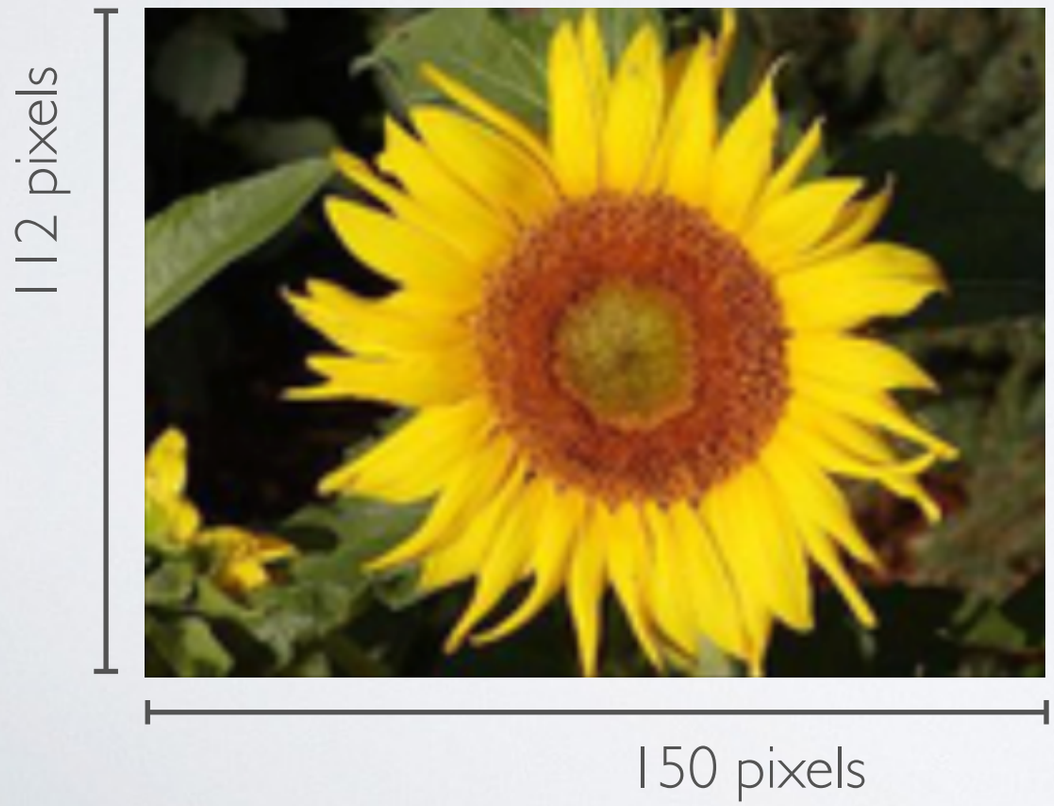
Convolutional Networks

COMPUTER VISION

Topics: computer vision, object recognition

- **GOAL**: Process visual data and accomplish some given task.
  - we will focus on **object recognition**: given some input image, identify which object it contains.

Caltech 101 dataset



"sun flower"

COMPUTER VISION

Topics: computer vision

- We can design neural networks that are specifically adapted for such problems
  - can deal with very high-dimensional inputs
    - 150 x 150 pixels = 22500 inputs, or 3 x 22500 if RGB pixels
  - can exploit the 2D topology of pixels (or 3D for video data)
  - can build-in invariance to certain variations we can expect
    - translations, illumination, etc.
- Convolutional networks leverage these ideas
  - local connectivity
  - parameter sharing
  - pooling / subsampling hidden units

COMPUTER VISION

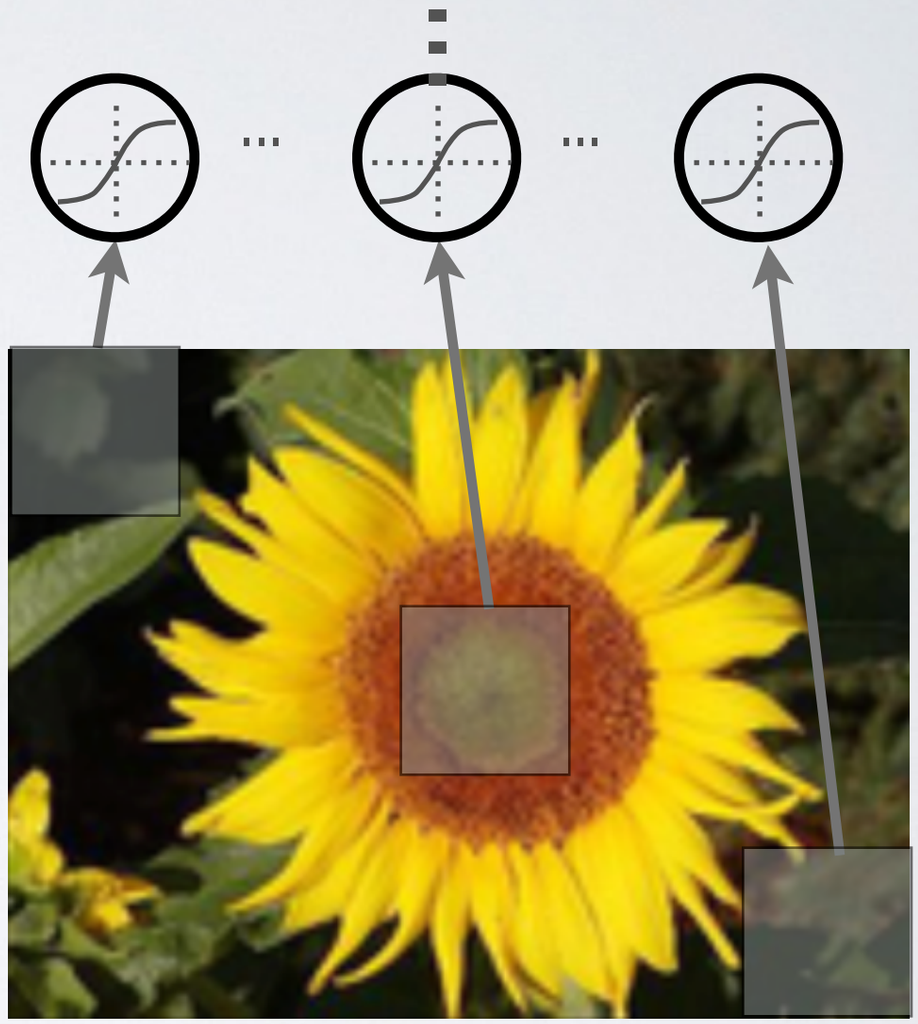
Topics: computer vision

- We can design neural networks that are specifically adapted for such problems
  - can deal with very high-dimensional inputs
    - 150 x 150 pixels = 22500 inputs, or 3 x 22500 if RGB pixels
  - can exploit the 2D topology of pixels (or 3D for video data)
  - can build in invariance to certain variations we can expect
    - translations, illumination, etc.
- Convolutional networks leverage these ideas
  - **local connectivity**
  - parameter sharing
  - pooling / subsampling hidden units

COMPUTER VISION

Topics: local connectivity

- First idea: use a local connectivity of hidden units
  - each hidden unit is connected only to a subregion (patch) of the input image
  - it is connected to all channels
    - 1 if greyscale image
    - 3 (R, G, B) for color image
- Solves the following problems:
  - fully connected hidden layer would have an unmanageable number of parameters
  - computing the linear activations of the hidden units would be very expensive

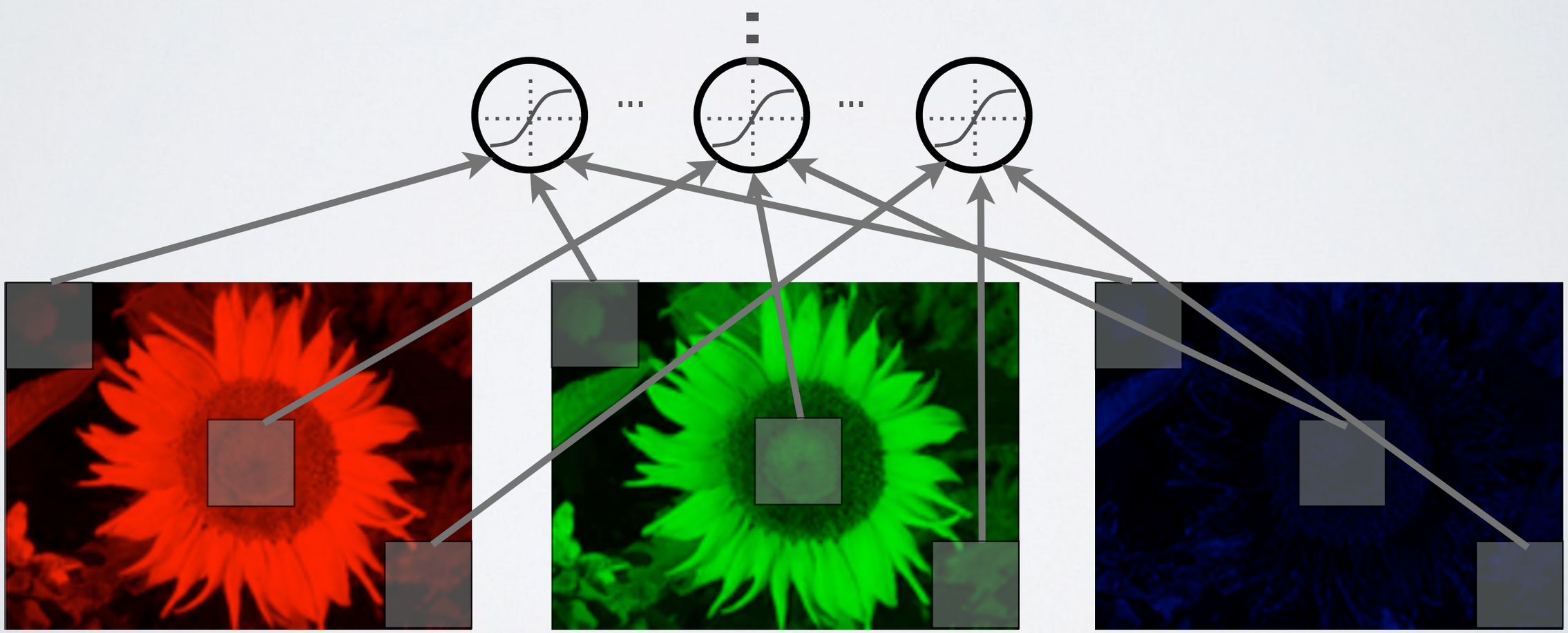


r = receptive

COMPUTER VISION

Topics: local connectivity

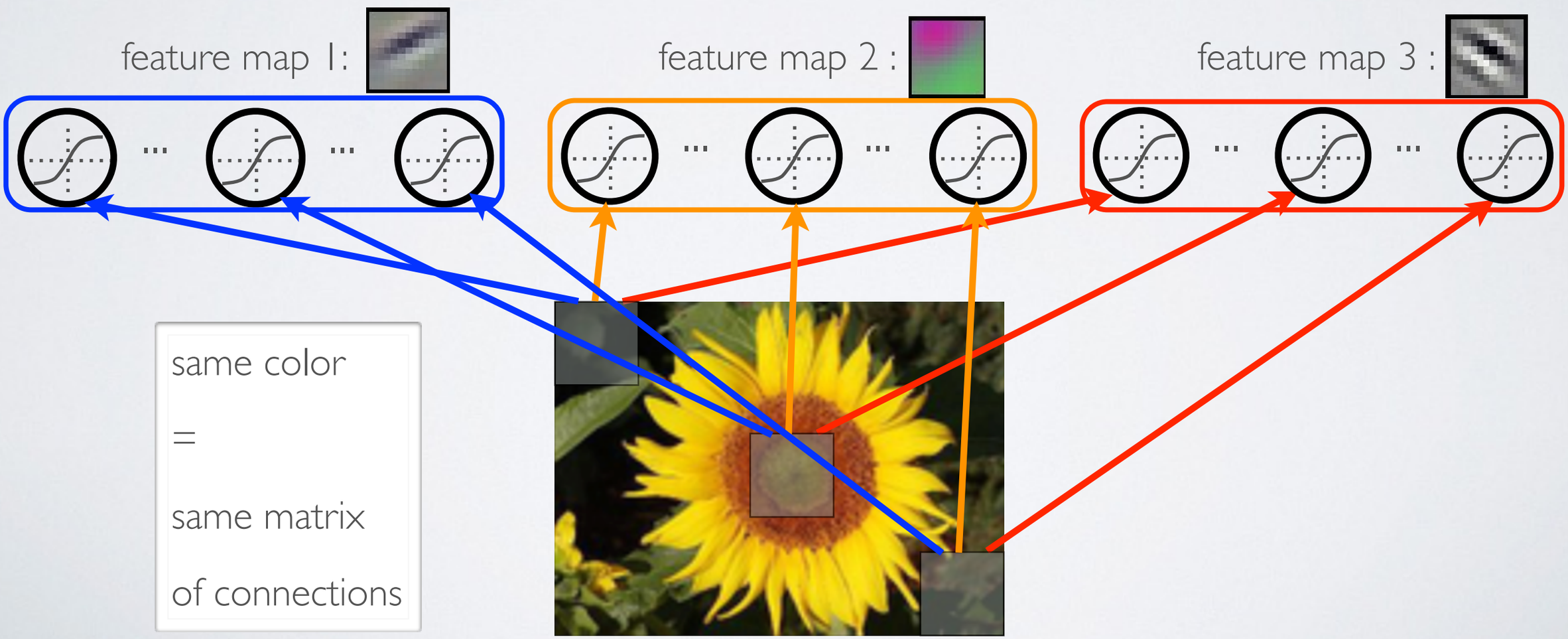
- Units are connected to all channels:
  - 1 channel if grayscale image, 3 channels (R, G, B) if color image



