

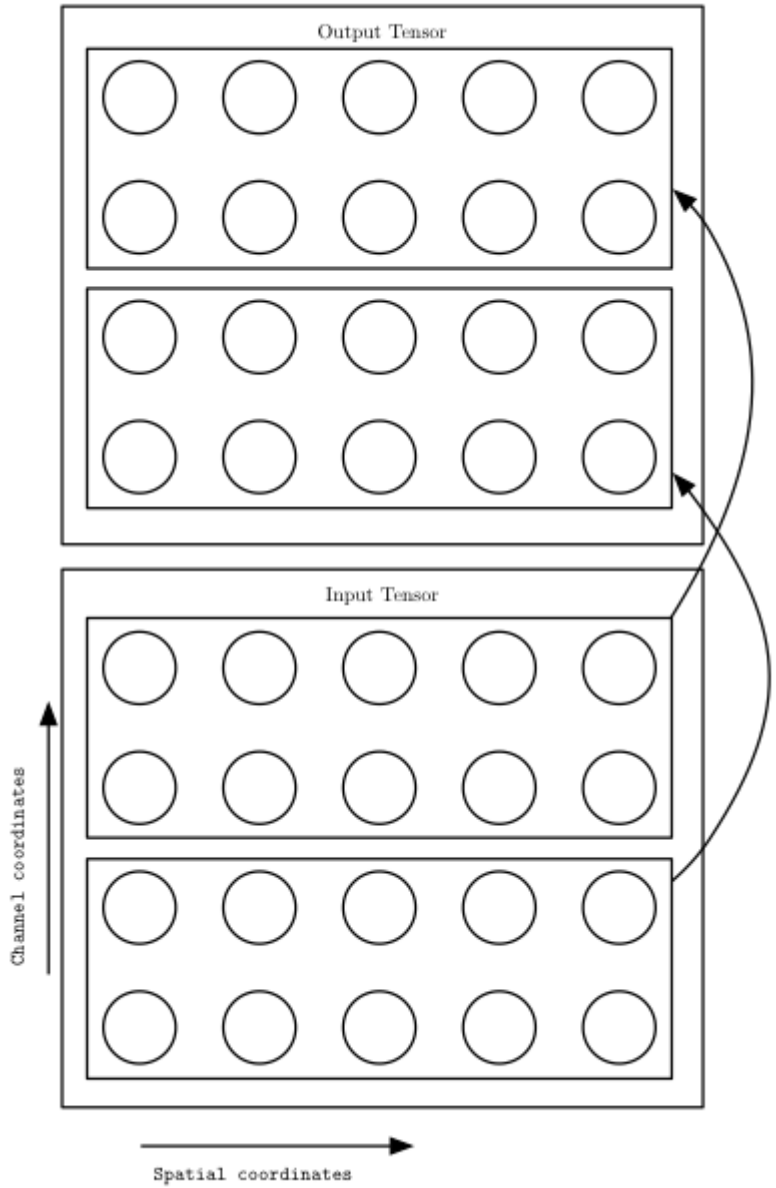


Convolution (one group shared everywhere)

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,j\%t,k\%t,l,m,n}$$

t: number of kernels

Partial Connectivity Between Channels



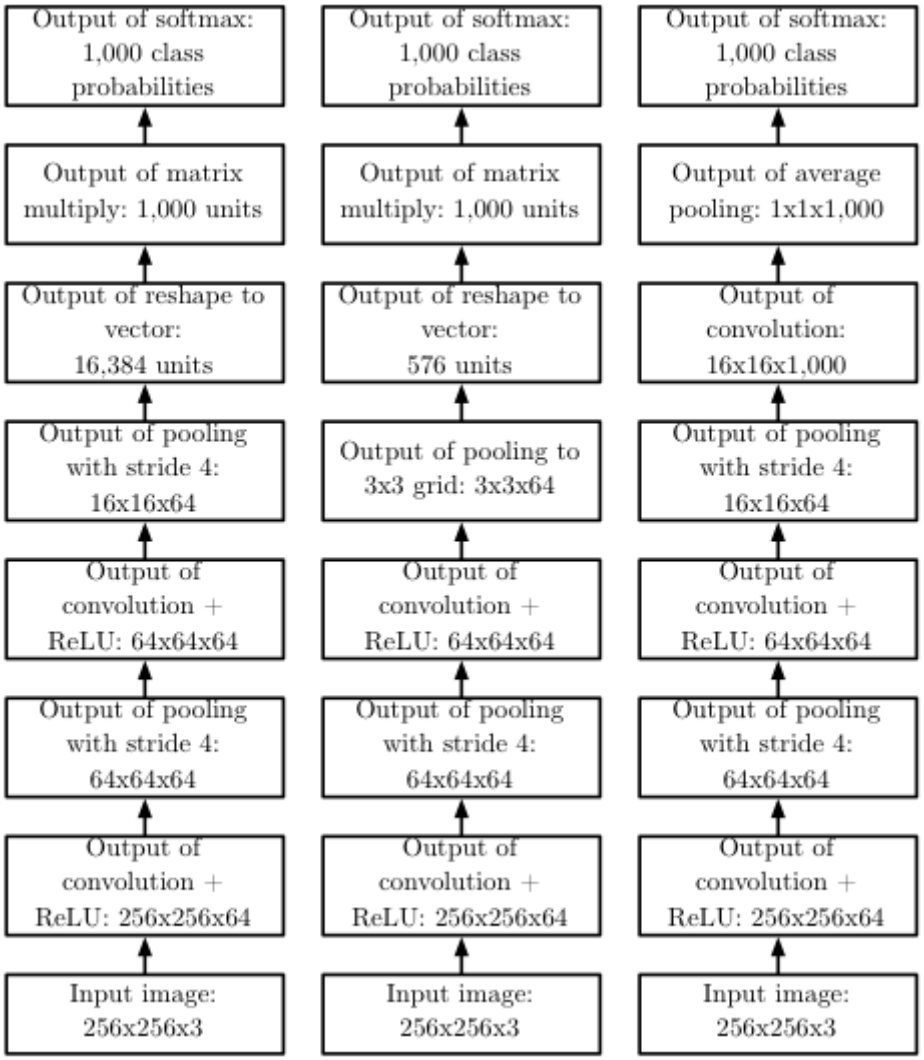
Efficient Convolution Algorithms

- A *d*-dimensional kernel is separable if it can be expressed as the outer product of *d* vectors.
- Convolution with a *d*-dimensional kernel can be decomposed into *d* one-dimensional convolutions.
- Complexity: $O(w^d) \rightarrow O(wd)$

Initialization of Kernels

- Randomly initialized
  - Random filters often work surprisingly well.
- Designed by hand
  - Edge detection, Haar wavelet, etc.
- Learned with an unsupervized criterion
  - K-means clustering

