Convolution Neural Network

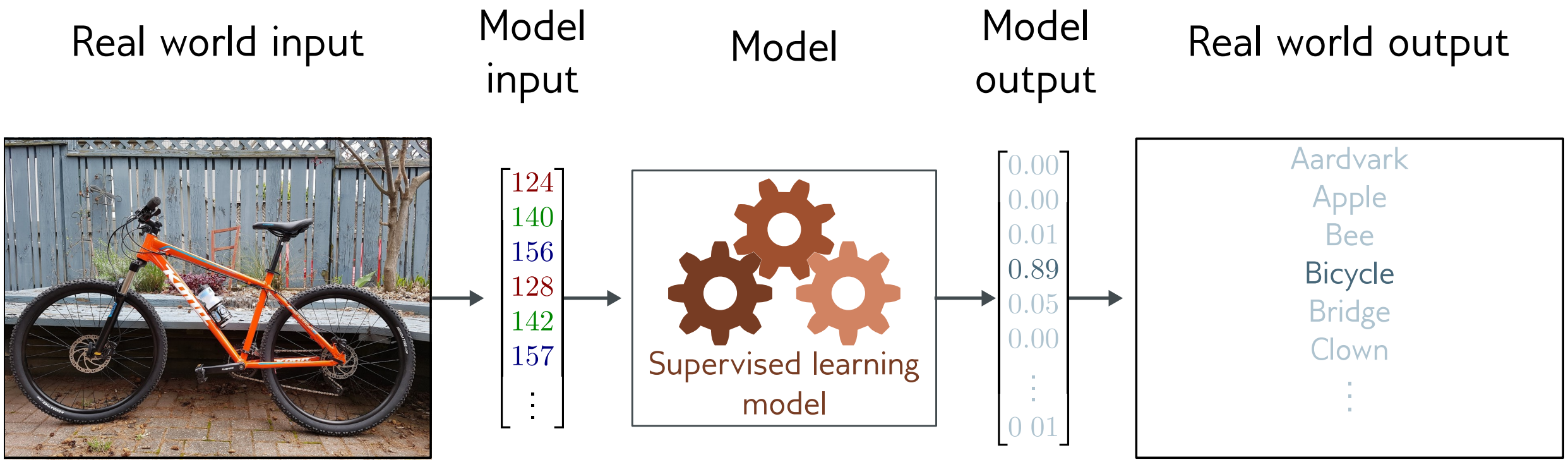
Dr. Abdul Samad

Adapted from Prof. Simon Prince

Convolutional networks

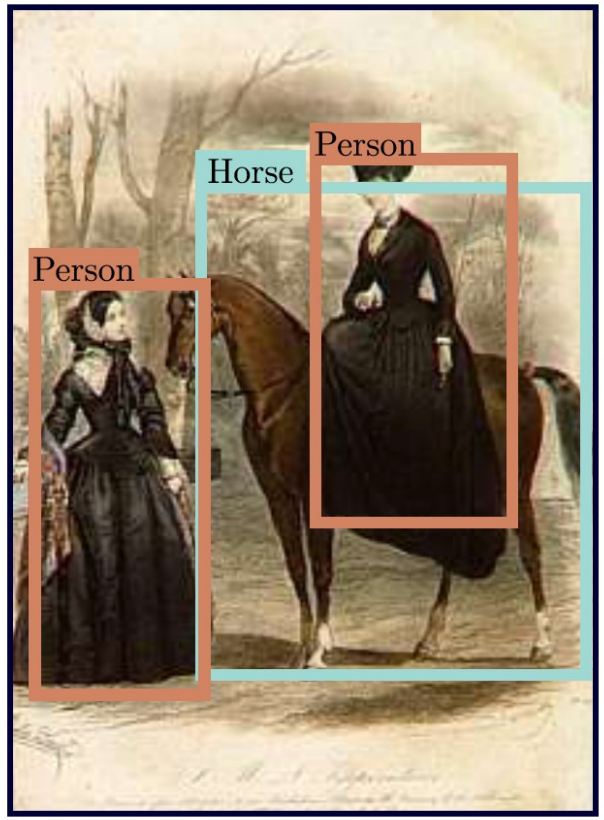
- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

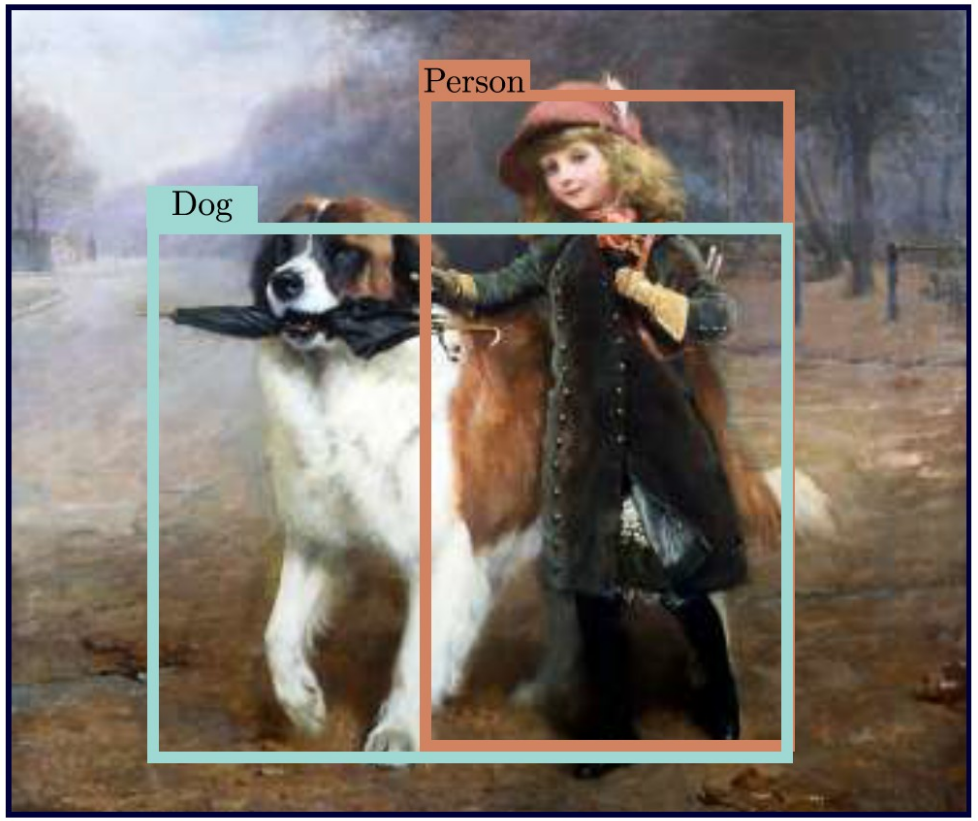
Image classification



- Multiclass classification problem (discrete classes, >2 possible classes)
- Convolutional network

Object detection





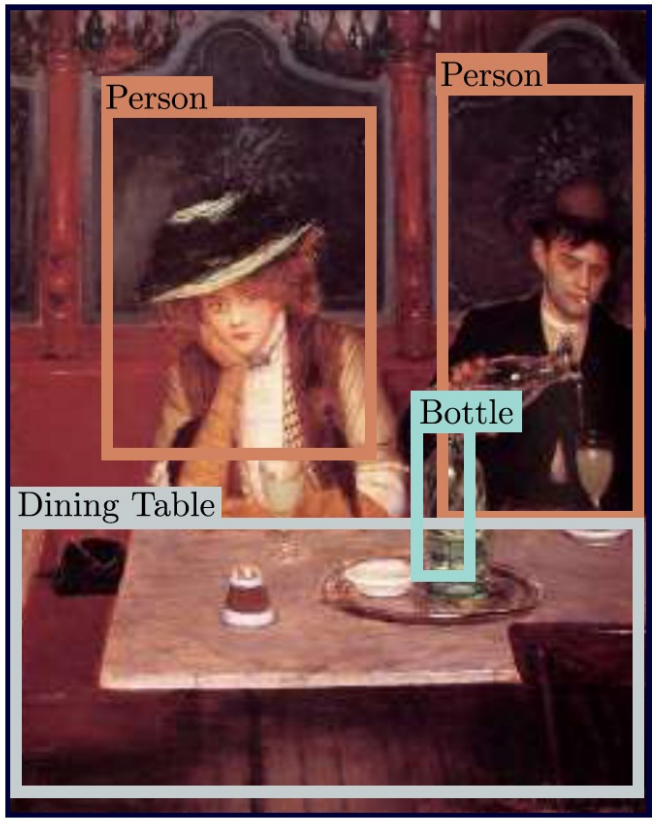
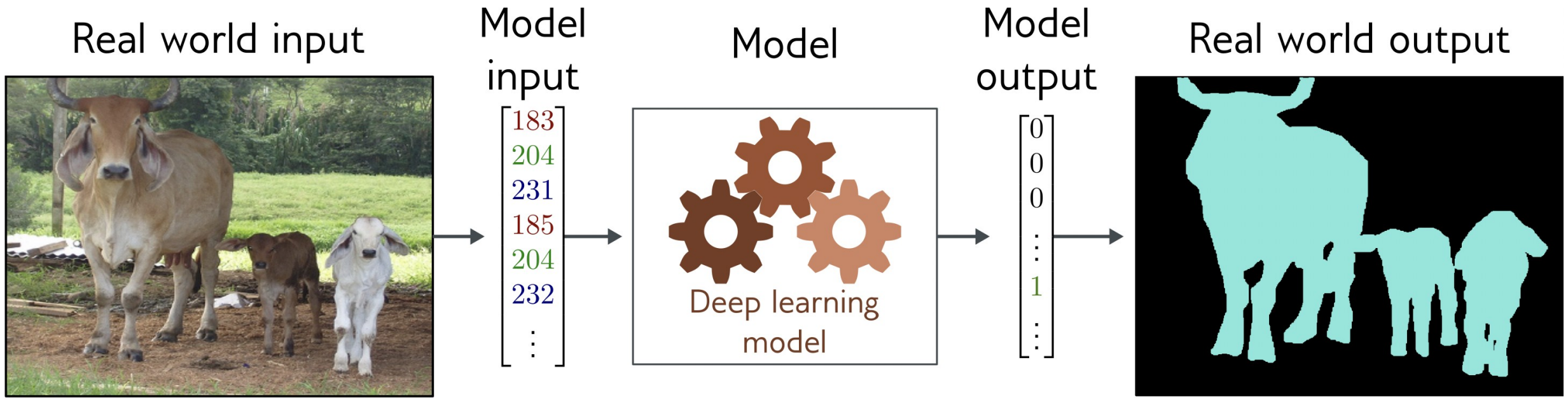


Image segmentation



- Multivariate binary classification problem (many outputs, two discrete classes)
- Convolutional encoder-decoder network

Networks for images

- Problems with fully-connected networks

1. Size
   - 224x224 RGB image = 150,528 dimensions
   - Hidden layers generally larger than inputs
   - One hidden layer = 150,520x150,528 weights -- 22 billion
2. Nearby pixels statistically related
   - But could permute pixels and relearn and get same results with FC
3. Should be stable under transformations
   - Don’t want to re-learn appearance at different parts of image

Convolutional networks

- Parameters only look at local image patches
- Share parameters across image

Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

Invariance

- A function f[x] is invariant to a transformation t[] if:

$$\mathbf{f}[\mathbf{t}[\mathbf{x}]] = \mathbf{f}[\mathbf{x}]$$

i.e., the function output is the same even after the transformation is applied.

Invariance example

e.g., Image classification

- Image has been translated, but we want our classifier to give the same result

Equivariance

- A function f[x] is equivariant to a transformation t[] if:

$$\mathbf{f}[\mathbf{t}[\mathbf{x}]] = \mathbf{t}[\mathbf{f}[\mathbf{x}]]$$

i.e., the output is transformed in the same way as the input

Equivariance example

e.g., Image segmentation

- Image has been translated and we want segmentation to translate with it




Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

Convolution* in 1D

- Input vector **x:**

$$\mathbf{x} = [x_1, x_2, \ldots, x_I]$$

- Output is weighted sum of neighbors:

$$z_i = \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}$$

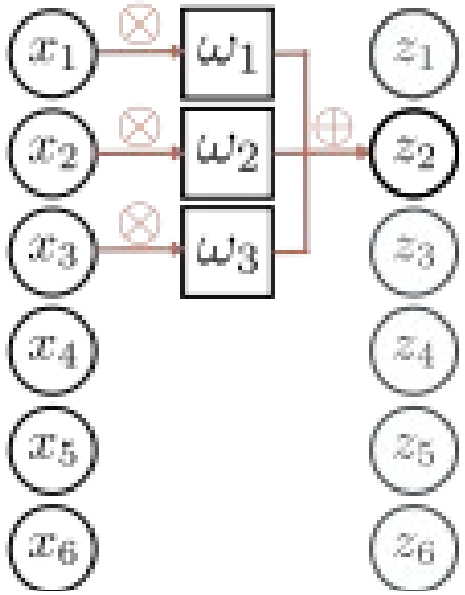
- Convolutional kernel or filter:

$$\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T$$

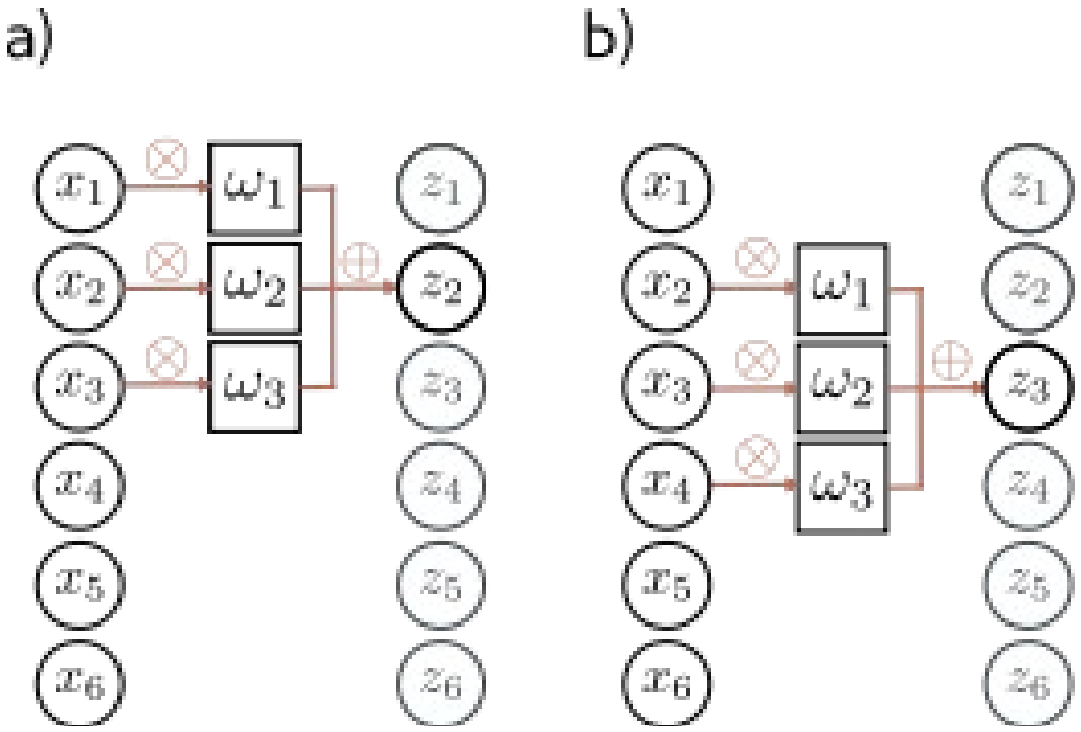
Kernel size = 3

Convolution with kernel size 3

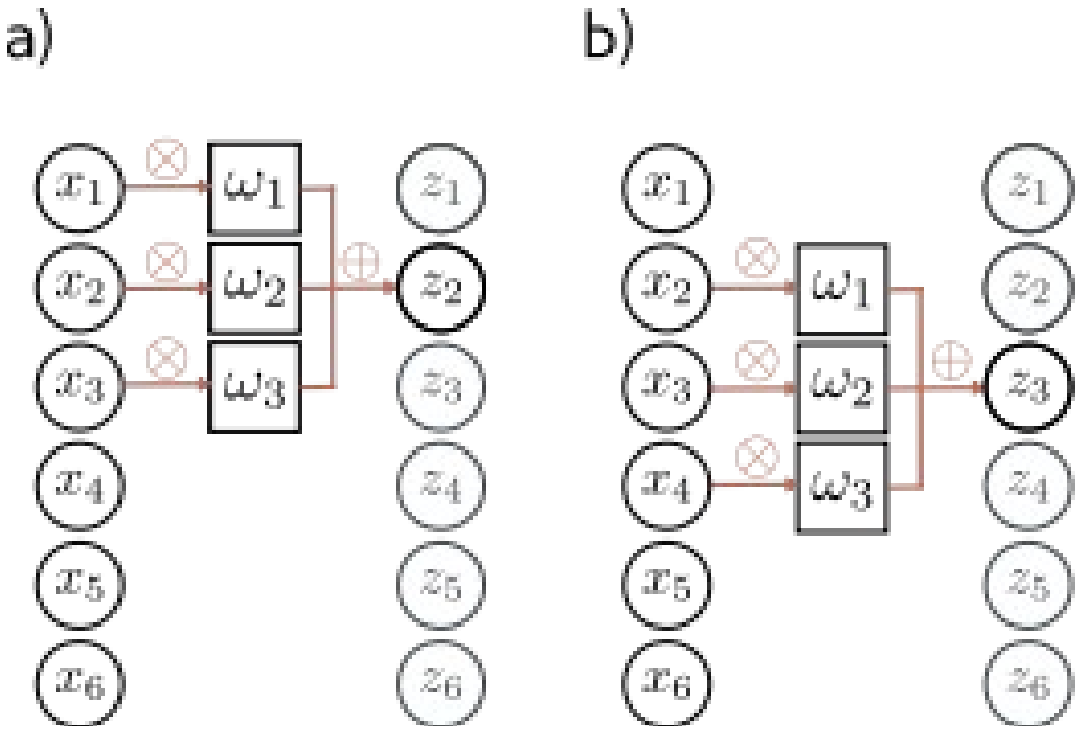
a)



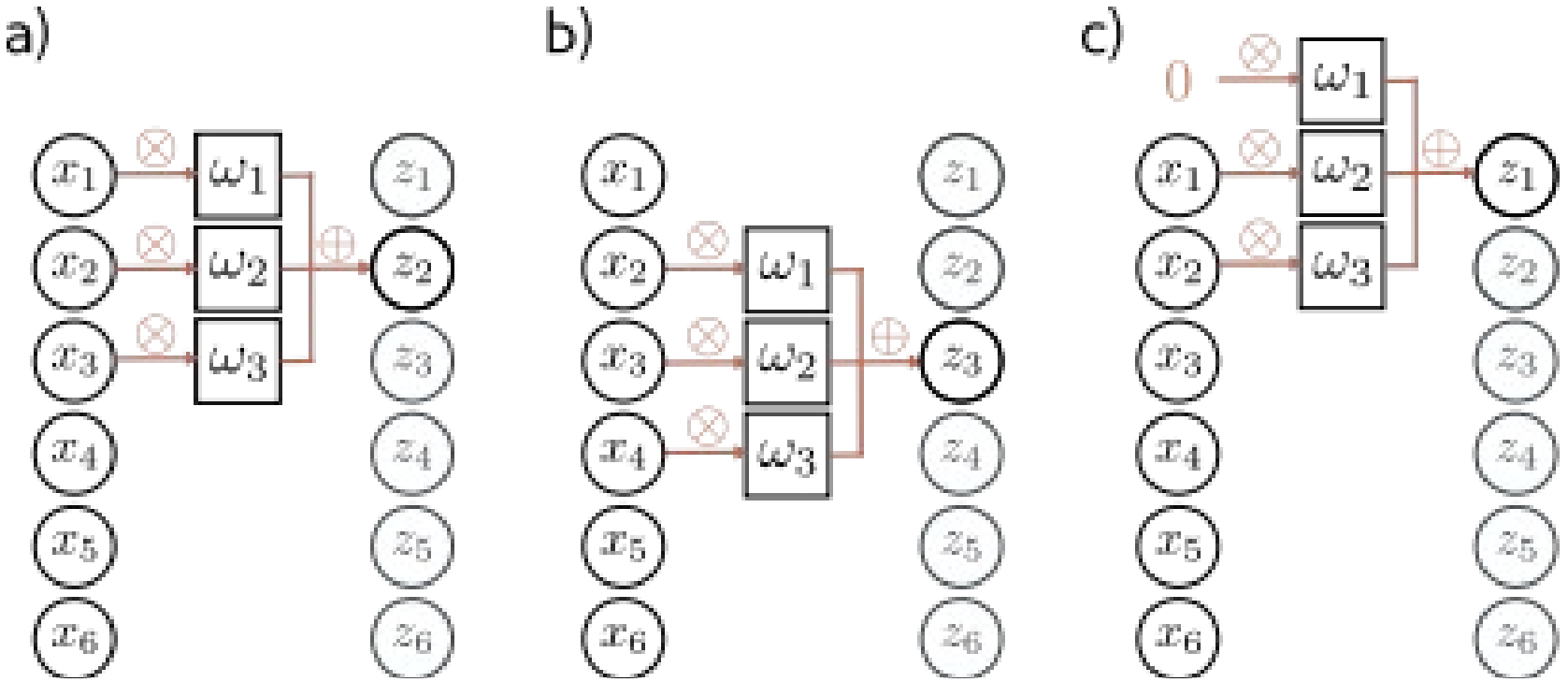
Convolution with kernel size 3



Convolution with kernel size 3

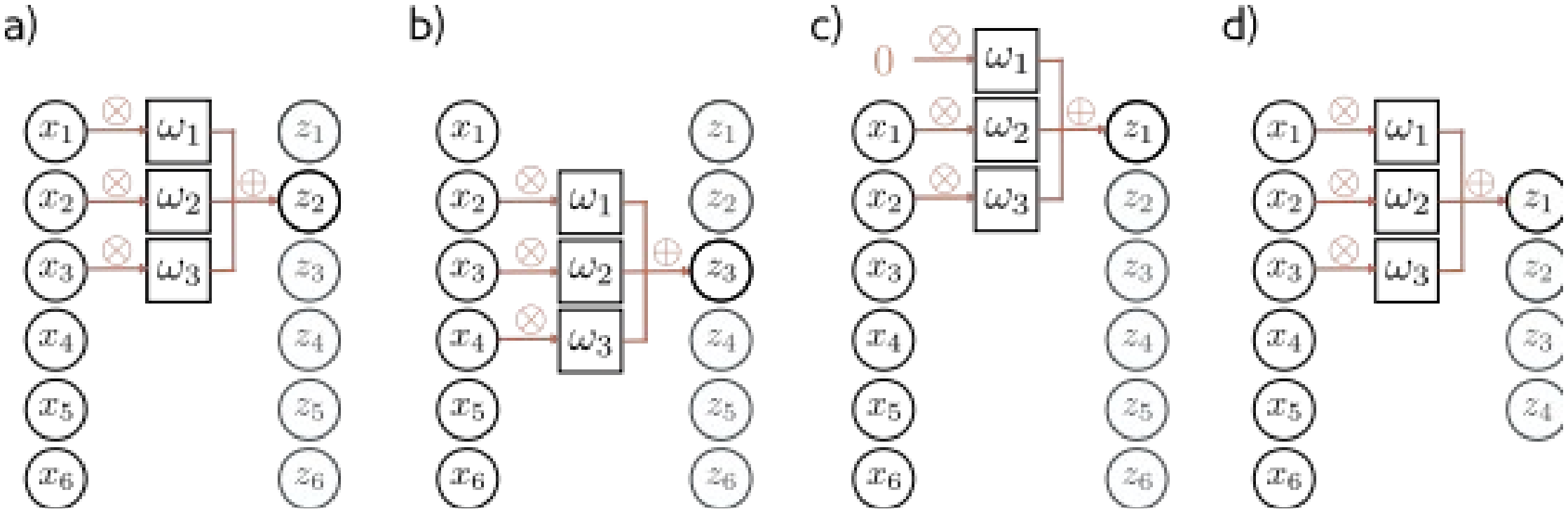


Zero padding



Treat positions that are beyond end of the input as zero.

