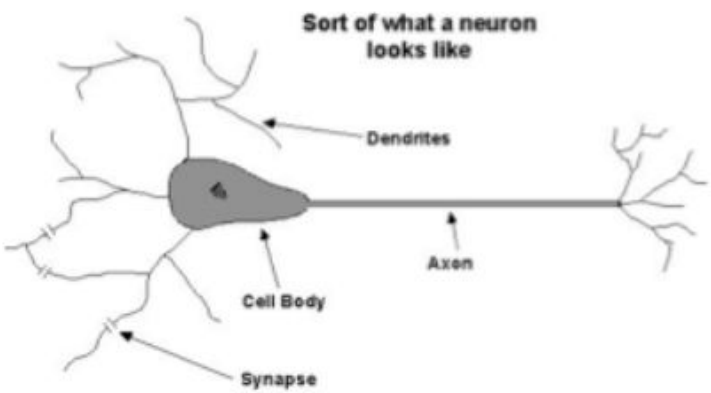
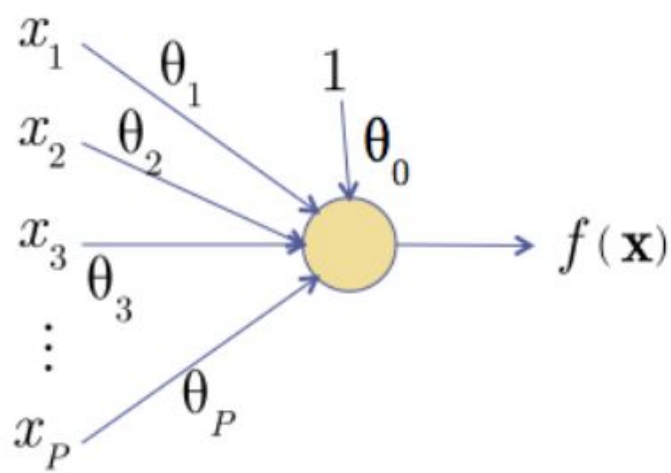
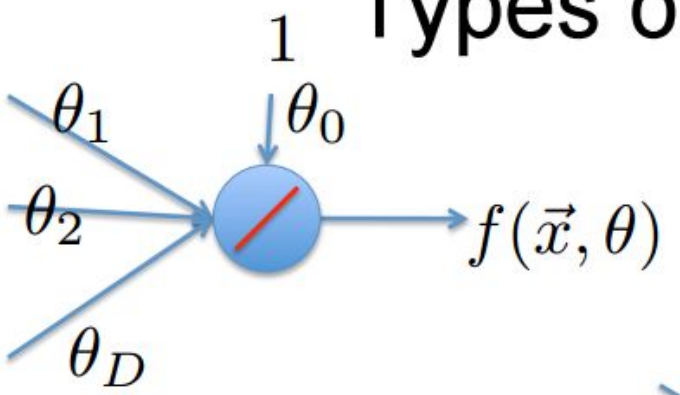
Recall: The Neuron Metaphor

- Neurons
  - accept information from multiple inputs,
  - transmit information to other neurons.
- Multiply inputs by weights along edges
- Apply some function to the set of inputs at each node

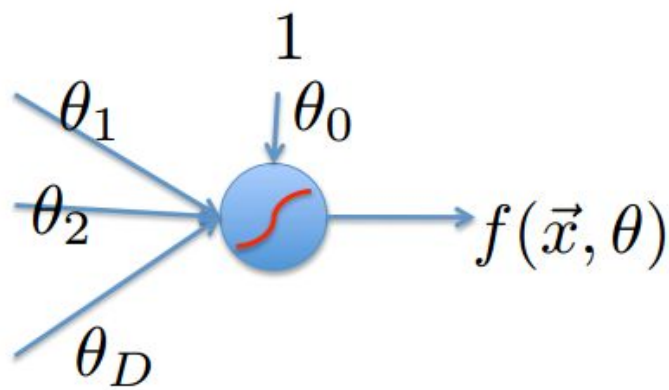




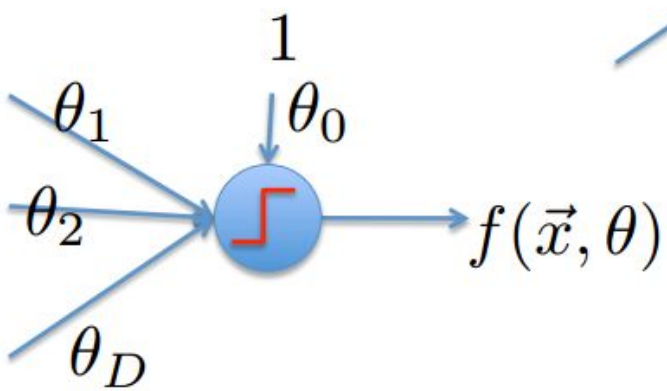
Types of Neurons



Linear Neuron



Logistic Neuron



Perceptron

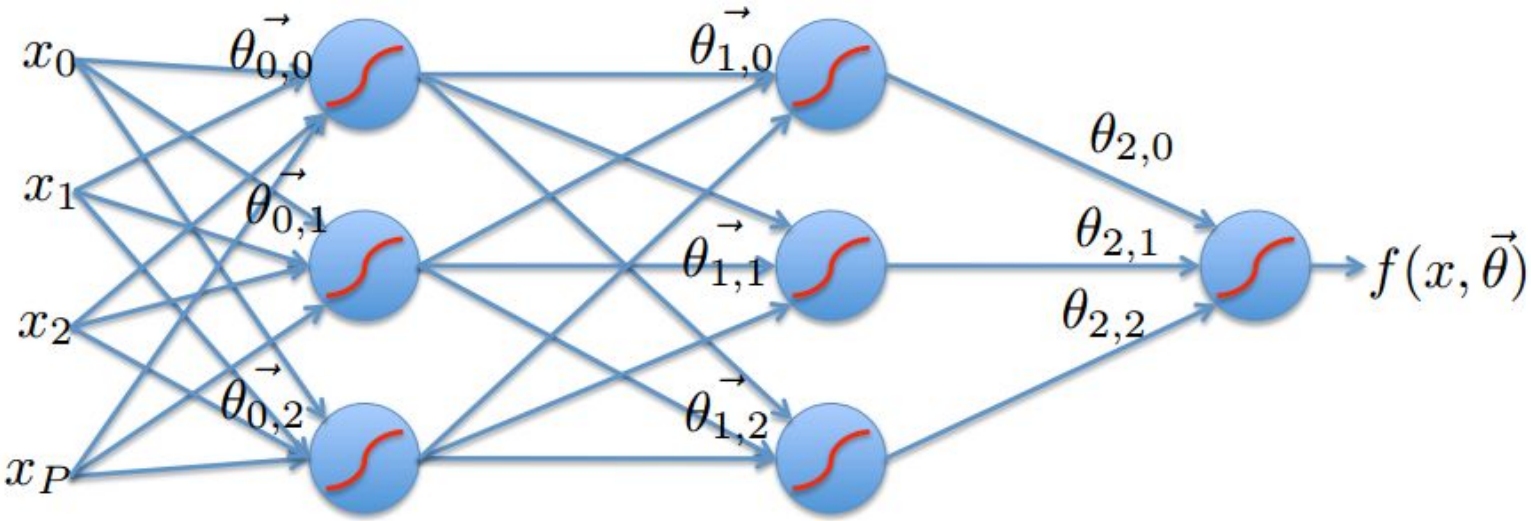
Potentially more. Require a convex loss function for gradient descent training.

Limitation

- A single “neuron” is still a linear decision boundary
- What to do?
- Idea: Stack a bunch of them together!

Multilayer Networks

- Cascade Neurons together
- The output from one layer is the input to the next
- Each Layer has its own sets of weights



A quick note

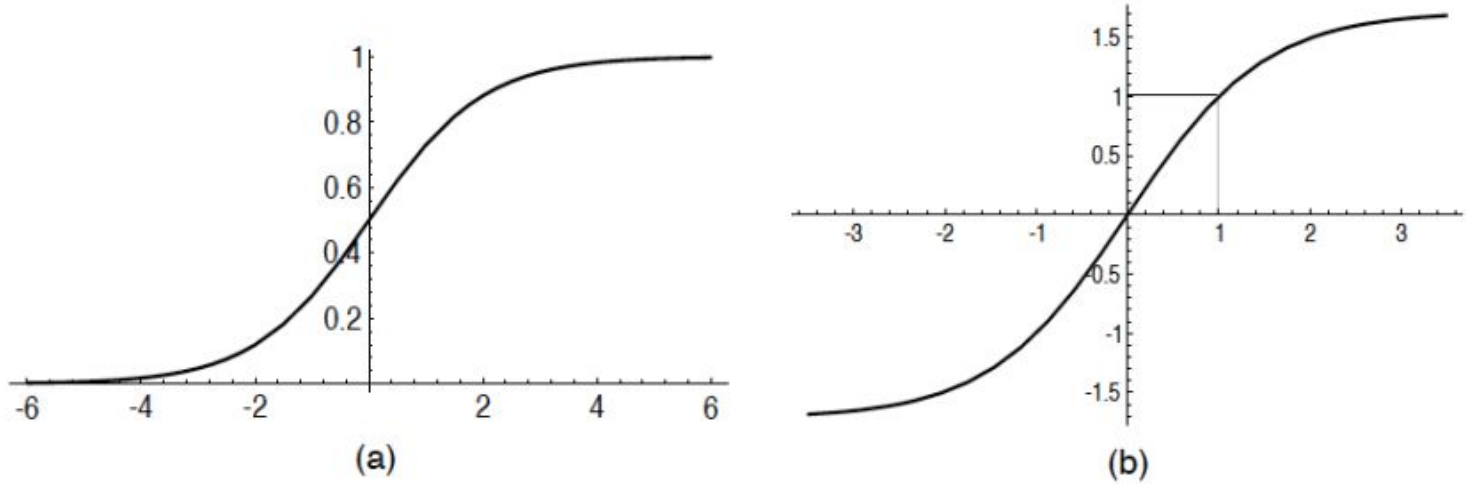
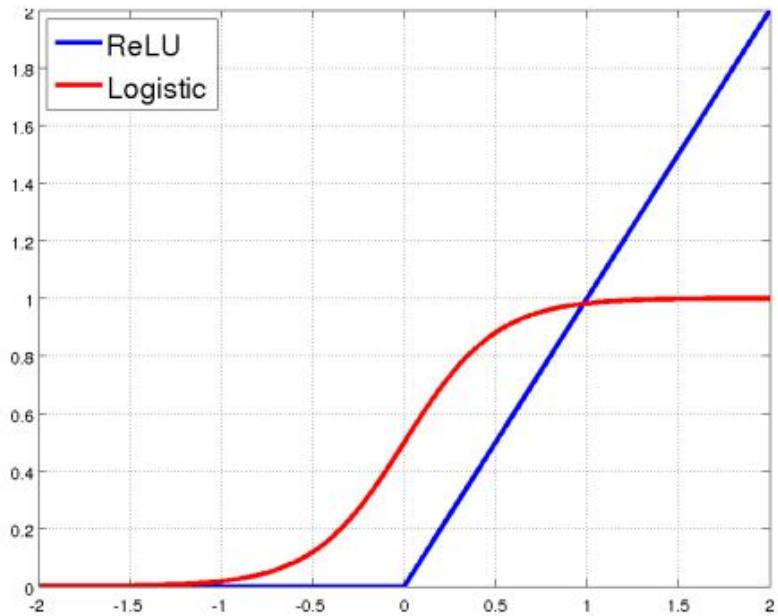
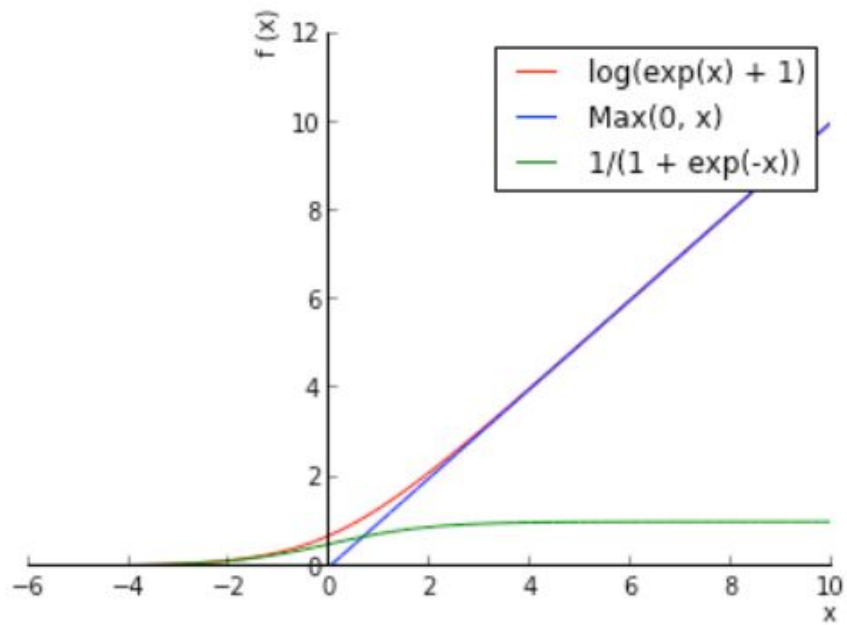


Fig. 4. (a) Not recommended: the standard logistic function, $f(x) = 1/(1 + e^{-x})$. (b) Hyperbolic tangent, $f(x) = 1.7159 \ \tanh\left(\frac{2}{3}x\right)$.

Image Credit: LeCun et al. '98

Rectified Linear Units (ReLU)





Supervised Learning

$\{(\boldsymbol{x}^i, y^i), i = 1 \ldots P\}$ training dataset

$\boldsymbol{x}^i$ i-th input training example

y^i i-th target label

P number of training examples

Goal: predict the target label of unseen inputs.

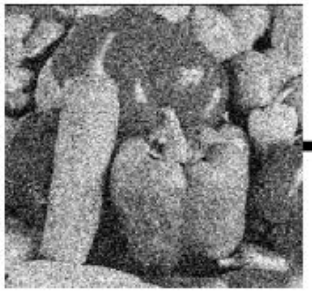
Supervised Learning: Examples

Classification

"dog"

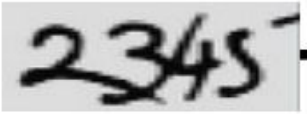
classification

Denoising



regression

OCR



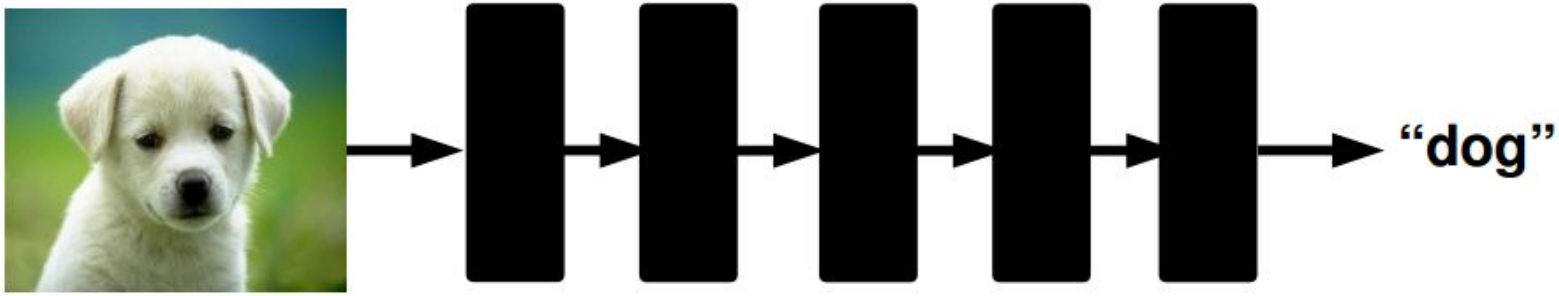
"2 3 4 5"

structured prediction

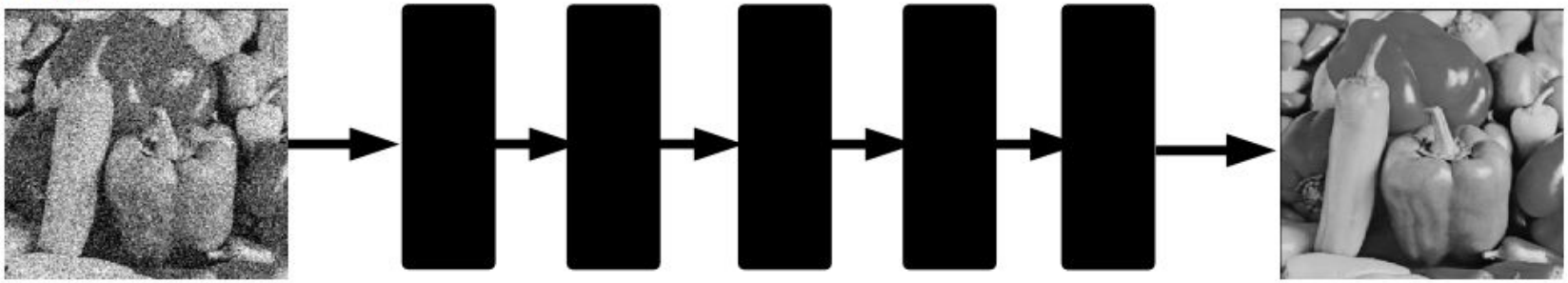
Ranzato

Supervised Deep Learning

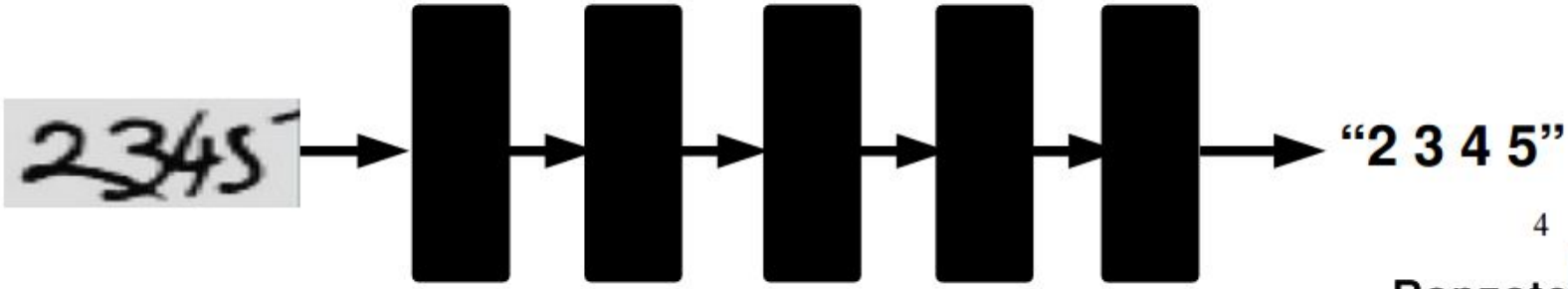
Classification



Denoising

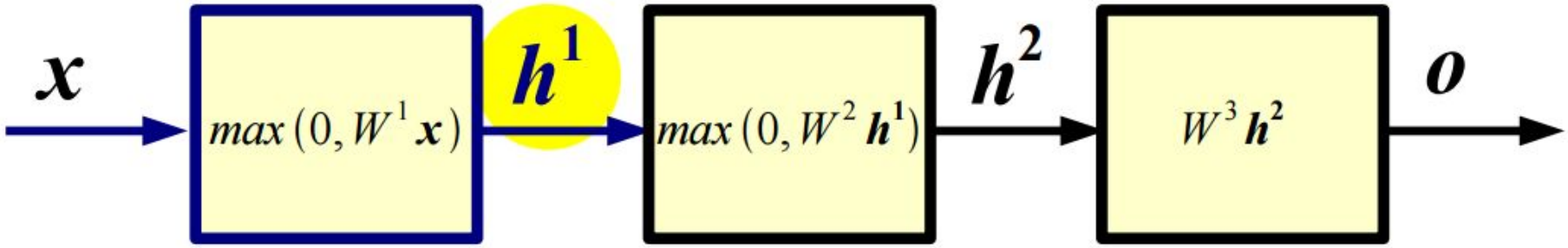


OCR



Ranzato

Forward Propagation



$\boldsymbol{x} \in R^D \quad W^1 \in R^{N_1 \times D} \quad \boldsymbol{b}^1 \in R^{N_1} \quad \boldsymbol{h}^1 \in R^{N_1}$