COMPUTER VISION

Topics: parameter sharing

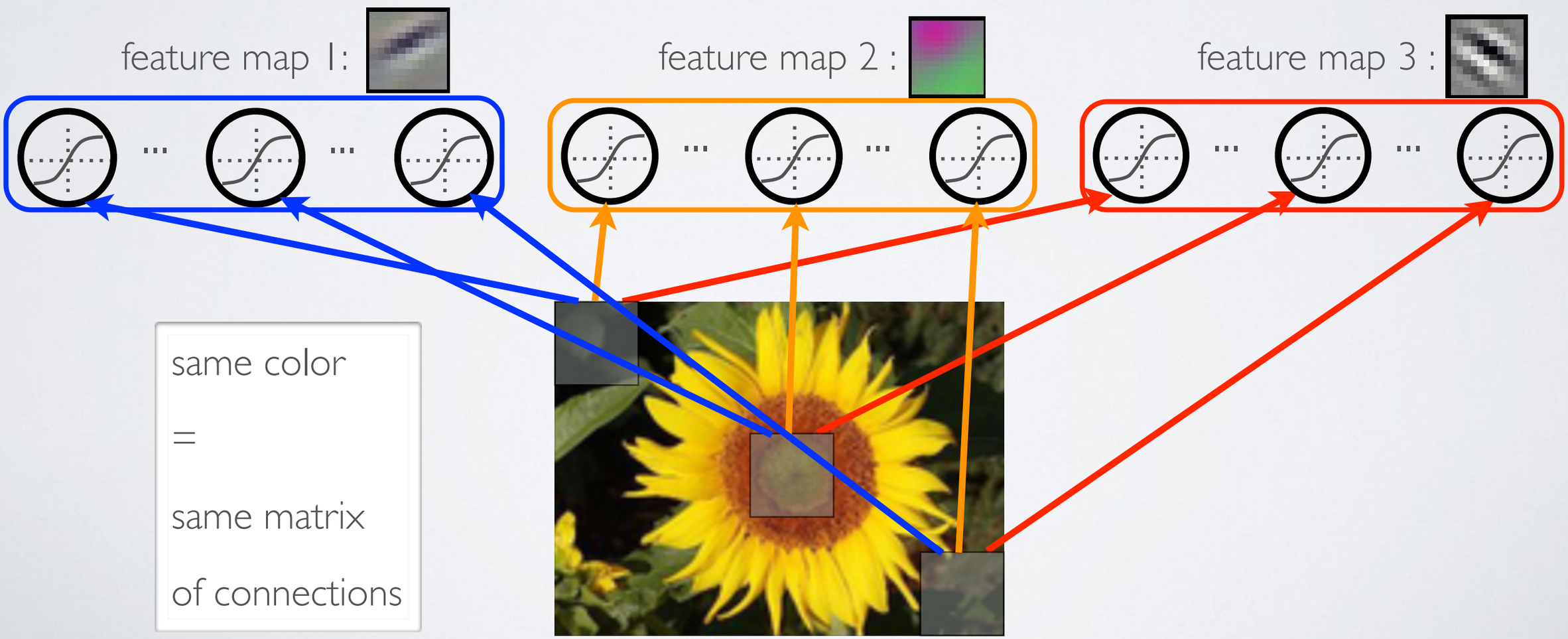
- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



COMPUTER VISION

Topics: parameter sharing

- Solves the following problems:
  - reduces even more the number of parameters
  - will extract the same features at every position (features are "equivariant")

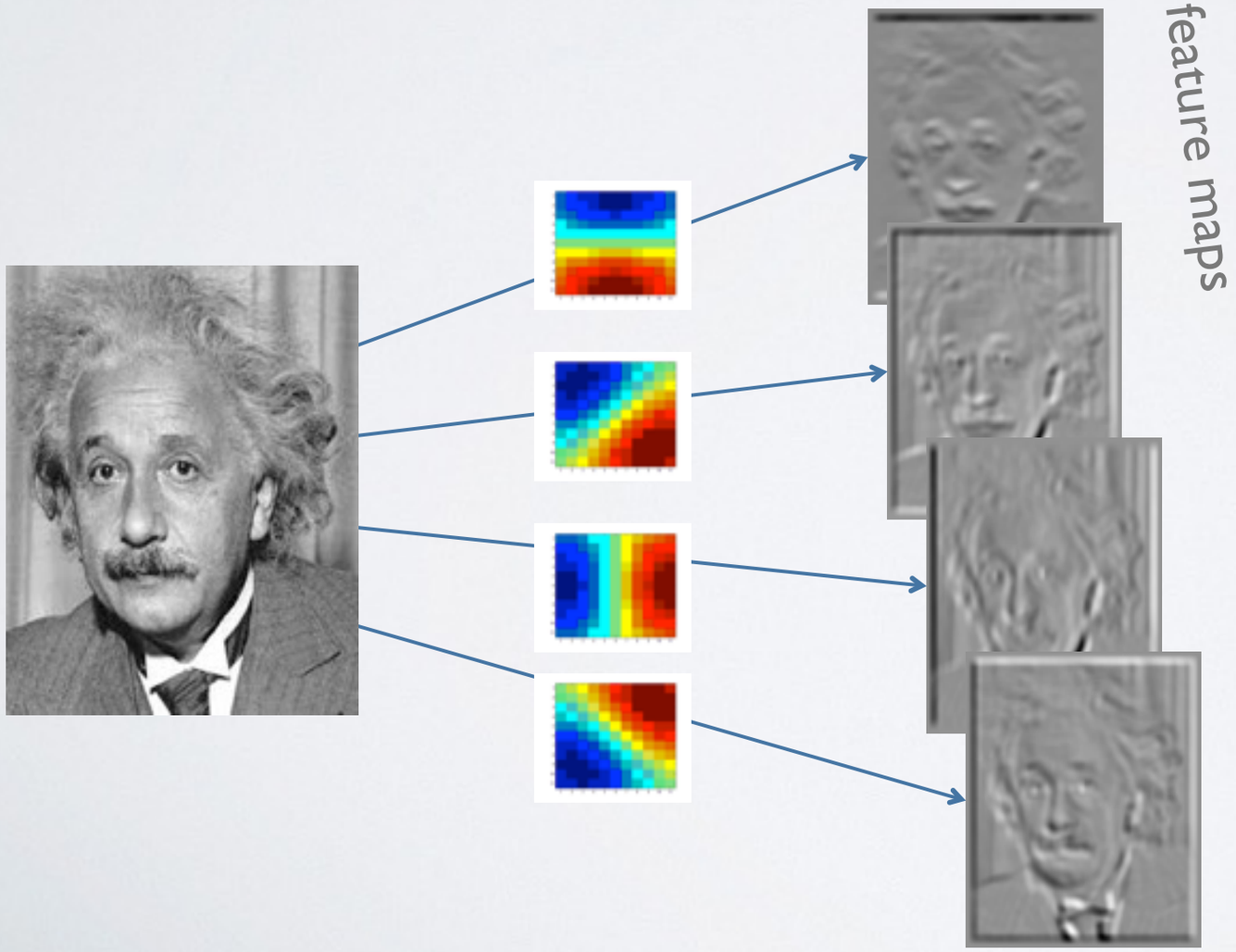


COMPUTER VISION

Topics: parameter sharing

Jarret et al. 2009

- Each feature map forms a 2D grid of features
  - can be computed with a discrete convolution ($*$) of a kernel matrix k_{ij} which is the hidden weights matrix W_{ij} with its rows and columns flipped



  - x_i is the i[th] channel of input
  - k_{ij} is the convolution kernel
  - g_j is a learned scaling factor
  - y_j is the hidden layer

$$y_j = g_j \tanh(\sum_i k_{ij} * x_i)$$

COMPUTER VISION

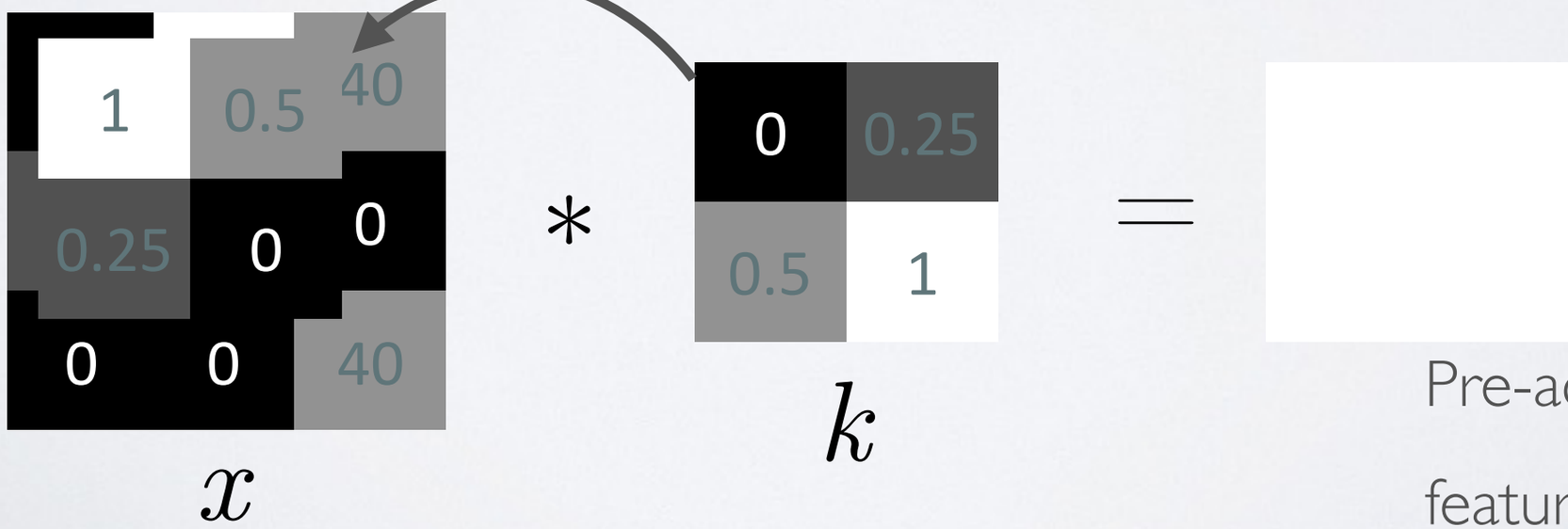
Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:

$\tilde{k}$ = k with rows and columns flipped



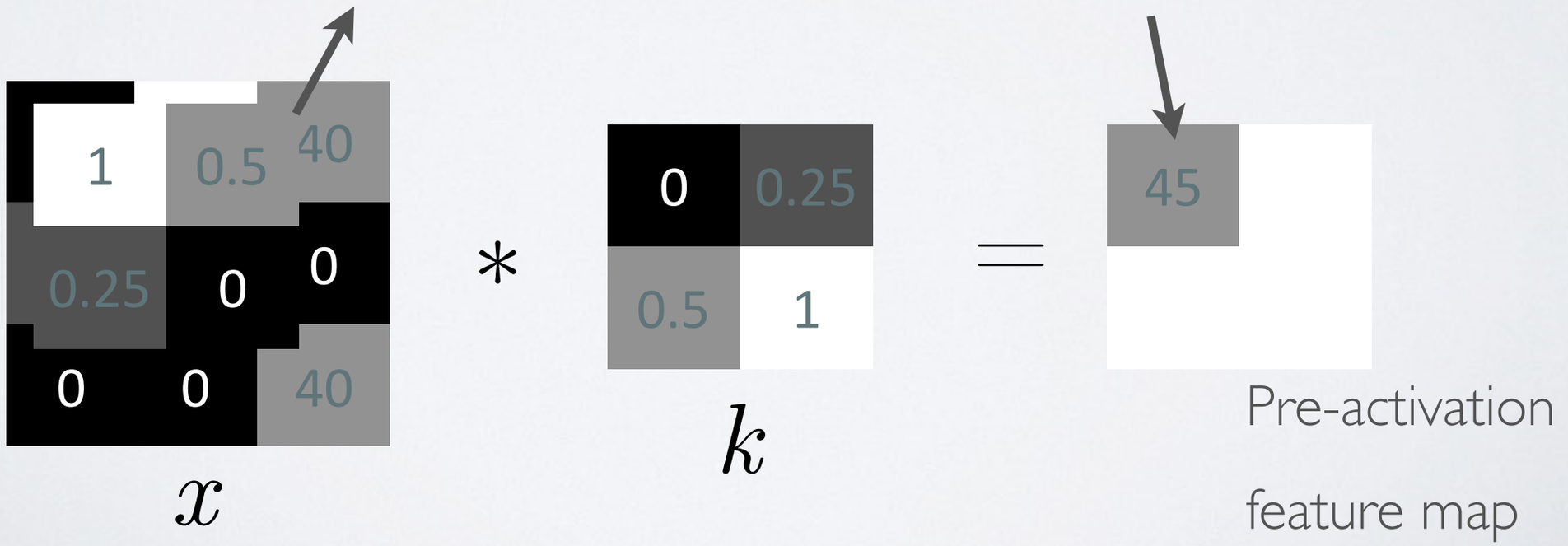
COMPUTER VISION

Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:



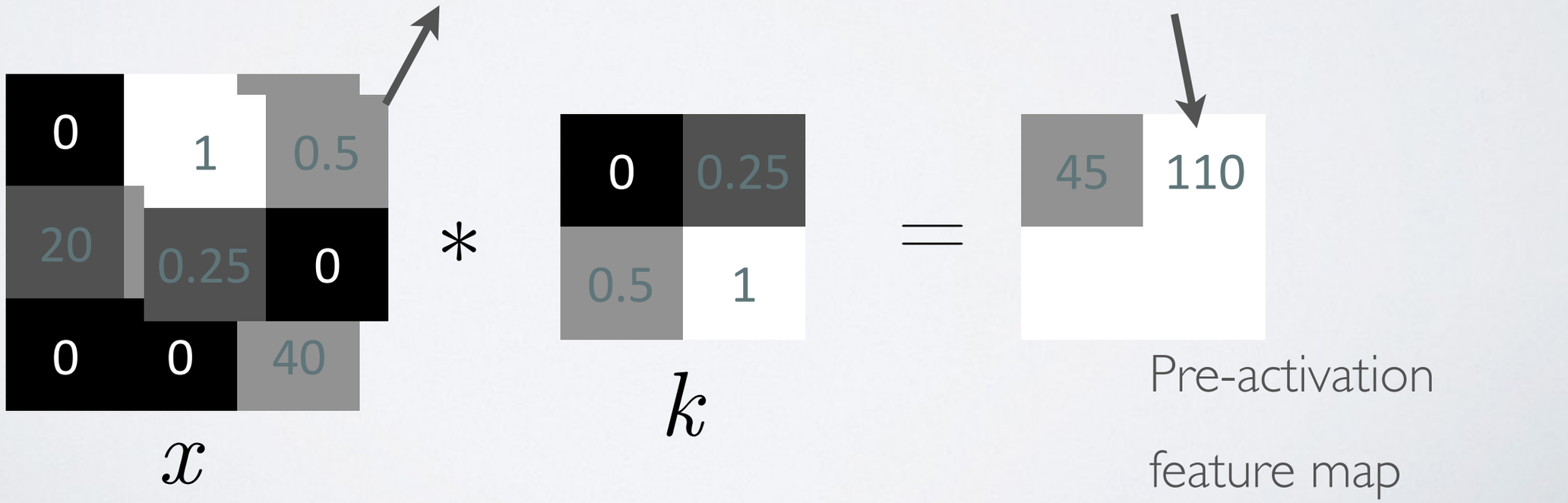
COMPUTER VISION

Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:



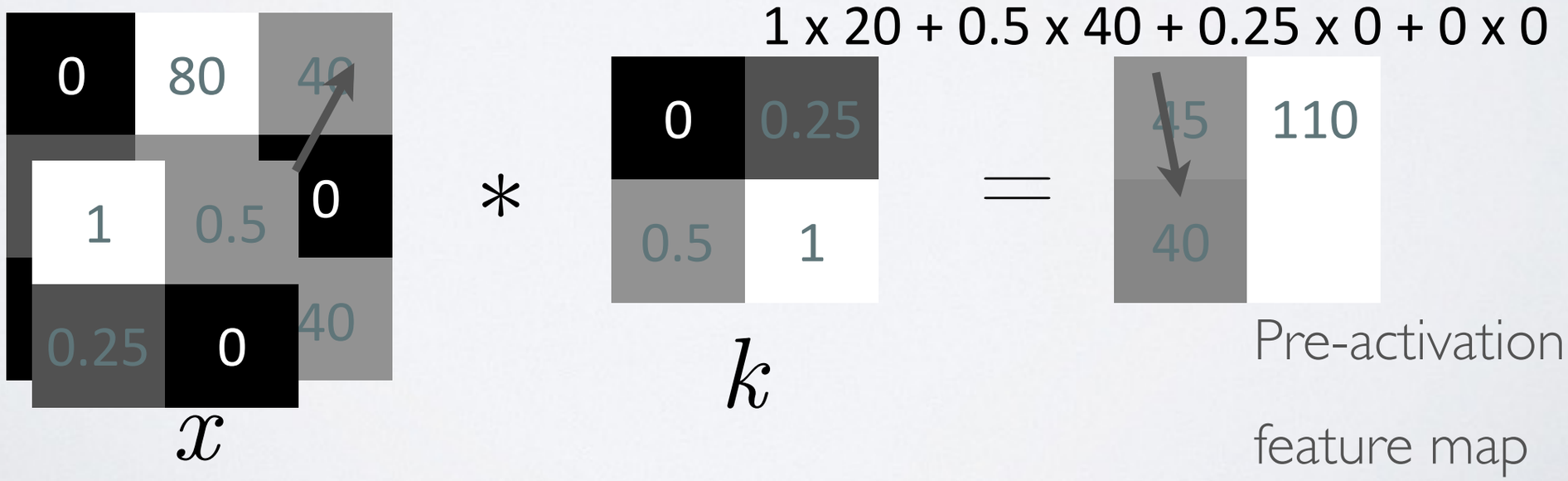
COMPUTER VISION

Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:



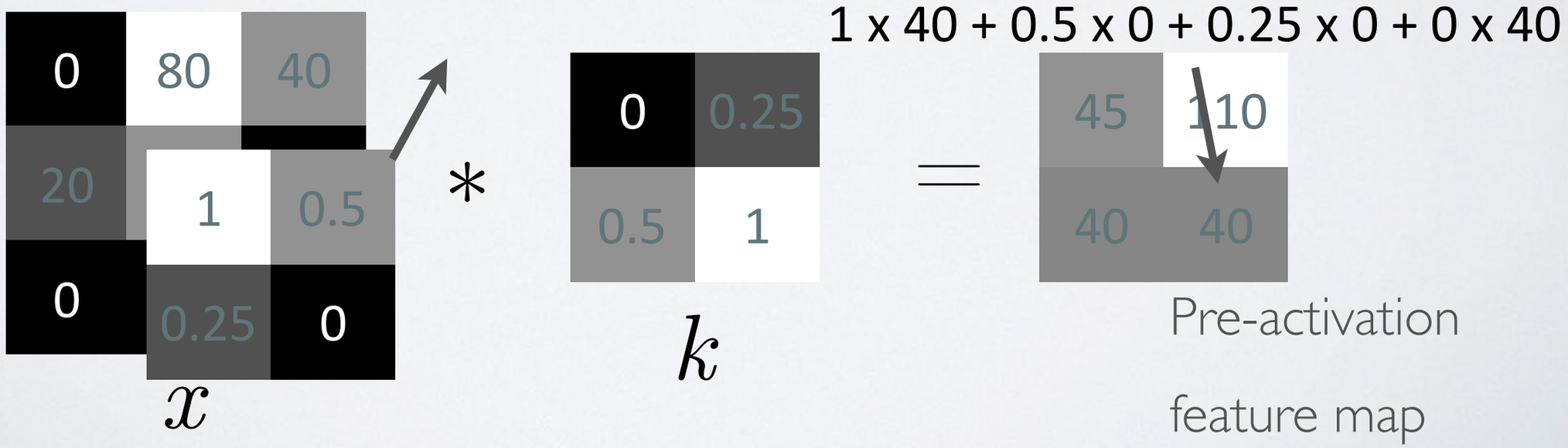
COMPUTER VISION

Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

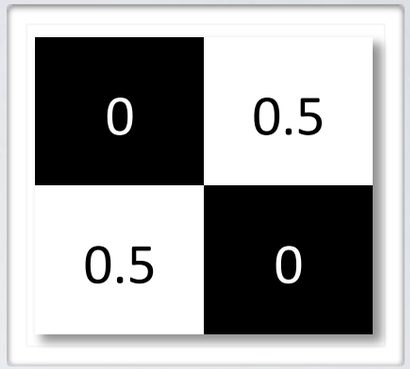
- Example:

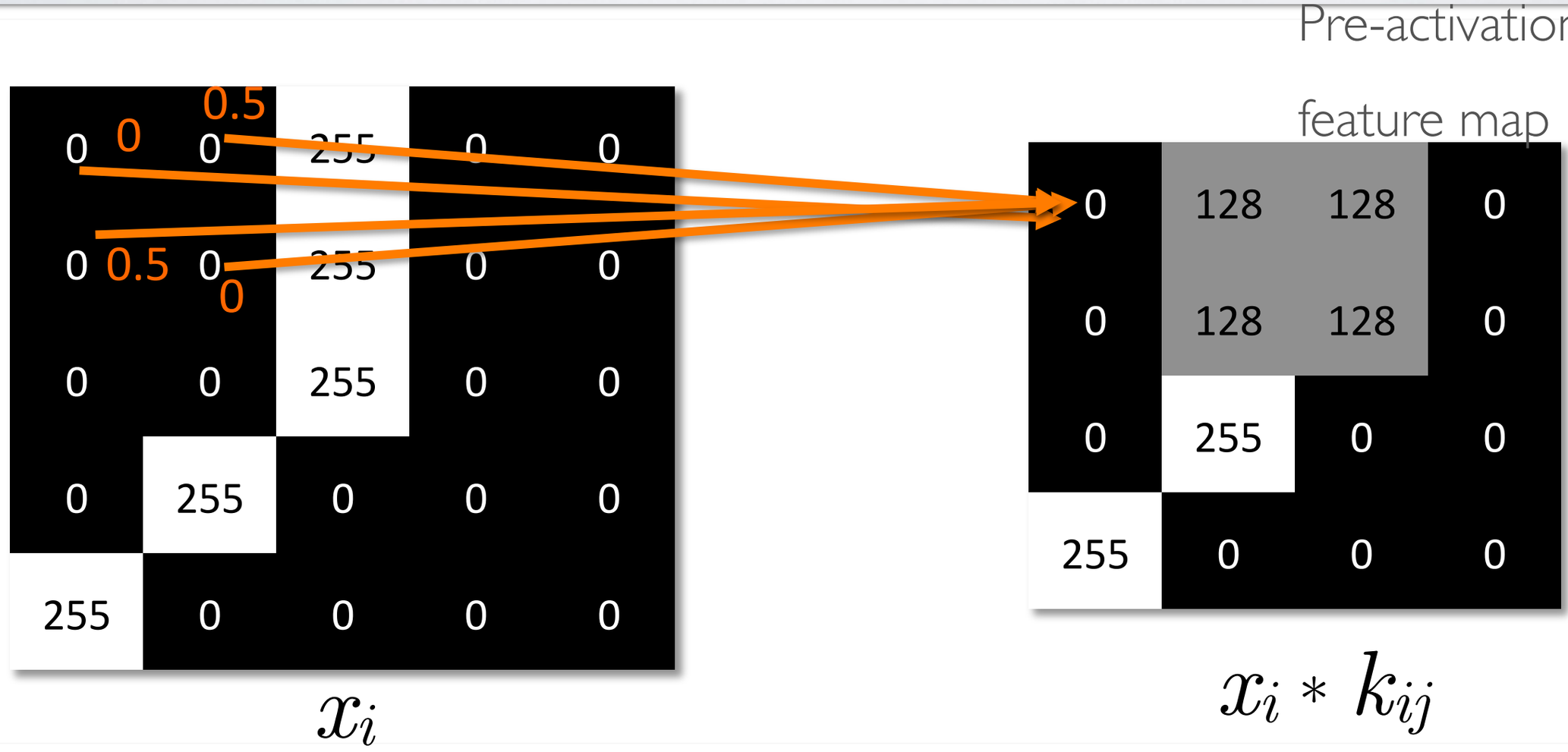


COMPUTER VISION

Topics: discrete convolution

- Simple illustration: $x_i * k_{ij}$ where $W_{ij} = \tilde{W}_{ij}$

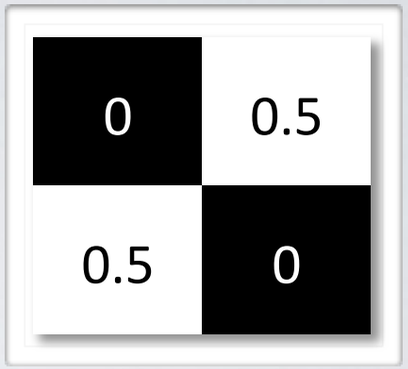


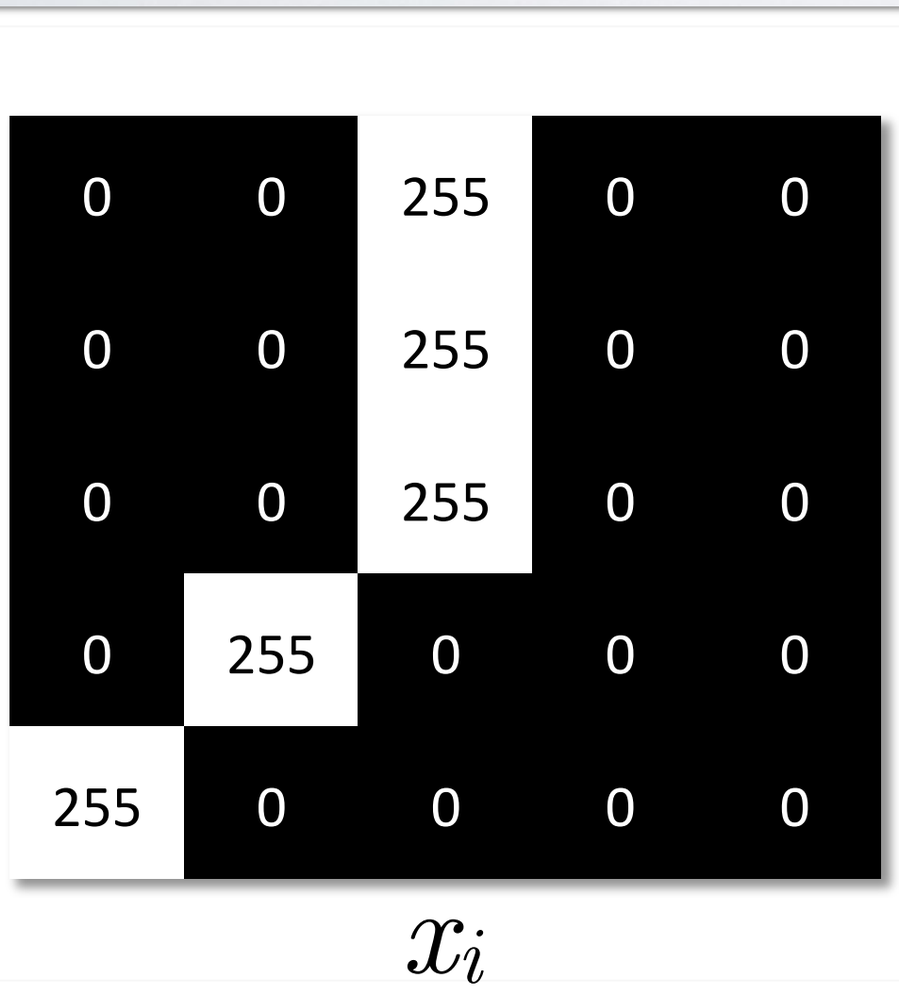


COMPUTER VISION

Topics: discrete convolution

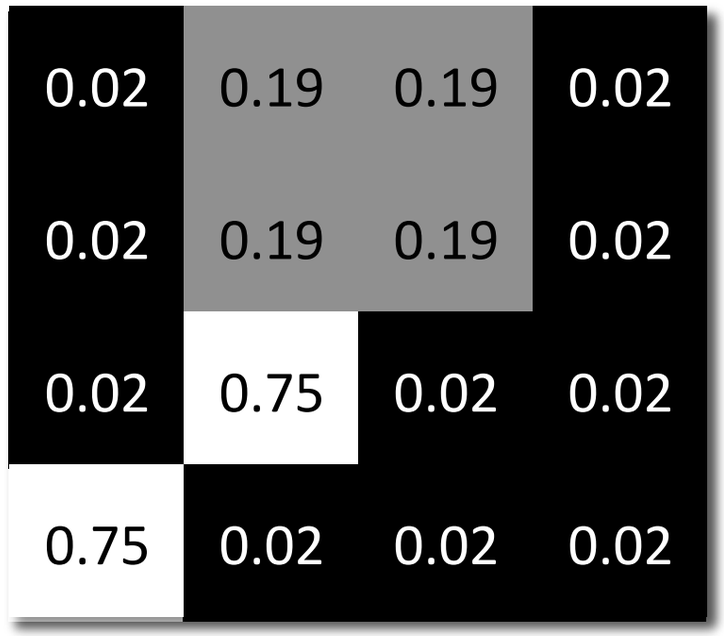
- With a non-linearity, we get a detector of a feature at any position in the image





x_i

feature map
(post-activation)

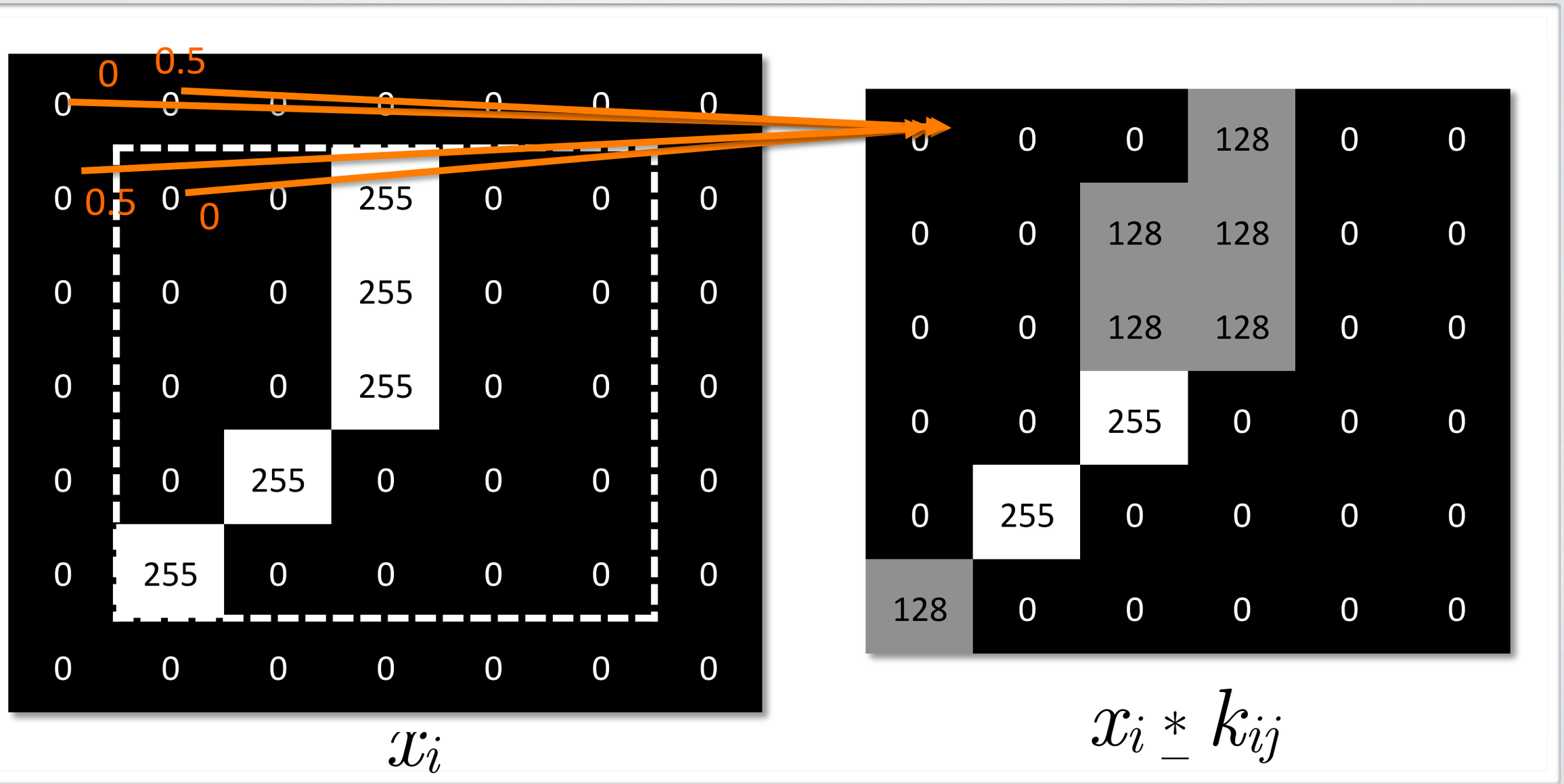


$$\text{sigm}(0.02\ x_i * k_{ij}\ \text{-}4)$$

COMPUTER VISION

Topics: discrete convolution

- Can use "zero padding" to allow going over the borders ($\underline{*}$)

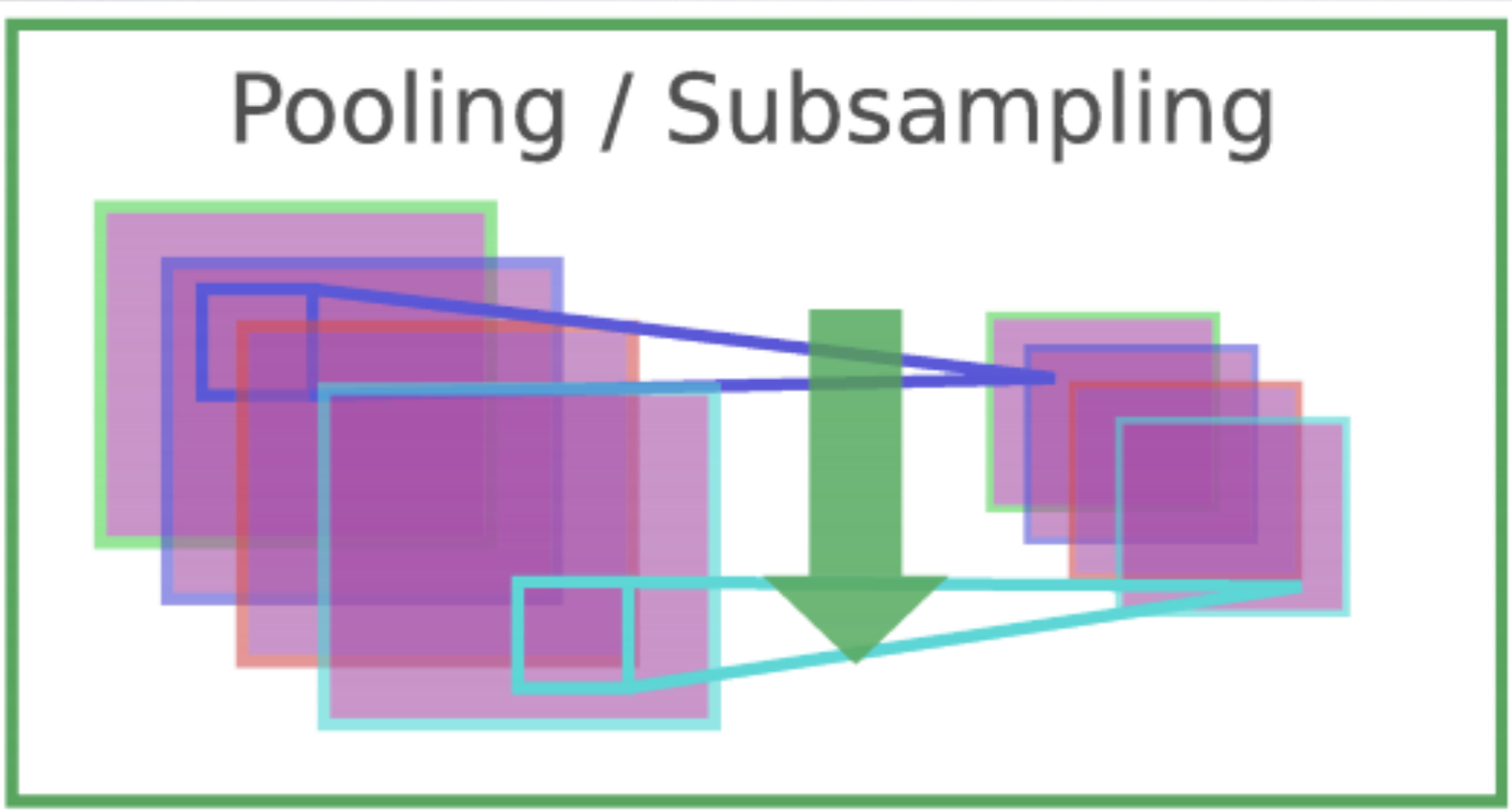


COMPUTER VISION

Topics: pooling and subsampling

Jarret et al. 2009

- Third idea: pool hidden units in same neighborhood
  - pooling is performed in non-overlapping neighborhoods (subsampling)



- $x_{i,j,k}$ is value of the i[th] feature map at position j,k
- p is vertical index in local neighborhood
- q is horizontal index in local neighborhood
- y_{ijk} is pooled and subsampled layer