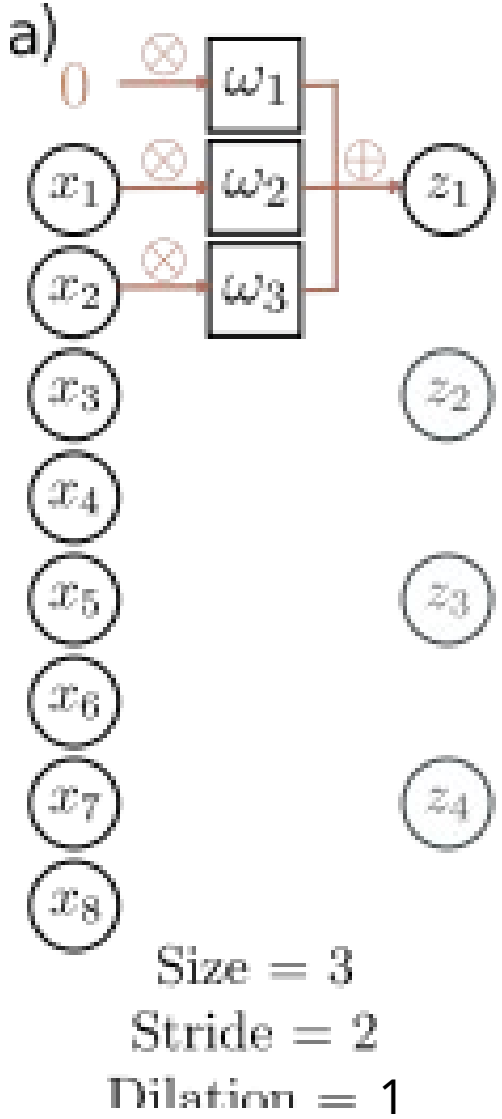
“Valid” convolutions

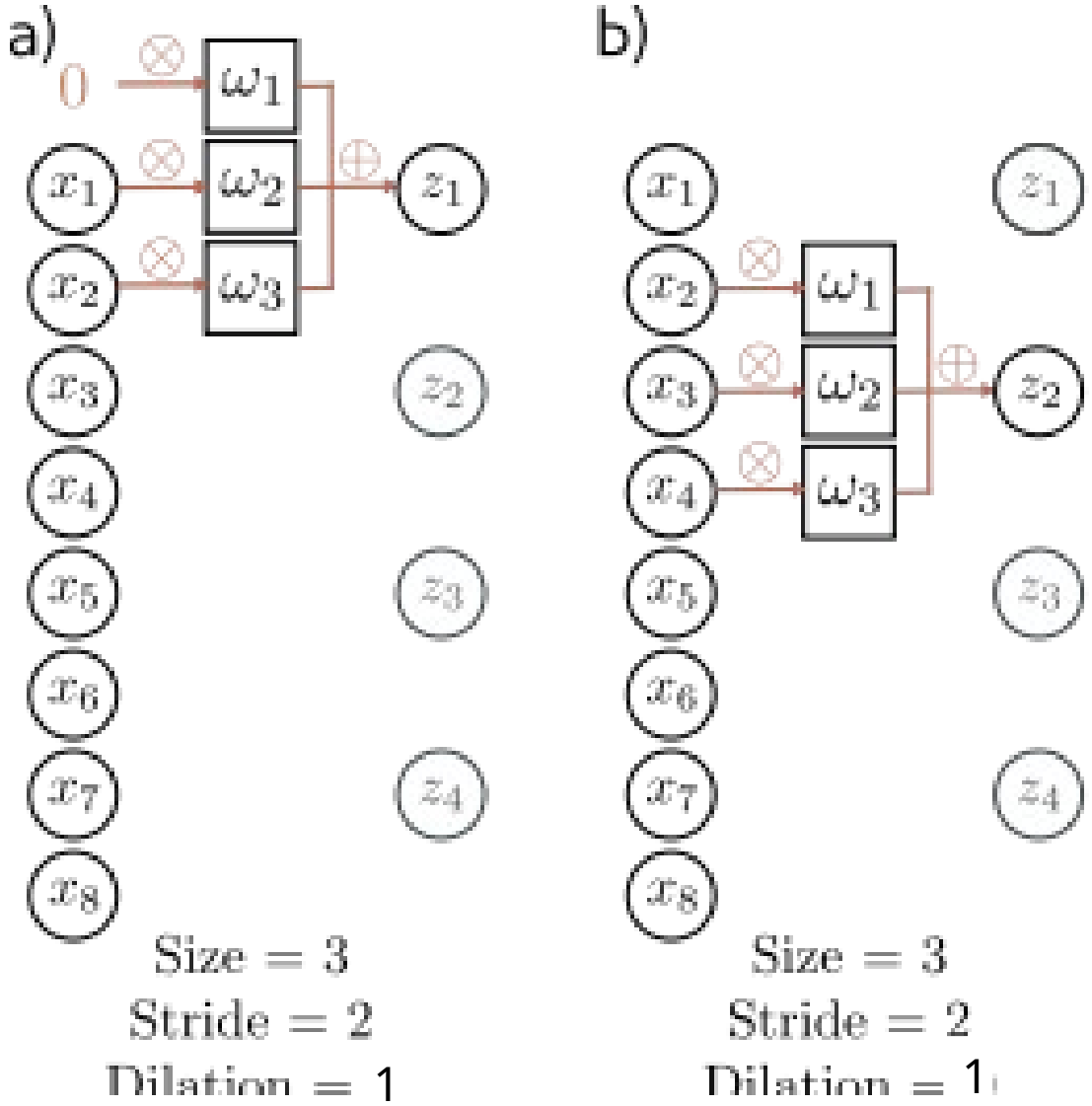


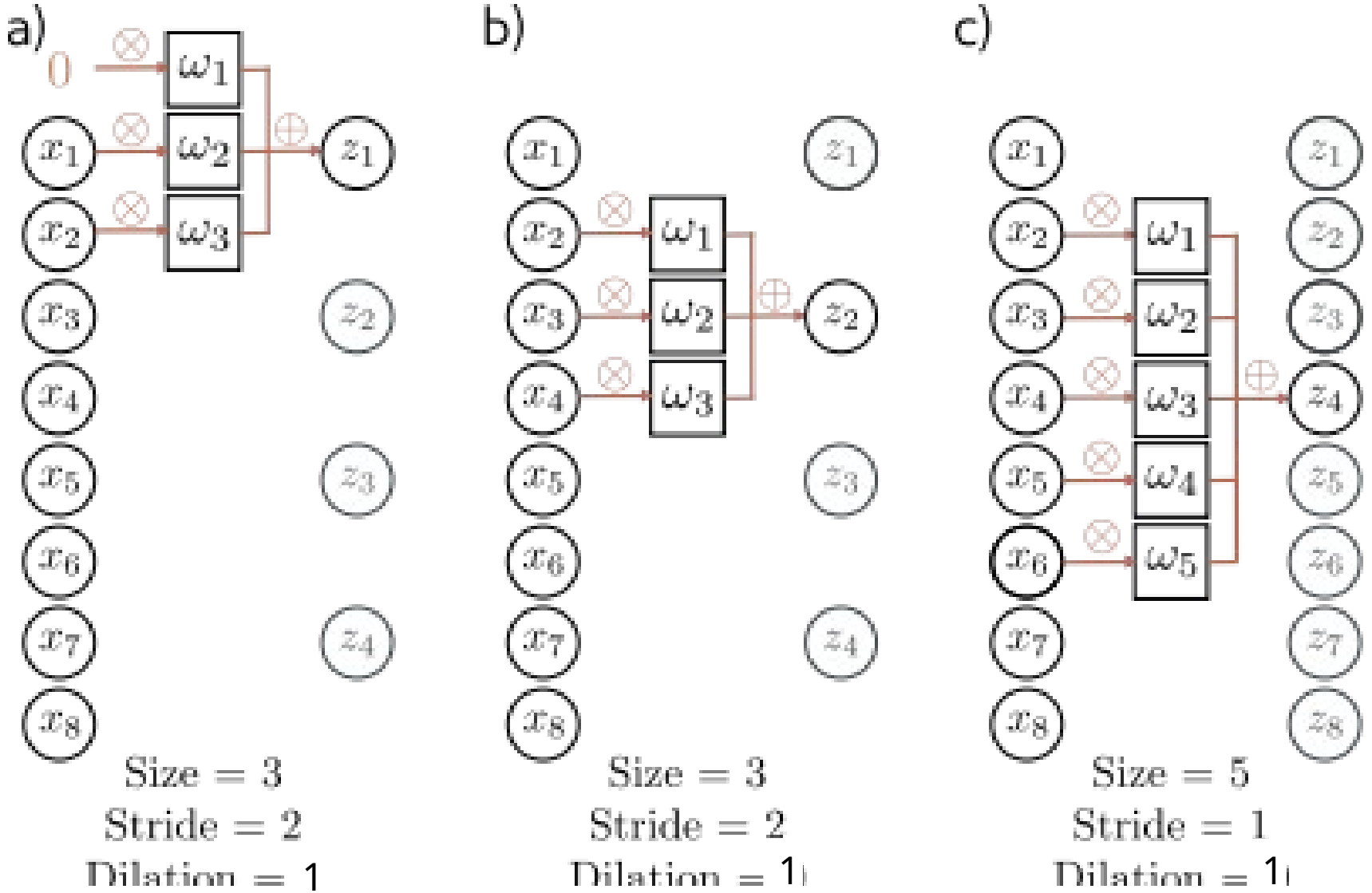
Only process positions where kernel falls in image (smaller output).

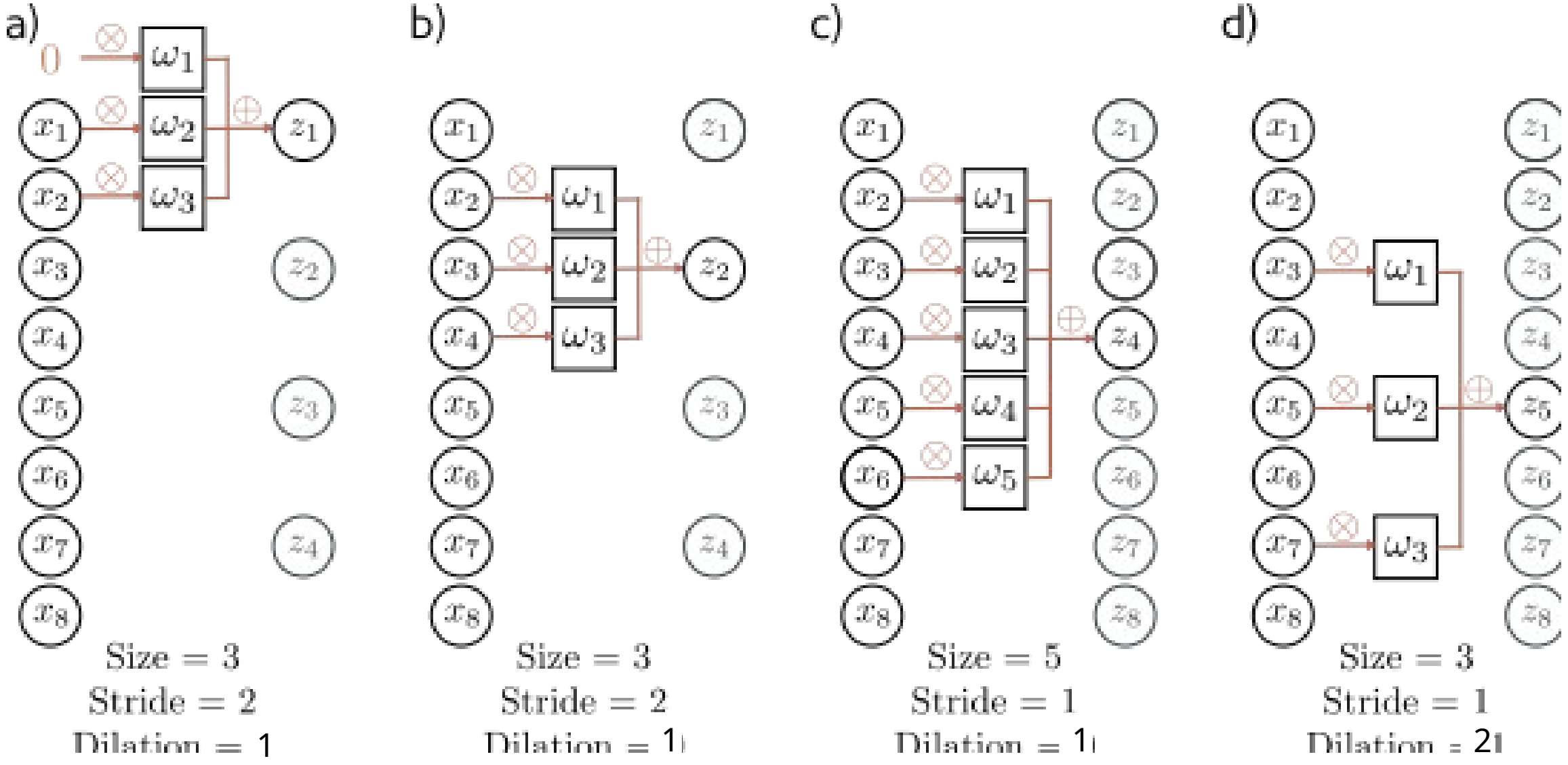
Stride, kernel size, and dilation

- Stride = shift by k positions for each output
  - Decreases size of output relative to input
- Kernel size = weight a different number of inputs for each output
  - Combine information from a larger area
  - But kernel size 5 uses 5 parameters
- Dilated or atrous convolutions = intersperse kernel values with zeros
  - Combine information from a larger area
  - Fewer parameters


a)
0
ω_1
ω_2
ω_3
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
z_1
z_2
z_3
z_4
Size = 3
Stride = 2
Dilation = 1


a)
0
ω_1
ω_2
ω_3
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
z_1
z_2
z_3
z_4
Size = 3
Stride = 2
Dilation = 1
b)
ω_1
ω_2
ω_3
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
z_1
z_2
z_3
z_4
Size = 3
Stride = 2
Dilation = 1


a)
b)
c)
Size = 3
Stride = 2
Dilation = 1
Size = 3
Stride = 2
Dilation = 1
Size = 5
Stride = 1
Dilation = 1


a)
b)
c)
d)
Size = 3
Stride = 2
Dilation = 1
Size = 3
Stride = 2
Dilation = 1
Size = 5
Stride = 1
Dilation = 1
Size = 3
Stride = 1
Dilation = 2

Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

Convolutional layer

$$h_i = \mathrm{a}\left[\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}\right]$$

$$= \mathrm{a}\left[\beta + \sum_{j=1}^{3} \omega_j x_{i+j-2}\right]$$

Special case of fully-connected network

Convolutional network:

$$h_i = \mathrm{a}\left[\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}\right]$$

$$= \mathrm{a}\left[\beta + \sum_{j=1}^{3} \omega_j x_{i+j-2}\right]$$

Fully connected network:

$$h_i = \mathrm{a}\left[\beta_i + \sum_{j=1}^{D} \omega_{ij} x_j\right]$$

Special case of fully-connected network

Convolutional network:

$$h_i = \mathrm{a}\left[\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}\right]$$

$$= \mathrm{a}\left[\beta + \sum_{j=1}^{3} \omega_j x_{i+j-2}\right]$$

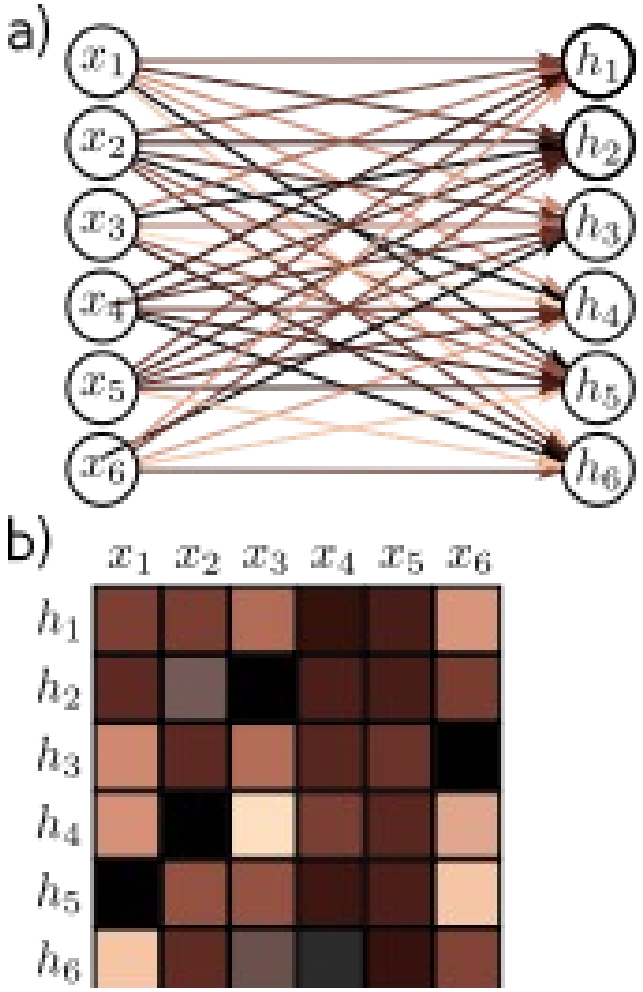
3 weights, 1 bias

Fully connected network:

$$h_i = \mathrm{a}\left[\beta_i + \sum_{j=1}^{D} \omega_{ij} x_j\right]$$

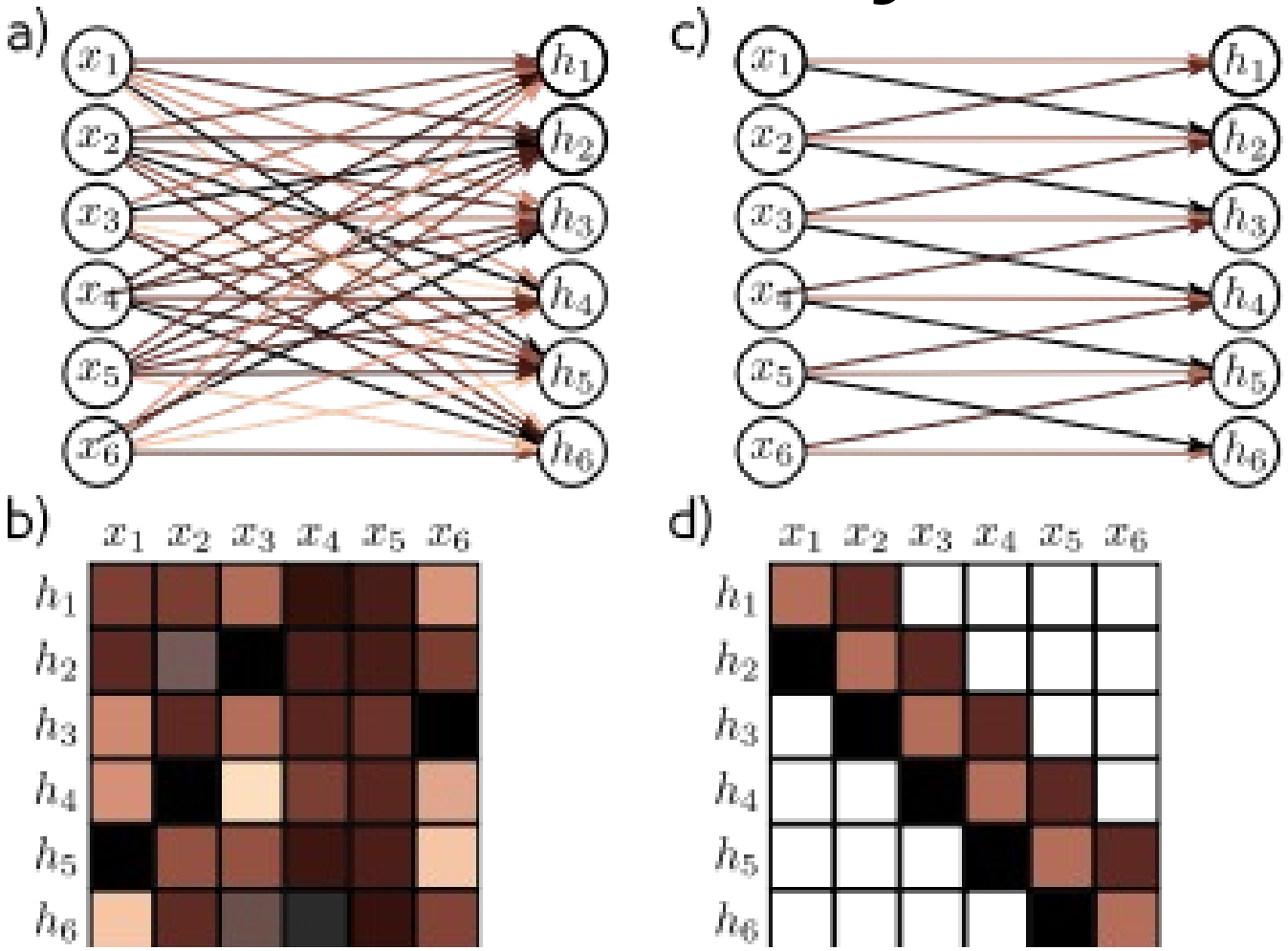
weights, D biases

Special case of fully-connected network



Fully connected network

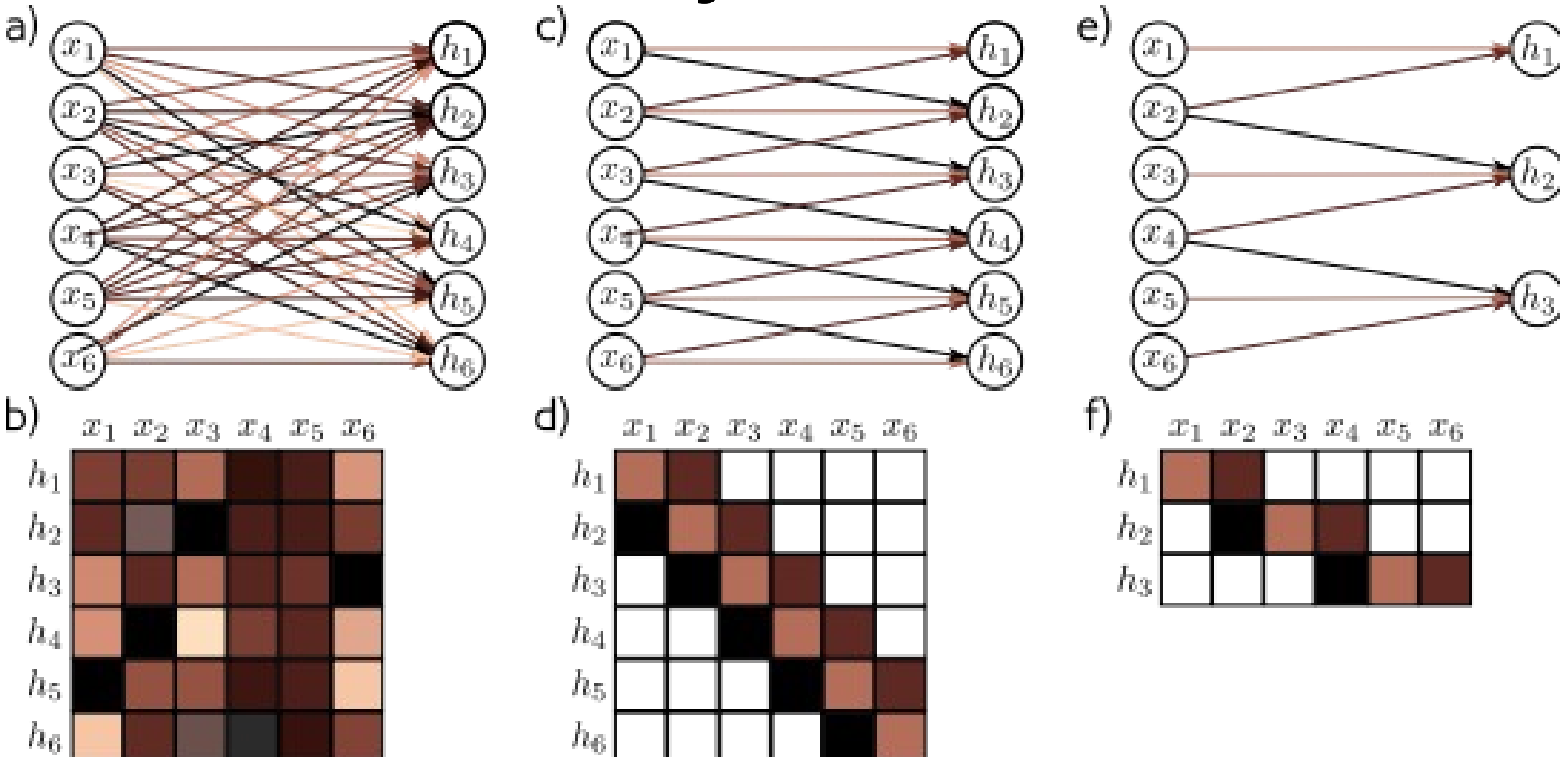
Special case of fully-connected network



Fully connected network

Convolution, kernel 3,
stride 1, dilation 1

Special case of fully-connected network



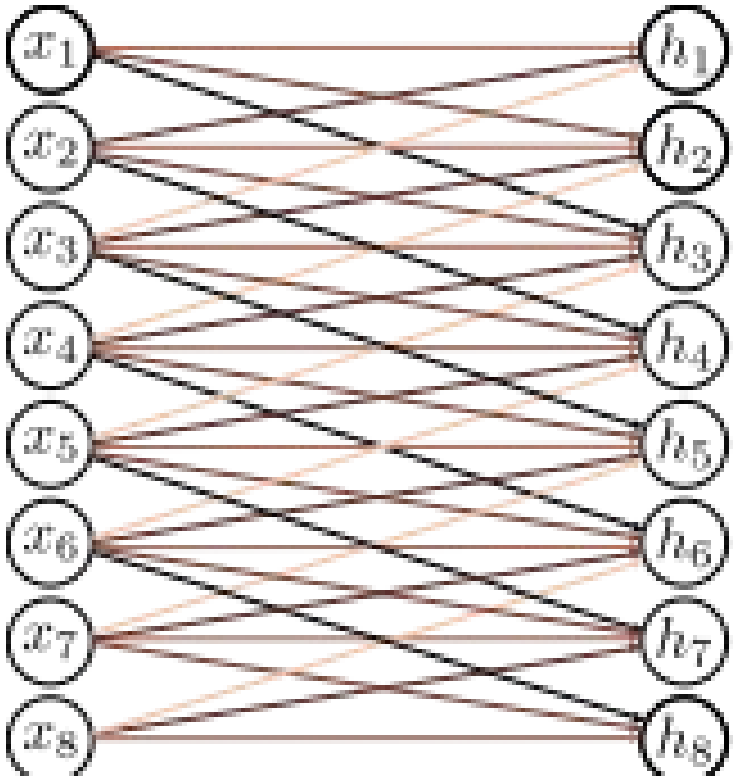
Fully connected network

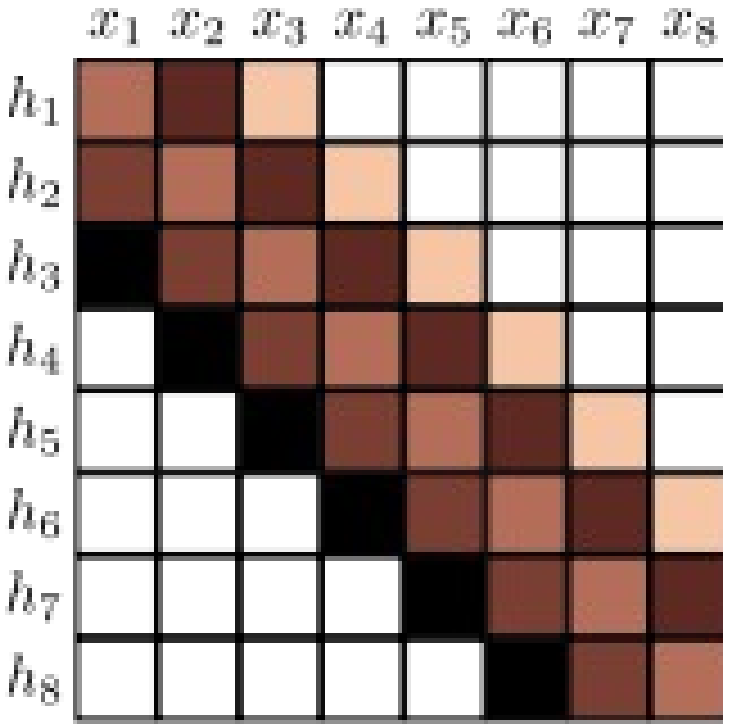
Convolution, size 3, stride 1, dilation 1, zero padding