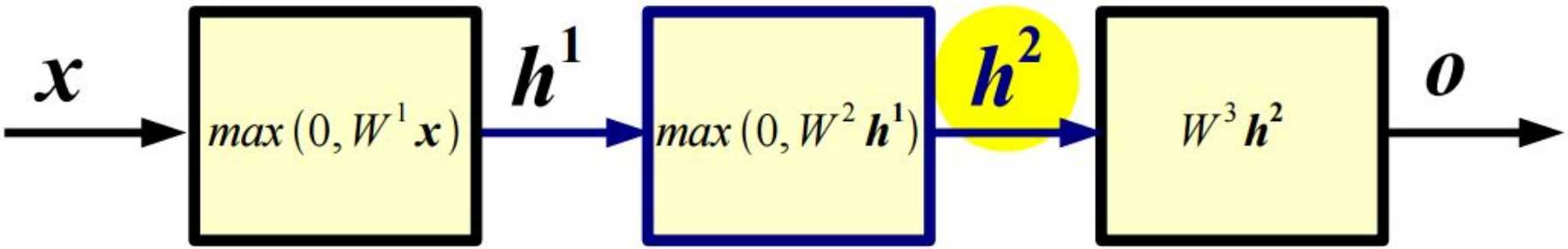
$$\boldsymbol{h}^1 = max(0, W^1 \boldsymbol{x} + \boldsymbol{b}^1)$$

W^1 1-st layer weight matrix or weights

$\boldsymbol{b}^1$ 1-st layer biases

The non-linearity $u = max(0, v)$ is called **ReLU** in the DL literature. Each output hidden unit takes as input all the units at the previous layer: each such layer is called "**fully connected**".

Forward Propagation



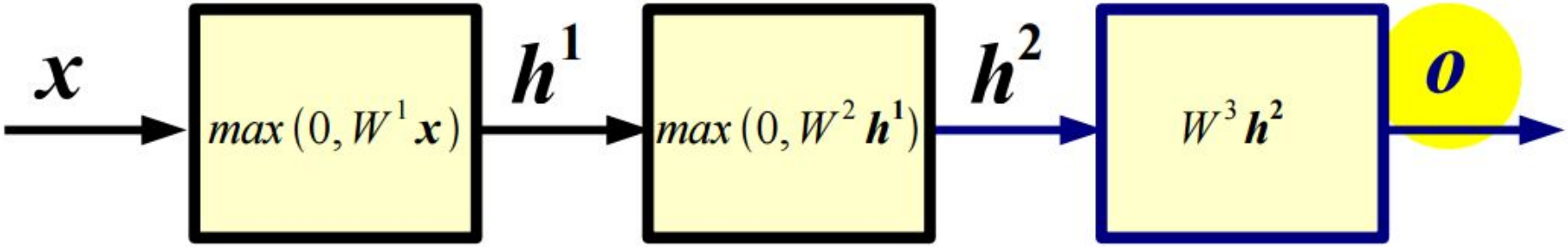
$\boldsymbol{h}^1 \in R^{N_1}$ $W^2 \in R^{N_2 \times N_1}$ $\boldsymbol{b}^2 \in R^{N_2}$ $\boldsymbol{h}^2 \in R^{N_2}$

$$\boldsymbol{h}^2 = max(0, W^2 \boldsymbol{h}^1 + \boldsymbol{b}^2)$$

W^2 2-nd layer weight matrix or weights

$\boldsymbol{b}^2$ 2-nd layer biases

Forward Propagation



$\boldsymbol{h}^2 \in R^{N_2}$ $W^3 \in R^{N_3 \times N_2}$ $\boldsymbol{b}^3 \in R^{N_3}$ $\boldsymbol{o} \in R^{N_3}$

$$\boldsymbol{o} = max(0, W^3 \boldsymbol{h}^2 + \boldsymbol{b}^3)$$

W^3 3-rd layer weight matrix or weights

$\boldsymbol{b}^3$ 3-rd layer biases

Training

Learning consists of minimizing the loss (plus some regularization term) w.r.t. parameters over the whole training set.

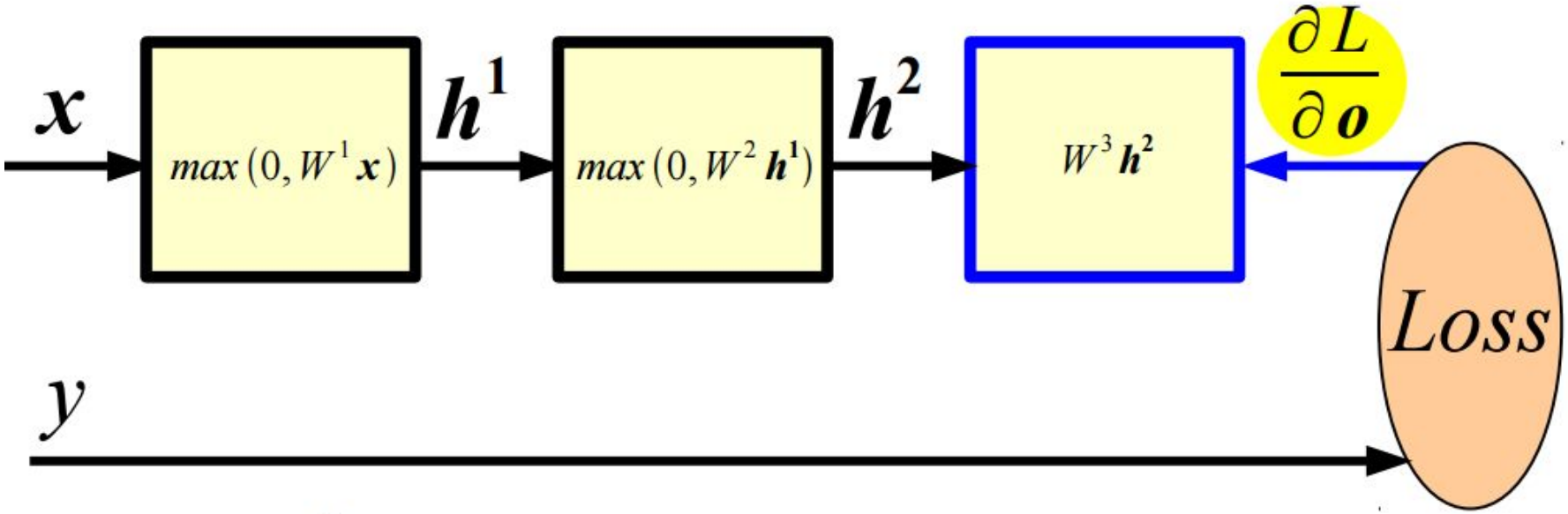
$$\boldsymbol{\theta}^* = arg\,min_{\boldsymbol{\theta}} \sum_{n=1}^{P} L(\boldsymbol{x}^n, y^n; \boldsymbol{\theta})$$

Question: How to minimize a complicated function of the parameters?

Answer: Chain rule, a.k.a. **Backpropagation**! That is the procedure to compute gradients of the loss w.r.t. parameters in a multi-layer neural network.

Rumelhart et al. "Learning internal representations by back-propagating.." Nature 1986

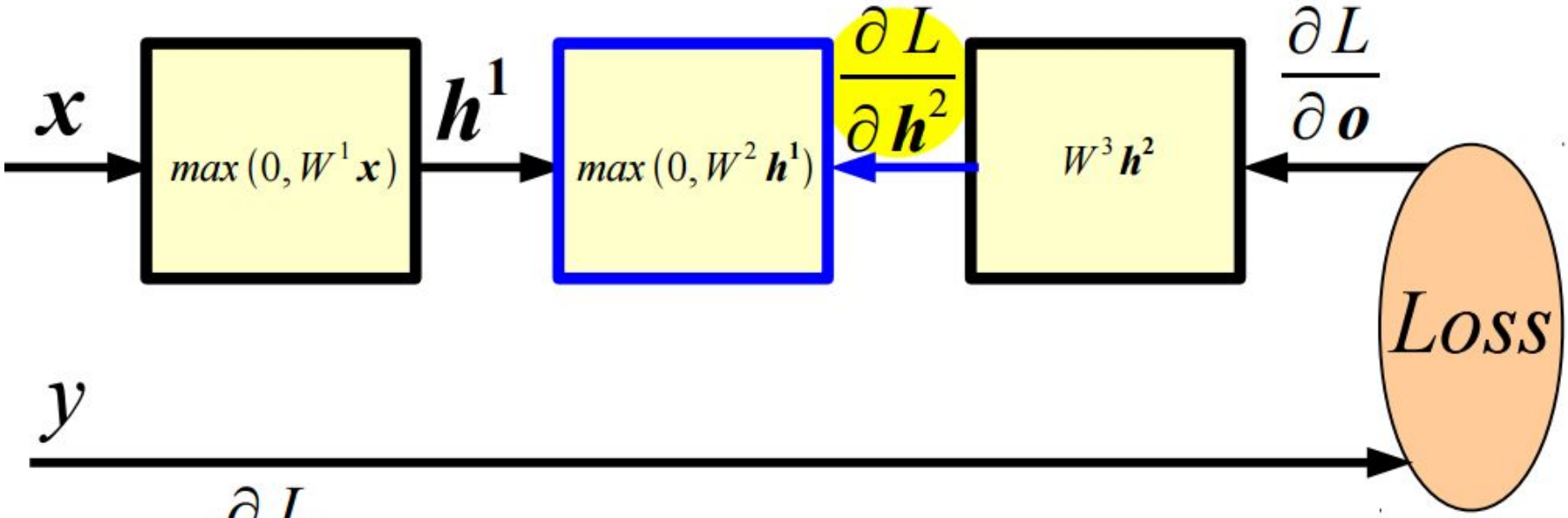
Backward Propagation



Given $\partial L / \partial \boldsymbol{o}$ and assuming we can easily compute the Jacobian of each module, we have:

$$\frac{\partial L}{\partial W^3} = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial W^3} \qquad \frac{\partial L}{\partial \boldsymbol{h}^2} = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial \boldsymbol{h}^2}$$

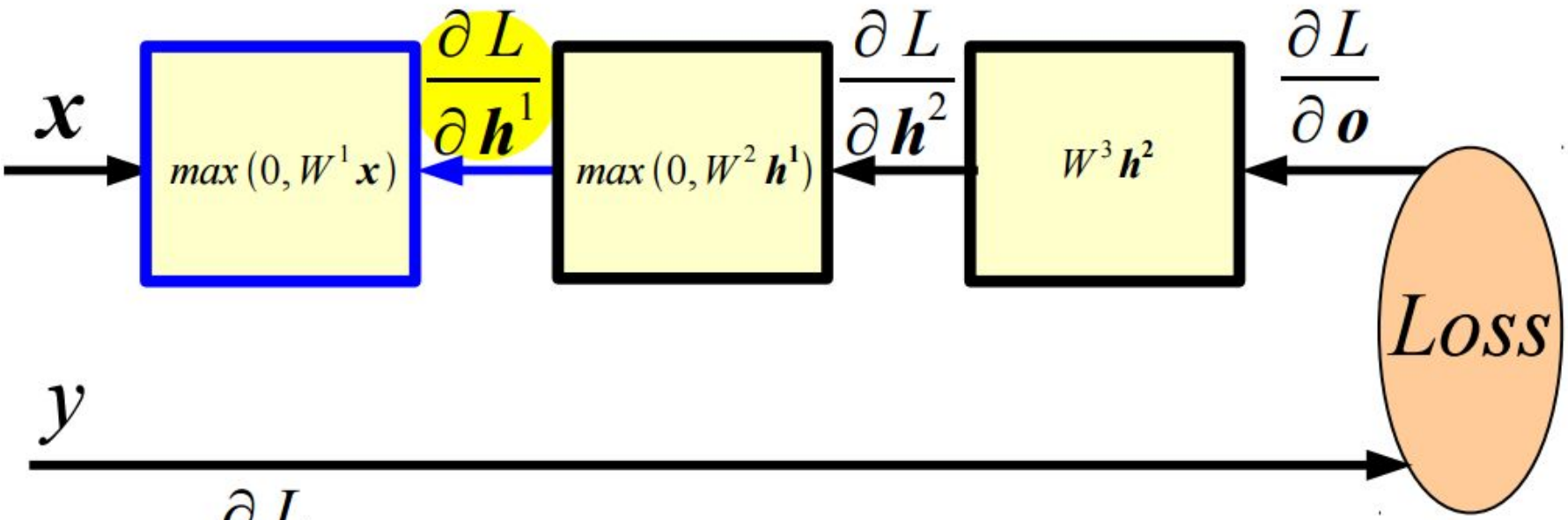
Backward Propagation



Given $\frac{\partial L}{\partial h^2}$ we can compute now:

$$\frac{\partial L}{\partial W^2} = \frac{\partial L}{\partial h^2} \frac{\partial h^2}{\partial W^2} \qquad \frac{\partial L}{\partial h^1} = \frac{\partial L}{\partial h^2} \frac{\partial h^2}{\partial h^1}$$

Backward Propagation

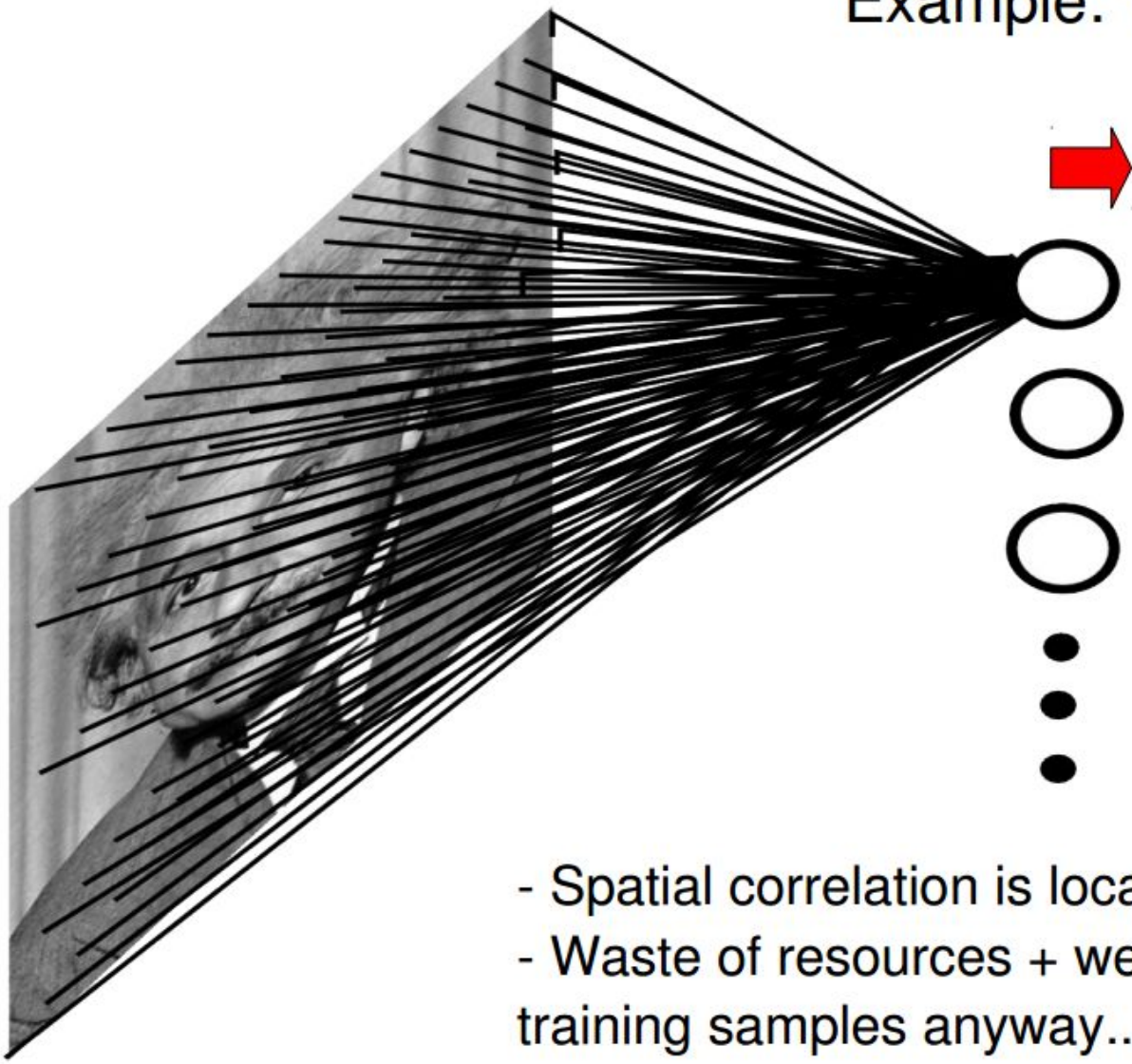


Given $\frac{\partial L}{\partial h^1}$ we can compute now:

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial h^1} \frac{\partial h^1}{\partial W^1}$$

Convolutional Neural Networks

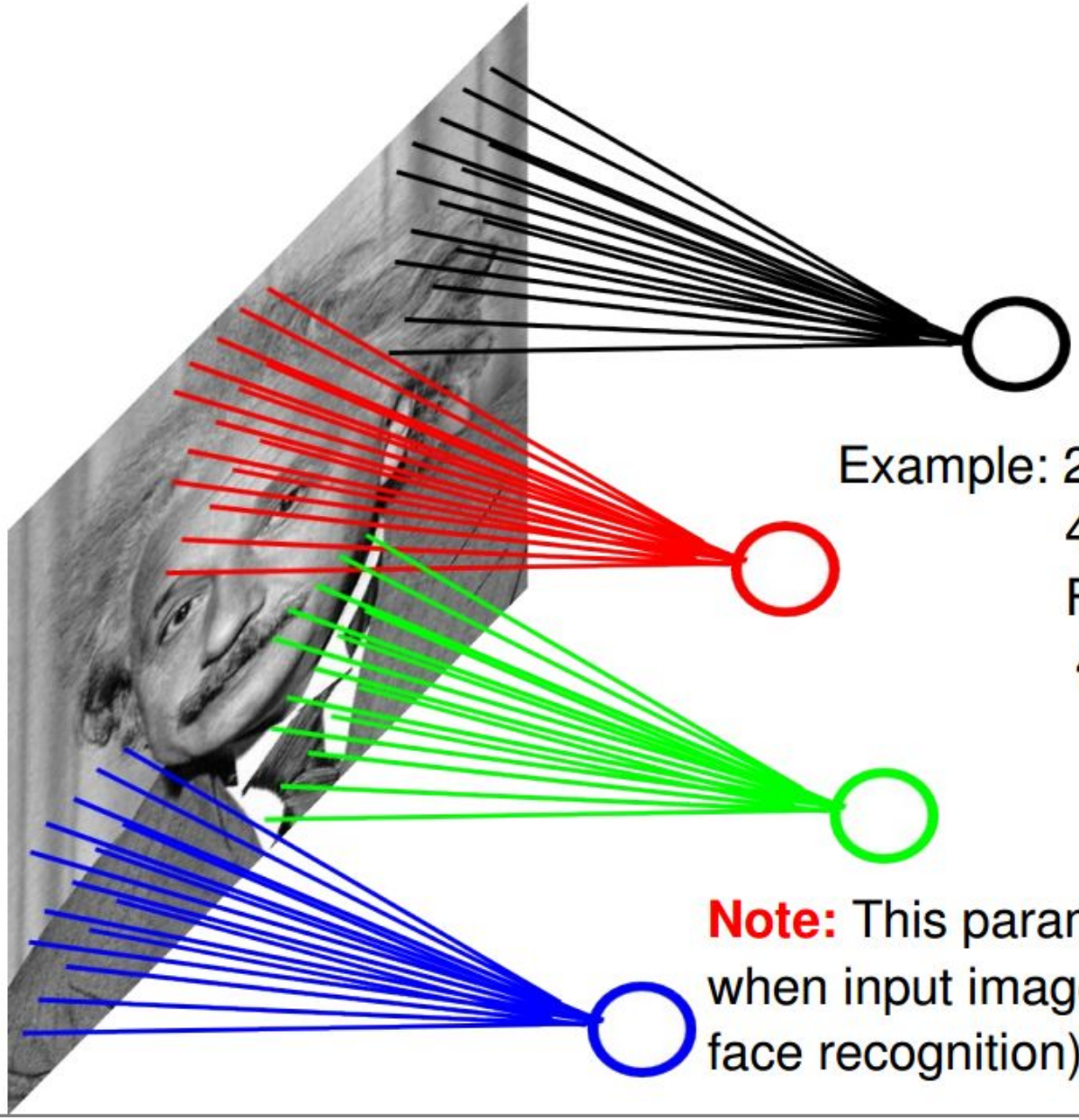
Fully Connected Layer



Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

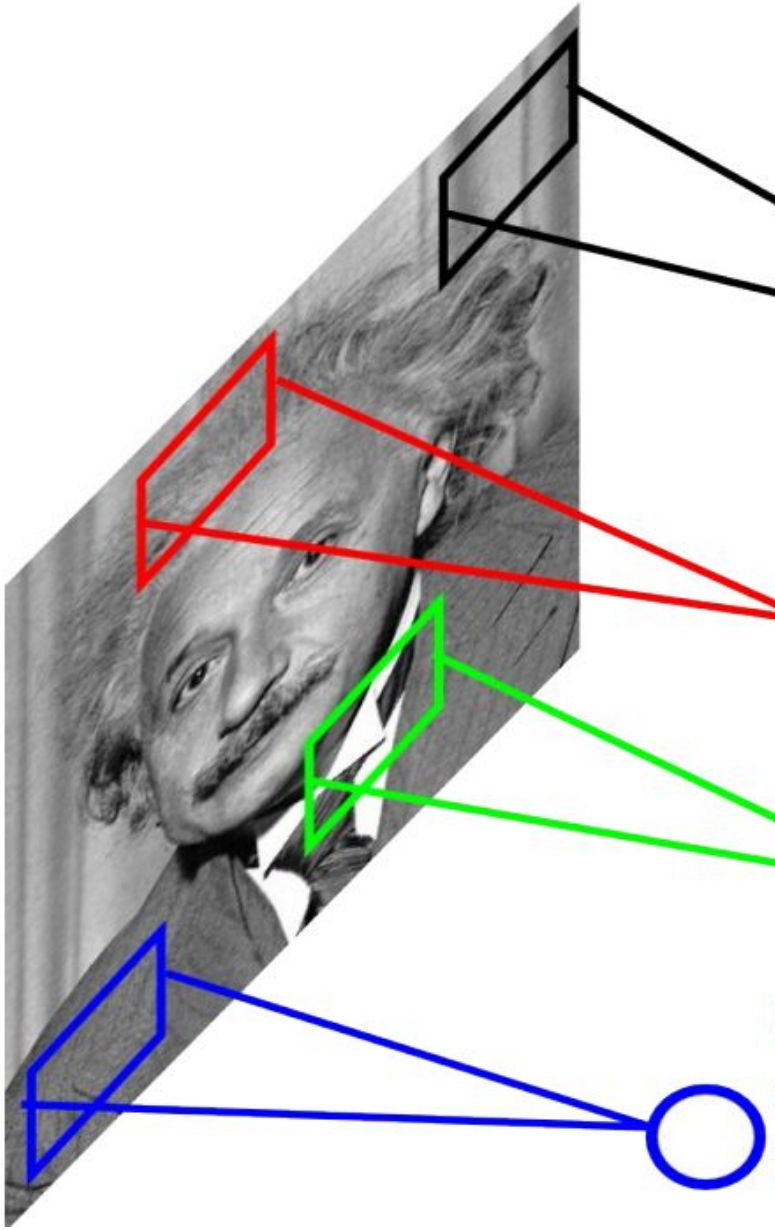
Locally Connected Layer



Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Locally Connected Layer



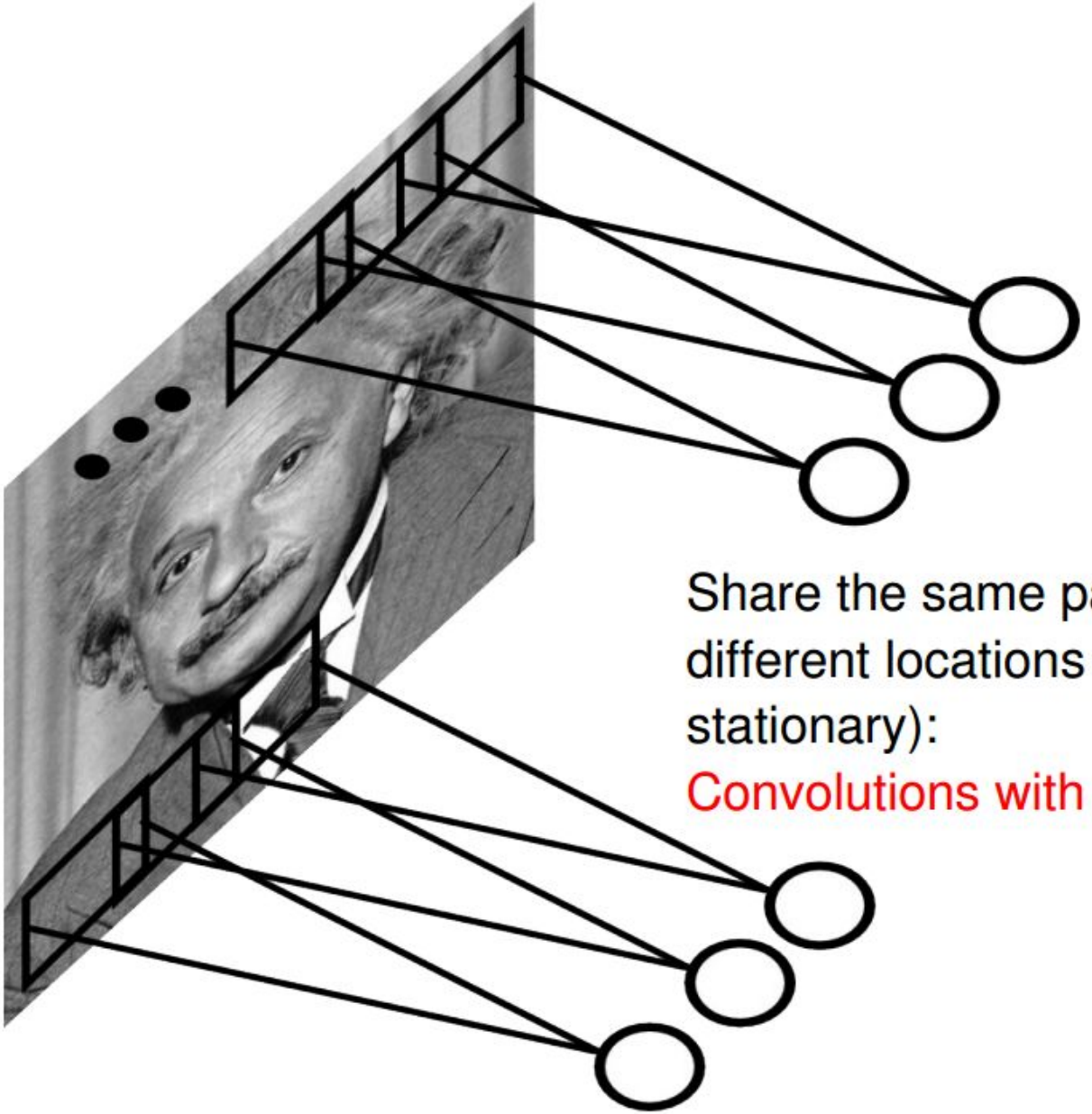
STATIONARITY? Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Ranzato

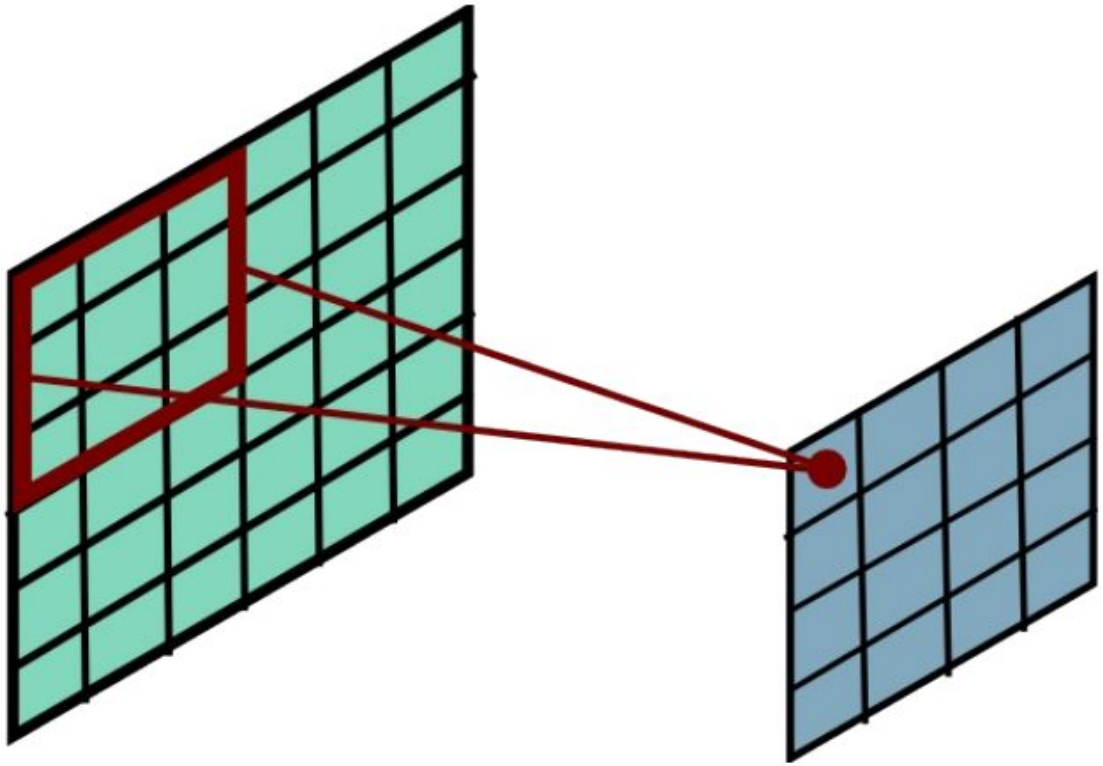
Convolutional Layer



Share the same parameters across different locations (assuming input is stationary):