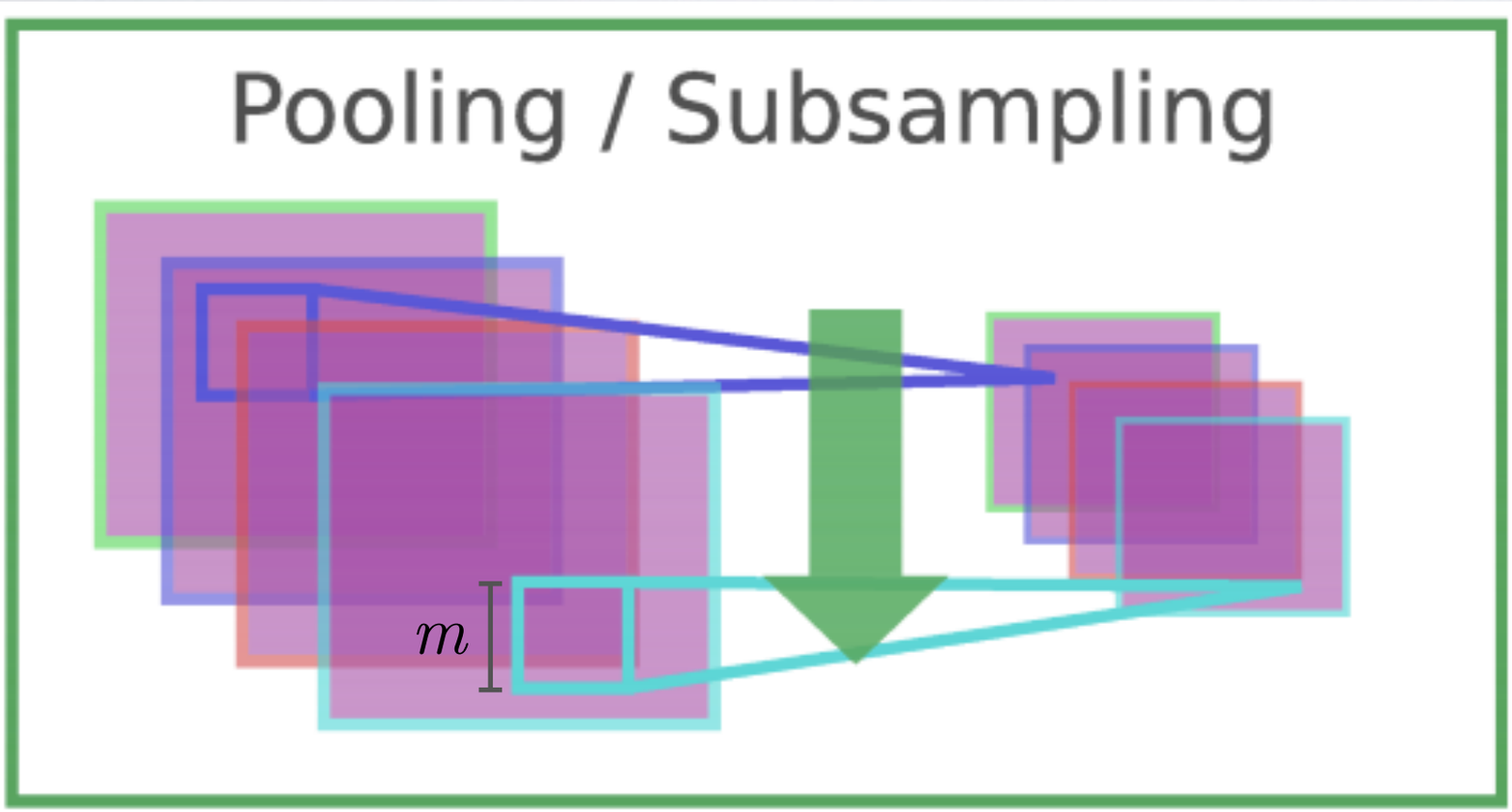
$$y_{ijk} = \max_{p,q} x_{i,j+p,k+q}$$

COMPUTER VISION

Topics: pooling and subsampling

Jarret et al. 2009

- Third idea: pool hidden units in same neighborhood
  - pooling is performed in non-overlapping neighborhoods (subsampling)



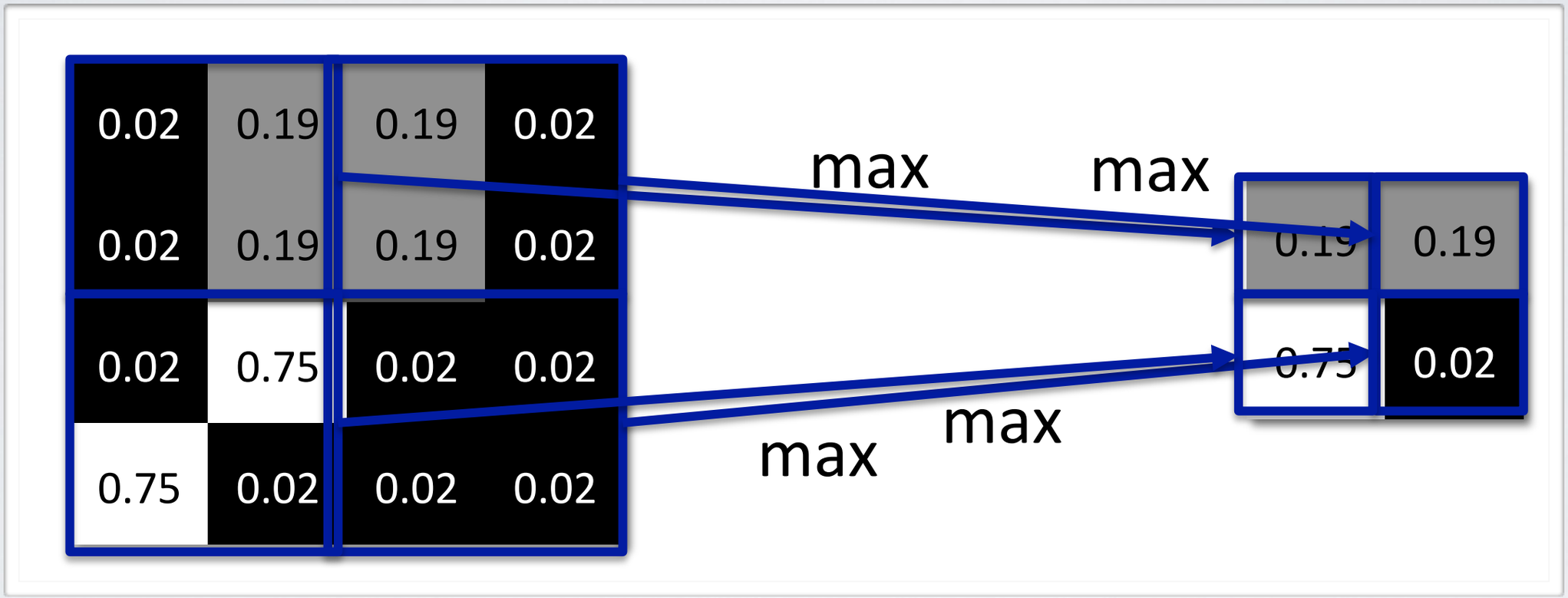
$$y_{ijk} = \frac{1}{m^2} \sum_{p,q} x_{i,j+p,k+q}$$

- $x_{i,j,k}$ is value of the i[th] feature map at position j,k
- p is vertical index in local neighborhood
- q is horizontal index in local neighborhood
- y_{ijk} is pooled and subsampled layer
- m is the neighborhood height/width

COMPUTER VISION

Topics: pooling and subsampling

- Third idea: pool hidden units in same neighborhood
  - pooling is performed in (mostly) non-overlapping neighborhoods (subsampling)

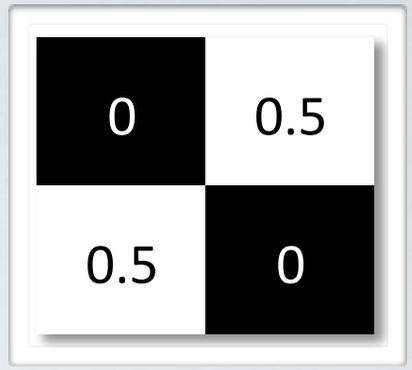


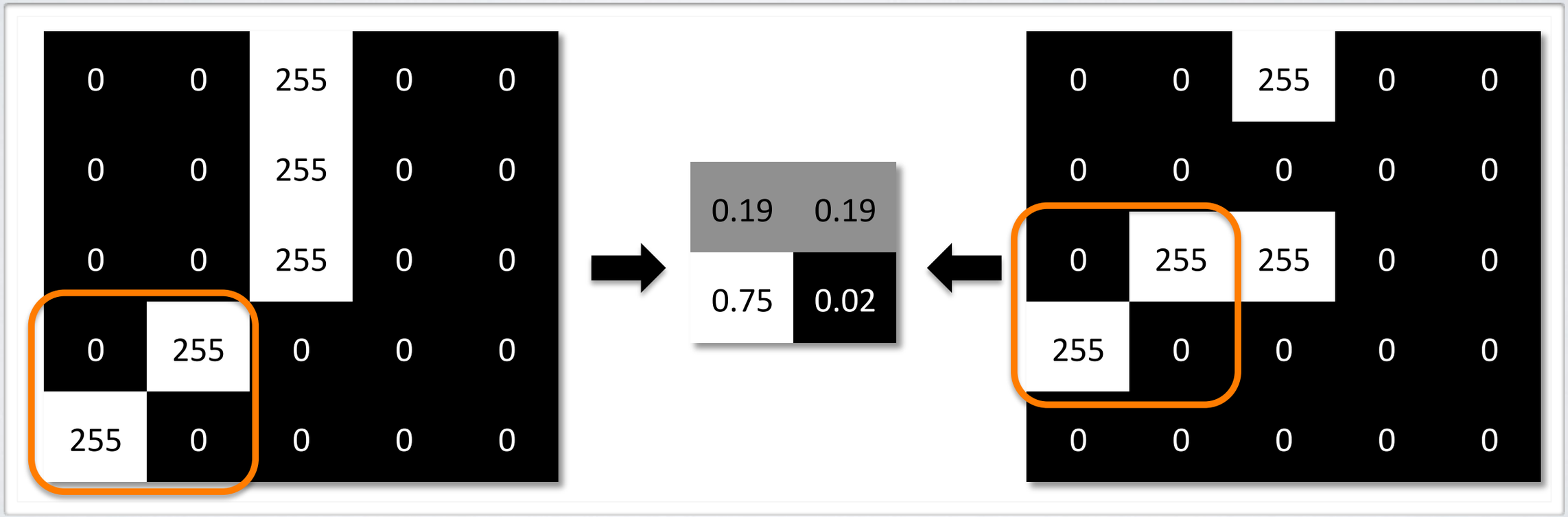
- Solves the following problems:
  - introduces invariance to local translations
  - reduces the number of hidden units in hidden layer

COMPUTER VISION

Topics: pooling and subsampling

- Illustration of local translation invariance
  - both images given the same feature map after pooling/subsampling

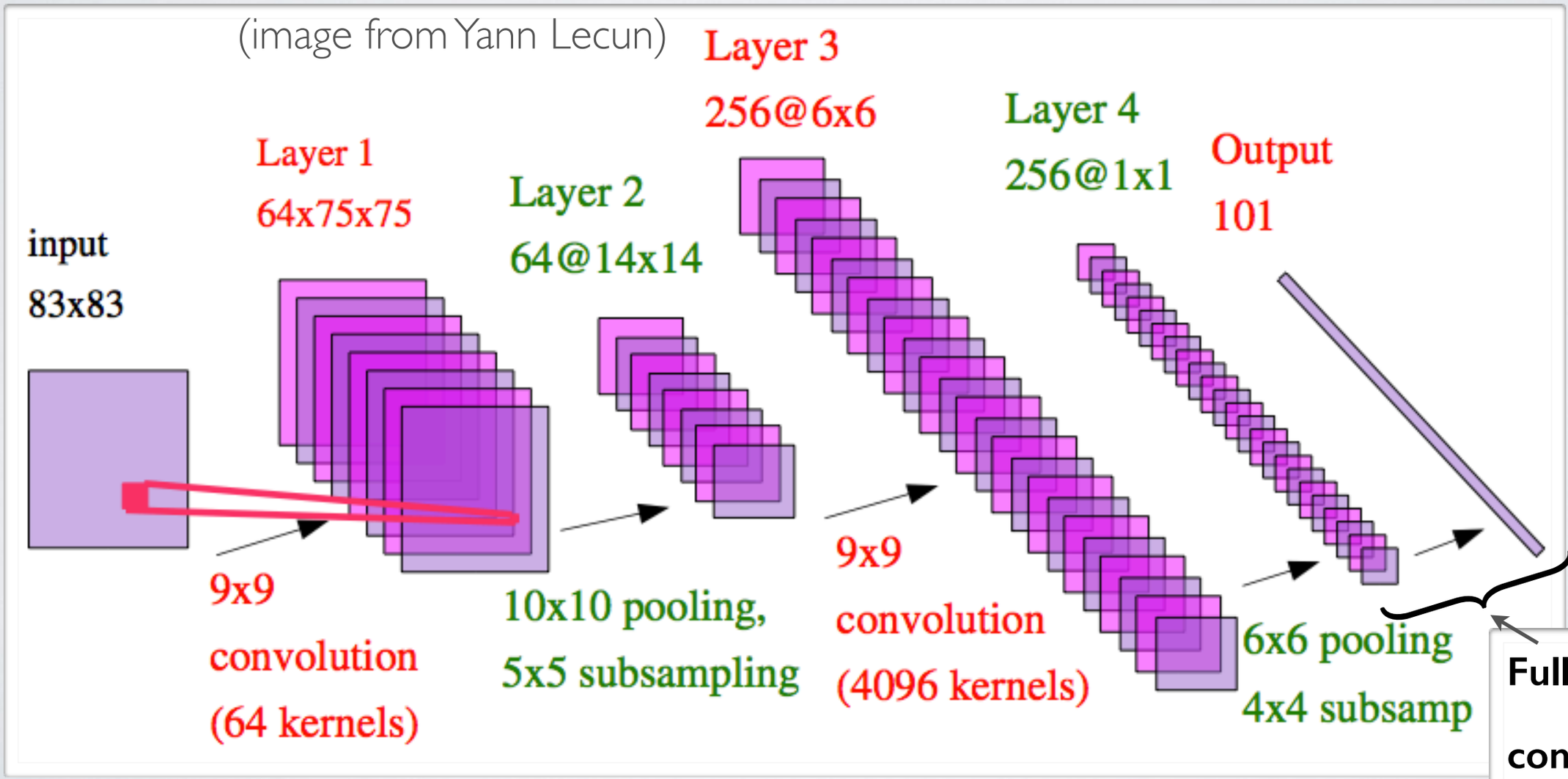




CONVOLUTIONAL NETWORK

Topics: convolutional network

- Convolutional neural network alternates between the convolutional and pooling layers