Convolution, size 3, stride 2, dilation 1, zero padding

Question 1

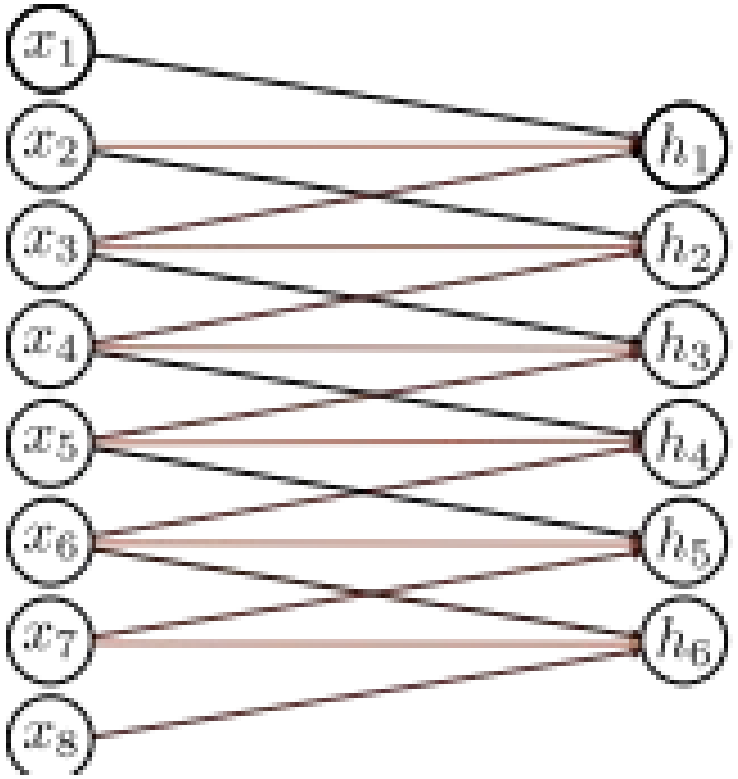
- Kernel size?
- Stride?
- Dilation?
- Zero padding / valid?

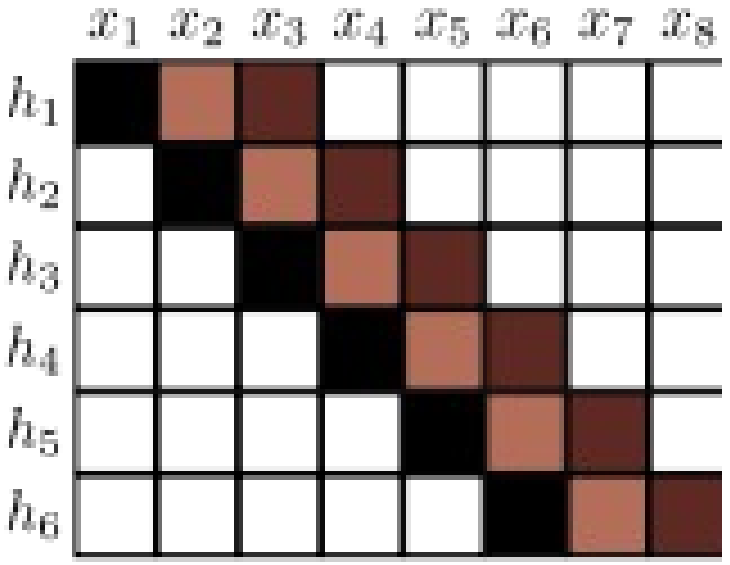




Question 2

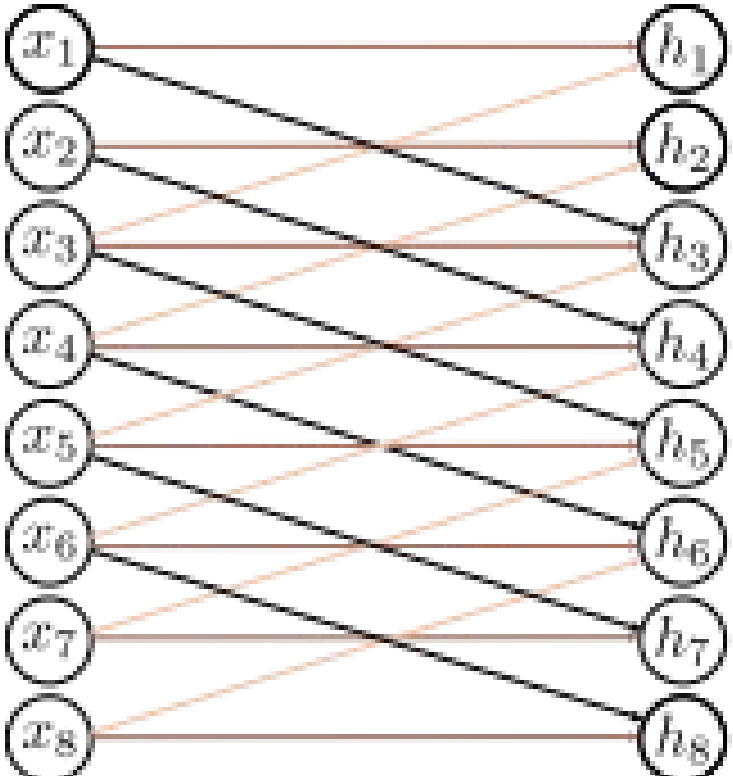
- Kernel size?
- Stride?
- Dilation?
- Zero padding / valid?

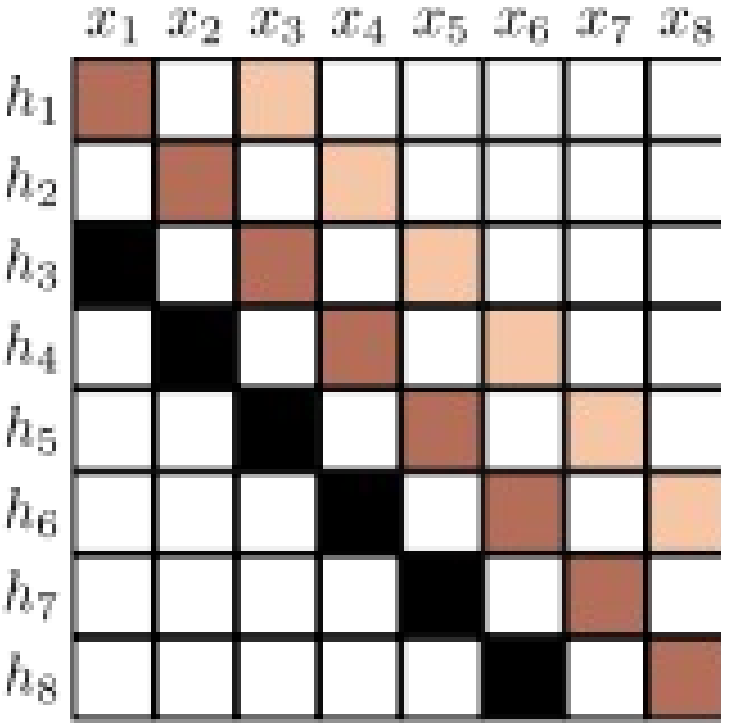




Question 3

- Kernel size?
- Stride?
- Dilation?
- Zero padding / valid?





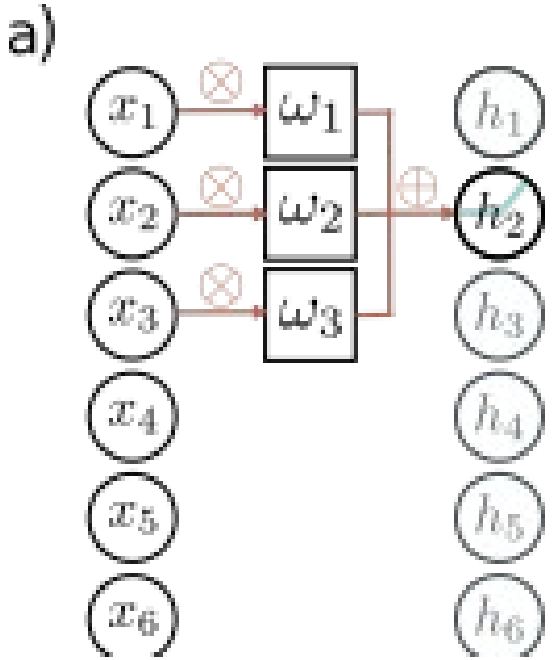
Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

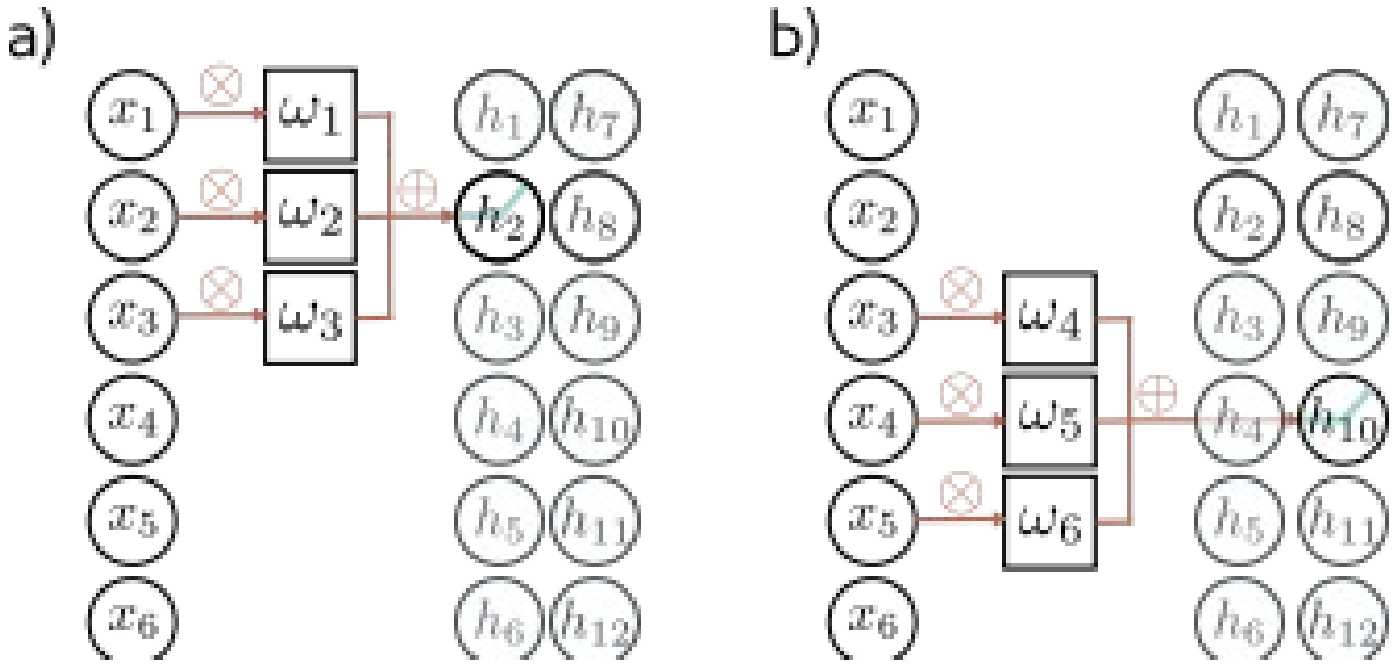
Channels

- The convolutional operation averages together the inputs
- Plus passes through ReLU function
- Has to lose information
- Solution:
  - apply several convolutions and stack them in channels
  - Sometimes also called feature maps

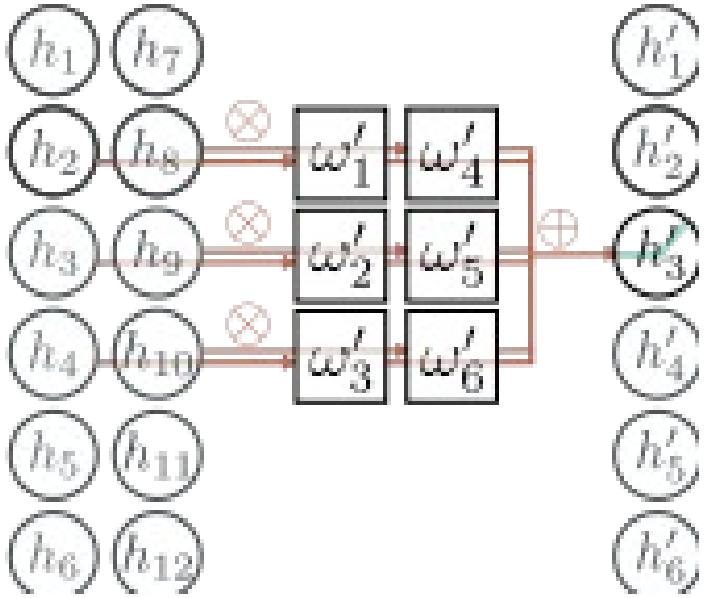
Two output channels, one input channel



Two output channels, one input channel



Two input channels, one output channel



How many parameters?

- If there are input channels and kernel size K

$$\boldsymbol{\Omega} \in \mathbb{R}^{C_i \times K} \qquad \boldsymbol{\beta} \in \mathbb{R}$$

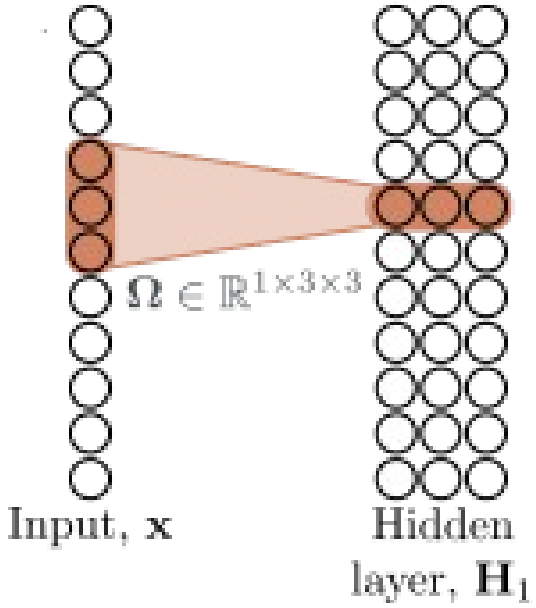
- If there are input channels and output channels

$$\boldsymbol{\Omega} \in \mathbb{R}^{C_i \times C_o \times K} \qquad \boldsymbol{\beta} \in \mathbb{R}^{C_o}$$

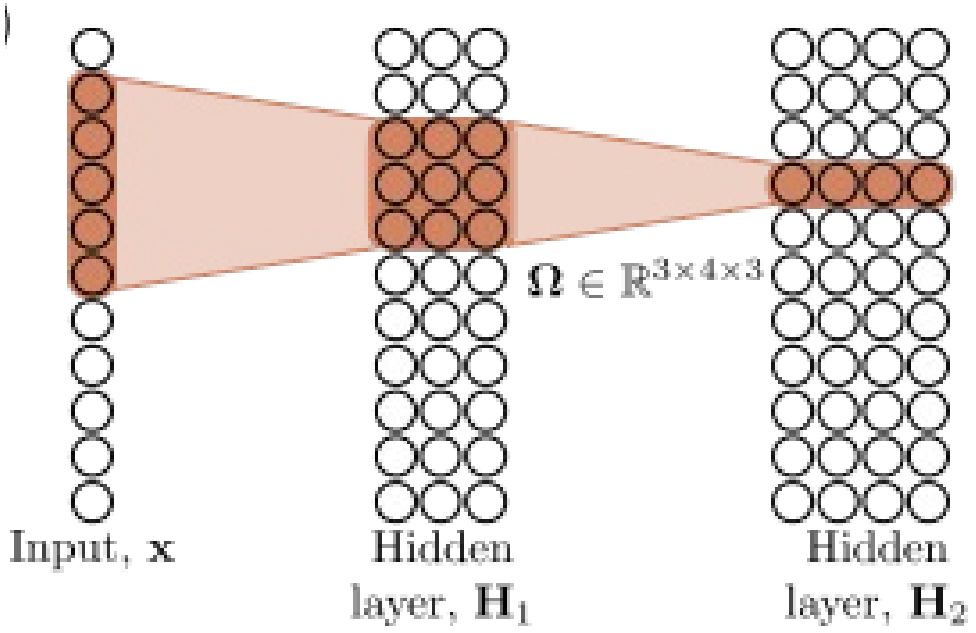
Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