Example Classification Architectures

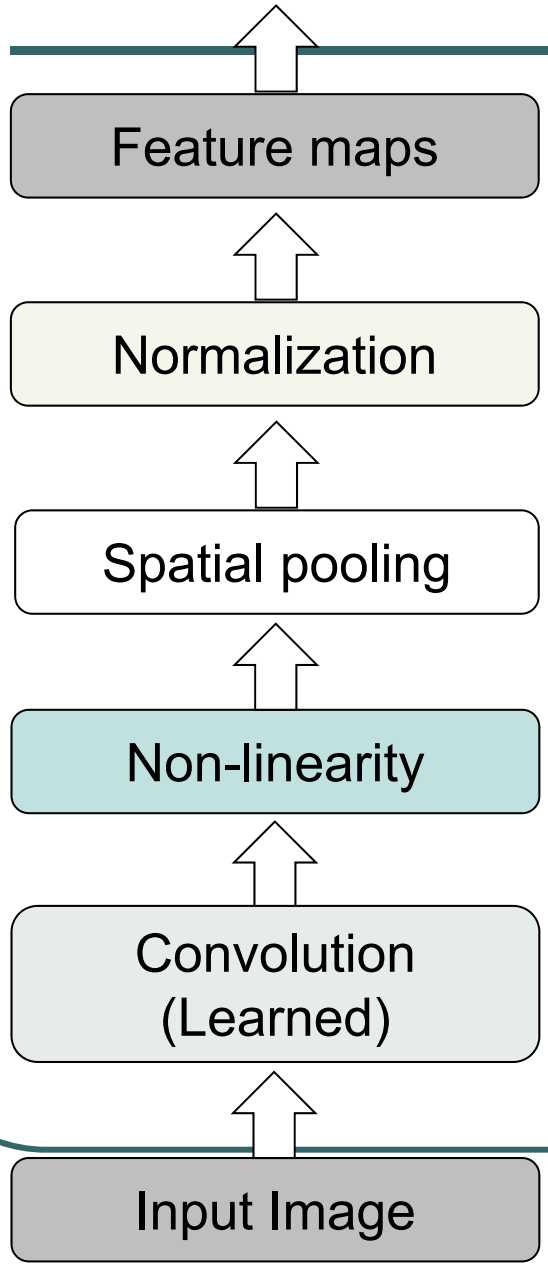


Fixed image size

Fixed number of variably-sized pools

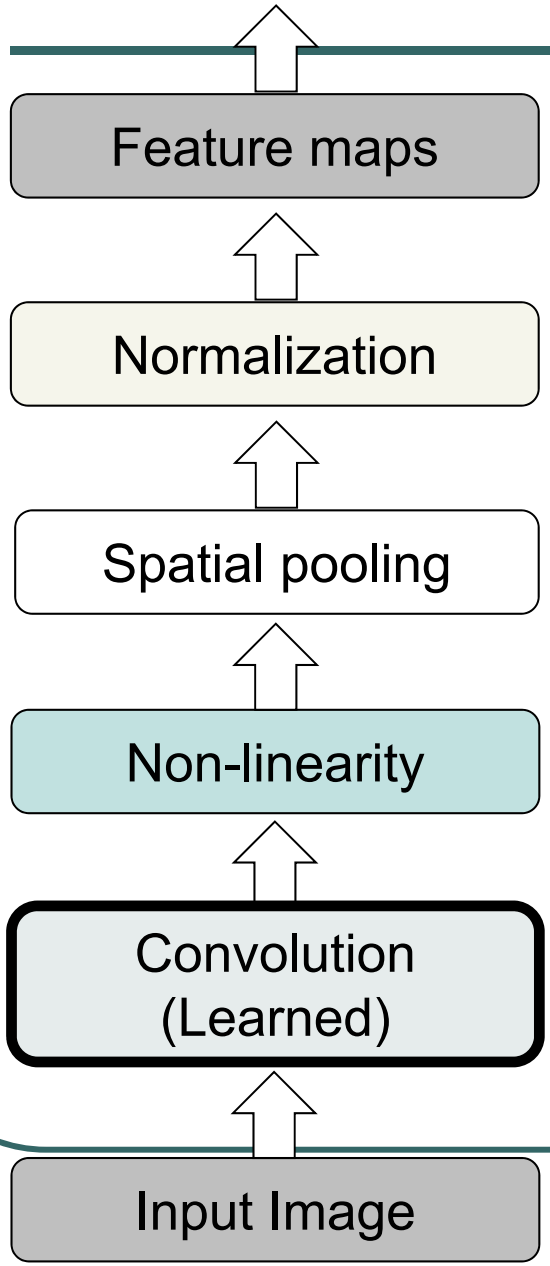
A feature map per class

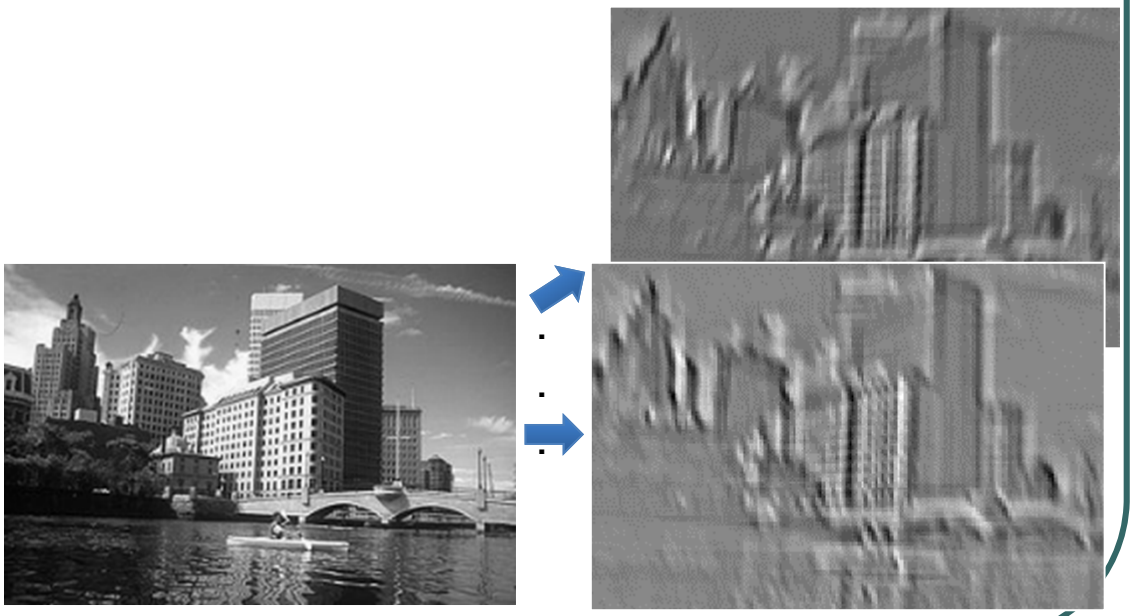
Convolutional Neural Networks



slide credit: S. Lazebnik

Convolutional Neural Networks



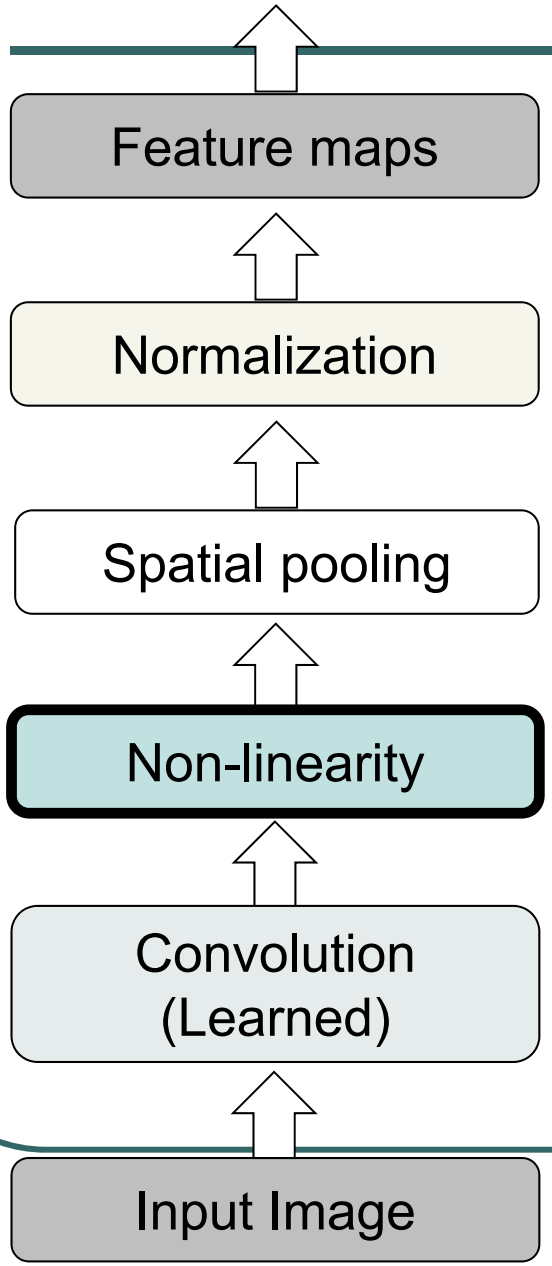


Input

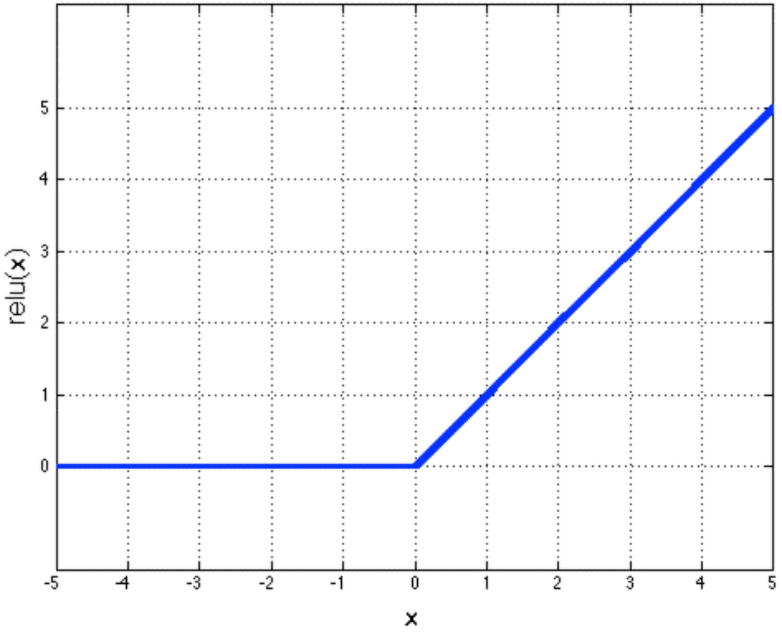
Feature Map

slide credit: S. Lazebnik

Convolutional Neural Networks

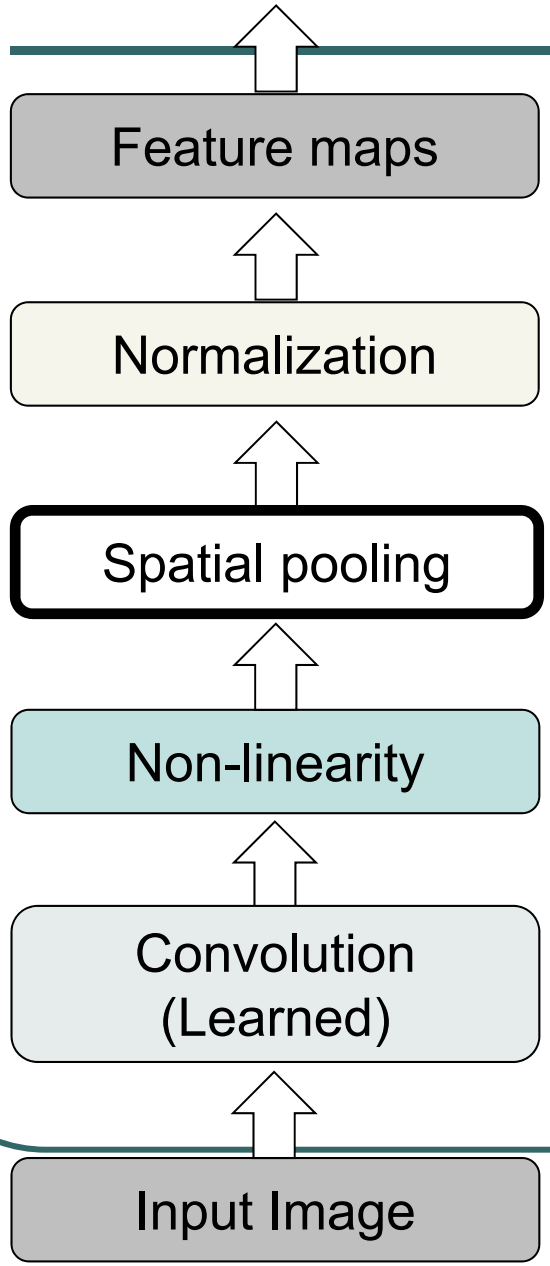


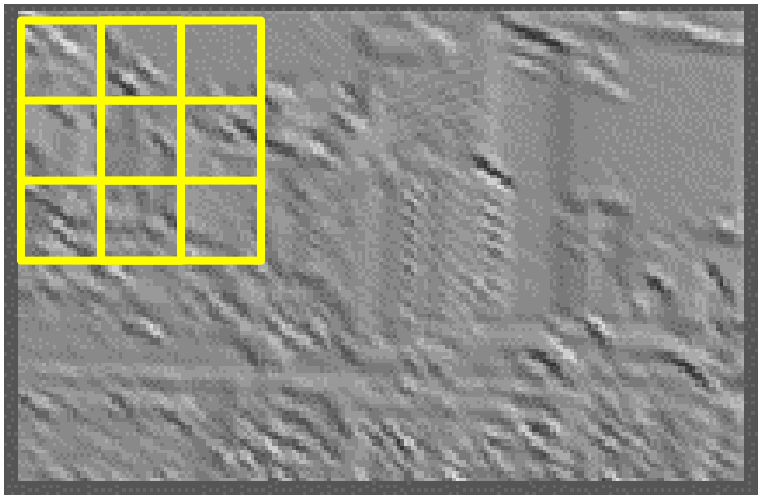
Rectified Linear Unit (ReLU)