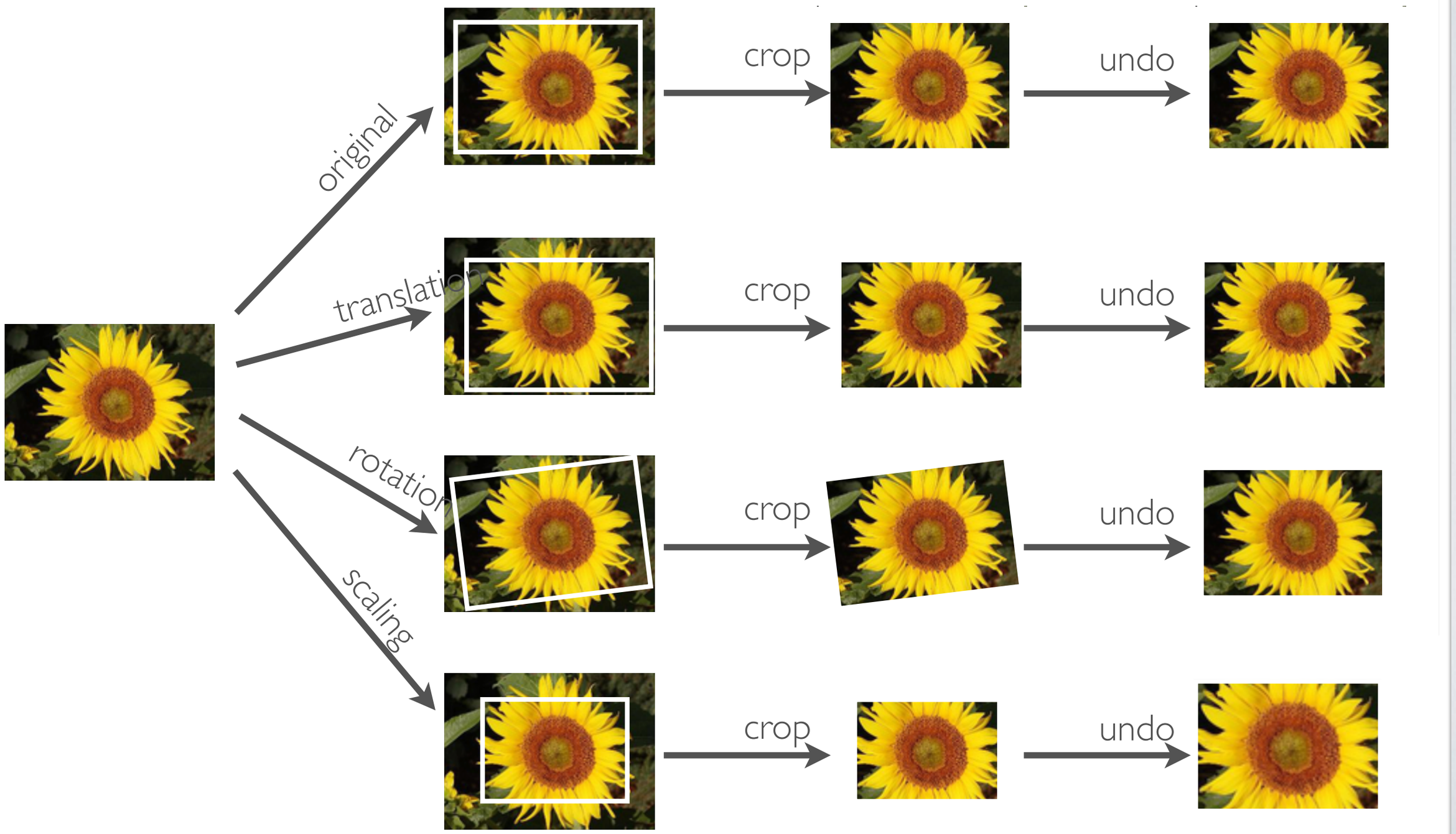
INVARIANCE BY DATA SET EXPANSION

Topics: generating additional examples

- Invariances built-in in convolutional network:
  - small translations: due to convolution and max pooling
- It is not invariant to other important variations such as rotations and scale changes
- However, it's easy to artificially generate data with such transformations
  - could use such data as additional training data
  - neural network will learn to be invariant to such transformations

INVARIANCE BY DATA SET EXPANSION

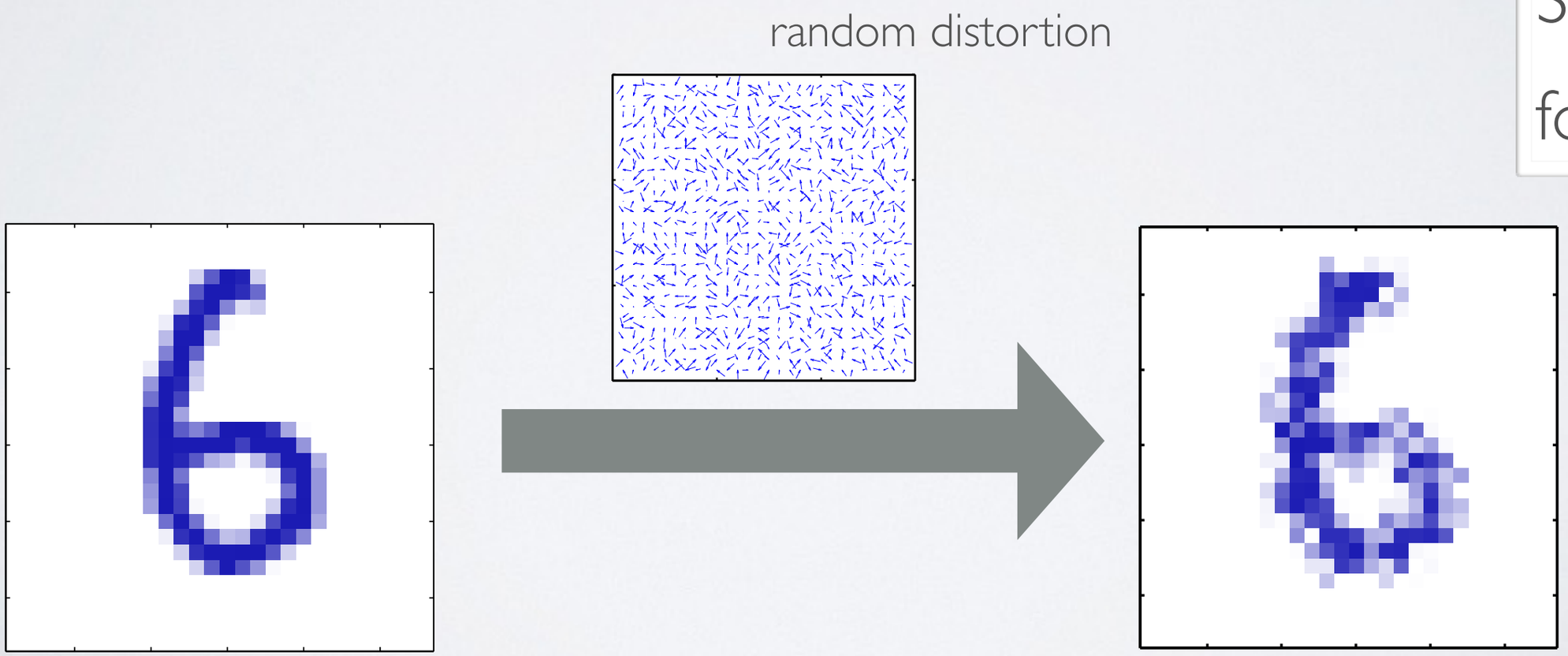
Topics: generating additional examples



INVARIANCE BY DATA SET EXPANSION

Topics: generating additional examples, distortion field

- Can add "elastic" deformations (useful in character recognition)
- We do this by applying a "distortion field" to the image
  - a distortion field specifies where to displace each pixel value



(from Bishop's book)

See Simard

for more d

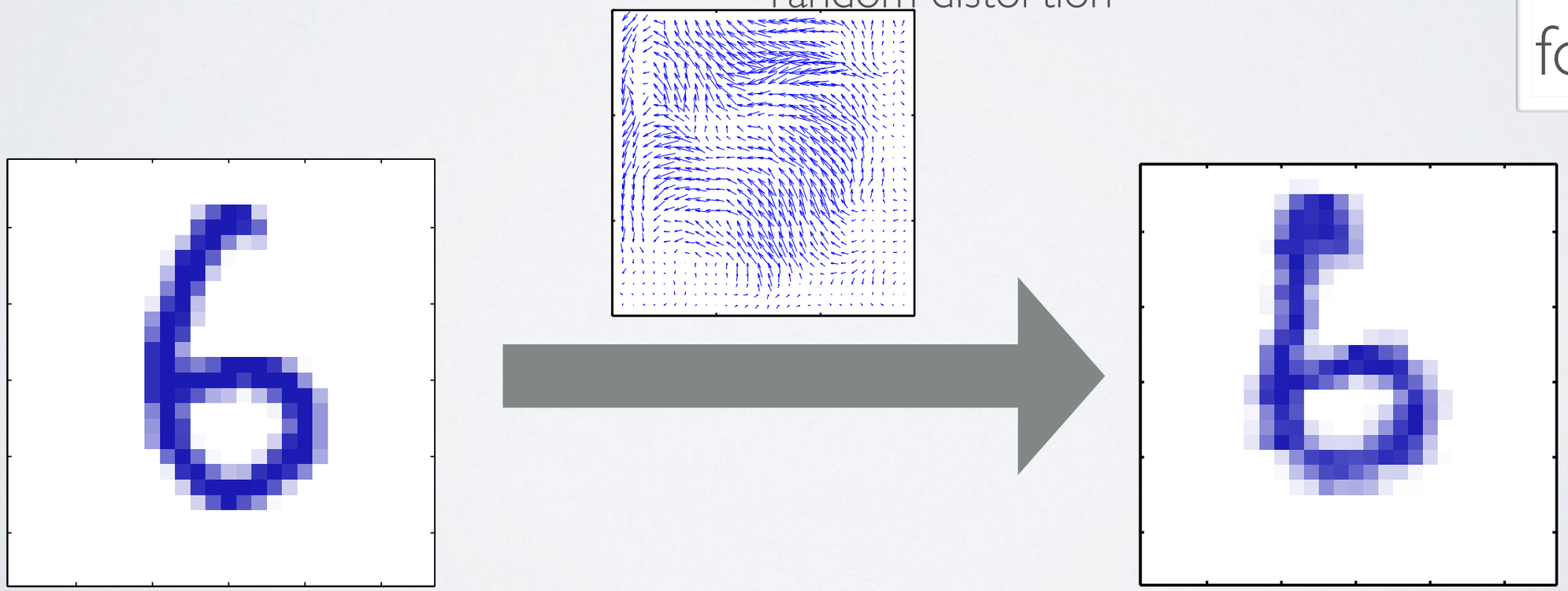
INVARIANCE BY DATA SET EXPANSION

Topics: generating additional examples, distortion field

- Can add "elastic" deformations (useful in character recognition)
- We do this by applying a "distortion field" to the image
  - a distortion field specifies where to displace each pixel value

smoothed

random distortion



(from Bishop's book)

See Simard

for more d

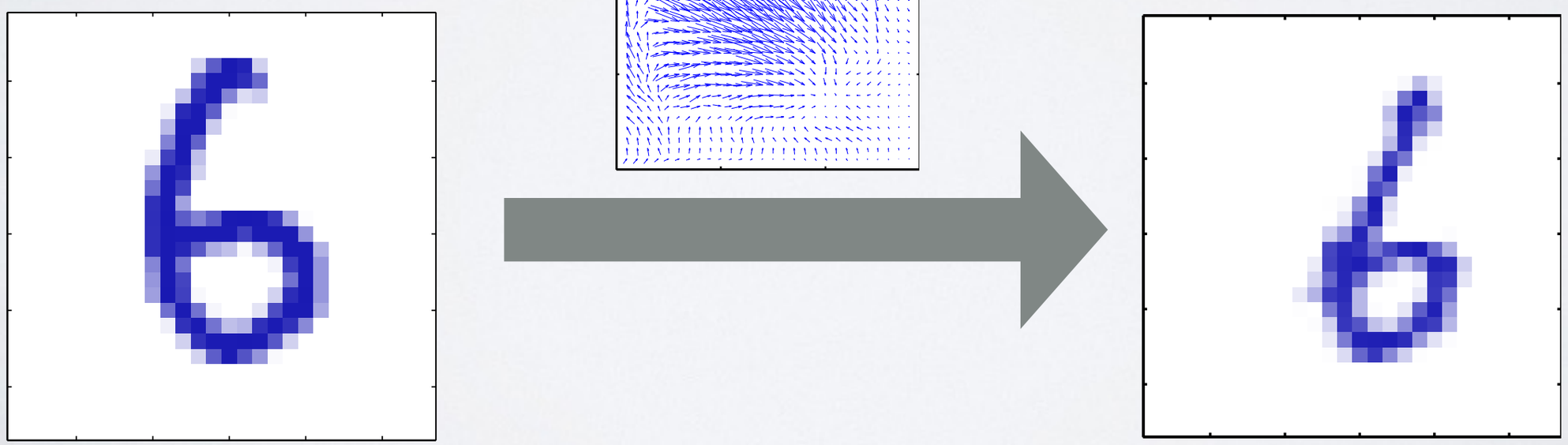
INVARIANCE BY DATA SET EXPANSION

Topics: generating additional examples, distortion field

- Can add "elastic" deformations (useful in character recognition)
- We do this by applying a "distortion field" to the image
  - a distortion field specifies where to displace each pixel value

smoothed

random distortion



(from Bishop's book)

