Convolutional Neural Networks

(CNNs, or ConvNet, DCN)

Sources:

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

Ch. 9, "Deep Learning" textbook
by Goodfellow et al.

Image Categorization

Training

Training Images

Training Labels

Image Features

Classifier Training

Trained Classifier

Testing

Test Image

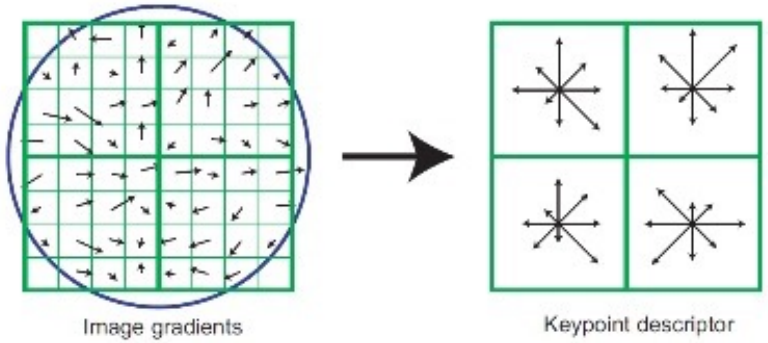
Image Features

Trained Classifier

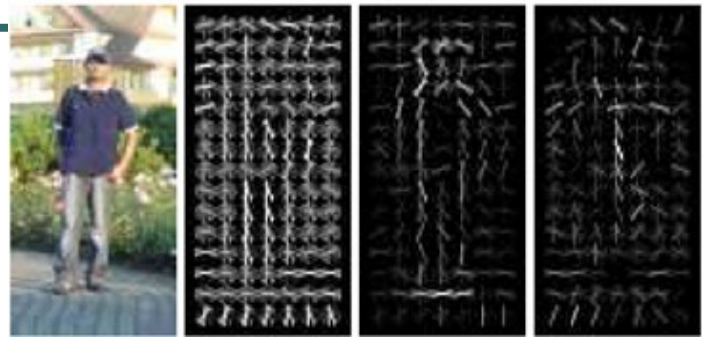
Prediction

Outdoor

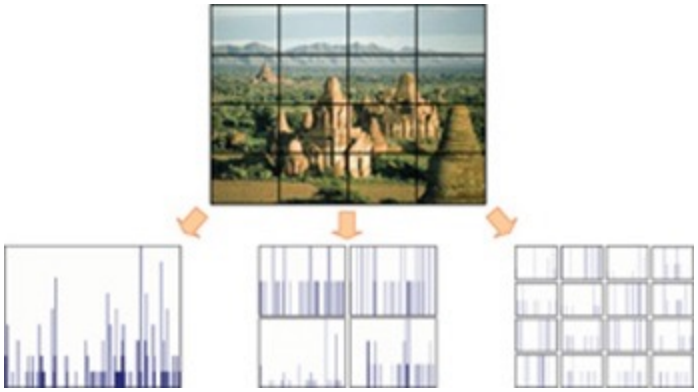
Features are the Keys



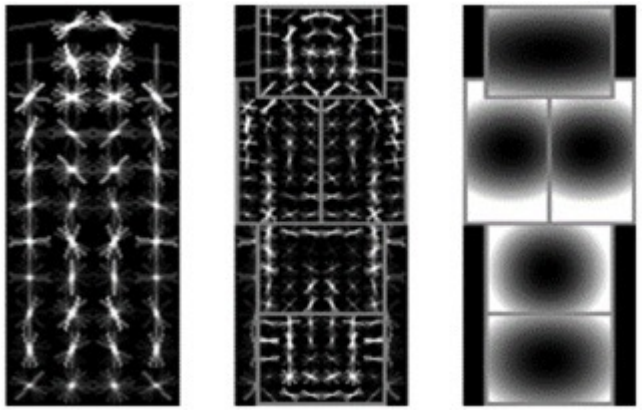
SIFT [Lowe IJCV 04]



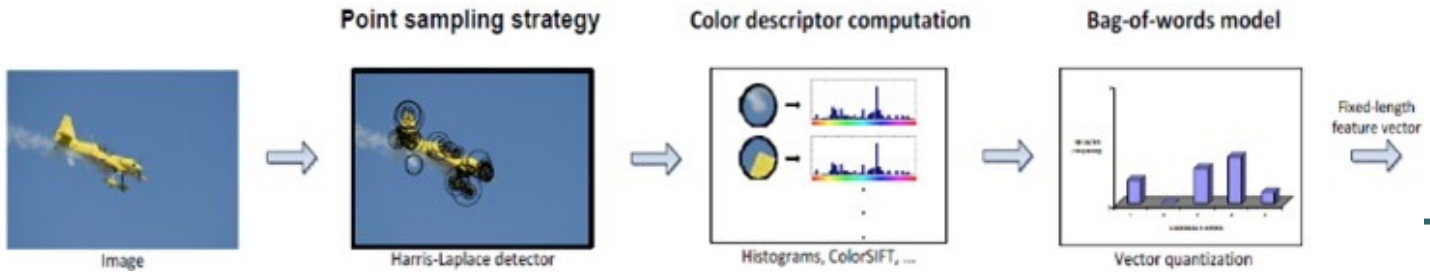
HOG [Dalal and Triggs CVPR 05]



SPM [Lazebnik et al. CVPR 06]



DPM [Felzenszwalb et al. PAMI 10]



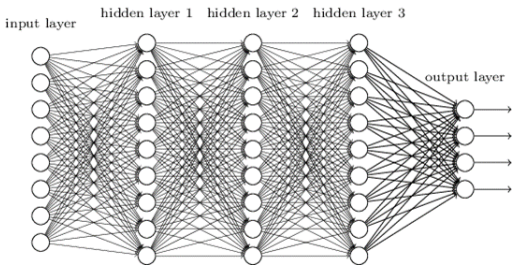
Color Descriptor [Van De Sande et al. PAMI 10]

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**

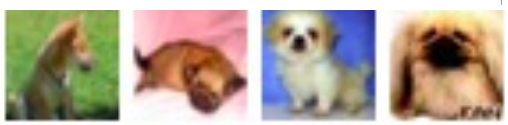
Convolutional Neural Networks

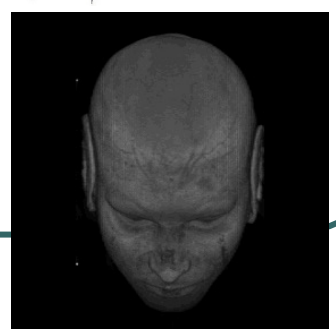
- Convolutional neural networks (CNNs) use convolution in place of matrix multiplication in at least one of the layers of neural networks.
  - Everything else stays the same
    - Maximum likelihood
    - Negative log-likelihood
    - Back-propagation



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

  - 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

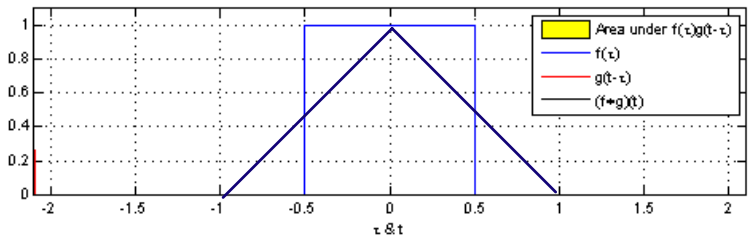




Convolution Operation

- Convolution of two 1-D signals: compare similarity of f and g

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



- 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 (actually cross-correlation) of a 2-D image with a 2-D kernel:

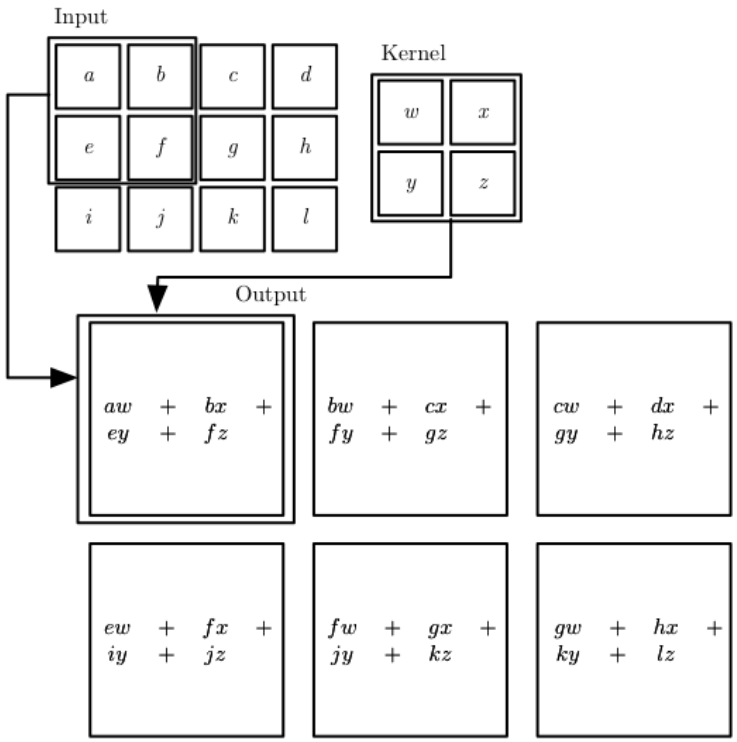
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

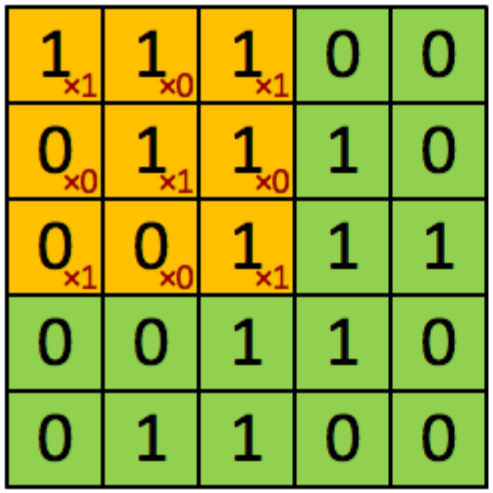
Convolution Operation

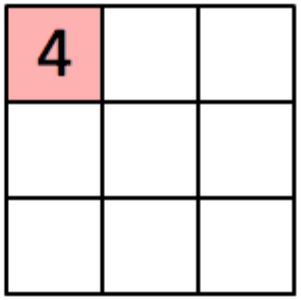
- Convolution of a 2-D image with a 2-D kernel:

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

feature map = input * kernel





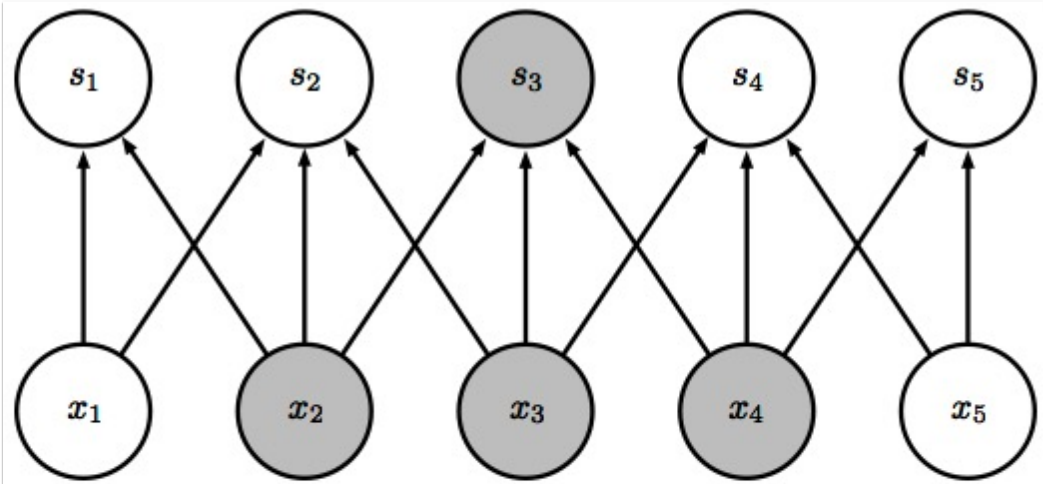


Motivation of CNN

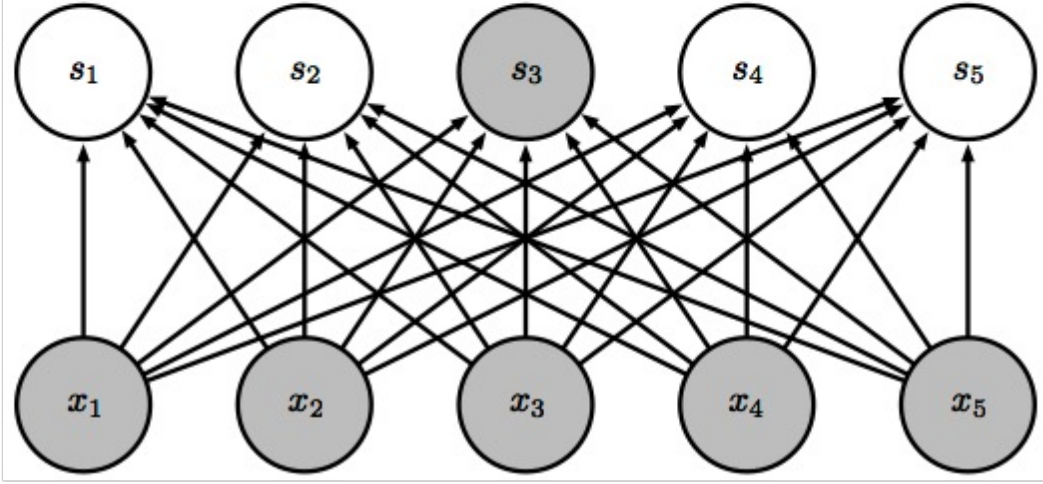
- Scale up neural networks to process very large signals / images / video sequences
- Three essential ideas:
  - Sparse and local connectivity
    - Kernel is usually much smaller than input
    - Less memory and computation required
  - Parameter (weight) sharing
    - Use the same set of parameters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

CNN: Local Connectivity

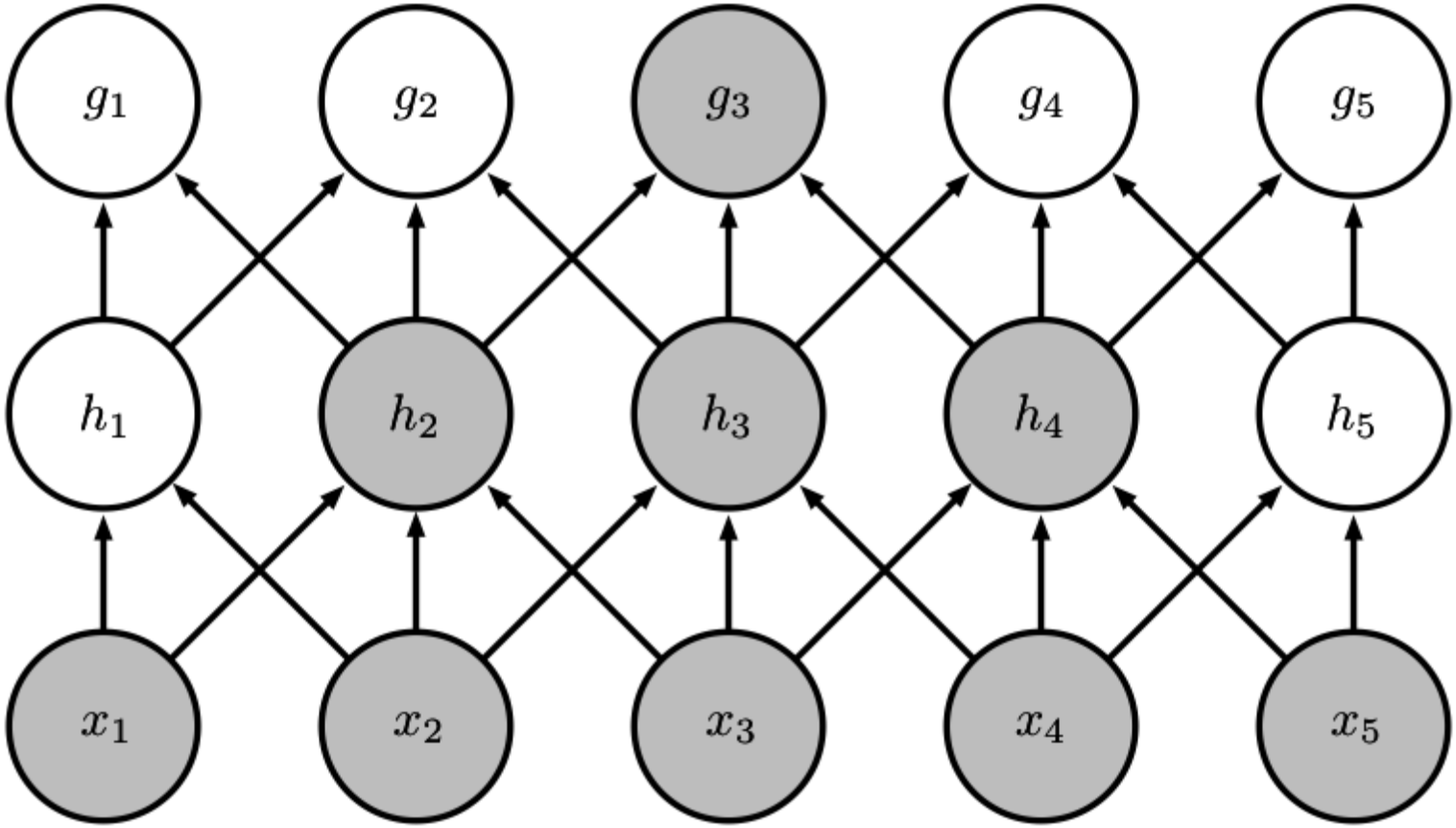
Sparse connections due to small convolution kernel