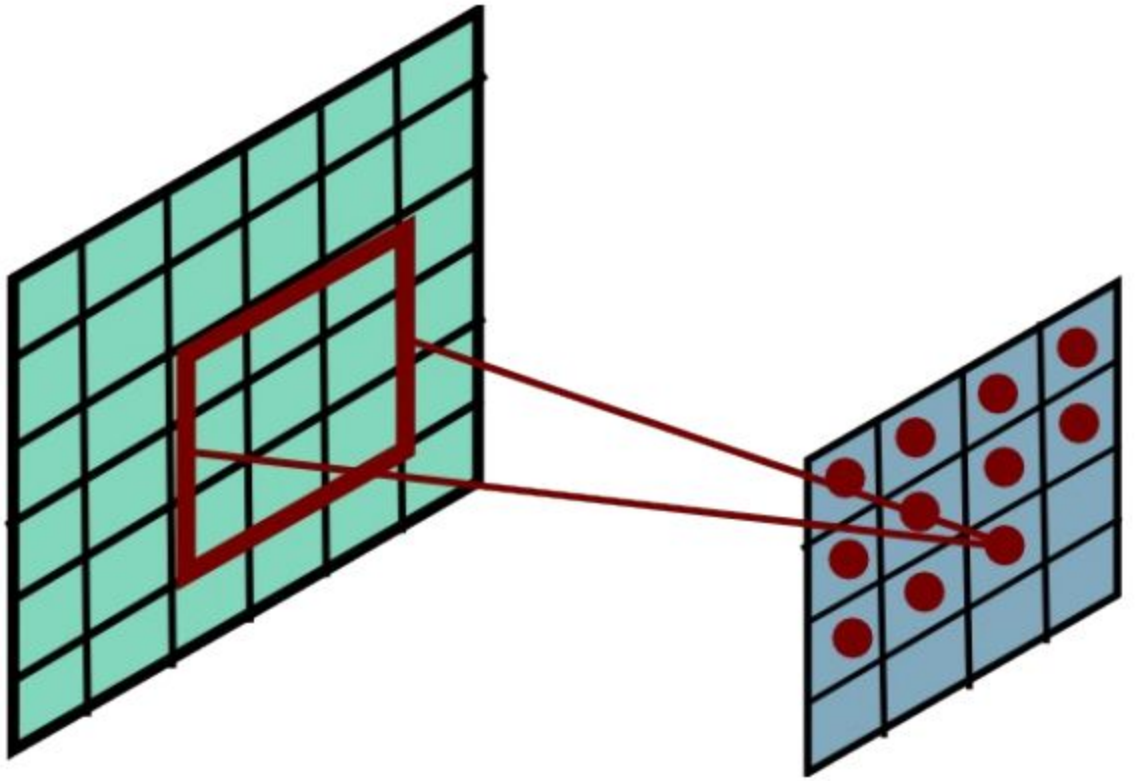
Convolutions with learned kernels

Convolutional Layer



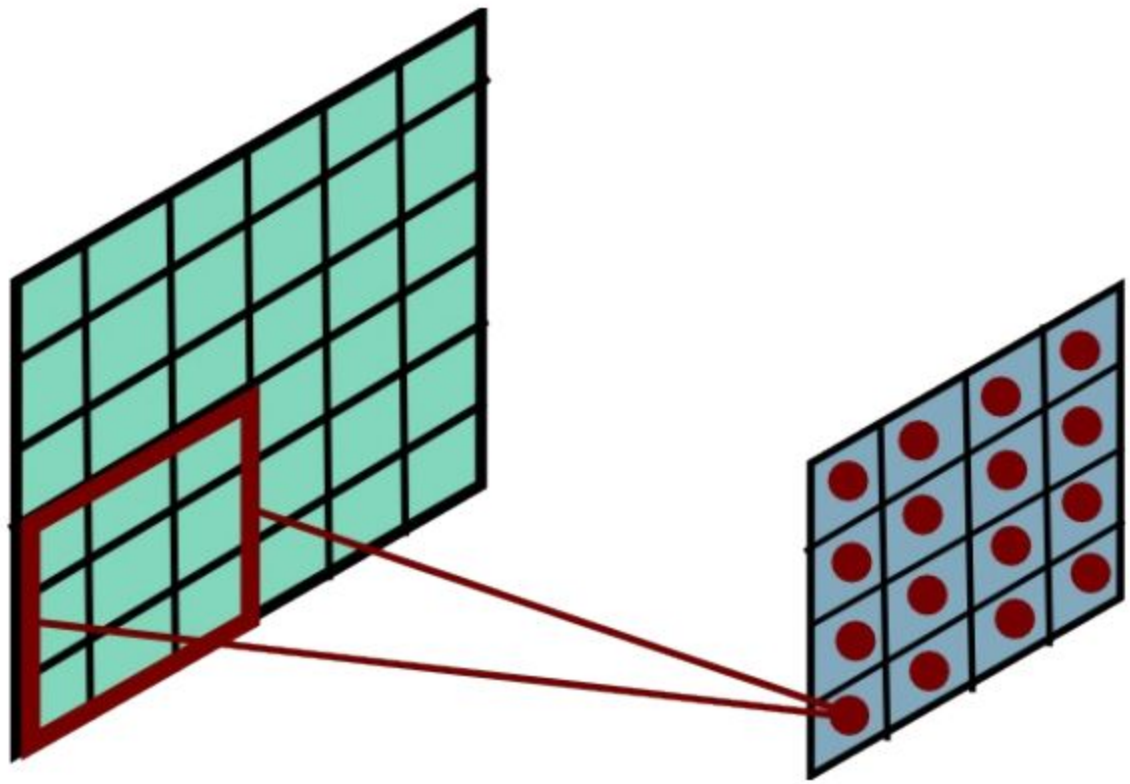
Ranzato

Convolutional Layer



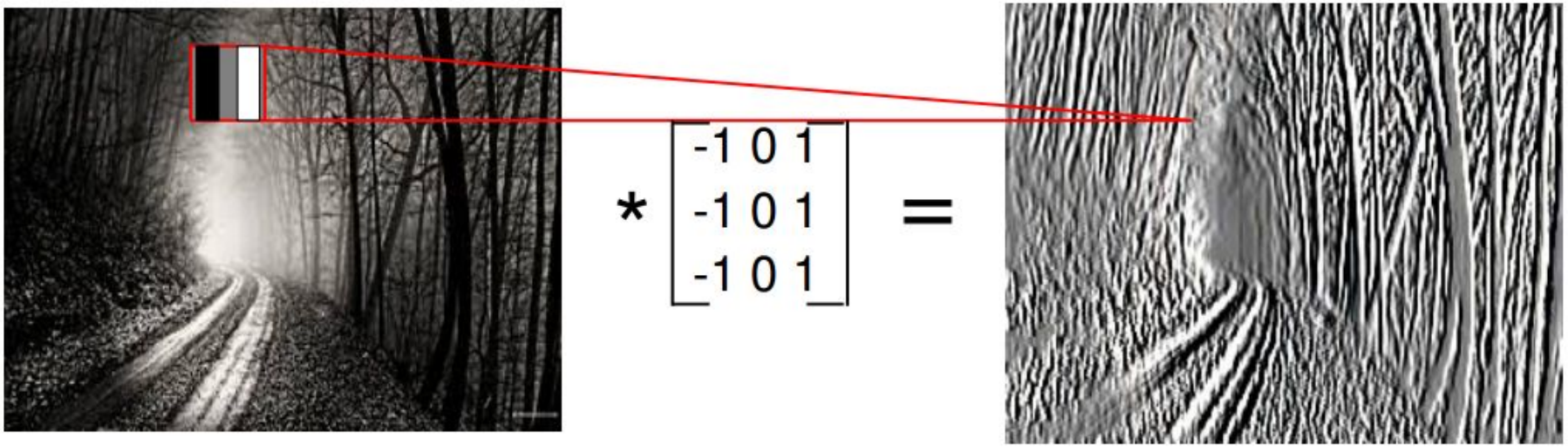
Ranzato

Convolutional Layer



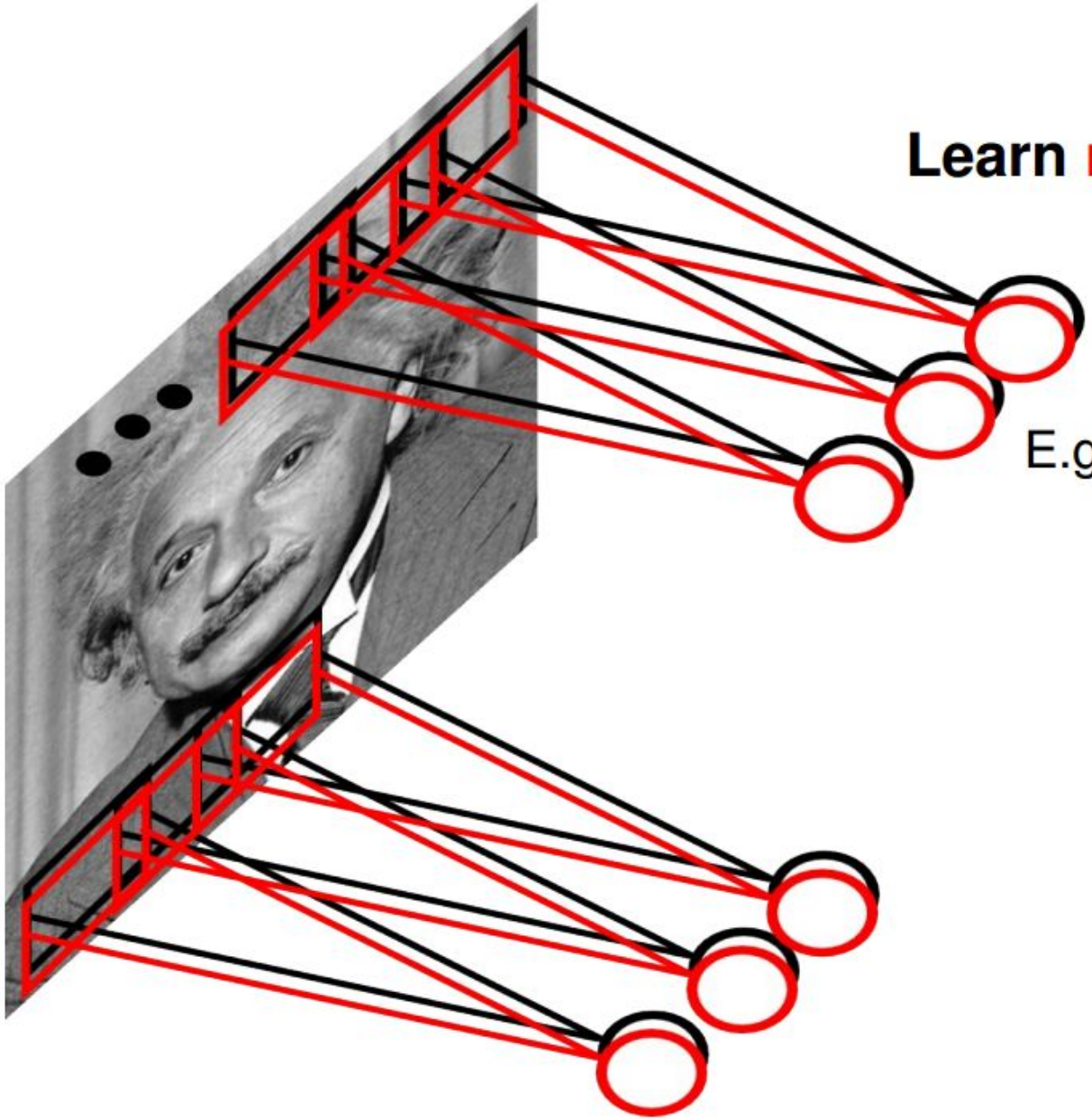
Ranzato

Convolutional Layer



Ranzato

Convolutional Layer



Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

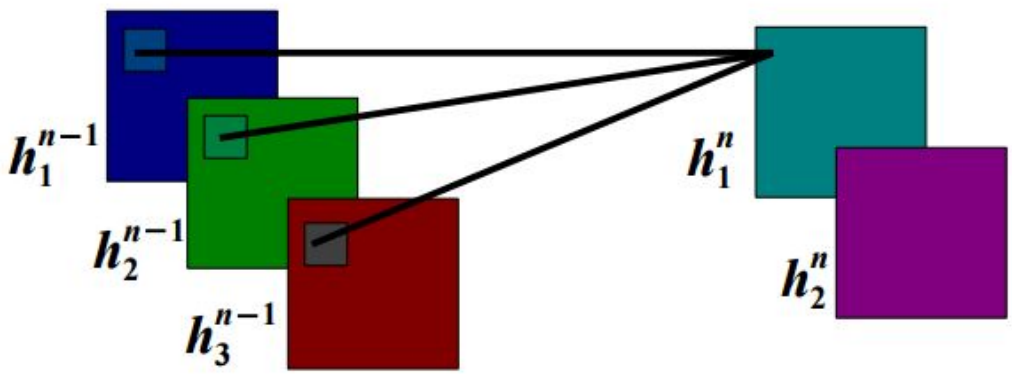
Convolutional Layer

$$h_j^n = max\left(0, \sum_{k=1}^{K} h_k^{n-1} * w_{kj}^n\right)$$

output feature map

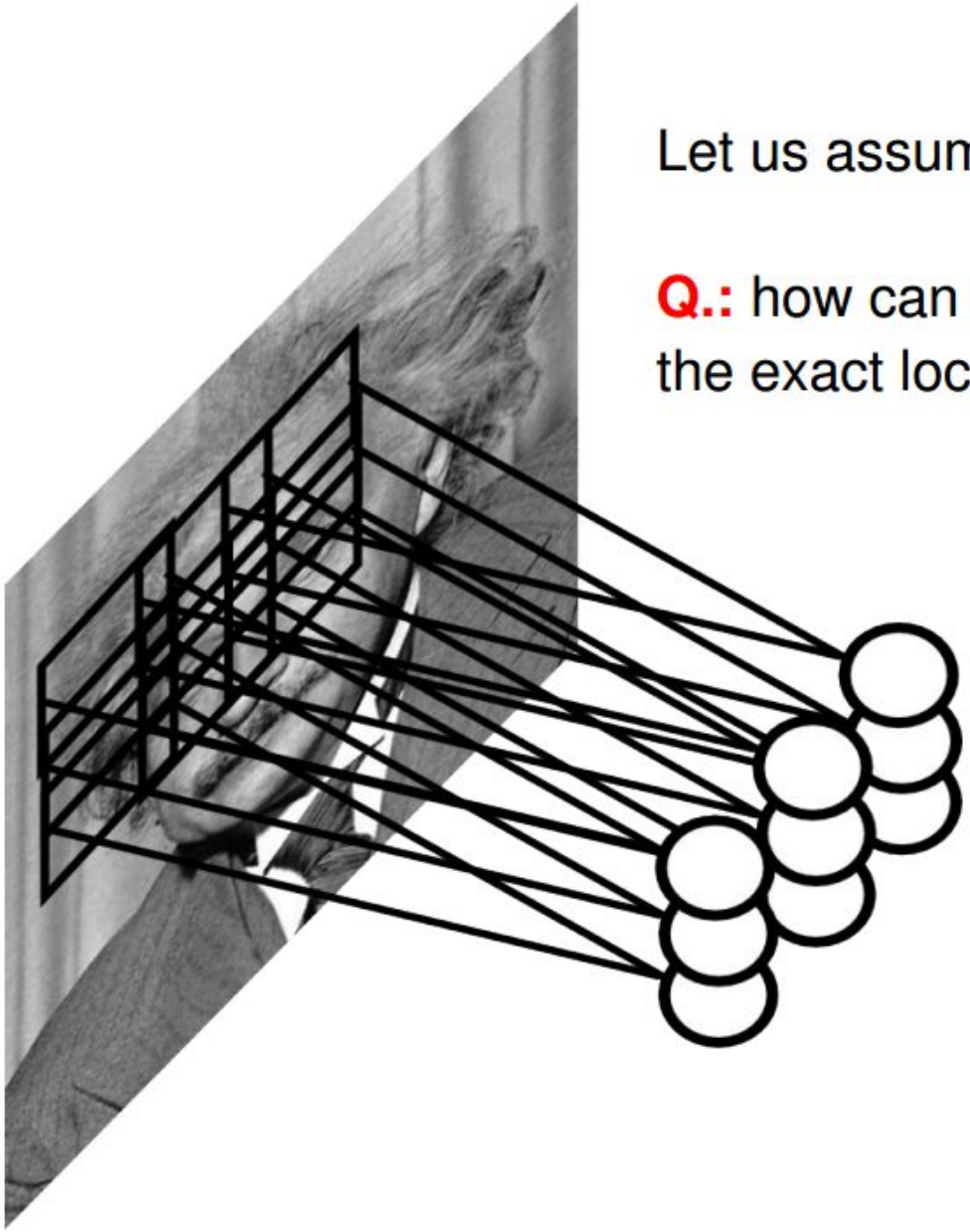
input feature map

kernel



Ranzato

Pooling Layer

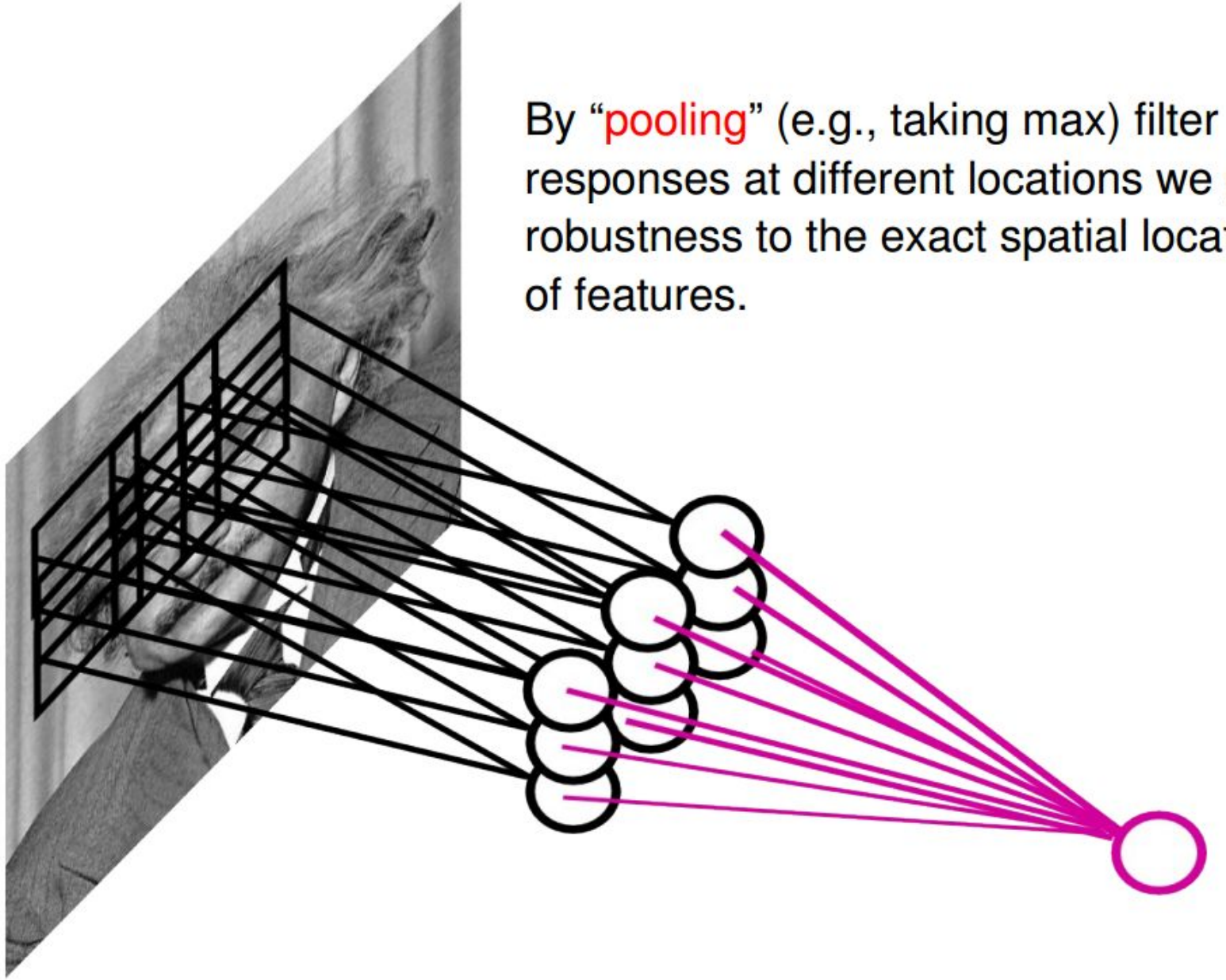


Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

Pooling Layer

By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.