See Simard

for more d

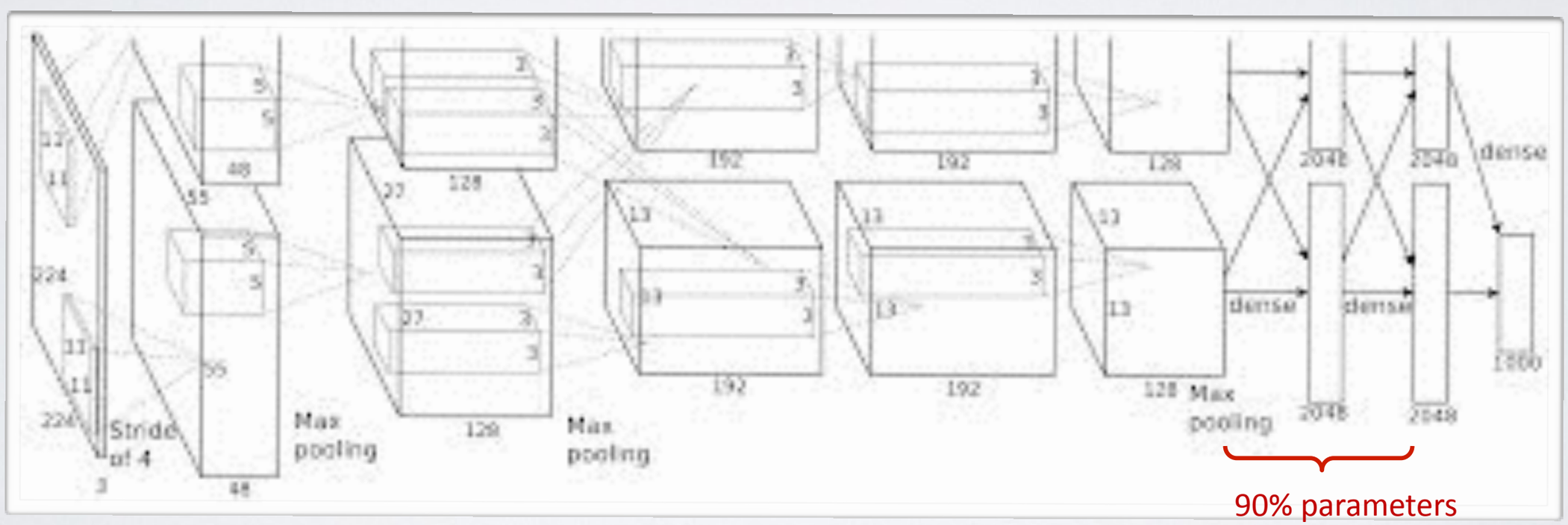
CONVOLUTIONAL NETWORK

Topics: convolutional network

- The network is trained by stochastic gradient descent
  - backpropagation is used similarly as in a fully connected network
    - ➡ need to pass gradients through the convolution operation and the pooling operation.
- Rectified linear activation functions and variants such as maxout (Goodfellow et al., 2013) are popular choices.
  - promote deeper models by allowing gradients to flow better.
- **Network-in-network** (Lin et al.; ICLR2014):
  - instead of convolving with a generalized linear unit (linear filter + nonlinear activation function), convolve a small network that includes internal hidden units.
  - Used in GoogLenet, the current state-of-the-art in the ImageNet challenge.

CONVNET IN ACTION

- SuperVision (a.k.a. AlexNet) CNN by the numbers
  - Trained on 1.2 million images, roughly 1K images for each of the 1K classes.
  - Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
  - 650,000 neurons, 60,000,000 parameters, 630,000,000 connections



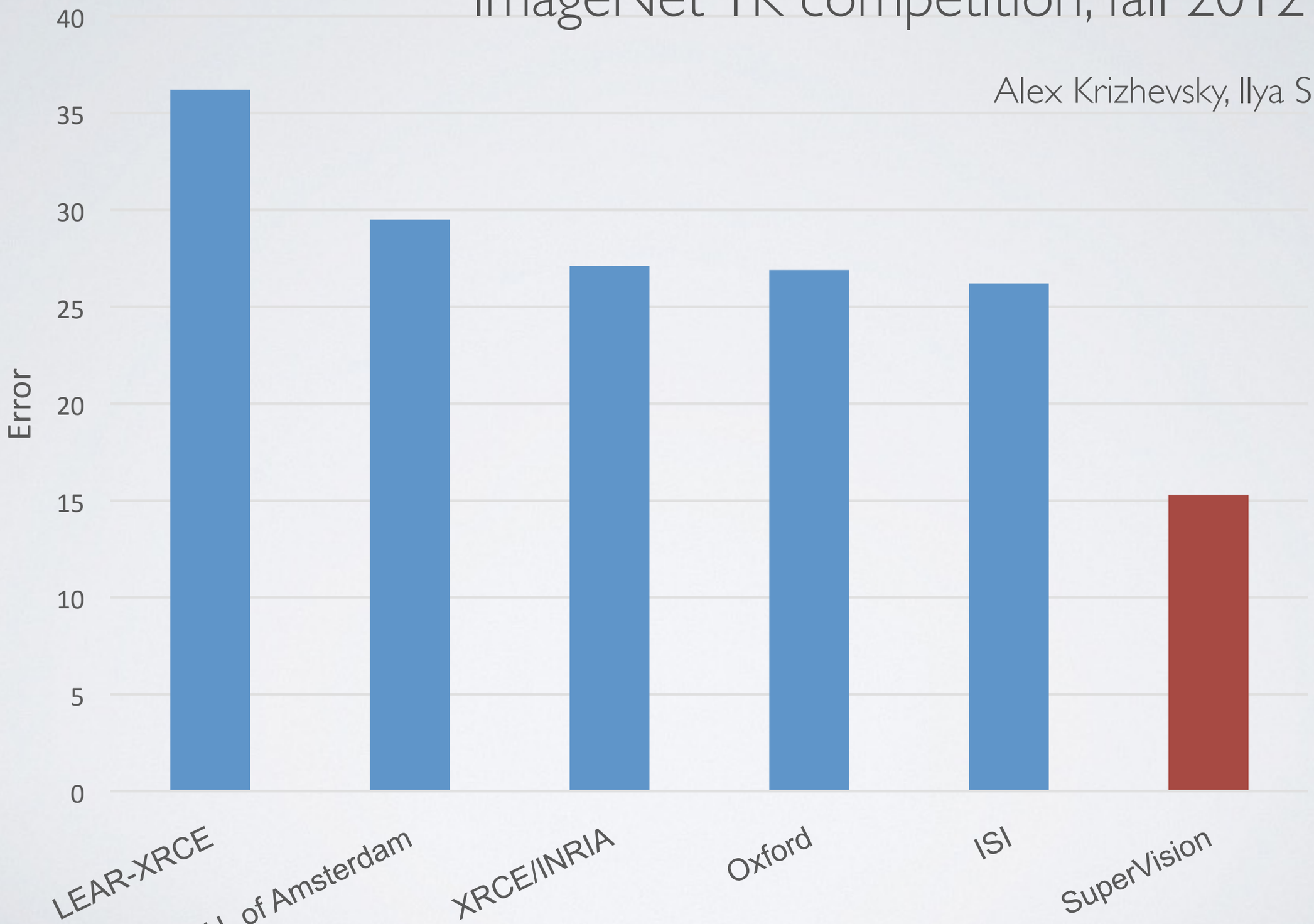
Input image

Output class prediction

CONVNET IN ACTION

ImageNet 1K competition, fall 2012

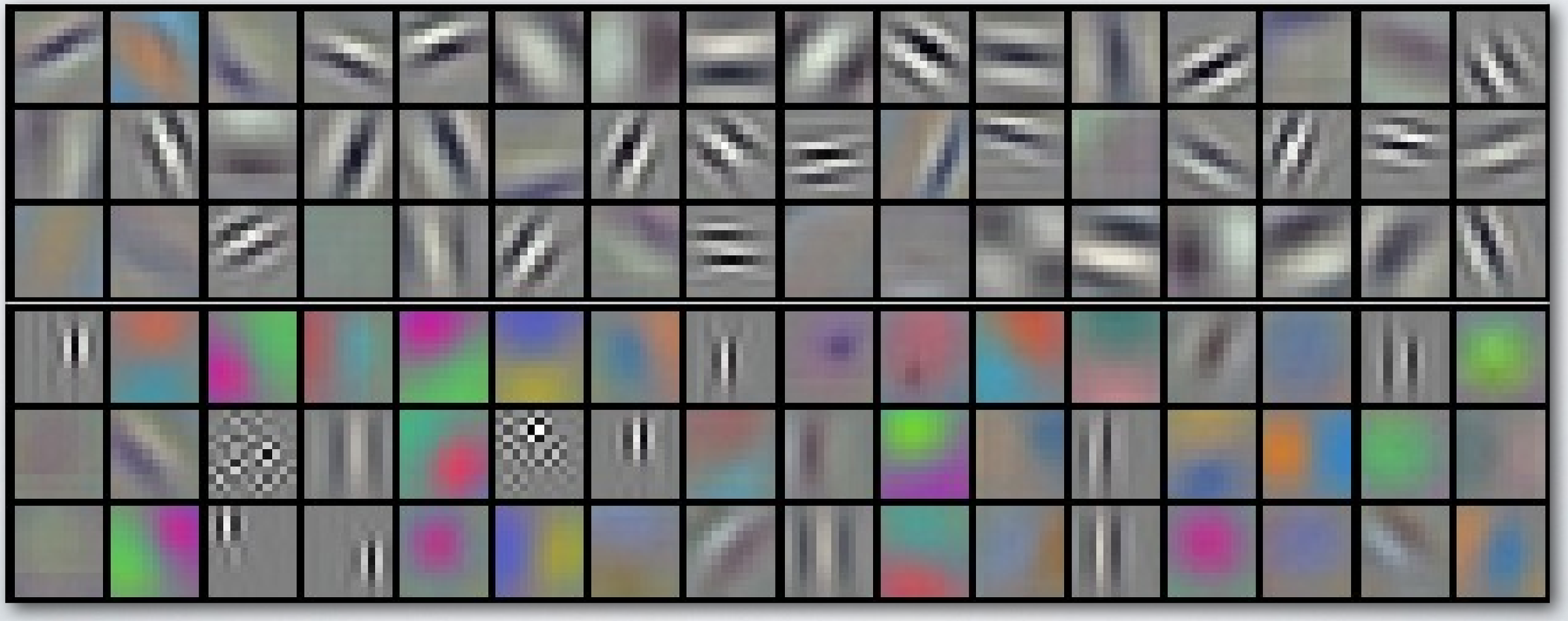
Alex Krizhevsky, Ilya Sutskever, Ge



CONVNET IN ACTION

- Training paradigm:
  - Rectified linear activation functions.
  - Trained with **Dropout**.
  - Dataset expansion (data augmentation) employed.

96 low-level learned features:



ONE YEAR LATER

ImageNet 1K competition, fall 2013

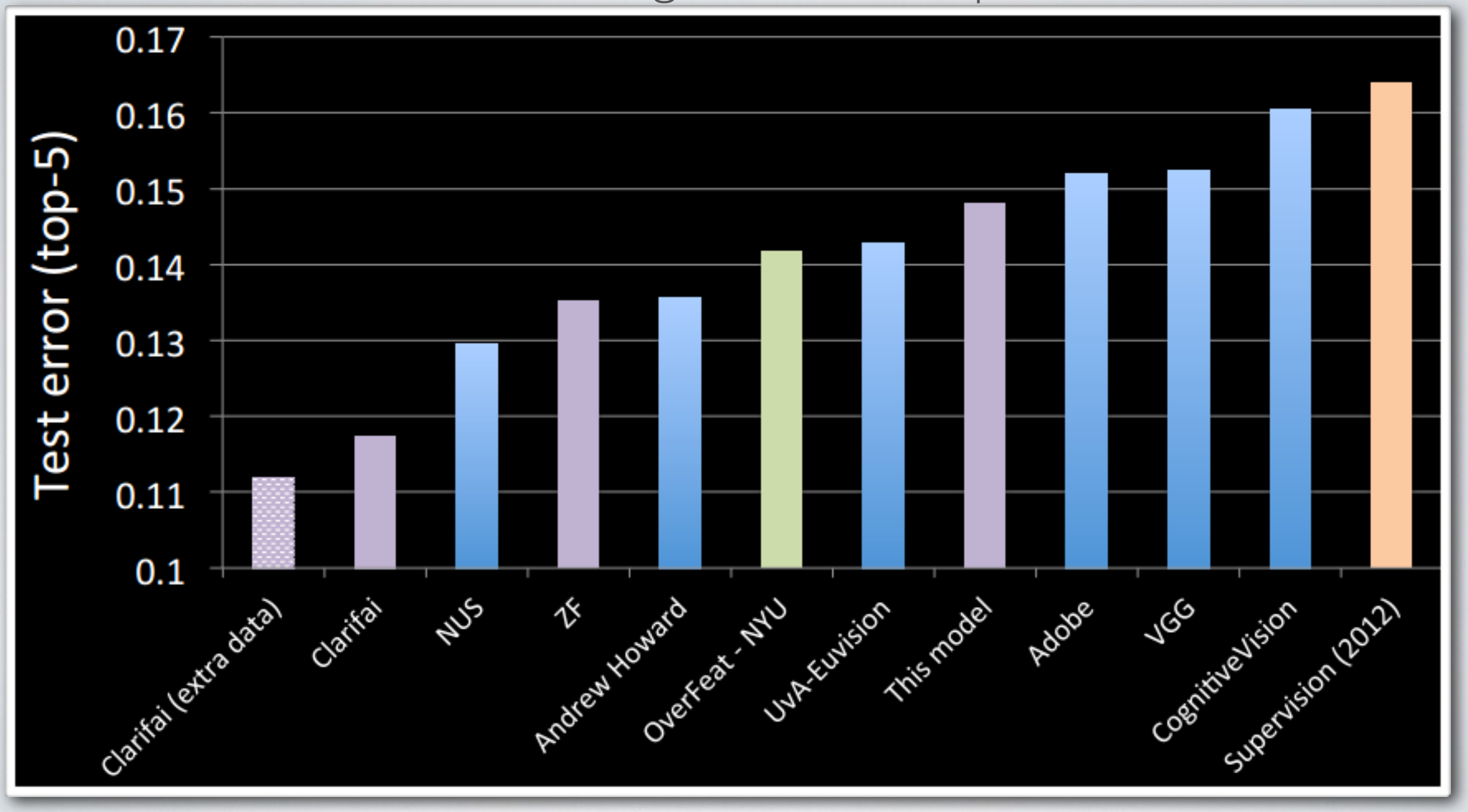
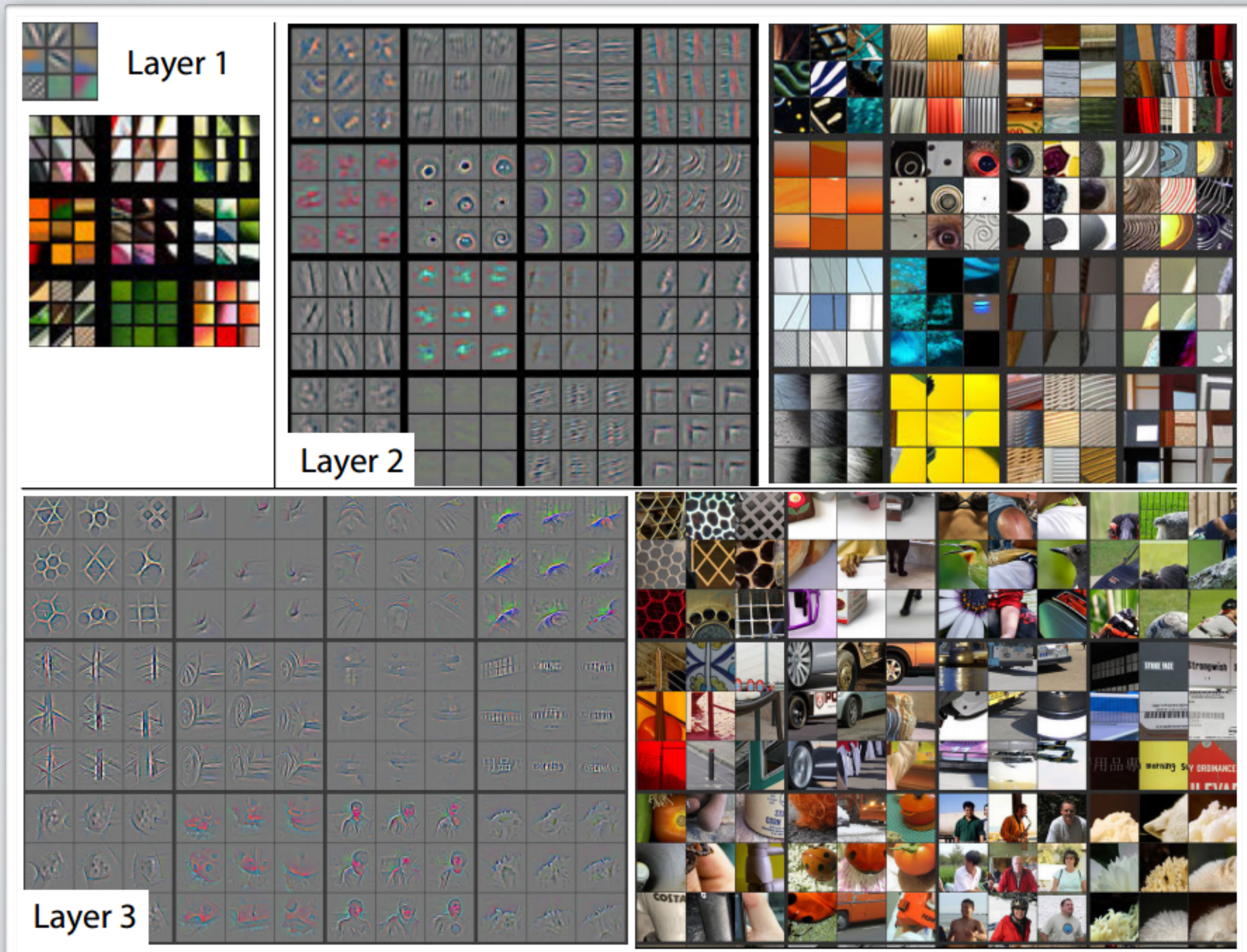


image from Li Deng, 2014

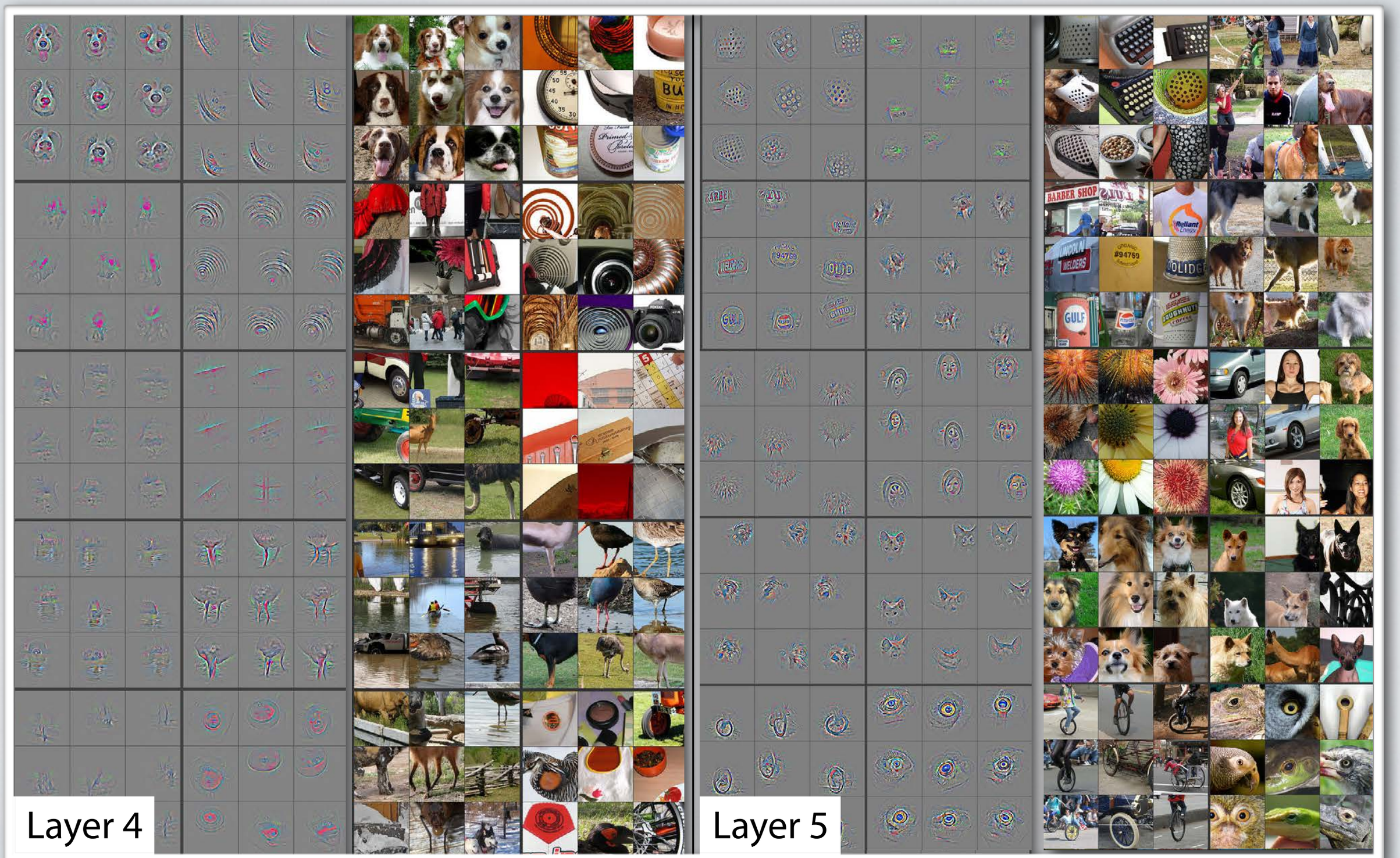
UNDERSTANDING CNNS

Image from Zeiler and Fergus, 2013



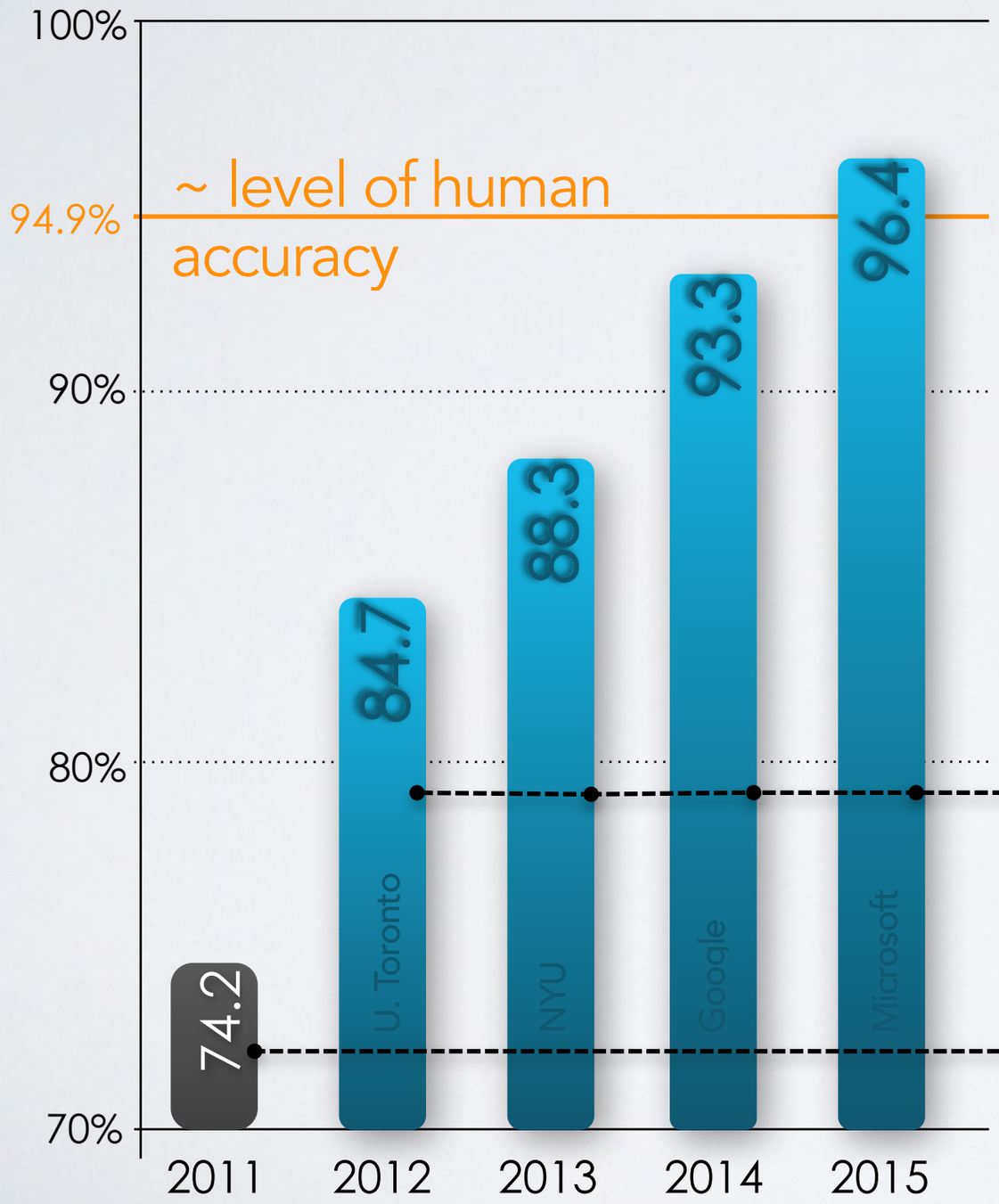
UNDERSTANDING CNNS

Image from Zeiler and



IMAGENET ACCURACY STILL IMPROVING

TOP-5 CLASSIFICATION TASK



ResNet

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Use
Deep Learning
over
Conventional
Computer Vision

RESNETS

(HE, ZHANG, REN AND SUN, 2015)

- Lastest state-of-the-art for ImageNet object recognition challenge.
- Uses “shortcut” connections that bypass the nonlinearity
  - Identity mapping for MLPs
  - Linear convolutions for ConvNets.