04/06/17

Convolutional Neural Networks

Computer Vision

CS 543 / ECE 549

University of Illinois

Derek Hoiem

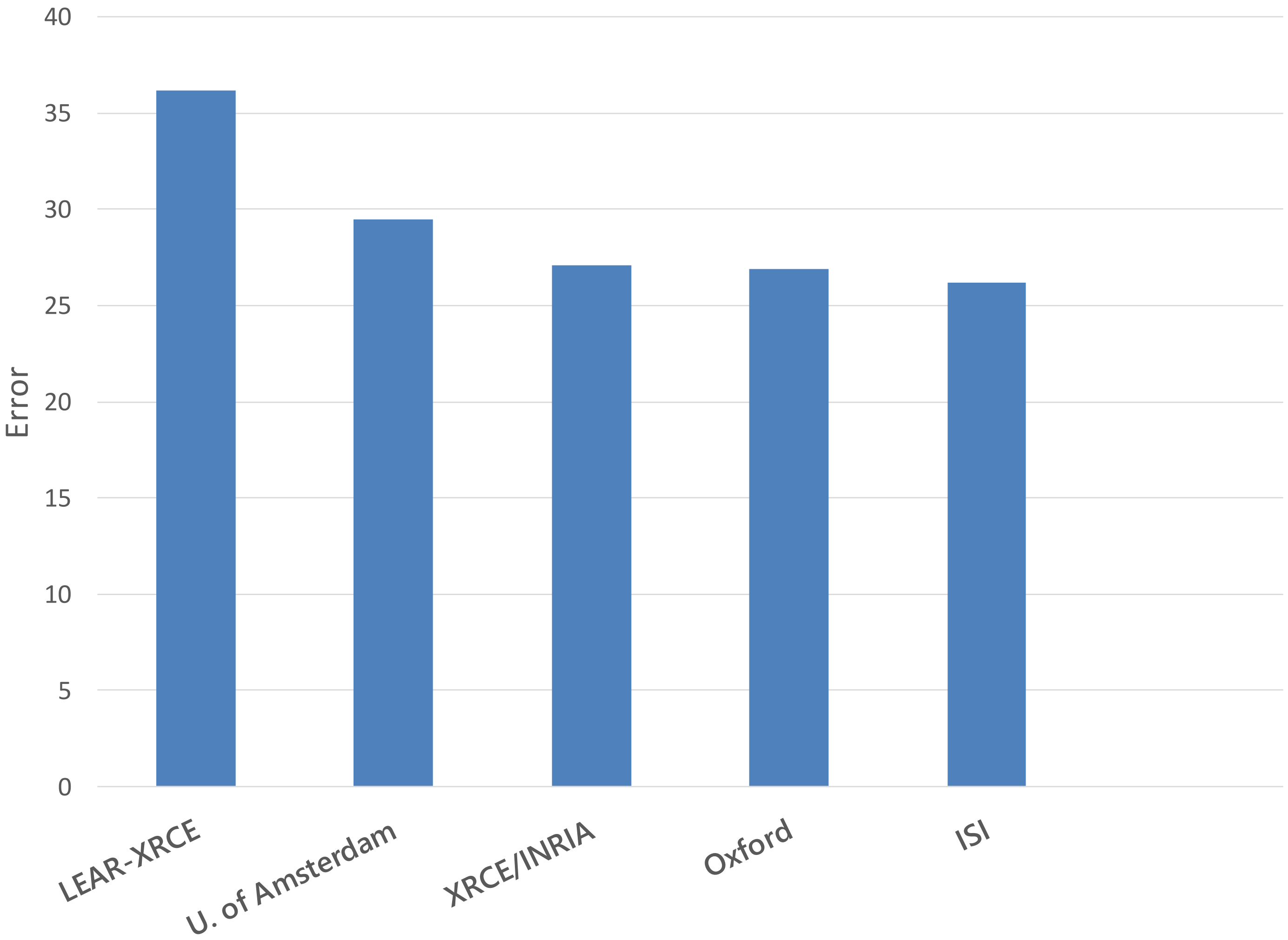
Many slides from Lana Lazebnik, and some from Jia-bin Huang

History of deep convolutional nets

- 1950’s: neural nets (perceptron) invented by Rosenblatt
- 1980’s/1990’s: Neural nets are popularized and then abandoned as being interesting idea but impossible to optimize or “unprincipled”
- 1990’s: LeCun achieves state-of-art performance on character recognition with convolutional network (main ideas of today’s networks)
- 2000’s: Hinton, Bottou, Bengio, LeCun, Ng, and others keep trying stuff with deep networks but without much traction/acclaim in vision
- 2010-2011: Substantial progress in some areas, but vision community still unconvinced
  - Some neural net researchers get ANGRY at being ignored/rejected
- 2012: shock at ECCV 2012 with ImageNet challenge

2012 ImageNet 1K

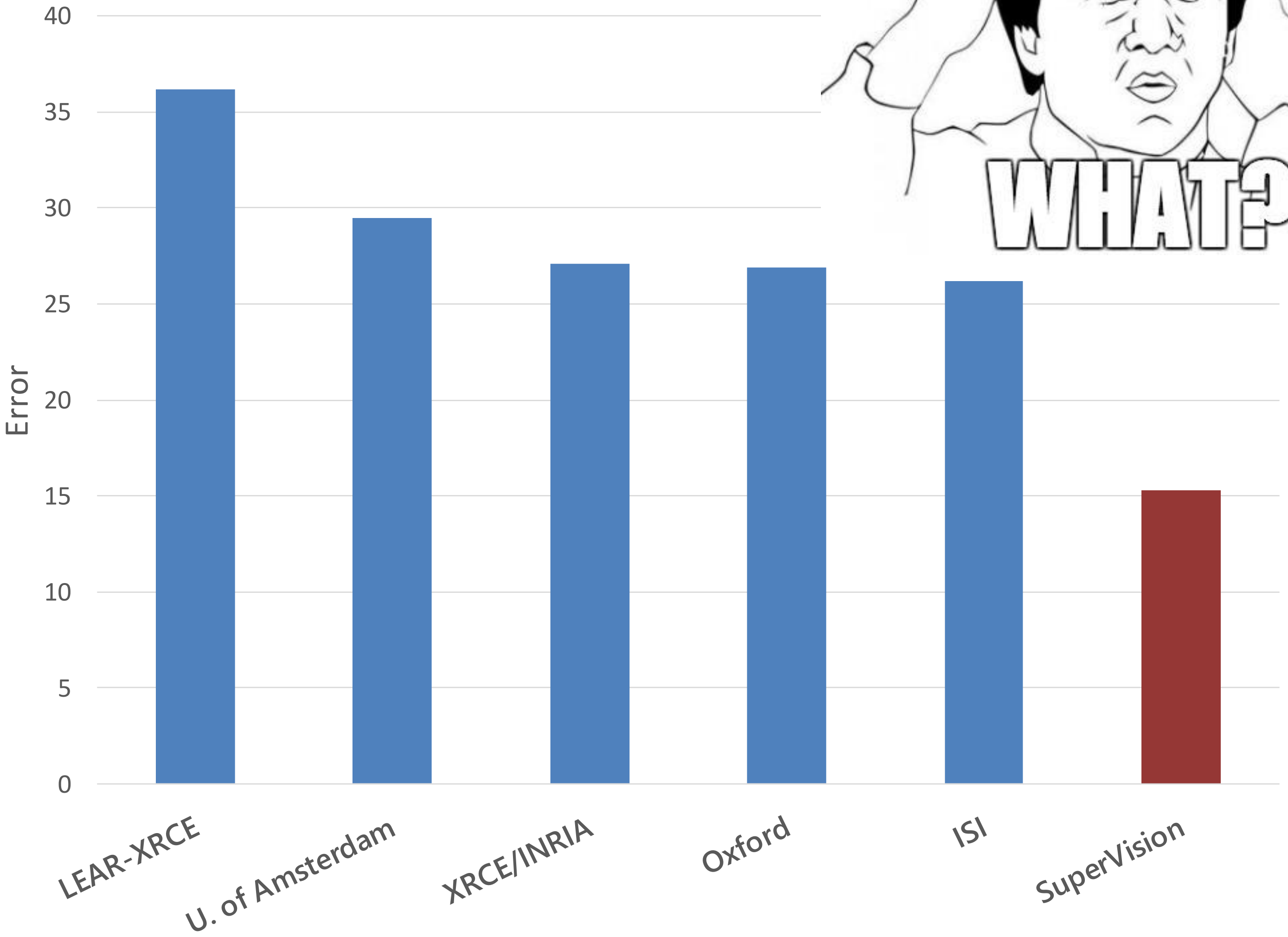
(Fall 2012)

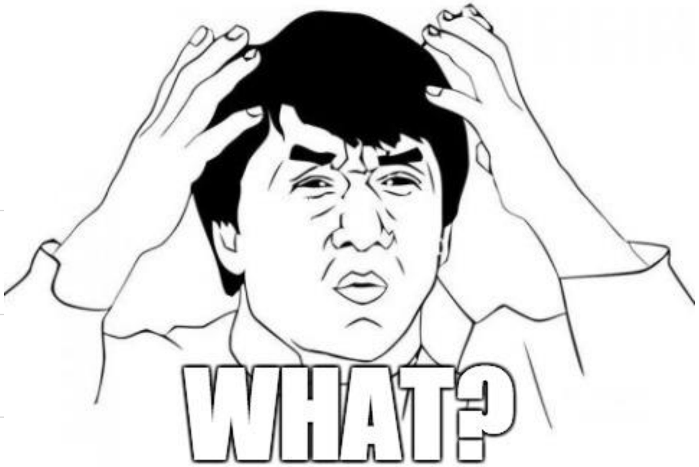


Slide: Jia-bin Huang

2012 ImageNet 1K

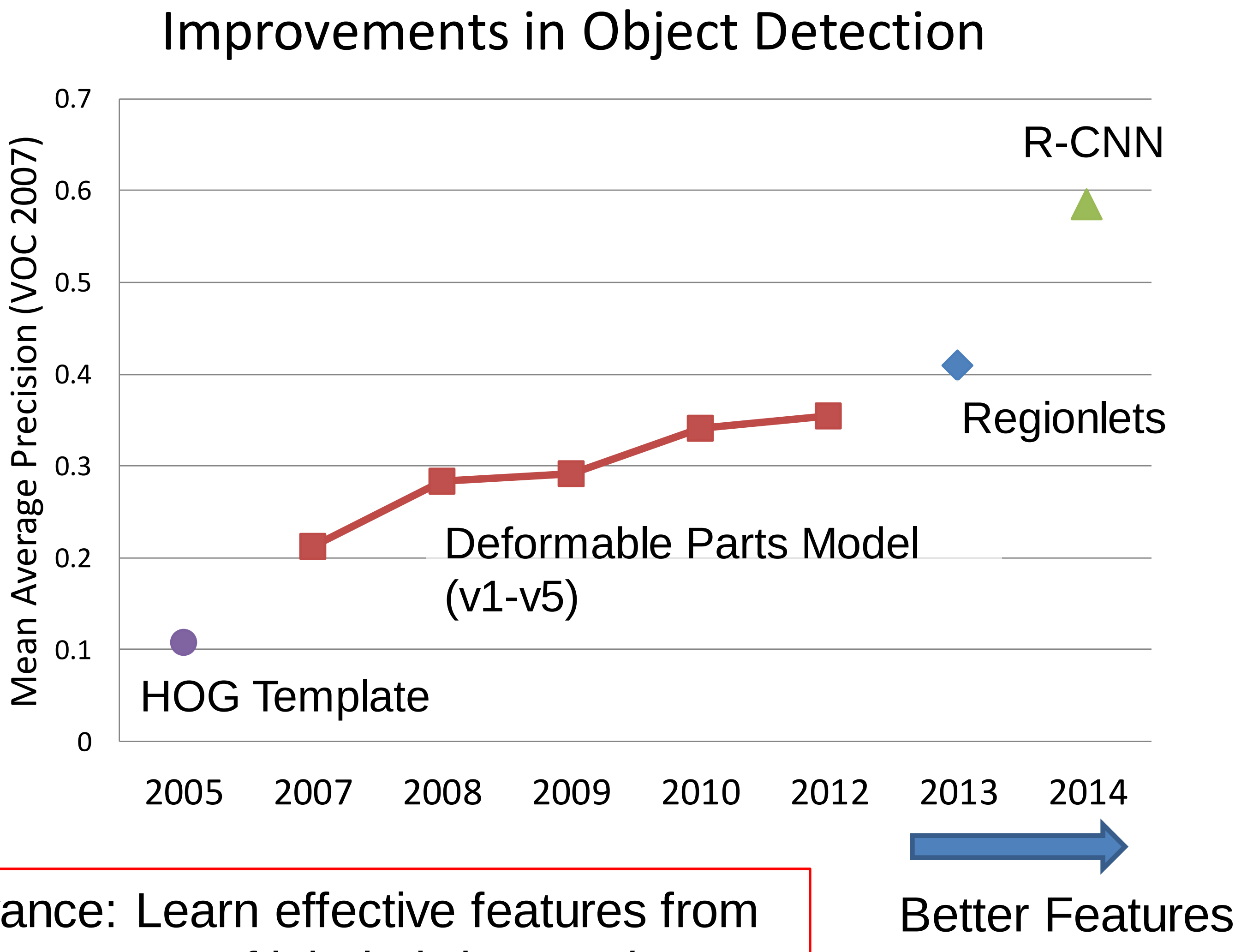
(Fall 2012)





Slide: Jia-bin Huang

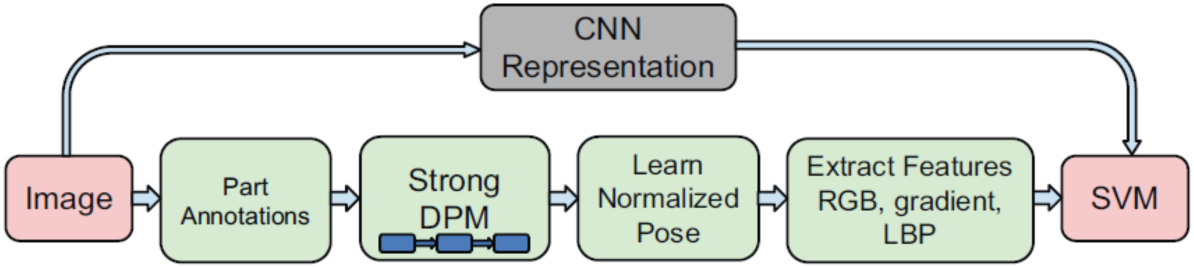
R-CNN demonstrates major detection improvement by pre-training on ImageNet and fine-tuning on PASCAL

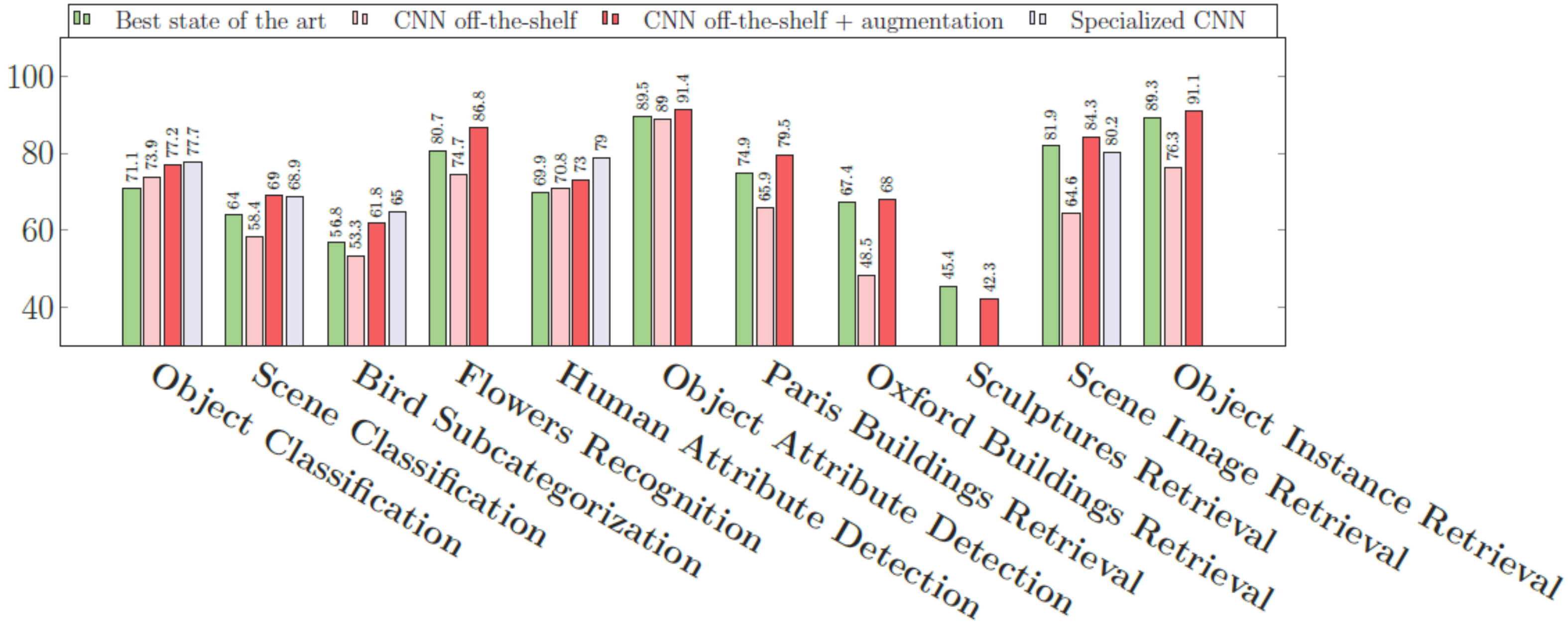


Key Advance: Learn effective features from massive amounts of labeled data *and* adapt to new tasks with less data

HOG: Dalal-Triggs 2005 DPM: Felzenszwalb et al. 2008-2012 Regionlets: Wang et al. 2013 R-CNN: Girshick et al. 2014

“CNN Features off-the-shelf: an Astounding Baseline for Recognition”





Razavian et al. CVPR 2014

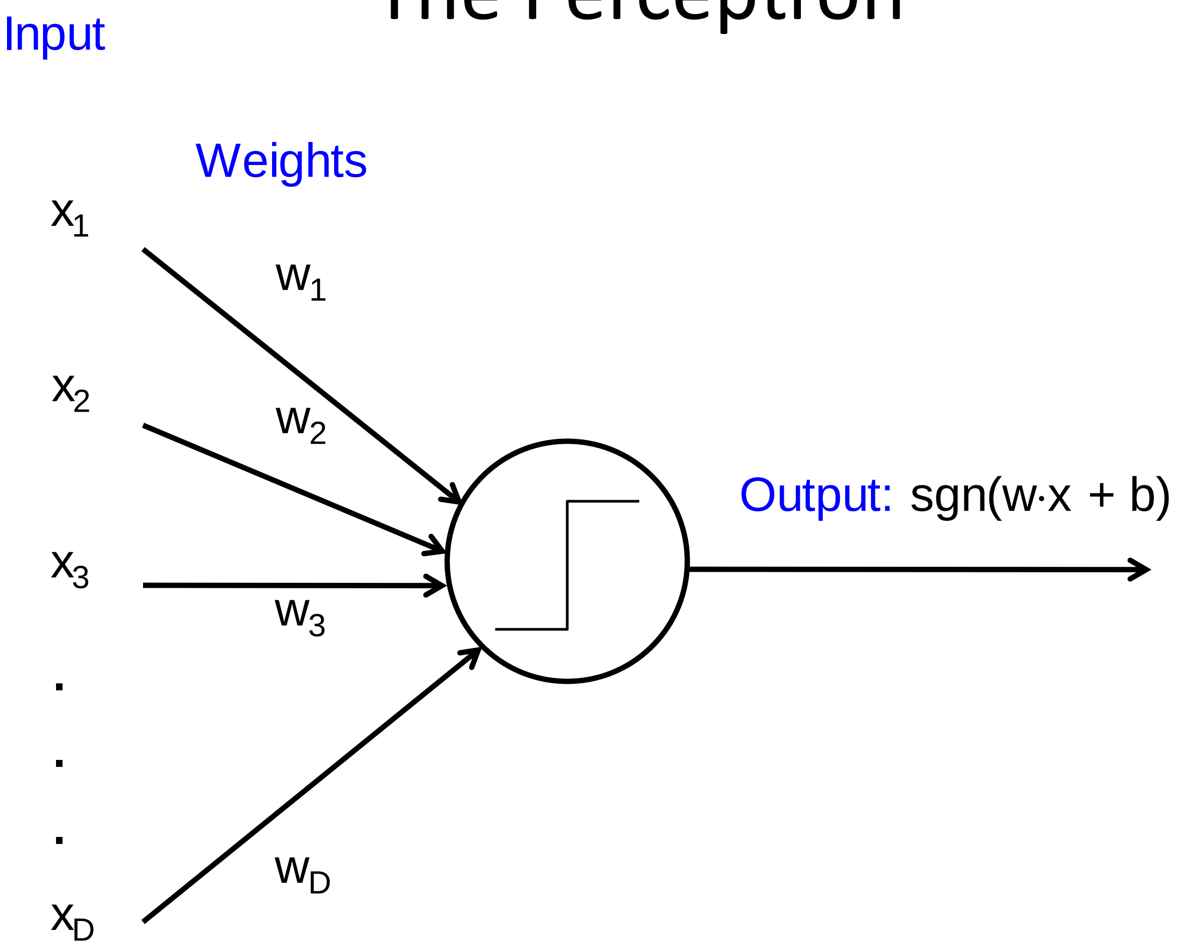
How it felt to be an object recognition researcher

https://youtu.be/XCtuZ-fDL2E?t=140

Slide: Lazebnik

Rewind…

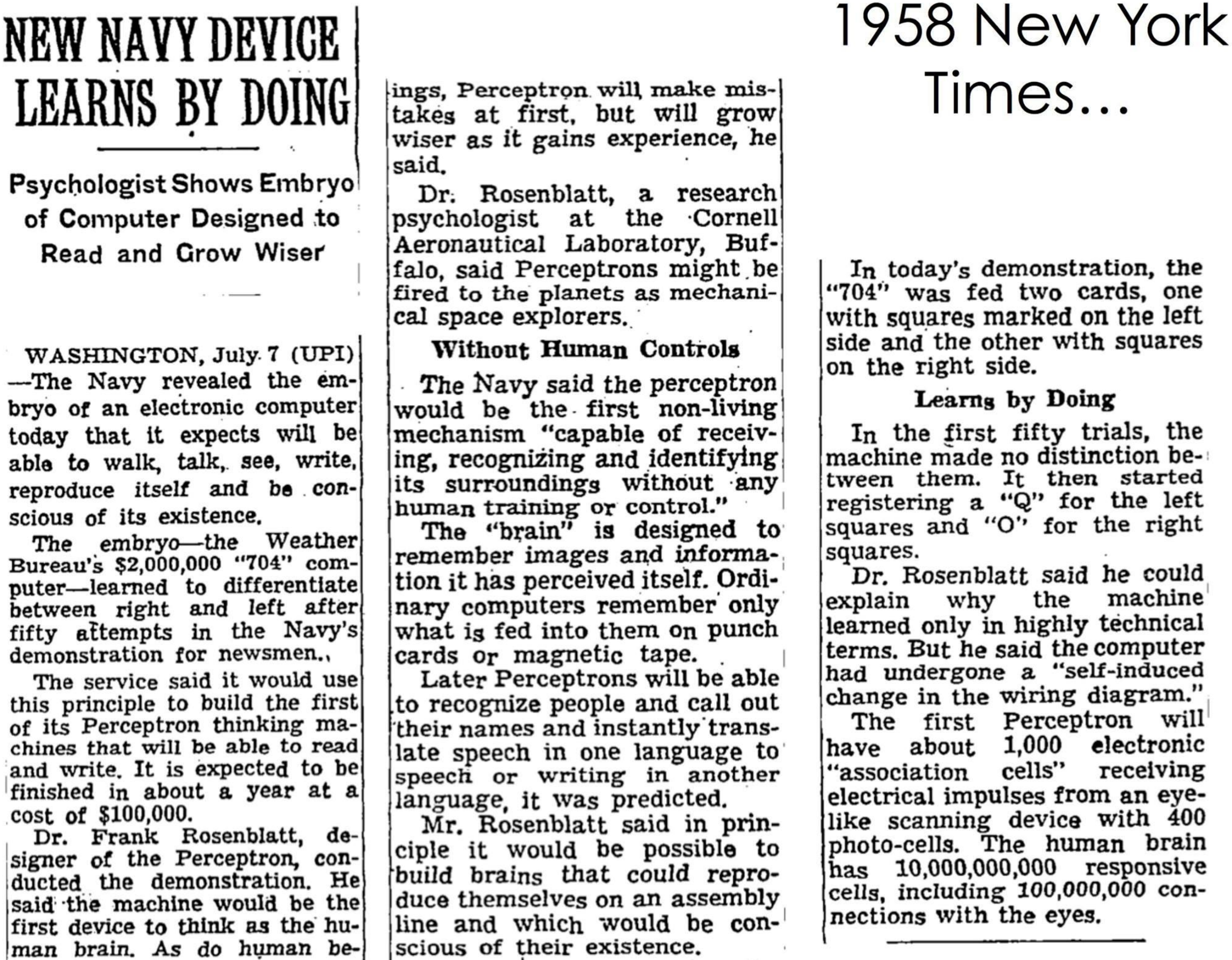
The Perceptron



Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408.