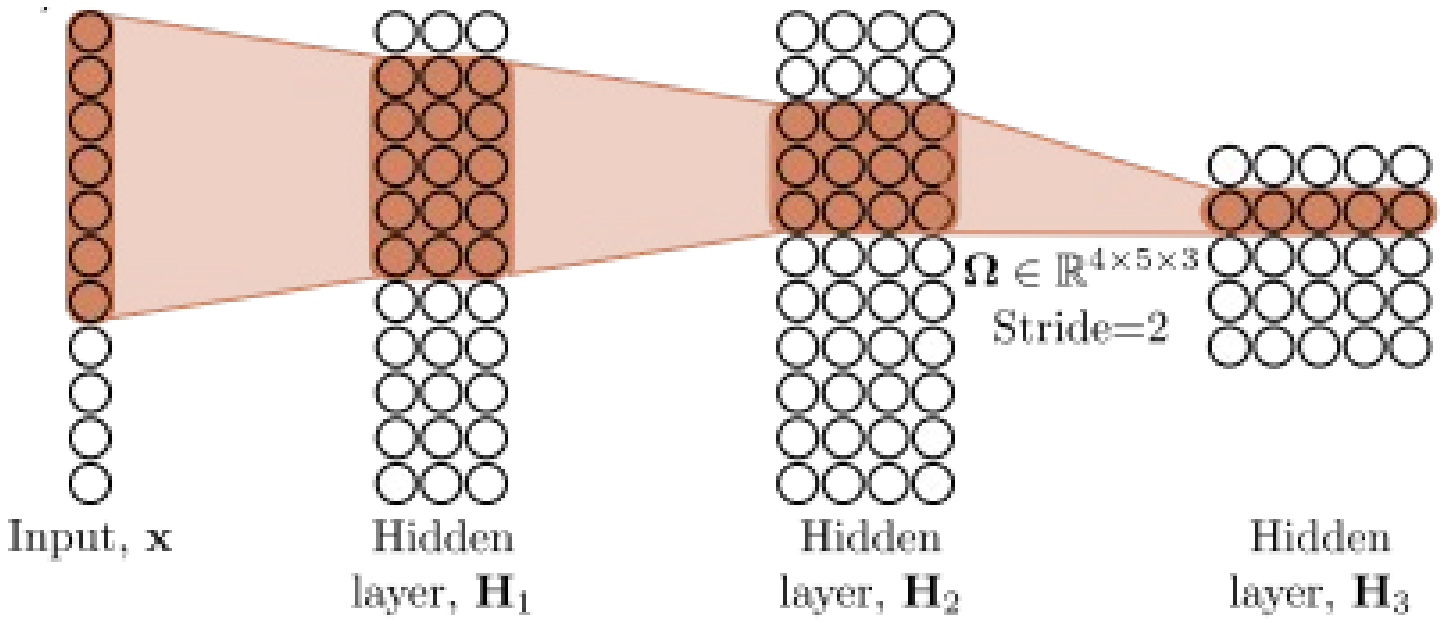
Receptive fields



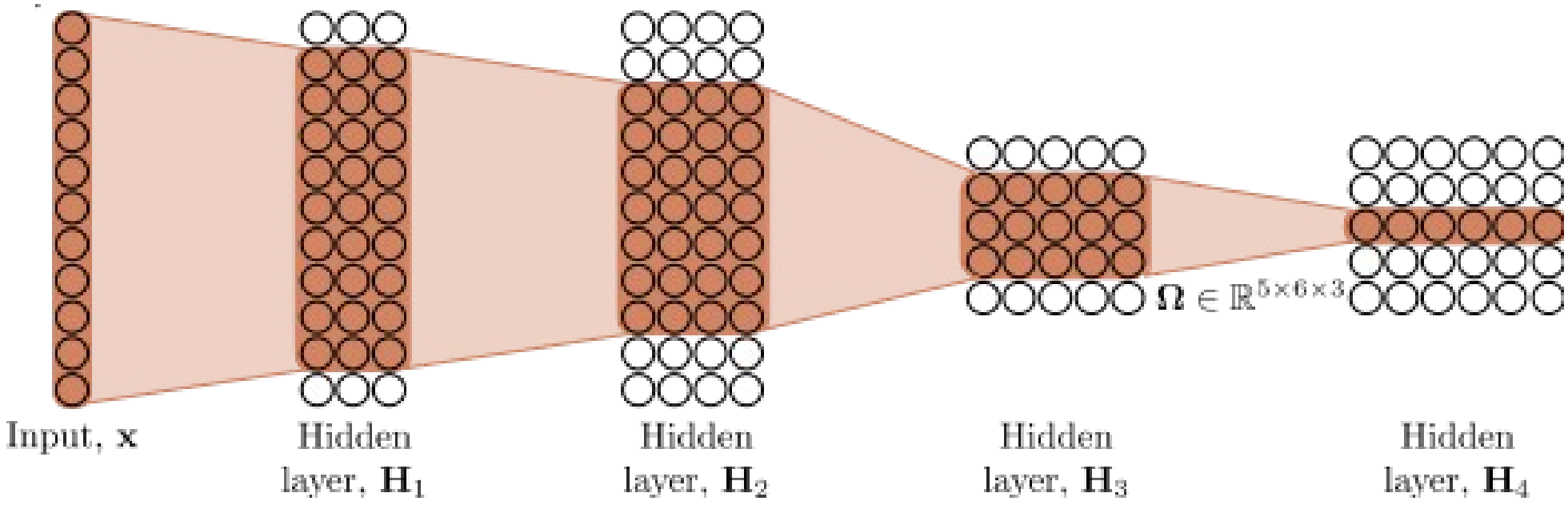
Receptive fields



Receptive fields



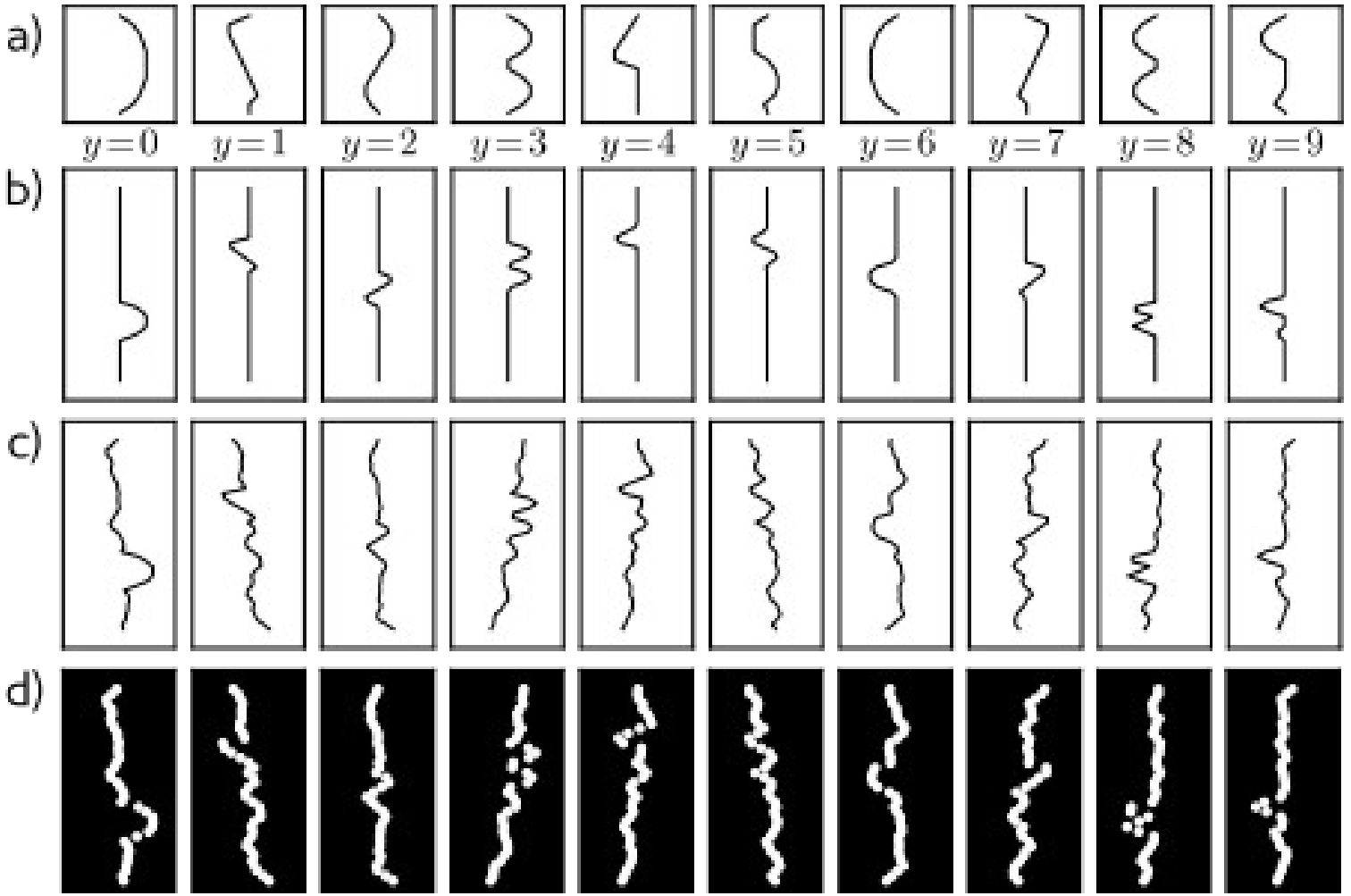
Receptive fields



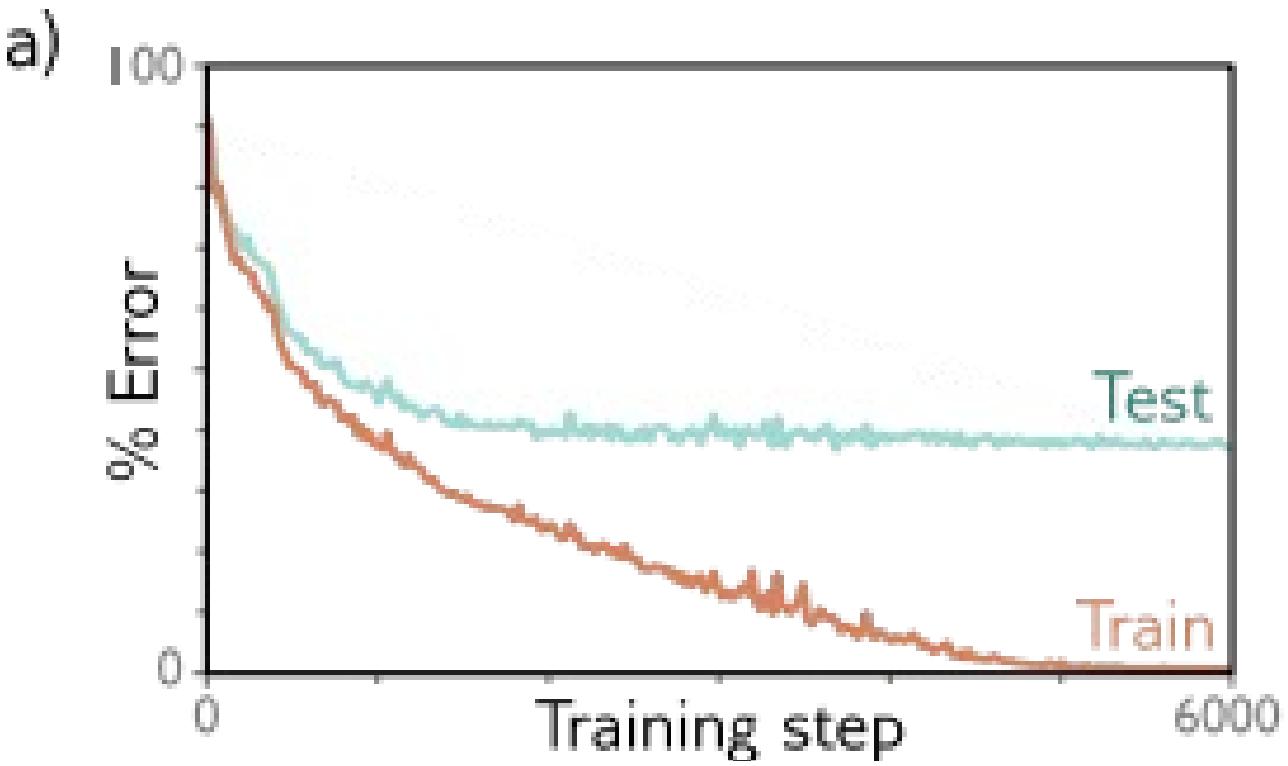
Convolutional networks

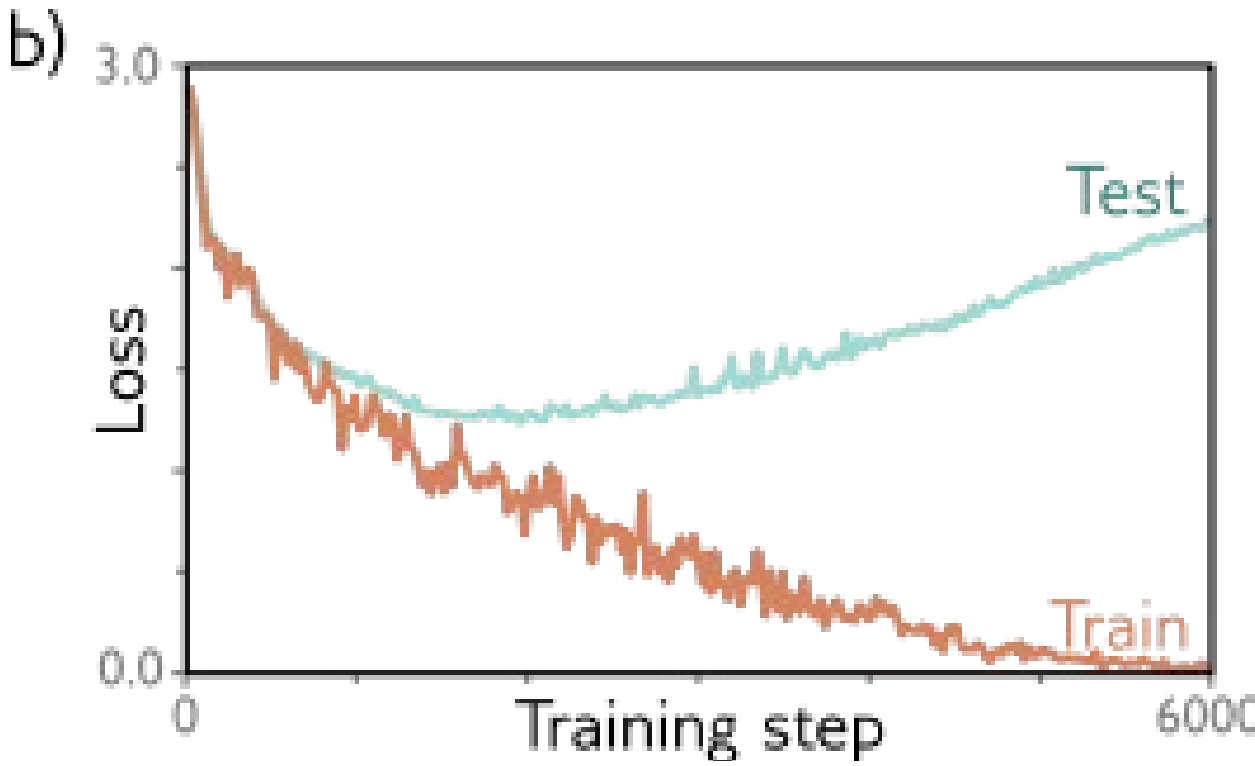
- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

MNIST 1D Dataset



MNIST-1D results for fully-connected network

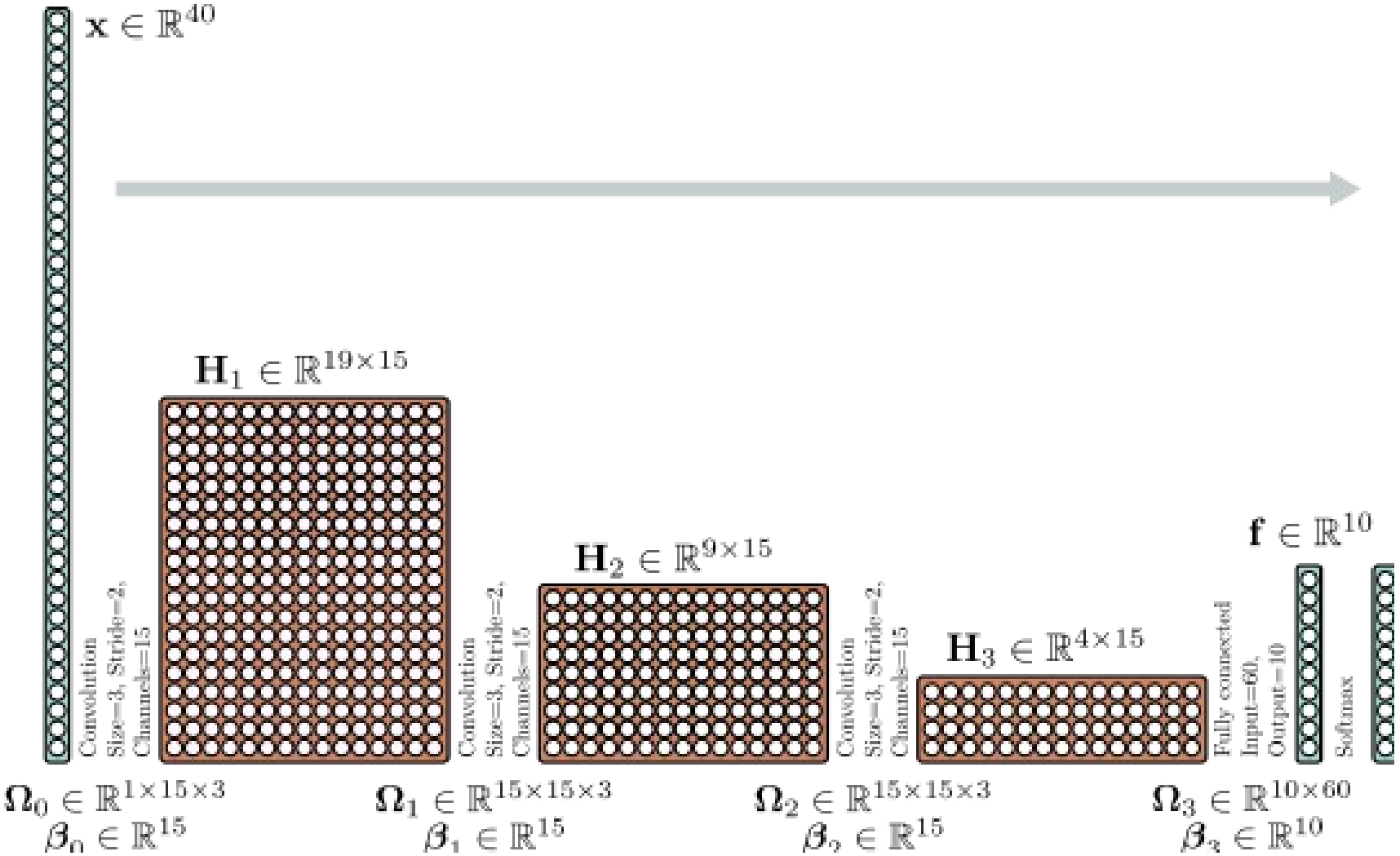




Convolutional network

- Four hidden layers
- Three convolutional layers
- One fully-connected layer
- Softmax at end
- Total parameters = 2050
- Trained for 100,000 steps with SGD, LR = 0.01, batch size 100

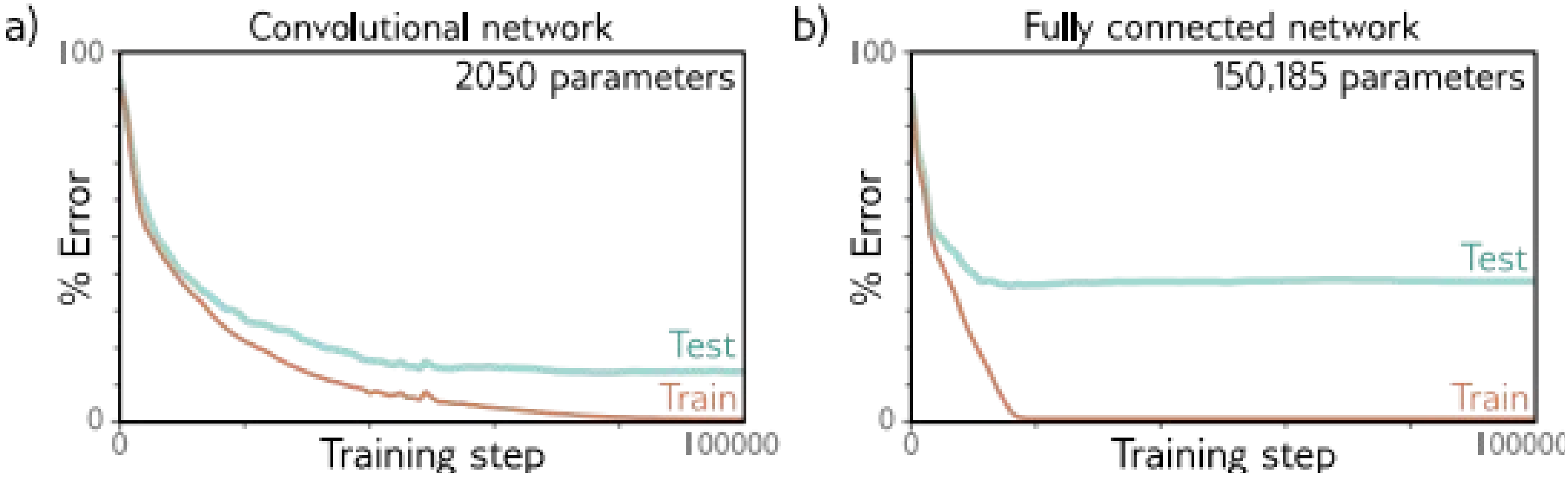
MNIST-1D convolutional network



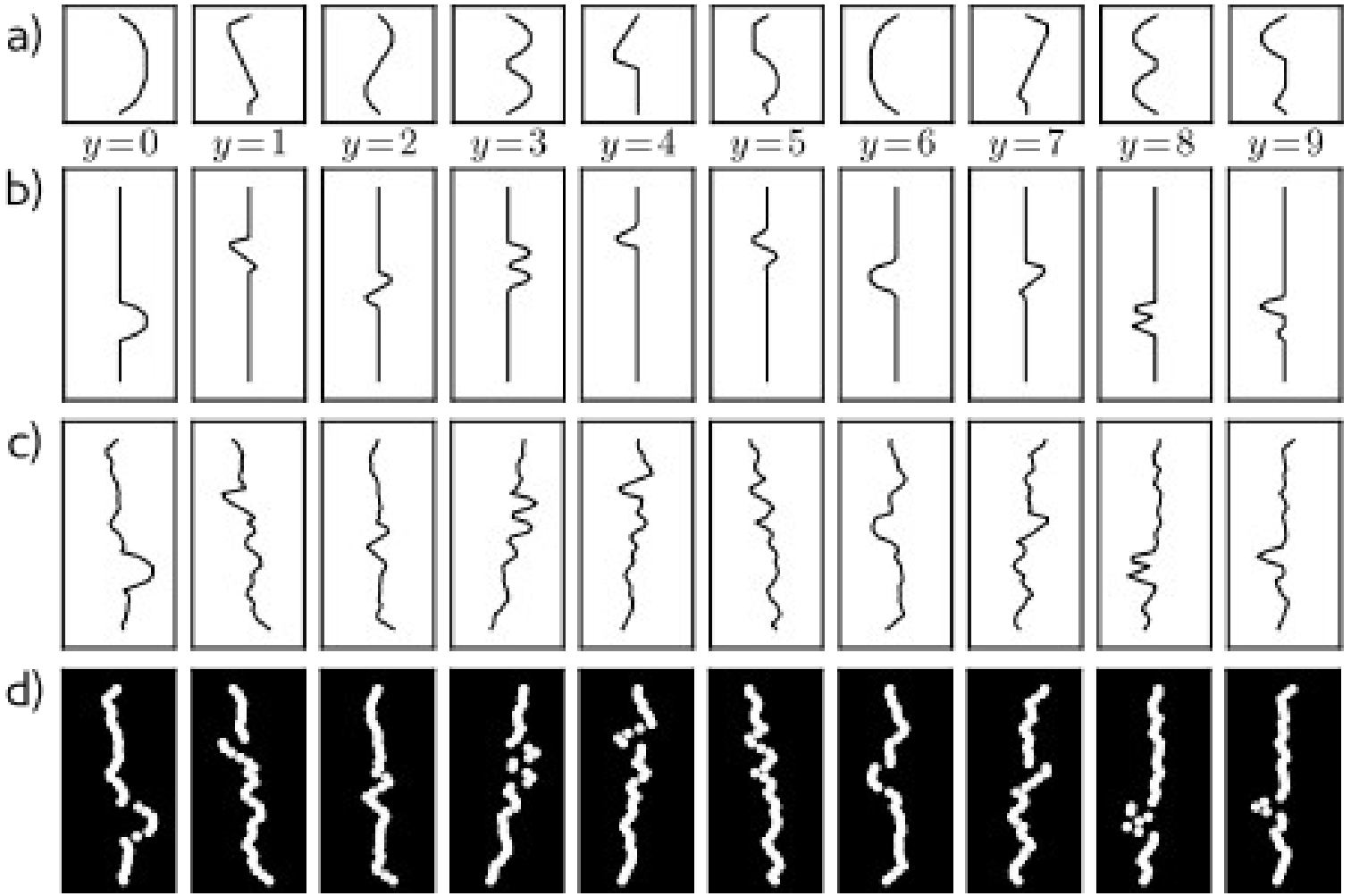
Fully connected network

- Exactly same number of layers and hidden units
- All fully-connected layers
- Total parameters = 150,185

Performance



MNIST 1D Dataset



Why?

- Better inductive bias
- Forced the network to process each location similarly
- Shares information across locations
- Search through a smaller family of input/ouput mappings, all of which are plausible

Convolution #2

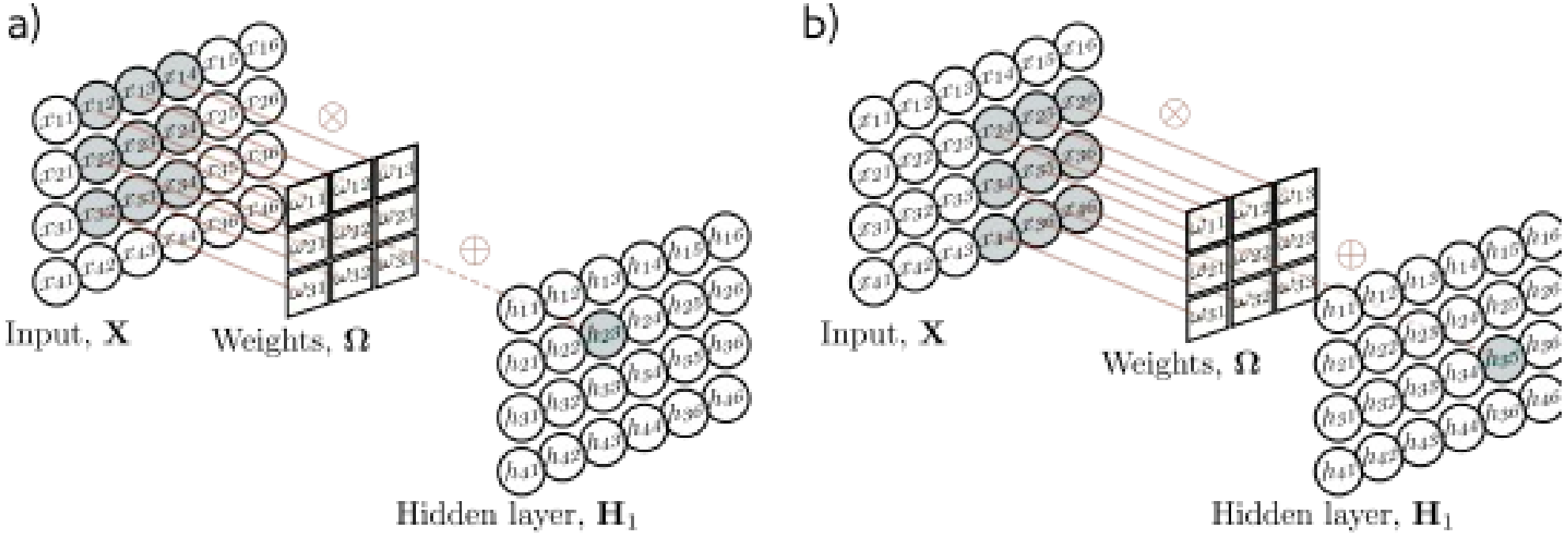
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

2D Convolution

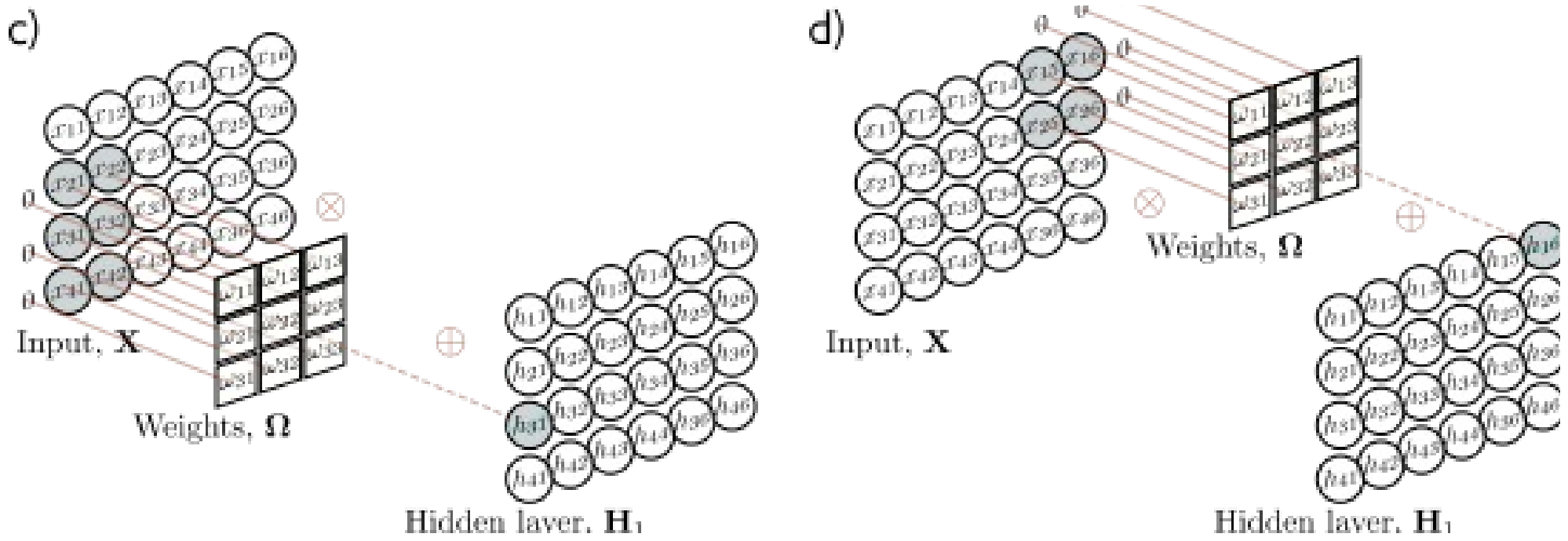
- Convolution in 2D
  - Weighted sum over a K x K region
  - K x K weights
- Build into a convolutional layer by adding bias and passing through activation function

$$h_{i,j} = \mathrm{a}\left[\beta + \sum_{m=1}^{3}\sum_{n=1}^{3}\omega_{m,n}x_{i+m-2,j+n-2}\right]$$