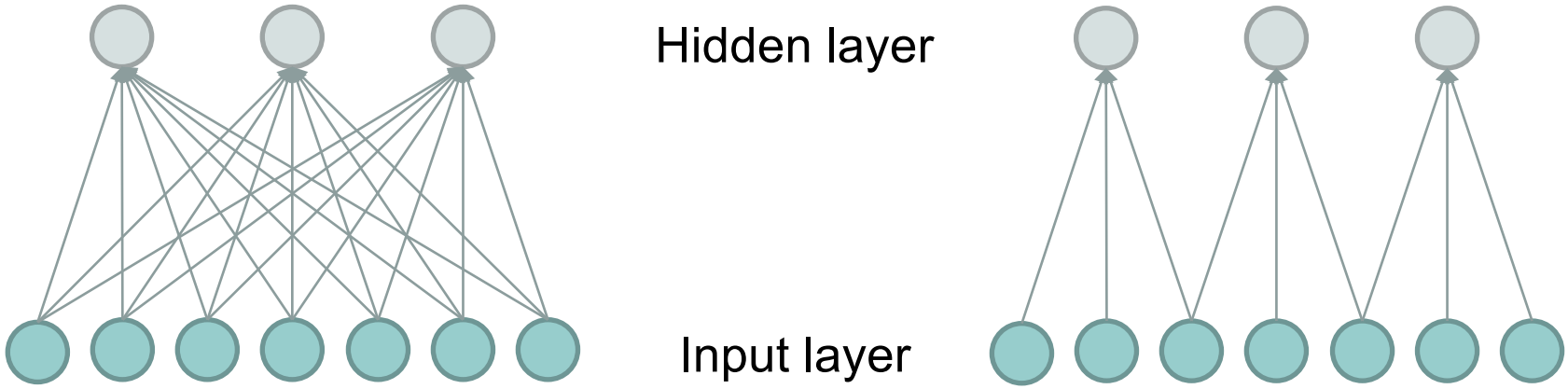
Dense connections



CNN: Growing Receptive Fields



CNN: Local Connectivity

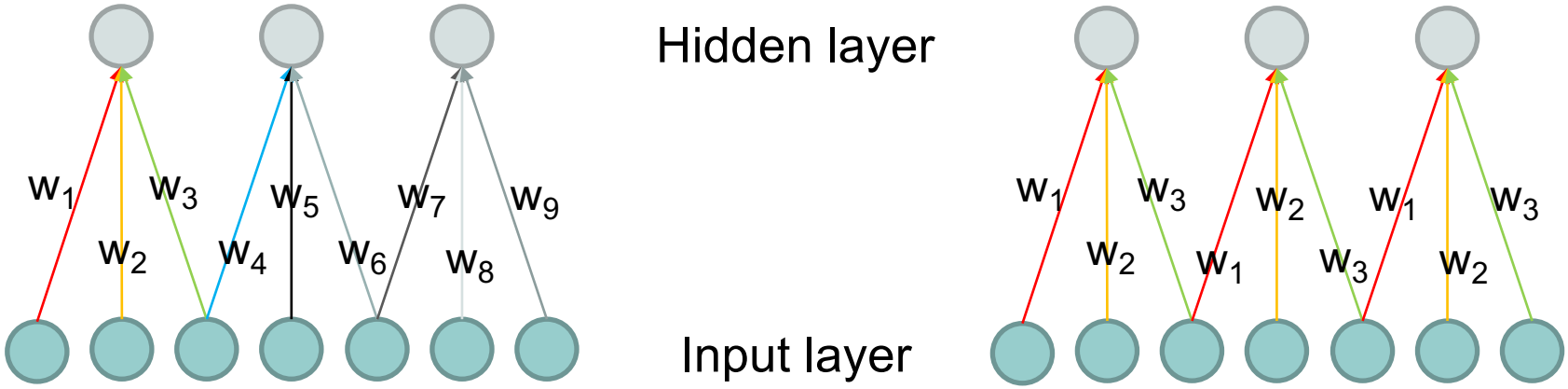


Global connectivity

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



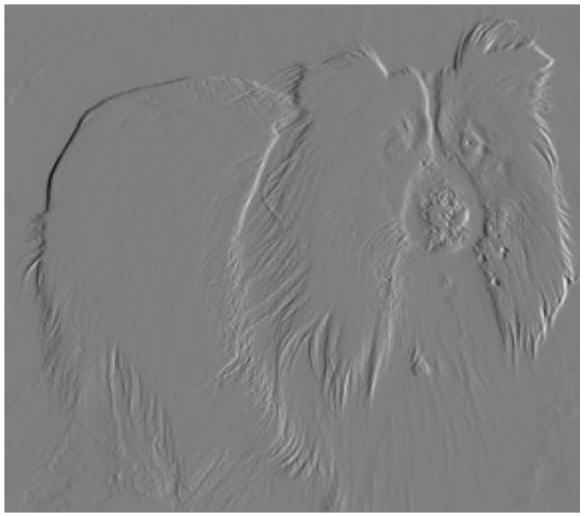
Without weight sharing

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



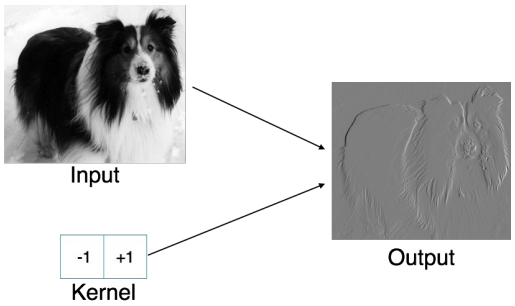
Output

-1	+1

Kernel

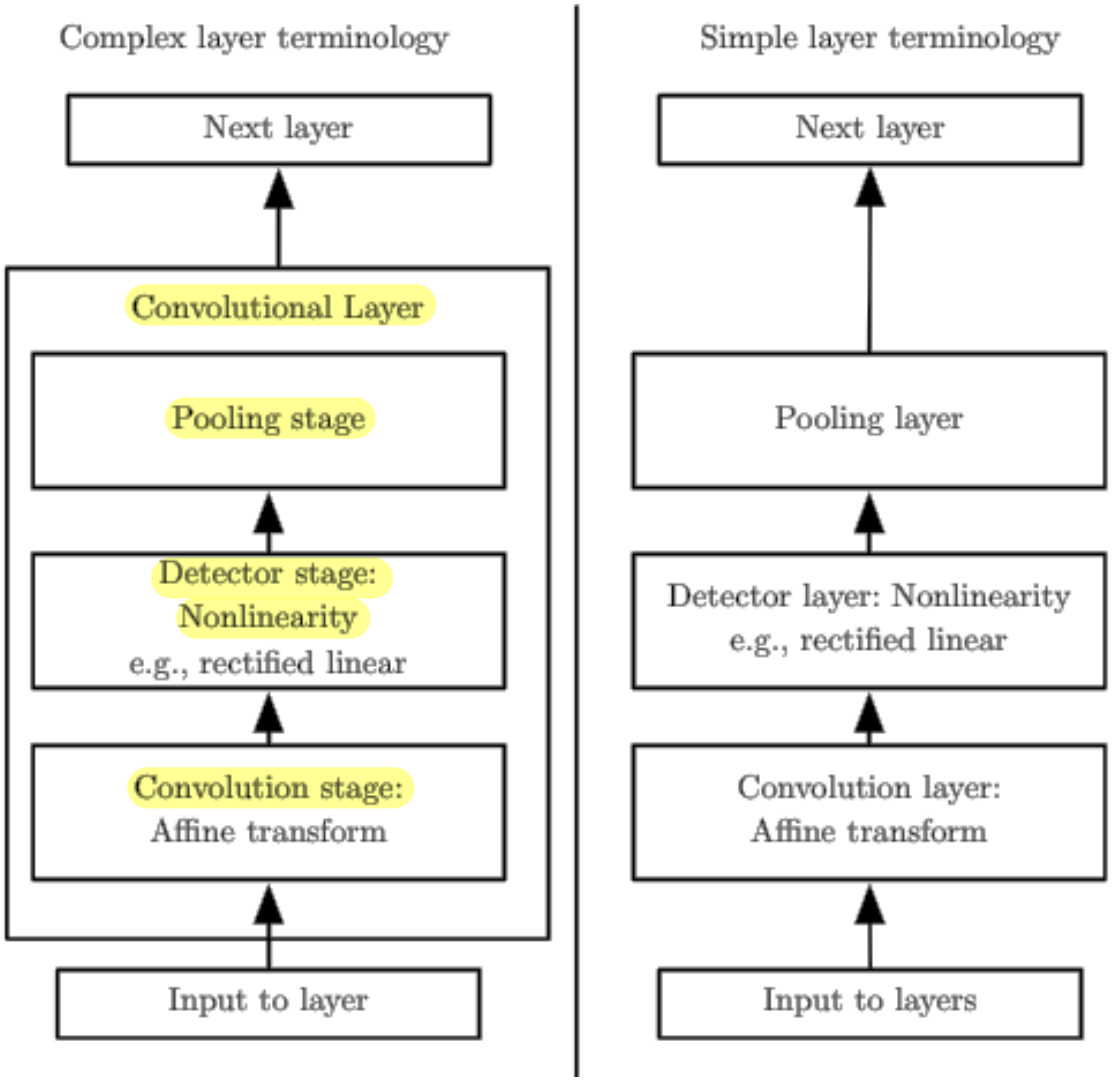
Efficiency of Convolution

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



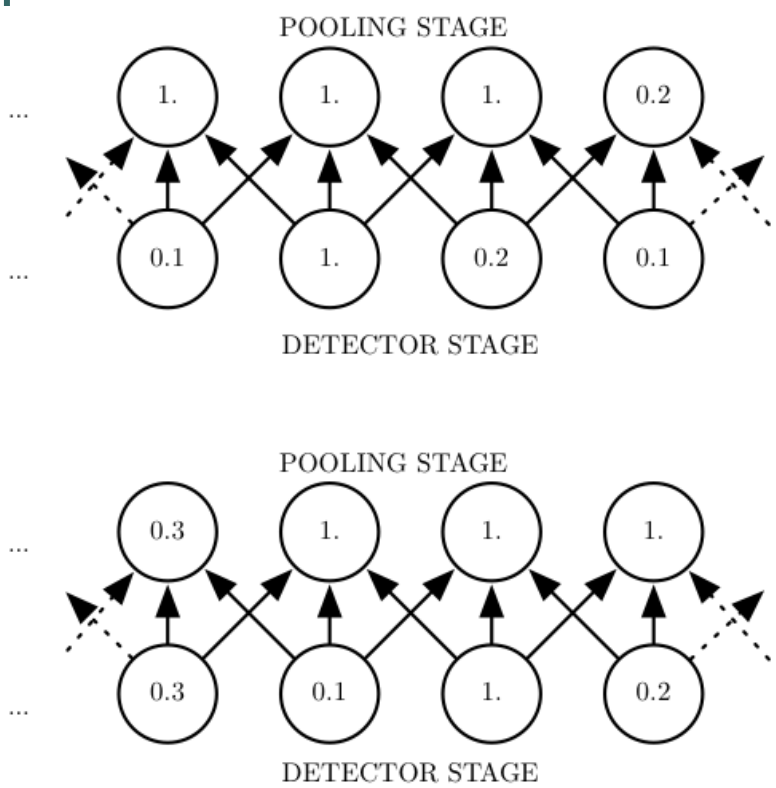
	Convolution	Dense matrix	Sparse matrix
Stored floats	2	319*280*320*280 > 8e9	2*319*280 = 178,640
Float muls or adds	319*280*3 = 267,960	> 16e9	Same as convolution (267,960)

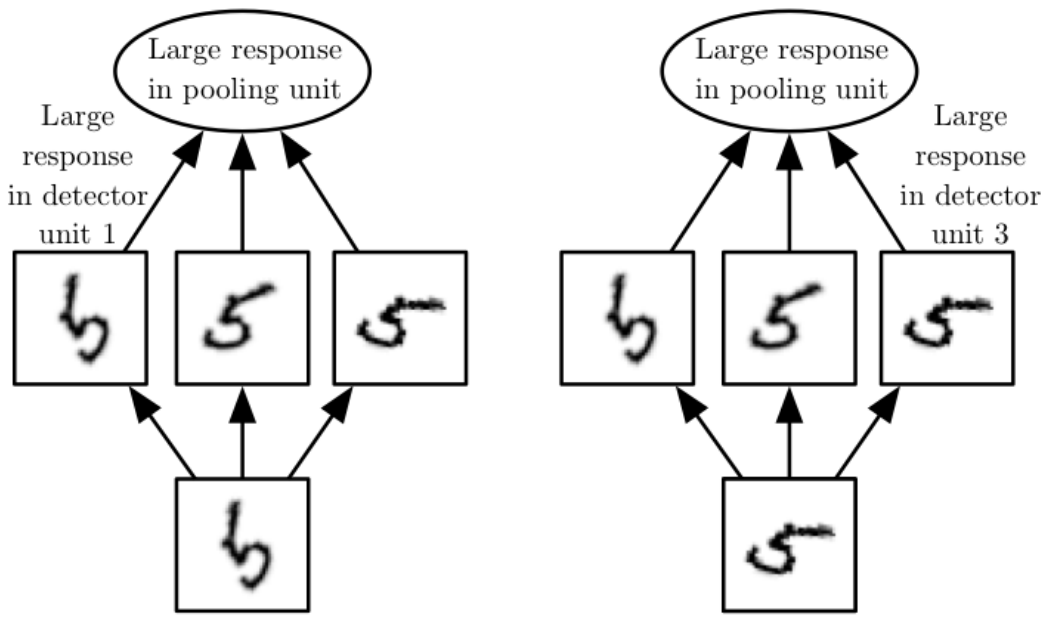
Components of CNN



Pooling

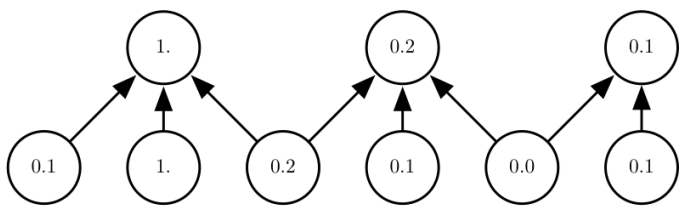
- A pooling function replaces the output of the net with a summary statistic of the nearby outputs.
- Pooling helps to make the representation become approximately invariant to small translation and rotation.





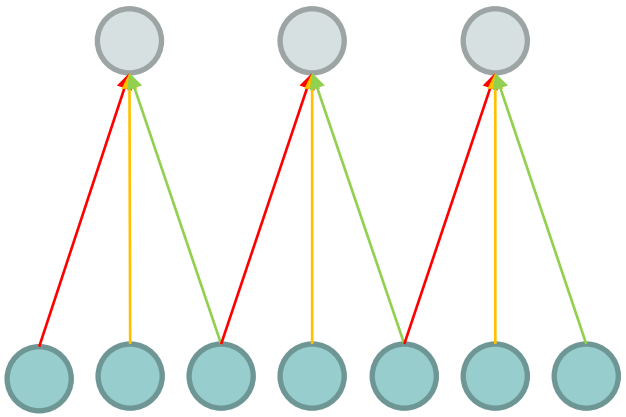
Pooling

- Examples of pooling 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
  - LSE (Log-Sum-Exp) Pooling $y = \log \sum_{i=1}^{n} e^{x_i}$
- Can use fewer pooling units than detector units (pooling with downsampling)
  - E.g., pooling width 3, stride 2



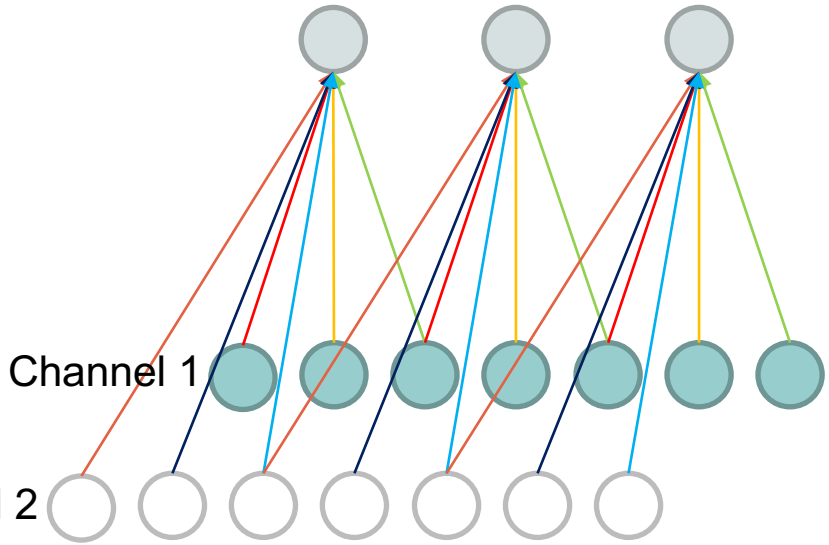
- Can help to handle inputs of varying size

CNN with Multiple Input Channels

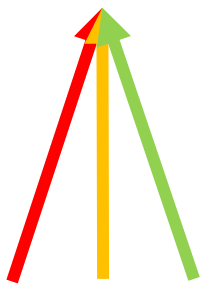


Hidden layer

Input layer

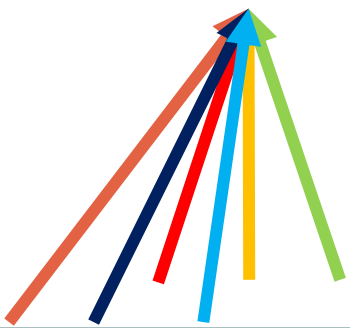


Single input channel



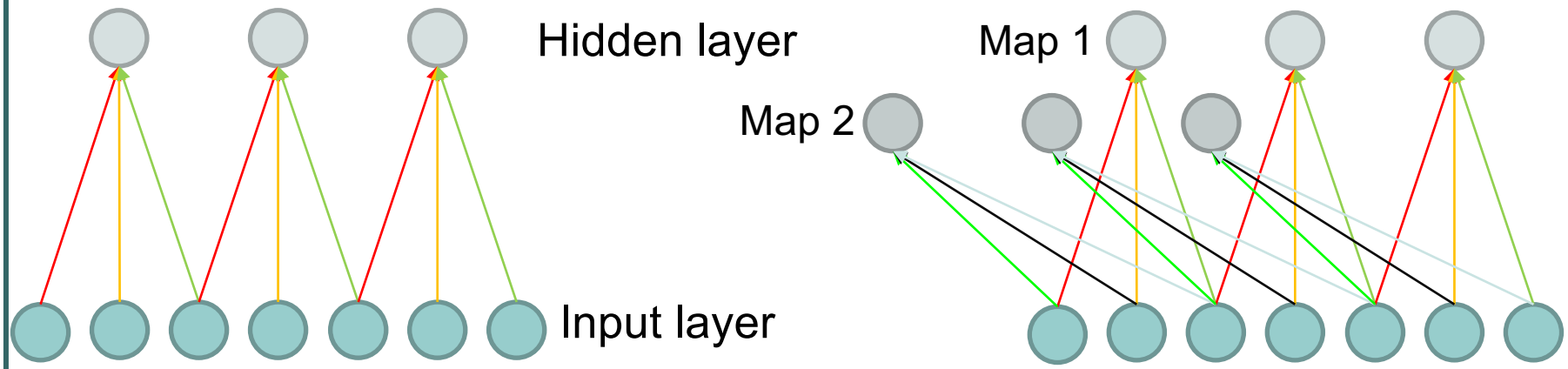
Filter weights

Multiple input channels



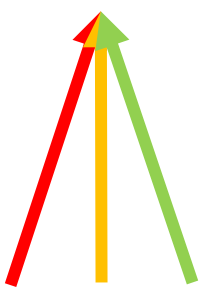
Filter weights

CNN with Multiple Output Maps

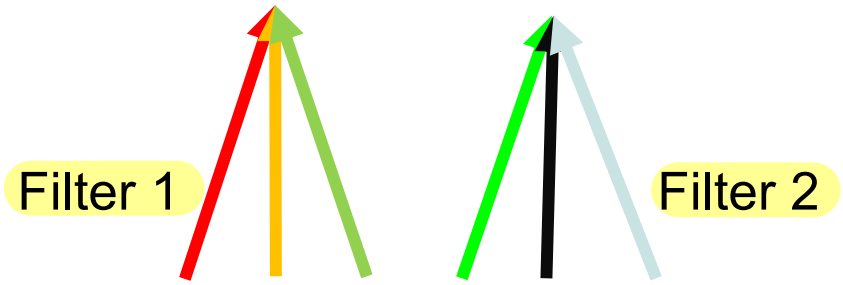


Single output map

Multiple output maps



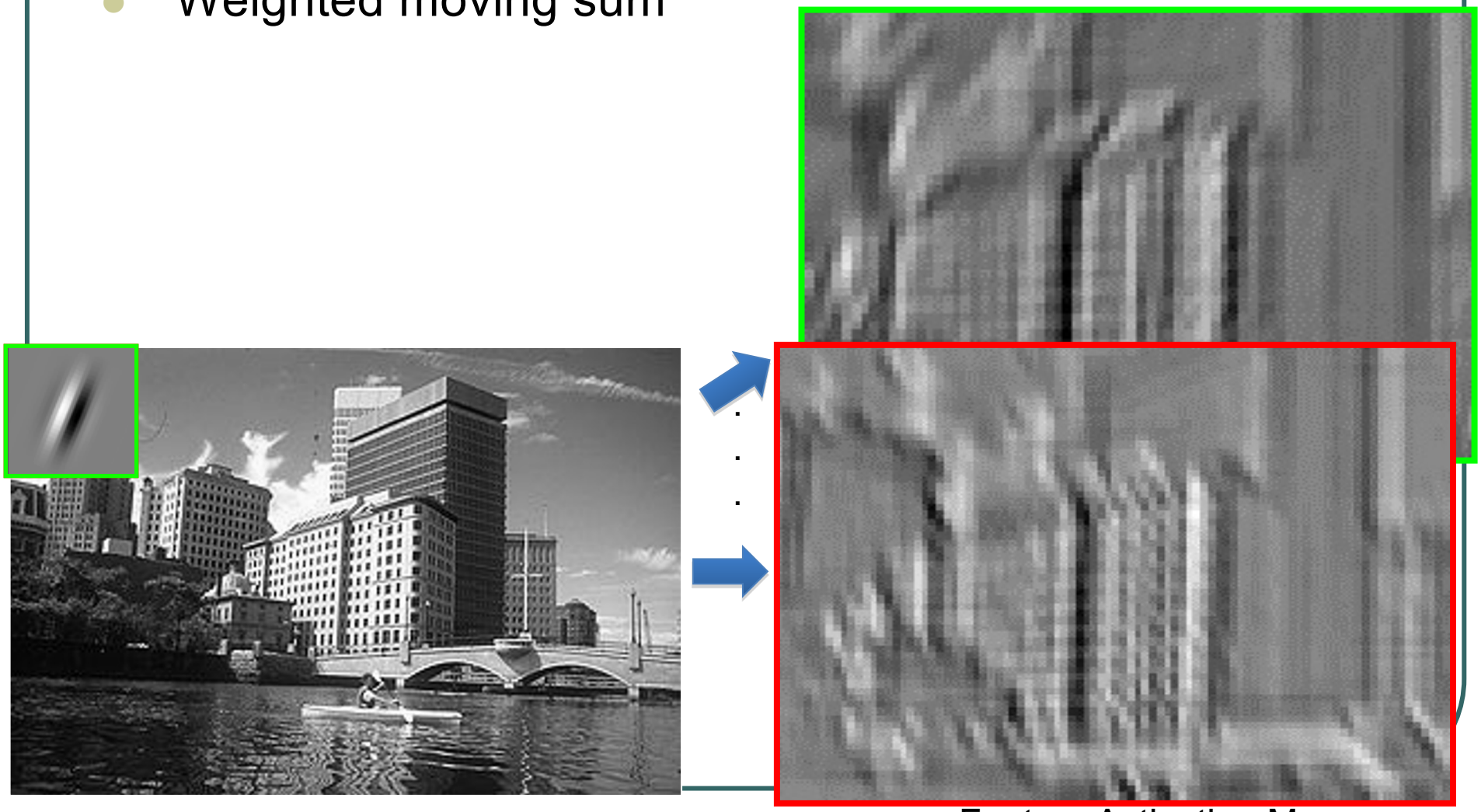
Filter weights



Filter weights

Multiple Feature Maps

- Weighted moving sum



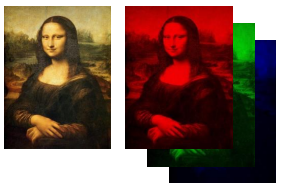
Input

Feature Activation Map

slide credit: S. Lazebnik

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)