Slide: Lazebnik

1958 New York Times...



NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.,

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

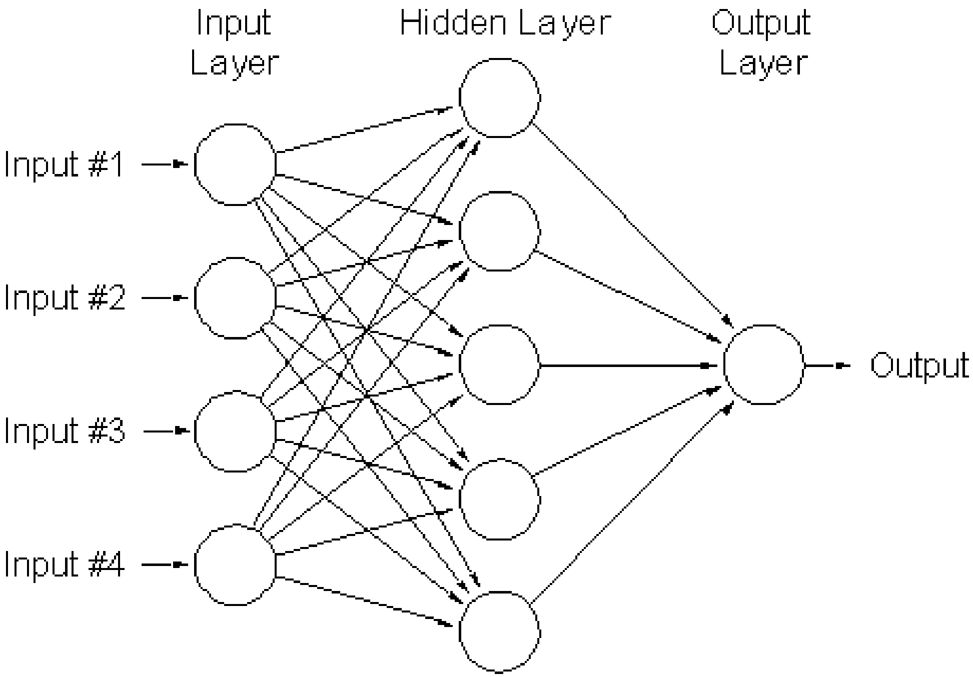
Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

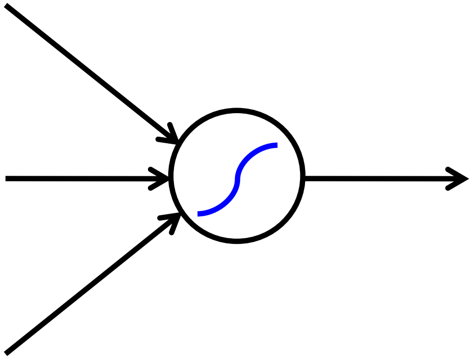
Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

Two-layer neural network



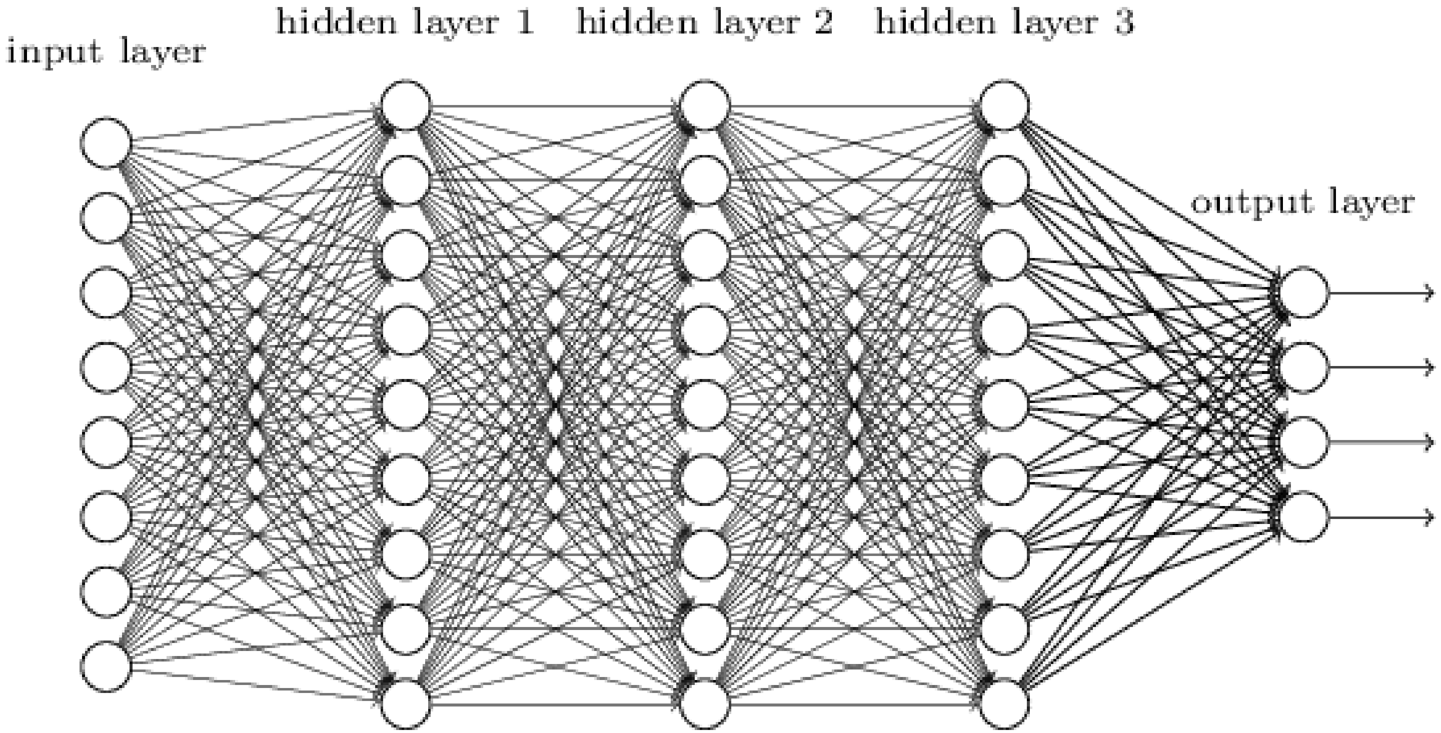
- Can learn nonlinear functions provided each perceptron has a differentiable nonlinearity



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

Slide: Lazebnik

Multi-layer neural network



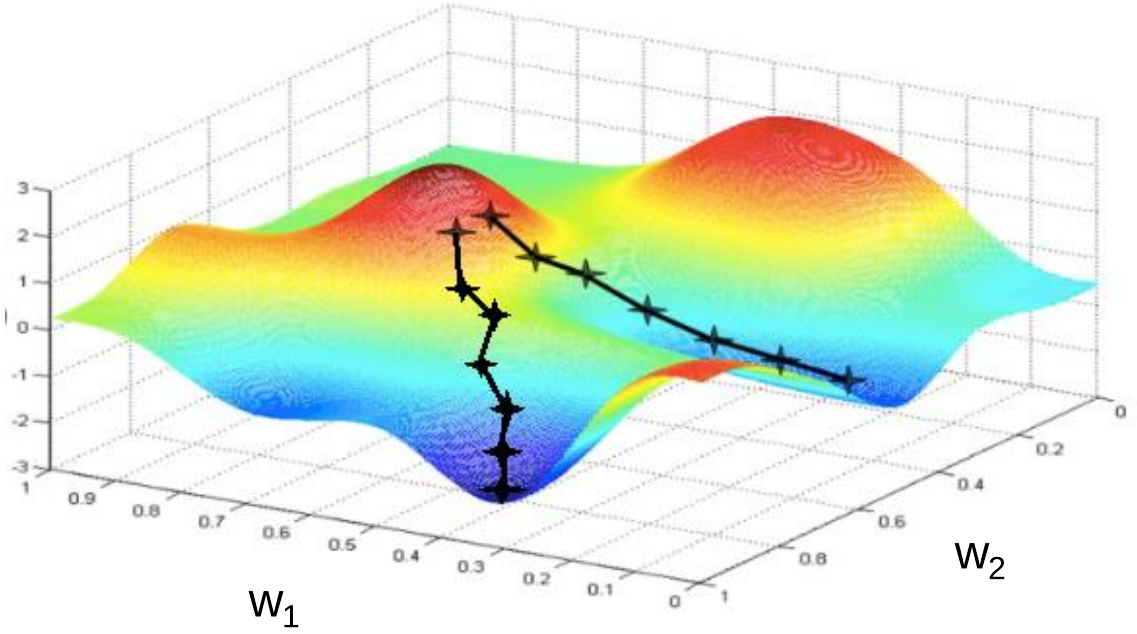
Slide: Lazebnik

Training of multi-layer networks

- Find network weights to minimize the *training error* between true and estimated labels of training examples, e.g.:

$$E(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - f_{\mathbf{w}}(\mathbf{x}_i) \right)^2$$

- Update weights by **gradient descent:** $\mathbf{w} \leftarrow \mathbf{w} - \alpha \dfrac{\partial E}{\partial \mathbf{w}}$



Slide: Lazebnik

Training of multi-layer networks

- Find network weights to minimize the *training error* between true and estimated labels of training examples, e.g.:

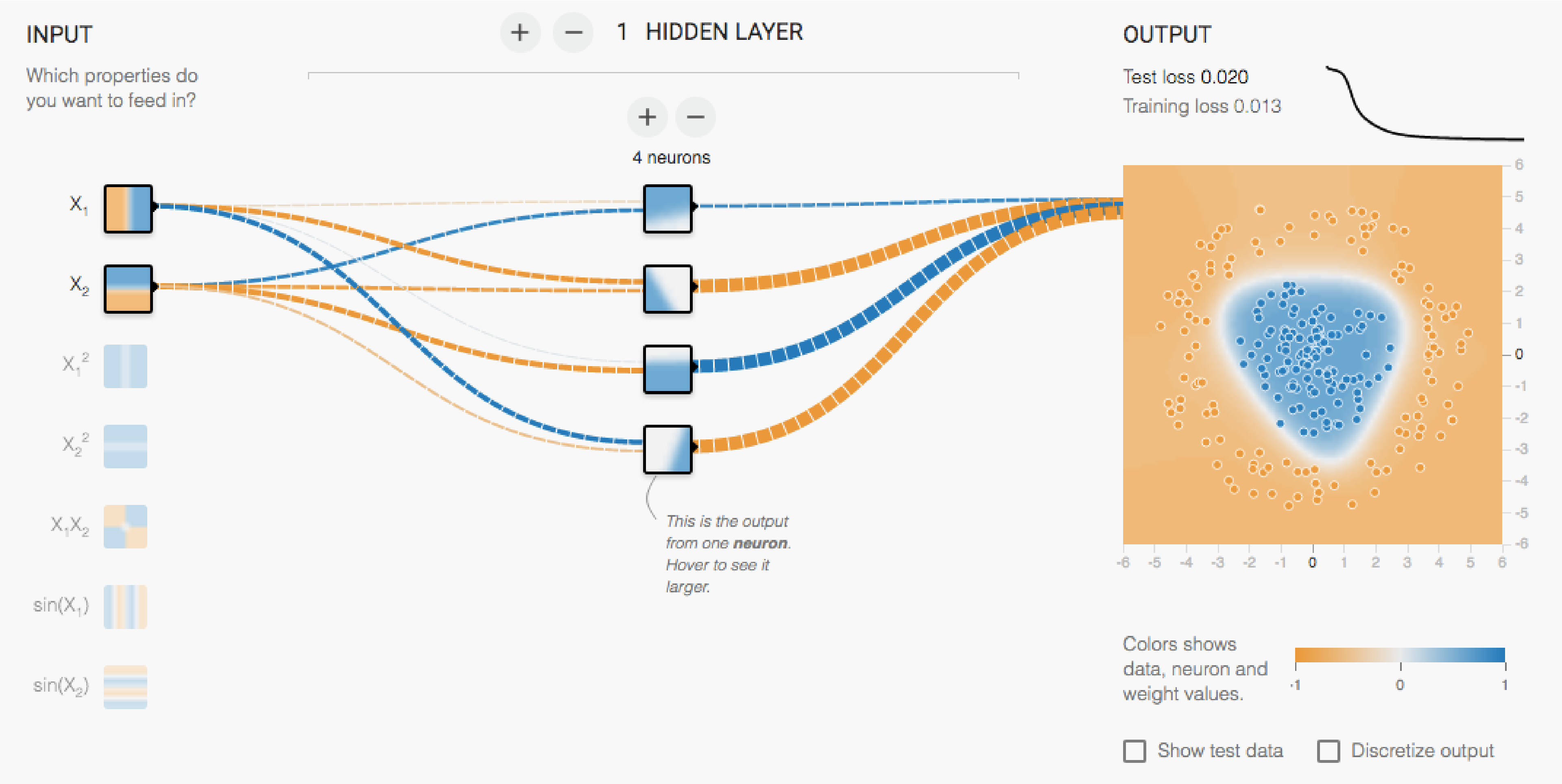
$$E(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - f_{\mathbf{w}}(\mathbf{x}_i) \right)^2$$

- Update weights by **gradient descent:** $\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$

- **Back-propagation:** gradients are computed in the direction from output to input layers and combined using chain rule
- **Stochastic gradient descent:** compute the weight update w.r.t. a small batch of examples at a time, cycle through training examples in random order in multiple epochs

Slide: Lazebnik

Multi-Layer Network Demo



INPUT

Which properties do you want to feed in?

1 HIDDEN LAYER

4 neurons

OUTPUT

Test loss 0.020

Training loss 0.013

This is the output from one ***neuron****. Hover to see it larger.*

Colors shows data, neuron and weight values.