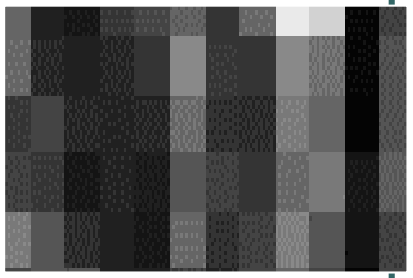
slide credit: S. Lazebnik

Convolutional Neural Networks





Max pooling

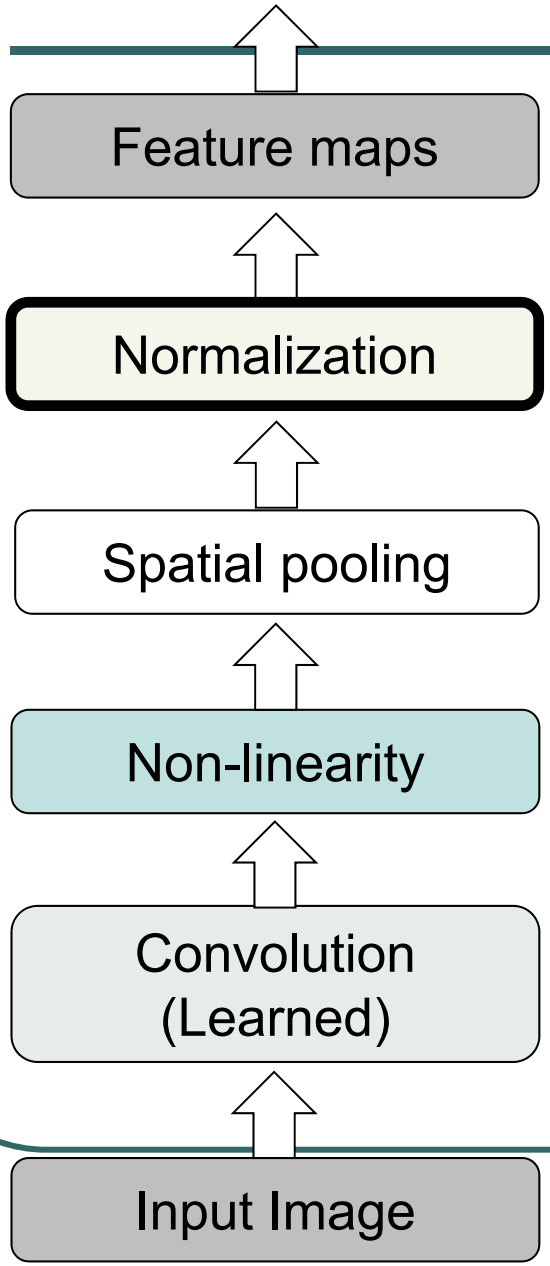


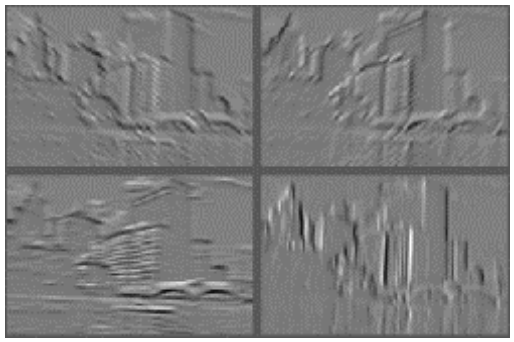
Max-pooling: a non-linear down-sampling

Provide *translation invariance*

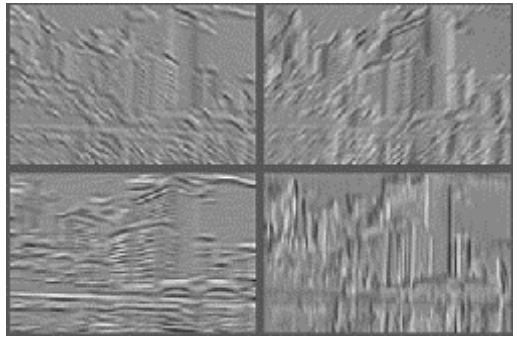
slide credit: S. Lazebnik

Convolutional Neural Networks





Feature Maps

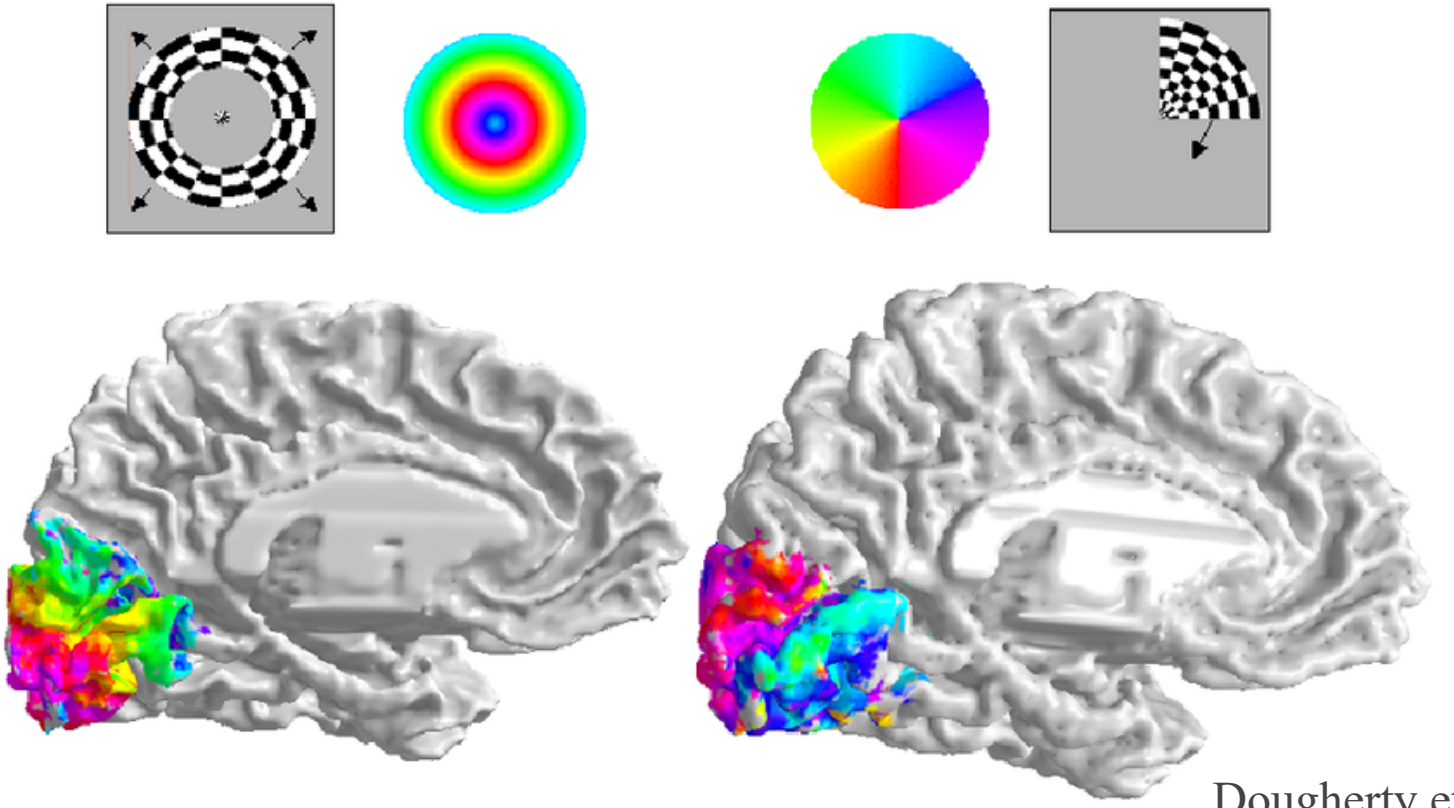


Feature Maps
After Contrast
Normalization

slide credit: S. Lazebnik

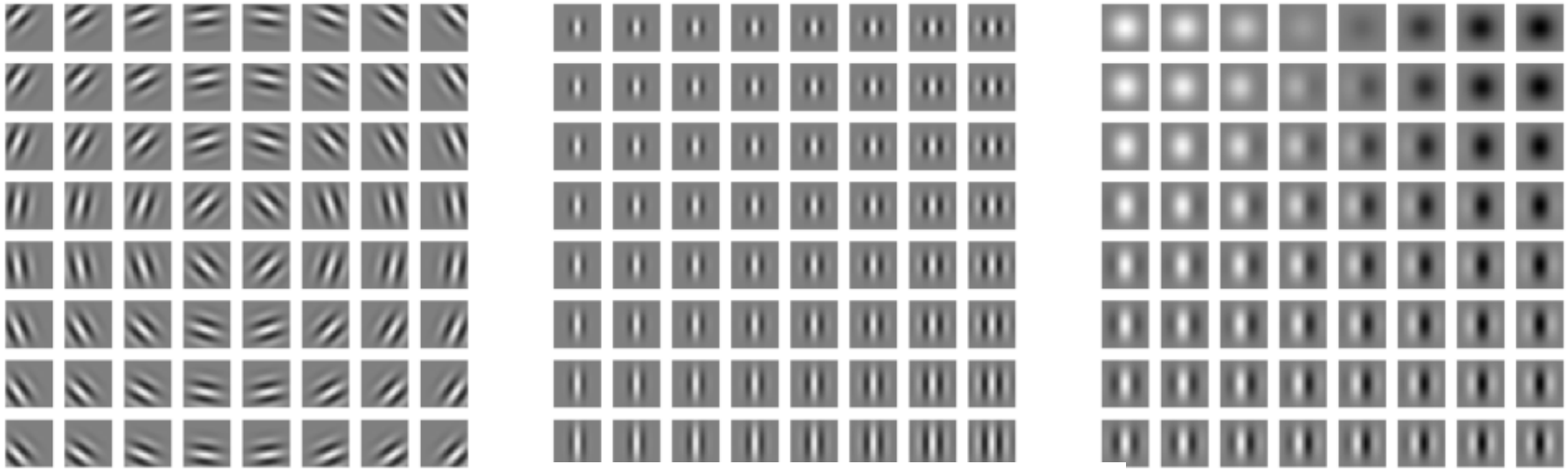
Retinotopy

- V1 is arranged in a spatial map called retinotopy.



Dougherty et al., 2003

Most V1 Cells Have Gabor Weights



$$s(I) = \sum_{x \in \mathbb{X}} \sum_{y \in \mathbb{Y}} w(x,y) I(x,y)$$