Pooling Layer: Examples

Max-pooling:

$$h_j^n(x, y) = max_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})$$

Average-pooling:

$$h_j^n(x, y) = 1/K \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})$$

L2-pooling:

$$h_j^n(x, y) = \sqrt{\sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})^2}$$

L2-pooling over features:

$$h_j^n(x, y) = \sqrt{\sum_{k \in N(j)} h_k^{n-1}(x, y)^2}$$

Convolutional Nets

- Example:
  - http://yann.lecun.com/exdb/lenet/index.html

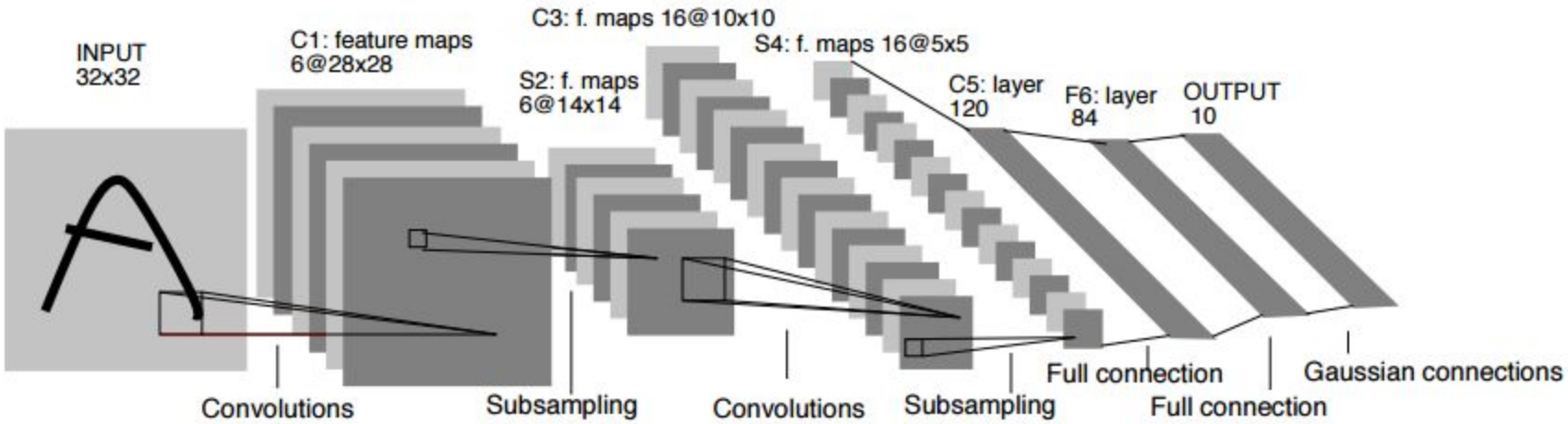
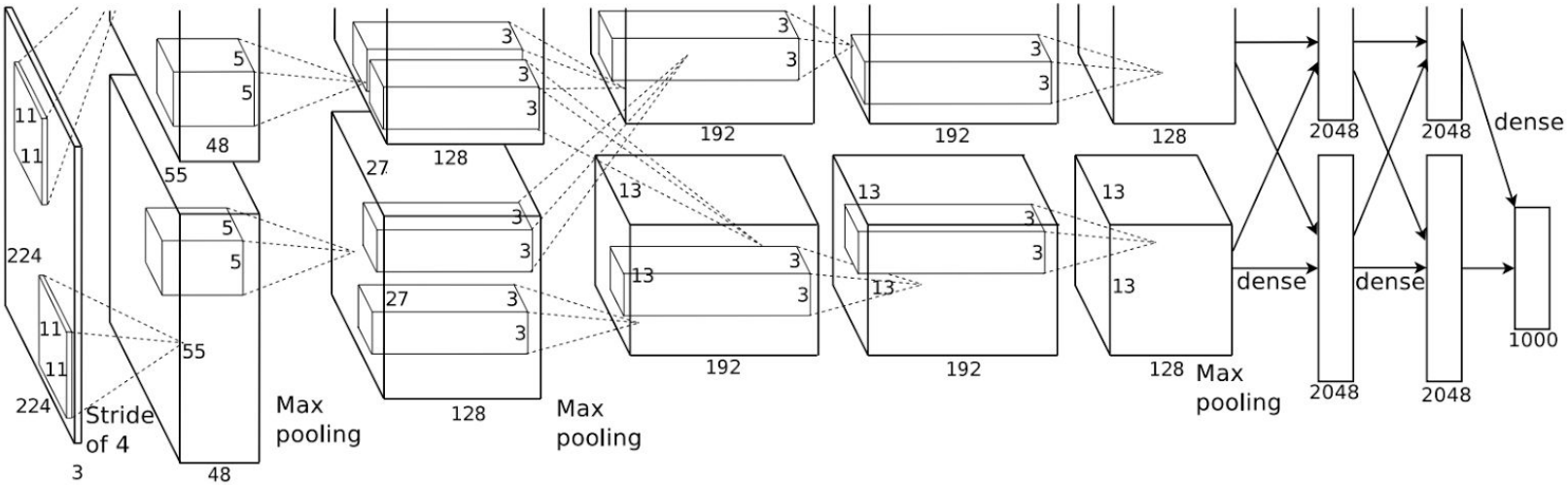


Image Credit: Yann LeCun, Kevin Murphy

Alex Net



Krizhevsky et al. NIPS 2012

Visualizing Learned Filters

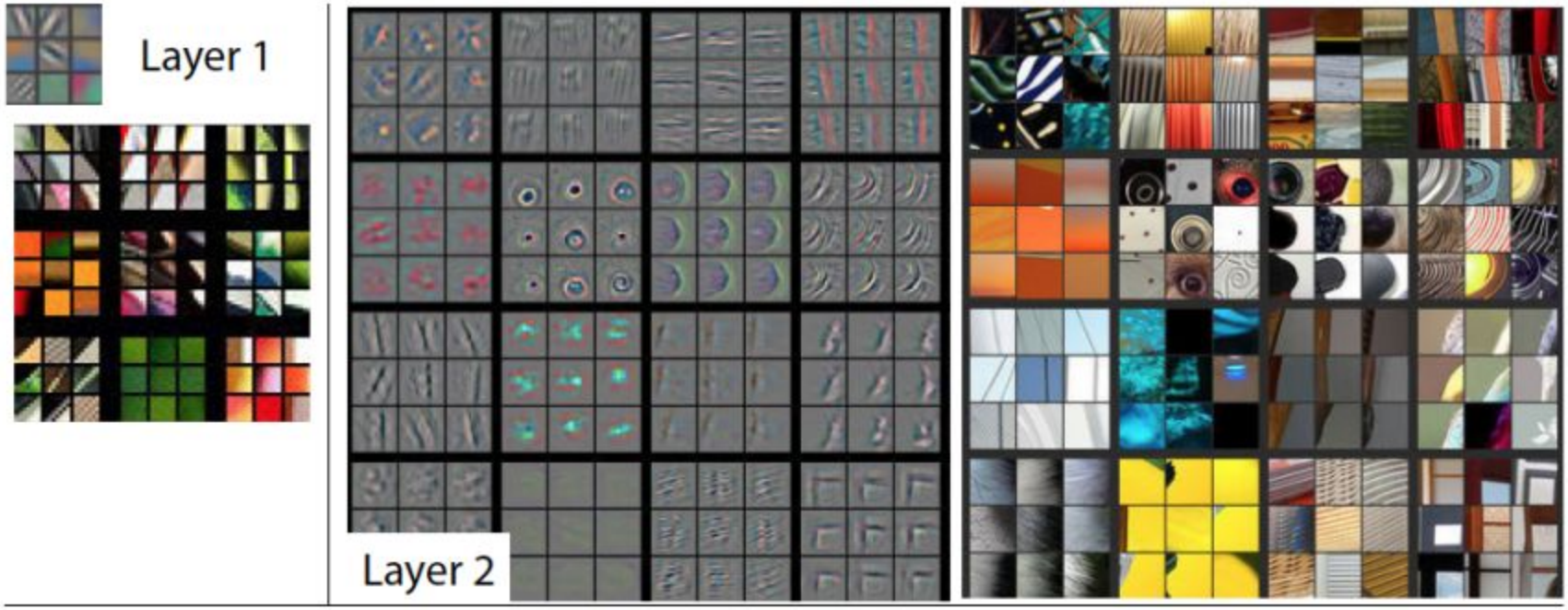


Figure Credit: [Zeiler & Fergus ECCV14]

Visualizing Learned Filters

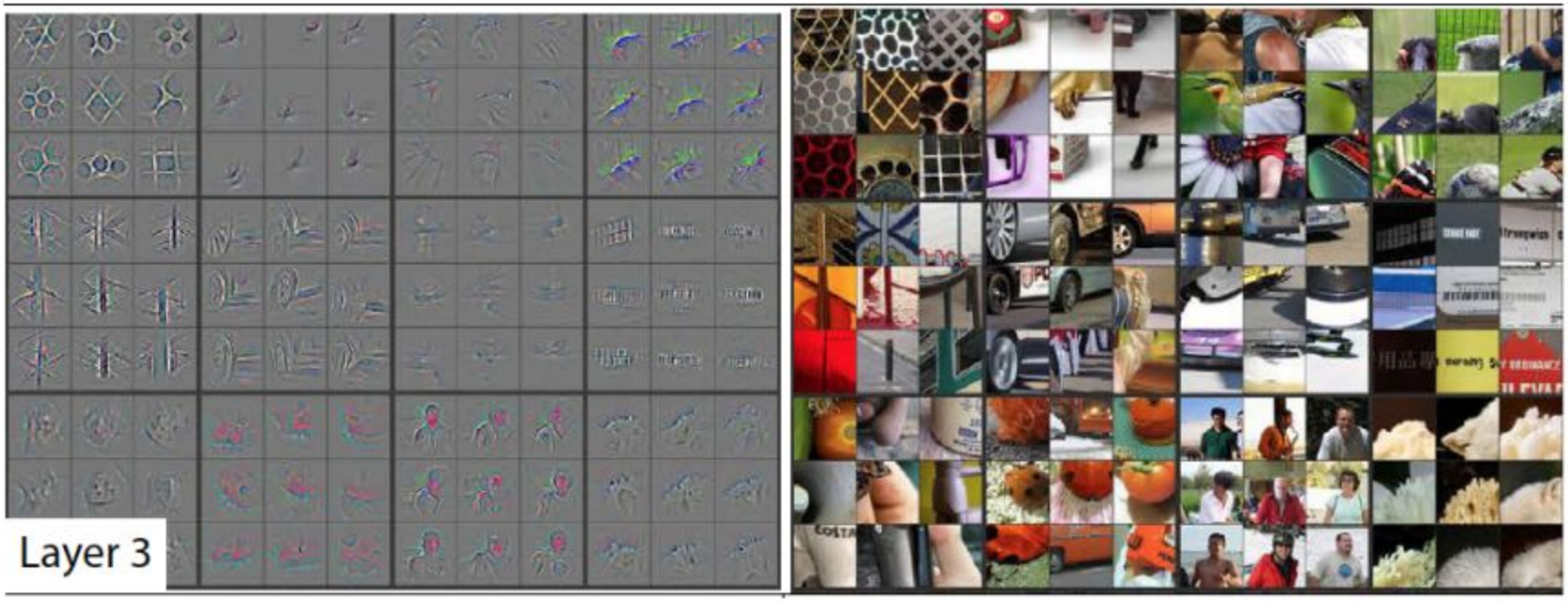


Figure Credit: [Zeiler & Fergus ECCV14]

Visualizing Learned Filters

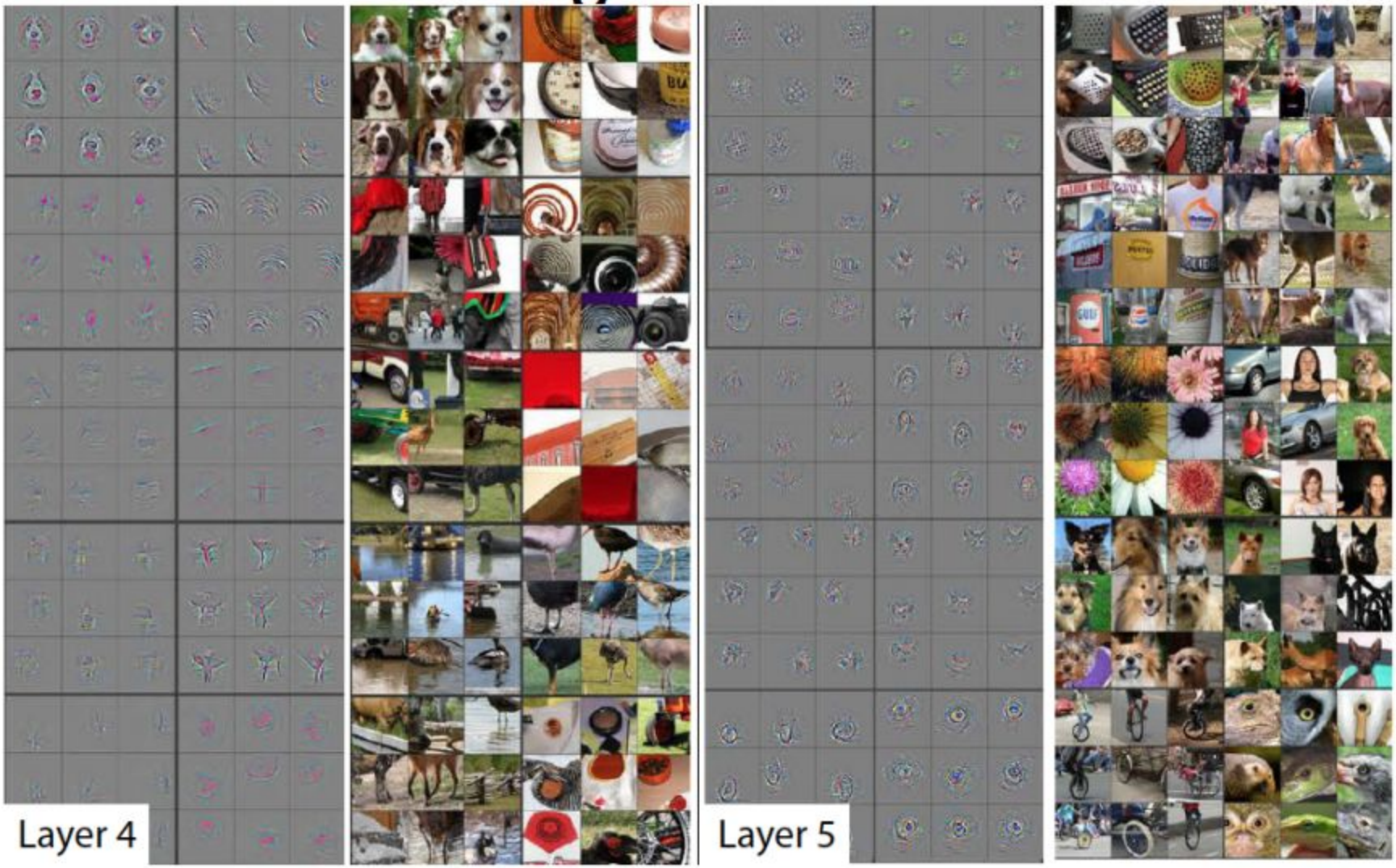
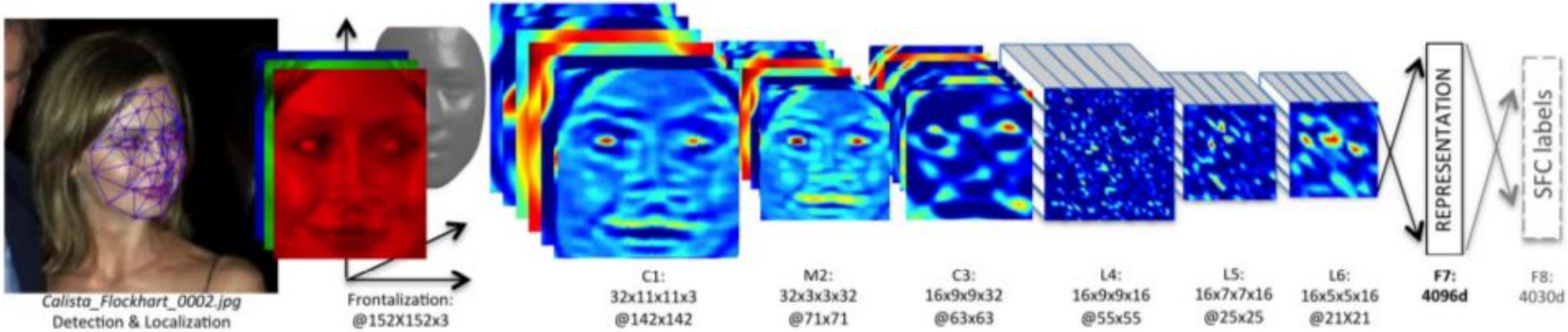


Figure Credit: [Zeiler & Fergus ECCV14]

Industry Deployment

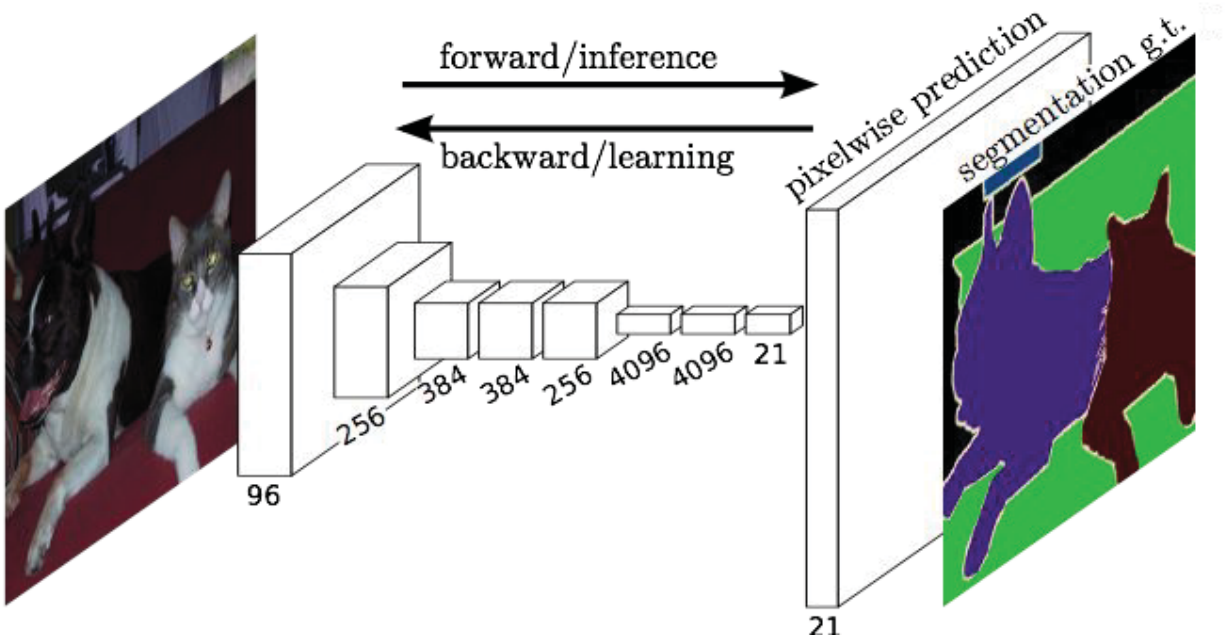
- Used in Facebook, Google, Microsoft
- Image Recognition, Speech Recognition, ….
- Fast at test time