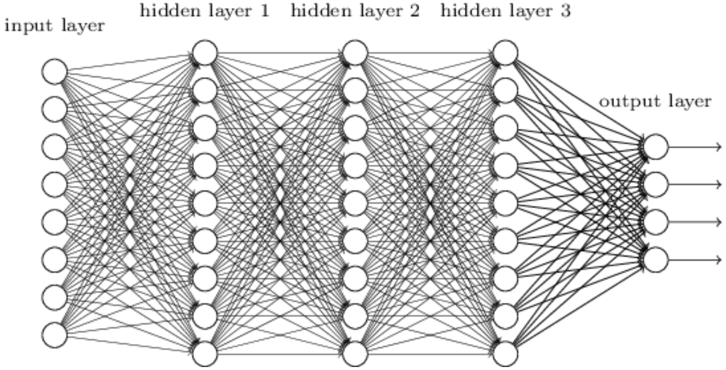
□ Show test data

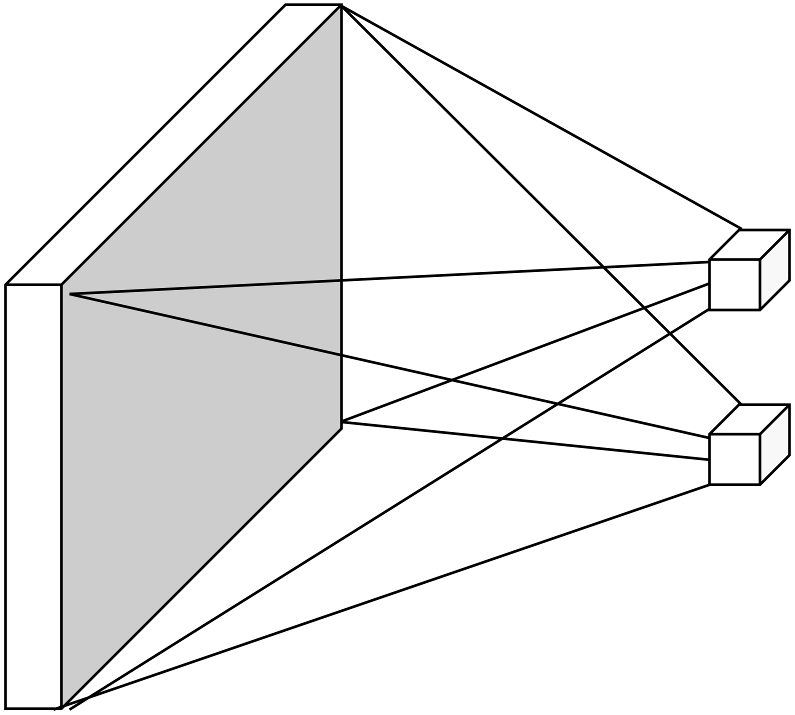
□ Discretize output

http://playground.tensorflow.org/

Slide: Lazebnik

From fully connected to convolutional networks



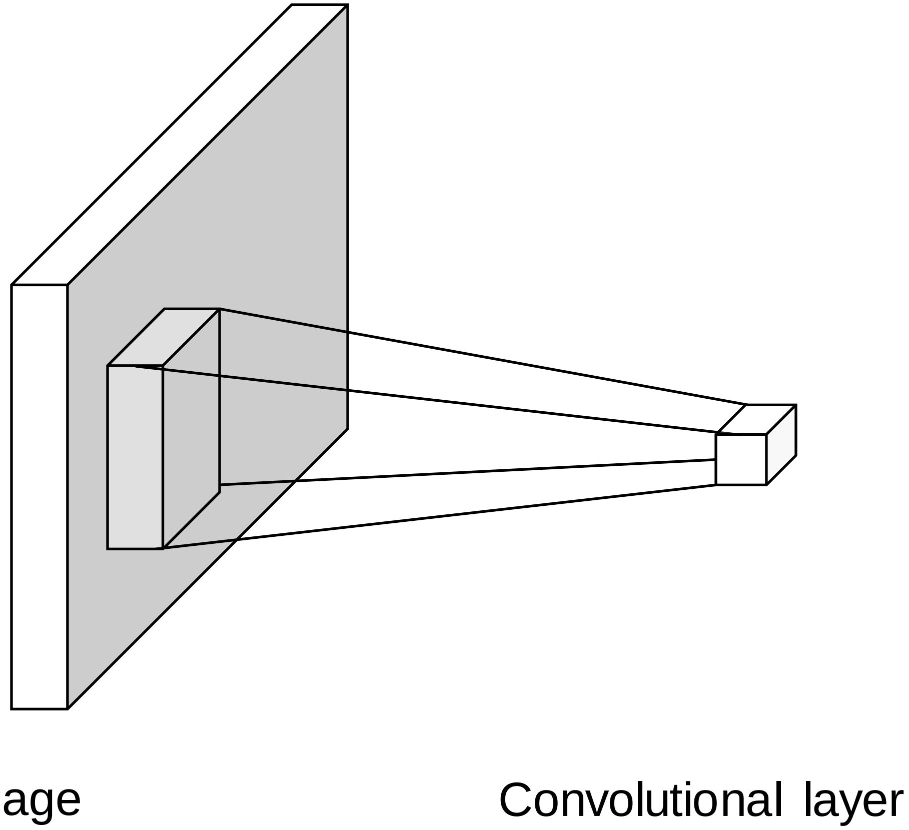


image

Fully connected layer

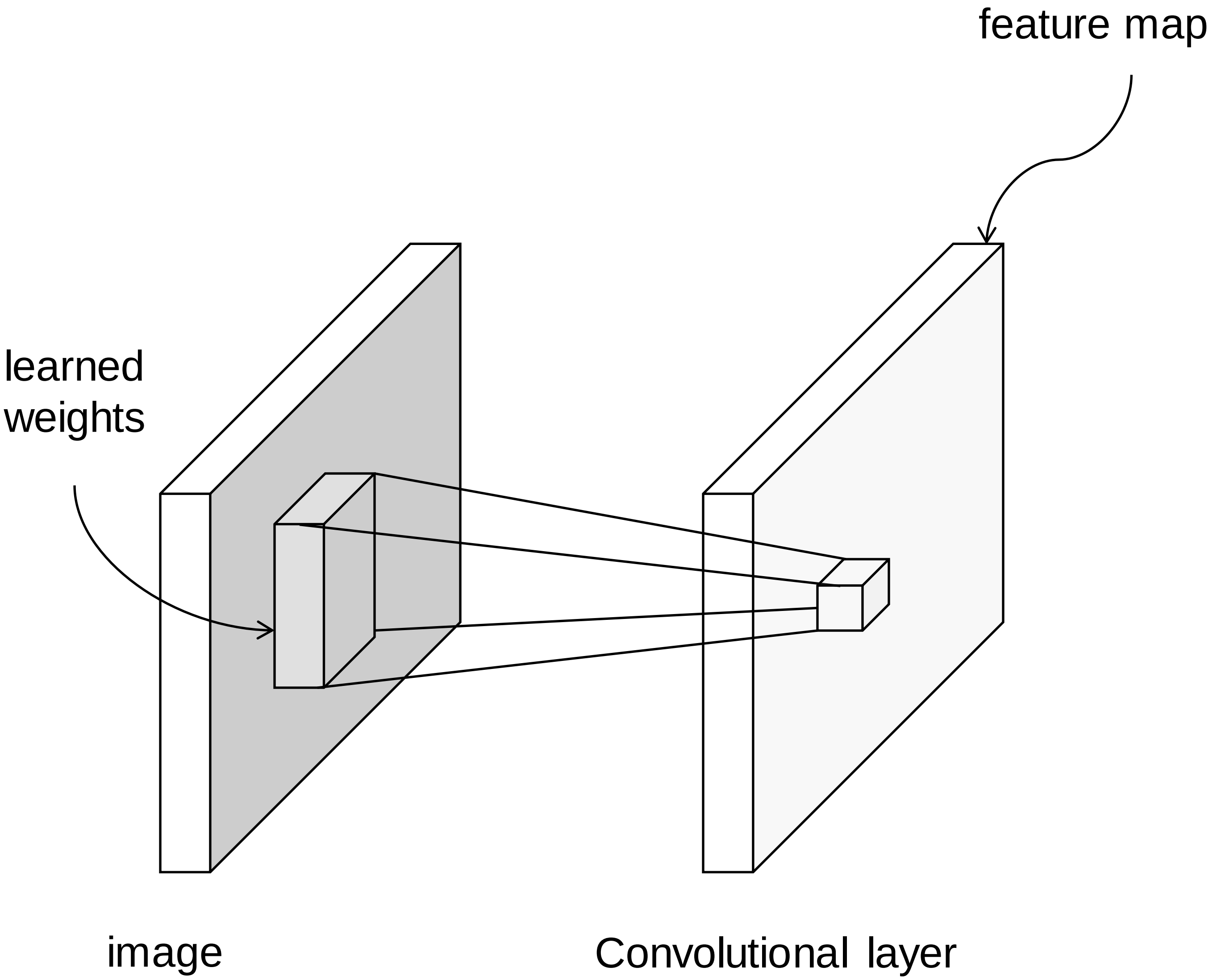
Slide: Lazebnik

From fully connected to convolutional networks



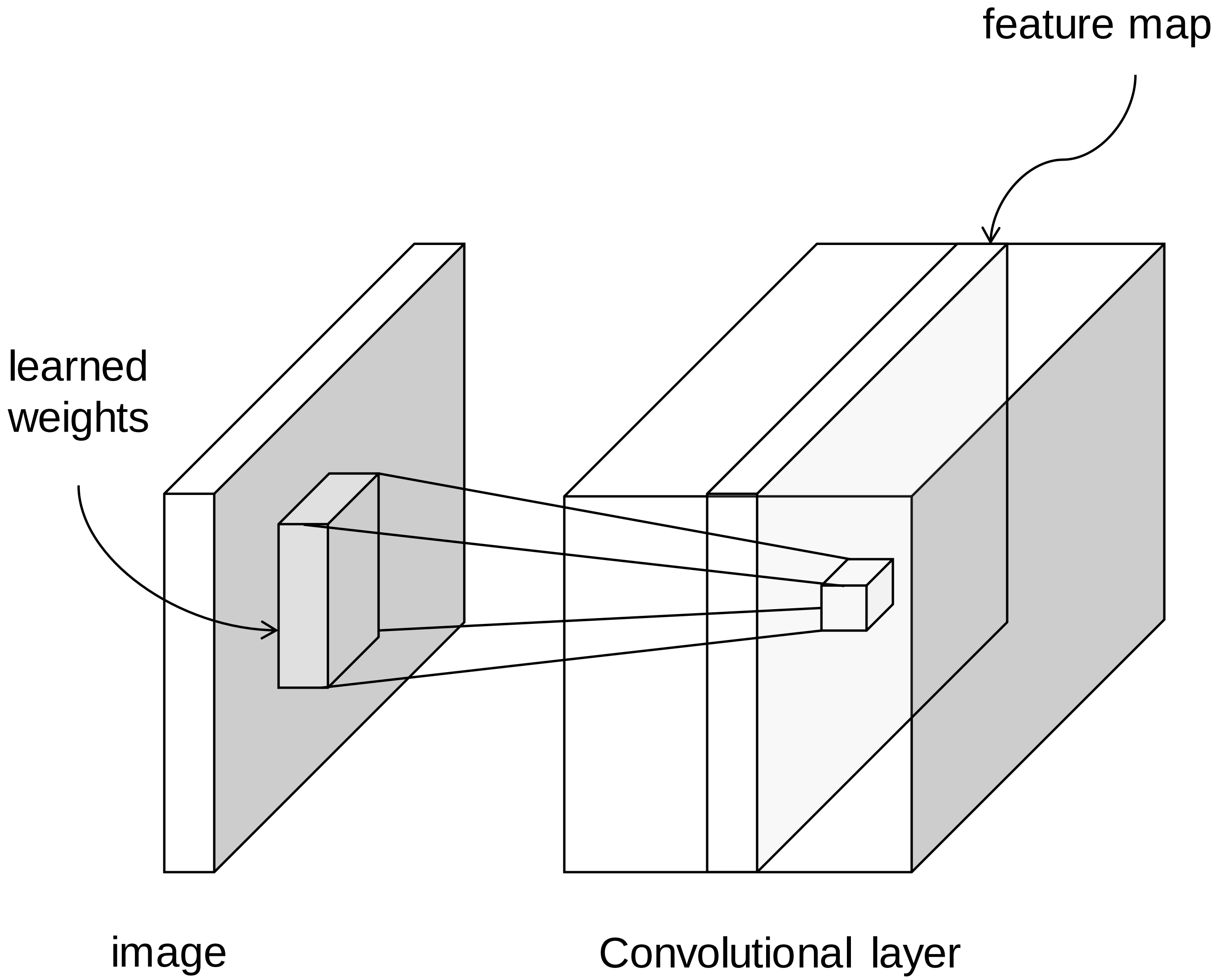
Slide: Lazebnik

From fully connected to convolutional networks



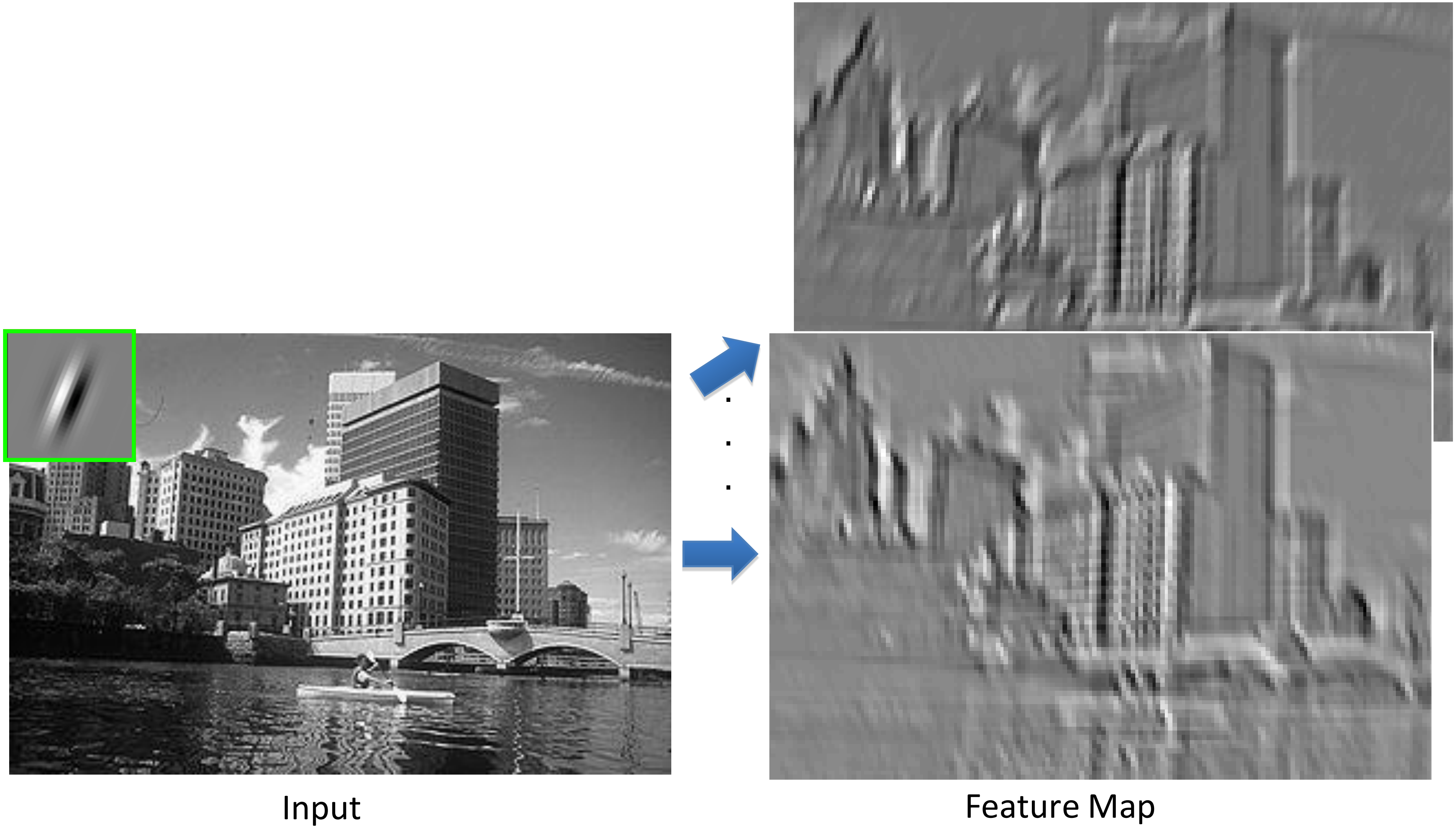
Slide: Lazebnik

From fully connected to convolutional networks



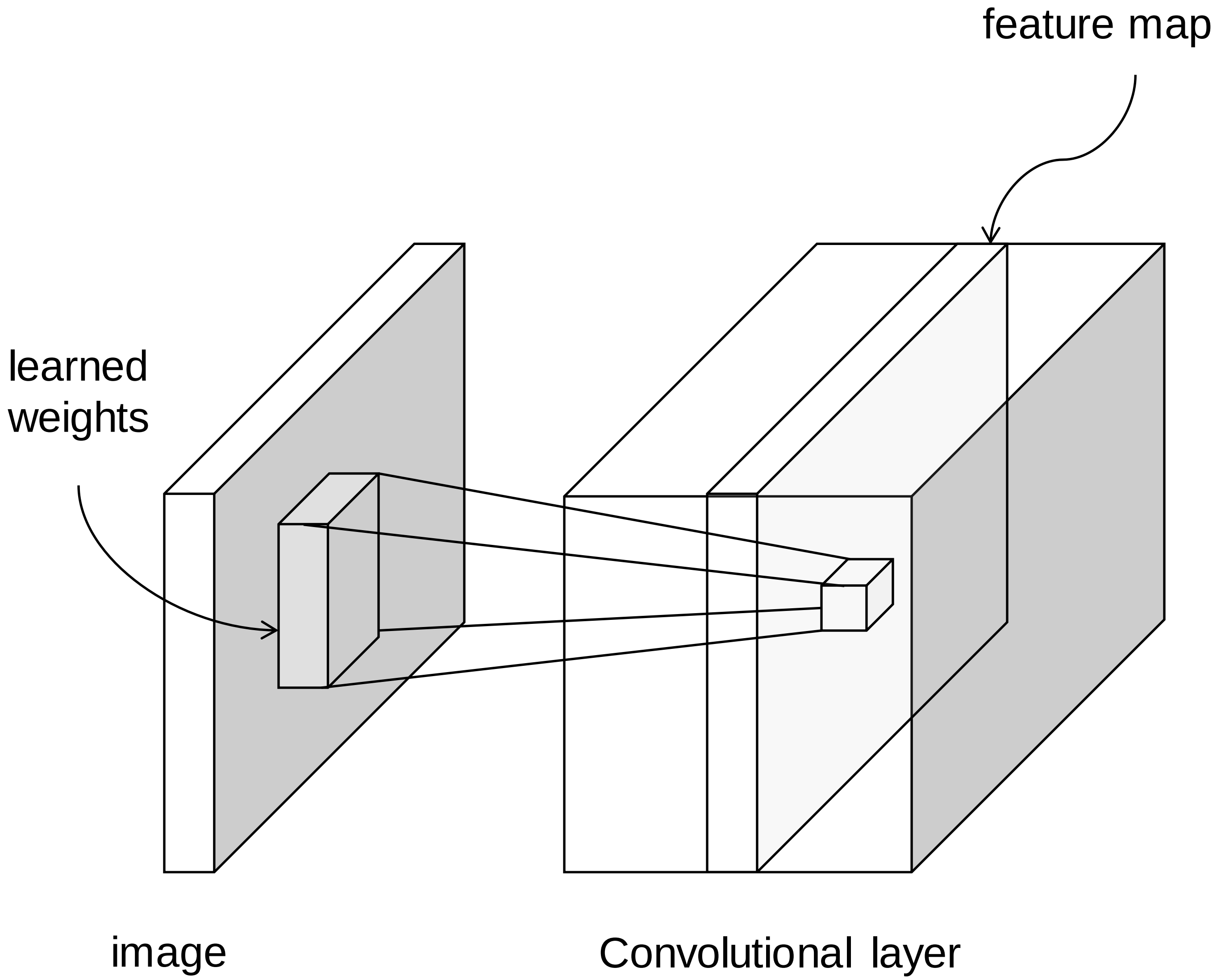
Slide: Lazebnik

Convolution as feature extraction



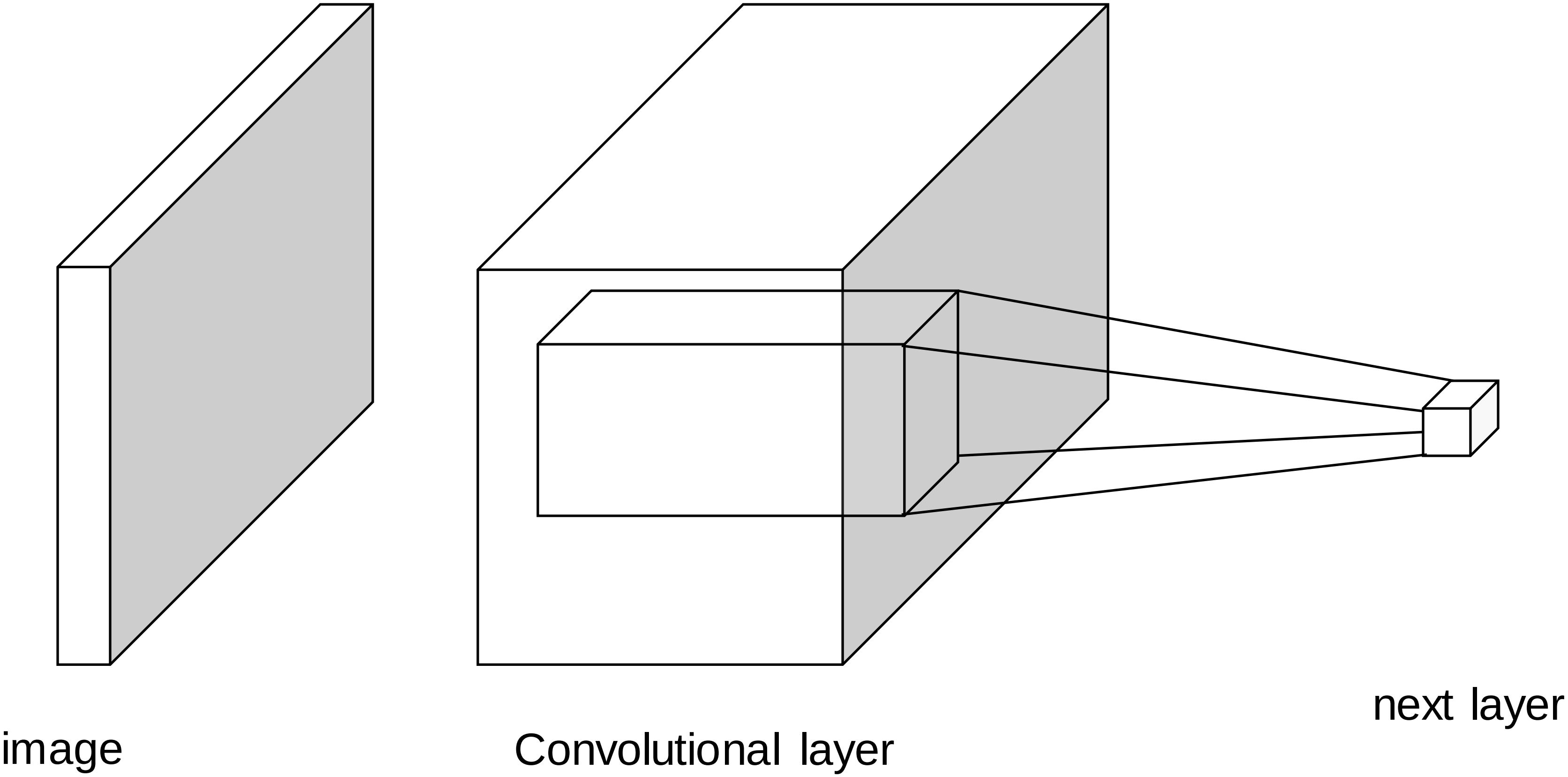
Slide: Lazebnik

From fully connected to convolutional networks



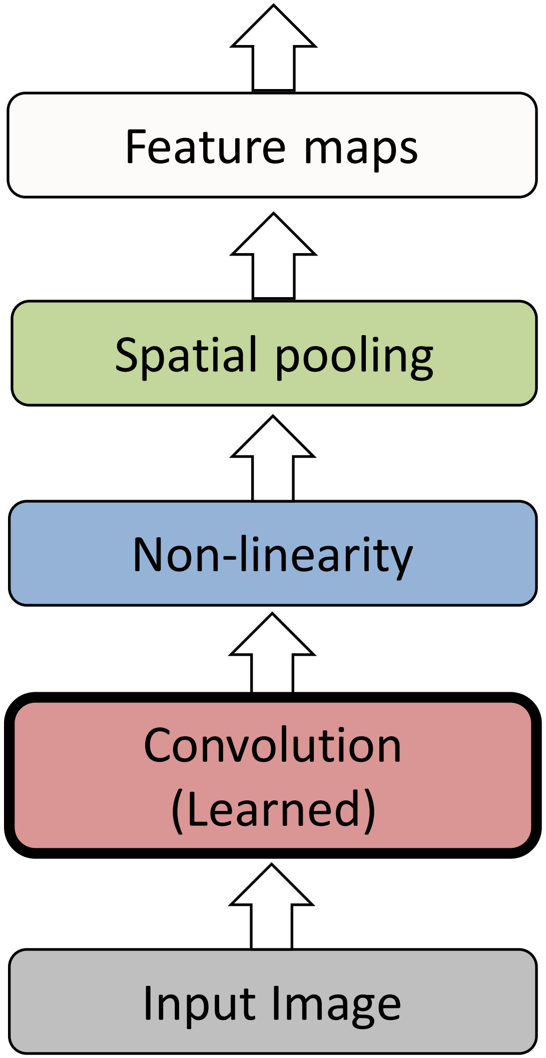
Slide: Lazebnik

From fully connected to convolutional networks



Slide: Lazebnik

Key operations in a CNN



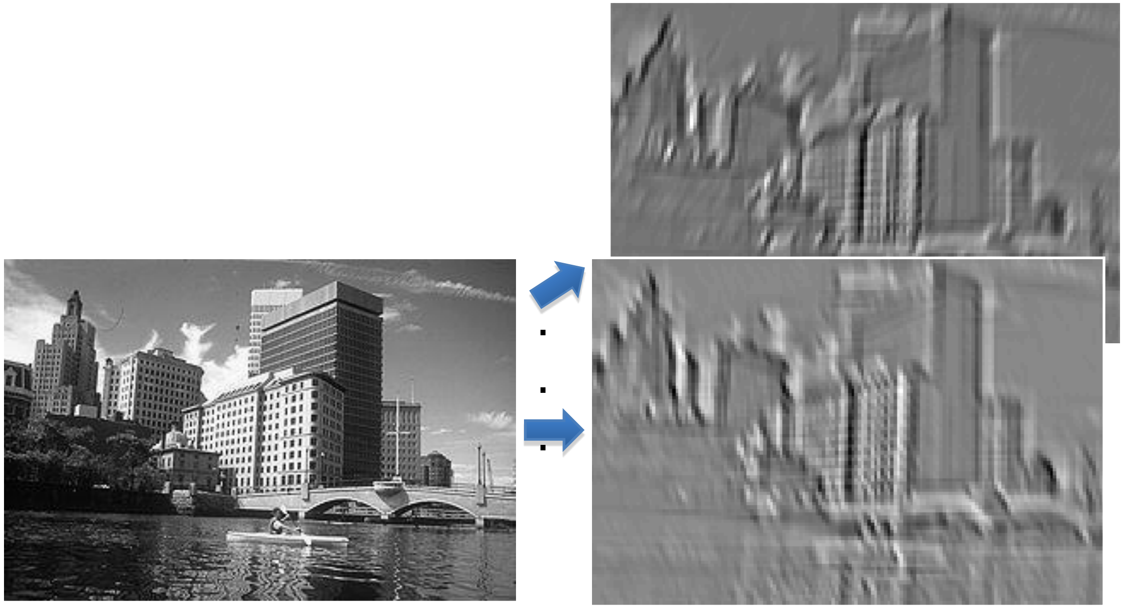
Feature maps

Spatial pooling

Non-linearity

Convolution (Learned)