$$w(x, y; \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \exp\left(-\beta_x x'^2 - \beta_y y'^2\right) \cos(f x' + \phi),$$

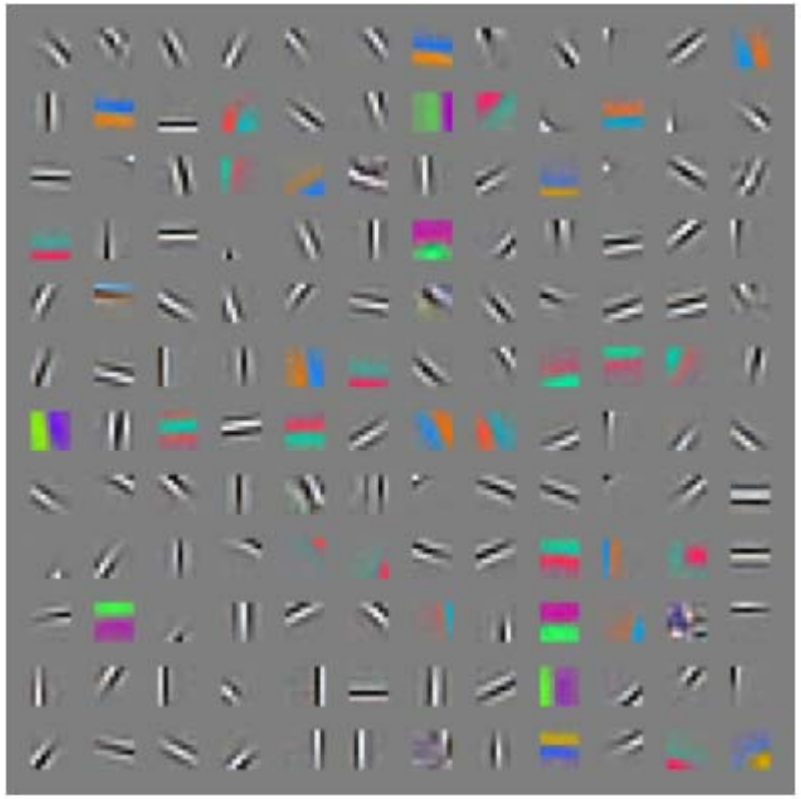
where

$$x' = (x - x_0)\cos(\tau) + (y - y_0)\sin(\tau)$$
$$y' = -(x - x_0)\sin(\tau) + (y - y_0)\cos(\tau)$$

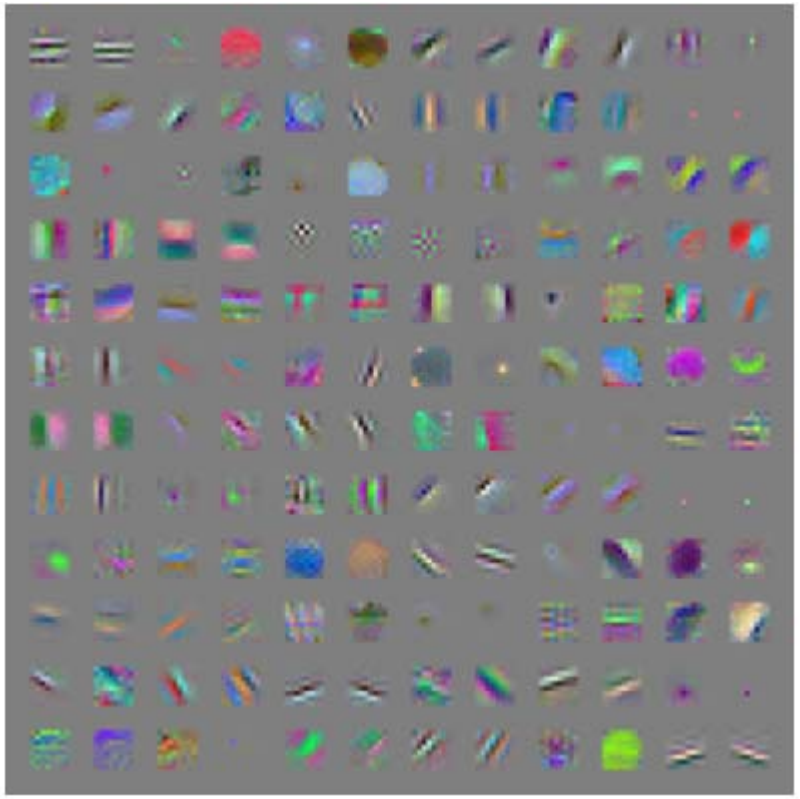
Gabor-like Learned Kernels

- Obtained by:

  unsupervised learning



  first layer of a fully supervised convolutional maxout network



Neocognitron [Fukushima, Biological Cybernetics 1980]

V1 contains many simple cells and complex cells [Hubel & Wiesel,1959].

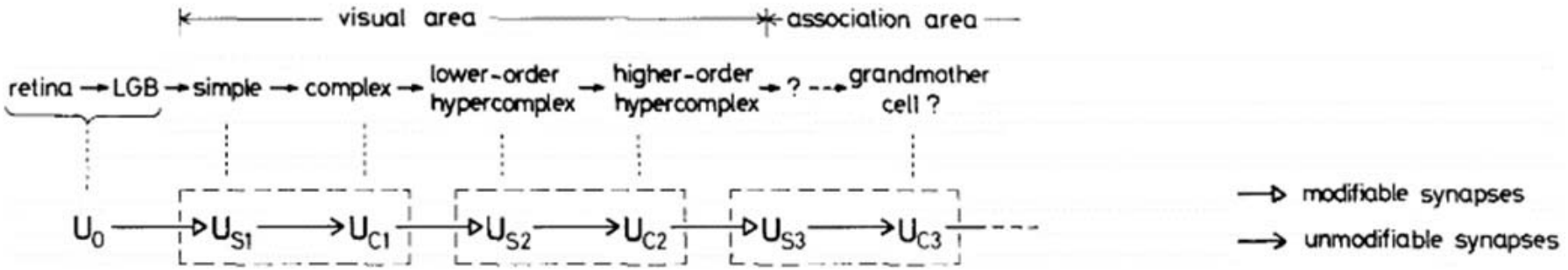
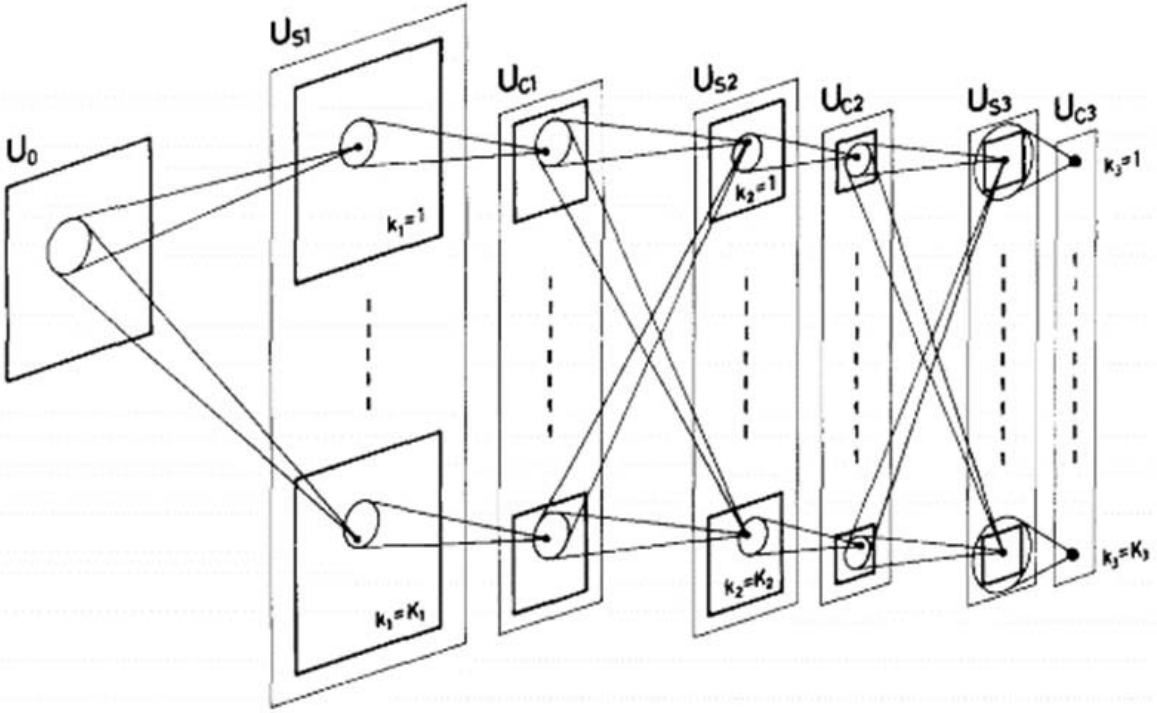