2D Convolution



2D Convolution



Padding

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Original Image

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Padded with 1 0’s

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{11} & a_{12} & a_{13} & 0 & 0 \\ 0 & 0 & a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & a_{31} & a_{32} & a_{33} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Padded with 2 0’s

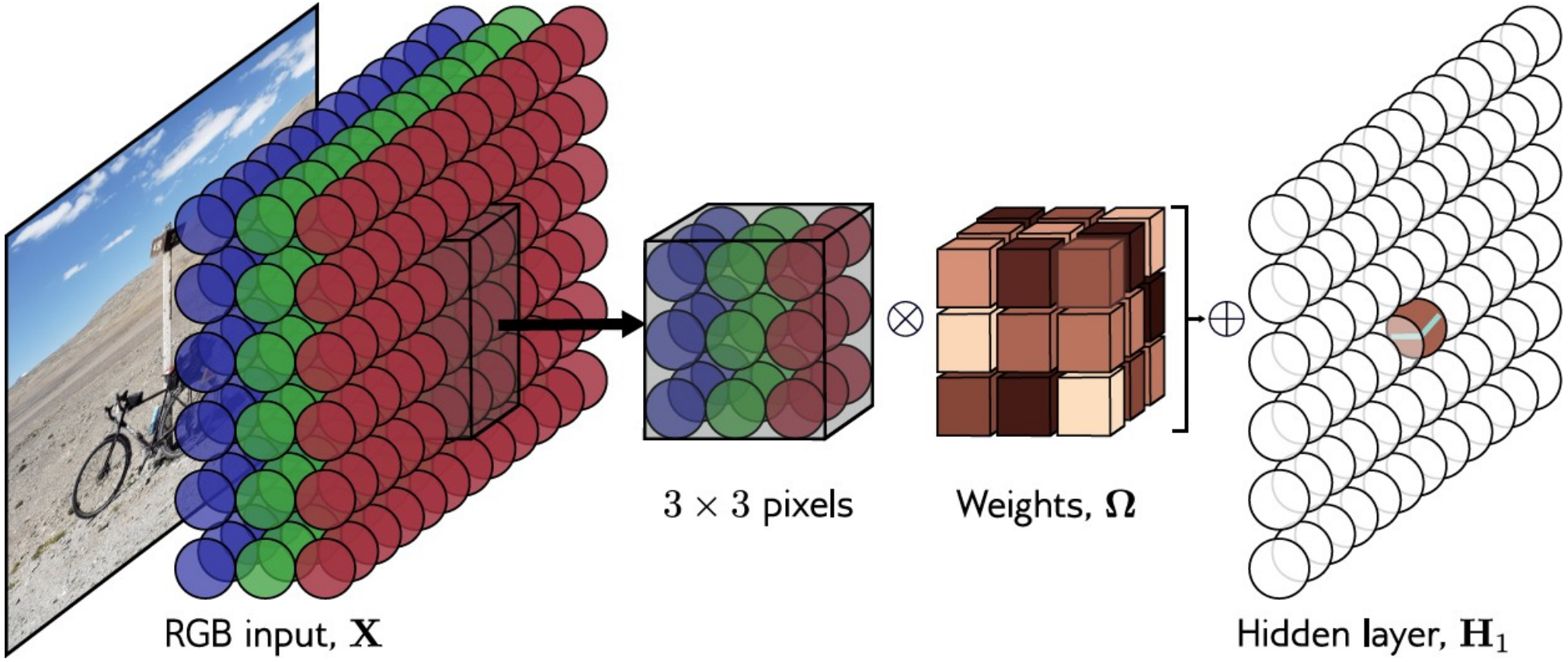
Same Padding:

Preserve the input dimensions after convolution.

Valid Padding:

No padding

Channels in 2D convolution



Kernel size, stride, dilation
all work as you would
expect

How many parameters?

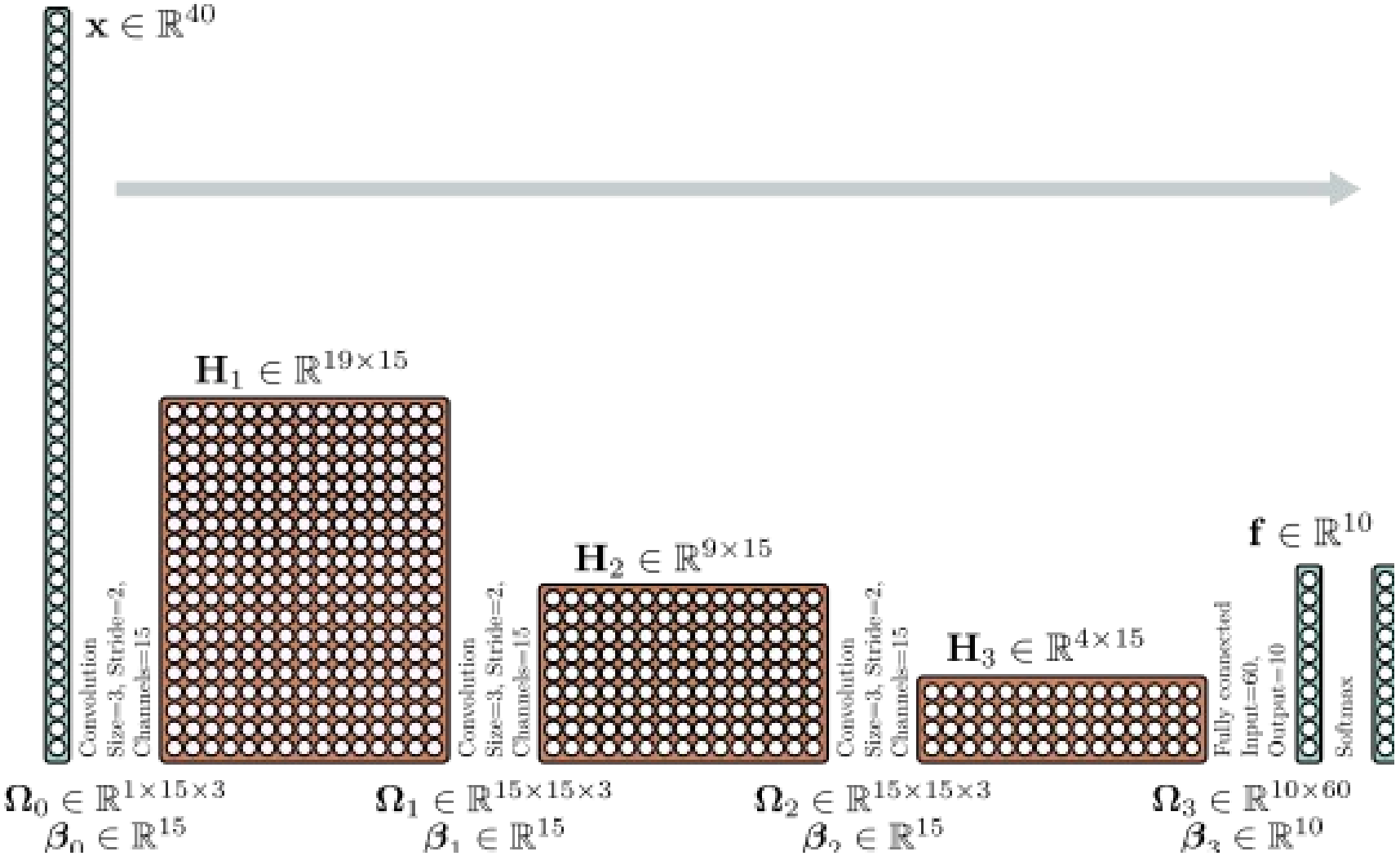
- If there are input channels and kernel size K x K

$$\boldsymbol{\omega} \in \mathbb{R}^{C_i \times K \times K} \qquad \boldsymbol{\beta} \in \mathbb{R}$$

- If there are input channels and output channels

$$\boldsymbol{\omega} \in \mathbb{R}^{C_i \times C_o \times K \times K} \qquad \boldsymbol{\beta} \in \mathbb{R}^{C_o}$$

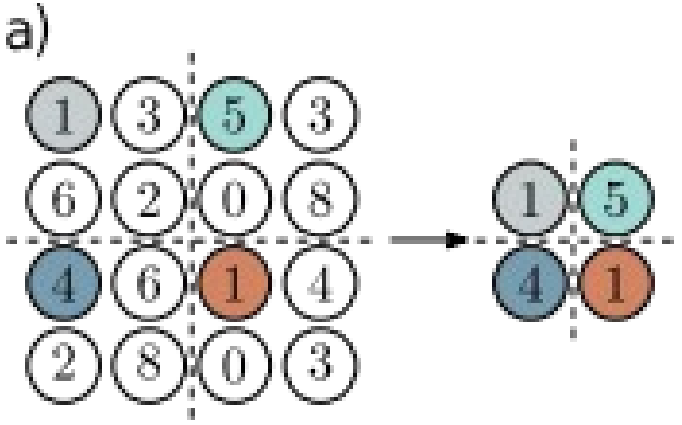
MNIST-1D convolutional network



Convolution #2

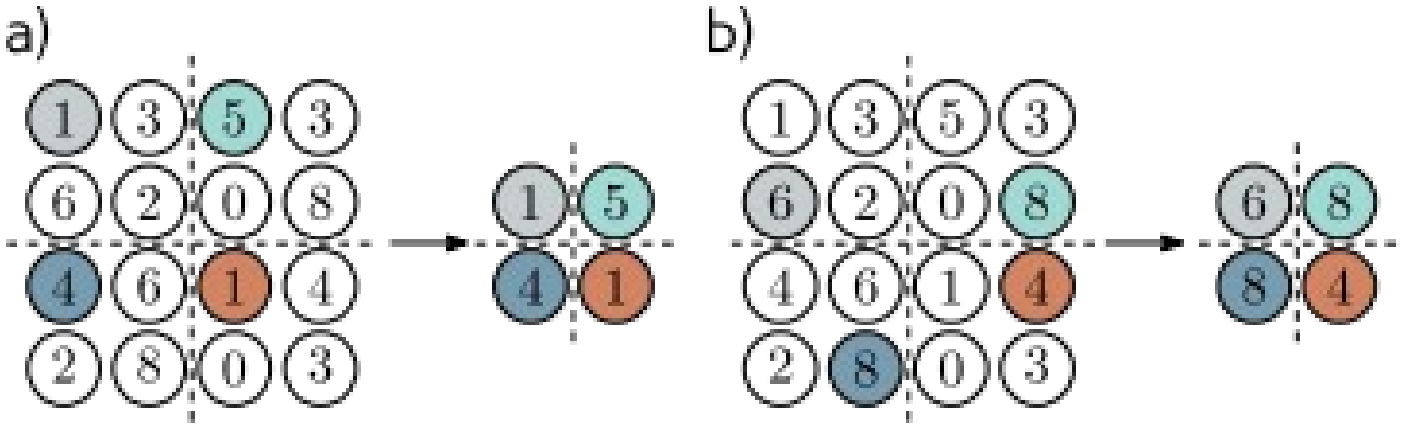
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

Downsampling



Sample every other position (equivalent to stride two)

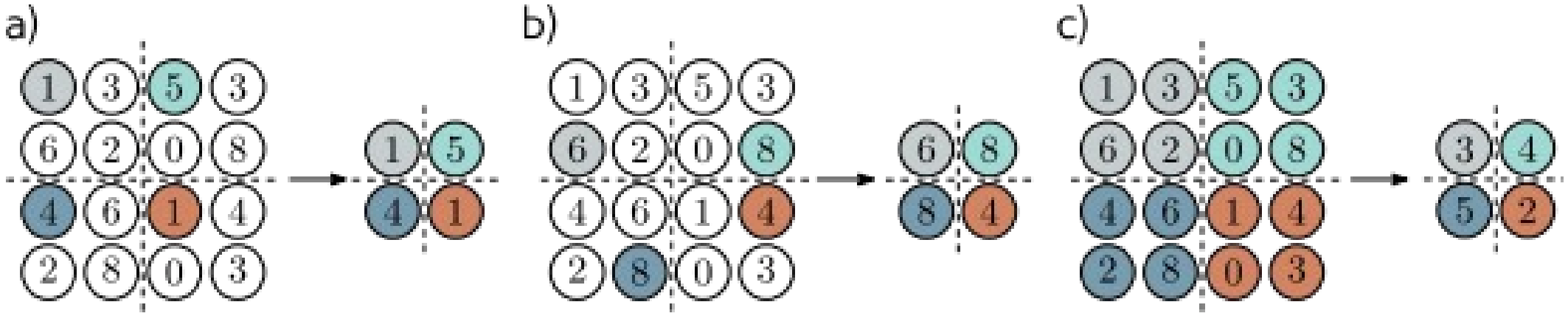
Downsampling



Sample every other position (equivalent to stride two)

Max pooling (partial invariance to translation)

Downsampling

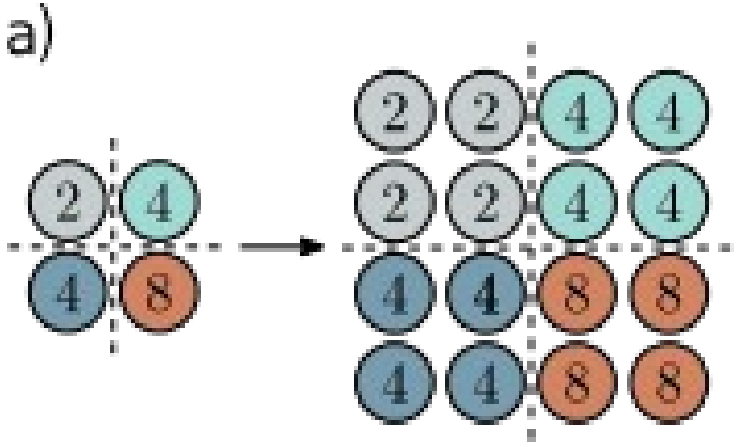


Sample every other position (equivalent to stride two)

Max pooling (partial invariance to translation)