Input Image



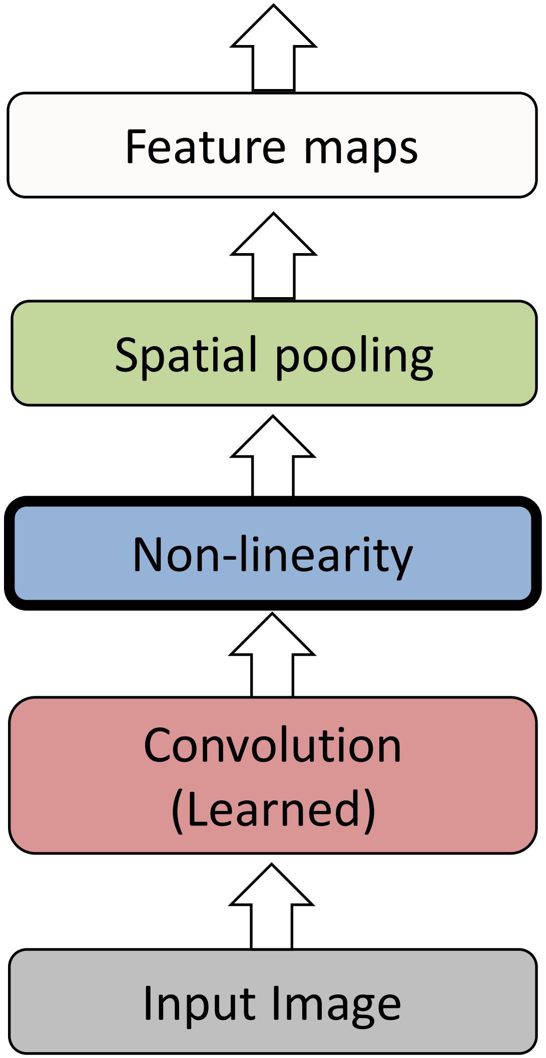
Input

Feature Map

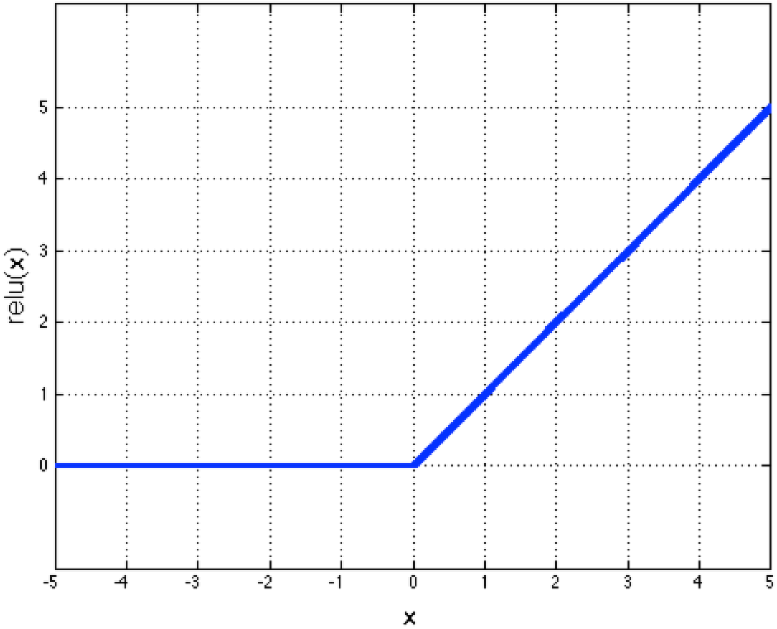
Source: R. Fergus, Y. LeCun

Slide: Lazebnik

Key operations



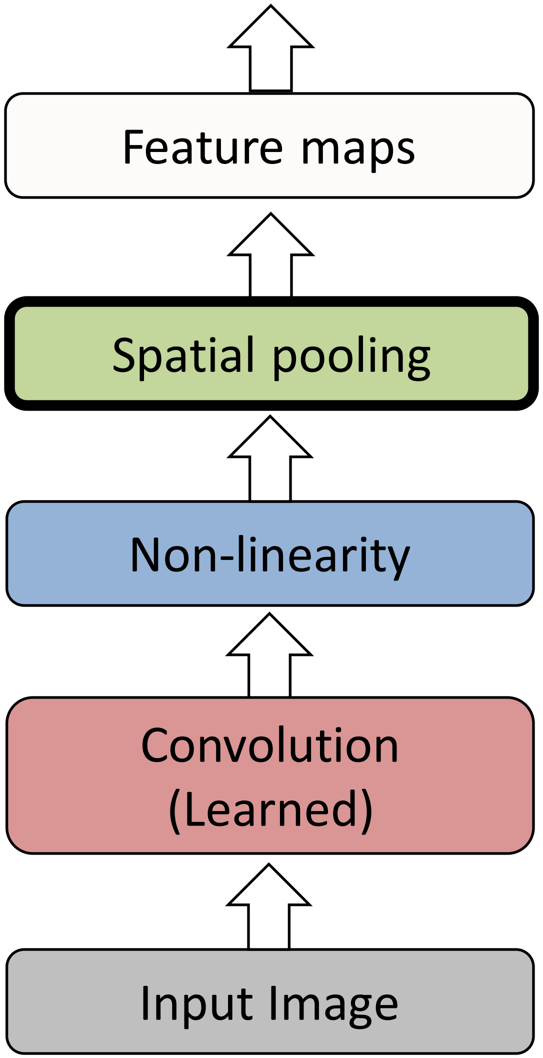
Rectified Linear Unit (ReLU)

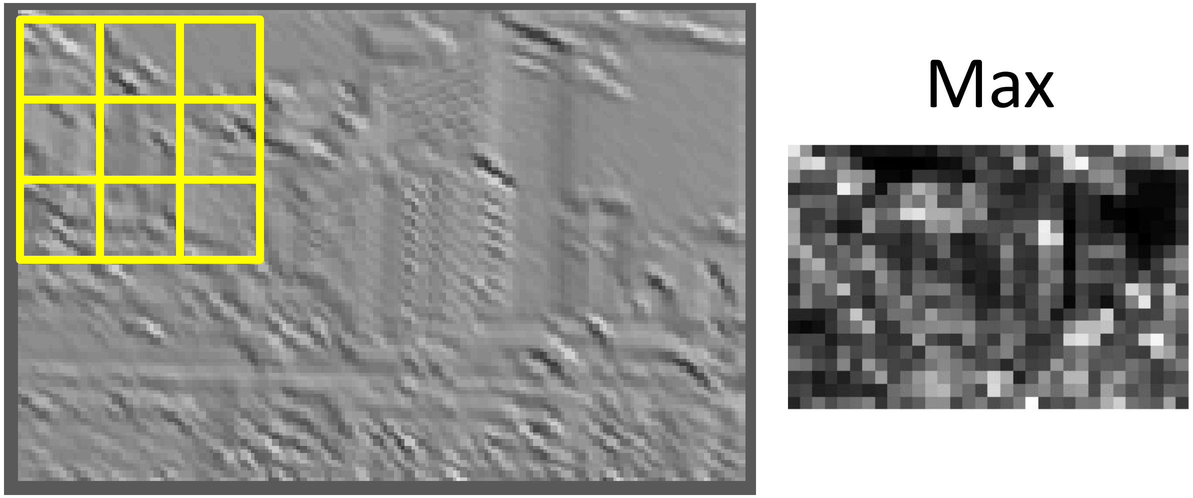


Source: R. Fergus, Y. LeCun

Slide: Lazebnik

Key operations





Source: R. Fergus, Y. LeCun

Slide: Lazebnik

Comparison to Pyramids with SIFT

Lowe [IJCV 2004]

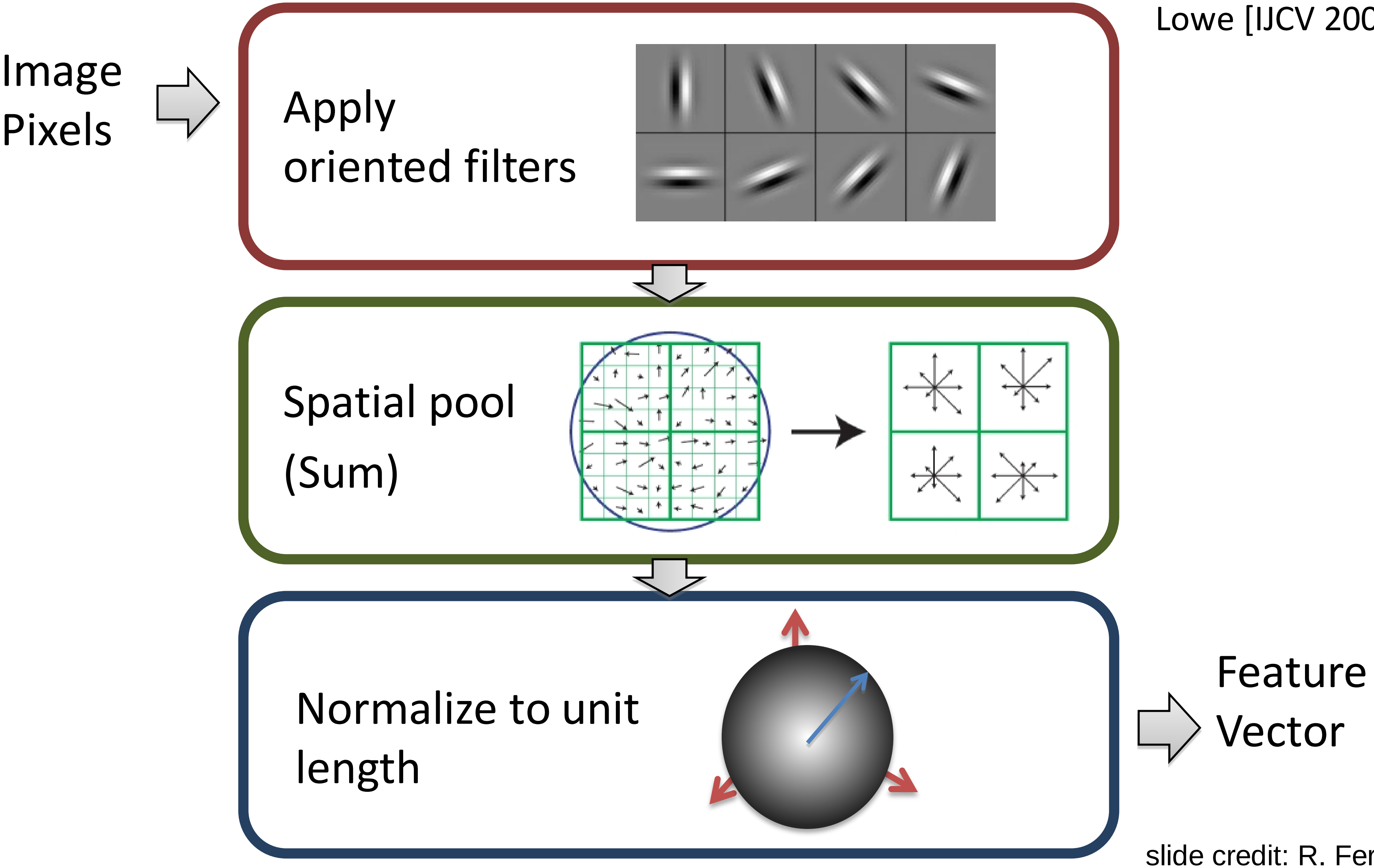


Image Pixels

Apply oriented filters

Spatial pool (Sum)

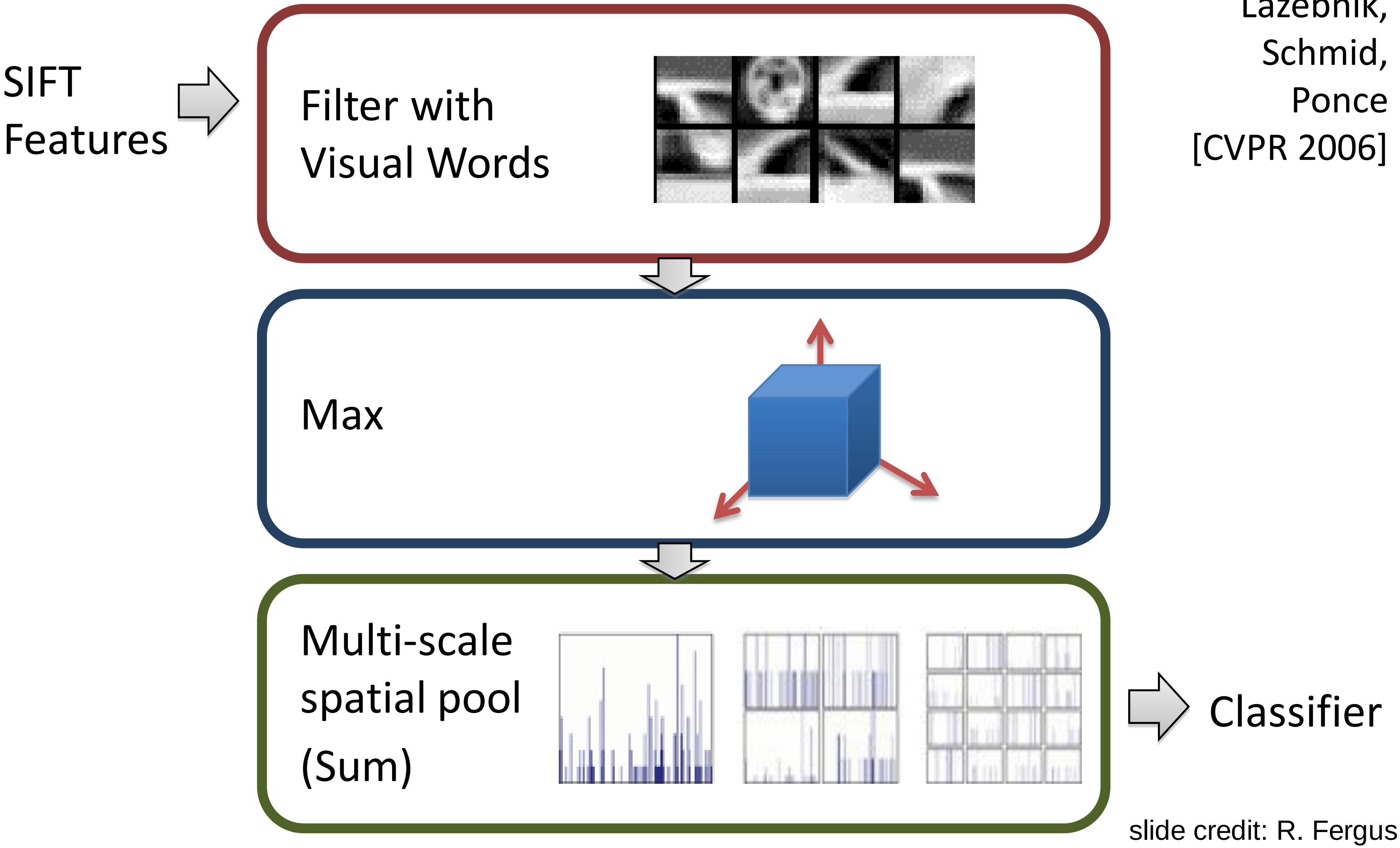
Normalize to unit length

Feature Vector

slide credit: R. Fergus

Comparison to Pyramids with SIFT

Lazebnik,
Schmid,
Ponce
[CVPR 2006]



SIFT Features

Filter with Visual Words

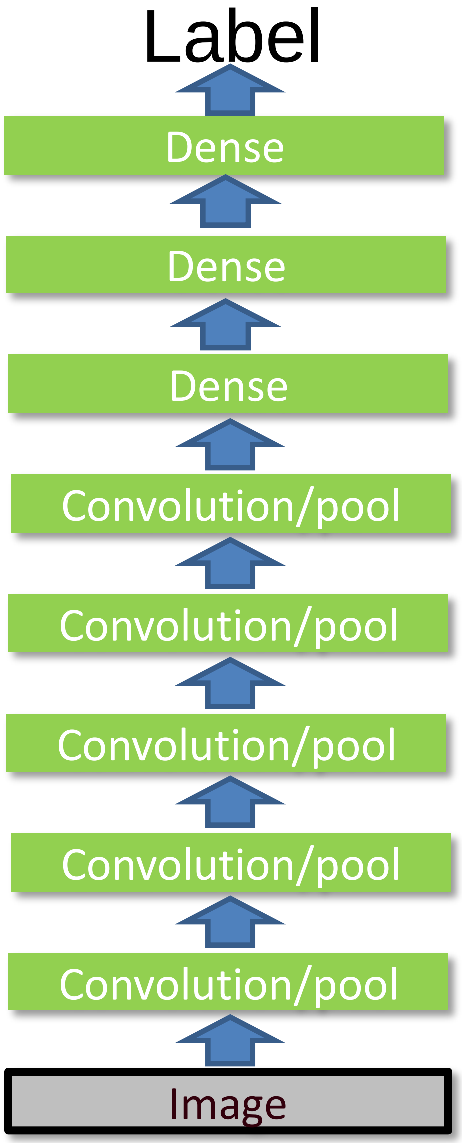
Max

Multi-scale spatial pool (Sum)

Classifier

slide credit: R. Fergus

Key idea: learn features and classifier that work well together ("end-to-end training")



Label

Dense

Dense

Dense

Convolution/pool

Convolution/pool

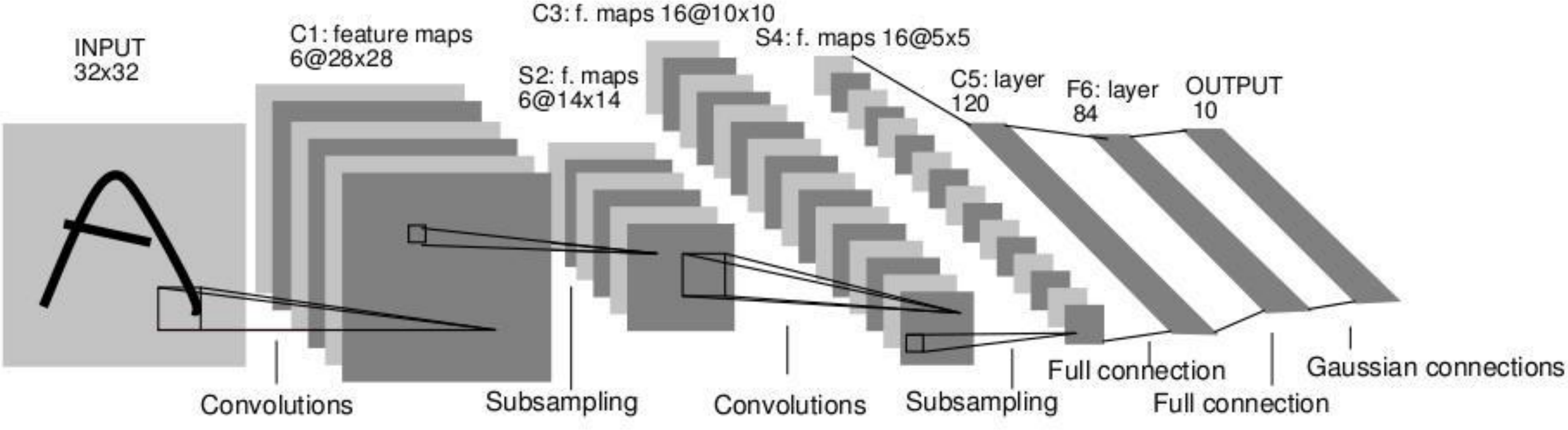
Convolution/pool

Convolution/pool

Convolution/pool

Image

LeNet-5



- Average pooling
- Sigmoid or tanh nonlinearity
- Fully connected layers at the end
- Trained on MNIST digit dataset with 60K training examples