Taigman et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR'14

Slide: R. Fergus

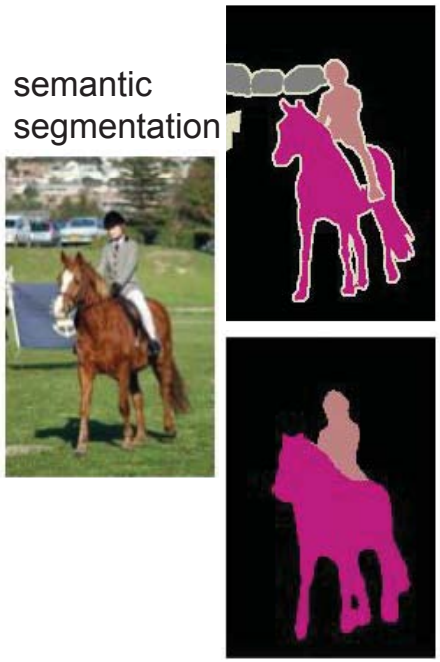
A Fuller Understanding of Fully Convolutional Networks



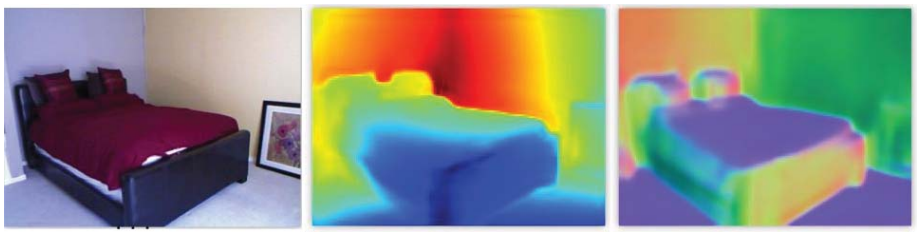
Evan Shelhamer* Jonathan Long* Trevor Darrell
UC Berkeley in CVPR'15, PAMI'16

pixels in, pixels out

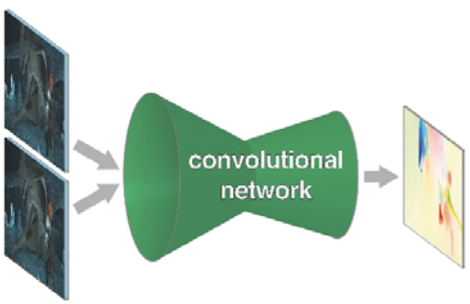
semantic segmentation



monocular depth + normals Eigen & Fergus 2015



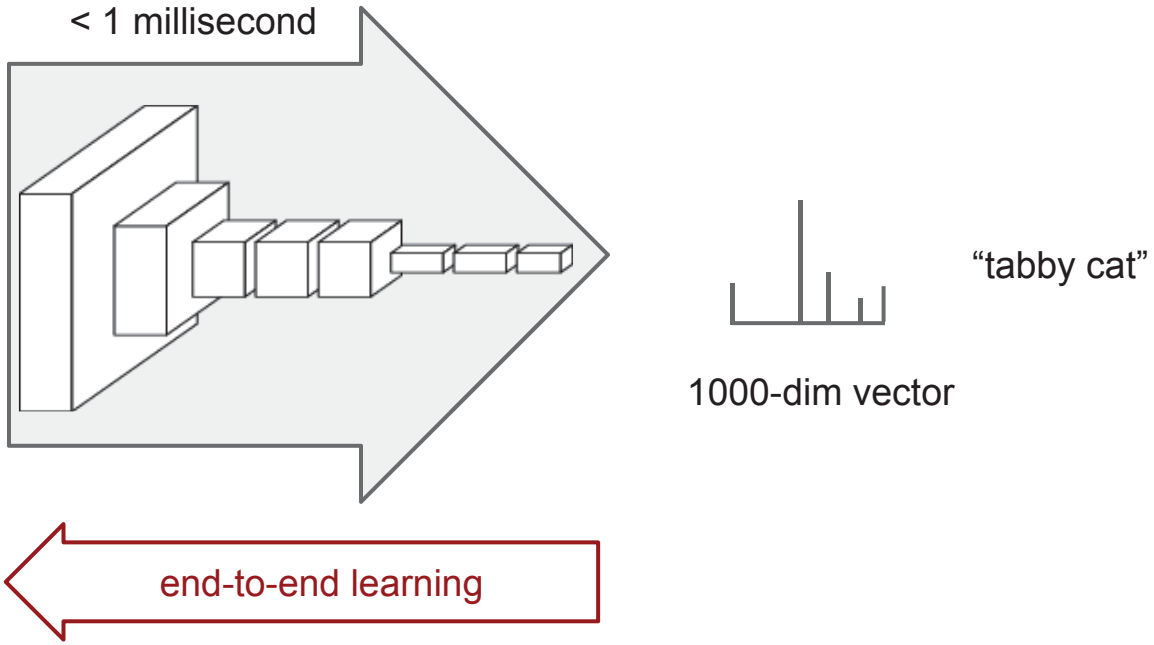
colorization
Zhang et al.2016



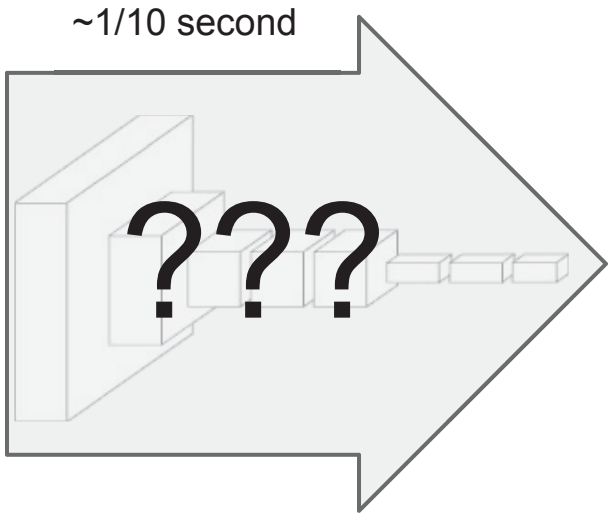
optical flow Fischer et al. 2015

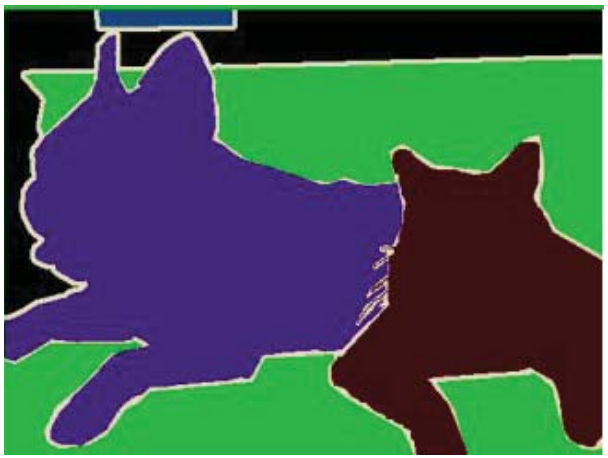
boundary prediction Xie & Tu 2015

convnets perform classification

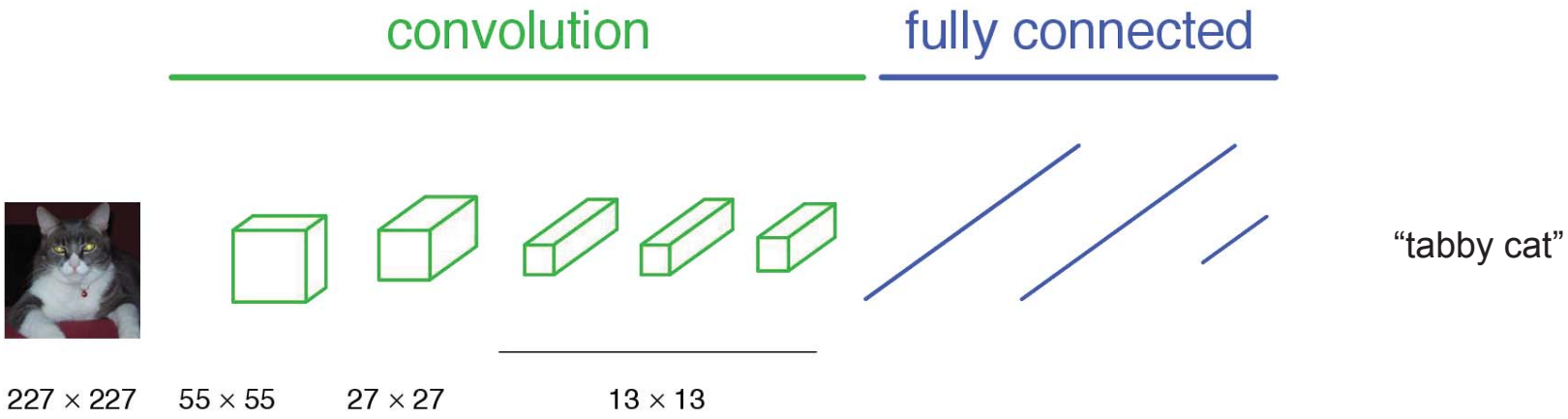


lots of pixels, little time?