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

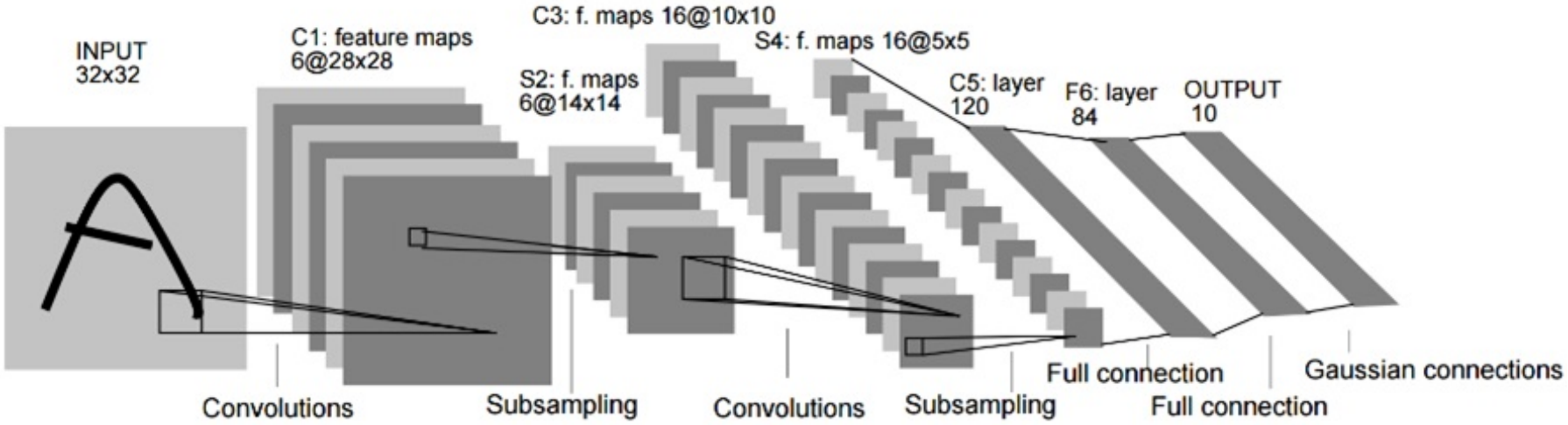


Deformation-Resistant Recognition

S-cells: (simple)
- extract local features

C-cells: (complex)
- allow for positional errors

LeNet [LeCun et al. 1998]

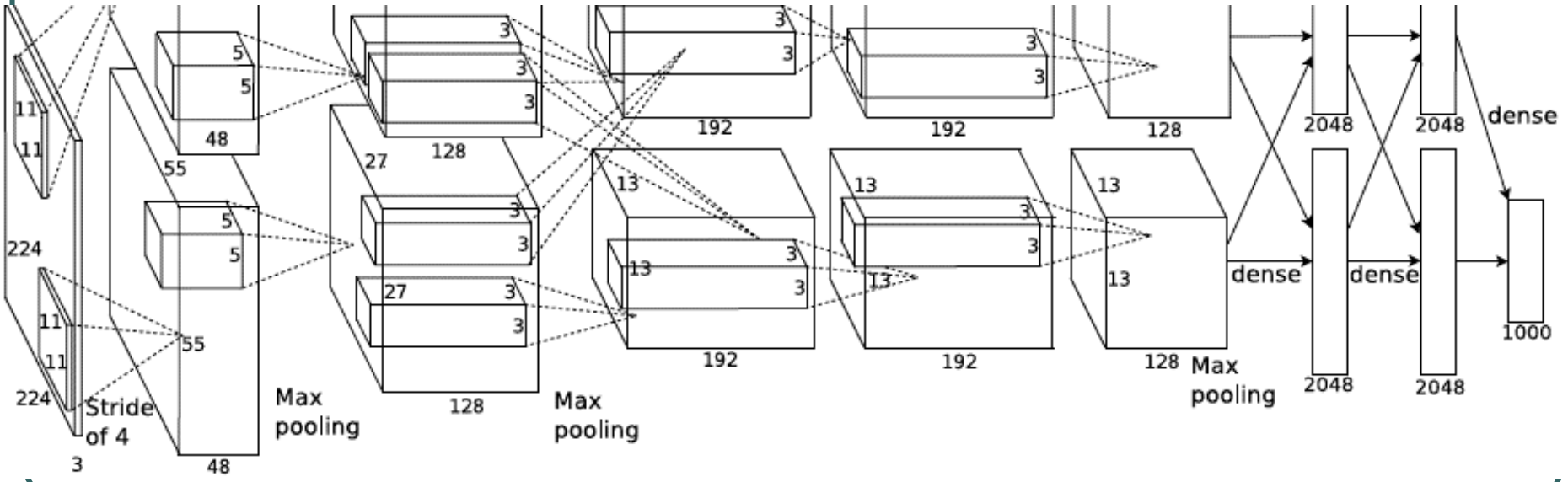


Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner 1998]

LeNet-1 from 1993

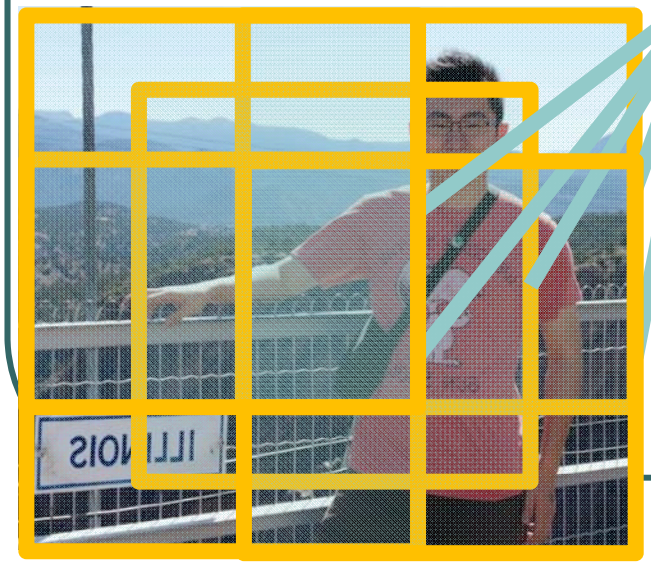
AlexNet

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week

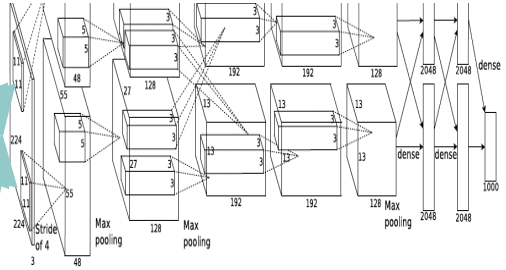


A. Krizhevsky, I. Sutskever, and G. Hinton,
ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Using CNN for Image Classification

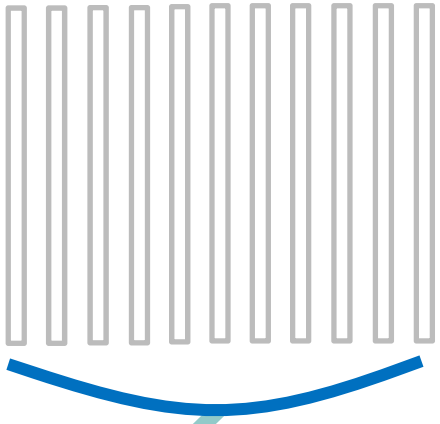


AlexNet



Fixed input size:
224x224x3

Fully connected layer Fc7
d = 4096



Averaging

d = 4096

Softmax Layer

"Jia-Bin"

IMAGENET

- Start in 2007 at Princeton
- Debut in CVPR'2009
- Stop in 2010
  - Total classes : 21K
  - Total images : 14M
- ILSVR Challenge: 2010 - present

Jia Deng

Fei-Fei Li

Info from http://www.image-net.org/

ImageNet Challenge 2012-2014

Best non-convnet in 2012: 26.2%

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

Team	Method	Error (top-5)
DeepImage - Baidu	Data augmentation + multi GPU	5.33%
PReLU-nets - MSRA	Parametric ReLU + smart initialization	4.94%
BN-Inception ensemble - Google	Reducing internal covariate shift	4.82%