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

Fast forward to the arrival of big visual data…

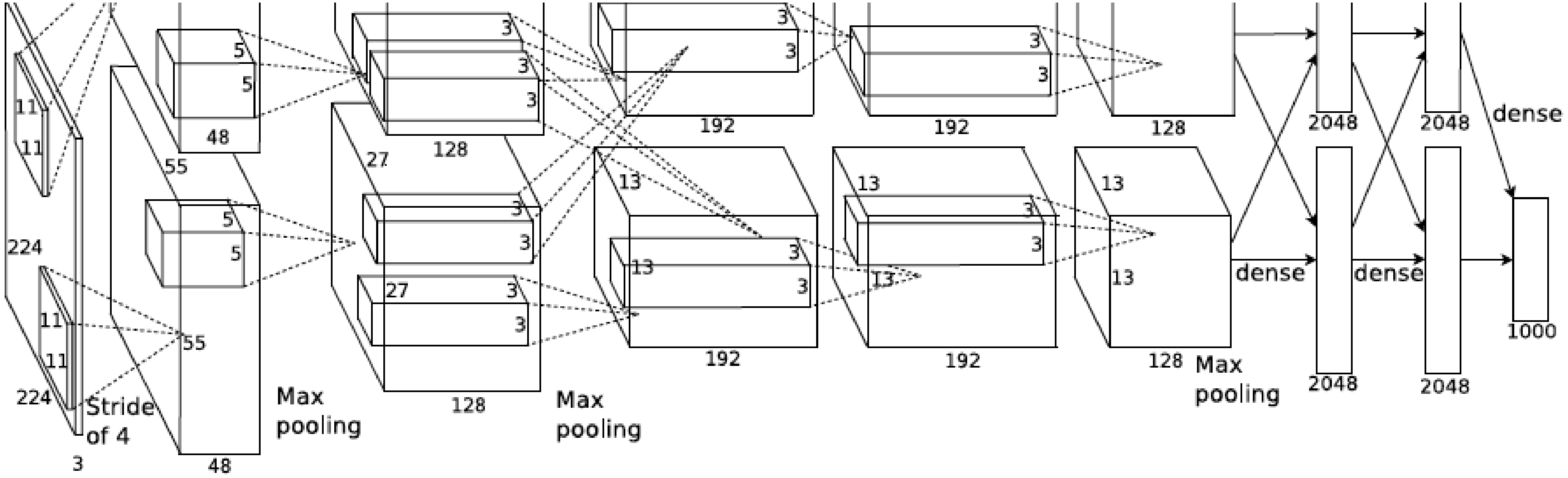
IMAGENET

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/

Slide: Lazebnik

AlexNet: ILSVRC 2012 winner

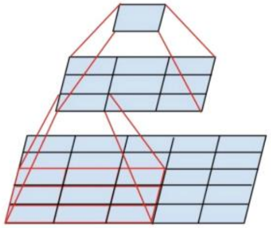


- Similar framework to LeNet but:
  - Max pooling, **ReLU nonlinearity**
  - **More data** and **bigger model** (7 hidden layers, 650K units, 60M params)
  - GPU implementation (**50x speedup** over CPU)
    - Trained on two GPUs for a week
  - Dropout regularization

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

VGGNet

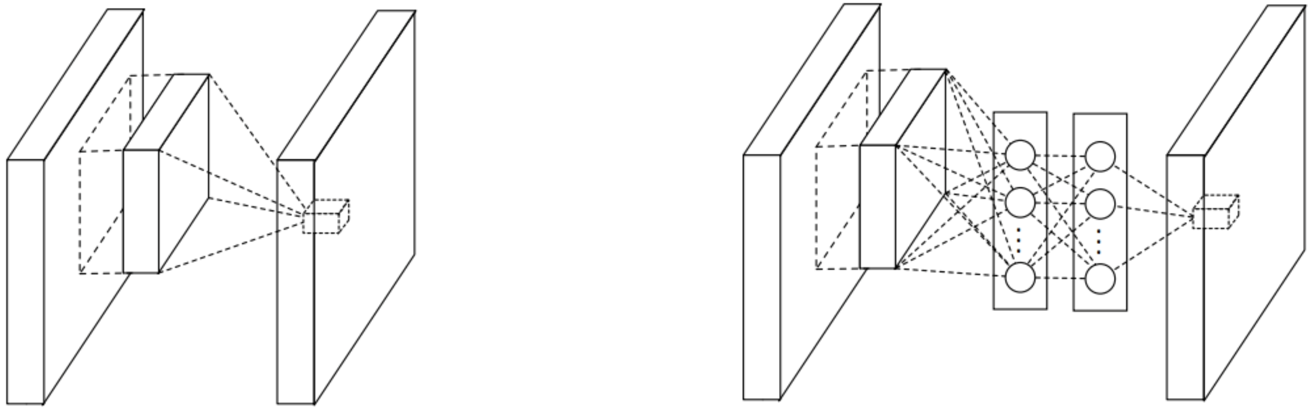
- Sequence of deeper networks trained progressively
- Large receptive fields replaced by successive layers of 3x3 convolutions (with ReLU in between)



  - One 7x7 conv layer with C feature maps needs 49C^2 weights, three 3x3 conv layers need only 27C^2 weights

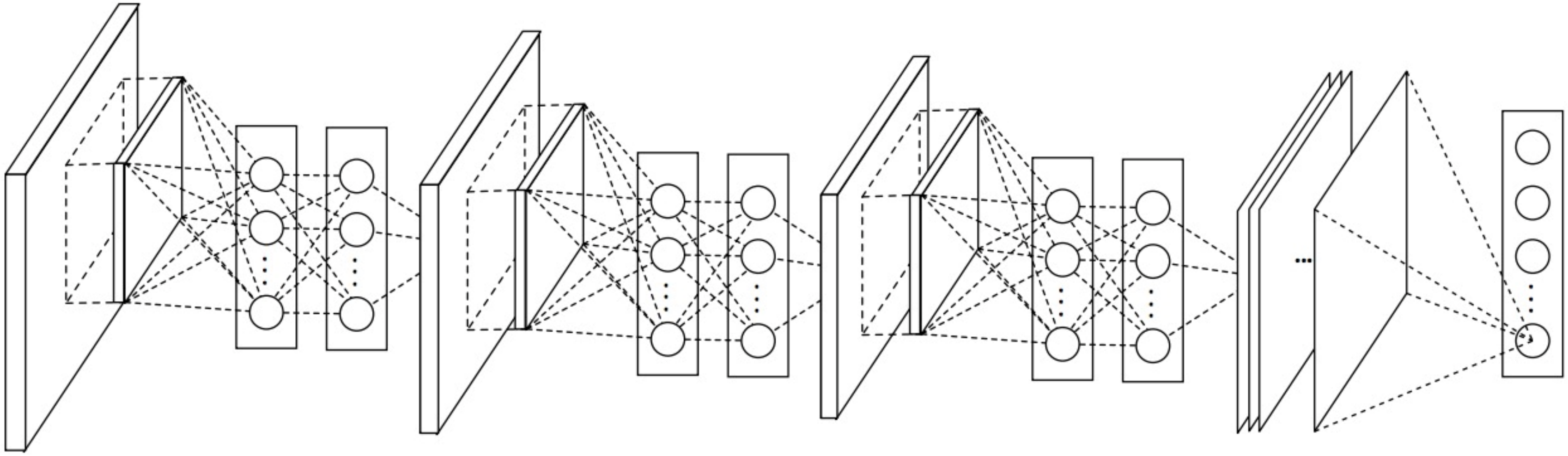
K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

Network in network



(a) Linear convolution layer

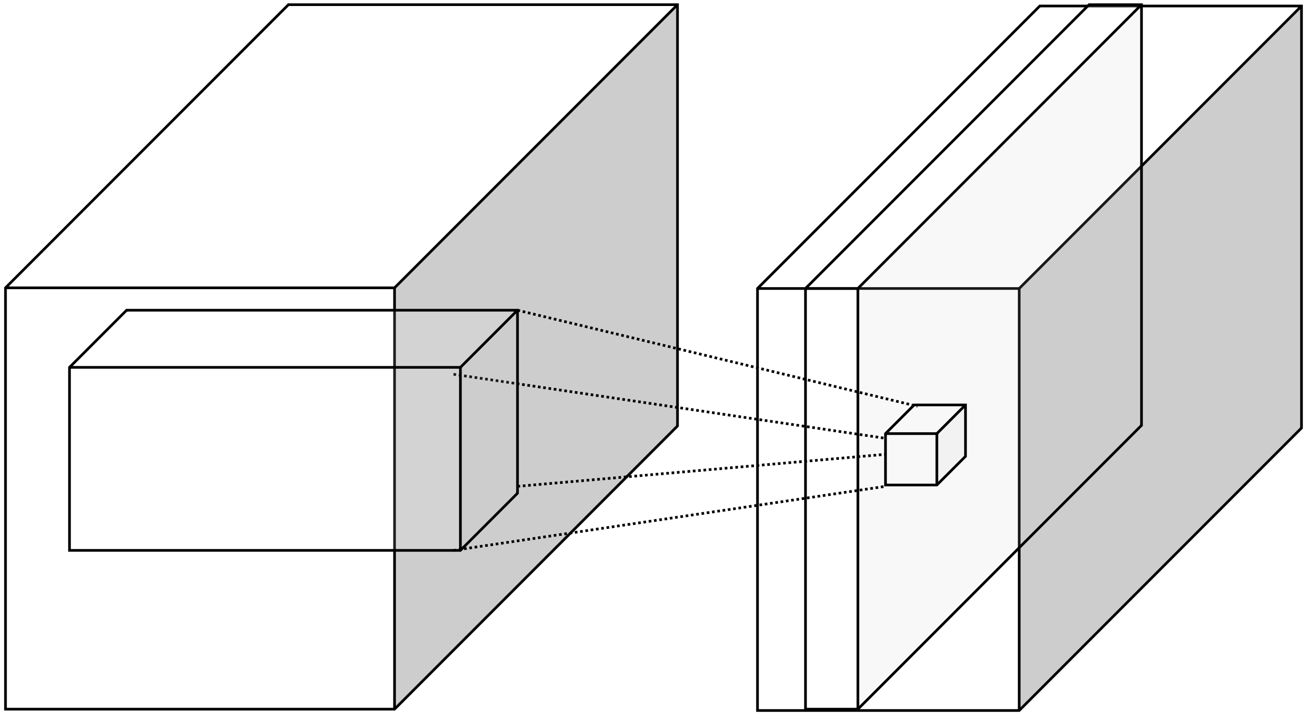
(b) Mlpconv layer



M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014

Slide: Lazebnik

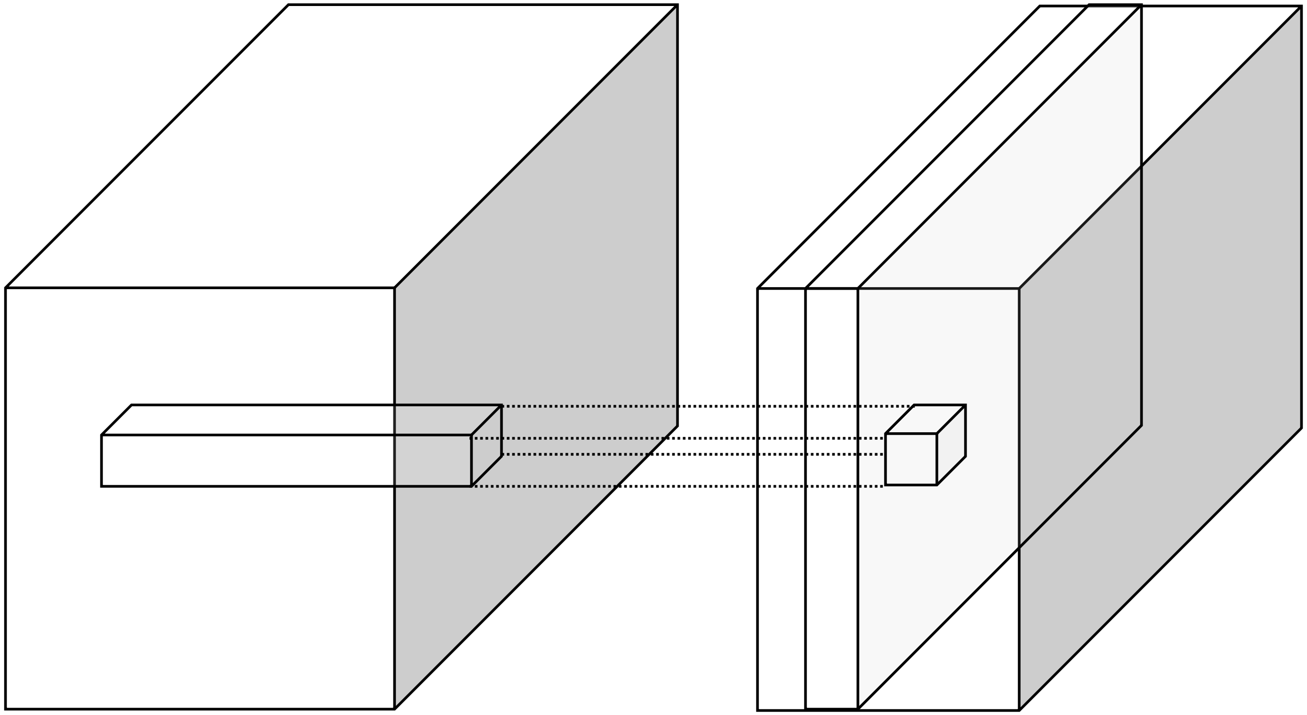
1x1 convolutions



conv layer

Slide: Lazebnik

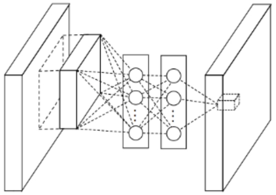
1x1 convolutions

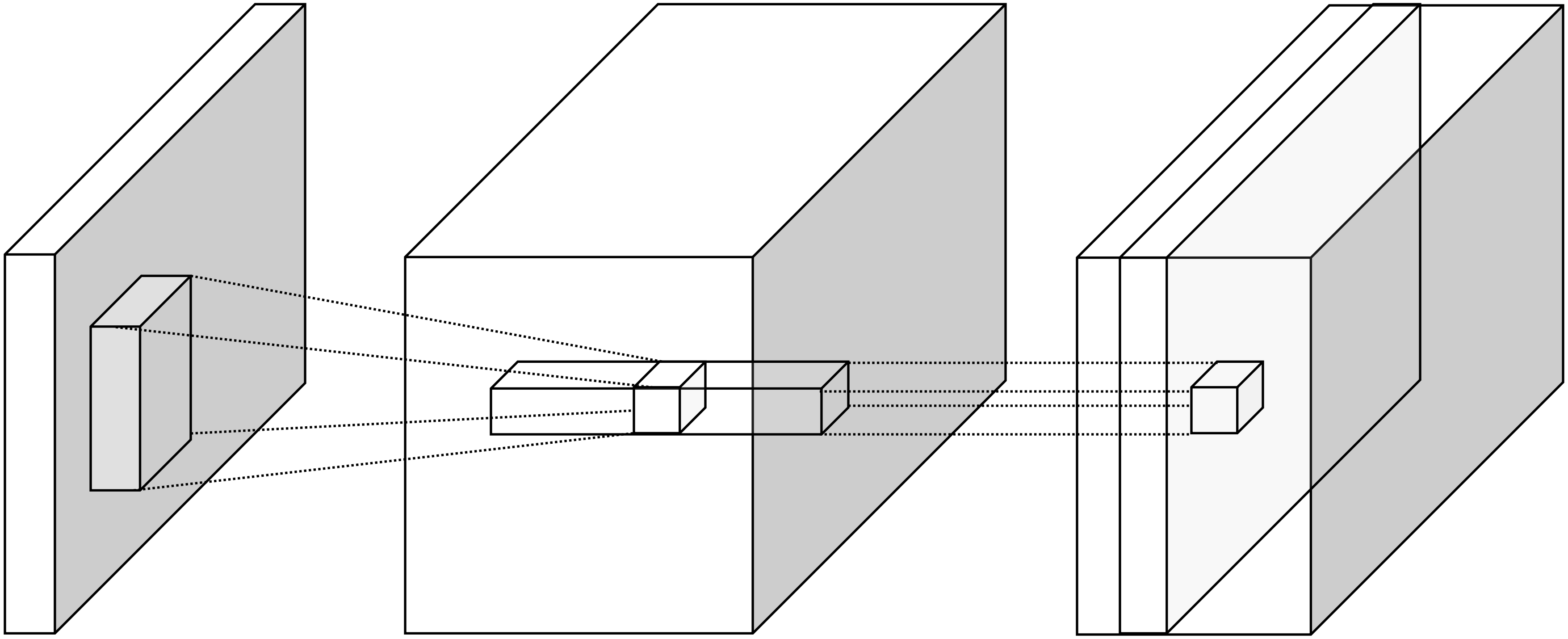


1x1 conv layer

Slide: Lazebnik

1x1 convolutions



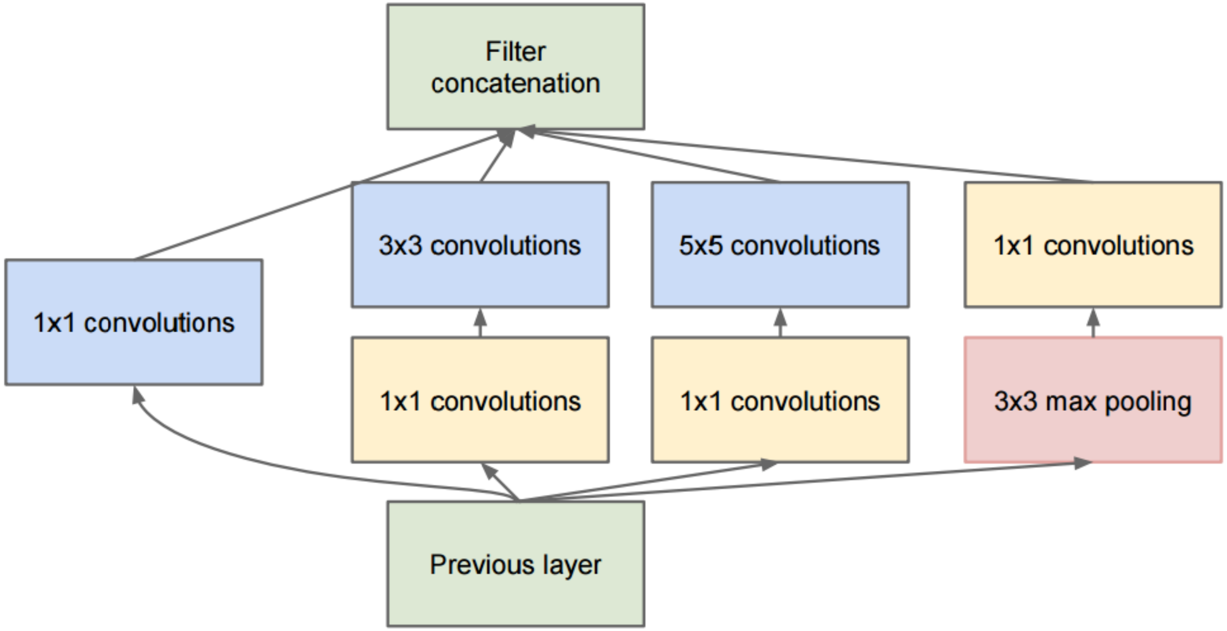


1x1 conv layer

Slide: Lazebnik

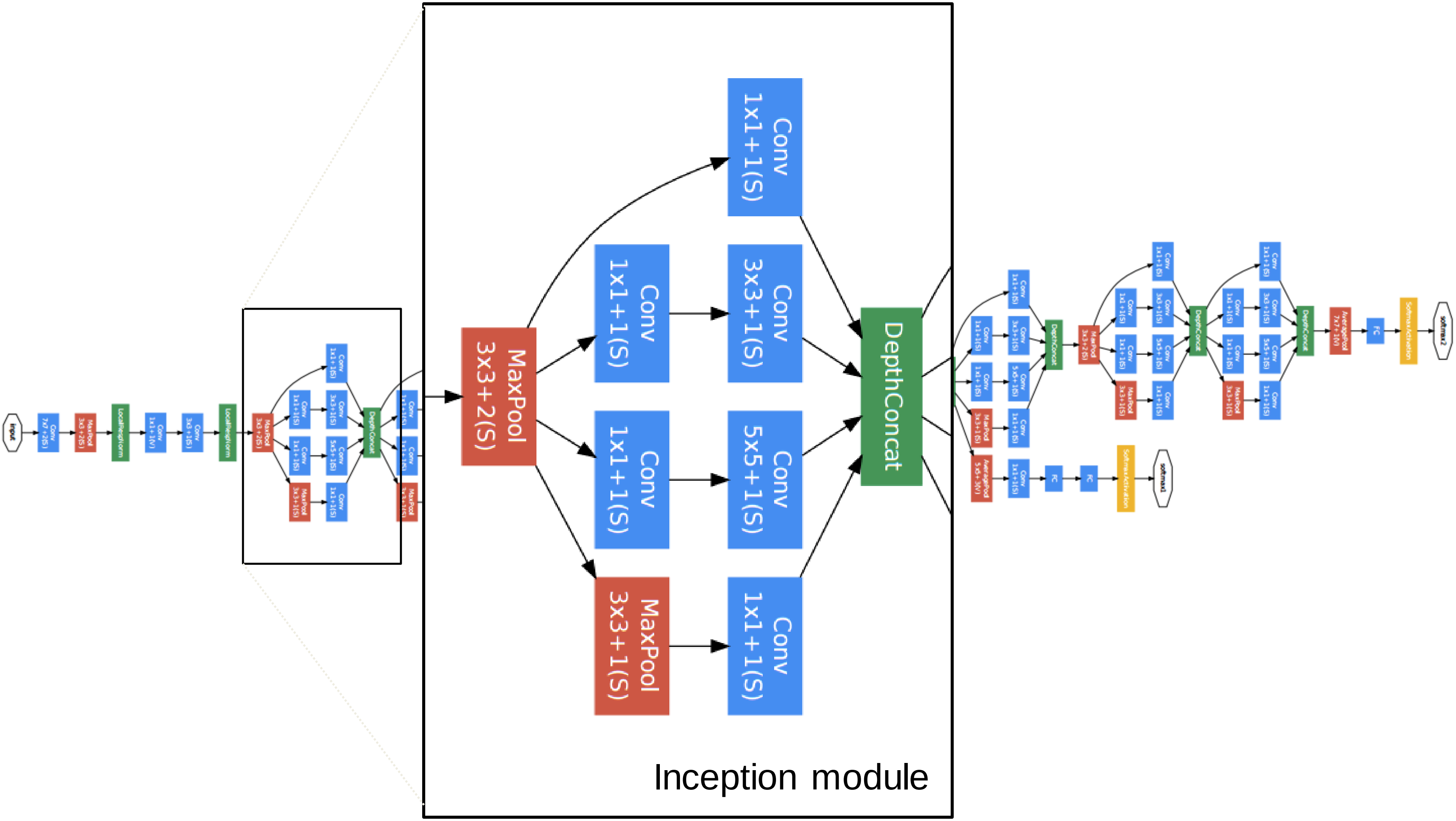
GoogLeNet: Inception module

- Parallel paths with different receptive field sizes and operations to capture sparse patterns of correlations
- 1x1 convolutions for dimensionality reduction before expensive convolutions