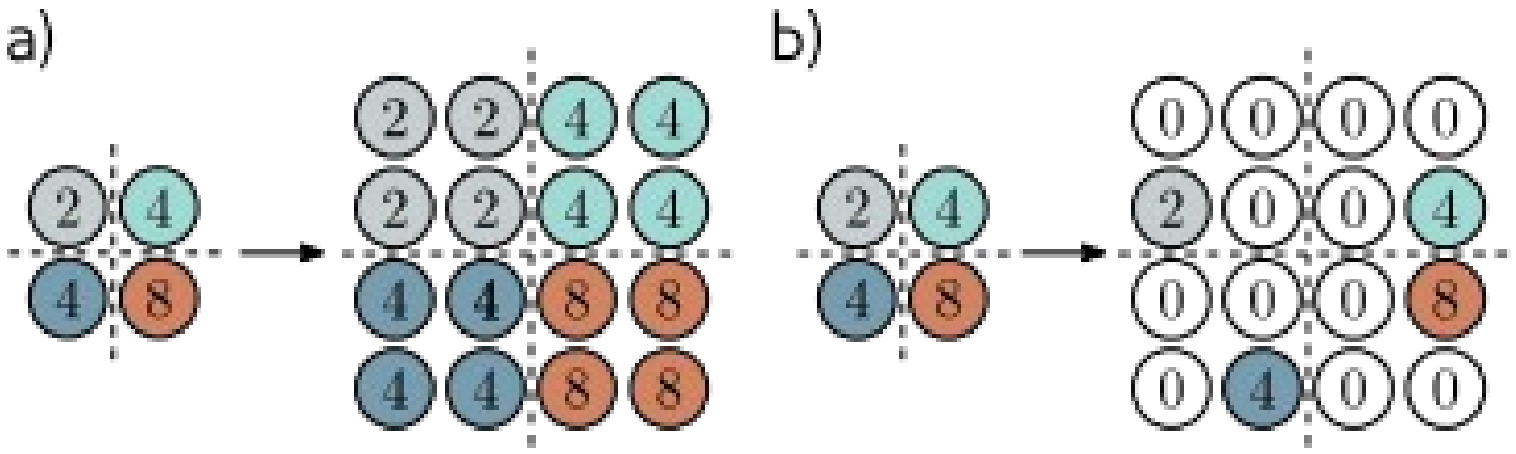
Mean pooling

Upsampling



Duplicate

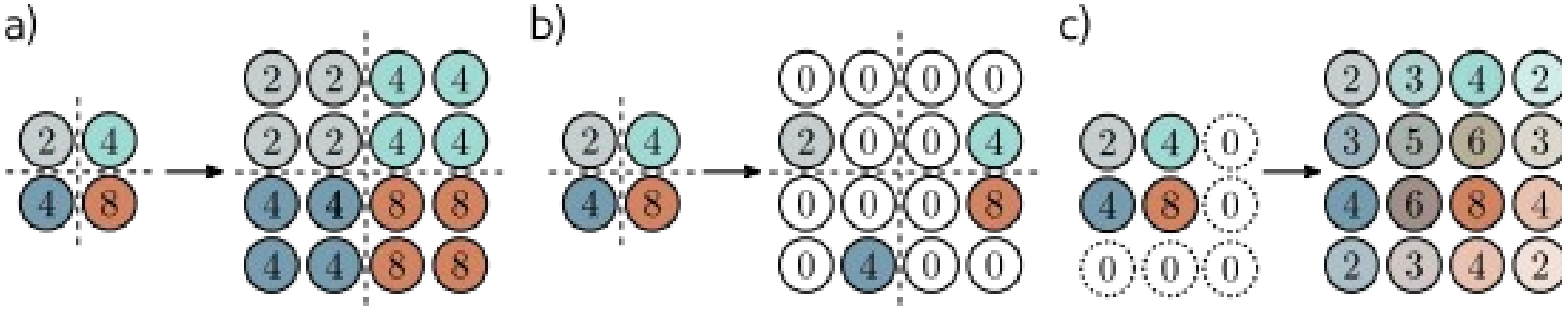
Upsampling



Duplicate

Max-upsampling

Upsampling

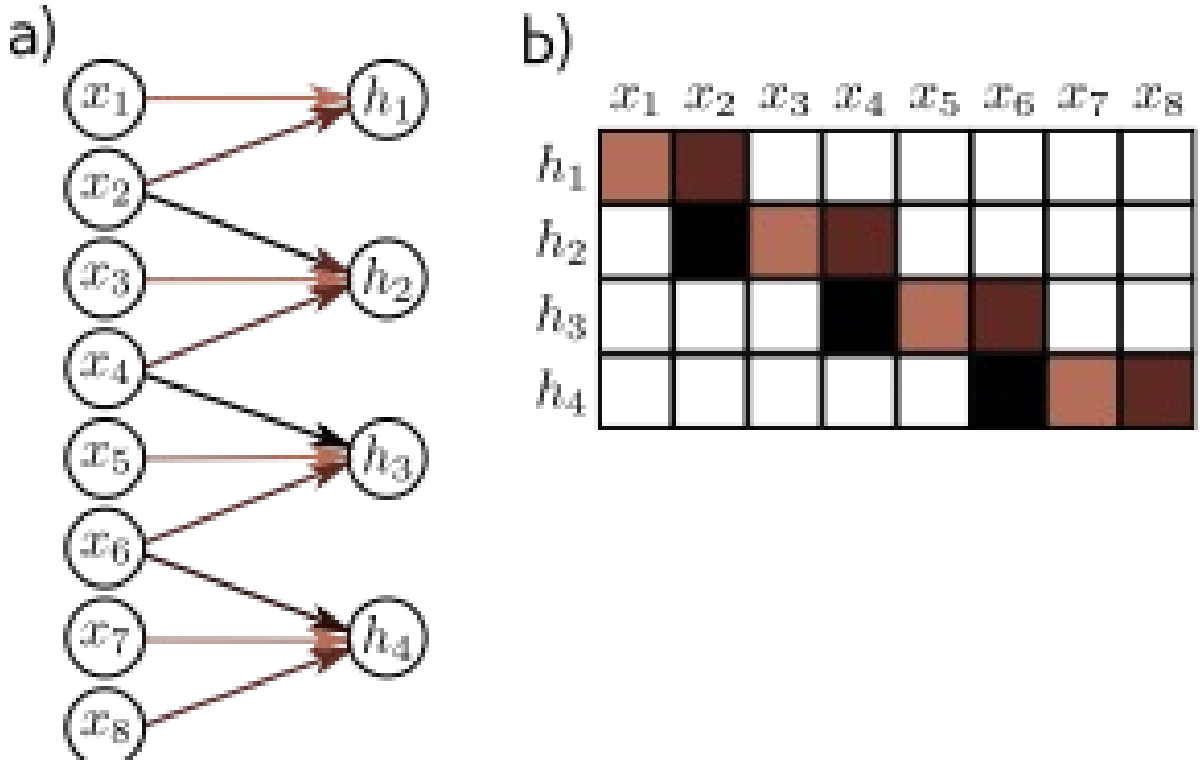


Duplicate

Max-upsampling

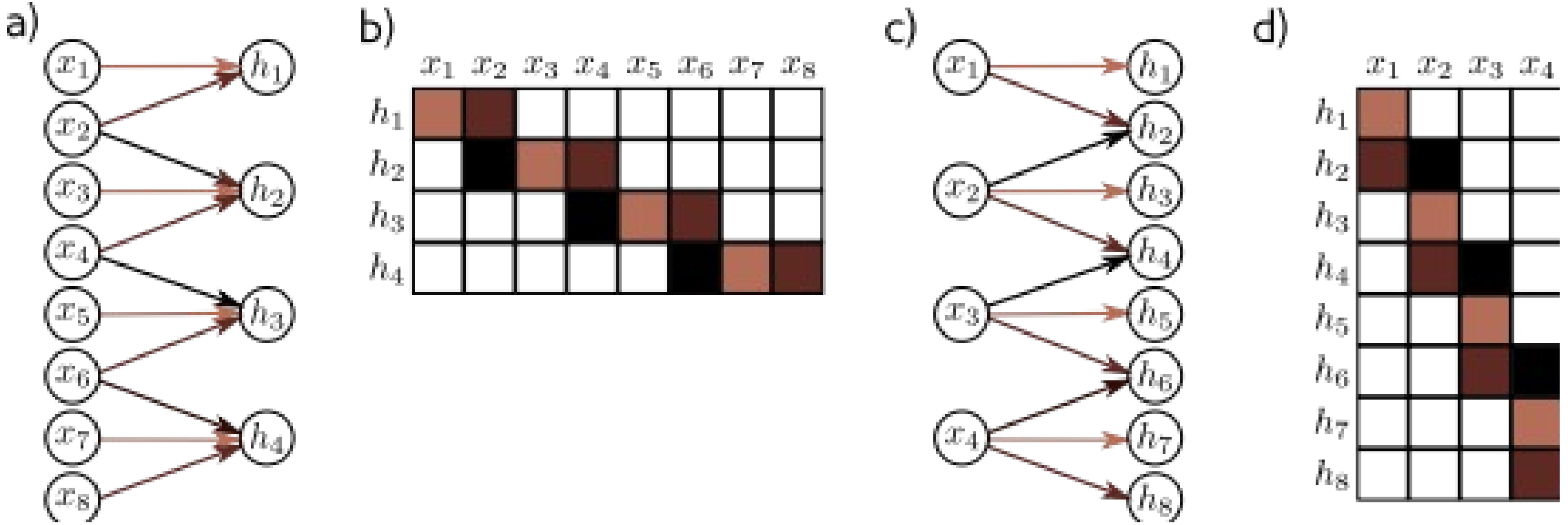
Bilinear interpolation

Transposed convolutions



Kernel size 3, Stride 2 convolution

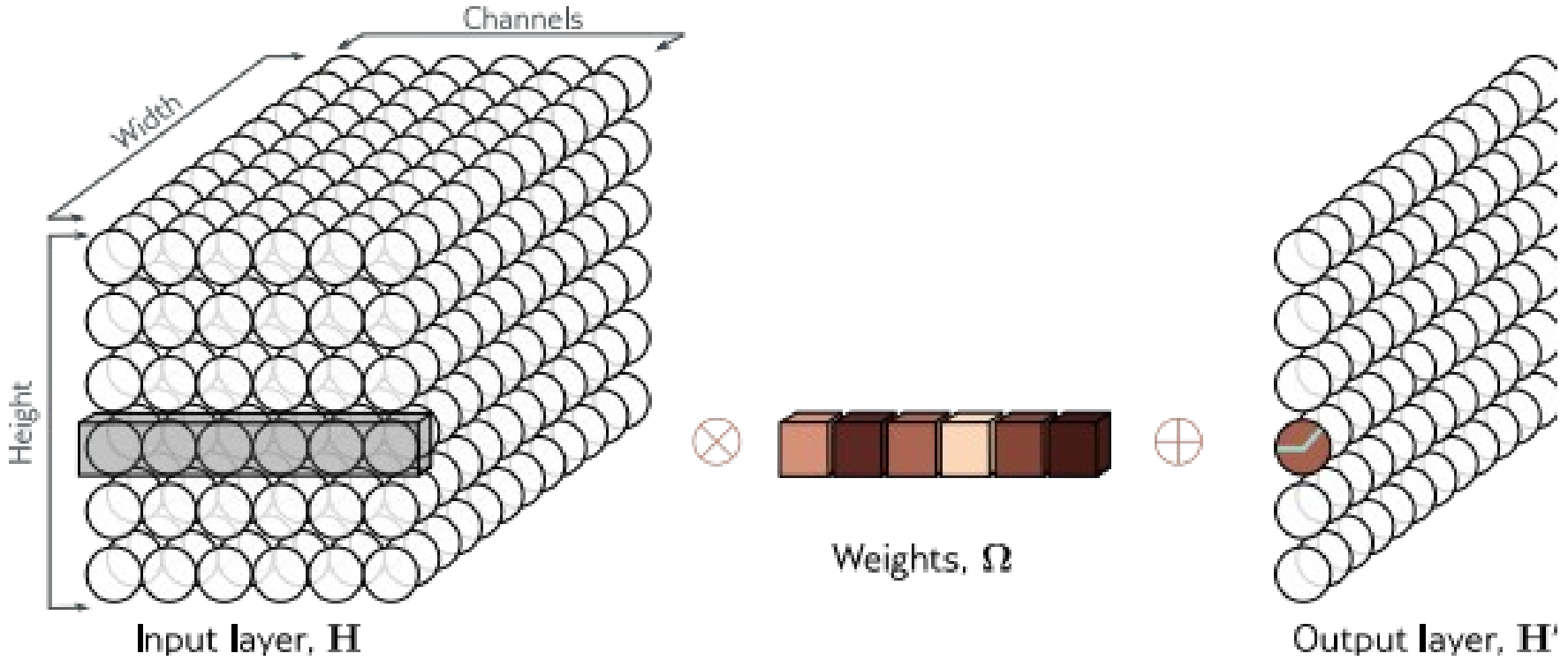
Transposed convolutions



Kernel size 3, Stride 2 convolution

Transposed convolution

1x1 convolution

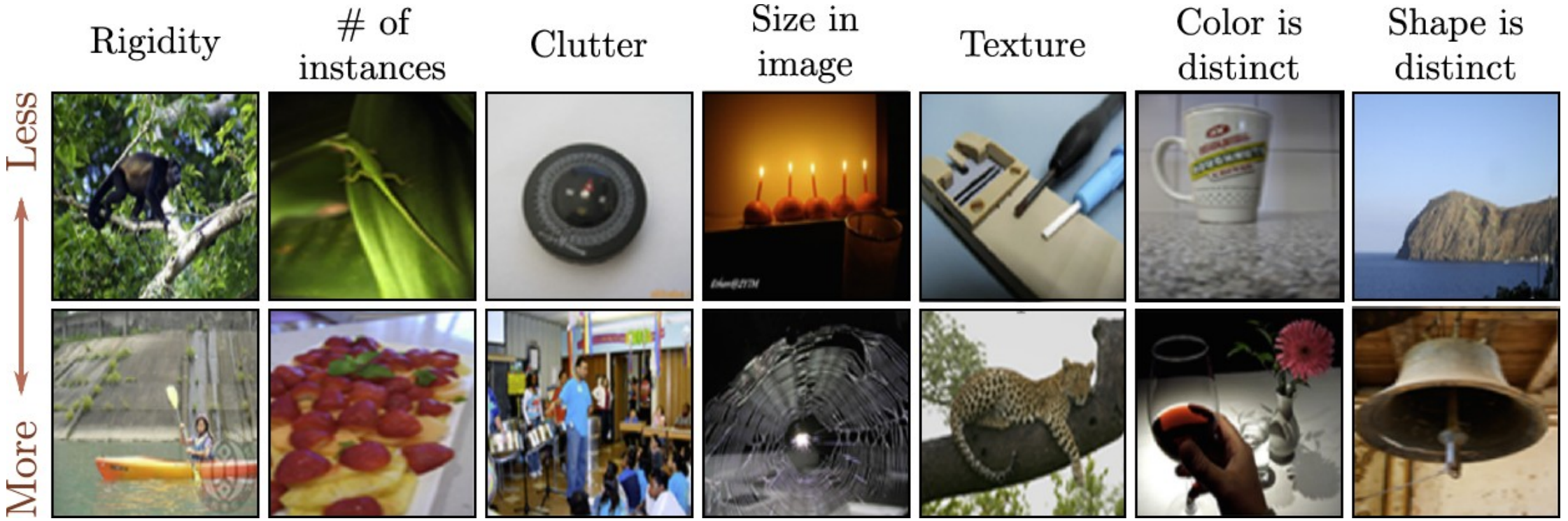


- Mixes channels
- Can change number of channels
- Equivalent to running same fully connected network at each position

Convolution #2

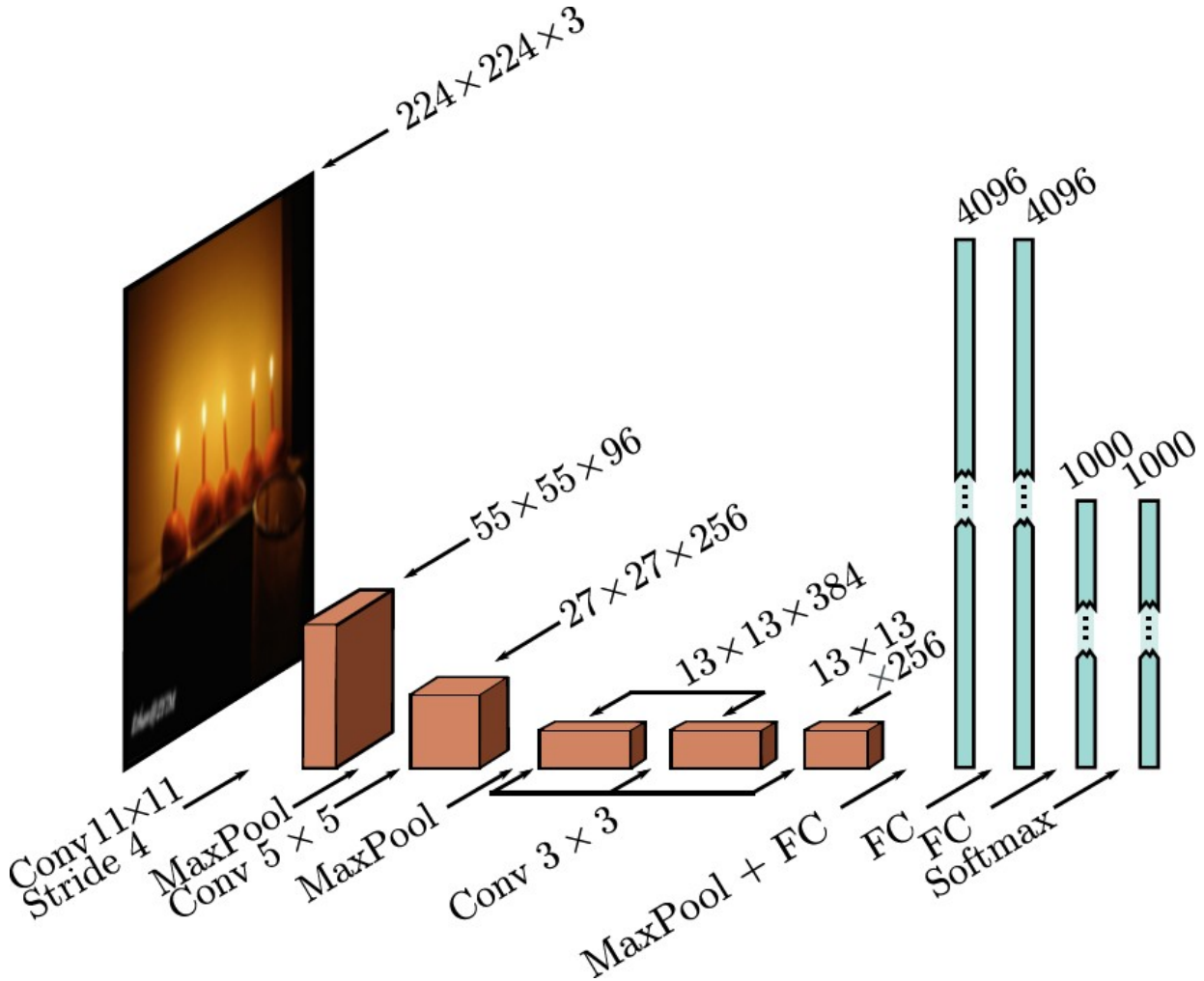
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

ImageNet database

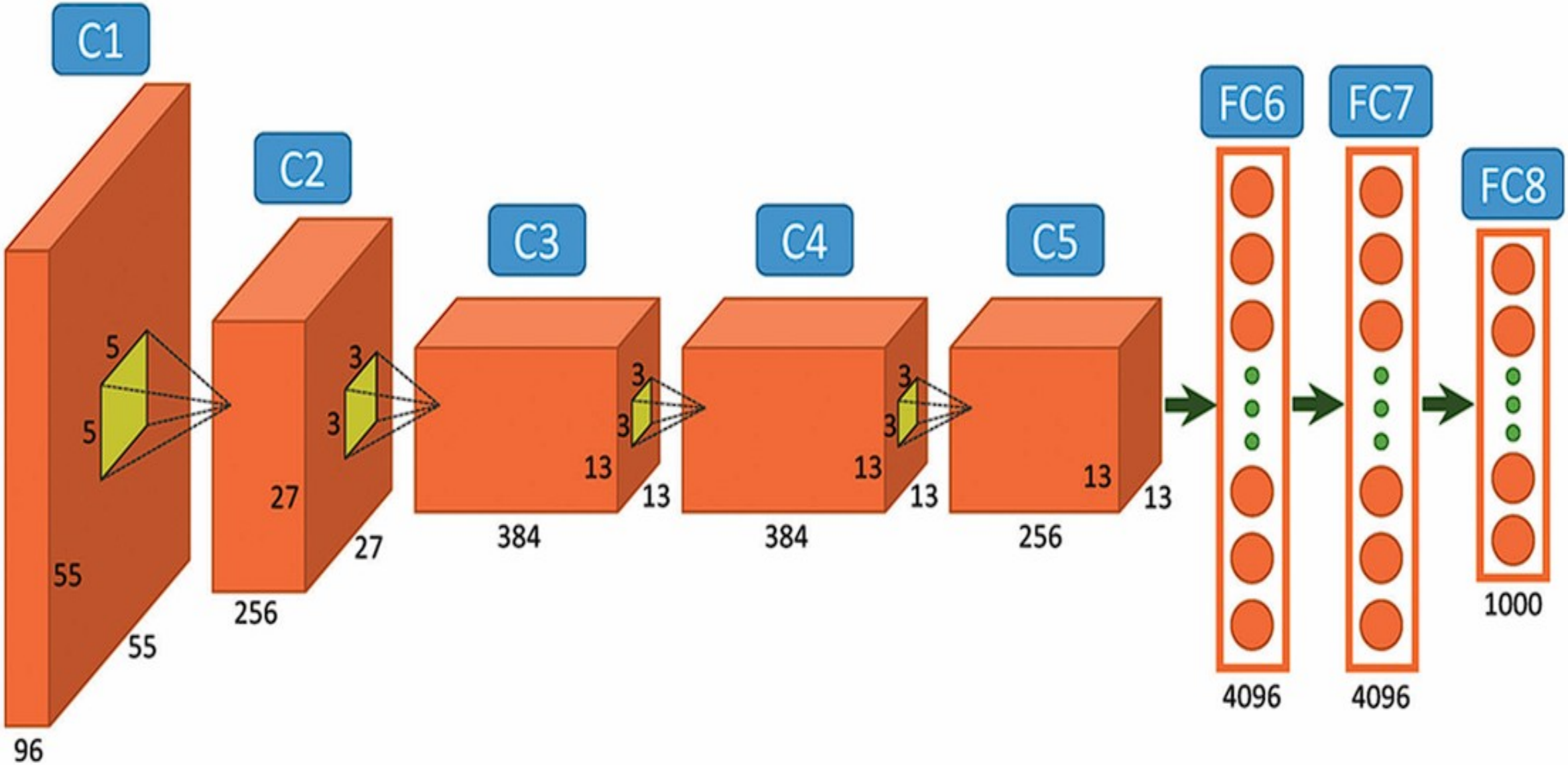


- 224 x 224 images
- 1,281,167 training images, 50,000 validation images, and 100,000 test images
- 1000 classes

AlexNet (2012)



Almost all the 60 million parameters parameters are in fully connected layers


C1
C2
C3
C4
C5
FC6
FC7
FC8
5
5
3
3
3
3
3
3
96
55
55
256
27
27
384
13
13
384
13
13
256
13
13
4096
4096
1000

- AlexNet structure
  - 5 Convolutional layers and 3 fully connected layers
  - 1st layer downsamples input using 11x11 kernel, stride of four 4 and creates 96 channels
  - 2nd layer downsamples using a max pooling layer before applying a 5×5 kernel to create 256 channels.
  - There are three more convolutional layers with kernel size 3×3 eventually resulting in a 13×13 representation with 256 channels.
  - The final max pooling layer yields 6x6 representation with 256 channels
  - Next 3 fully connected layers contain 4096, 4096 and 1000 neurons respectively

Let’s do some calculations

- Calculating height, width of convolutional layer
- output_size = ceil((input_size - kernel_size + 2 * padding) / stride) + 1
- Example:
  - input_size=224x224
  - kernel_size=11x11
  - stride=4
  - No Padding in this one
  - output_Size=(224- 11 +2*0)/4 +1
    = 213/4 +1
    = 53.25 +1
    = 54.25
    = 55

  This would hence create a convolution of size 55x55
- Calculation for Max Pooling layer is also same but without the padding term

2nd Layer calculation Example

- Calculating height, width of convolutional layer
- input_size=27x27
- kernel_size=5x5
- stride=2
- padding=2
- output_Size=(27 - 5 +2*2)/1 +1

  = 26/1 +1

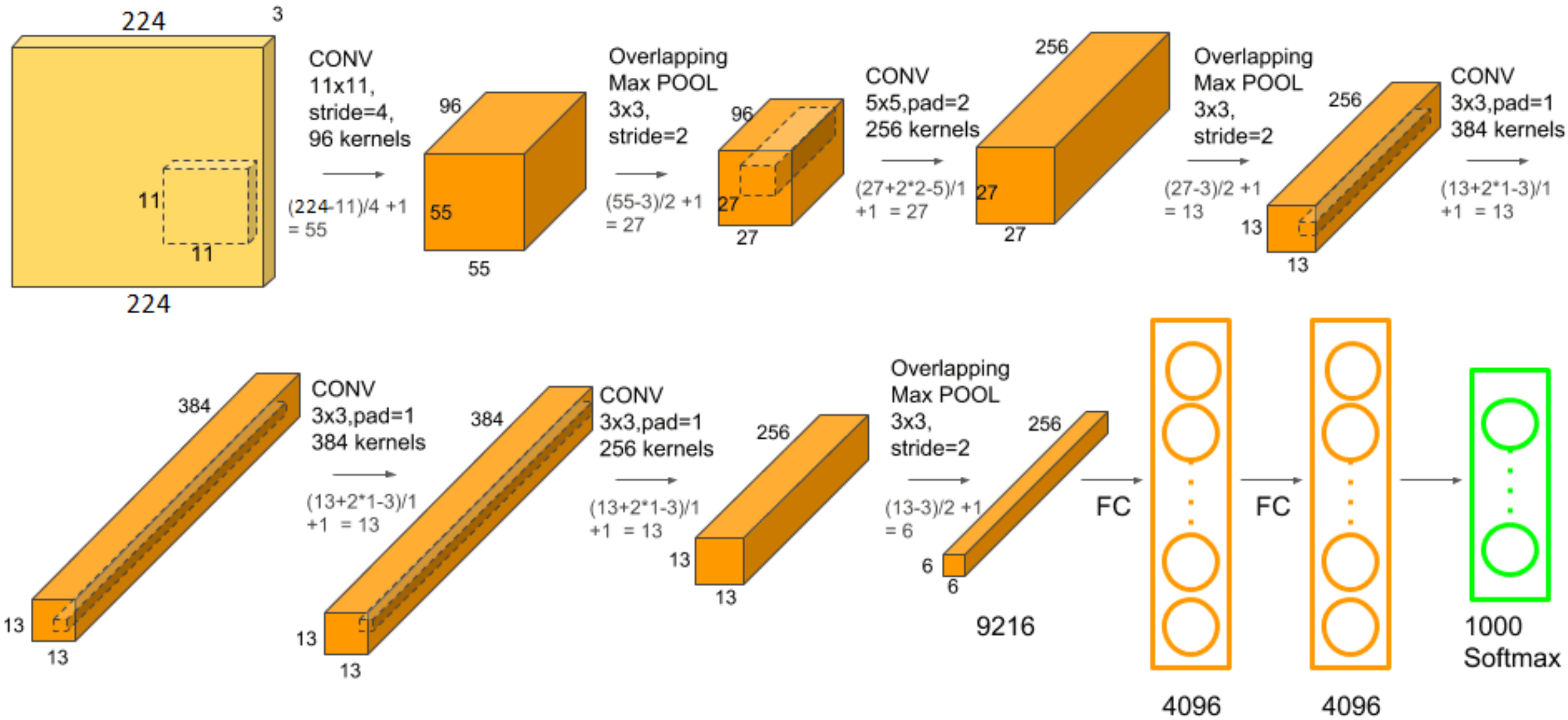
  = 26 +1

  = 27

- Let’s look at the number of parameters
- 1st layer has each kernel of size 11x11x3 and there are 96 kernels.
- Each kernel would also have bias
- Hence, no. of parameters = (11x11x3+1)x96=34944

Size	Operation	Kernel Size	No. of Kernels/ Depth	Stride	Padding	Number of Parameters	output_size
224 * 224 * 3	Conv1 + Relu	11 * 11	96	4		(11*11*3+1)*96 =34944	
55 * 55 * 96	Max Pooling	3 * 3		2			(55-3+2*0)/2 +1 =27
27 * 27 * 96	Conv2 + Relu	5 * 5	256	1	2	(5*5*96+1)*256 =614656	(27-5+2*2)/2 +1 =27
27 * 27 * 256	Max Pooling	3 * 3		2			(27-3+2*0)/2 +1 =13
13 * 13 * 256	Conv3 + Relu	3 * 3	384	1	1	(3*3*256+1)*384 =885120	(13-3+2*1)/1 +1 =13
13 * 13 * 384	Conv4 + Relu	3 * 3	384	1	1	(3*3*384+1)*384 =1327488	(13-3+2*1)/1 +1 =13
13 * 13 * 384	Conv5 + Relu	3 * 3	256	1	1	(3*3*384+1)*256 =884992	(13-3+2*1)/1 +1 =13
13 * 13 * 256	Max Pooling	3 * 3		2			

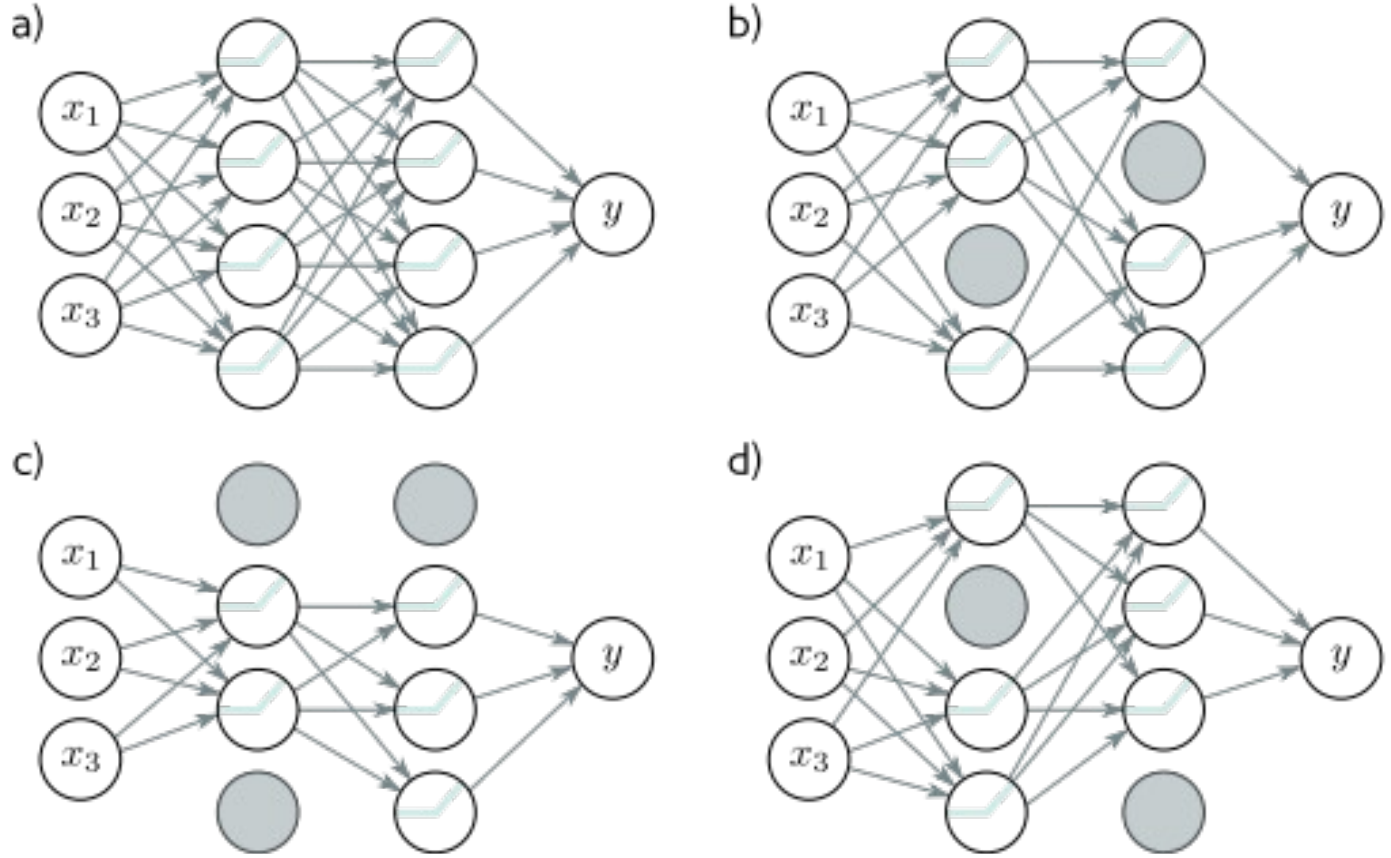
Size	Operation	No. of Parameters
	Dropout (rate 0.5)	
256 * 6 * 6	FC6 + Relu	(256 * 6 * 6 + 1)* 4096=37752832
	Dropout (rate 0.5)	
4096	FC7 + Relu	(4096+1) * 4096=16781312
4096	FC8 + Relu	(4096 +1) * 1000=4097000
	1000 classes	
Overall	Conv VS FC	Conv:3.7million (6%) , FC: 58.6 million (94%)


224
3
224
11
11
CONV
11x11,
stride=4,
96 kernels
(224-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

Data augmentation

- Data augmentation a factor of 2048 using (i) spatial transformations and (ii) modifications of the input intensities.