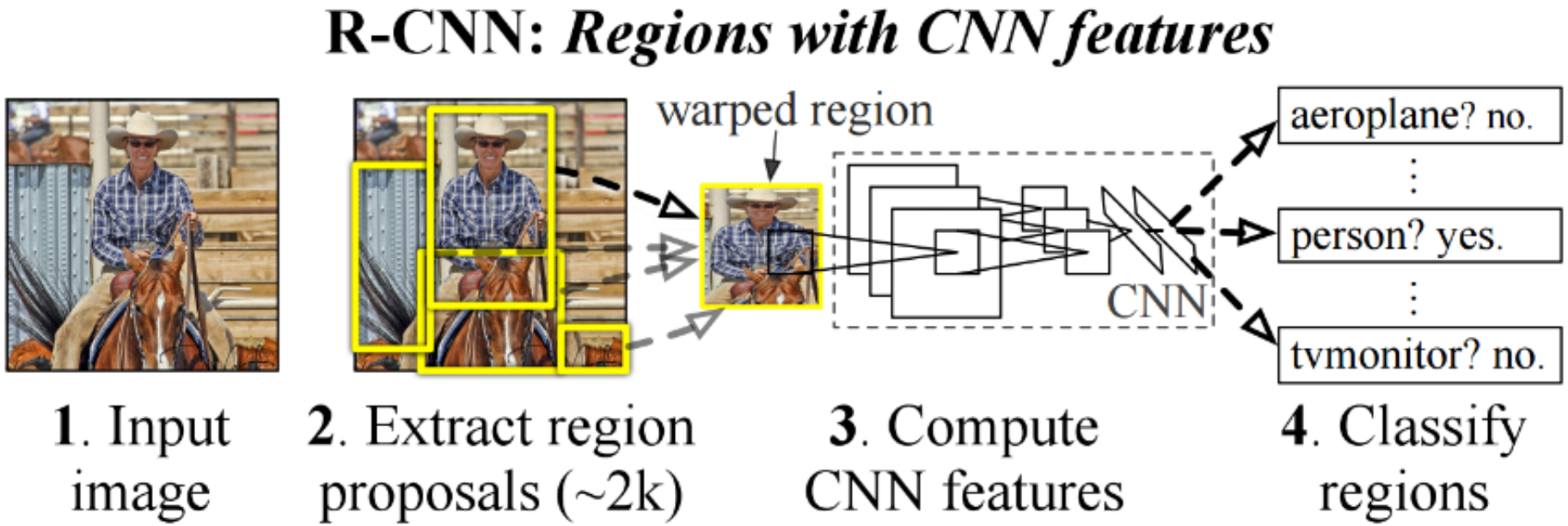
Beyond Classification

- Detection
- Segmentation
- Regression
- Pose estimation
- Matching patches
- Synthesis
- Style transfer

and many more…

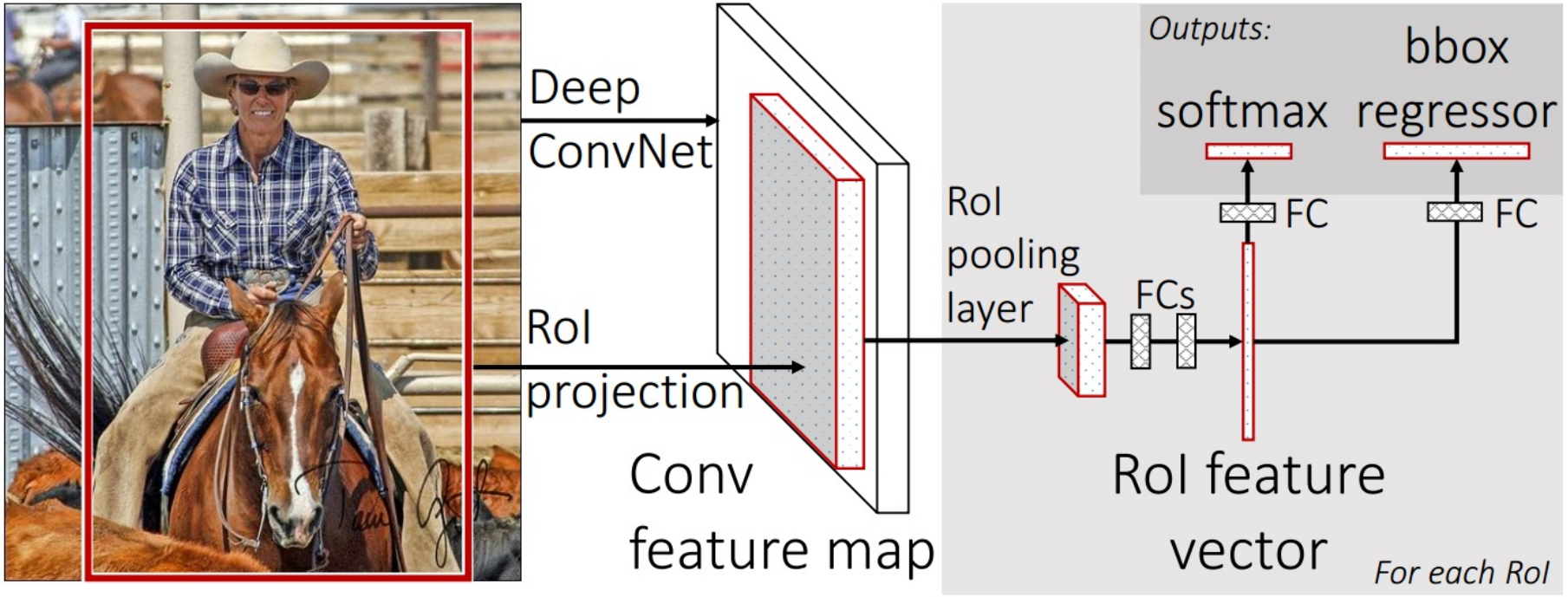
R-CNN: Regions with CNN features

- Trained on ImageNet classification
- Finetune CNN on PASCAL



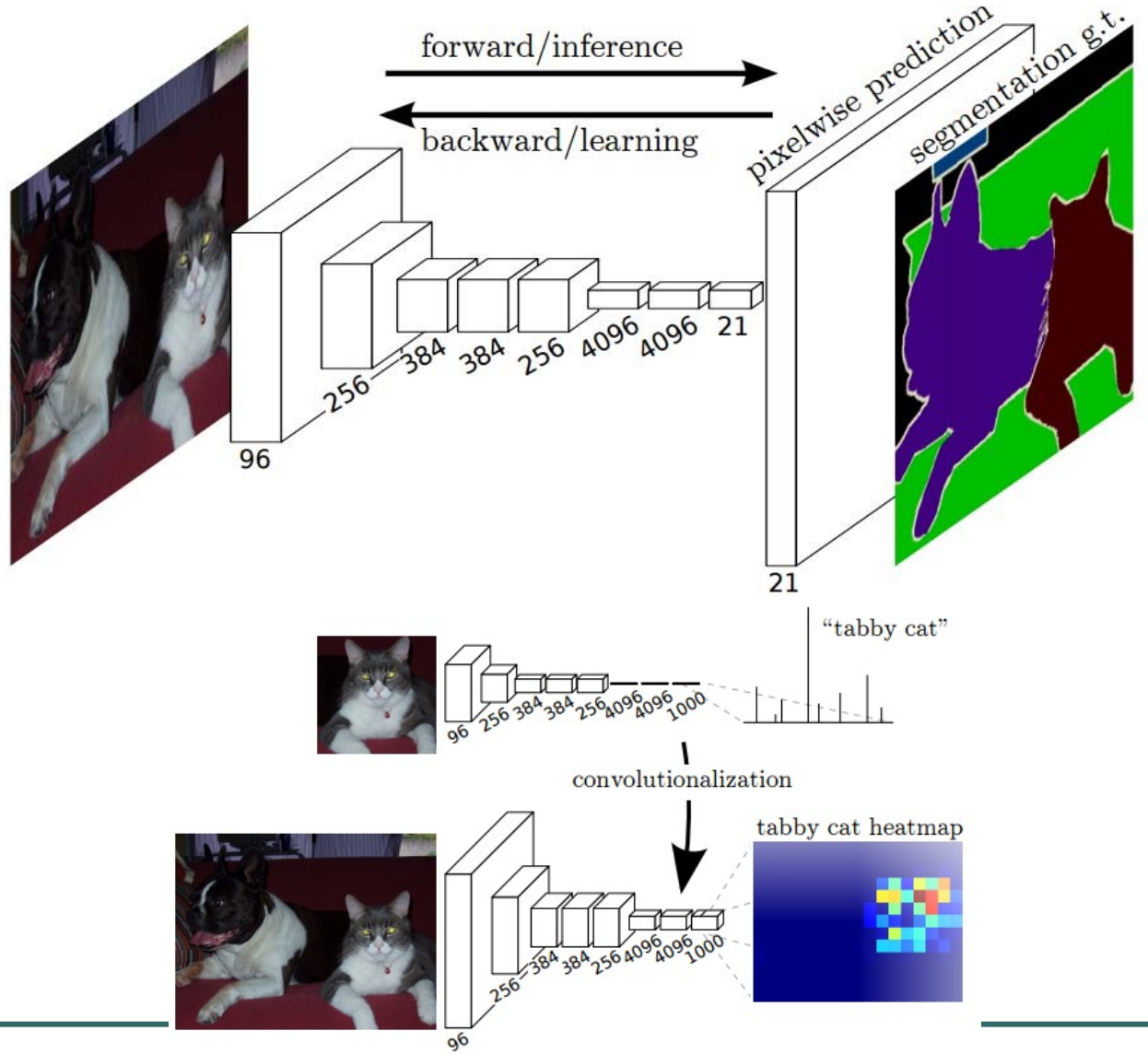
RCNN [Girshick et al. CVPR 2014]