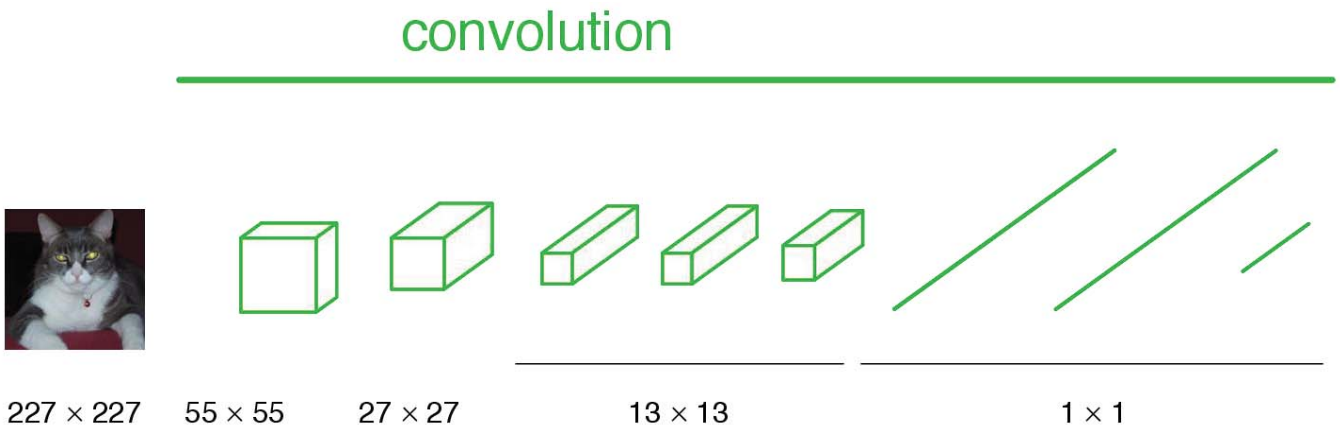




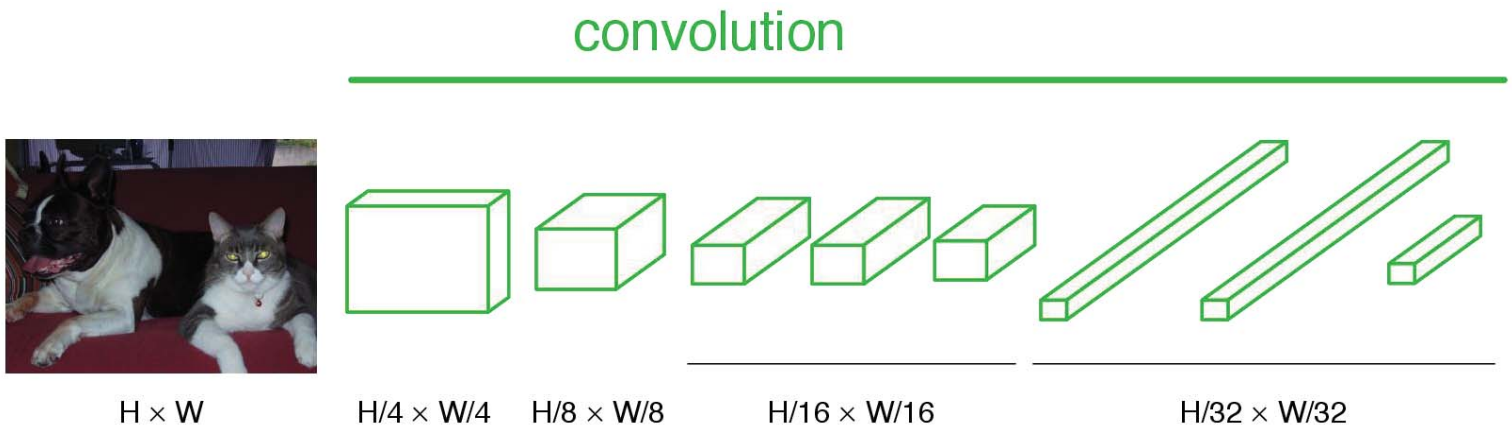
a classification network



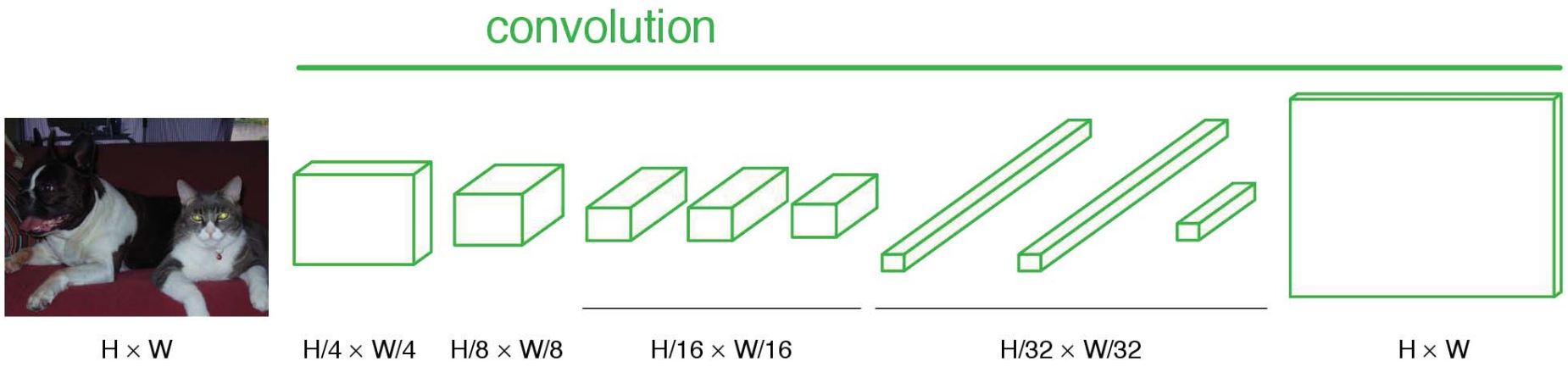
becoming fully convolutional



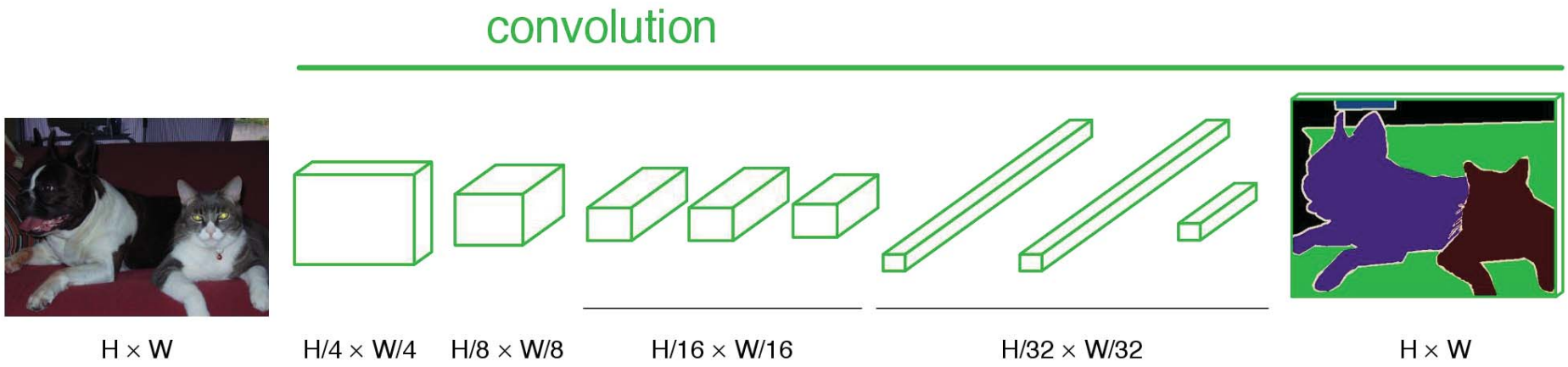
becoming fully convolutional



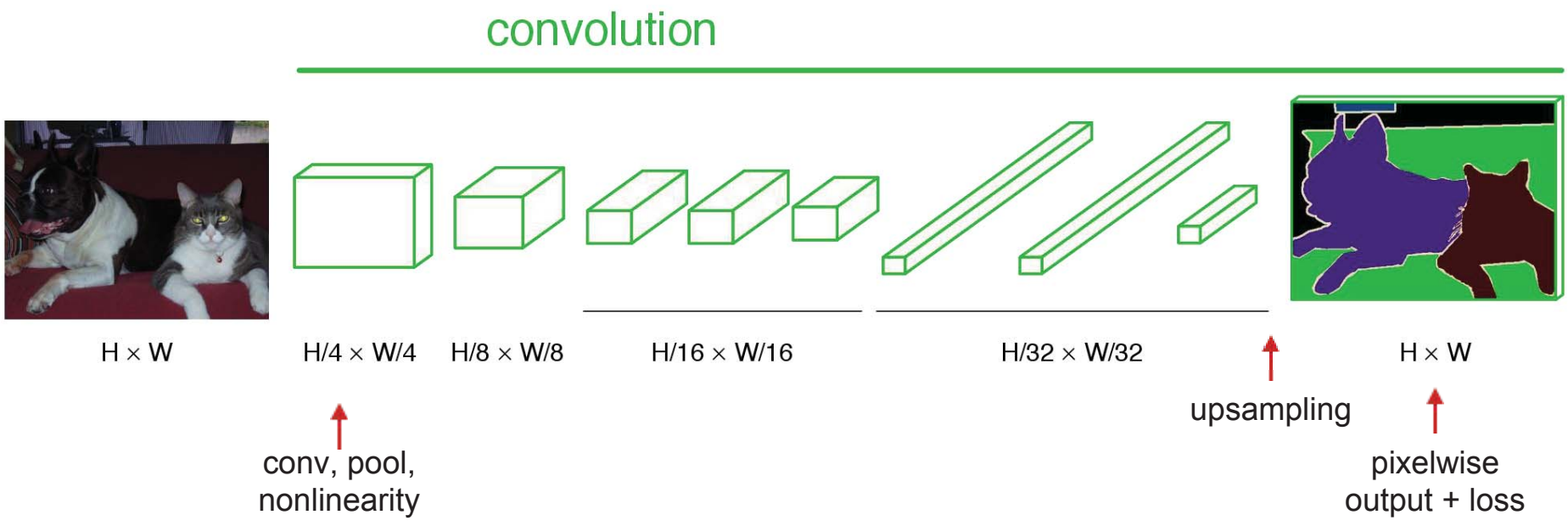
upsampling output



end-to-end, pixels-to-pixels network

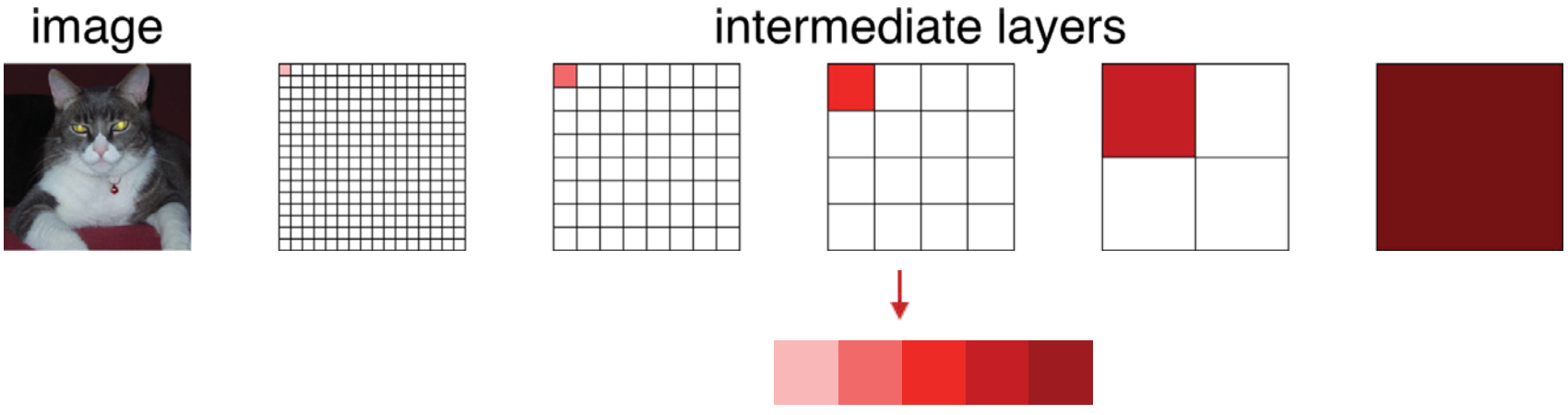


end-to-end, pixels-to-pixels network



spectrum of deep features

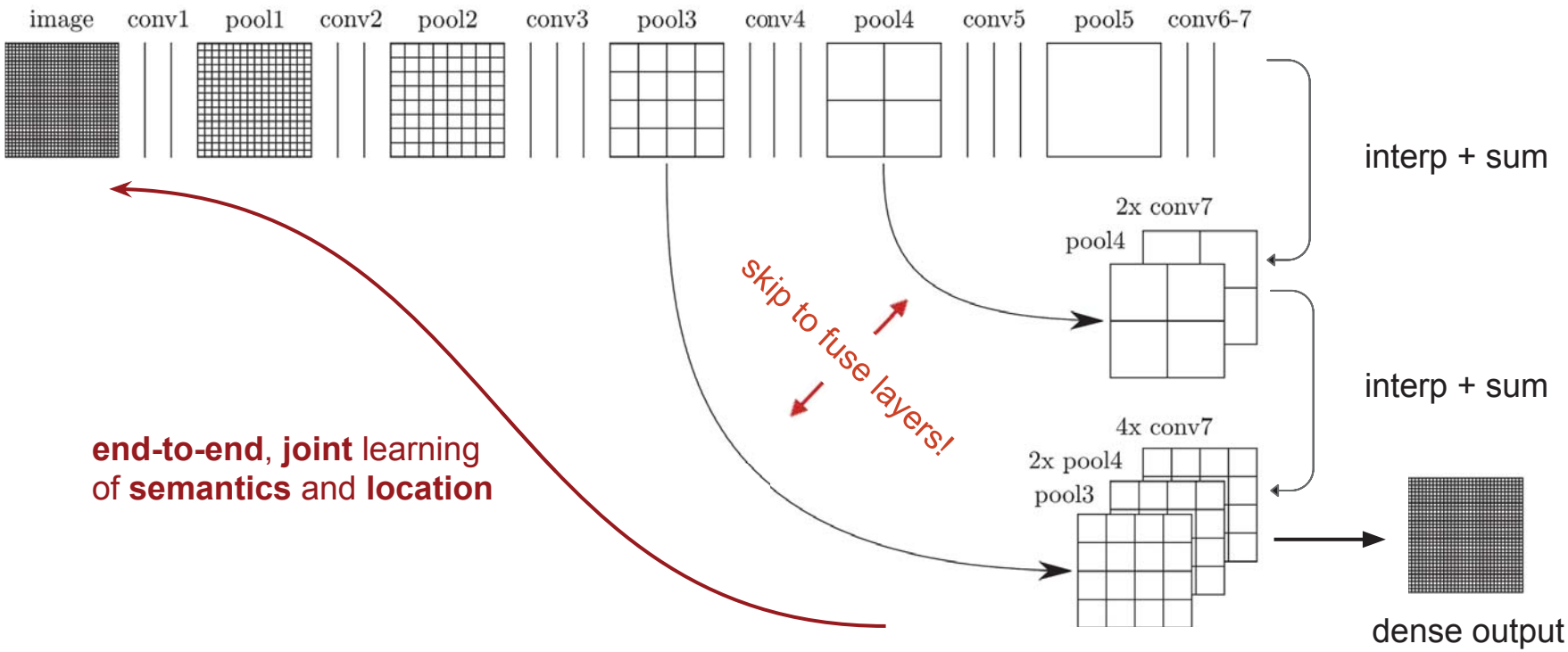
combine *where* (local, shallow) with *what* (global, deep)



fuse features into **deep jet**

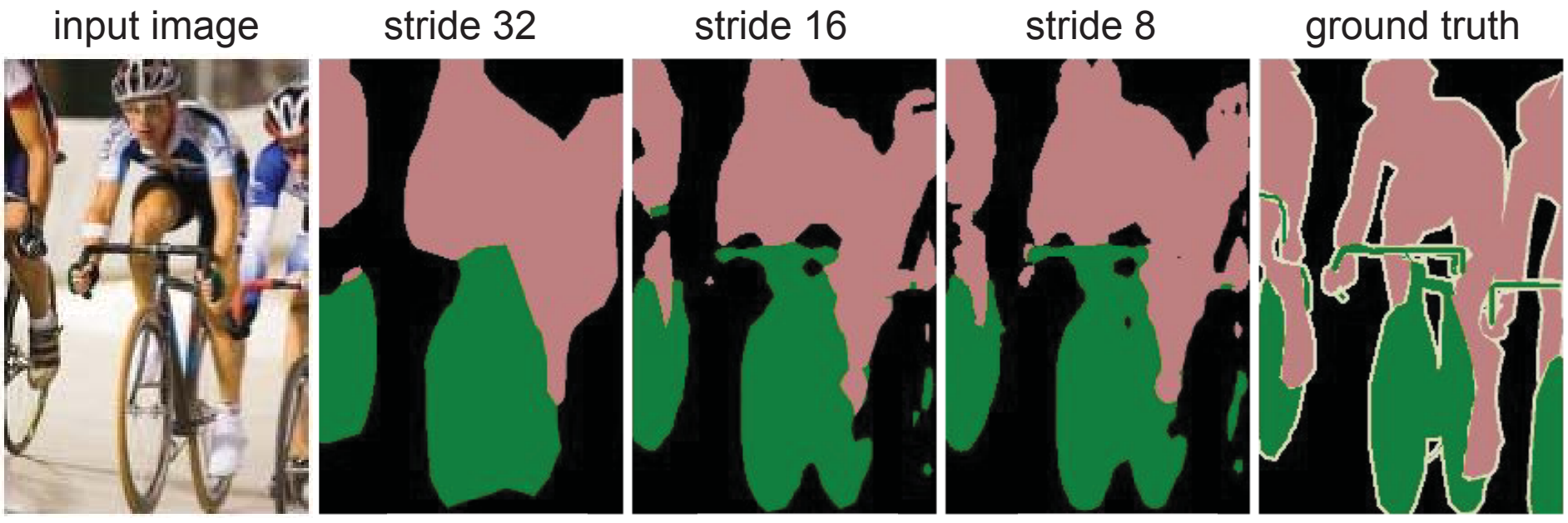
(cf. Hariharan et al. CVPR15 “hypercolumn”)

skip layers

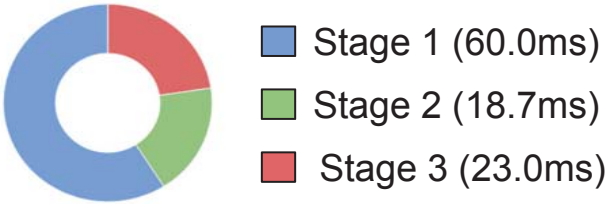


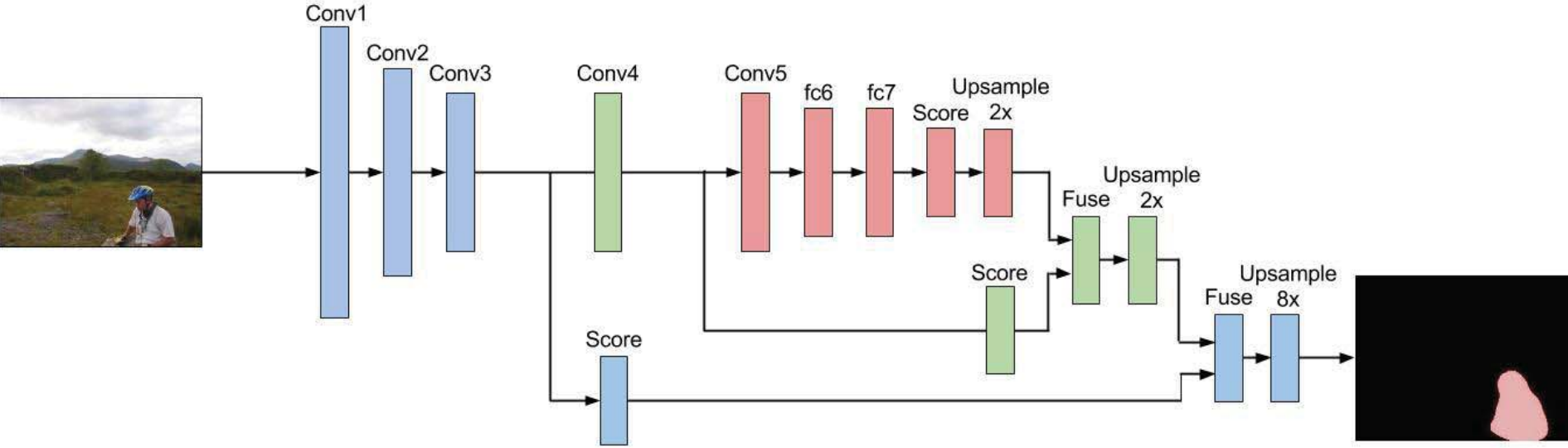
end-to-end, **joint** learning
of **semantics** and **location**

skip layer refinement

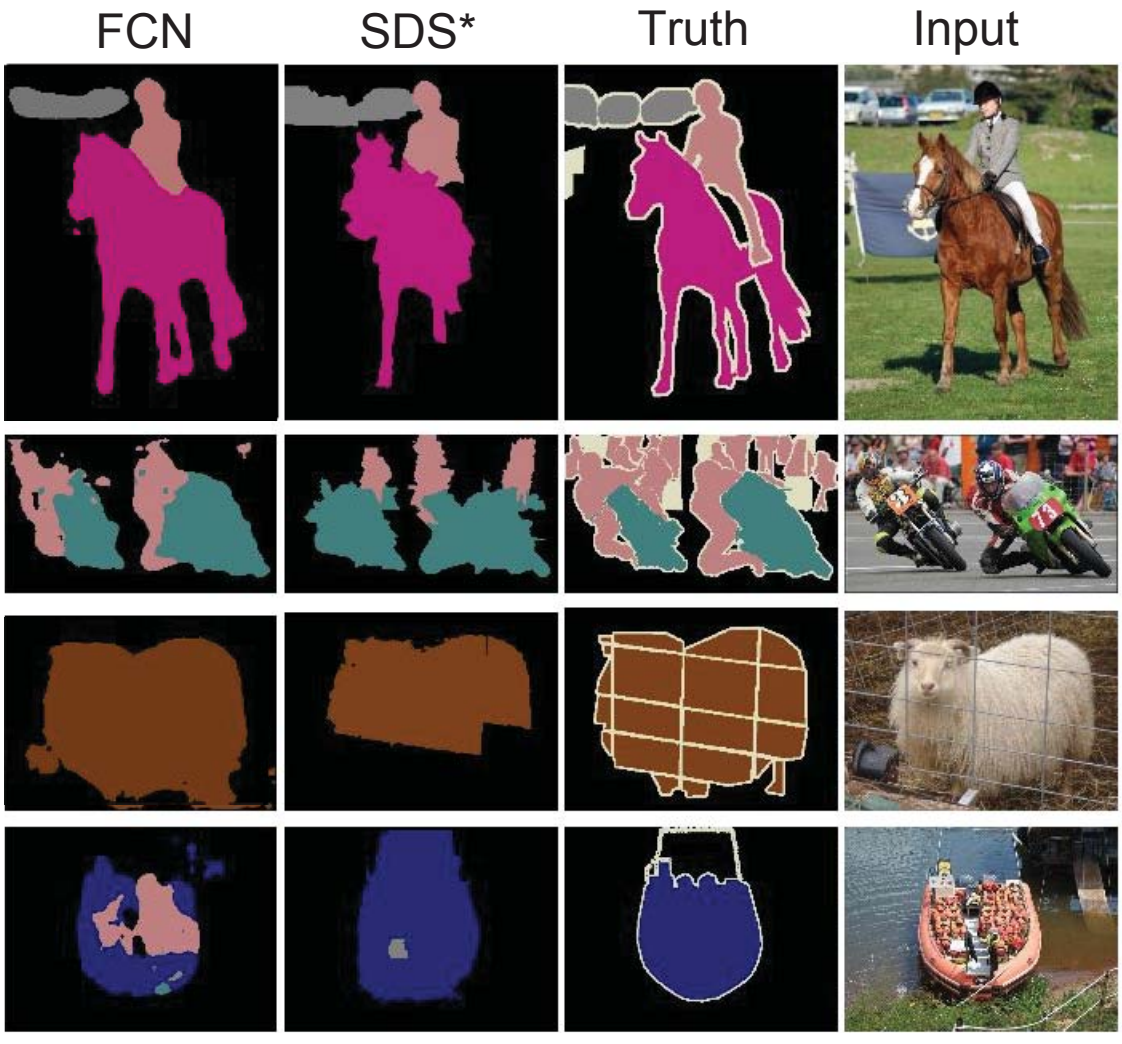


skip FCN computation





A multi-stream network that fuses features/predictions across layers

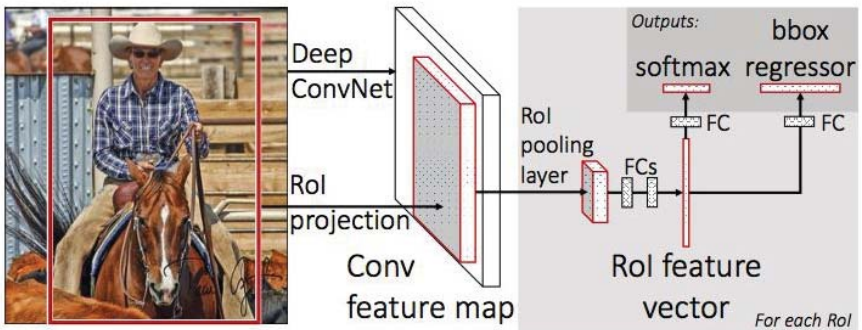


Relative to prior state-of-the-art SDS:

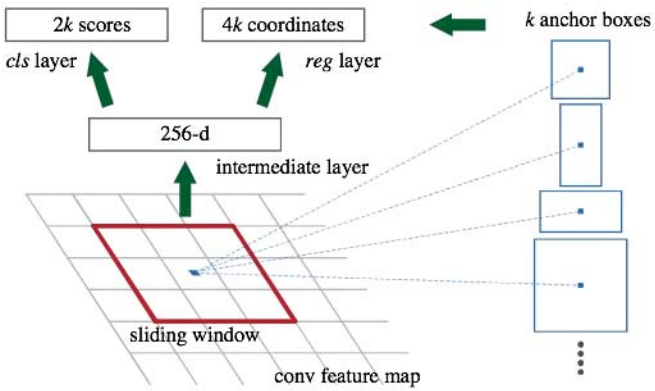
- 30% relative improvement for mean IoU
- 286× faster