Dropout

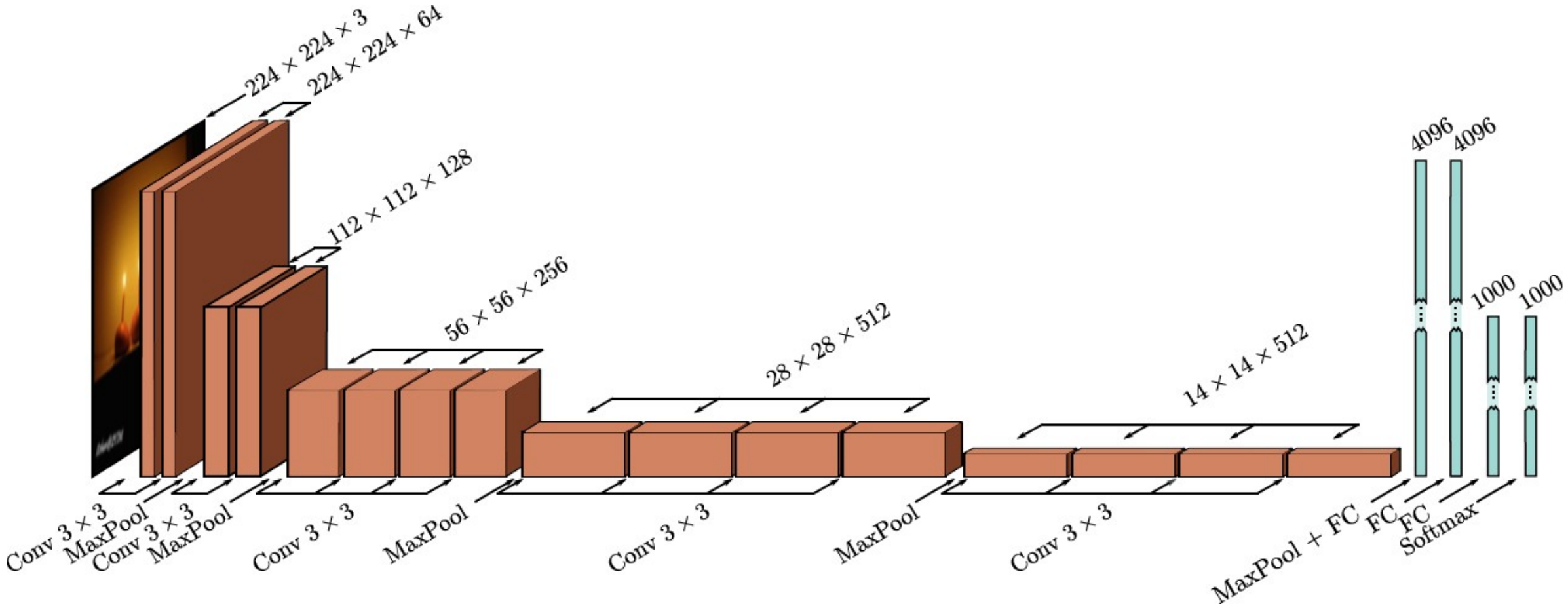


- Dropout was applied in the fully connected layers

Details

- At test time average results from five different cropped and mirrored versions of the image
- SGD with a momentum coefficient of 0.9 and batch size of 128.
- L2 (weight decay) regularizer used.
- This system achieved a 16.4% top-5 error rate and a 38.1% top-1 error rate.

VGG (2015)


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
4096
4096
1000
1000
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool + FC
FC
FC
Softmax

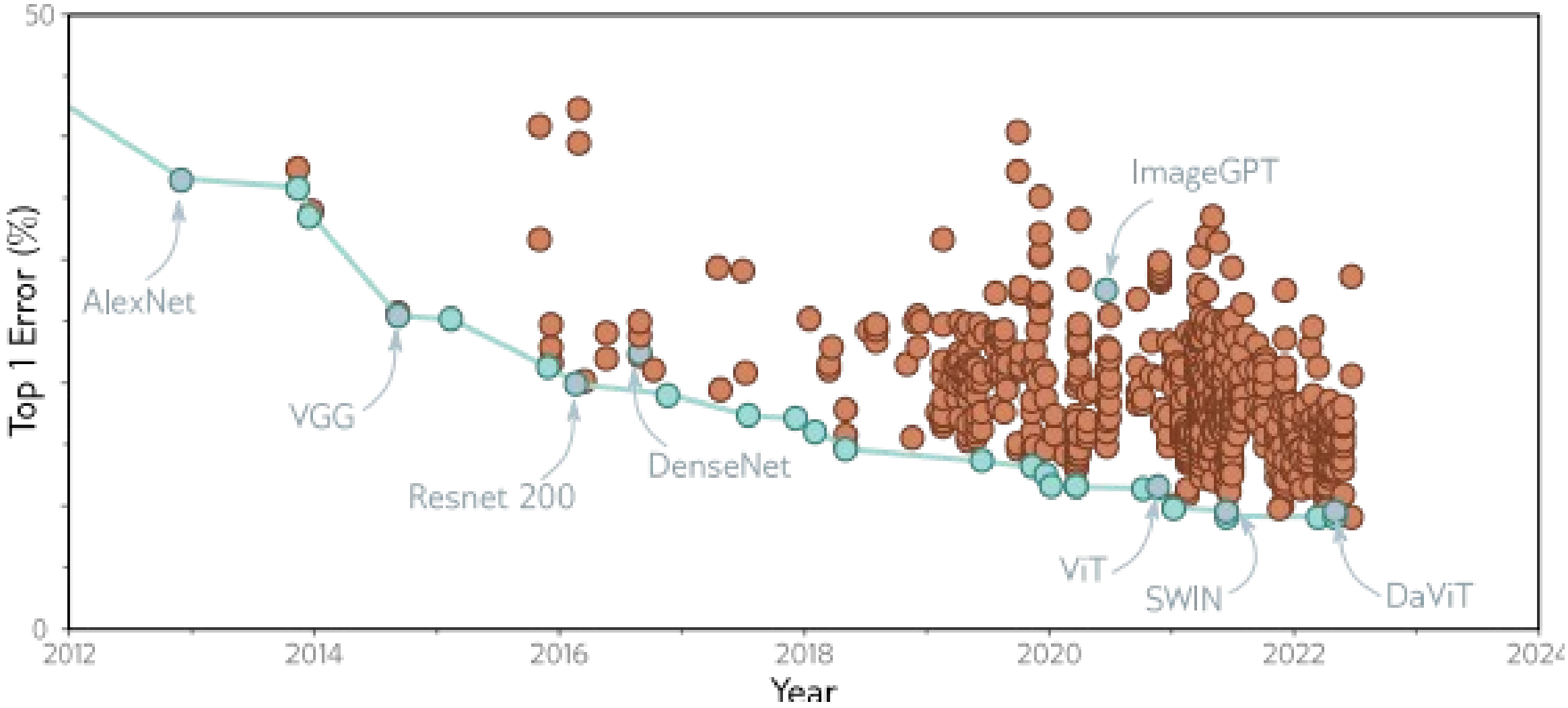
Activity 8

- Make a similar table as in slides no. 81 and 82.
- Calculate the output size and number of parameters for VGG

Details

- 19 hidden layers
- 144 million parameters
- 6.8% top-5 error rate, 23.7% top-1 error rate

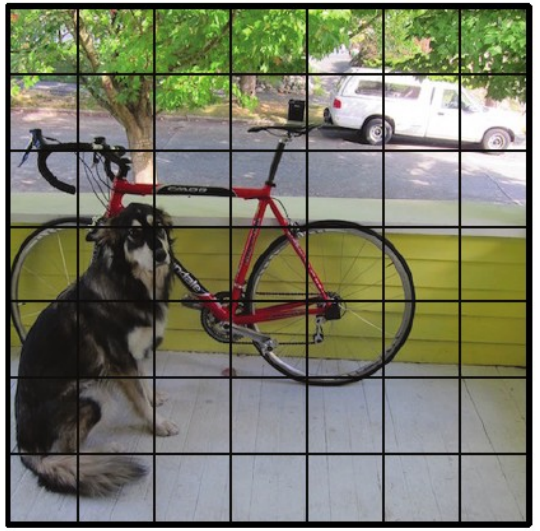
ImageNet History



Convolution #2

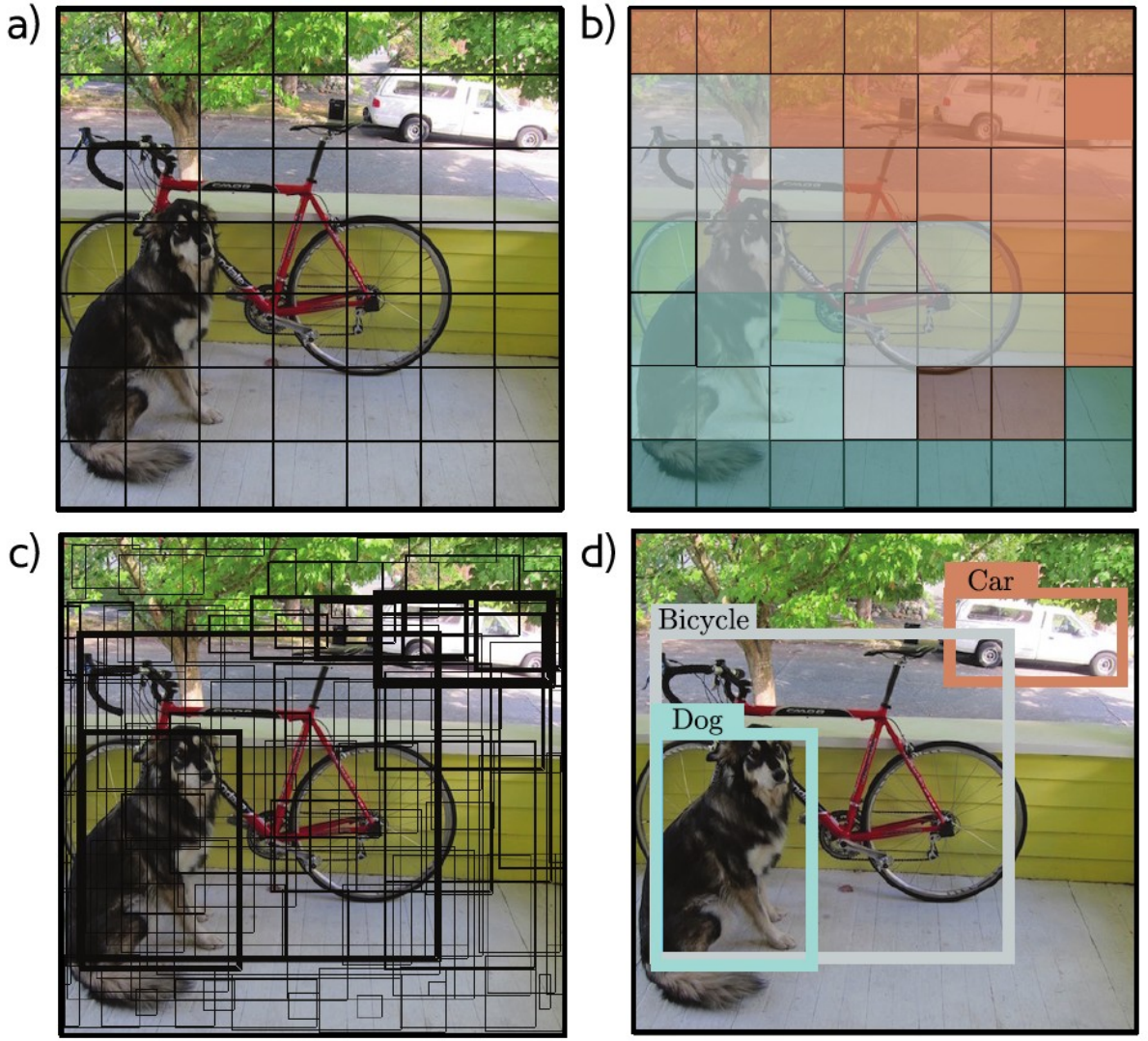
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

You Only Look Once (YOLO)

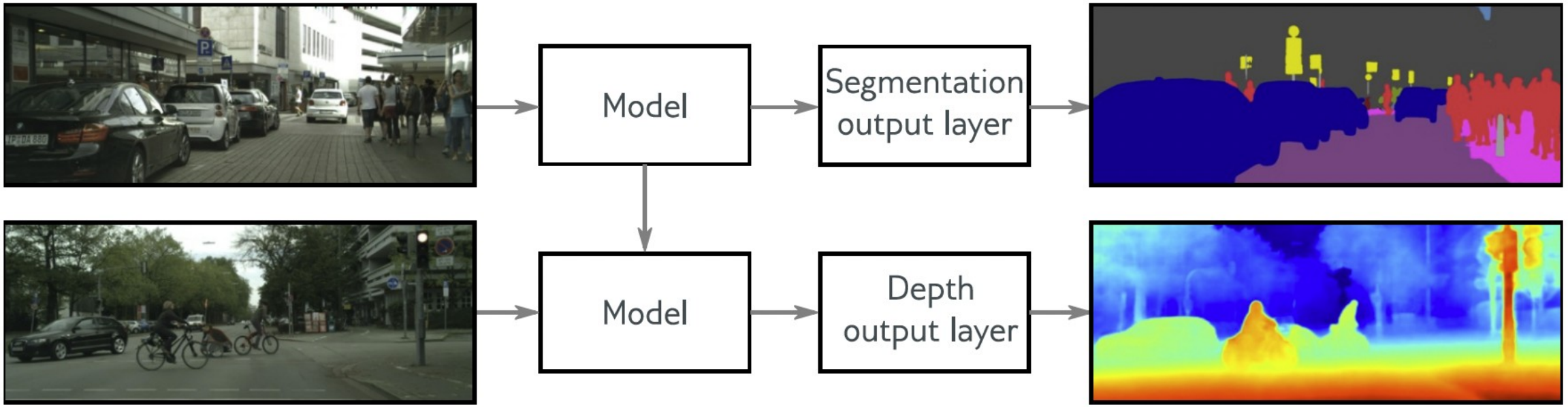


- Network similar to VGG (448x448 input)
- 7×7 grid of locations
- Predict class at each location
- Predict 2 bounding boxes at each location
  - Five parameters –x,y, height, width, and confidence
- Momentum, weight decay, dropout, and data augmentation
- Heuristic at the end to threshold and decide final boxes

Object detection (YOLO)

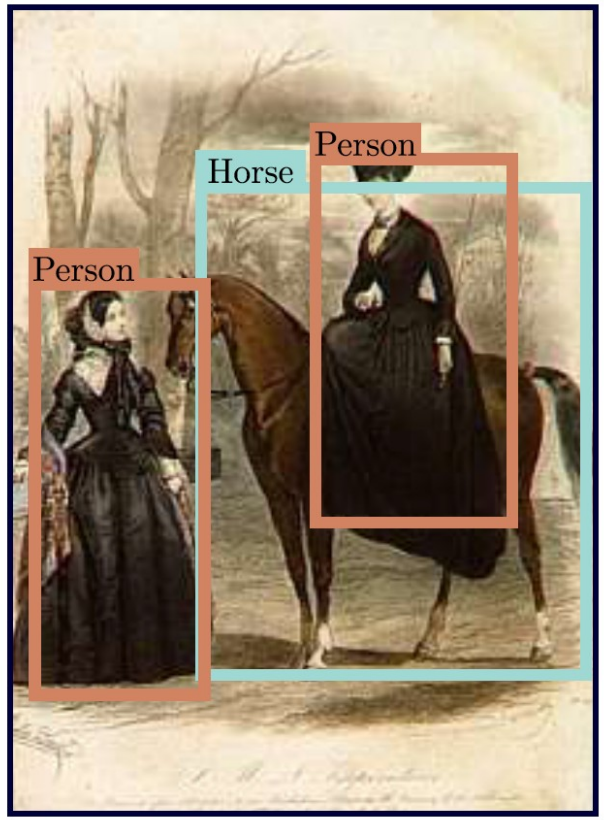


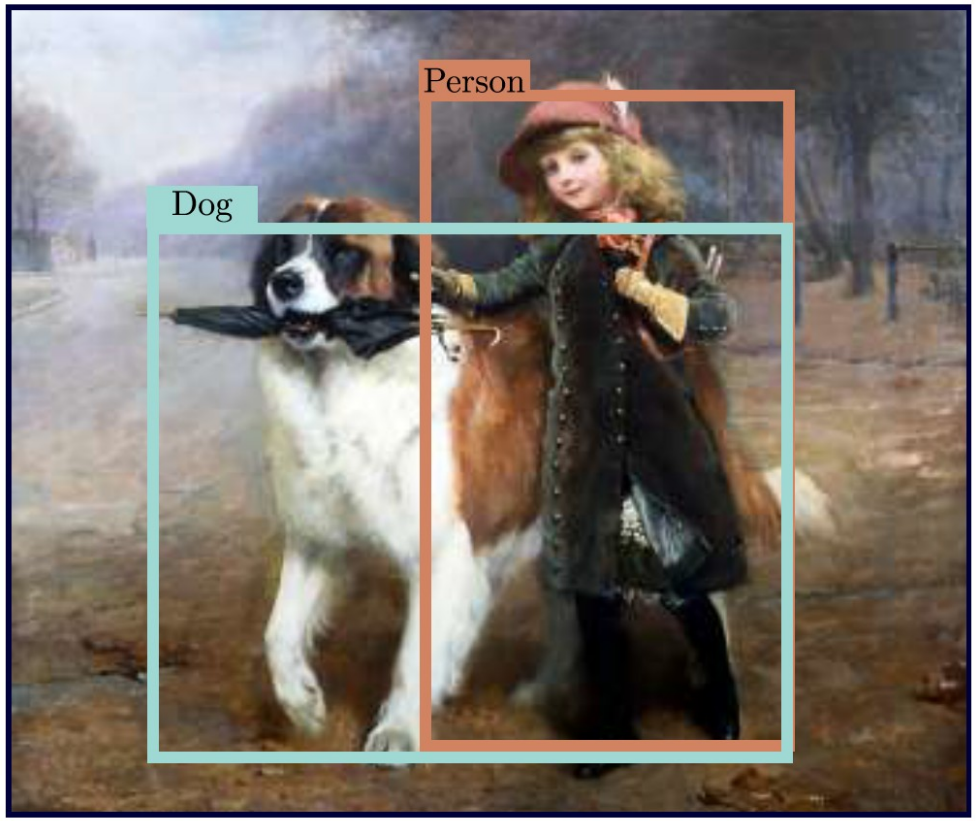
Transfer learning

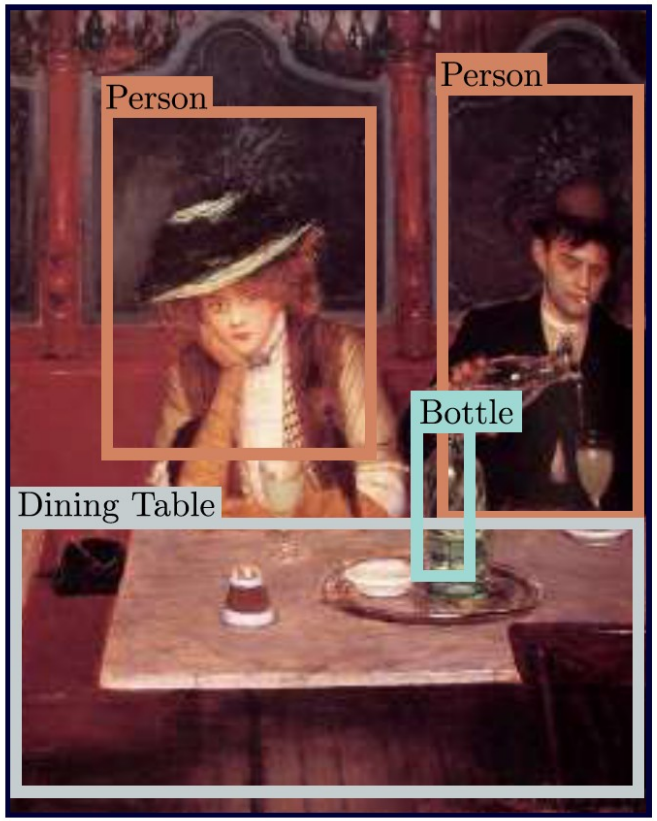


Transfer learning from ImageNet classification

Results


Person
Horse
Person

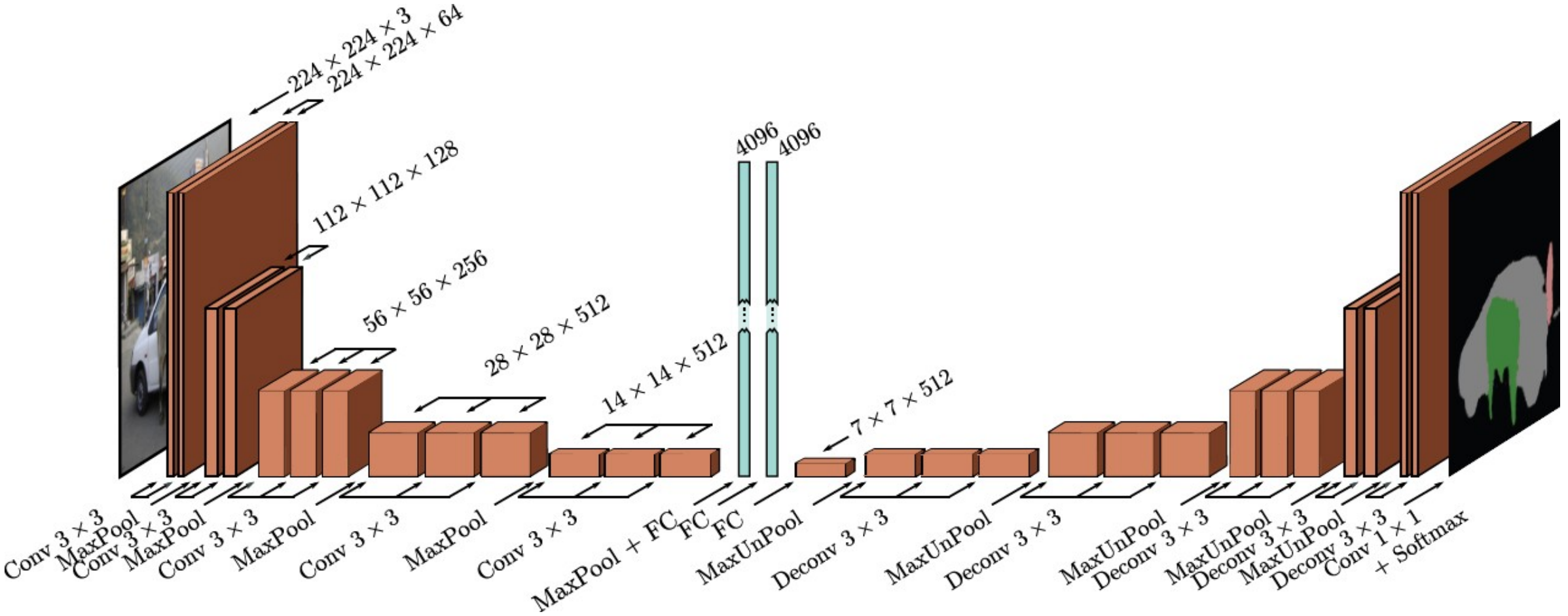

Person
Dog


Person
Person
Bottle
Dining Table

Convolution #2

- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

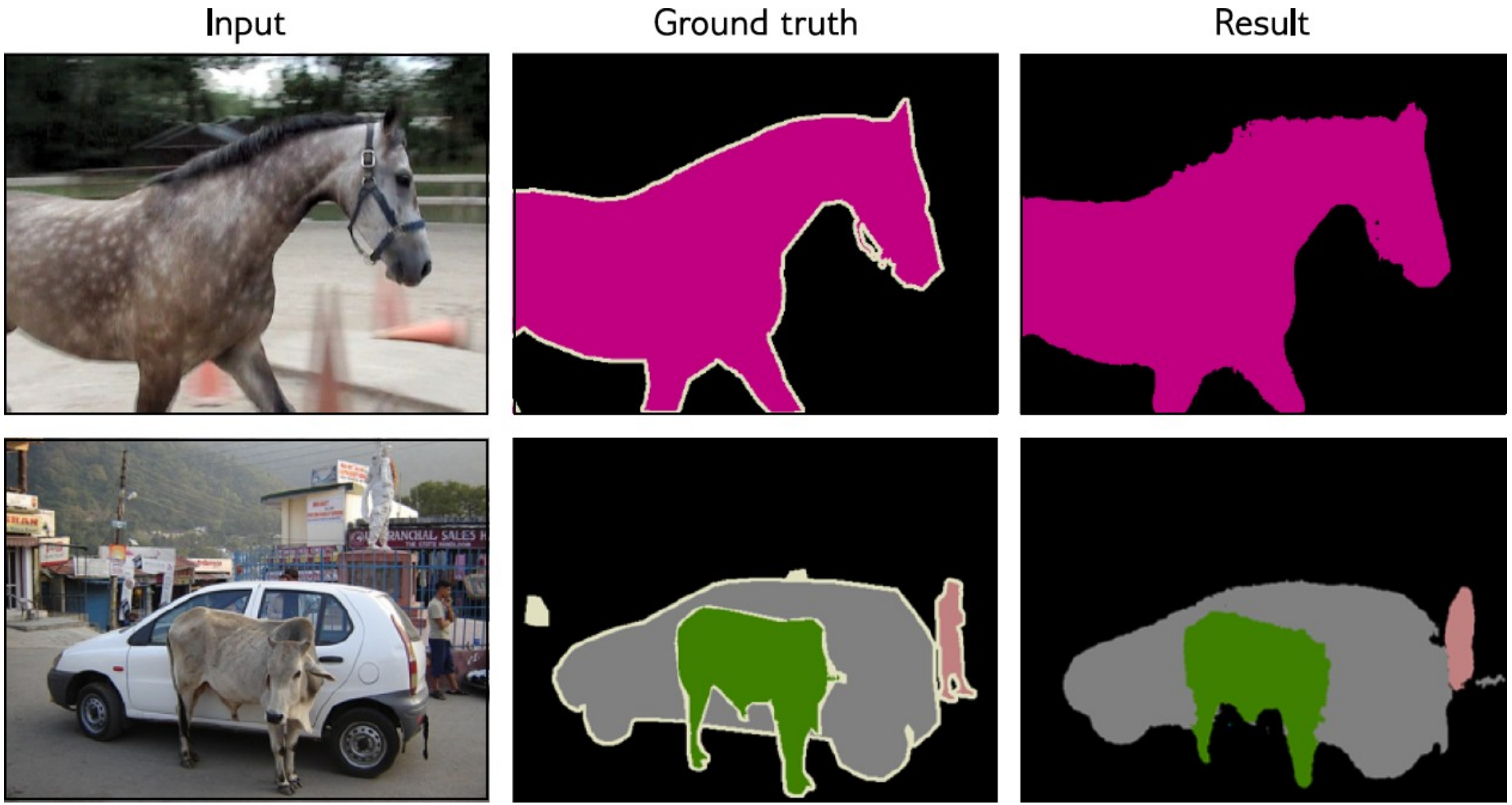
Semantic Segmentation (2015)