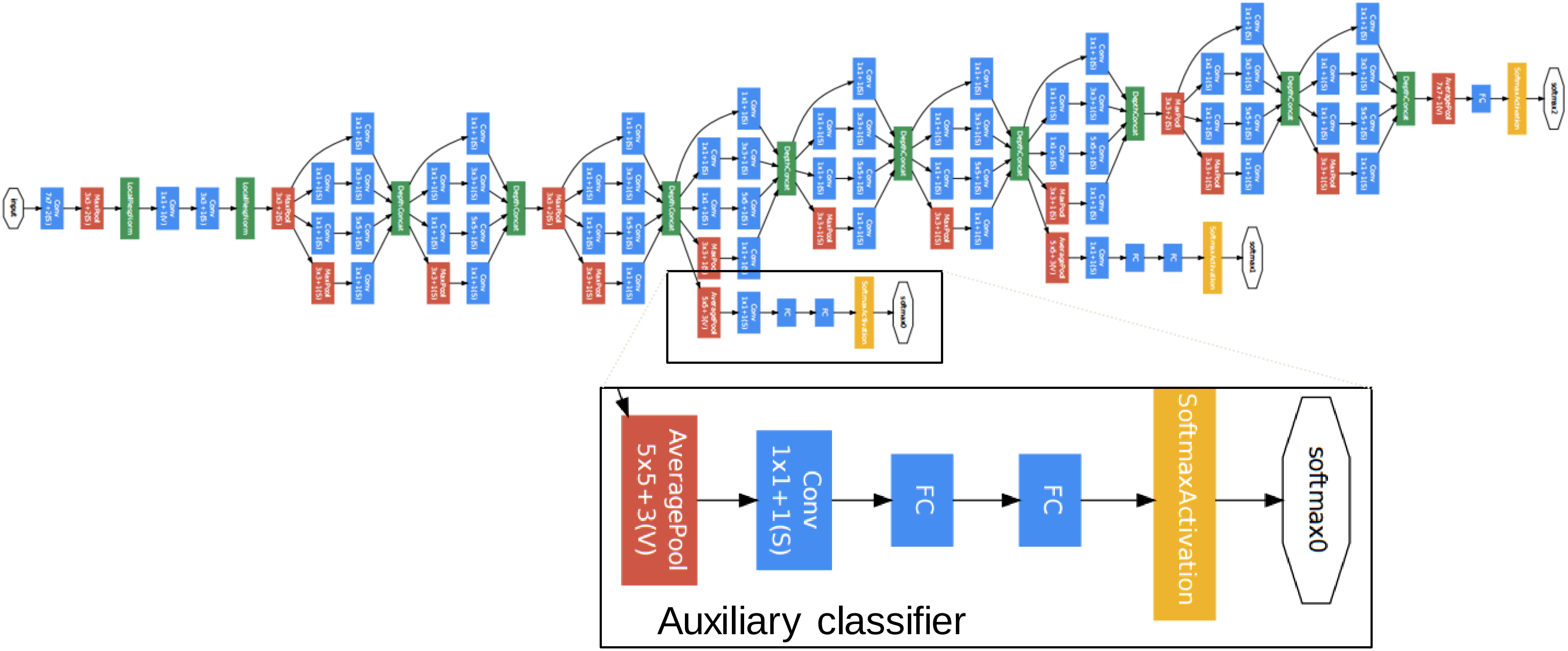
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

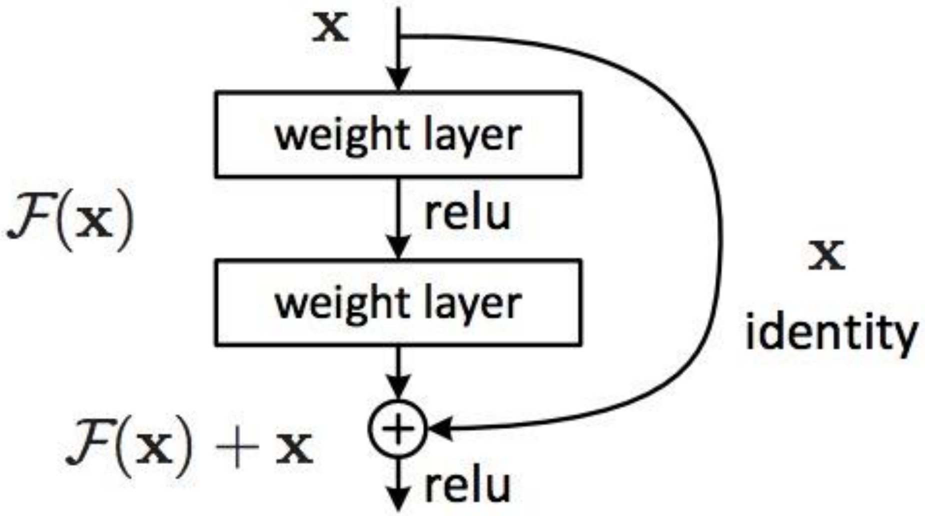
GoogLeNet



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

ResNet: the residual module

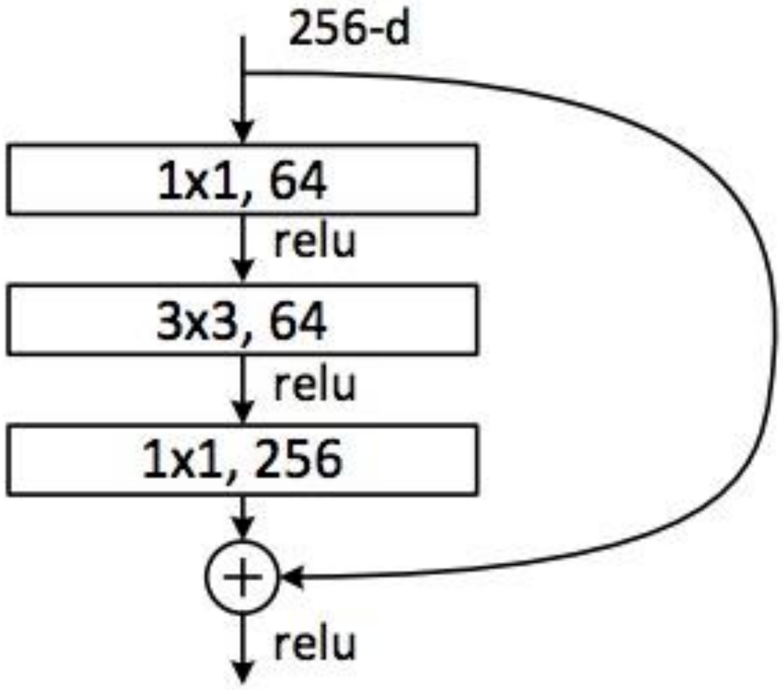
- Introduce *skip* or *shortcut* connections (existing before in various forms in literature)
- Make it easy for network layers to represent the identity mapping
- For some reason, need to skip at least two layers



Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

ResNet

Deeper residual module (bottleneck)

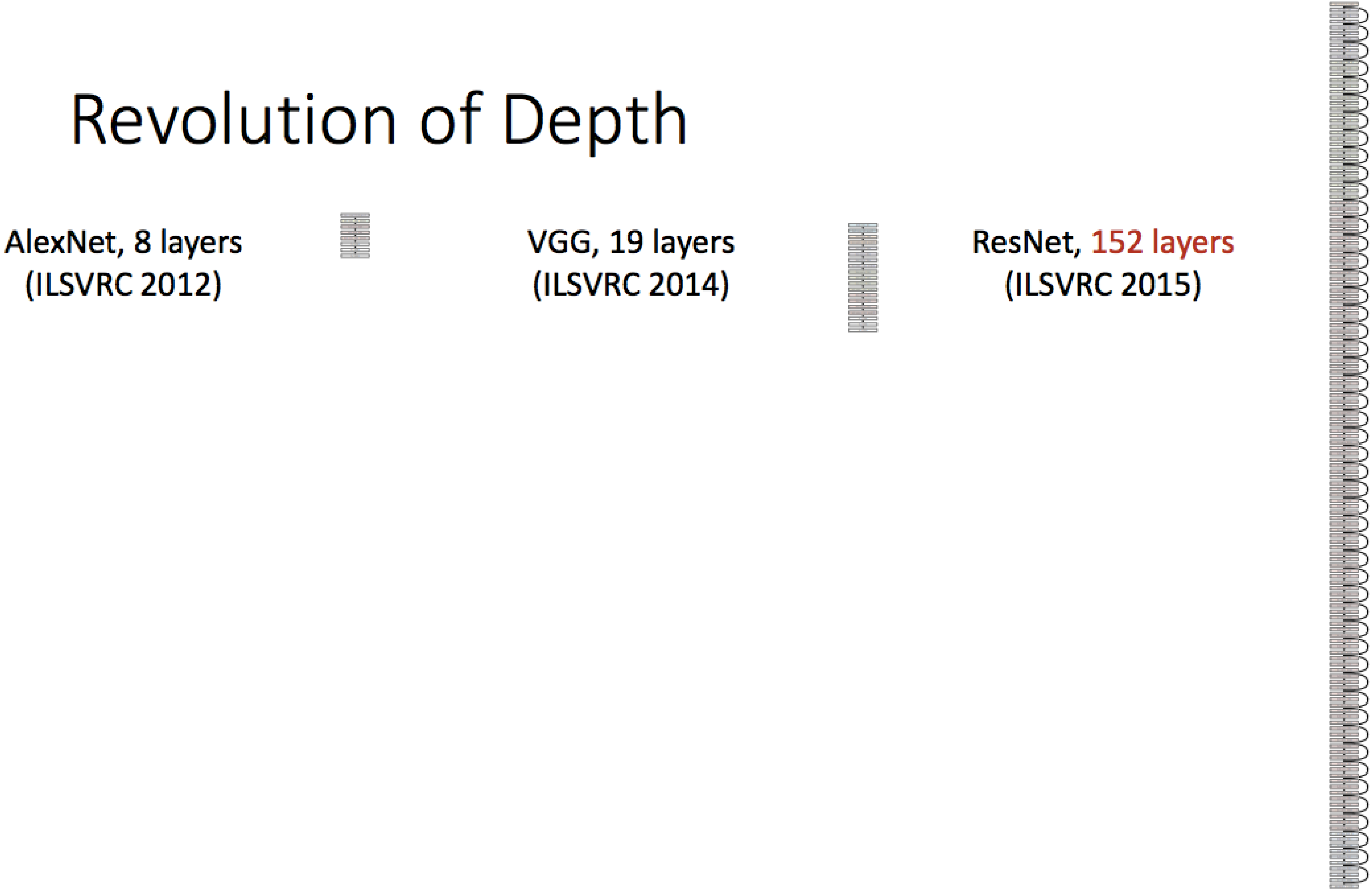


- Directly performing 3x3 convolutions with 256 feature maps at input and output: 256 x 256 x 3 x 3 ~ 600K operations
- Using 1x1 convolutions to reduce 256 to 64 feature maps, followed by 3x3 convolutions, followed by 1x1 convolutions to expand back to 256 maps:
  256 x 64 x 1 x 1 ~ 16K
  64 x 64 x 3 x 3 ~ 36K
  64 x 256 x 1 x 1 ~ 16K
  Total: ~70K

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

Slide: Lazebnik

ResNet: going real deep



Revolution of Depth

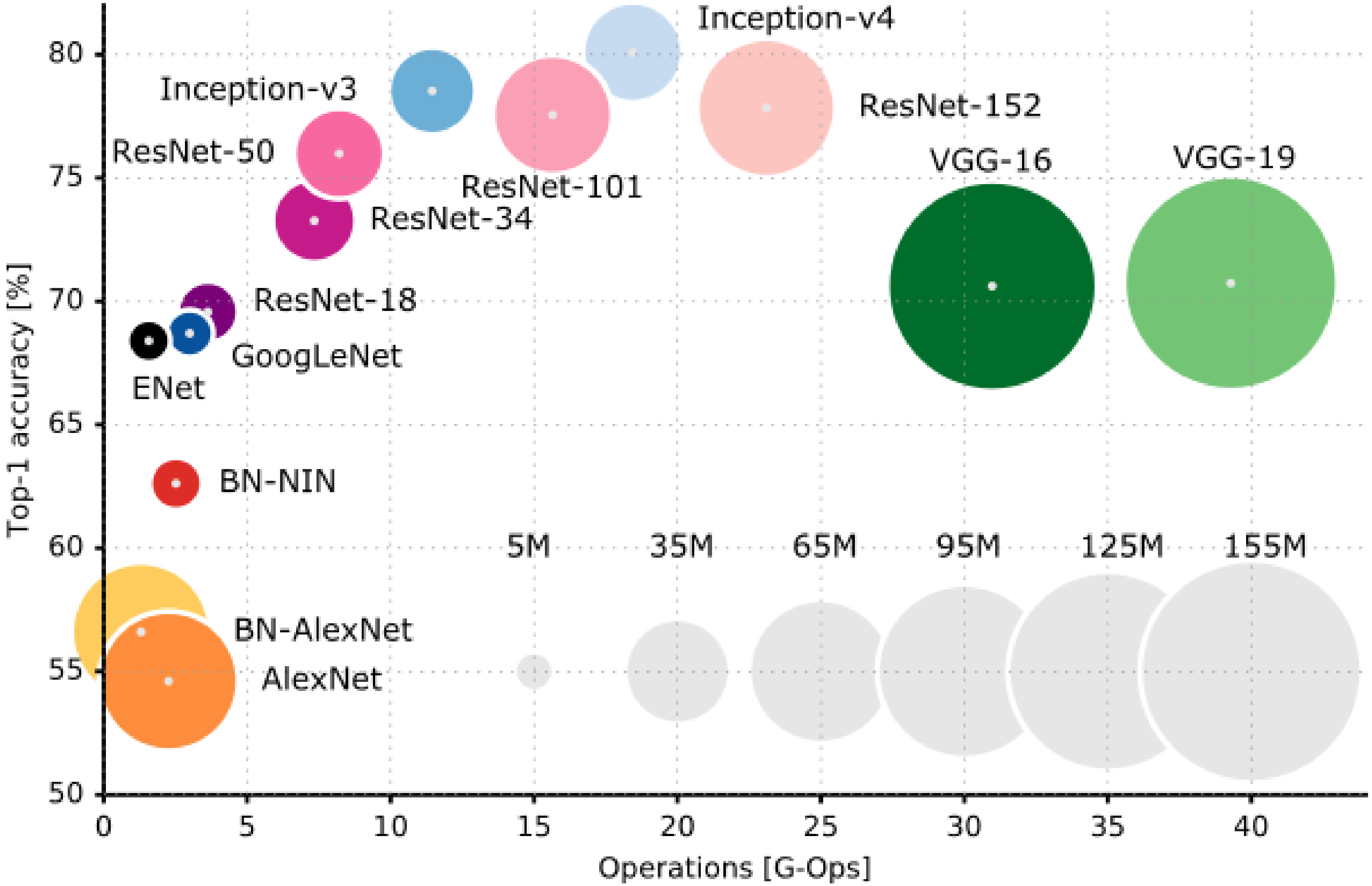
AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016

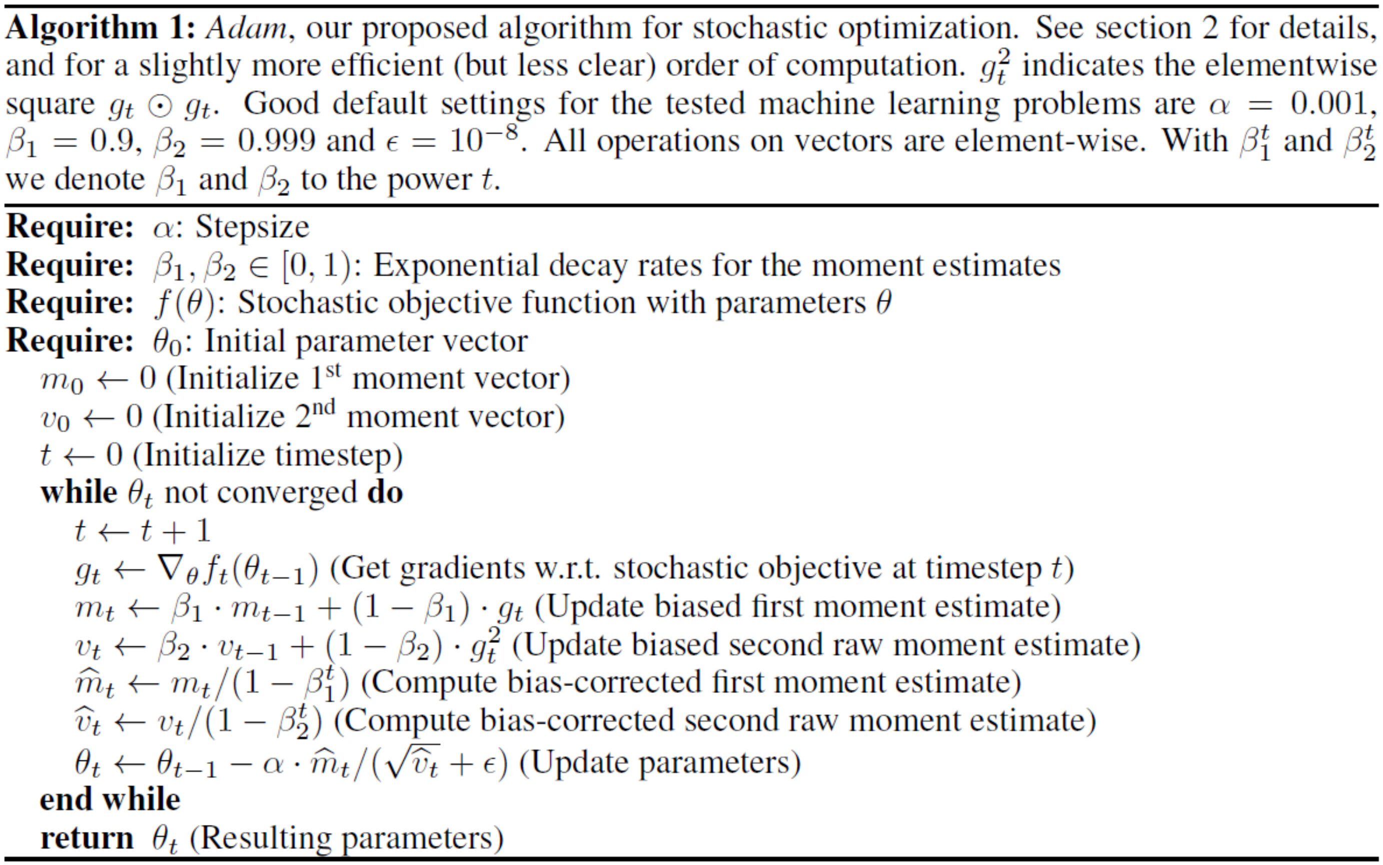
Bigger not better: innovations typically reduce parameters, despite deeper nets



Key ideas of CNN Architectures

- Convolutional layers
  - Same local functions evaluated everywhere → much fewer parameters
- Pooling
  - Larger receptive field and translational invariance
- ReLU: maintain a gradient signal over large portion of domain
- Limit parameters
  - Sequence of 3x3 filters instead of large filters (also encodes that local pixels are more relevant)
  - 1x1 convs to reduce feature dimensions
- Skip network
  - Prevents having to maintain early layers (just add residual)
  - Acts as ensemble

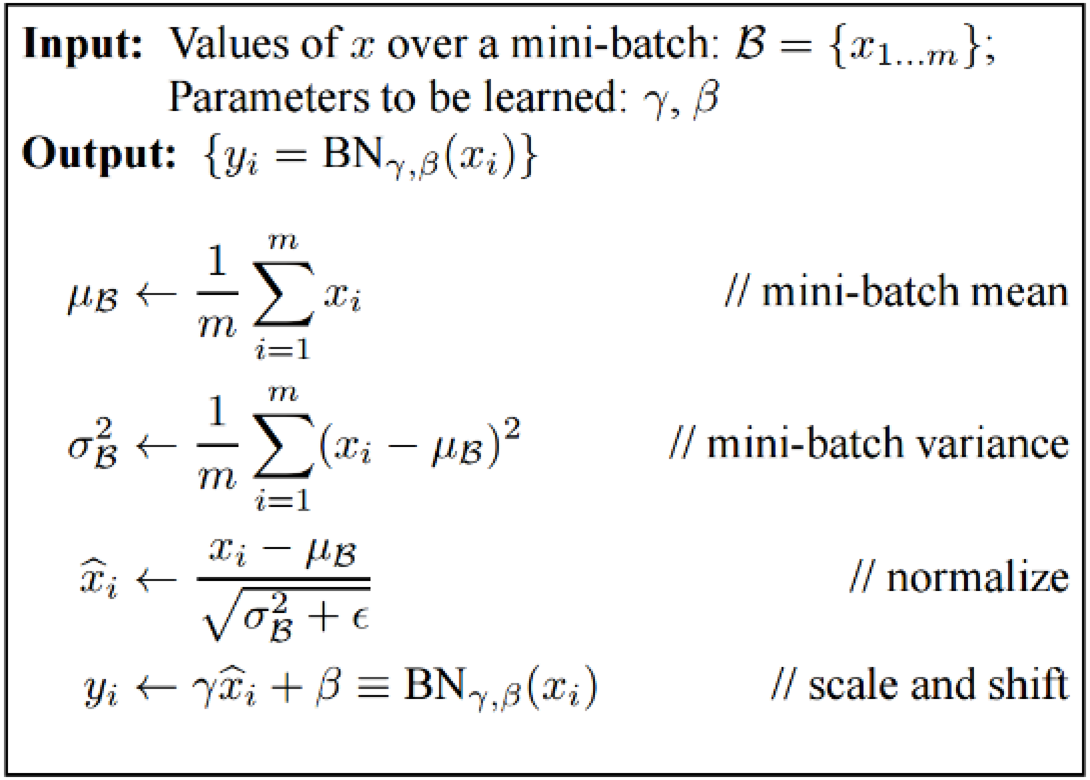
Optimization



Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t/(\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Batch Normalization