*Simultaneous Detection and Segmentation
Hariharan et al. ECCV14

detection: fully conv. proposals



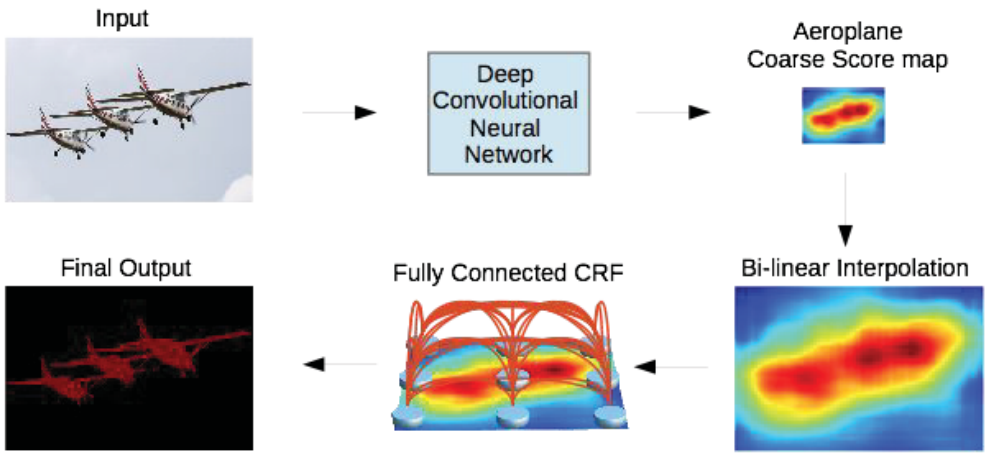
Fast R-CNN, *Girshick* ICCV'15

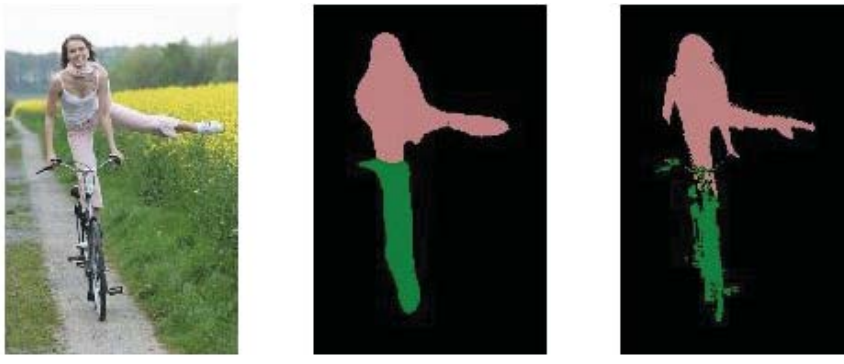


Faster R-CNN, *Ren et al.* NIPS'15

end-to-end detection by proposal FCN RoI classification

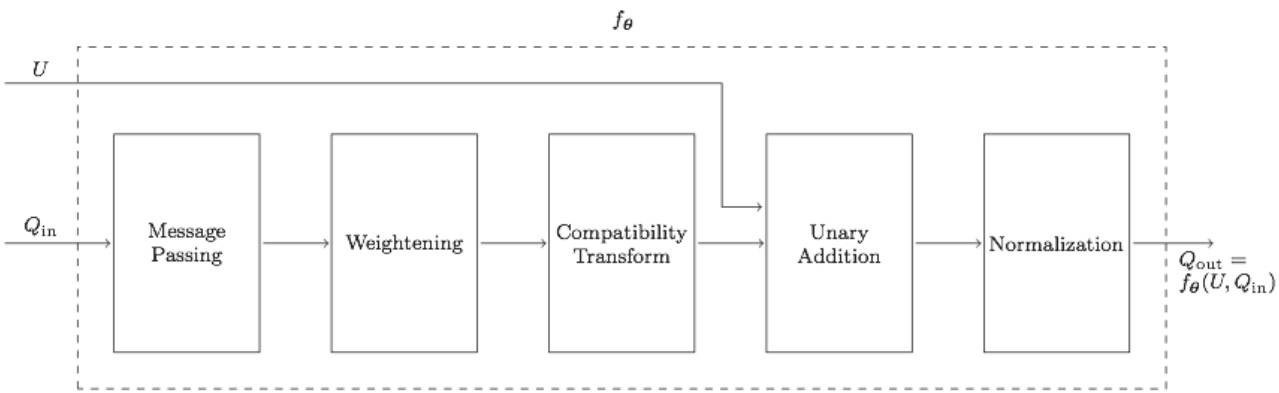
fully conv. nets + structured output





Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs.
Chen* & Papandreou* et al. ICLR 2015.

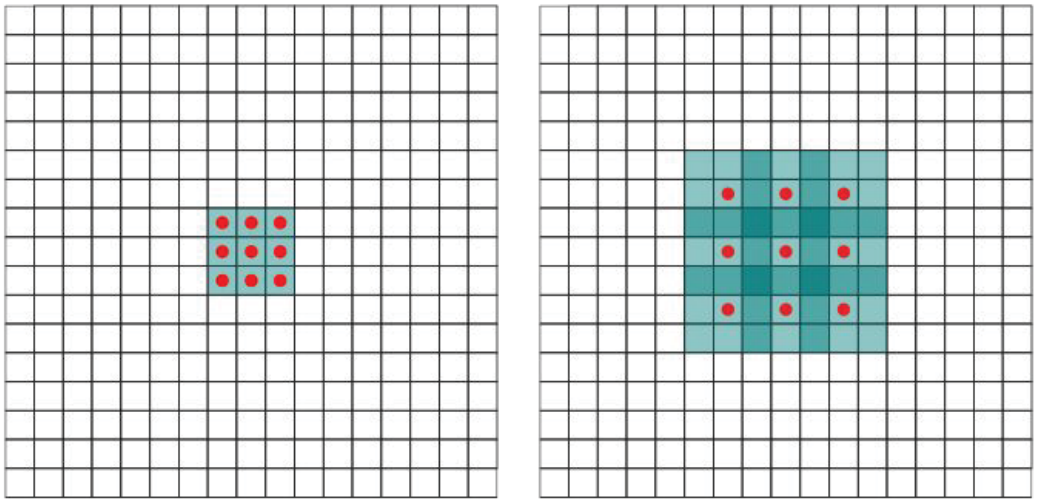
fully conv. nets + structured output



Method	Without COCO	With COCO
Plain FCN-8s	61.3	68.3
FCN-8s and CRF disconnected	63.7	69.5
End-to-end training of CRF-RNN	69.6	72.9

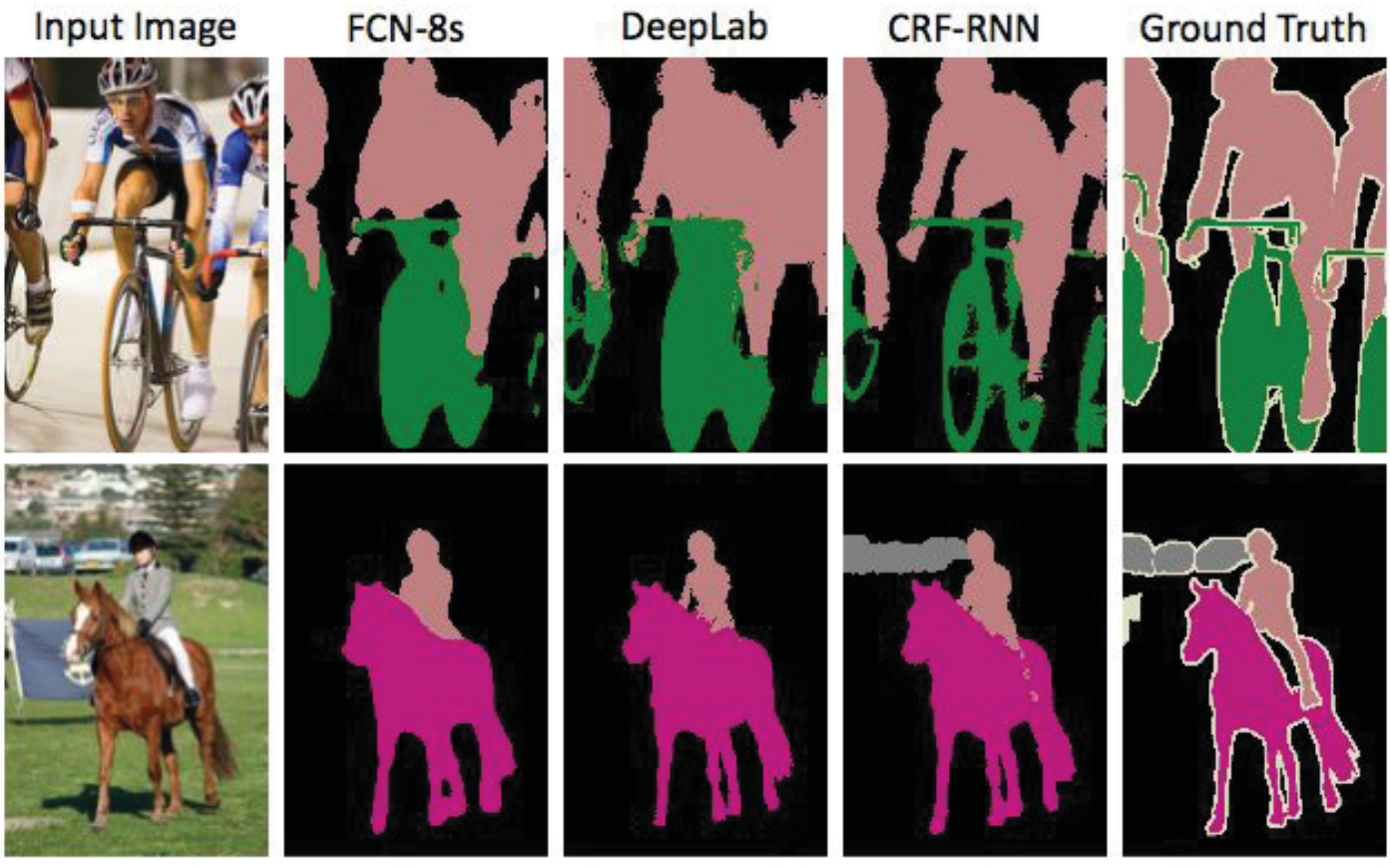
Conditional Random Fields as Recurrent Neural Networks. *Zheng* & Jayasumana* et al.* ICCV 2015.

dilation for structured output



- enlarge effective receptive field for same no. params
- raise resolution
- convolutional context model: similar accuracy to CRF but non-probabilistic

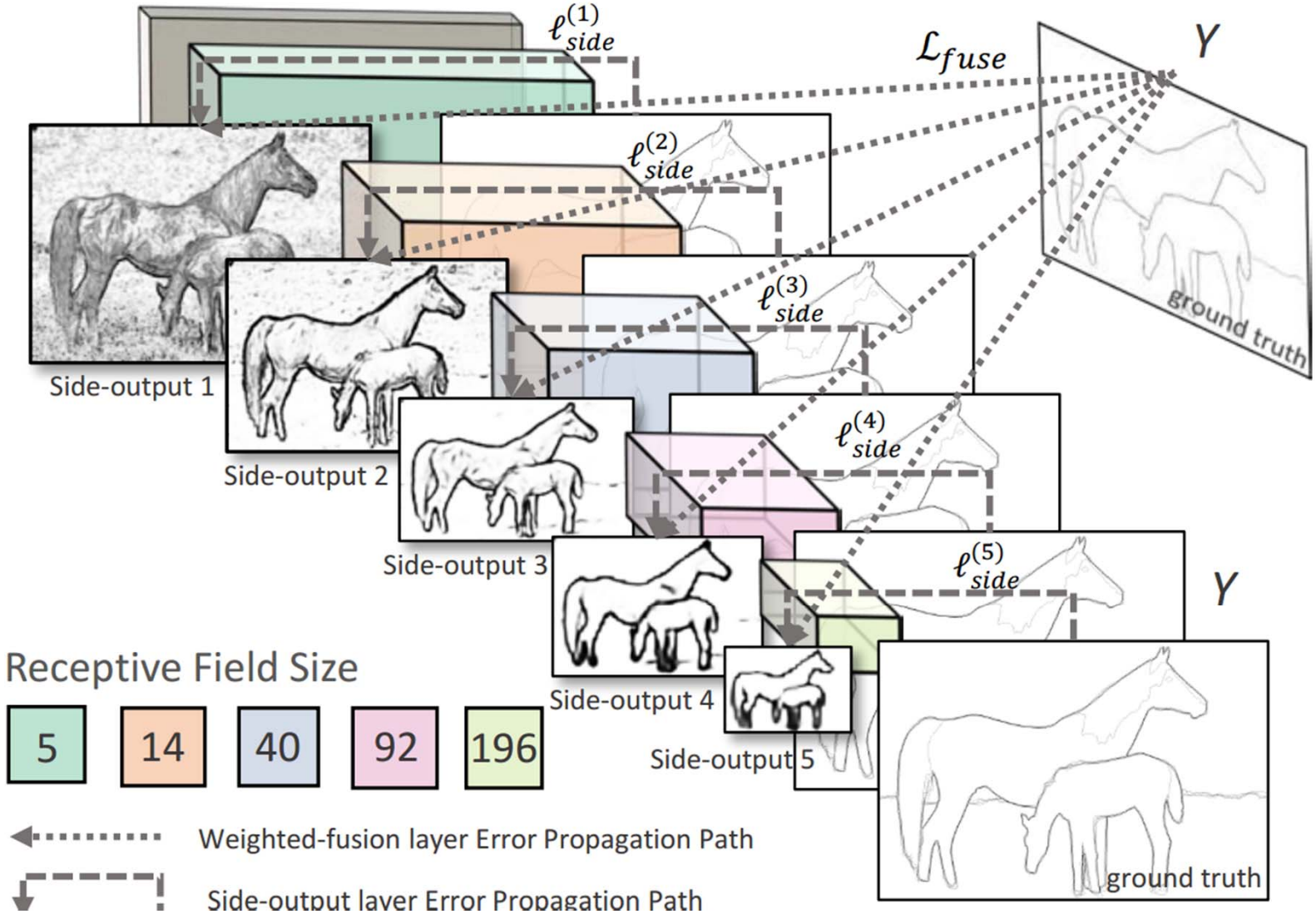
Multi-Scale Context Aggregation by Dilated Convolutions. *Yu & Koltun.* ICLR 2016



[comparison credit: CRF as RNN, Zheng* & Jayasumana* et al. ICCV 2015]

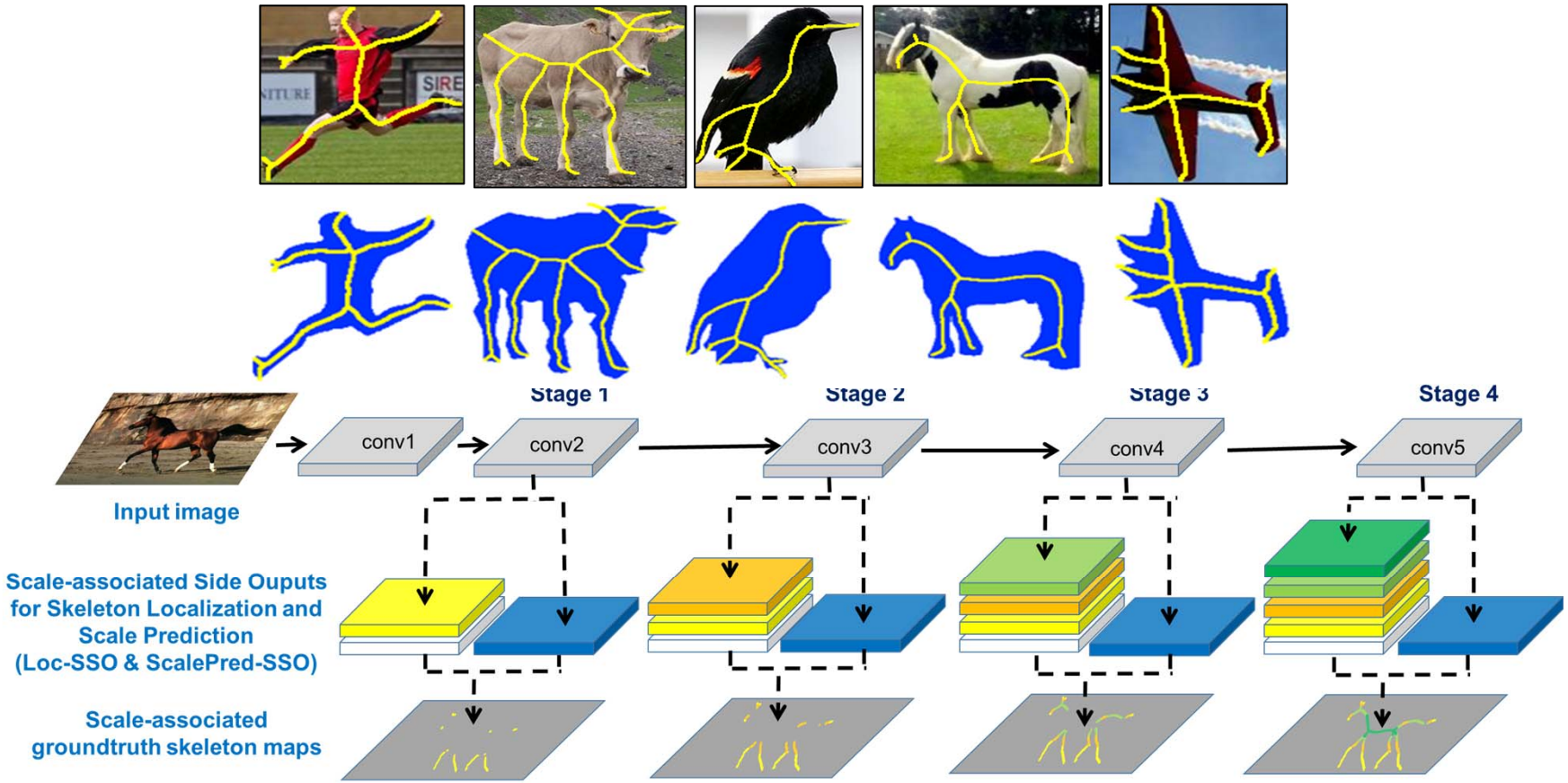
DeepLab: Chen* & Papandreou* et al. ICLR 2015. **CRF-RNN**: Zheng* & Jayasumana* et al. ICCV 2015

Edge Detection



Saining Xie and Zhuowen Tu, "Holistically-Nested Edge Detection", ICCV 2015

Object Skeleton Detection



- **Wei Shen**, Kai Zhao, Yuan Jiang, Yan Wang, Zhijiang Zhang, Xiang Bai. Object Skeleton Extraction in Natural Images by Fusing Scale-associated Deep Side Outputs. **CVPR**, 2016.
- **Wei Shen**, Kai Zhao, Yuan Jiang, Yan Wang, Xiang Bai, Alan Yuille. DeepSkeleton: Learning Multi-task Scale-associated Deep Side Outputs for Object Skeleton Extraction in Natural Images. **IEEE Trans. Image Processing**, 2017.