Encoder

Decoder

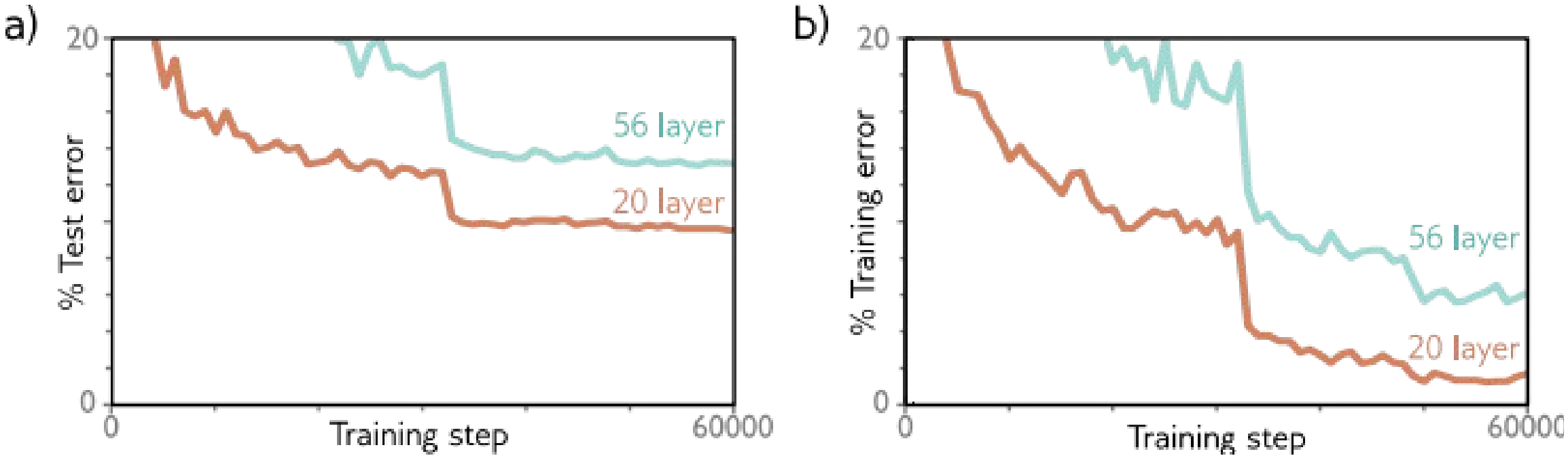
Semantic segmentation results



Convolution #2

- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

CIFAR Image classification for deeper networks



Regular network:

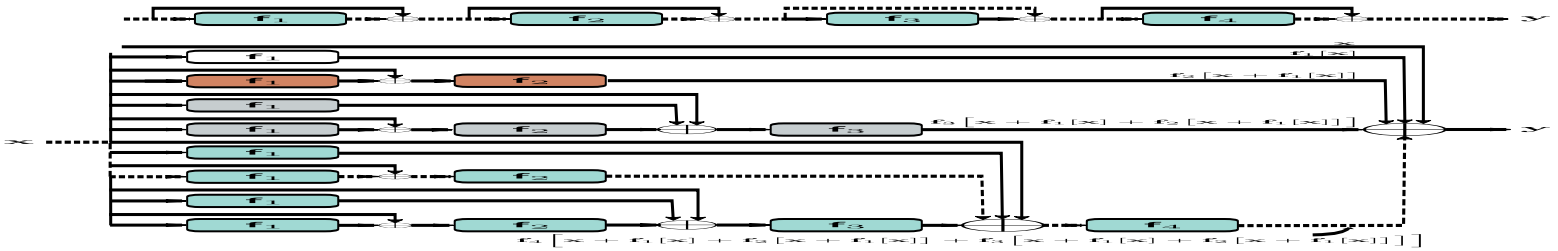
$$\mathbf{h}_1 = \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1]$$
$$\mathbf{h}_2 = \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2]$$
$$\mathbf{h}_3 = \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3]$$
$$\mathbf{y} = \mathbf{f}_4[\mathbf{h}_3, \boldsymbol{\phi}_4]$$

Regular network:

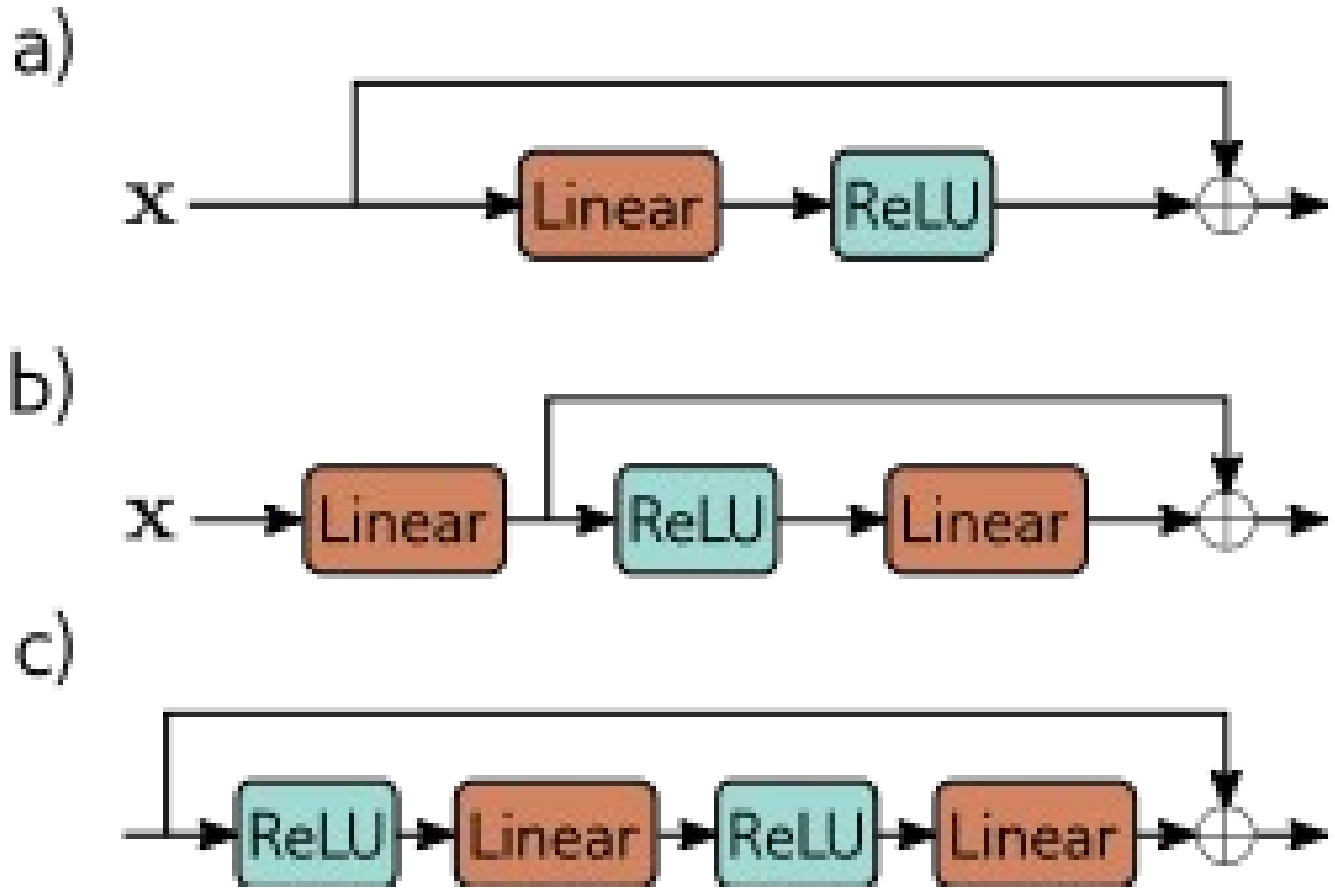
$$\mathbf{h}_1 = \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1]$$
$$\mathbf{h}_2 = \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2]$$
$$\mathbf{h}_3 = \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3]$$
$$\mathbf{y} = \mathbf{f}_4[\mathbf{h}_3, \boldsymbol{\phi}_4]$$

Residual network (2016):

$$\mathbf{h}_1 = \mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1]$$
$$\mathbf{h}_2 = \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2]$$
$$\mathbf{h}_3 = \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3]$$
$$\mathbf{y} = \mathbf{h}_3 + \mathbf{f}_4[\mathbf{h}_3, \boldsymbol{\phi}_4]$$

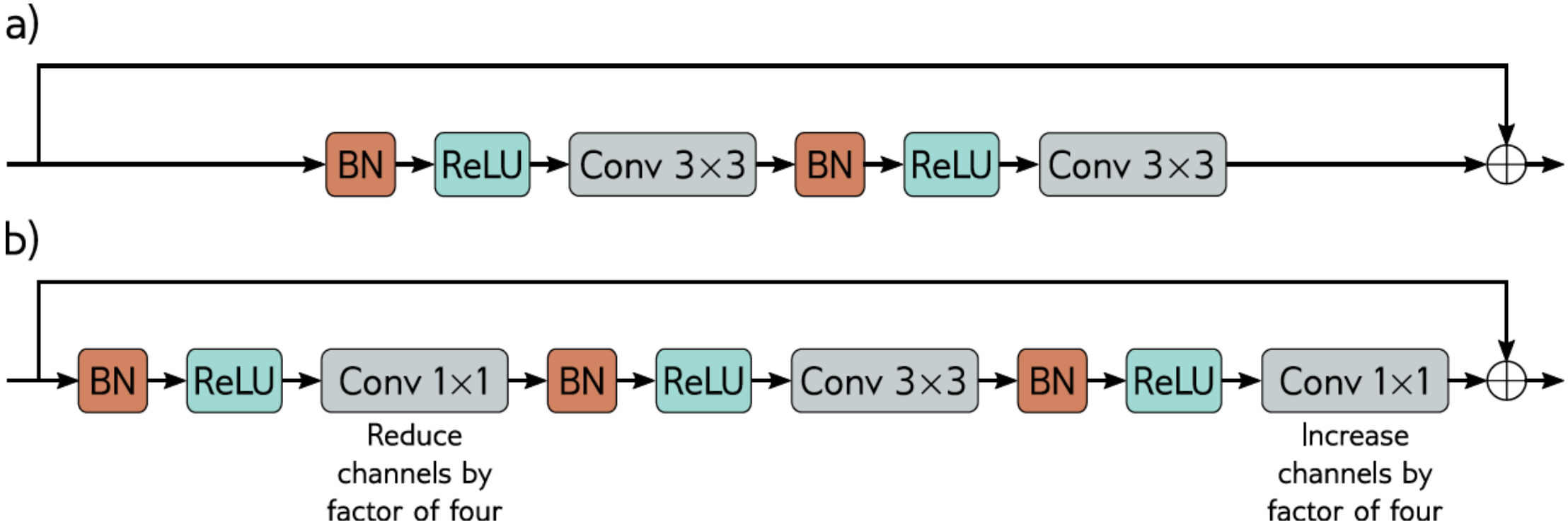


Order of operations is important

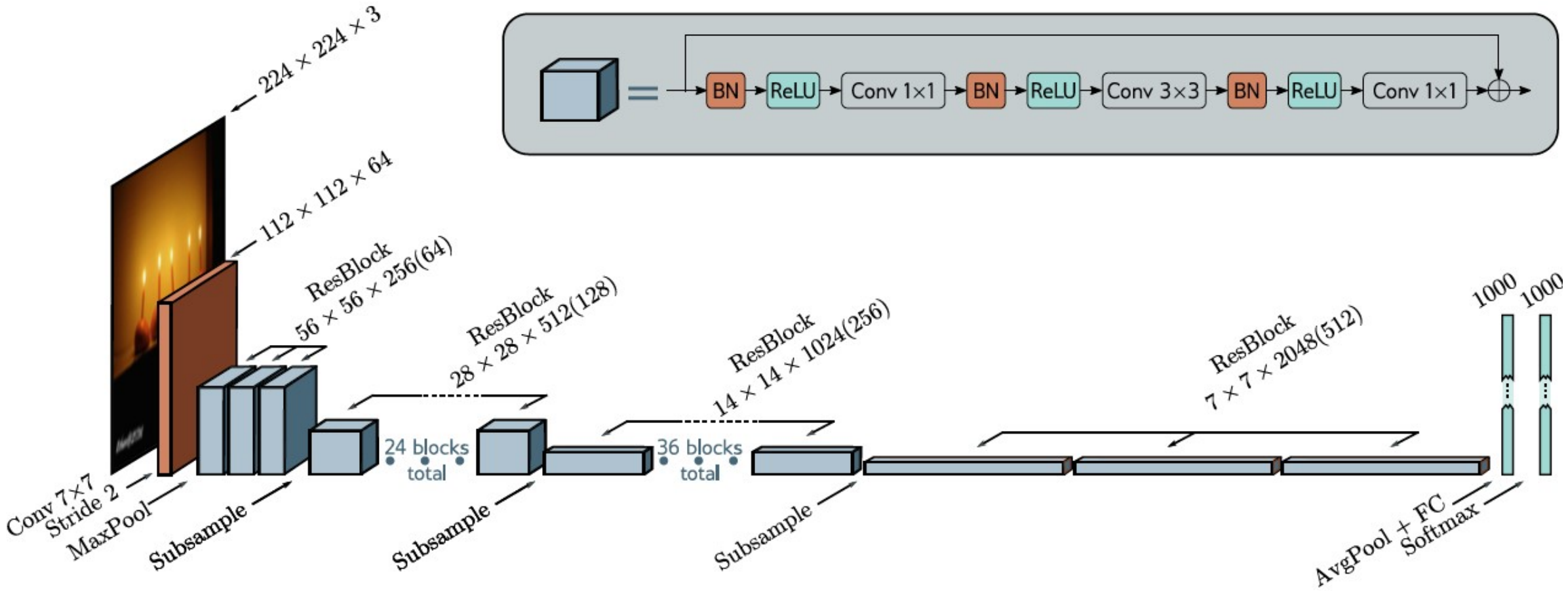


+ Batch Norm

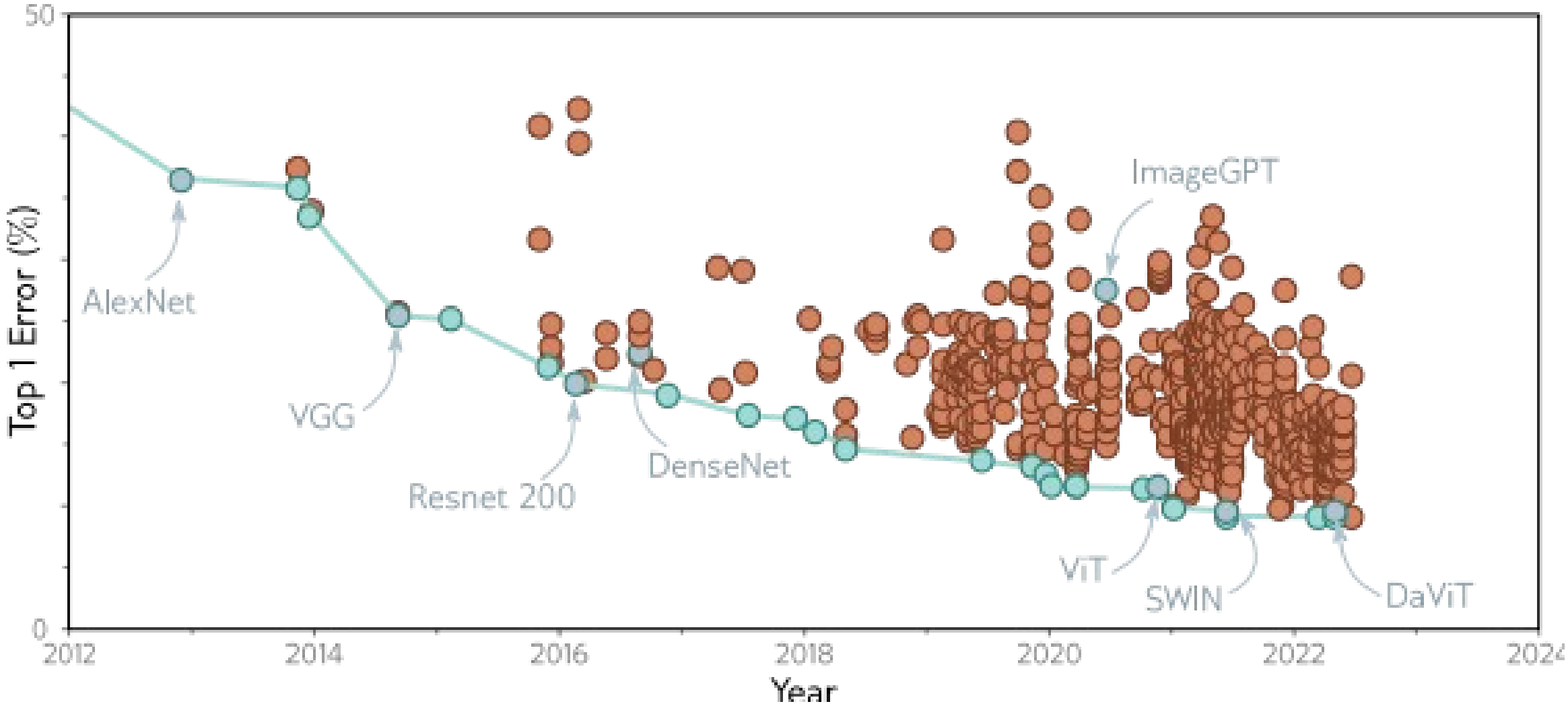
Resnet Block



Resnet 200 (2016)



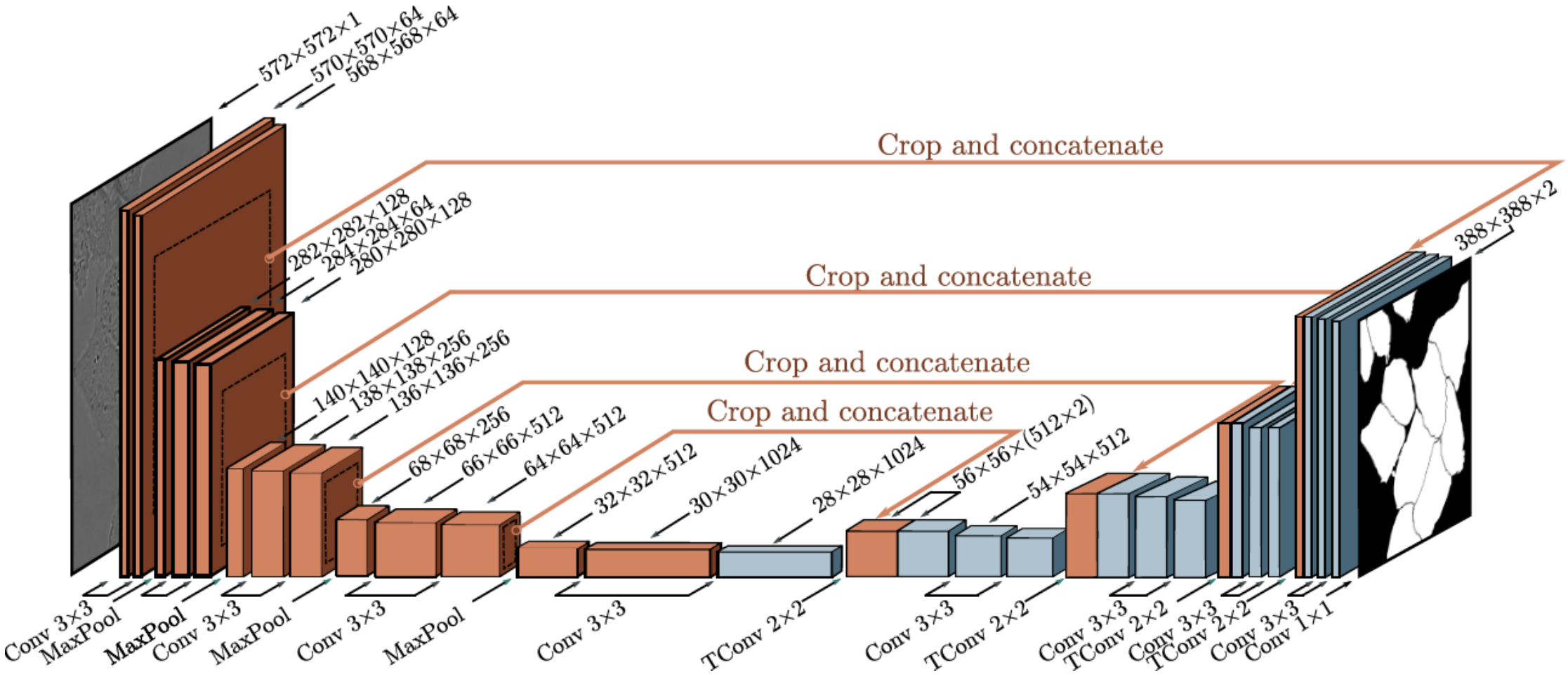
ImageNet History



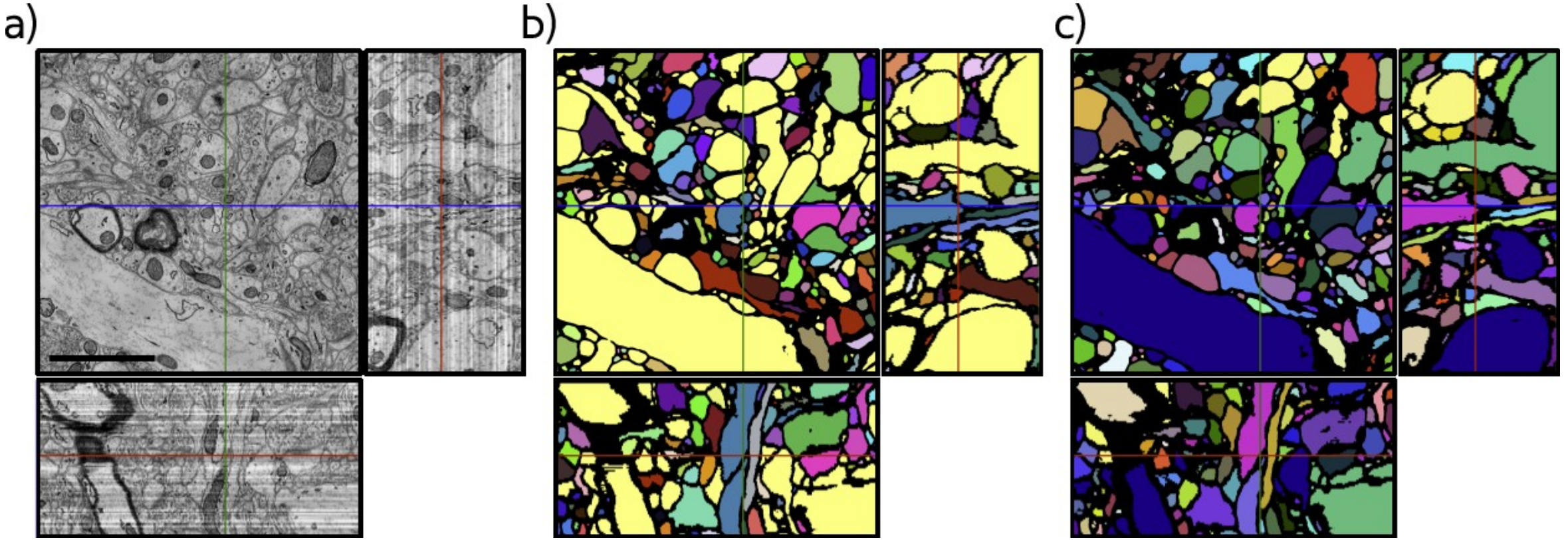
Convolution #2

- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

U-Net (2016)


572×572×1
570×570×64
568×568×64
Crop and concatenate
282×282×128
284×284×64
280×280×128
Crop and concatenate
140×140×128
138×138×256
136×136×256
Crop and concatenate
68×68×256
66×66×512
64×64×512
Crop and concatenate
32×32×512
30×30×1024
28×28×1024
56×56×(512×2)
54×54×512
388×388×2
Conv 3×3
MaxPool
MaxPool
Conv 3×3
MaxPool
Conv 3×3
MaxPool
Conv 3×3
TConv 2×2
Conv 3×3
TConv 2×2
Conv 3×3
TConv 2×2
Conv 3×3
TConv 2×2
Conv 3×3
Conv 1×1

U-Net Results



Stacked hourglass networks (2016)