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

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

Multi-channel Convolution

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

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

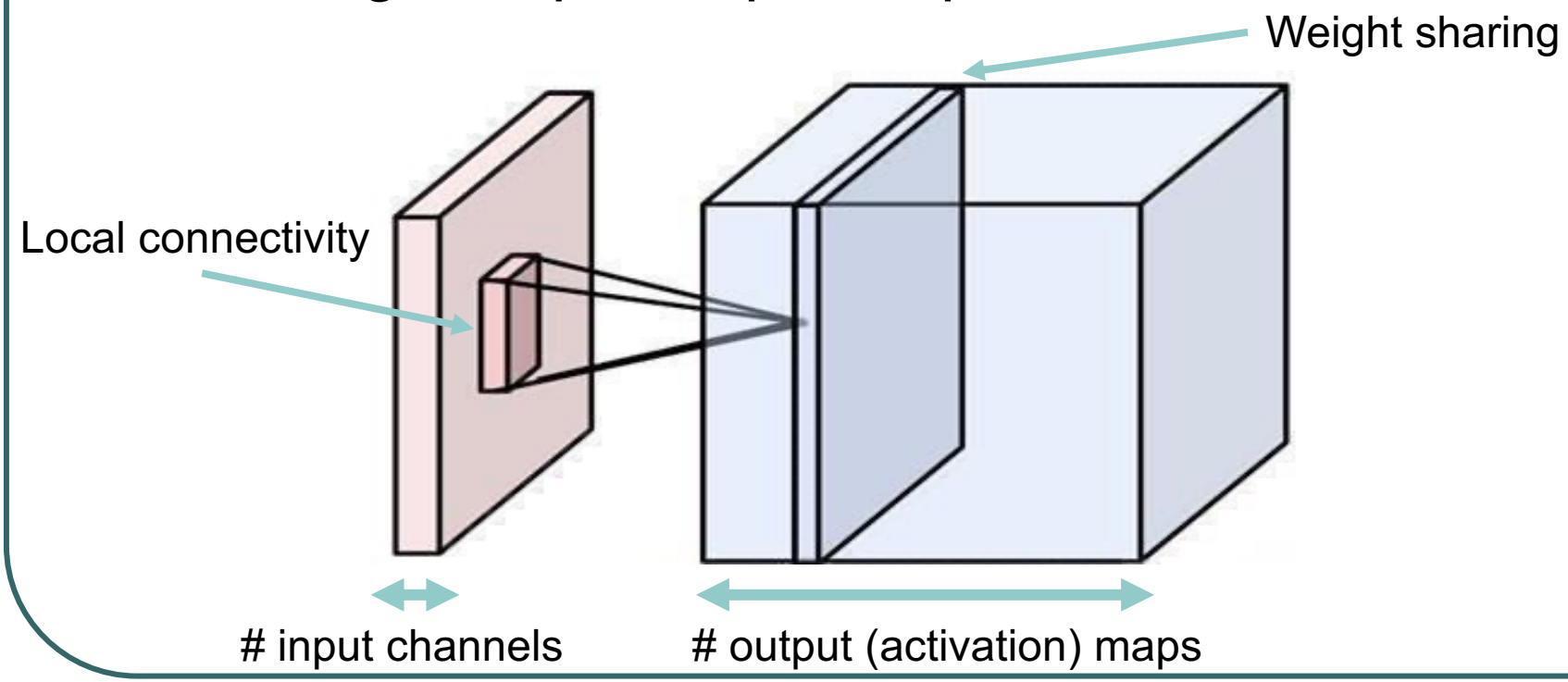
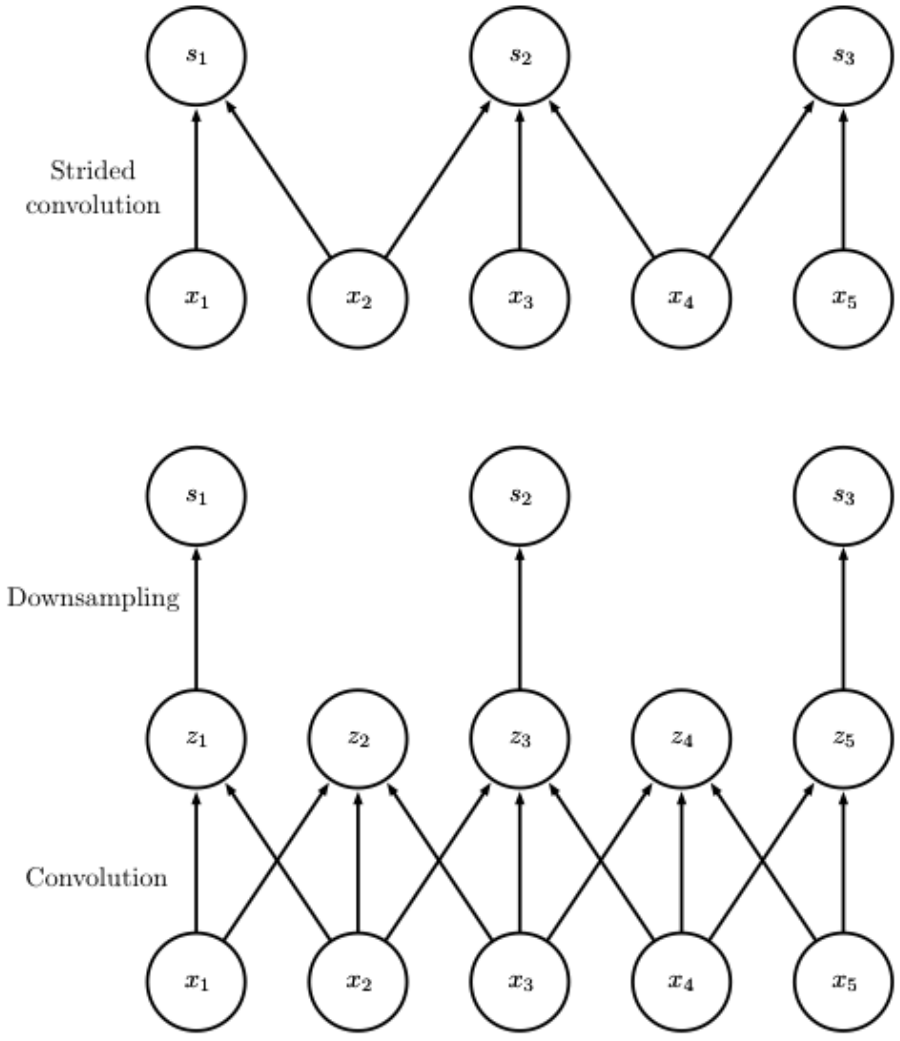


Image credit: A. Karpathy

Convolution with Stride



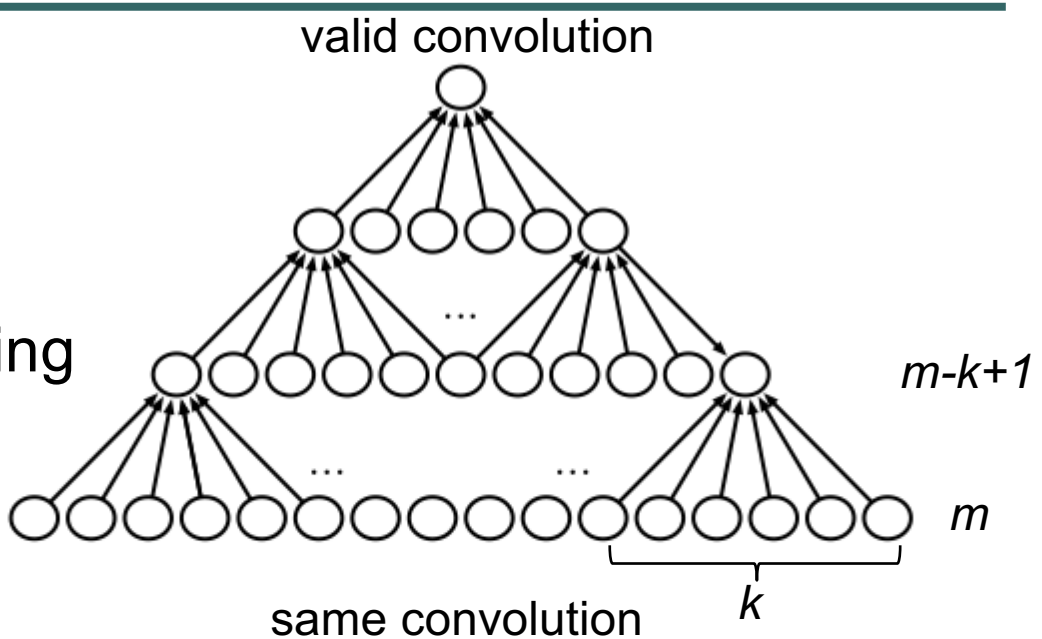
$$Z_{i,j,k} = c(\mathbf{K}, \mathbf{V}, s)_{i,j,k}$$

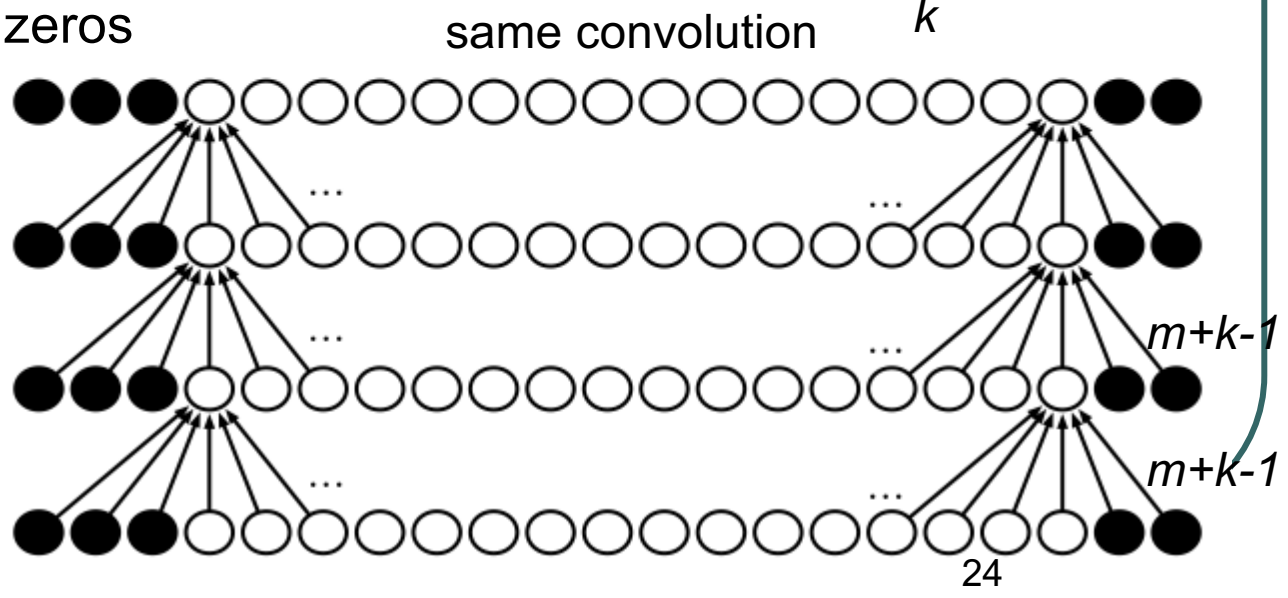
$$= \sum_{l,m,n} V_{l,(j-1)s+m,(k-1)s+n} K_{i,l,m,n}$$

s: stride

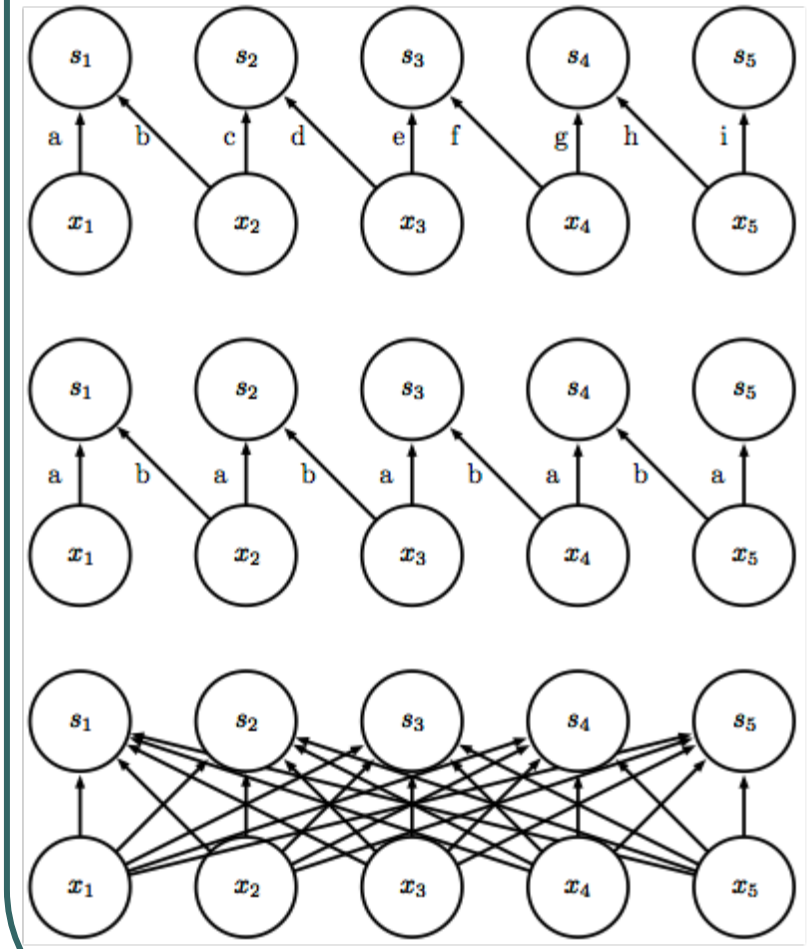
Zero-padding Controls Size

- No zero-padding
  - Valid convolution
  - Shrinking size
- Just enough zero-padding
  - Same convolution
  - Keep the size
  - Padding with k-1 zeros
- Padding with enough zeros
  - Full convolution
  - Every pixel is visited k times.





Local Connections



Local connection:
like convolution,
but no sharing

unshared convolution

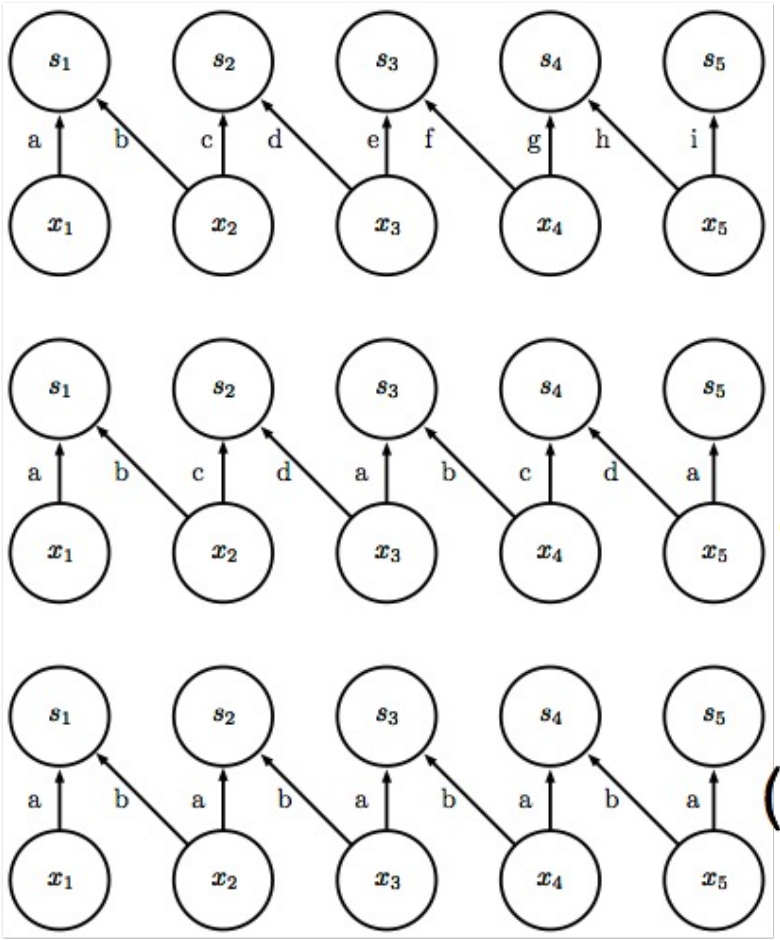
$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,j,k,l,m,n}$$

K: 6D tensor

Convolution

Fully connected

Tiled Convolution



Local connection (no sharing)

Tiled convolution (cycle between groups of shared parameters)

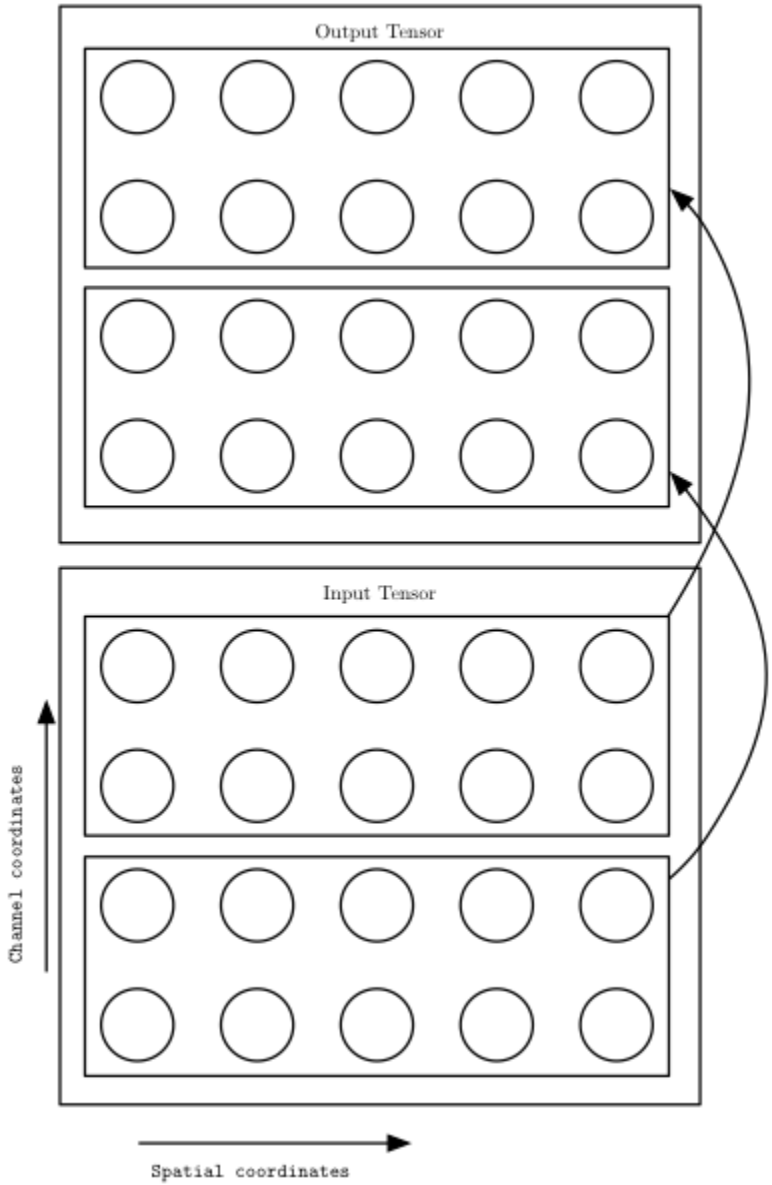
Convolution (one group shared everywhere)

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

t: number of kernels

Partial Connectivity Between Channels

- Each output channel *i* is a function of only a subset of input channels.



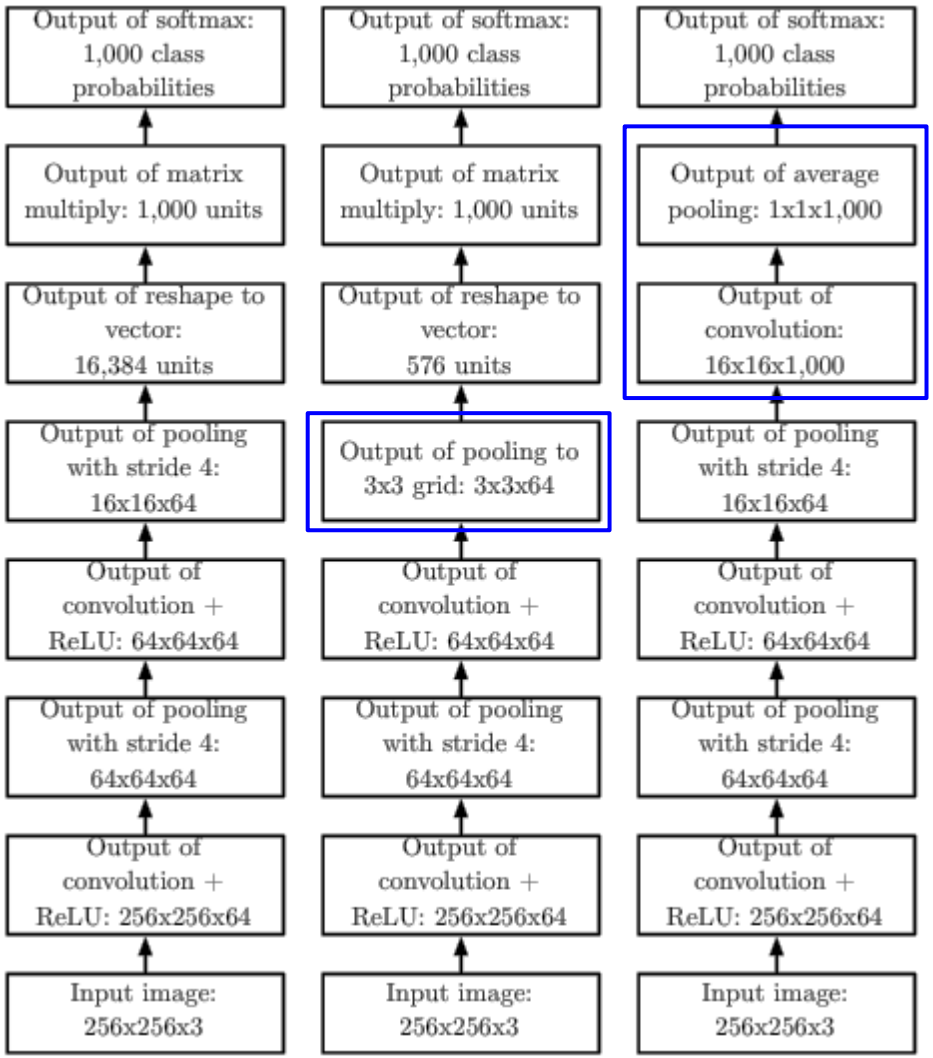
Efficient Convolution Algorithms

- 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 unsupervised criterion
  - K-means clustering