Fast R-CNN



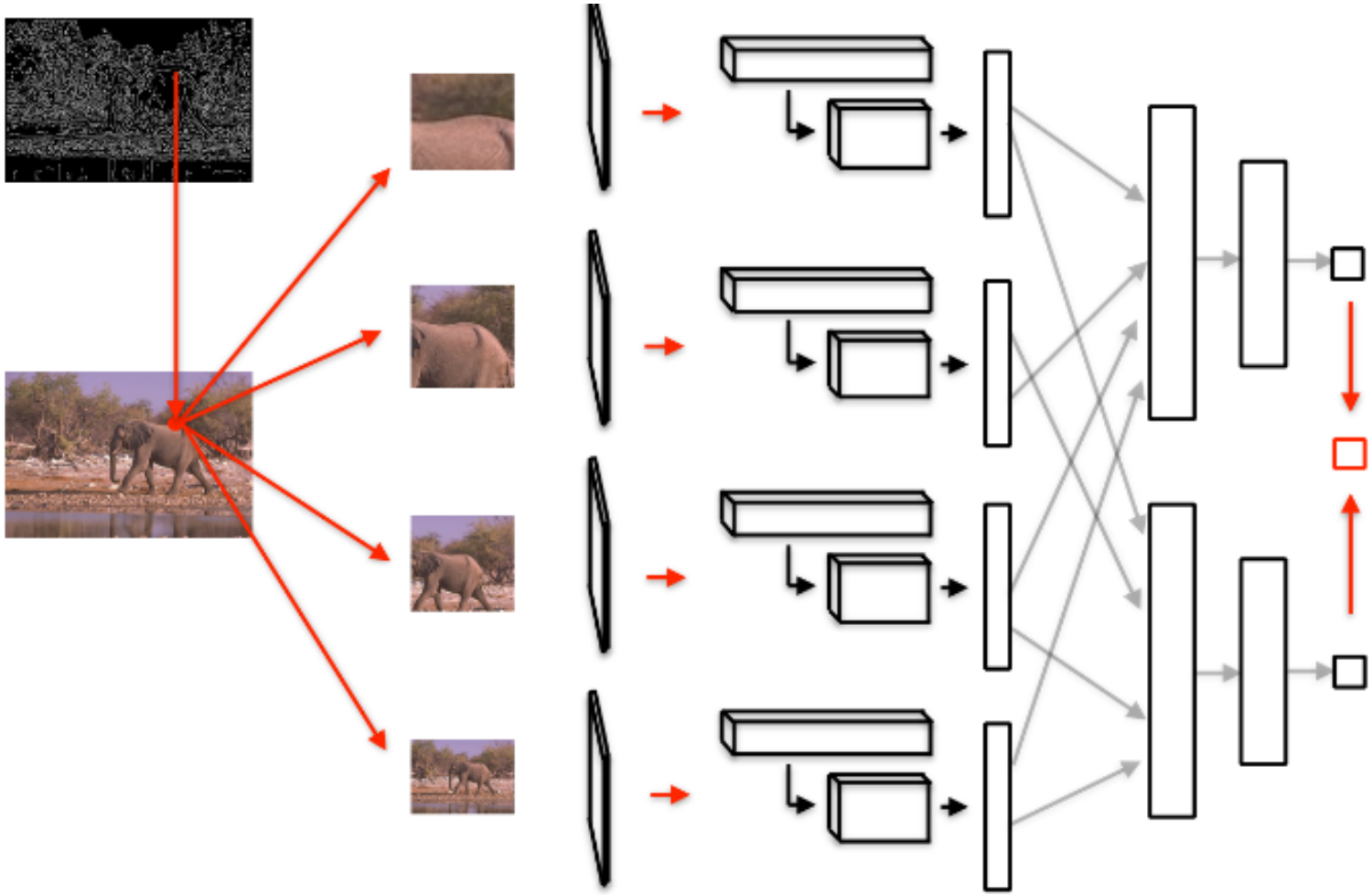
Fast RCNN [Girshick, R. 2015]
https://github.com/rbgirshick/fast-rcnn

Labeling Pixels: Semantic Labels



Fully Convolutional Networks for Semantic Segmentation [Long et al. CVPR 2015]

Labeling Pixels: Edge Detection

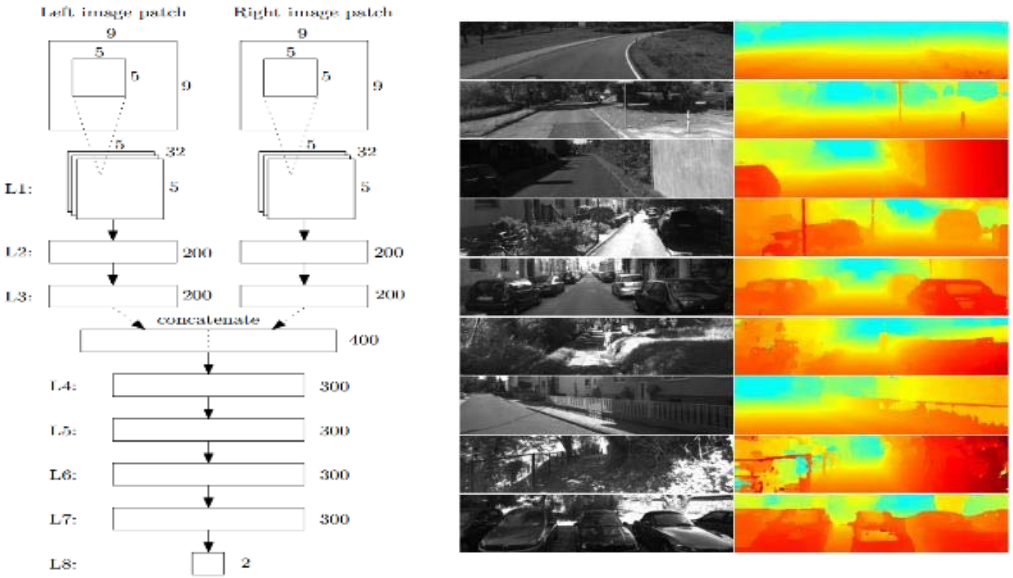


DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection
[Bertasius et al. CVPR 2015]

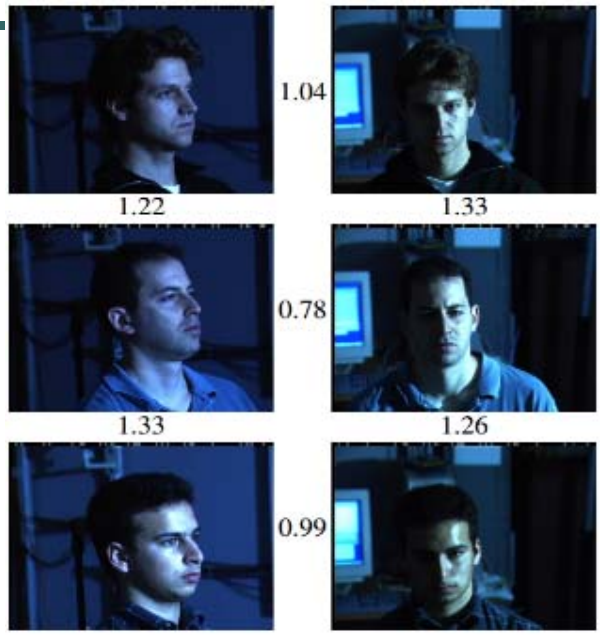
CNN for Regression

DeepPose [Toshev and Szegedy CVPR 2014]

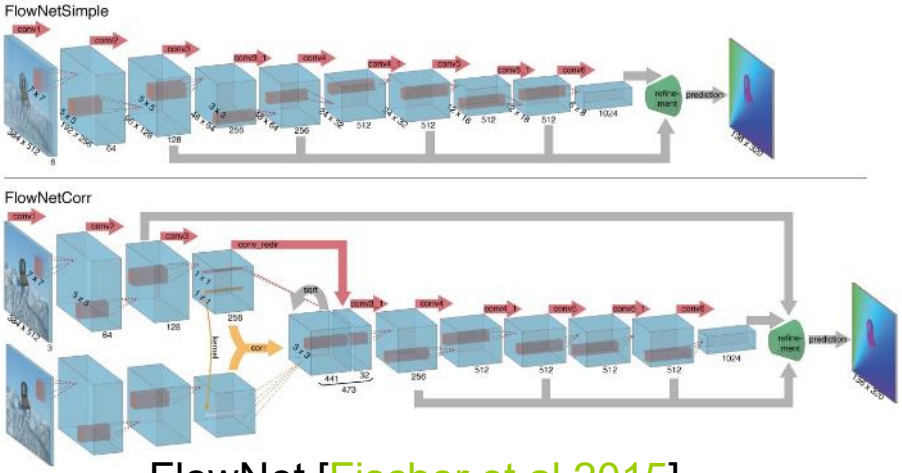
CNN as a Similarity Measure for Matching



Stereo matching [Zbontar and LeCun CVPR 2015]
Compare patch [Zagoruyko and Komodakis 2015]



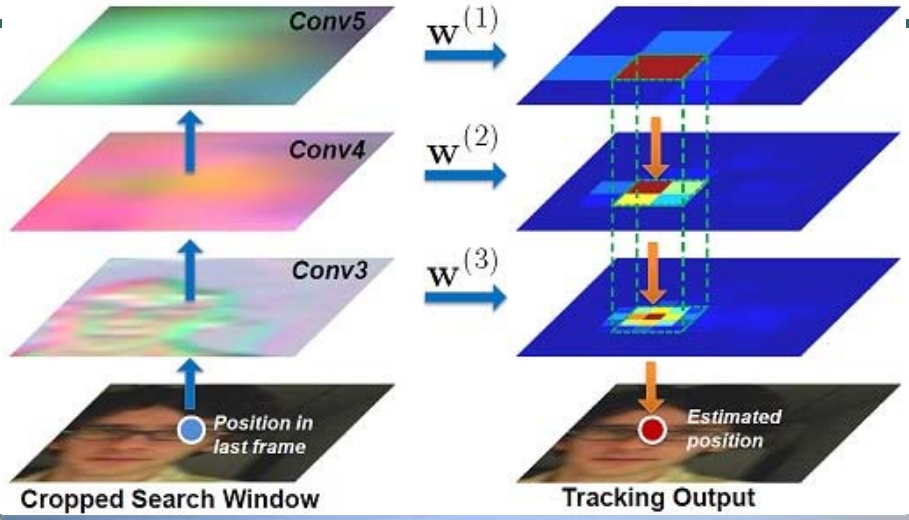
FaceNet [Schroff et al. 2015]