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

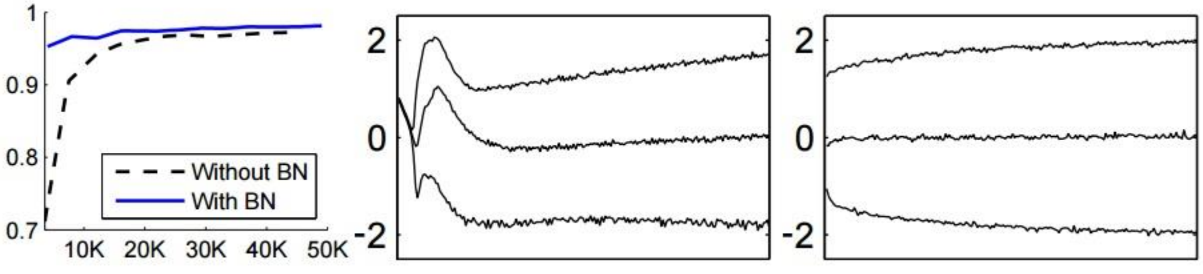
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



(a) (b) Without BN (c) With BN

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

Key ideas of optimization

- Stochastic gradient descent (SGD) in batches
  - Batch size 128 or 256 recommended
  - Use ADAM for gradient/momentum

- Normalize inputs/features (similar idea to whitening)
  - Batchnorm normalizes inputs to each layer by estimate (e.g. moving average) of mean/std

- Crazy optimization problem (so many local minima), but
  - Model capacity is larger than needed to help ensure that important patterns are discovered
  - Many solutions are similarly good (e.g. can permute layers without effect)

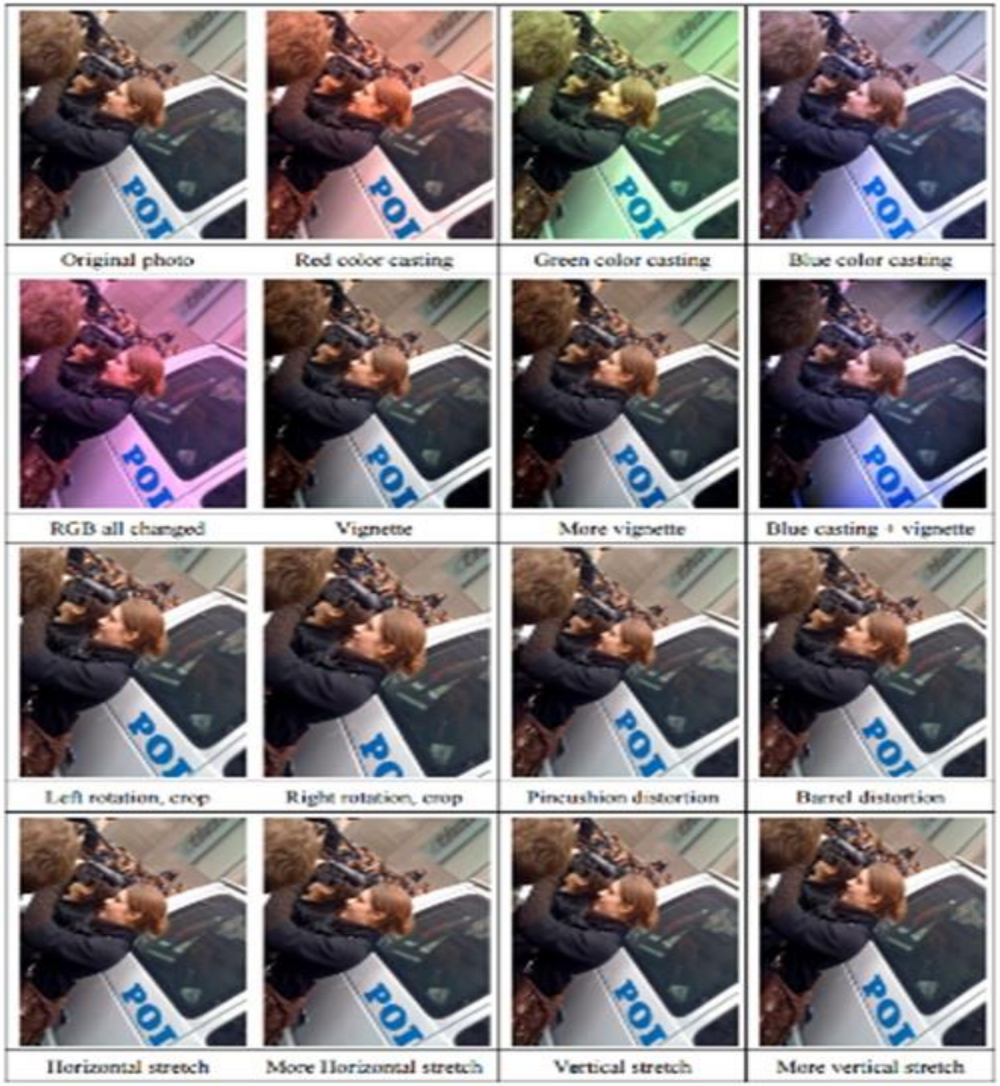
Good discussion post on local minima

Slide: Jiabin Huang

Data Augmentation (Jittering)

- Create *virtual* training samples
  - Horizontal flip
  - Random crop
  - Color casting
  - Geometric distortion

- Idea goes back to Pomerleau 1995 at least (neural net for car driving)



Original photo

Red color casting

Green color casting

Blue color casting

RGB all changed

Vignette

More vignette

Blue casting + vignette

Left rotation, crop

Right rotation, crop

Pincushion distortion

Barrel distortion

Horizontal stretch

More Horizontal stretch

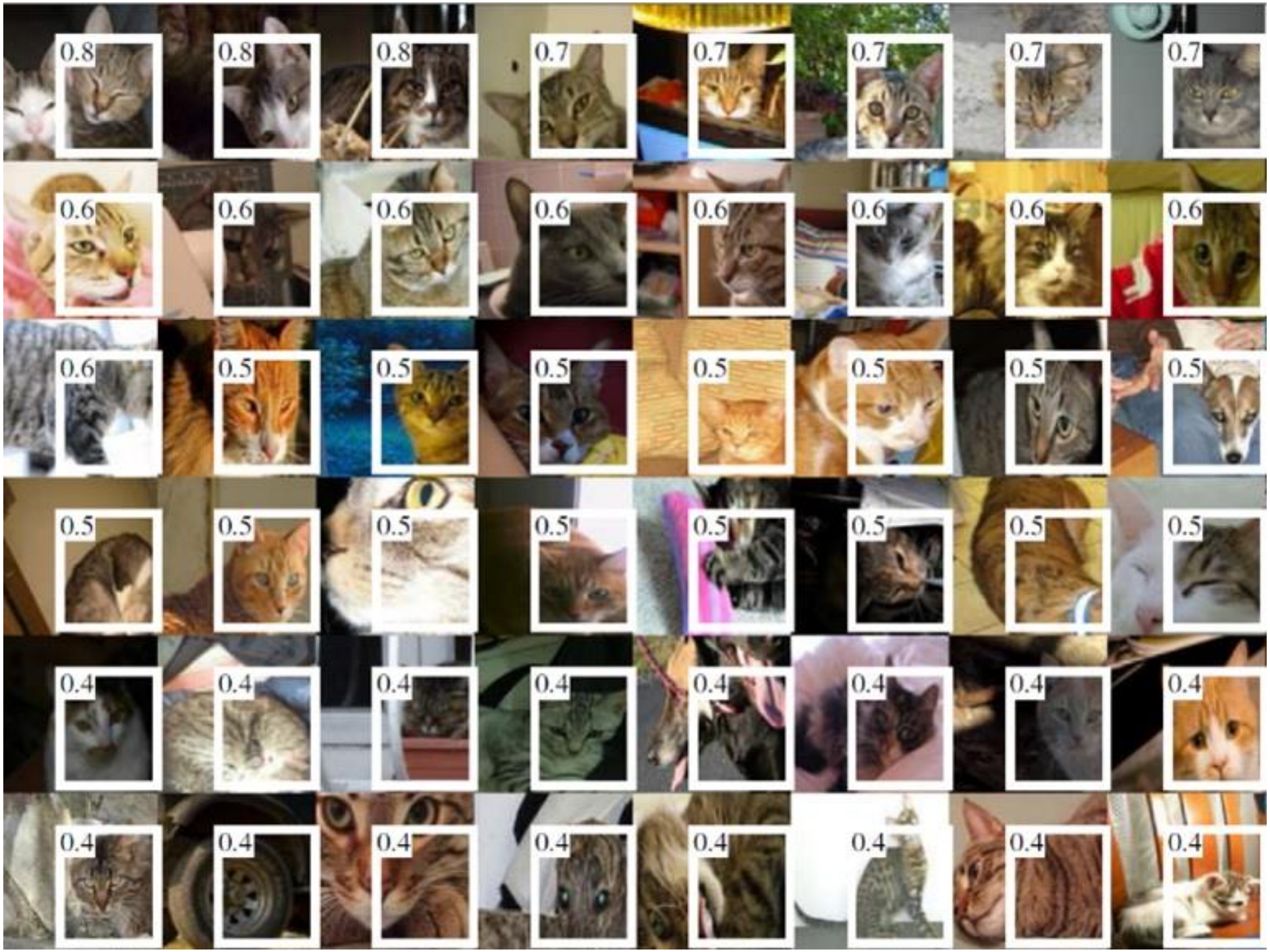
Vertical stretch

More vertical stretch

Deep Image [Wu et al. 2015]

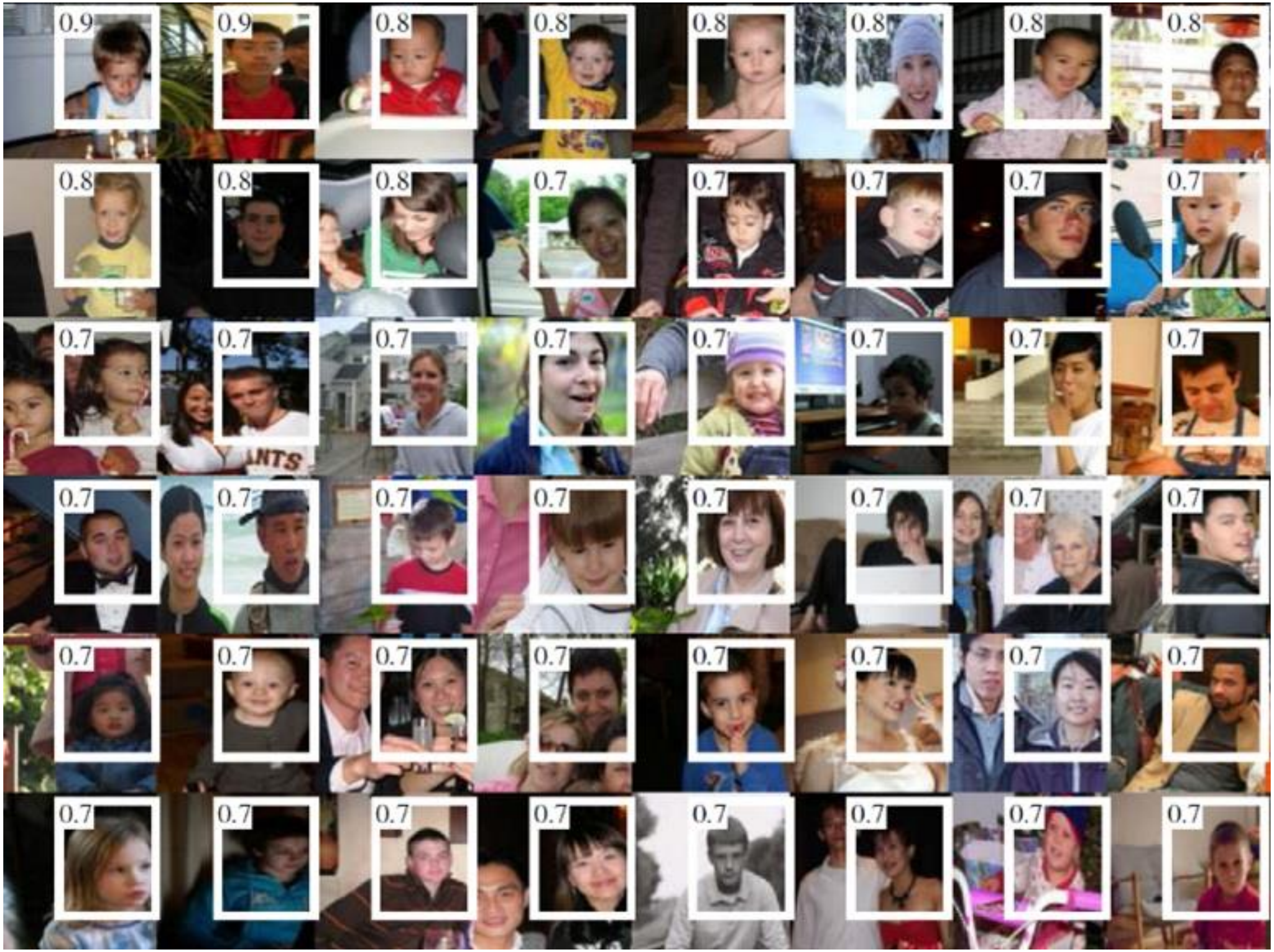
What does the CNN learn?

Individual Neuron Activation



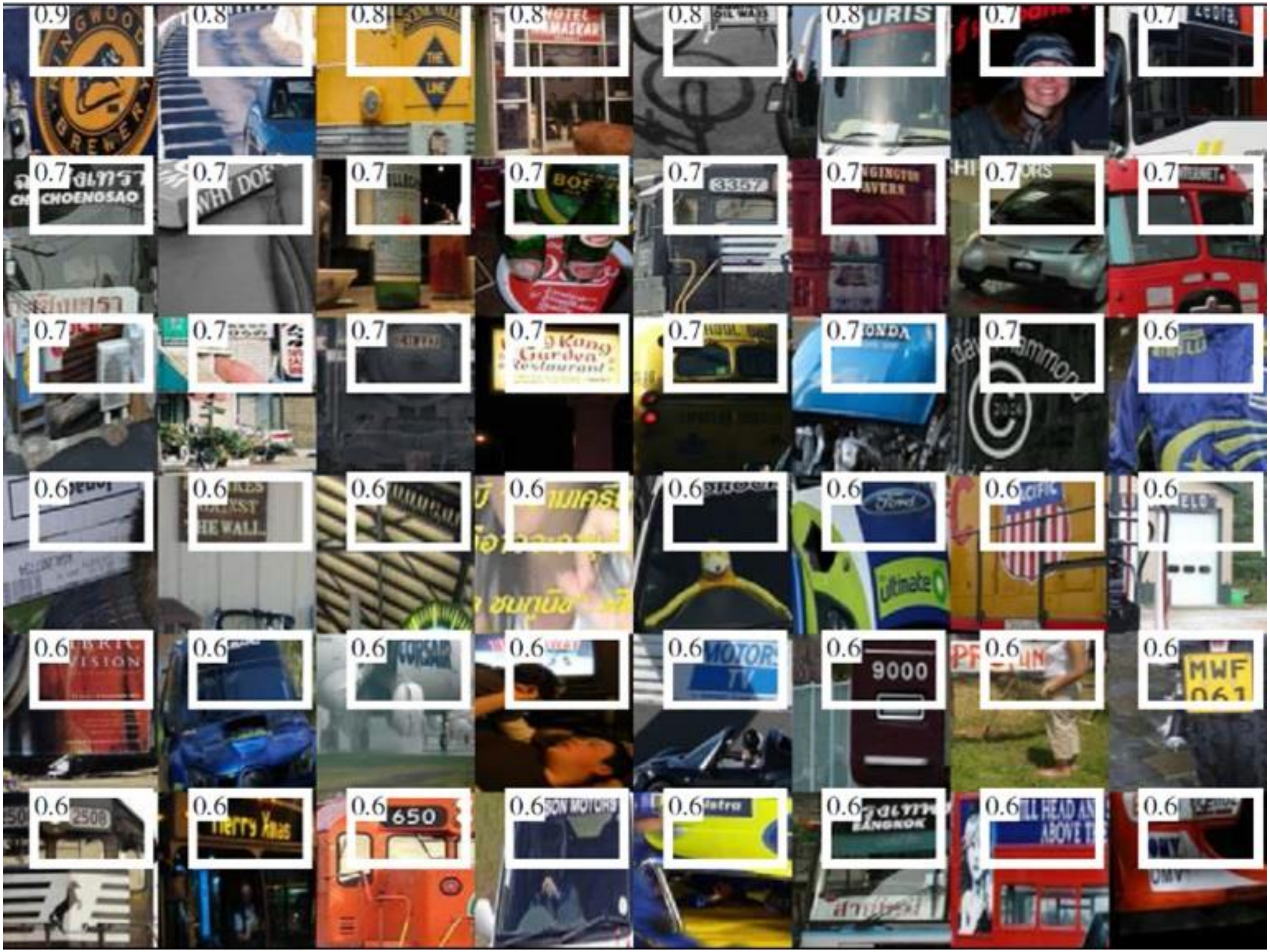
RCNN [Girshick et al. CVPR 2014]

Individual Neuron Activation



RCNN [Girshick et al. CVPR 2014]

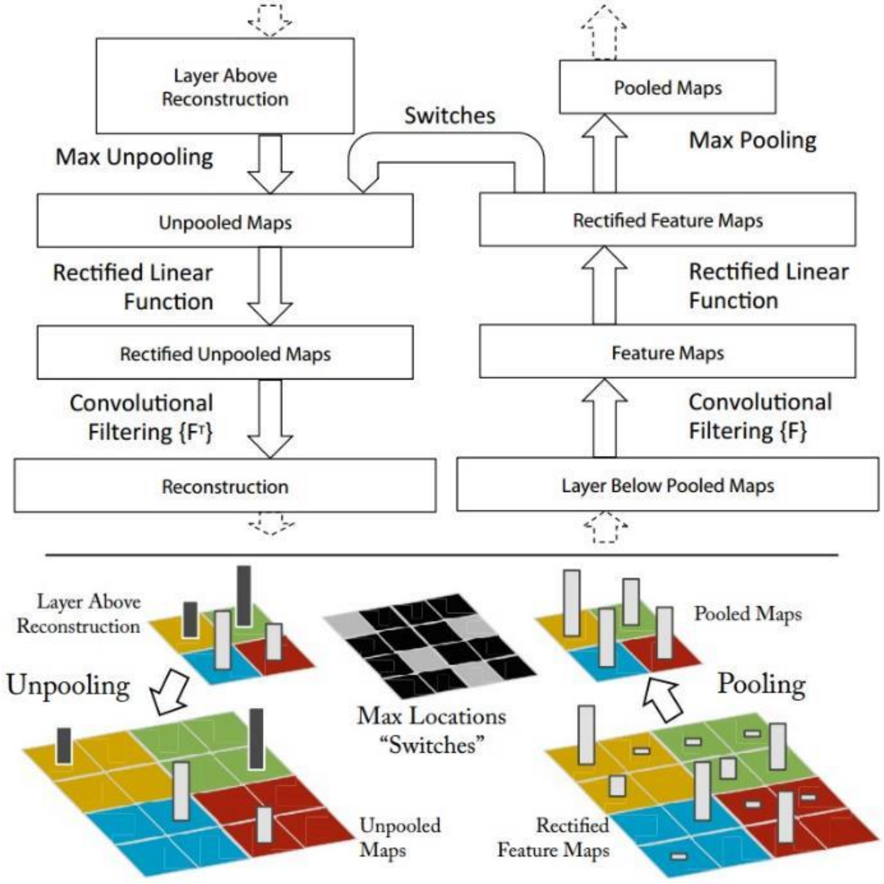
Individual Neuron Activation



RCNN [Girshick et al. CVPR 2014]

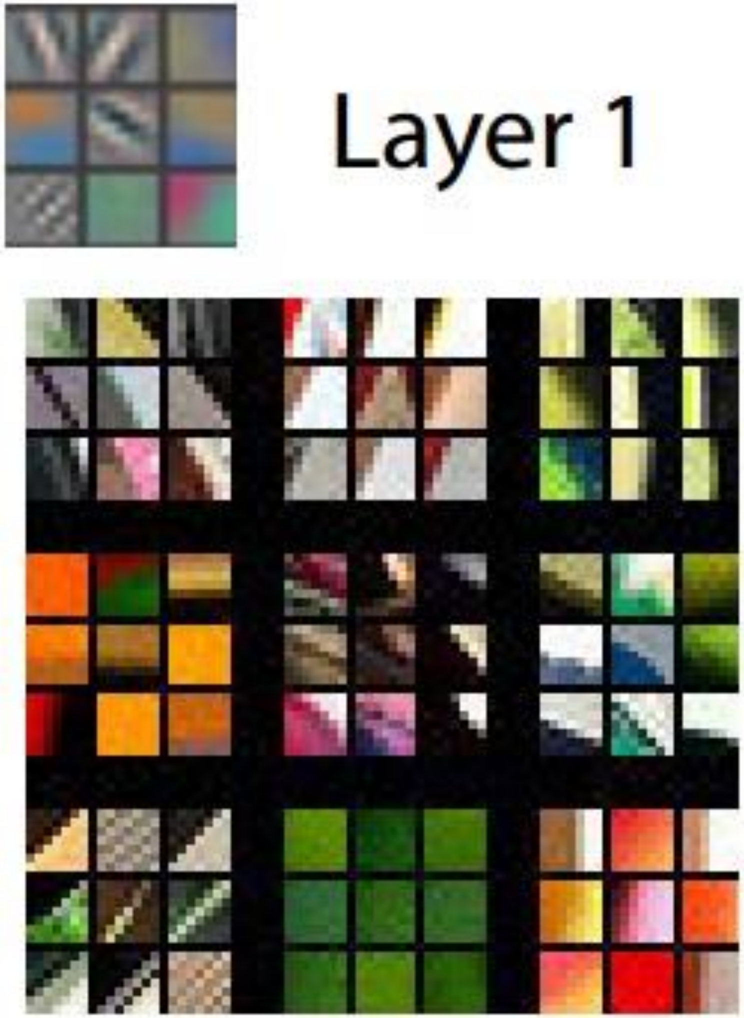
Map activation back to the input pixel space

- What input pattern originally caused a given activation in the feature maps?