Example Classification Architectures

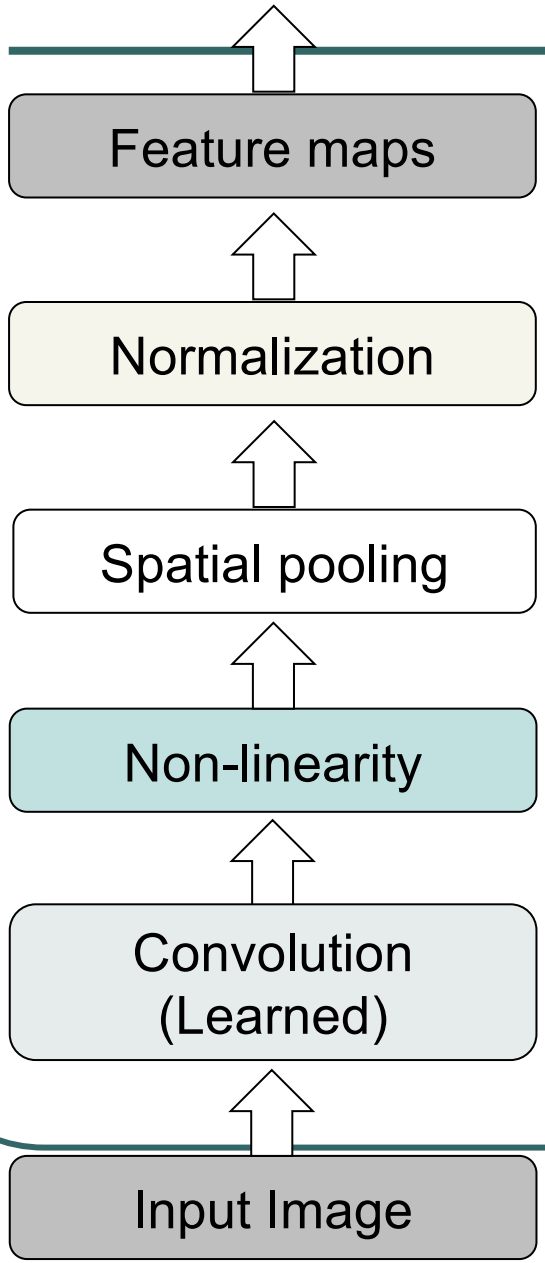


Fixed image size

Fixed number of variably-sized pools

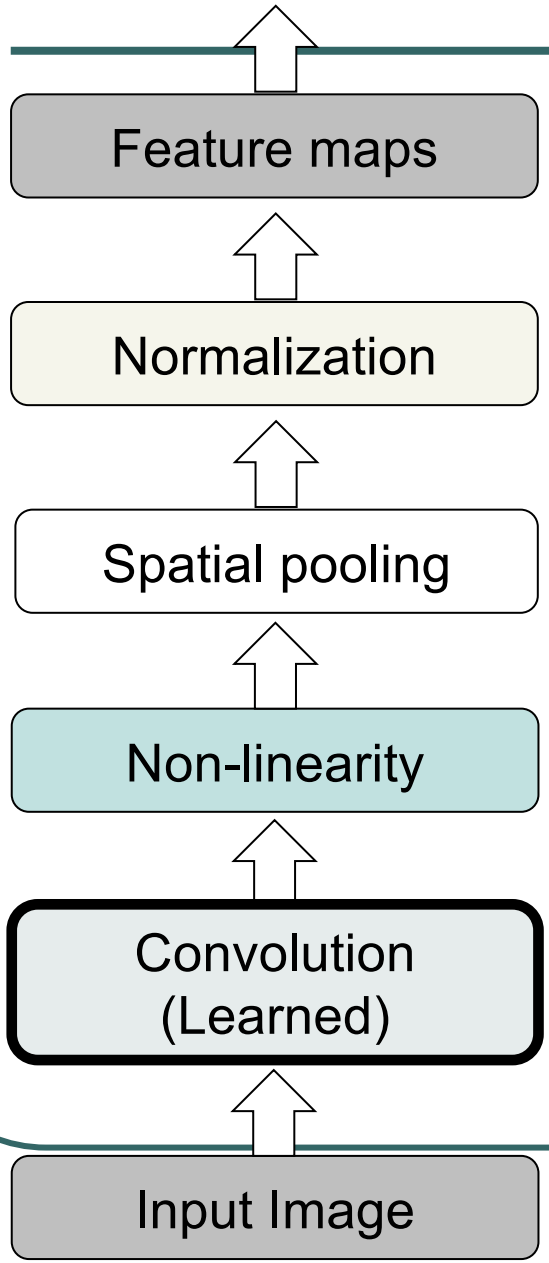
A feature map per class

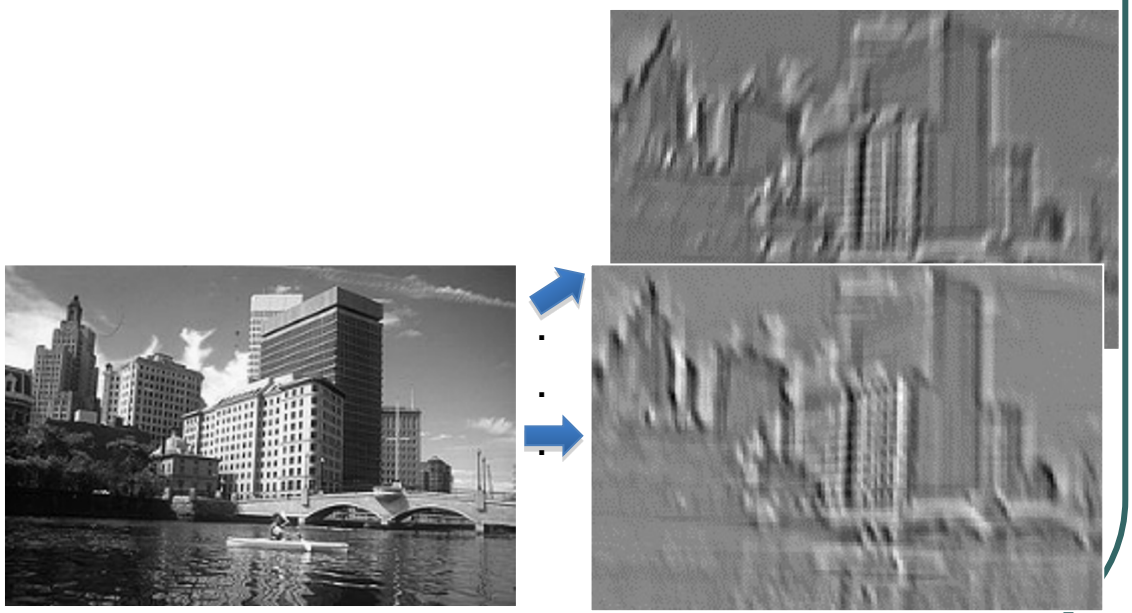
Convolutional Neural Networks



slide credit: S. Lazebnik

Convolutional Neural Networks



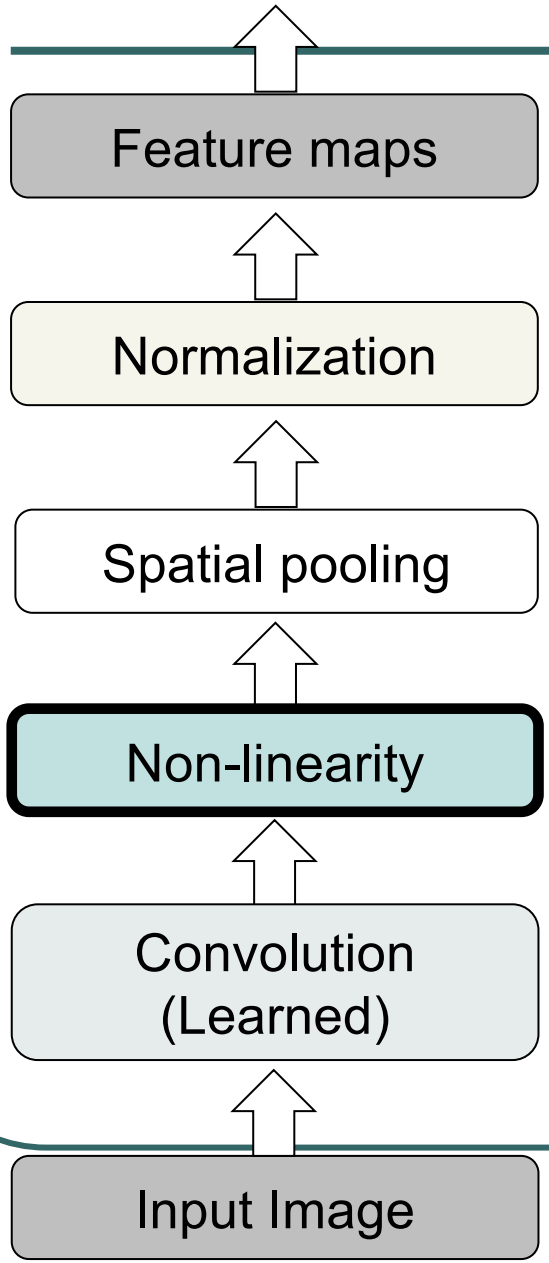


Input

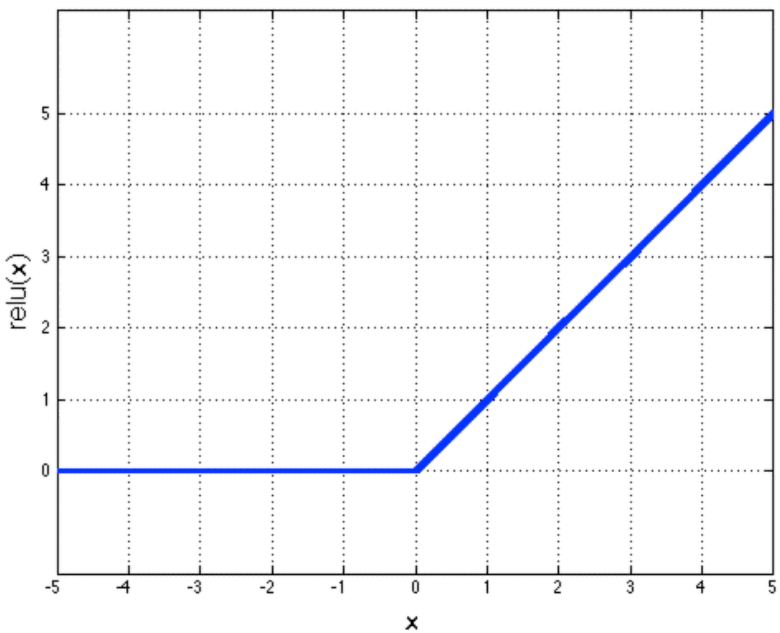
Feature Map

slide credit: S. Lazebnik

Convolutional Neural Networks

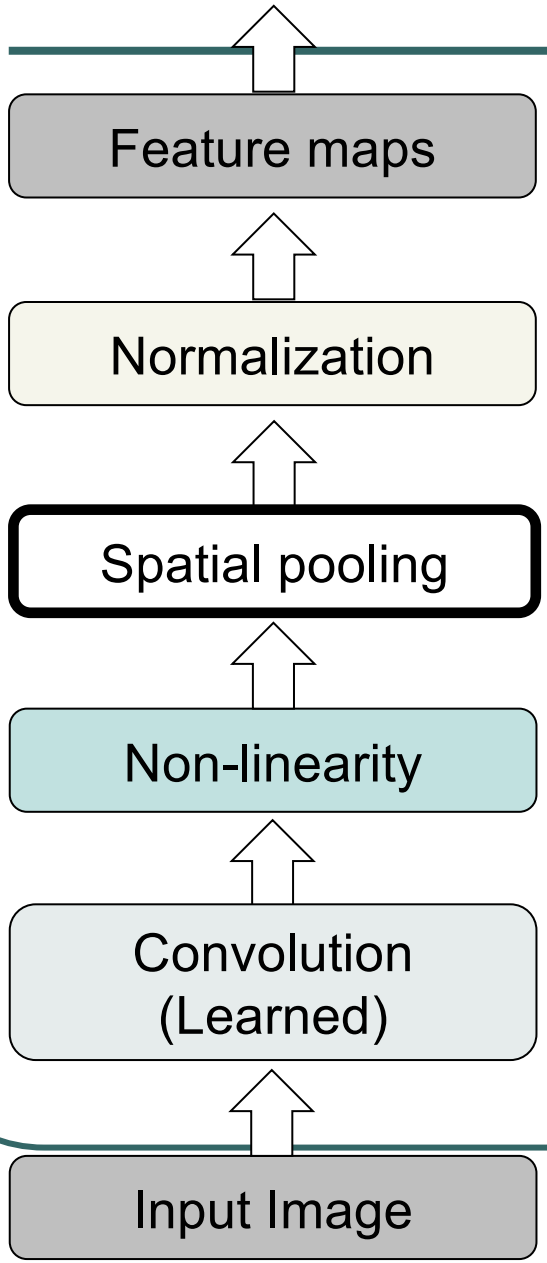


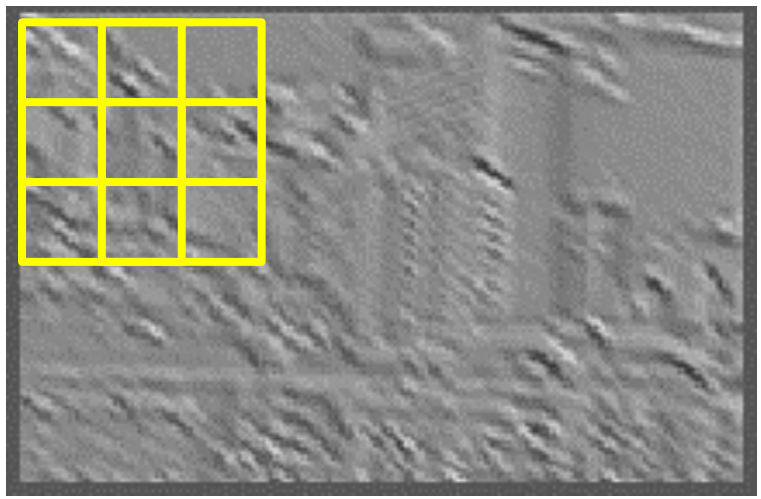
Rectified Linear Unit (ReLU)



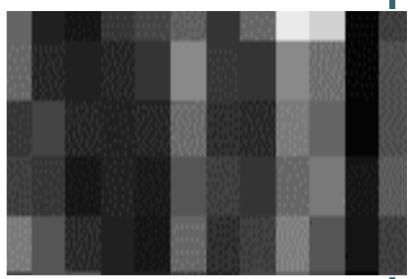
slide credit: S. Lazebnik

Convolutional Neural Networks





Max pooling

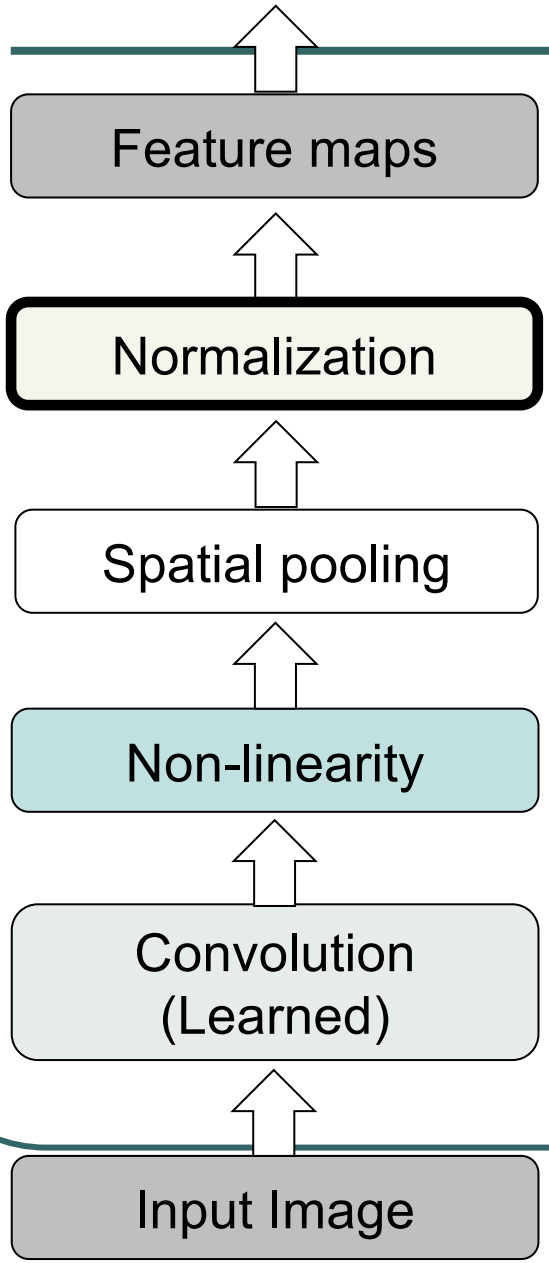


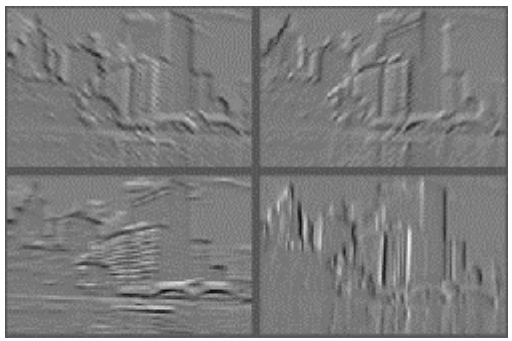
Max-pooling: a non-linear down-samp

Provide *translation invariance*

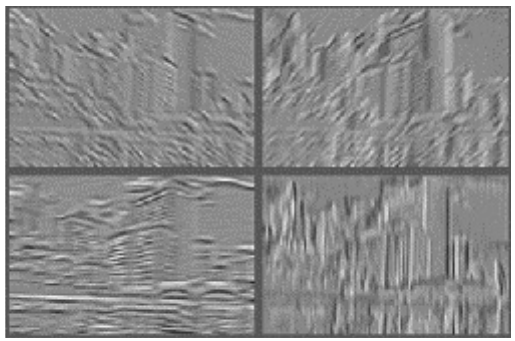
slide credit: S. Lazebnik

Convolutional Neural Networks





Feature Maps

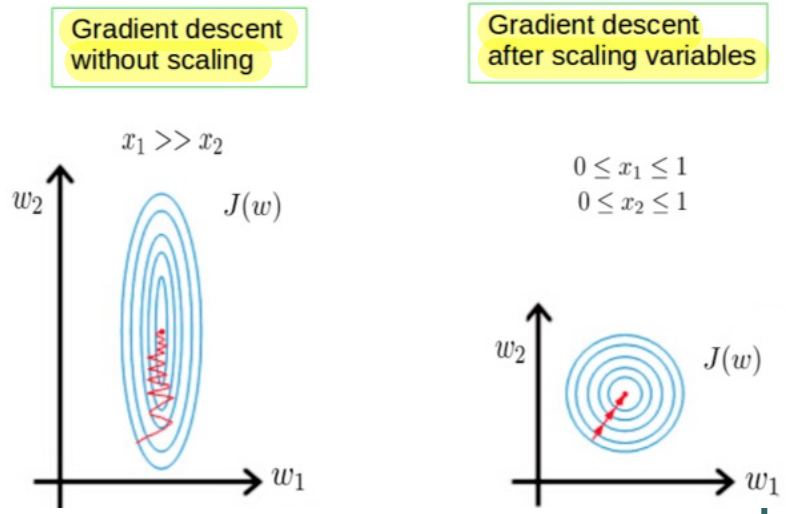


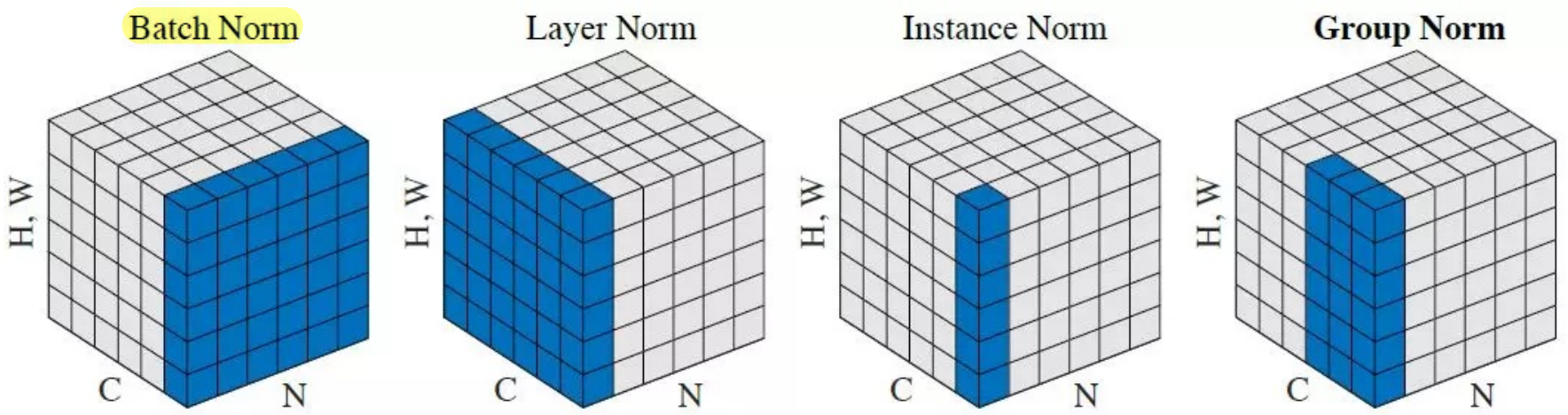
Feature Maps
After Contrast
Normalization

slide credit: S. Lazebnik

Normalization

- To improve the speed, performance, and stability of artificial neural networks.



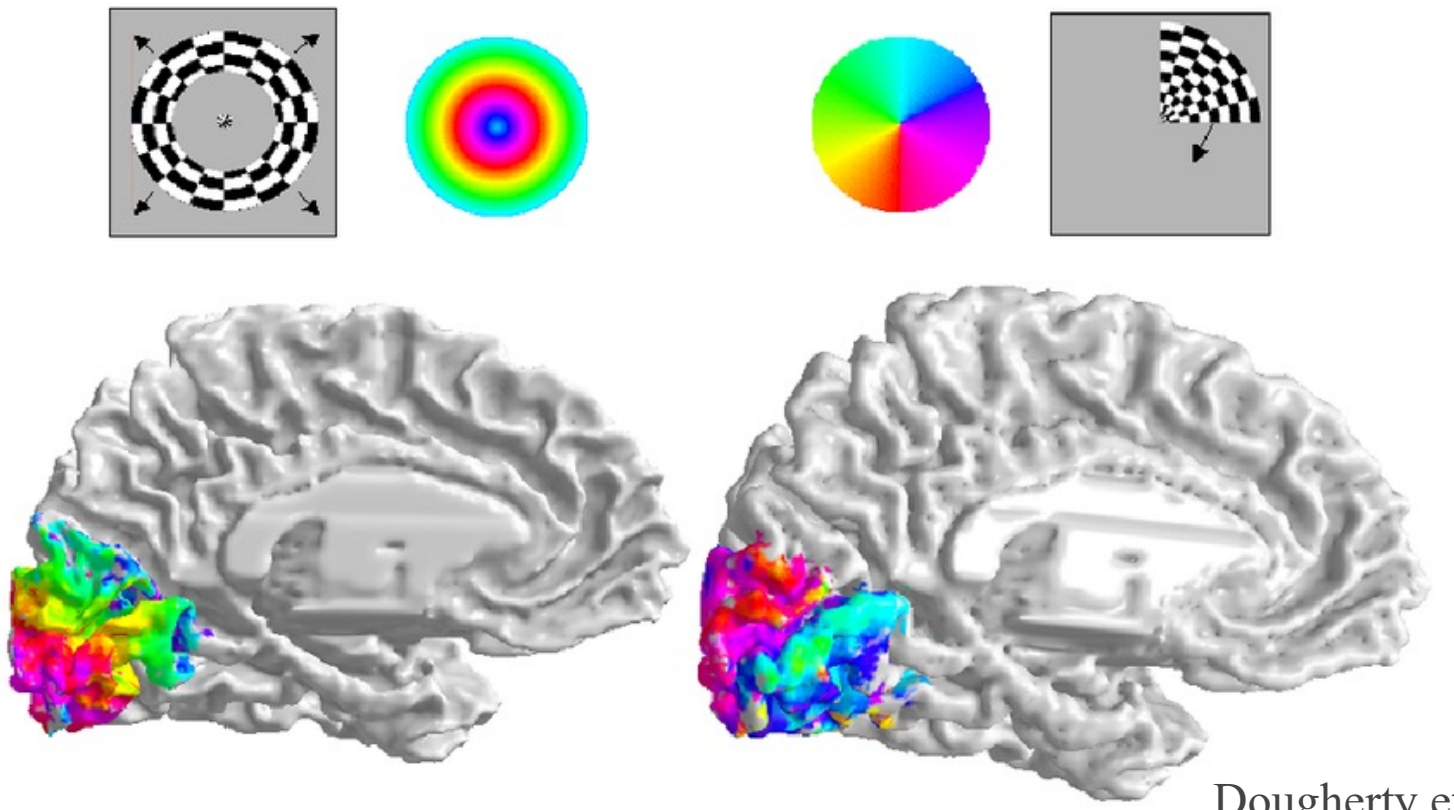


$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \text{ and } \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{{\sigma_B^{(k)}}^2 + \epsilon}}, \text{ where } k \in [1, d] \text{ and } i \in [1, m]$$

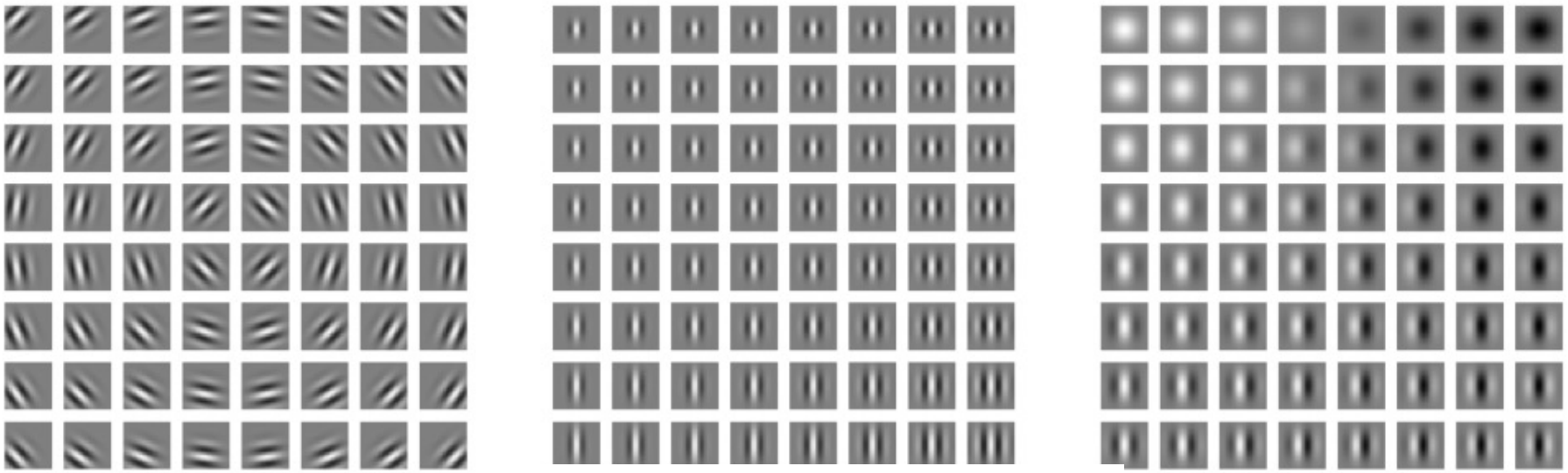
Retinotopy

- V1 is arranged in a spatial map called retinotopy.