FlowNet [Fischer et al 2015]

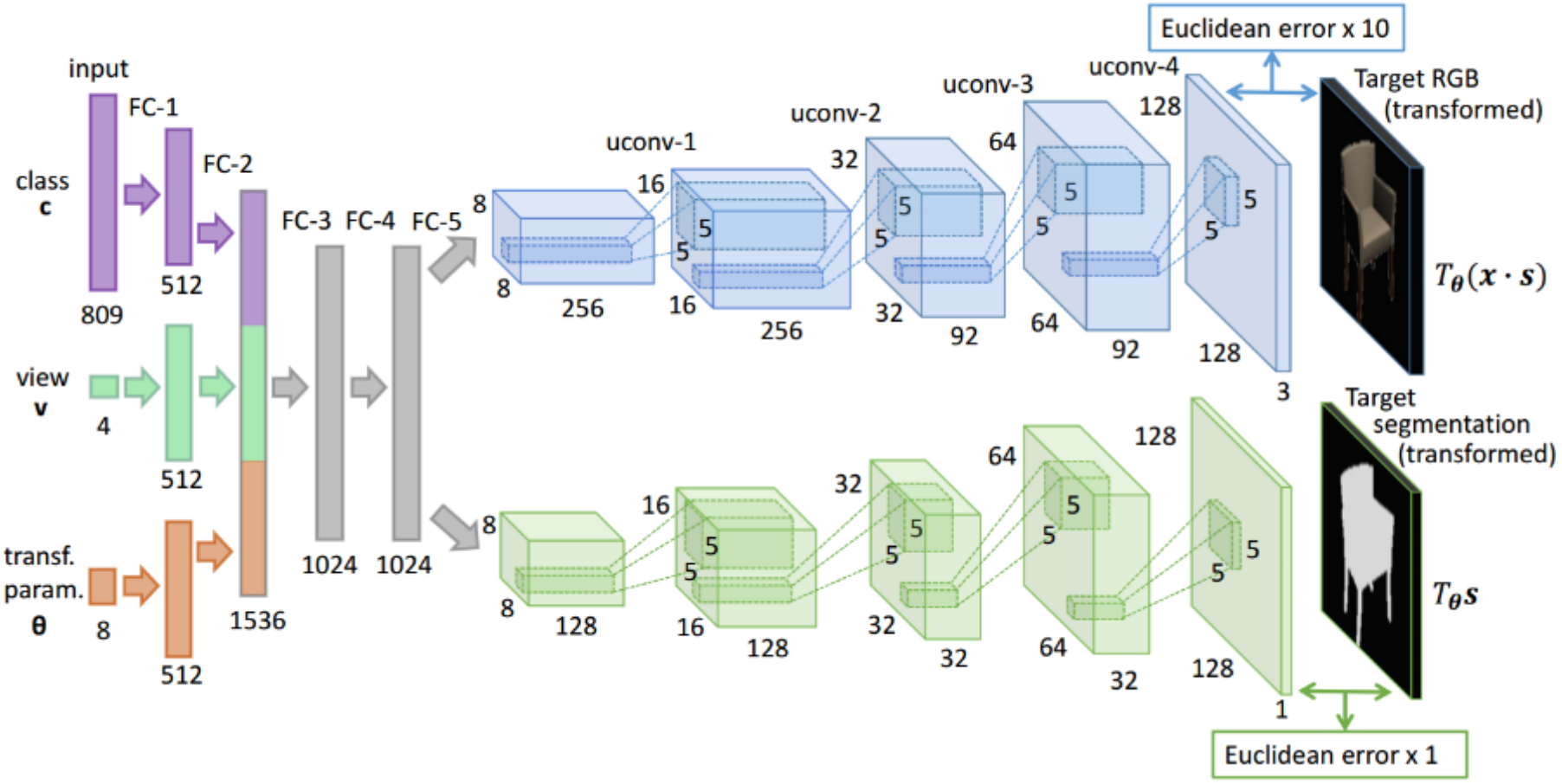
Match ground and aerial images
[Lin et al. CVPR 2015]

CNN for Online Visual Tracking



Hierarchical Convolutional Features for Visual Tracking [Ma et al. ICCV 2015]

CNN for Image Generation

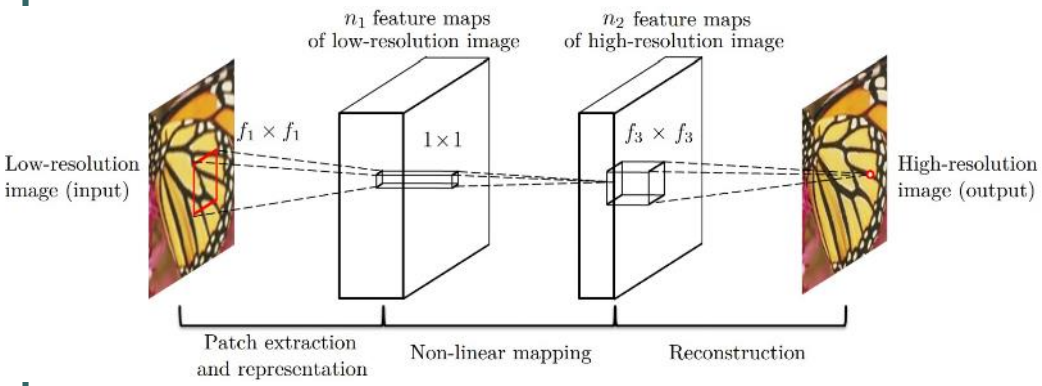


Learning to Generate Chairs with Convolutional Neural Networks [Dosovitskiy et al. CVPR 2015]

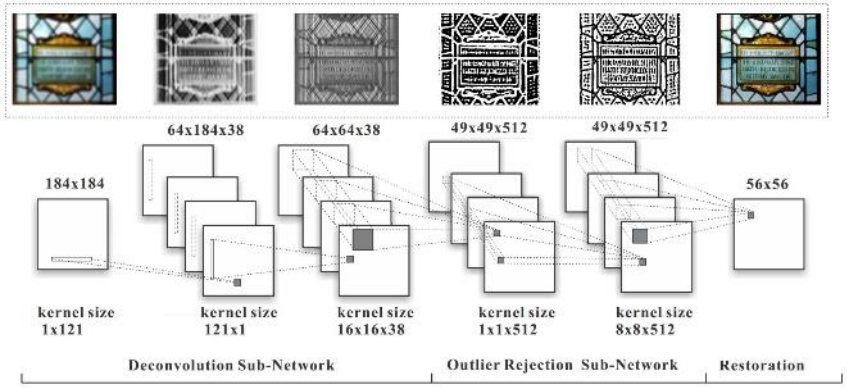
Chair Morphing

Learning to Generate Chairs with Convolutional Neural Networks [Dosovitskiy et al. CVPR 2015]

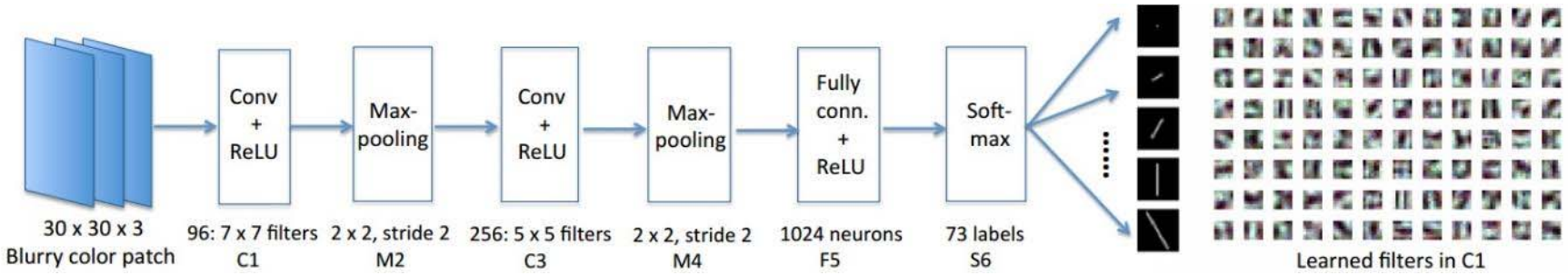
CNN for Image Restoration/Enhancement



Super-resolution
[Dong et al. ECCV 2014]



Non-blind deconvolution
[Xu et al. NIPS 2014]

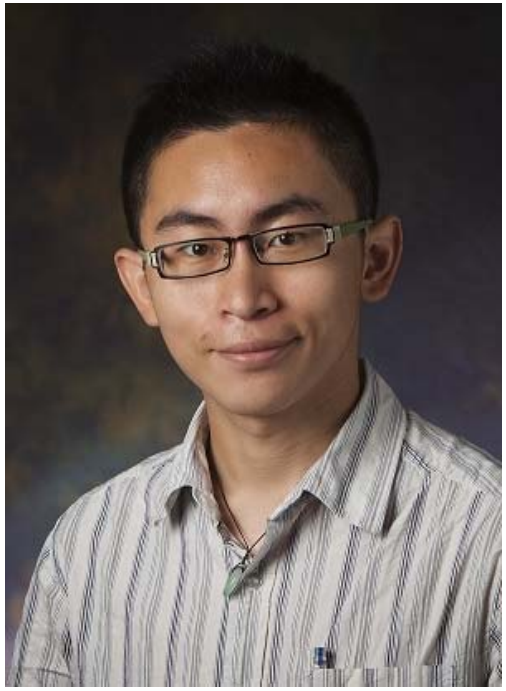


Non-uniform blur estimation
[Sun et al. CVPR 2015]

Style Transfer

- Find an output image with
  - similar activations of early layers (low-level) of source image
  - similar activations of later layers (high-level) of target image

Source image



Target image

Output (deepart)

A Neural Algorithm of Artistic Style [Gatys et al. 2015]