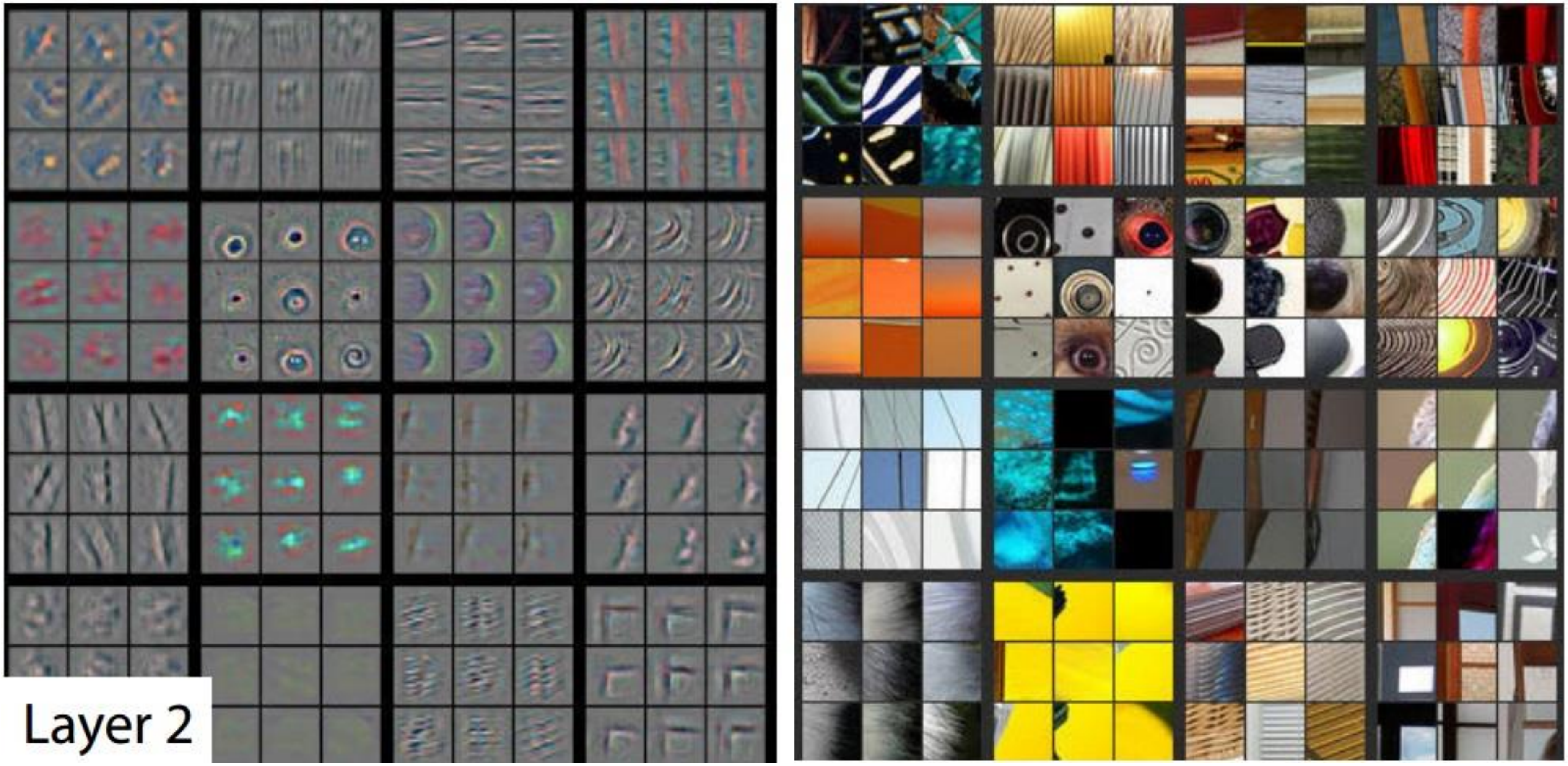
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 1



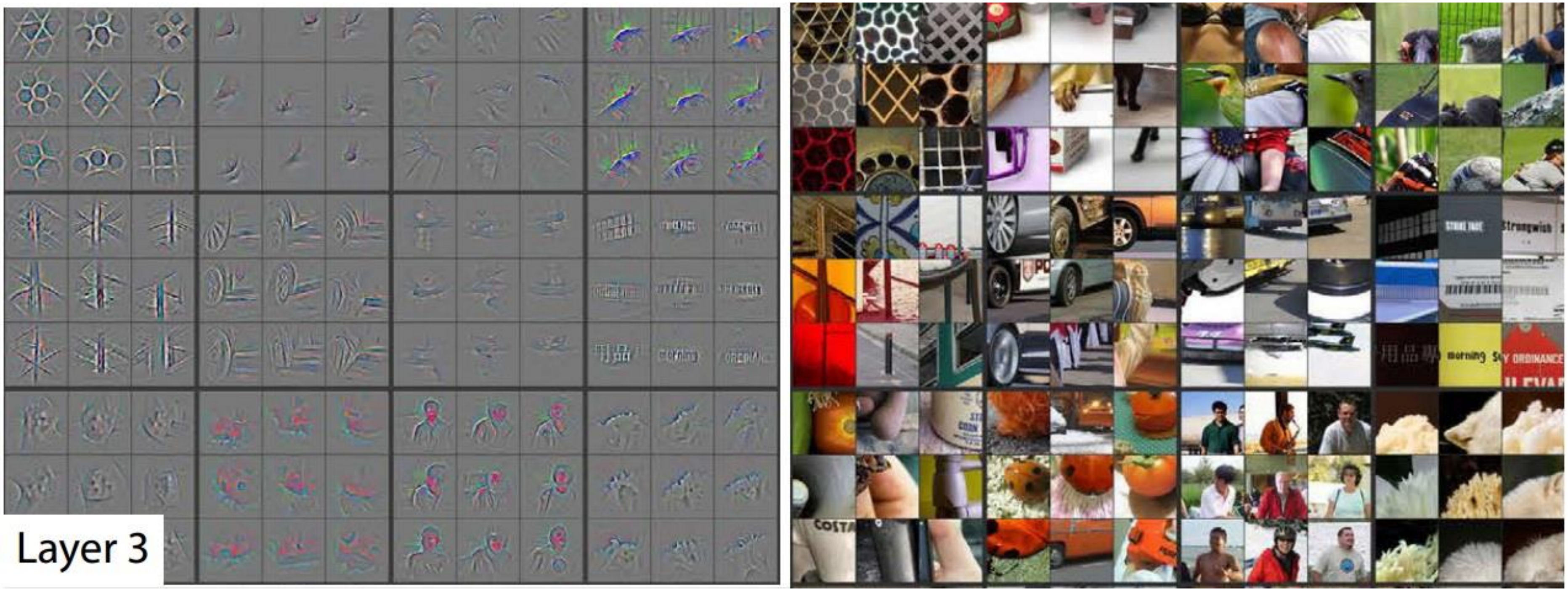
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 2



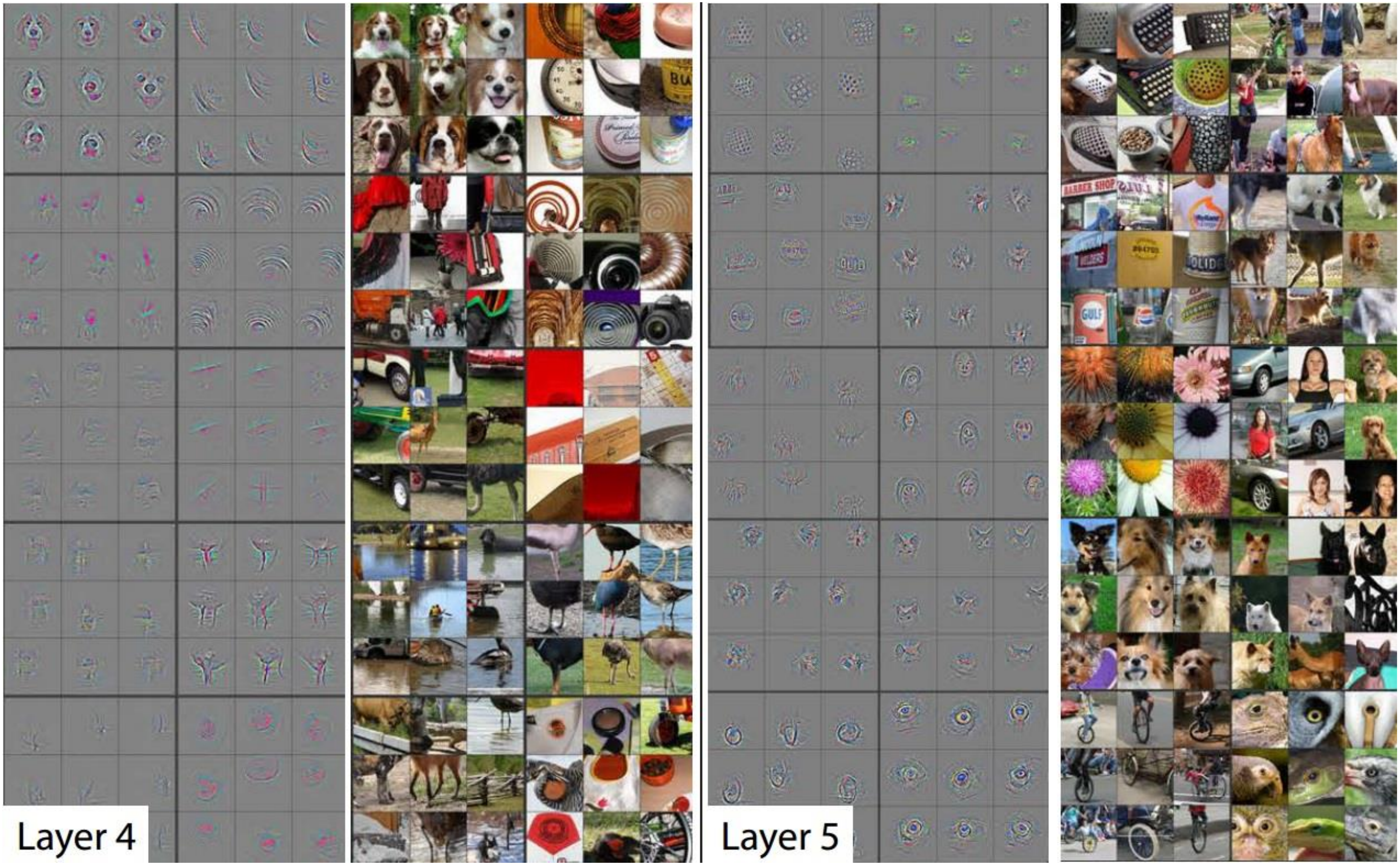
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 3



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

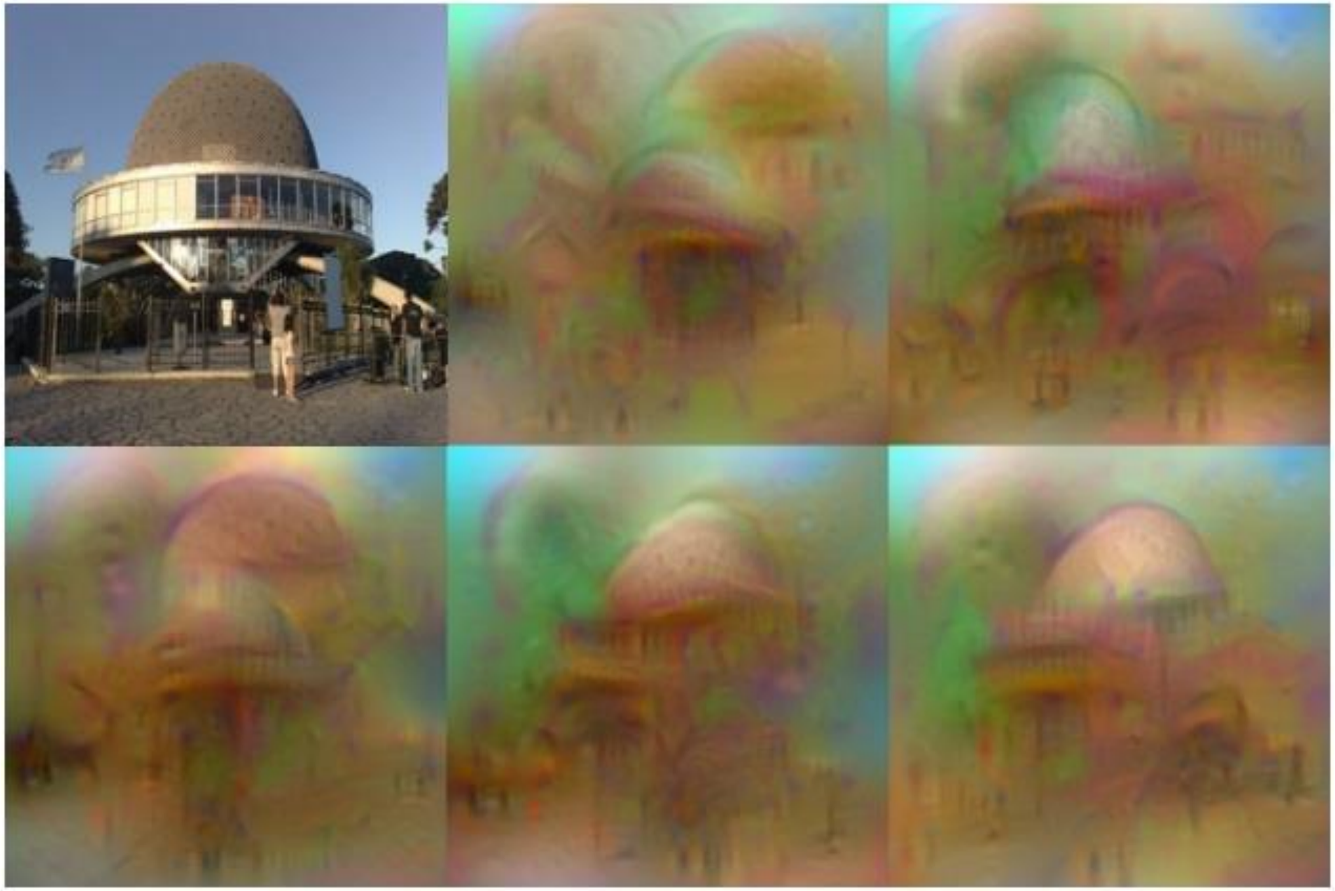
Layer 4 and 5



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Invert CNN features

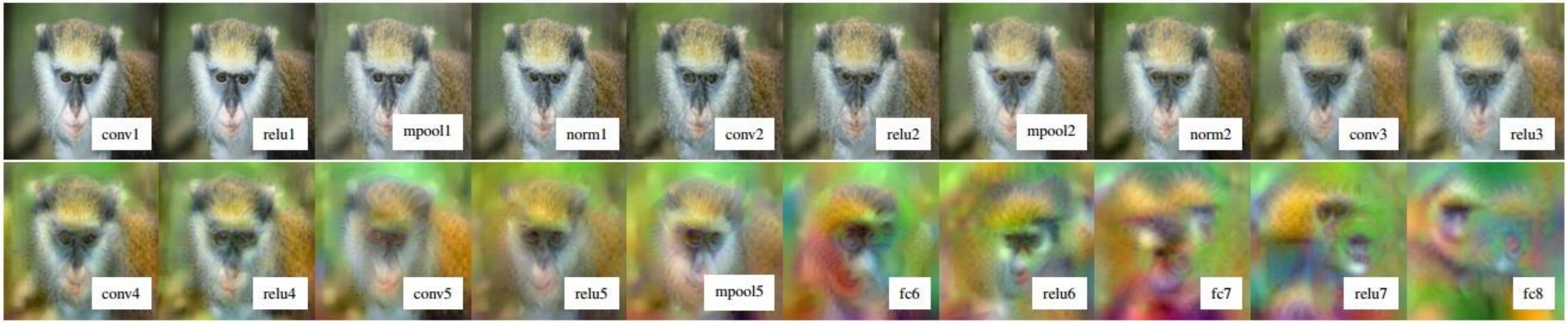
- Reconstruct an image from CNN features



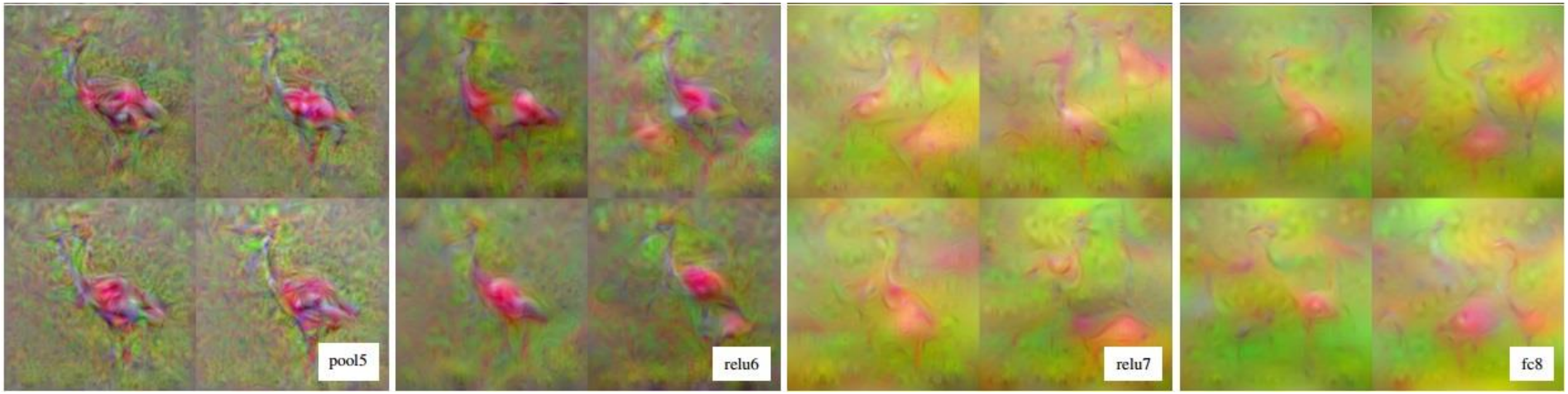
Understanding deep image representations by inverting them
[Mahendran and Vedaldi CVPR 2015]

CNN Reconstruction

Reconstruction from different layers



Multiple reconstructions

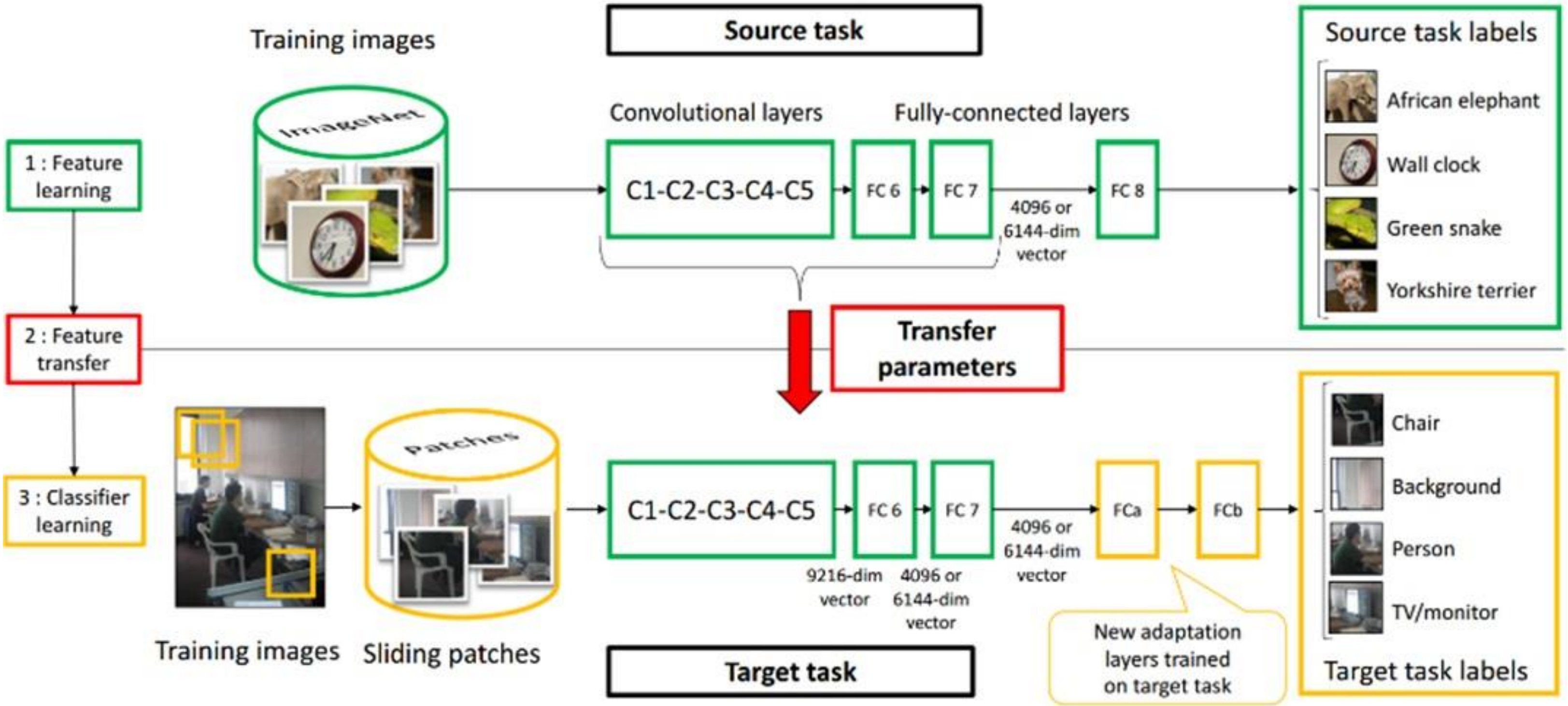


Understanding deep image representations by inverting them
[Mahendran and Vedaldi CVPR 2015]

Slide: Jiabin Huang

Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.
- Weight initialization for CNN



Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks [Oquab et al. CVPR 2014]

Tools

- Caffe
- cuda-convnet2
- Torch
- MatConvNet
- Pylearn2
- TensorFlow

Reading list

- https://culurciello.github.io/tech/2016/06/04/nets.html
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.
- A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012
- D. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, ICLR 2015
- M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014 (best paper award)
- K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
- M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014
- C. Szegedy et al., Going deeper with convolutions, CVPR 2015
- C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016
- K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (best paper award)

Next week

- Object detection and pixel labeling