Dougherty et al., 2003

Gabor Model for the Receptive Field of V1 Cell



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

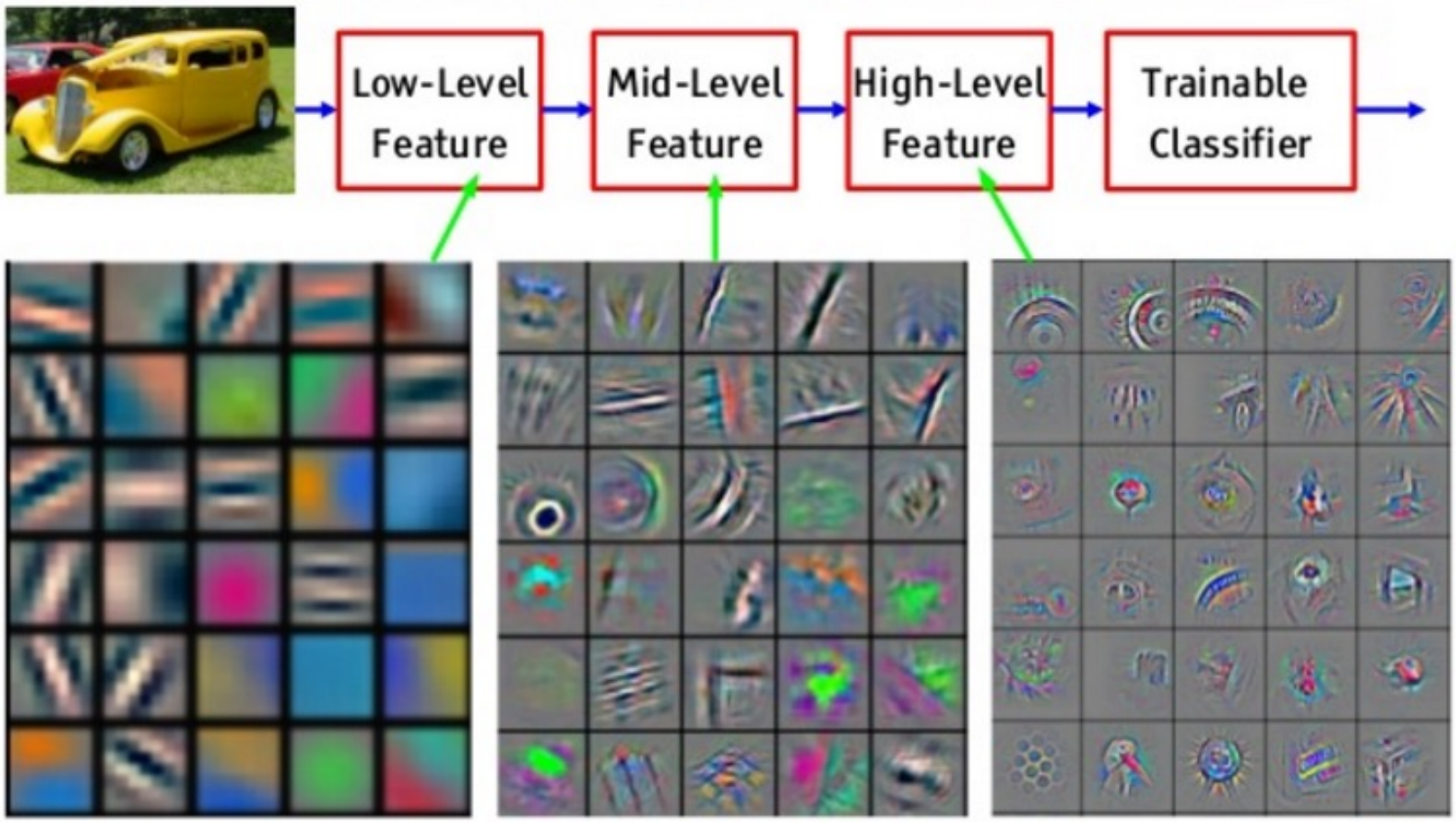
$$w(x, y; \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \exp\left(-\beta_x x'^2 - \beta_y y'^2\right) \cos(f x' + \phi),$$

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

- CNN:



adapted from Jeff Dean presentation 2016

Neocognitron [Fukushima, Biological Cybernetics 1980s]

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

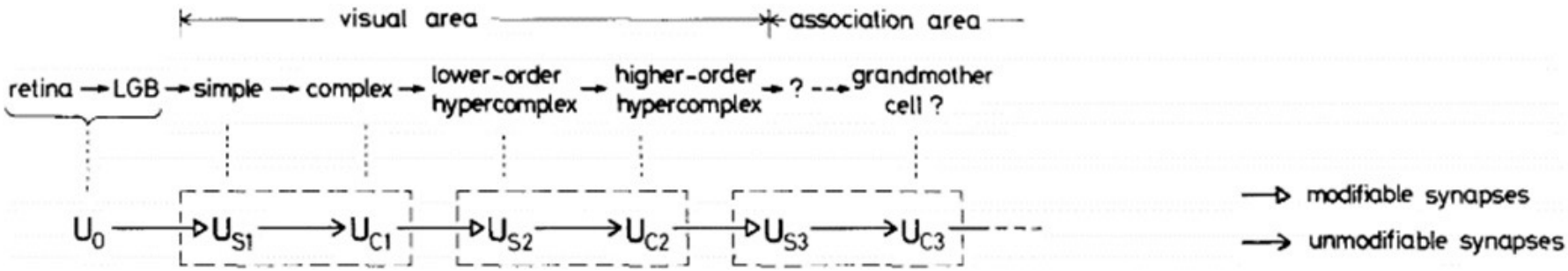
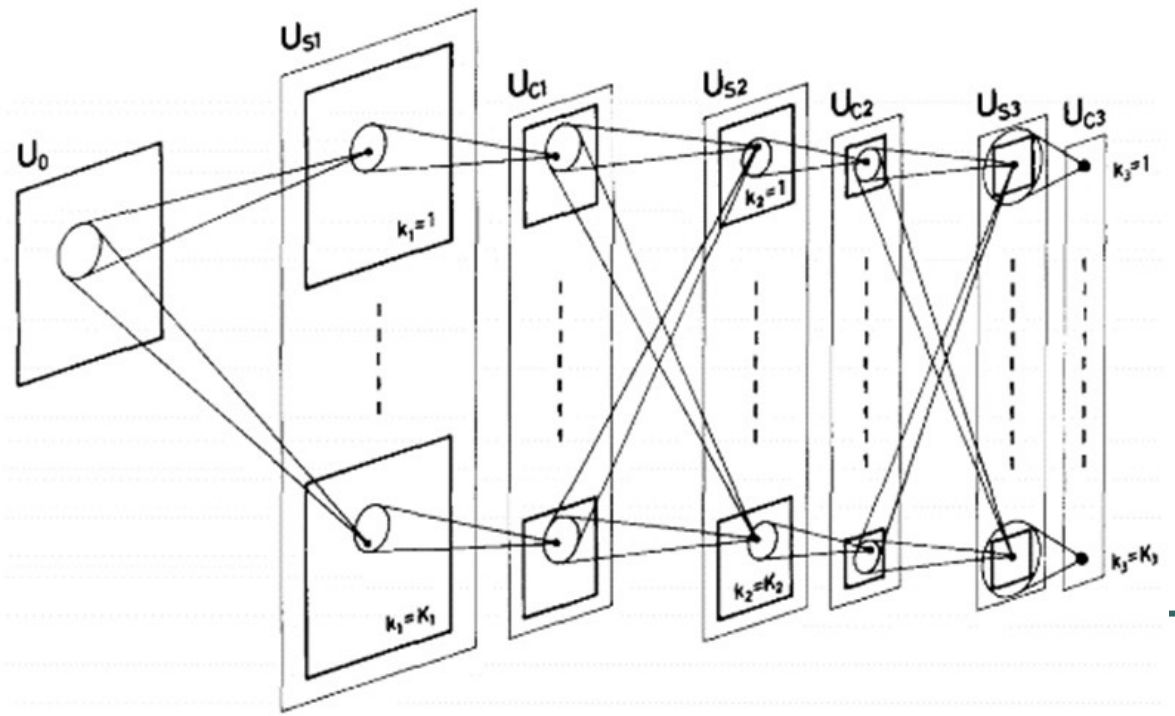


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

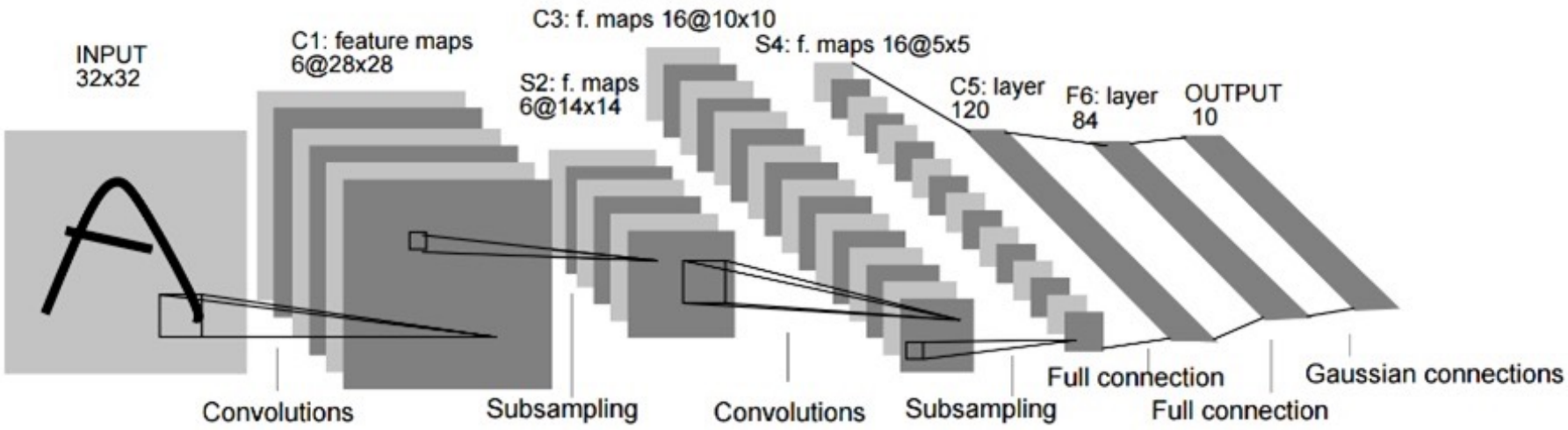


Deformation-Resistant Recognition

S-cells: (simple)
- extract local features

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

LeNet [LeCun et al. 1998]



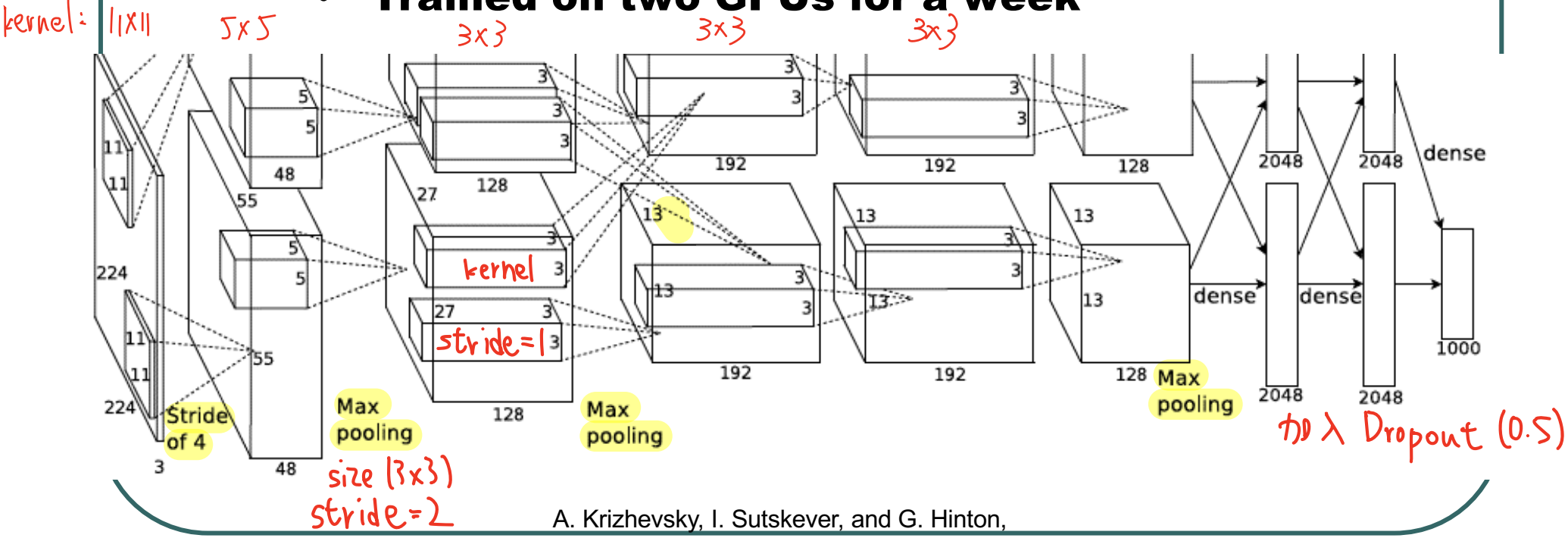
Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner 1998]

LeNet-1

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**

Use ReLU



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

IMAGENET

- Start in 2007 at Princeton
- Debut in CVPR' 2009
  - Total classes : 21.8K
  - Total images : 14M
- ILSVR Challenge: 2010 - 2017

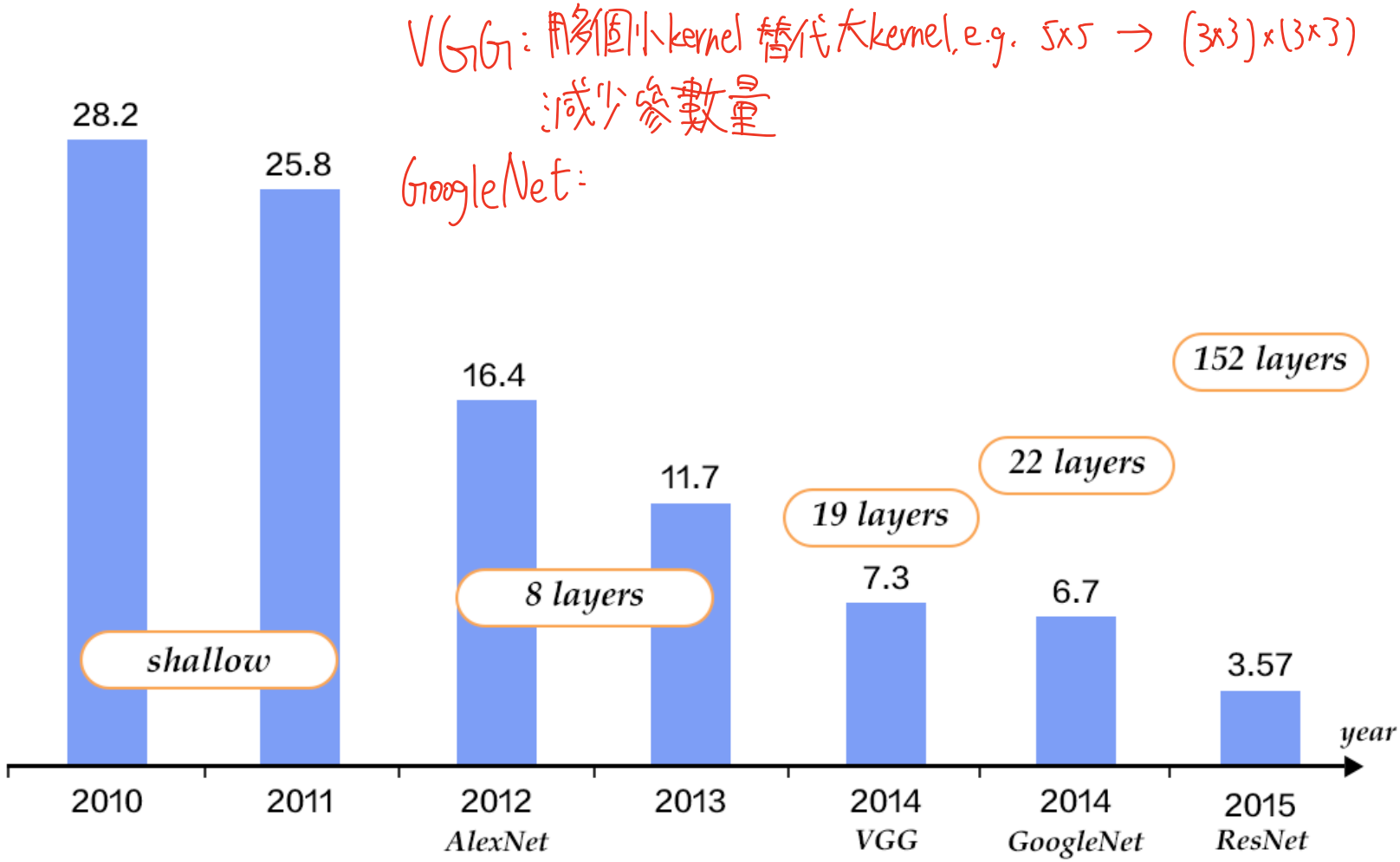
Jia Deng

Fei-Fei Li

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

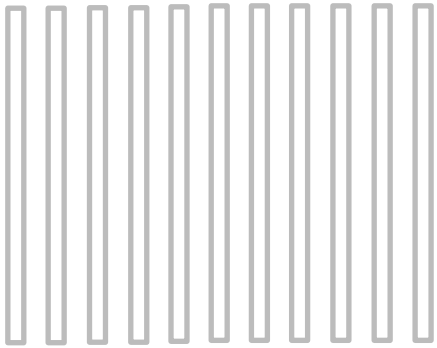
ImageNet Challenge 2012-2014

Best non-convnet in 2012: 26.2%

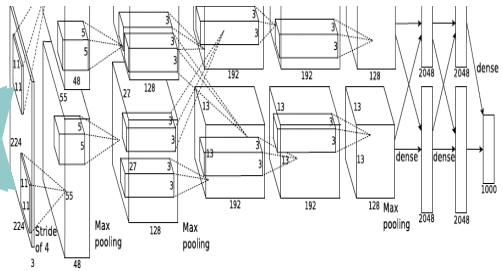


Using CNN for Image Classification

Fully connected layer Fc7
d = 4096



AlexNet



Averaging

Fixed input size:
224x224x3

d = 4096

Softmax Layer

"Jia-Bin"

Neuroscience-Inspired Recurrent Network

Recurrent system of human brain

(*IEEE ISPACS*, 2017)

Perception of an ambiguous figure is affected by own-age social biases, *Scientific Reports*, 2018

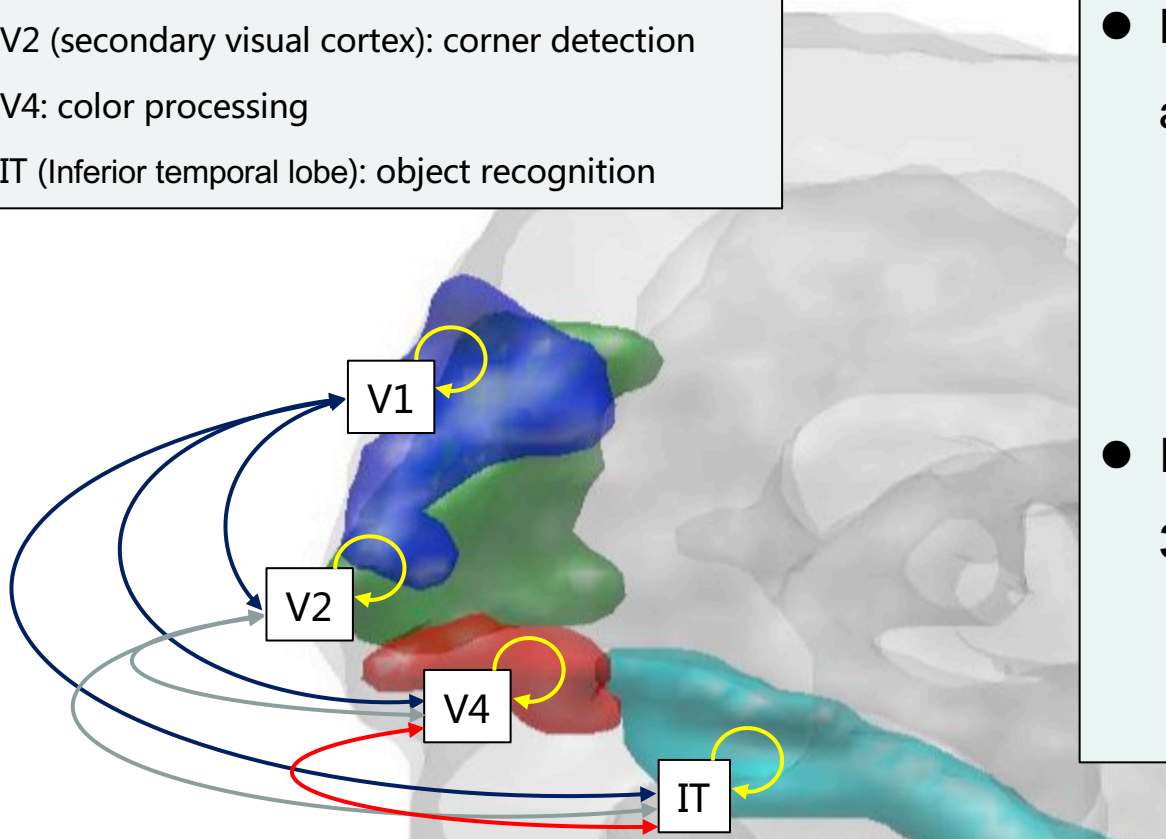
Human brain is a recurrent system

V1 (primary visual cortex): orientation

V2 (secondary visual cortex): corner detection

V4: color processing

IT (Inferior temporal lobe): object recognition



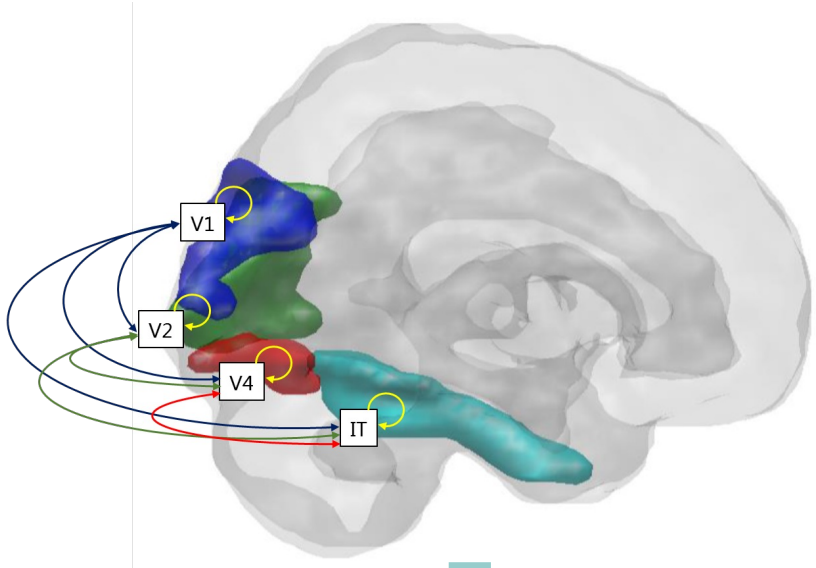
- Human can recognize image in a single forward
  - Potter et al. *AP&P* 2014
  - Takes about **13** ms
  - 63% accuracy
- In normal, human takes about **350** ms to recognize an object
  - Almost 100% accuracy
  - Lots of recurrences

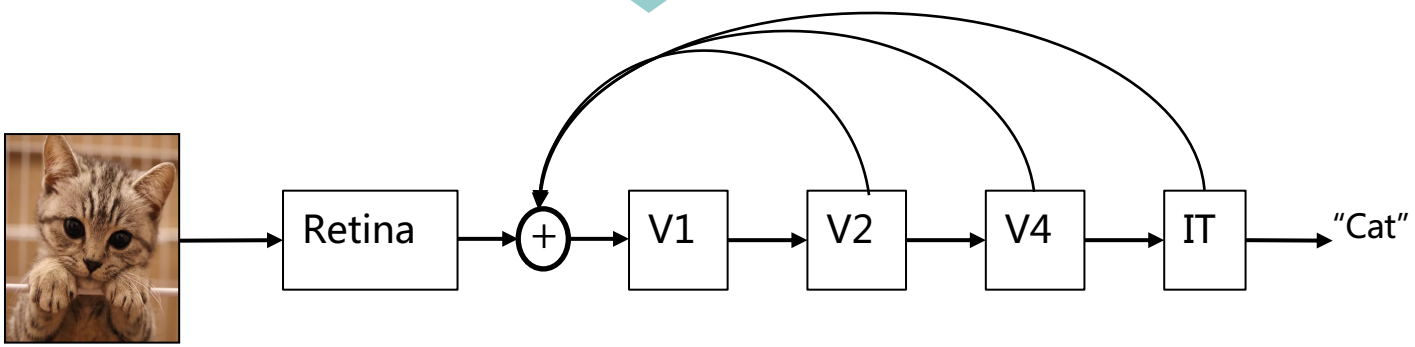
Local and long-range connection

- Top-down modulation

Ventral Stream: Object Recognition

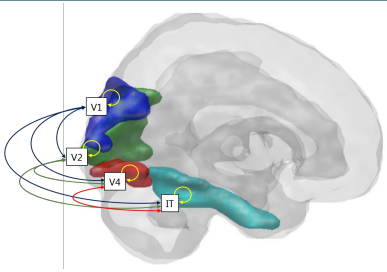
Proposed recurrent model

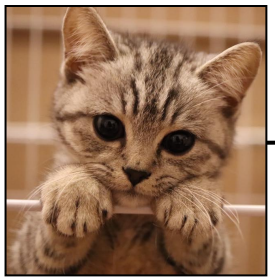




A recurrent network has long-range connection

Unfold Recurrent Model to A Forward Network





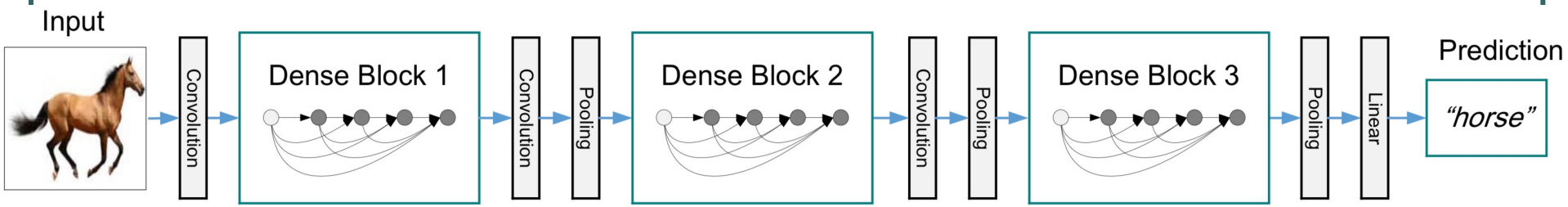
Retina + V1 V2 V4 IT "Cat"

Recurrence 2

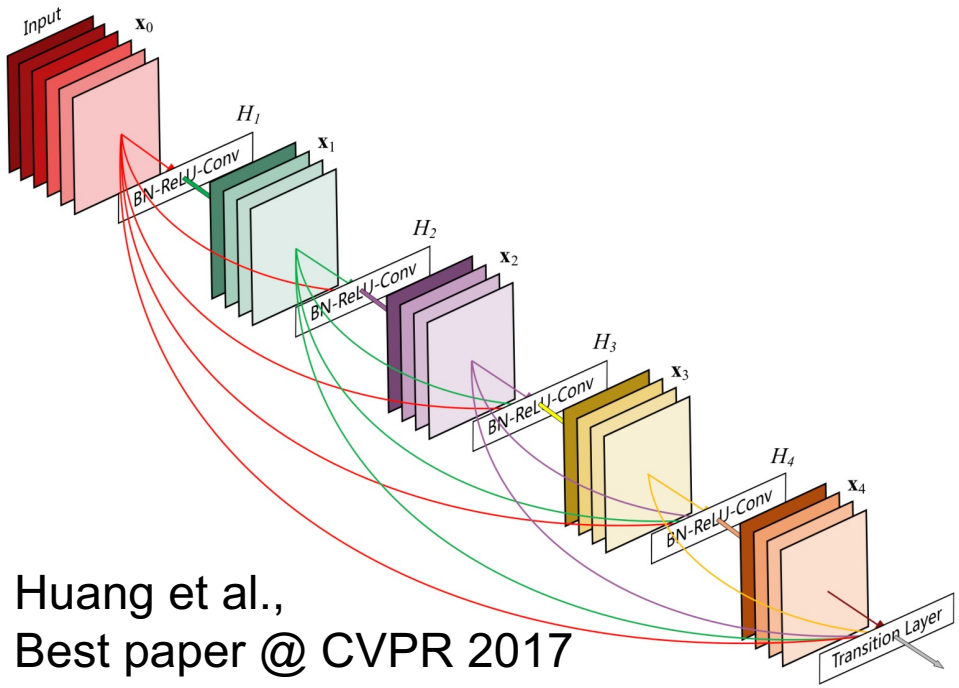
Retina V1 V2 V1 V2 V4 V1 V2 V1 V2 V4 IT ..... "Cat"

conv conv conv ...

DenseNet



- Feature reuse
- High parameter efficiency



Huang et al.,
Best paper @ CVPR 2017

Beyond Classification

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

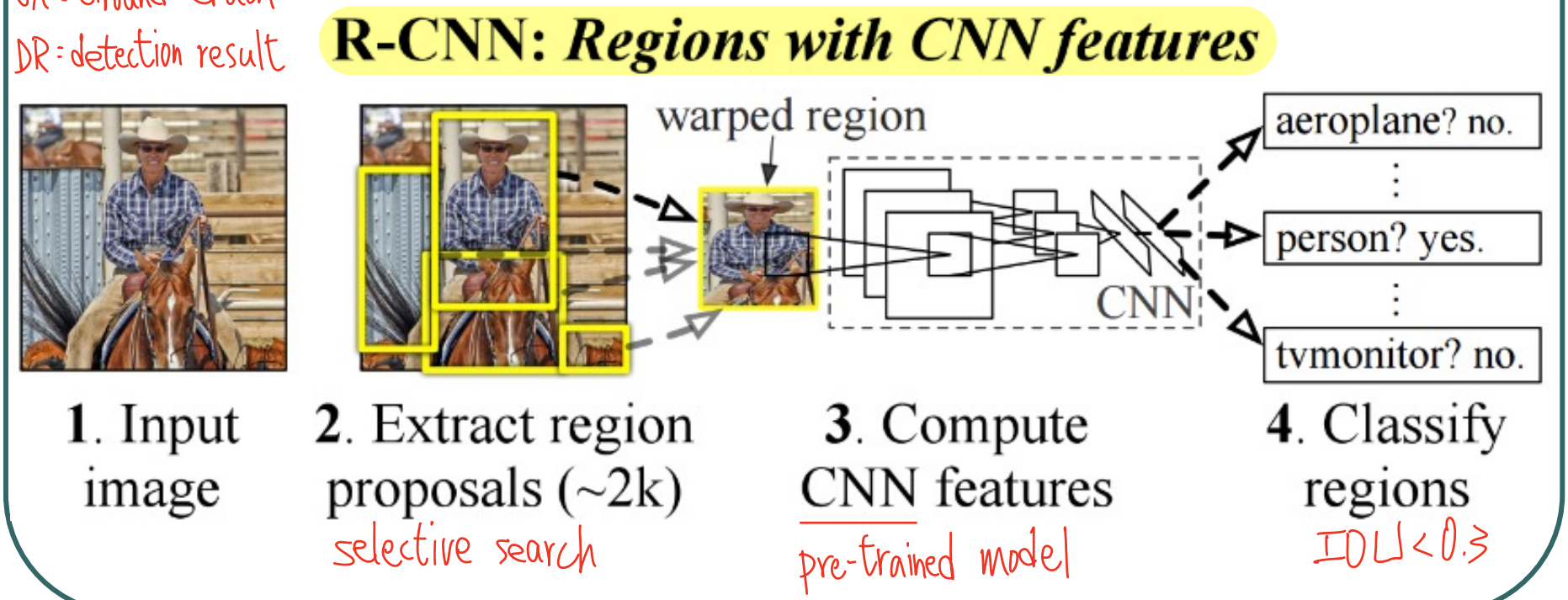
and many more…

R-CNN: Regions with CNN features

- Trained on ImageNet classification
- Finetune CNN on PASCAL (data sets for object classification)

$IOU = \frac{GT \cap DR}{GT \cup DR}$, if IOU > 0.5 : positive, use to fine-tune CNN

GT: Ground truth
DR: detection result



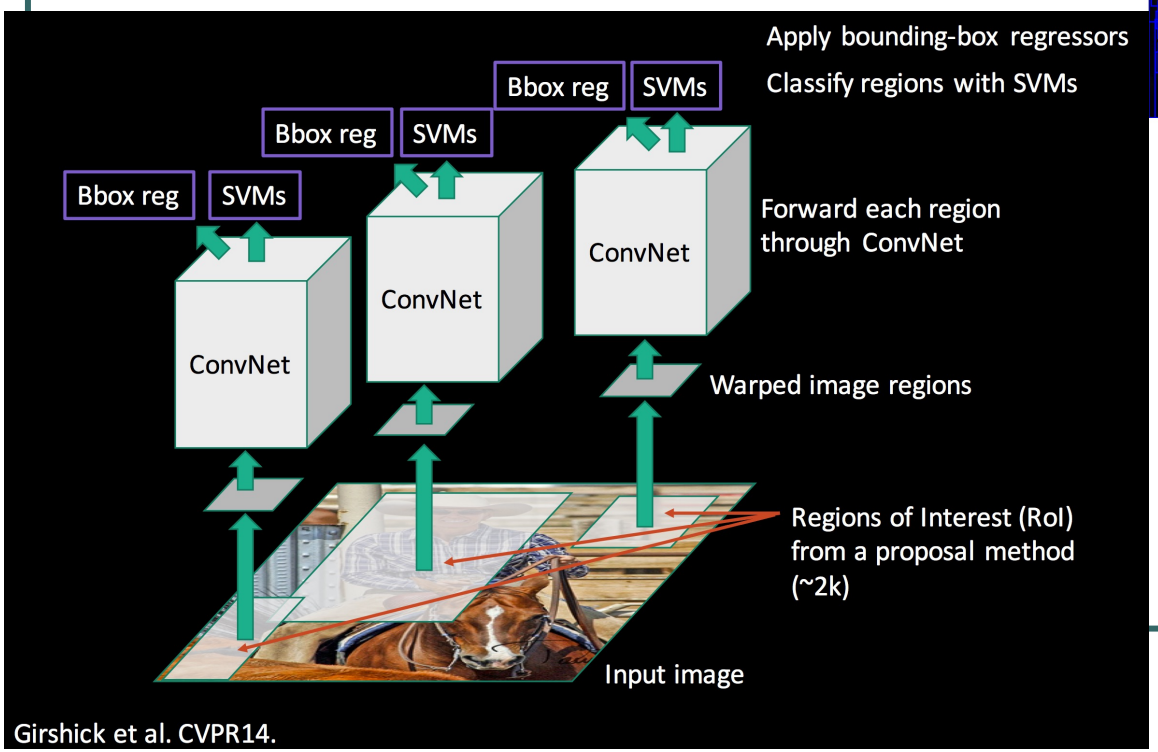
RCNN [Girshick et al. CVPR 2014]

R-CNN: Regions with CNN features

Problem: 1. 每一個 region proposal 都要經過一次 CNN 計算 → 超慢
2. not end-to-end struture

- Selective search for region proposals

van de Sande et al. ICCV'11)

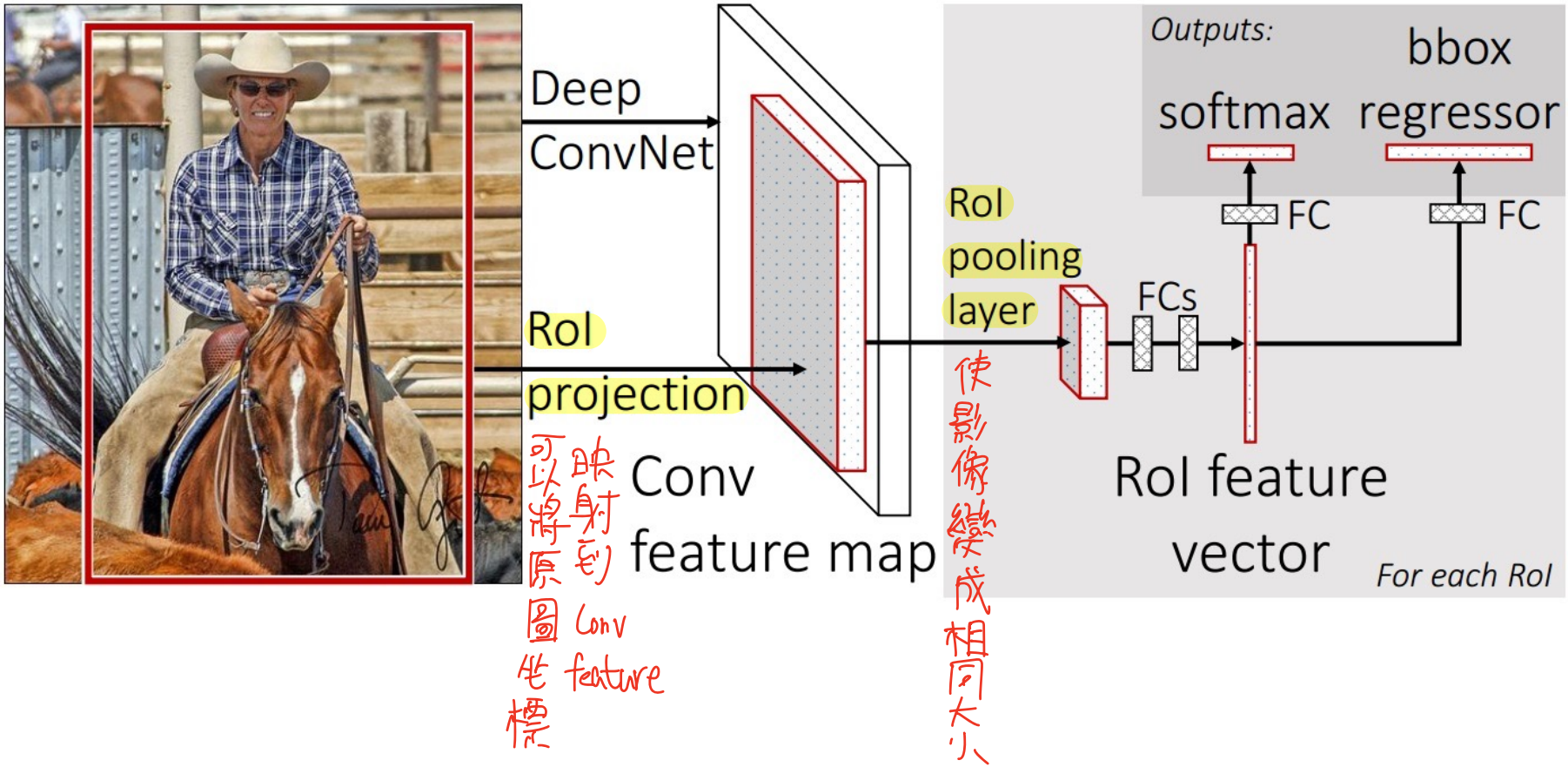


Girshick et al. CVPR14.

RCNN [Girshick et al. CVPR 2014]

Fast R-CNN

Region of interesting (ROI)



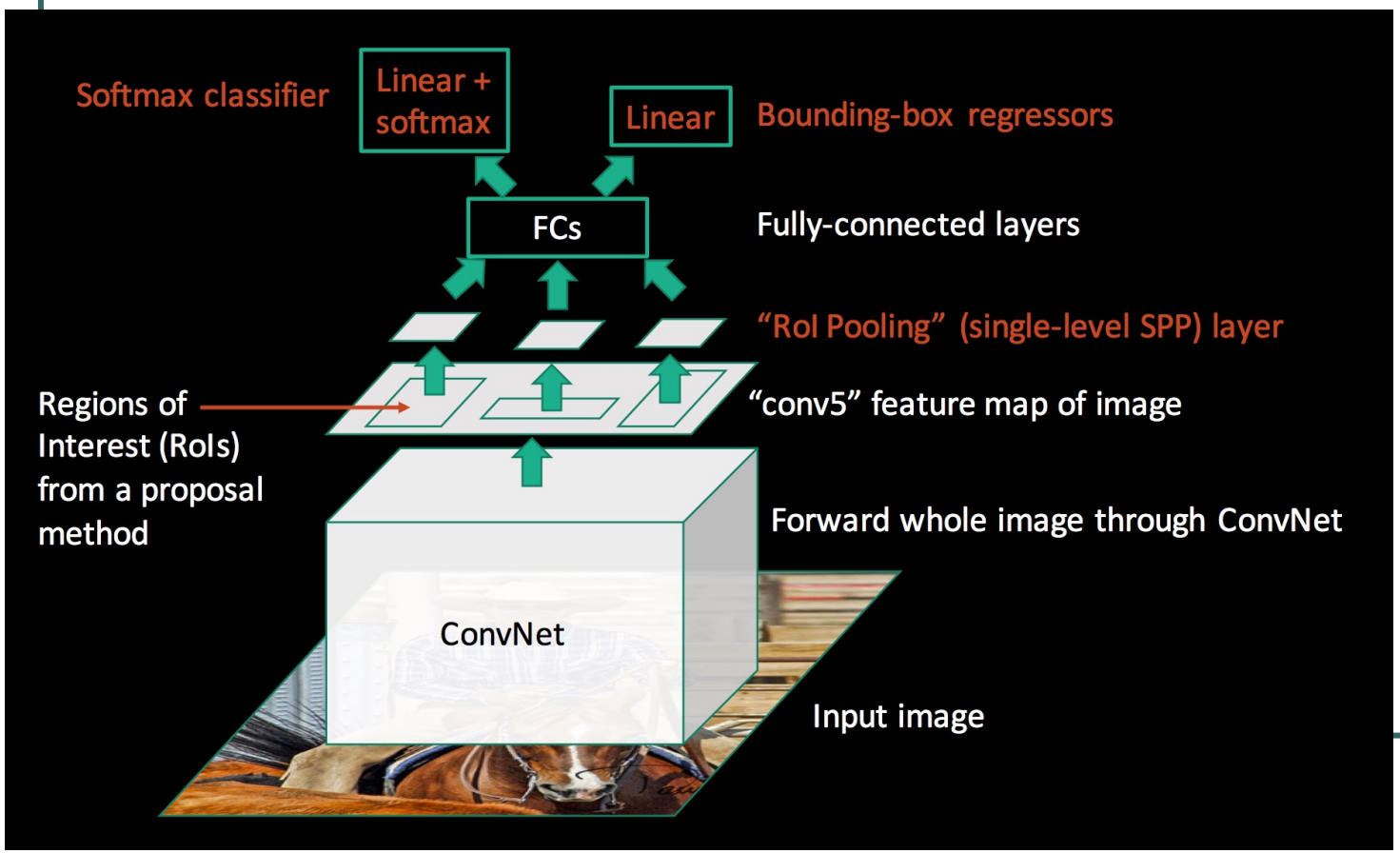
Fast RCNN [Girshick, R 2015]

https://github.com/rbgirshick/fast-rcnn

Fast R-CNN

Problems: still use selective search to decide ROI → too slow

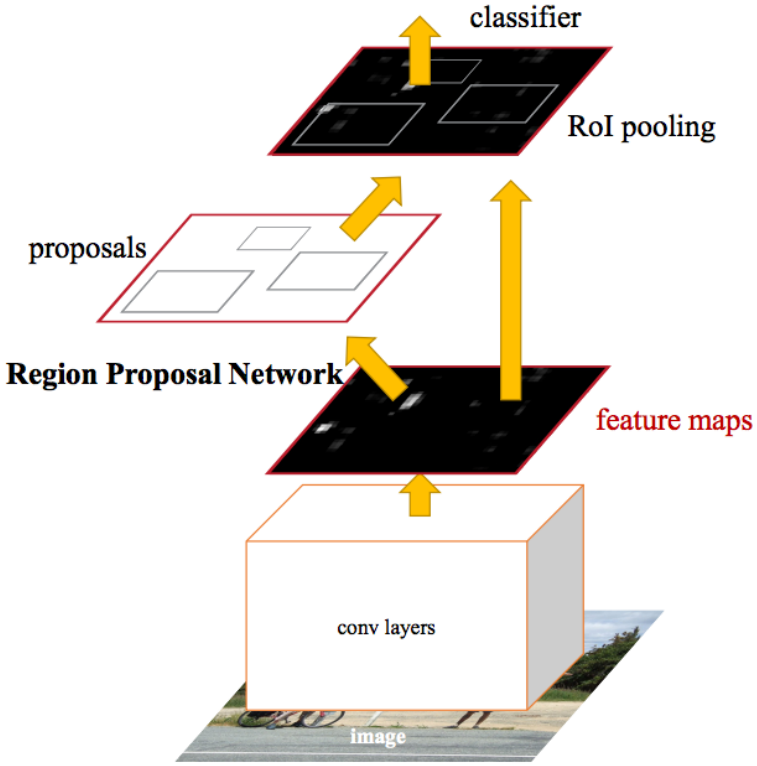
- Fast (avoid duplicate calculation of CNN features)

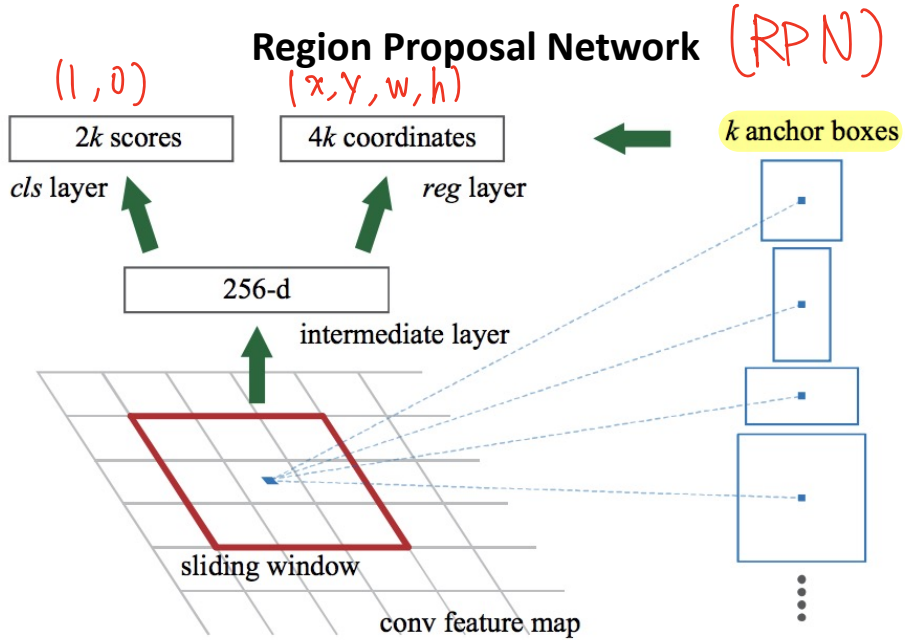


Fast RCNN [Girshick, R 2015]

Faster R-CNN

- Determine region proposals on feature map, instead of image.

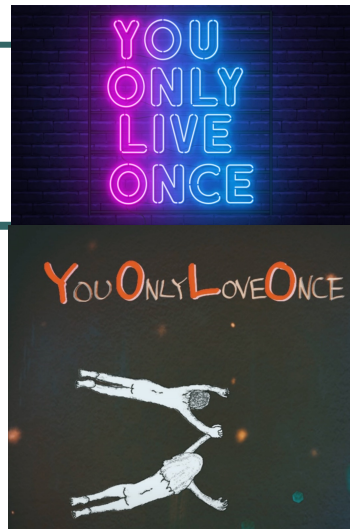




Backbone, eg, VGG16: 7x7x256

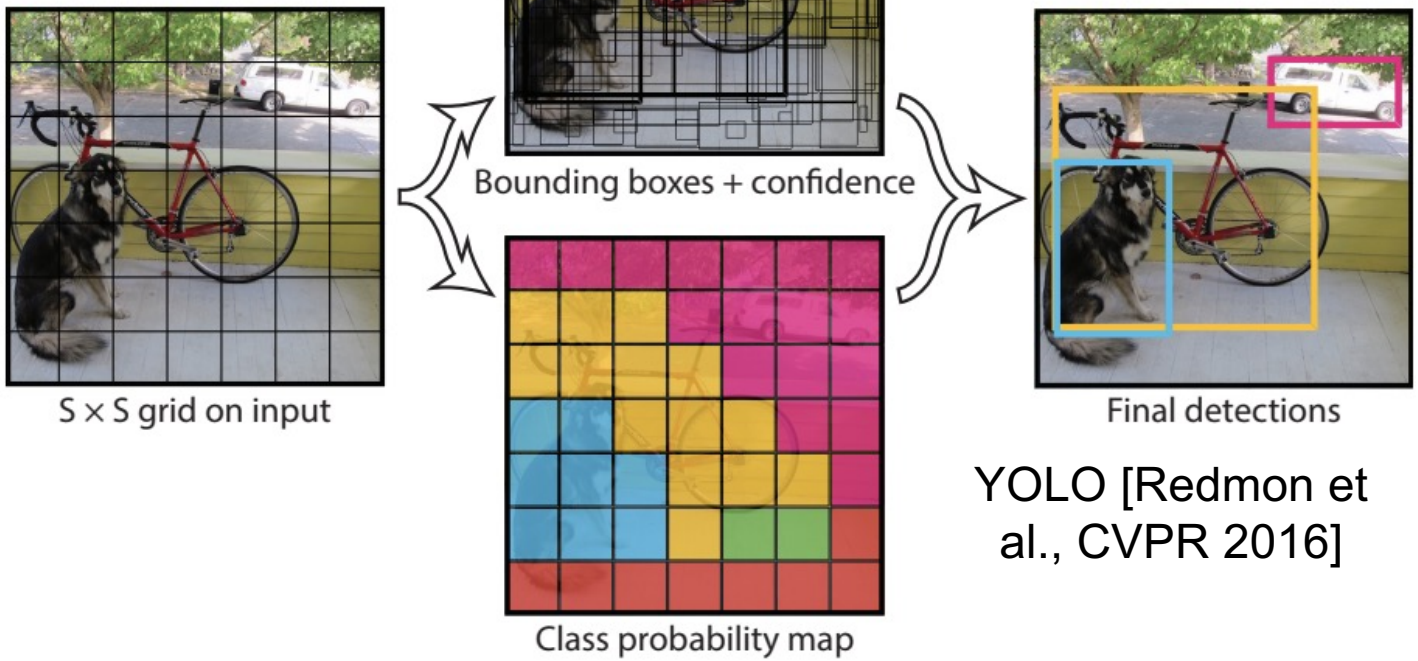
Faster RCNN [Ren, S, et al., 2015]

YOLO (You Only Look Once)



- YOLO prediction vector:
  - SxSx(Bx5+C) E.g., 7x7x(2x5+20)
  - B: #bounding boxes
  - C: #classes
  - 5: x,y,w,h,confidence

confidence
= Pr(object) * IOU(ground truth)



YOLO [Redmon et al., CVPR 2016]

YOLO (You Only Look Once)

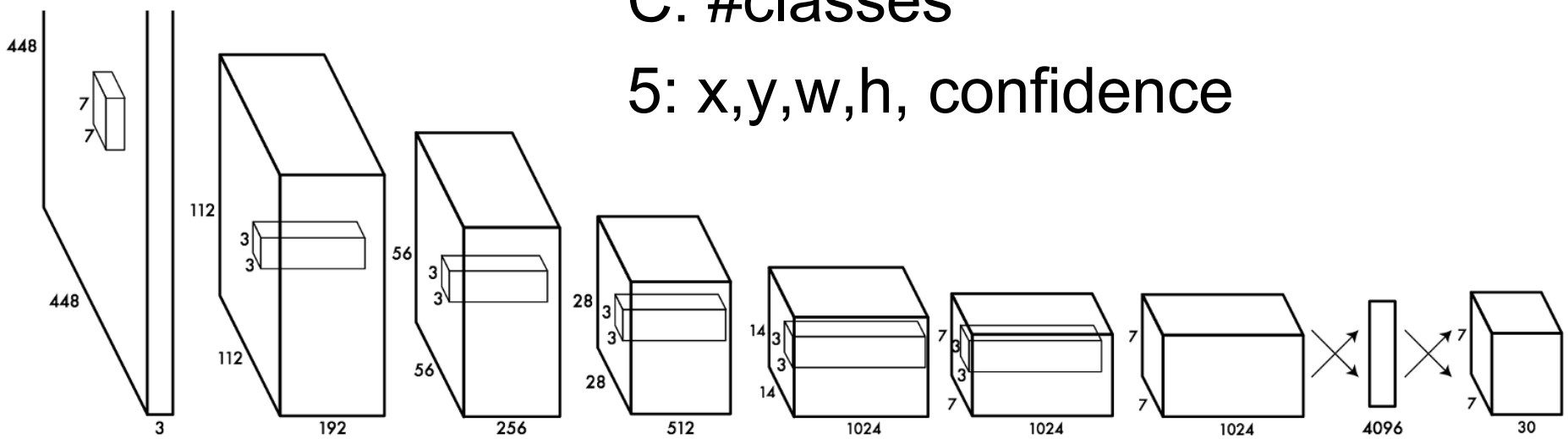
- YOLO prediction vector:

  SxSx(Bx5+C) E.g., 7x7x(2x5+20)

  B: #bounding boxes

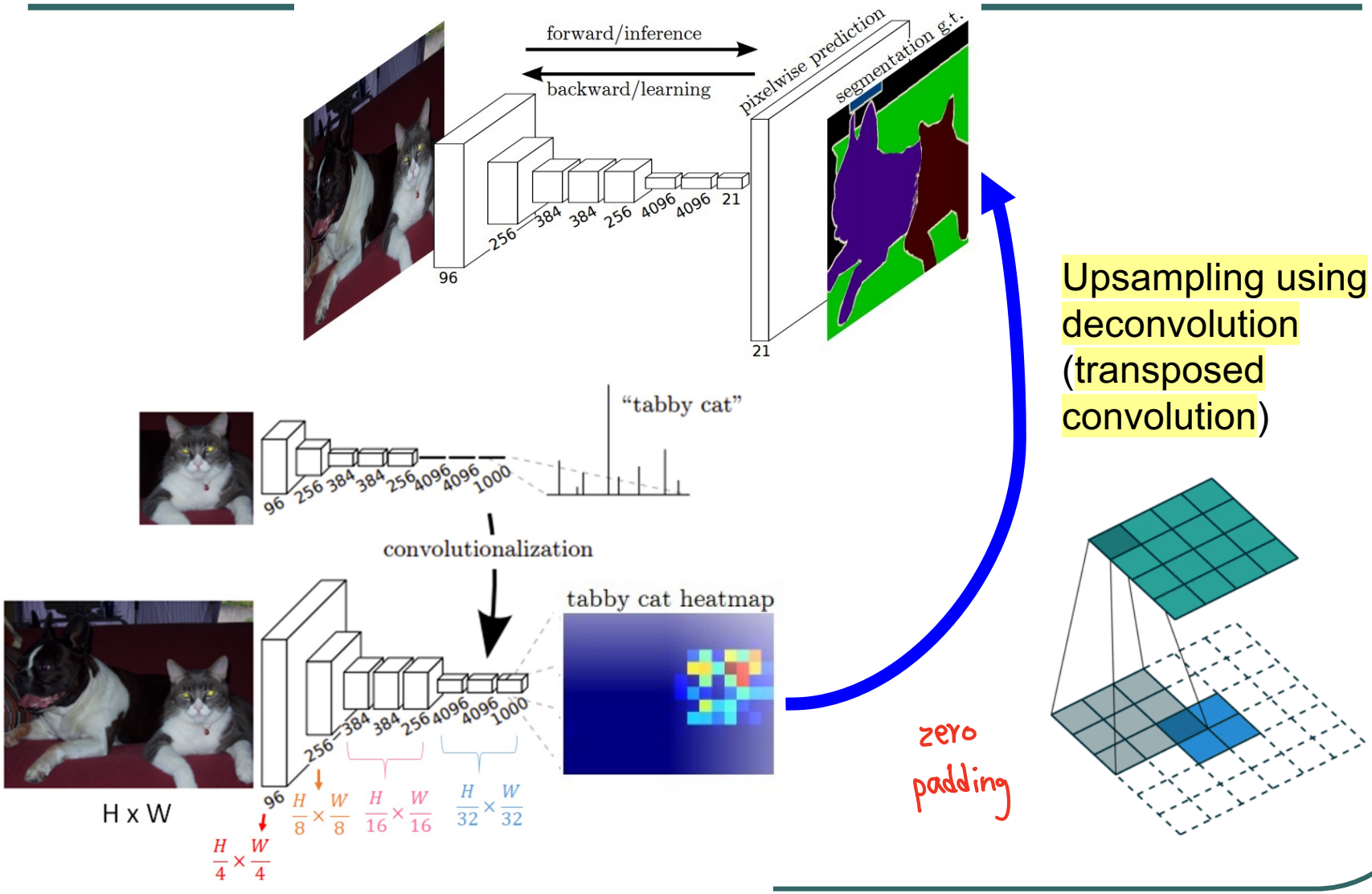
  C: #classes

  5: x,y,w,h, confidence



YOLO [Redmon et al., CVPR 2016]

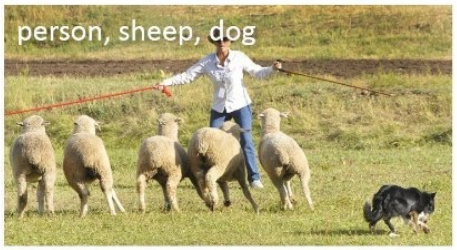
Labeling Pixels: Semantic Labels



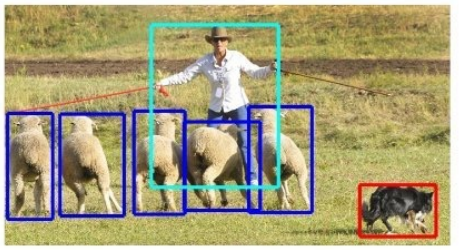
Fully Convolutional Networks for Semantic Segmentation [Long et al. CVPR 2015]

Mask R-CNN

- Instance segmentation with Mask R-CNN

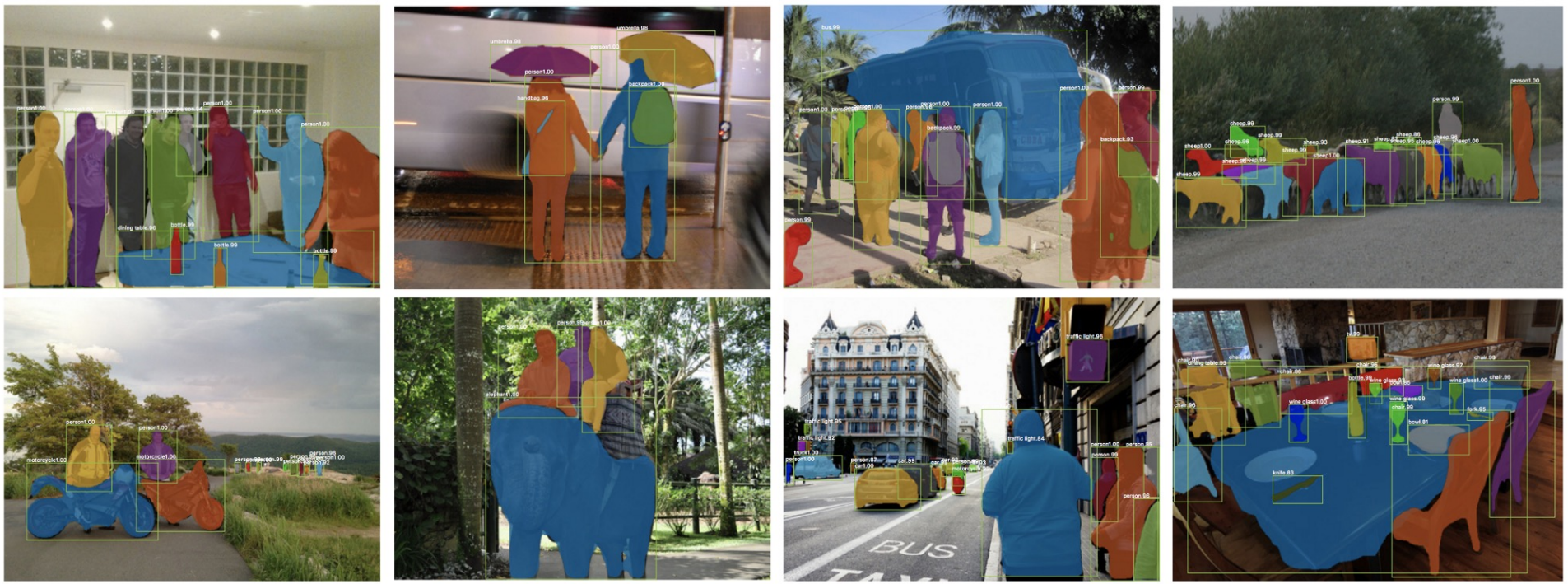


(a) Image classification


(b) Object localization


(c) Semantic segmentation


(d) Instance segmentation

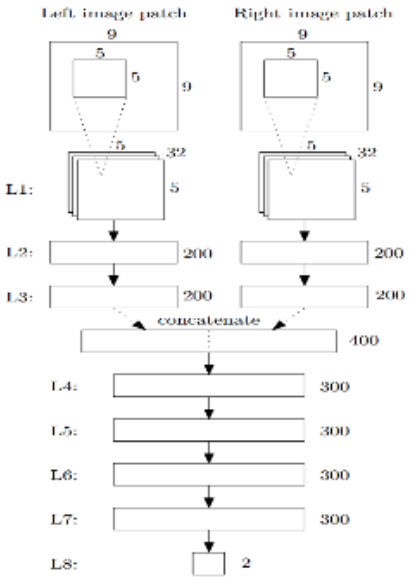


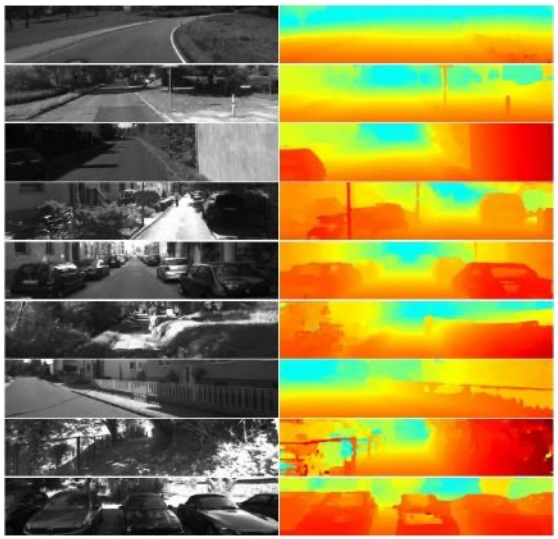
Mask RCNN [Kaiming He, 2017]

CNN for Regression

DeepPose [Toshev and Szegedy CVPR 2014]

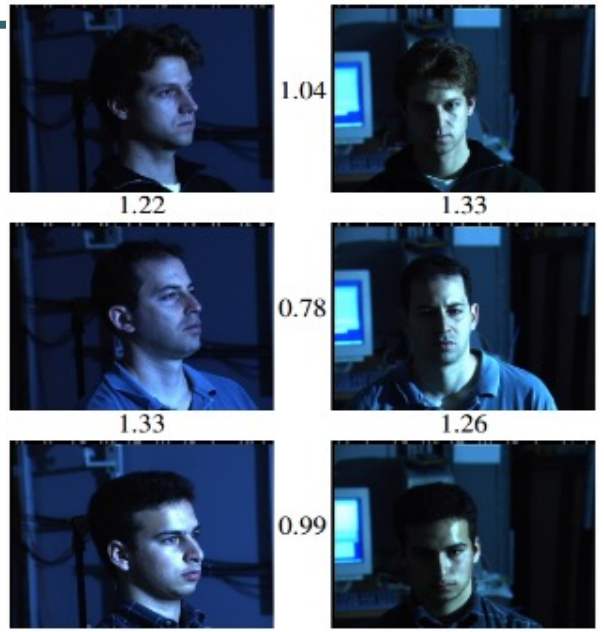
CNN as a Similarity Measure for Matching



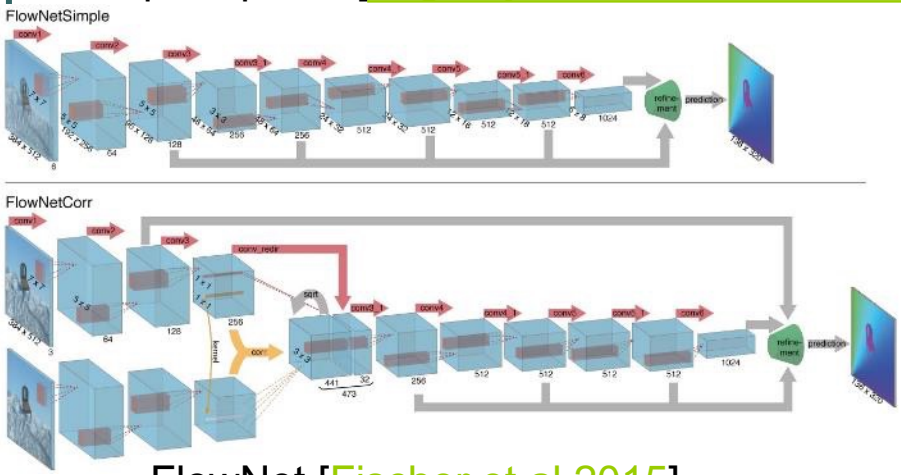


Stereo matching [Zbontar and LeCun CVPR 2015]

Compare patch [Zagoruyko and Komodakis 2015]



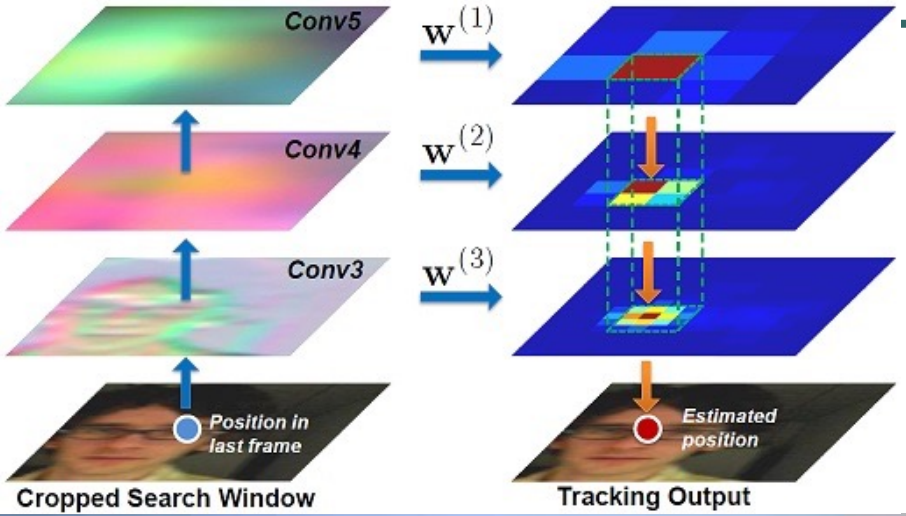
FaceNet [Schroff et al. 2015]



FlowNet [Fischer et al 2015]

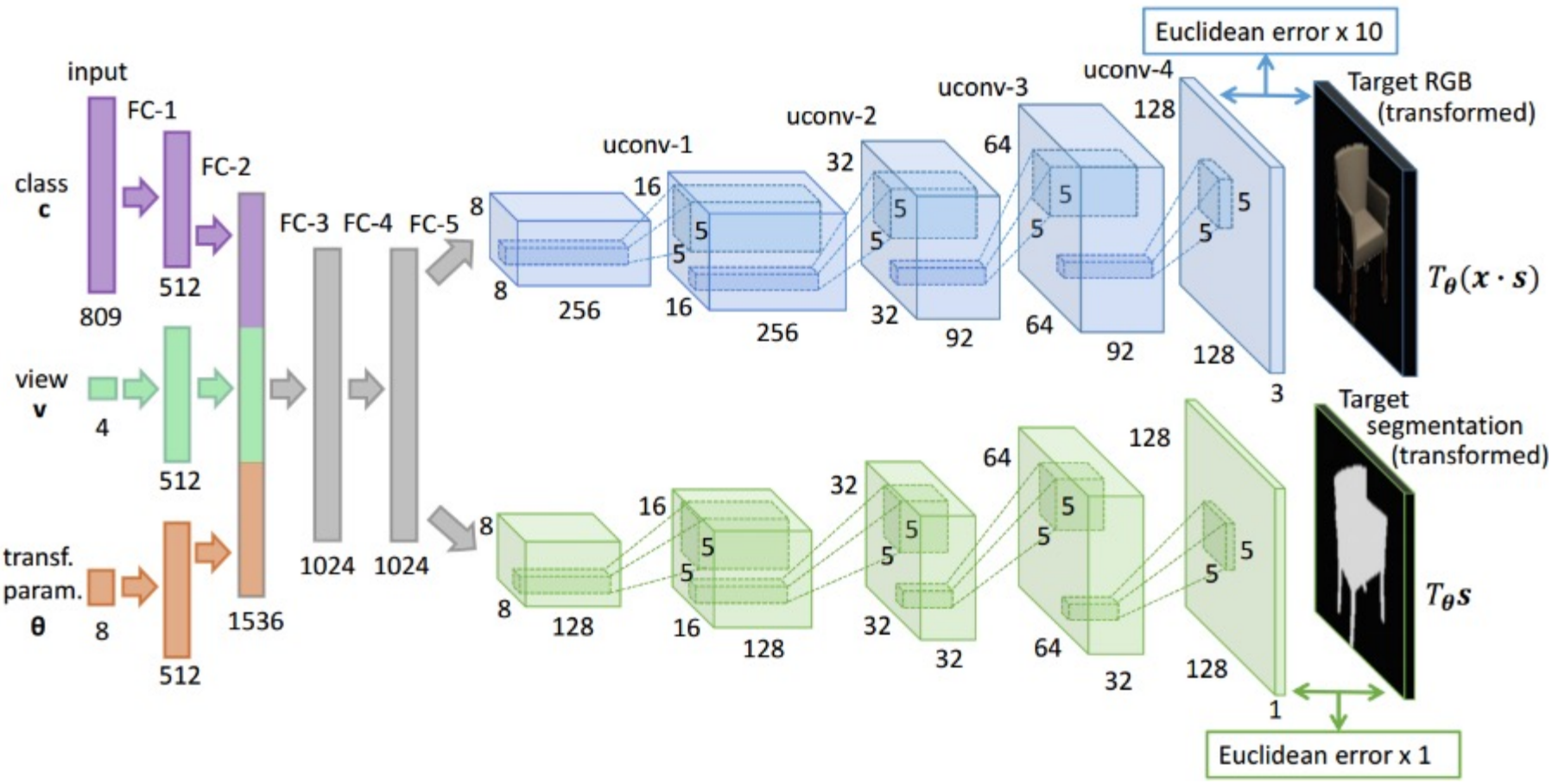
Match ground and aerial images
[Lin et al. CVPR 2015]

CNN for Online Visual Tracking



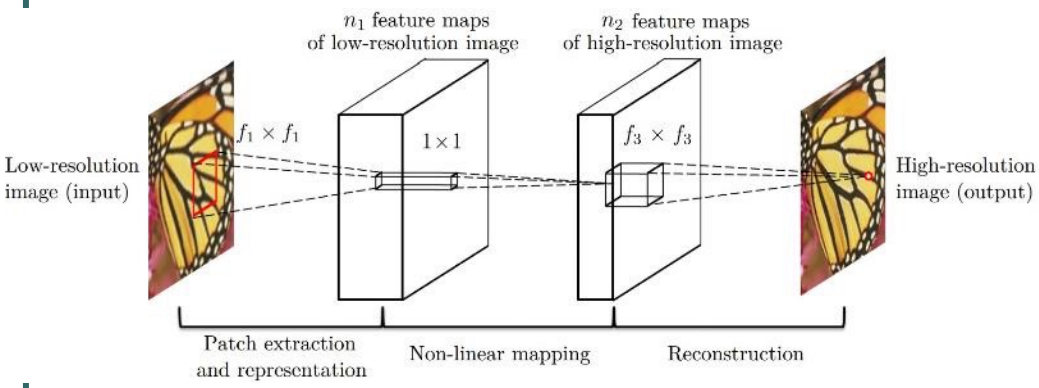
Hierarchical Convolutional Features for Visual Tracking [Ma et al. ICCV 2015]

CNN for Image Generation

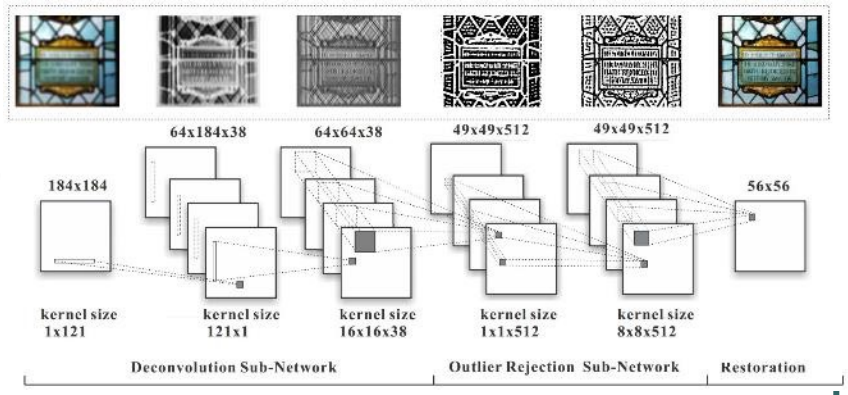


Learning to Generate Chairs with Convolutional Neural Networks [Dosovitskiy et al. CVPR 2015]

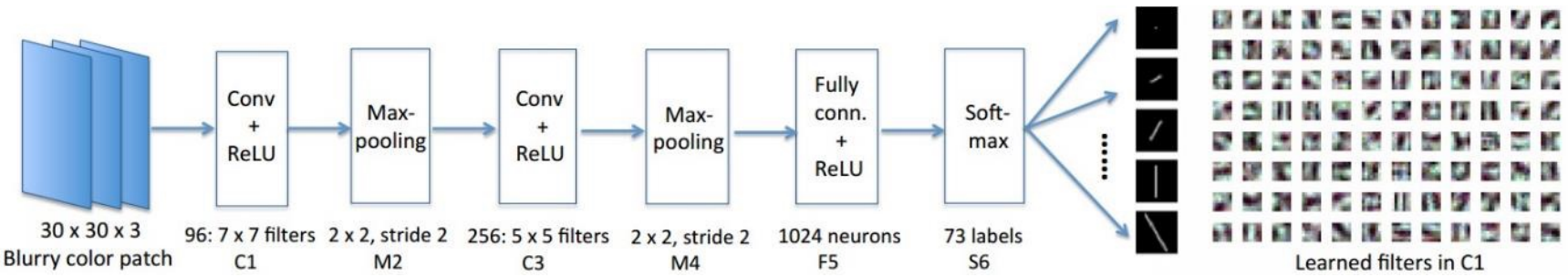
CNN for Image Restoration/Enhancement



Super-resolution
[Dong et al. ECCV 2014]



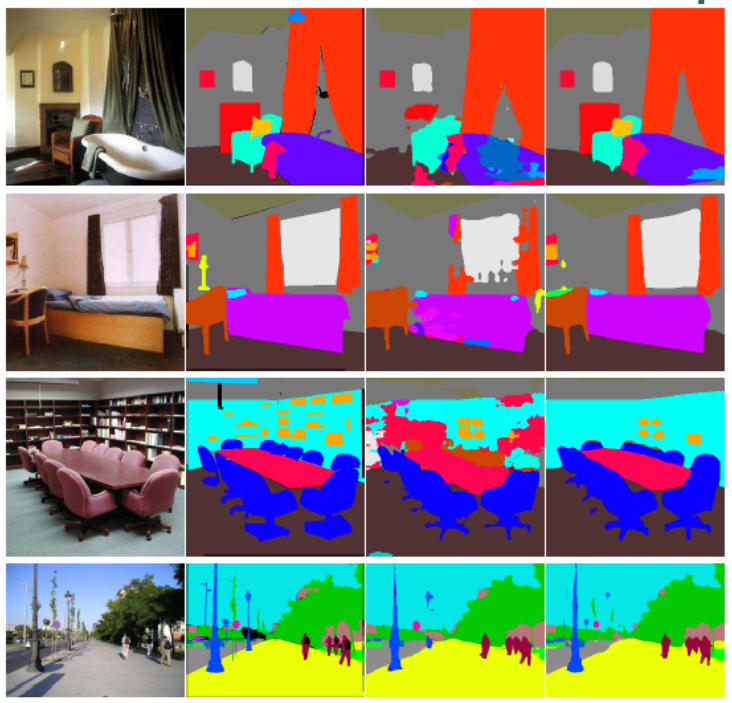
Non-blind deconvolution
[Xu et al. NIPS 2014]



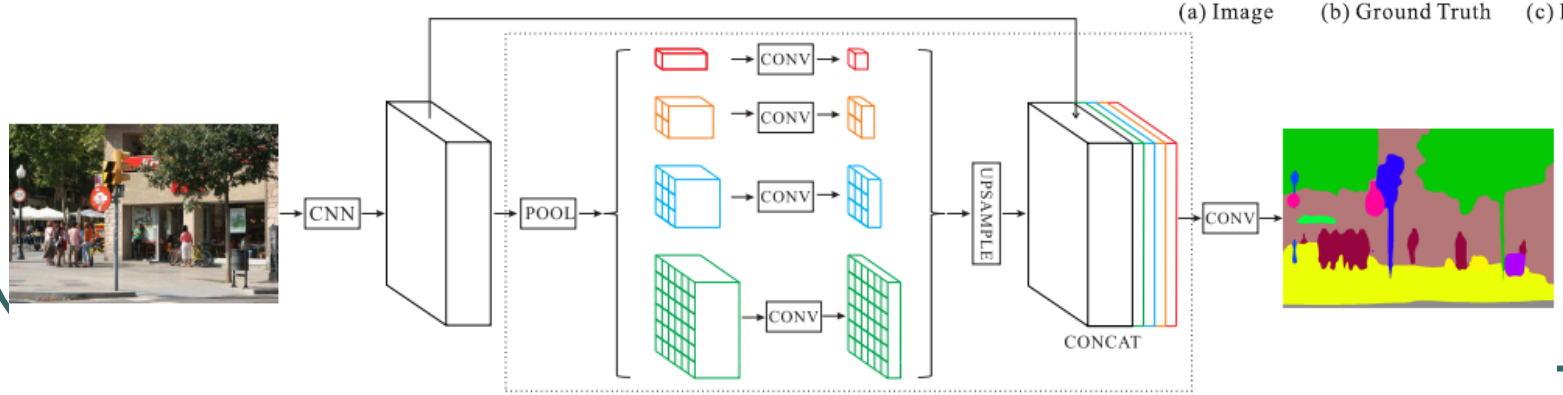
Non-uniform blur estimation
[Sun et al. CVPR 2015]

Semantic segmentation

- Pyramid Scene Parsing, PSPNet (CVPR 2017)



(a) Image (b) Ground Truth (c) Baseline (d) PSPNet

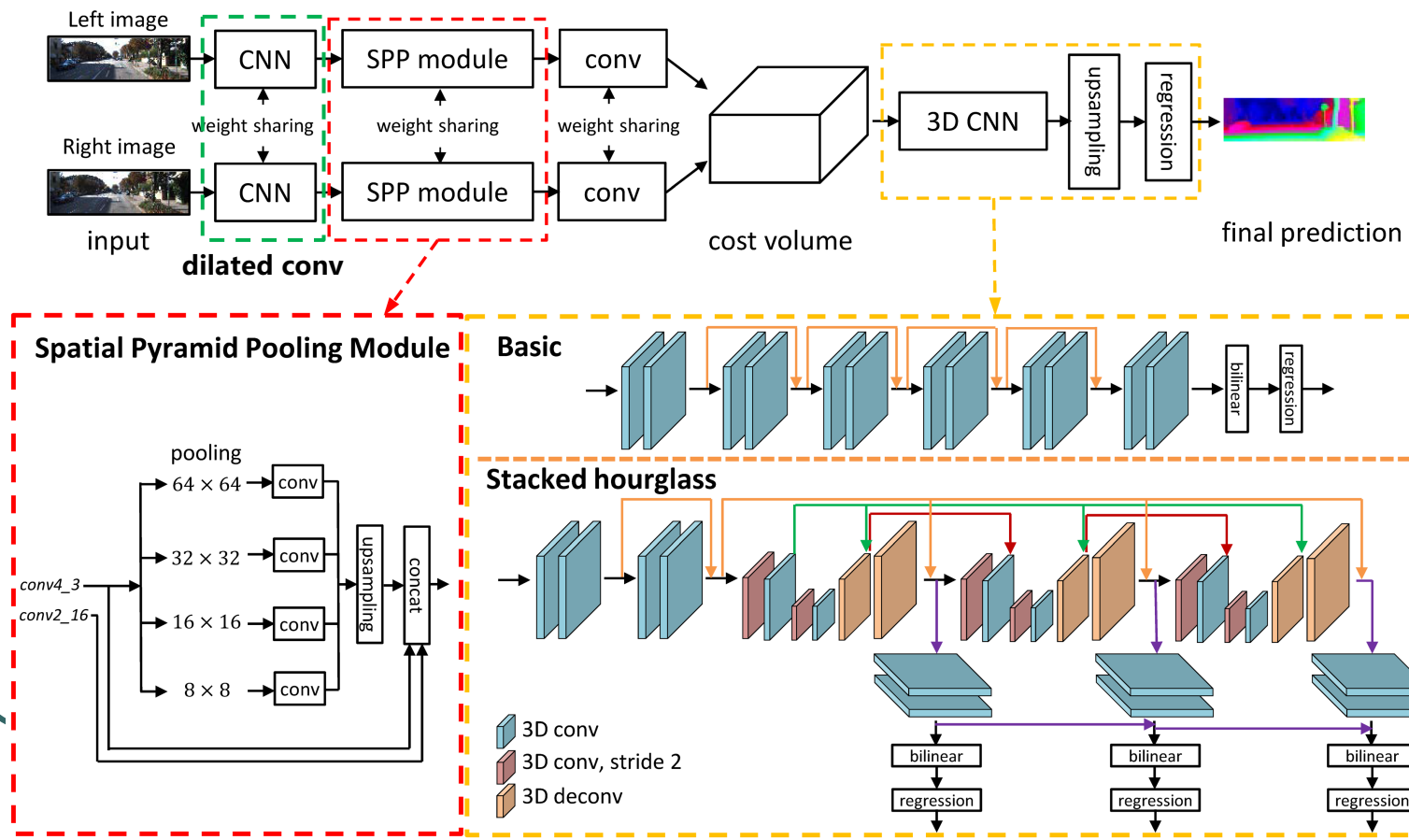


(a) Input Image (b) Feature Map (c) Pyramid Pooling Module (d) Final Prediction

Pyramid Stereo Matching Network

Our Solution: Exploiting Global Context

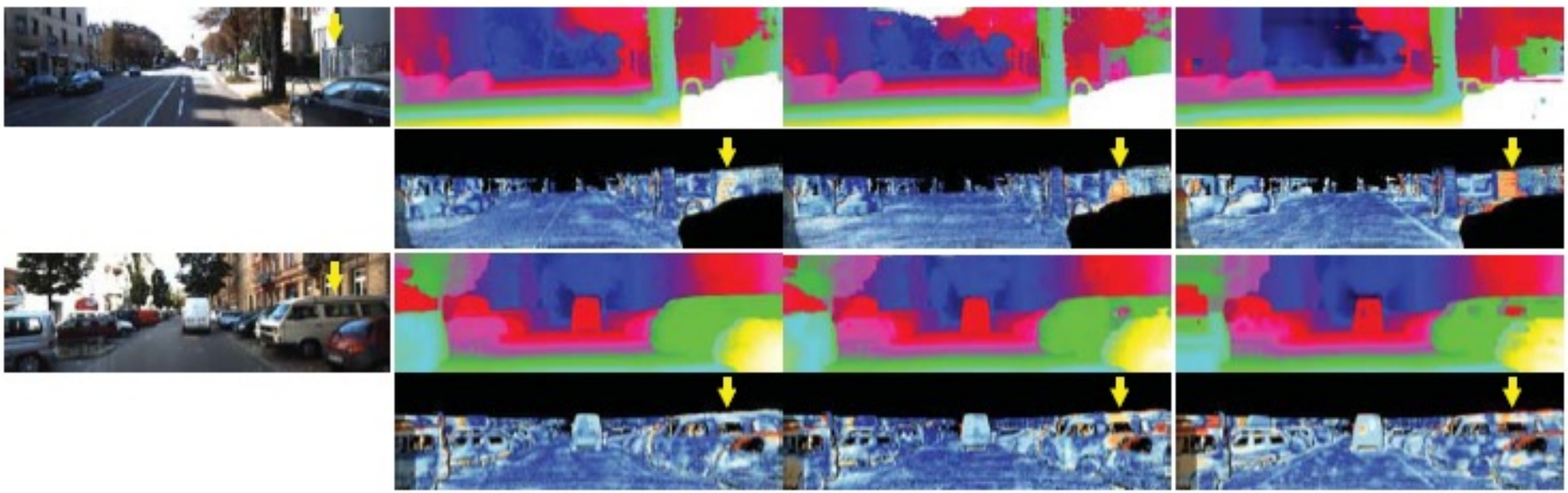
- Pyramid Stereo Matching Network (PSMNet)
- Multi-scale features for matching: dilated convolution, spatial pyramid pooling
- 3D CNN for processing cost volume



Stereo matching

- PSMnet (CVPR 2018)

Rank	Method	All (%)			Noc (%)			Runtime (s)
		D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	
1	PSMNet (ours)	**1.86**	4.62	**2.32**	**1.71**	4.31	**2.14**	0.41
3	iResNet-i2e2 [14]	2.14	3.45	2.36	1.94	3.20	2.15	0.22
6	iResNet [14]	2.35	**3.23**	2.50	2.15	**2.55**	2.22	**0.12**
8	CRL [21]	2.48	3.59	2.67	2.32	3.12	2.45	0.47
11	GC-Net [13]	2.21	6.16	2.87	2.02	5.58	2.61	0.90



(a) PSMNet (b) GC-Net (c) MC-CNN

Style Transfer

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

Style image Content image Output

Exploiting Spatial Relation for Reducing Distortion in Style Transfer [Chang and Chen, WACV 2021]

Style Transfer