

CM20315 - Machine Learning

Prof. Simon Prince

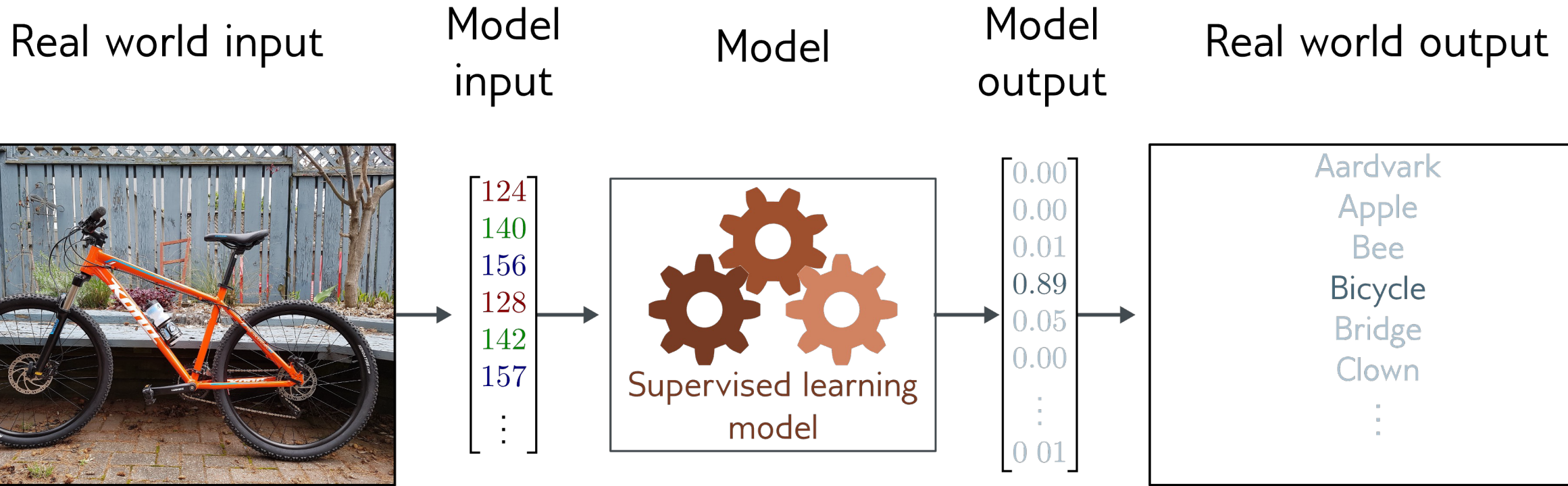
10. Convolutional networks



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

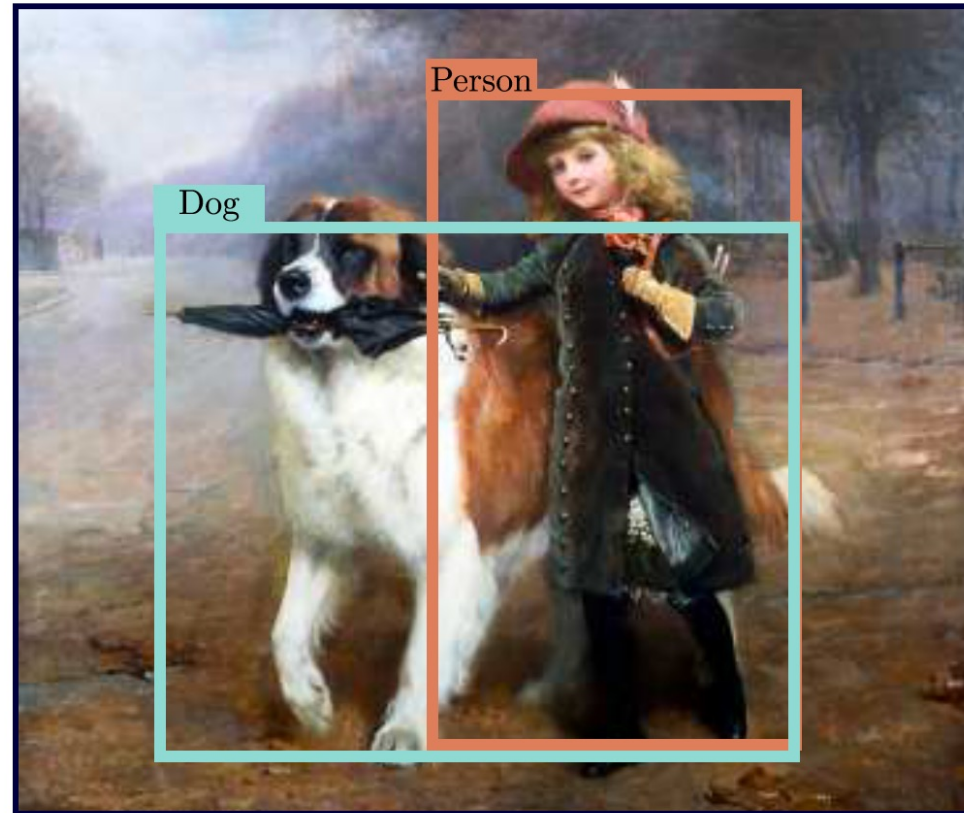
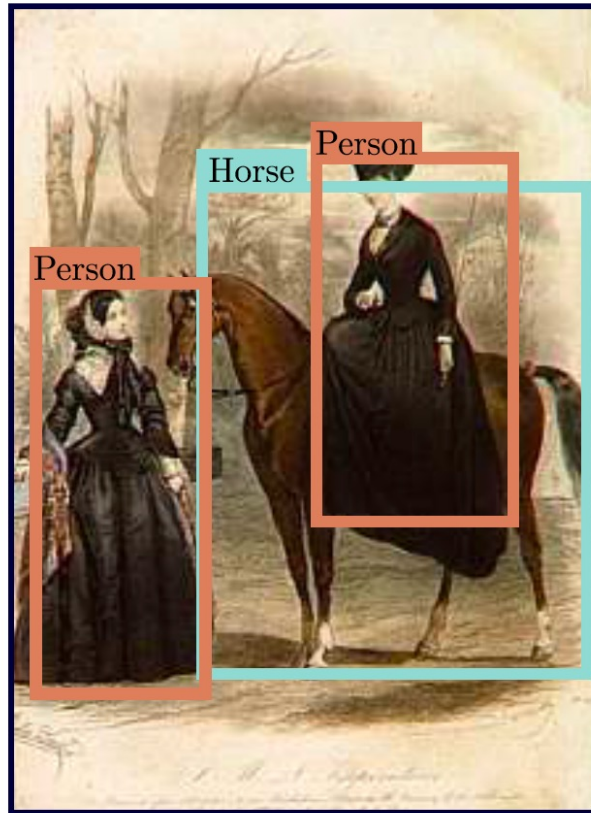
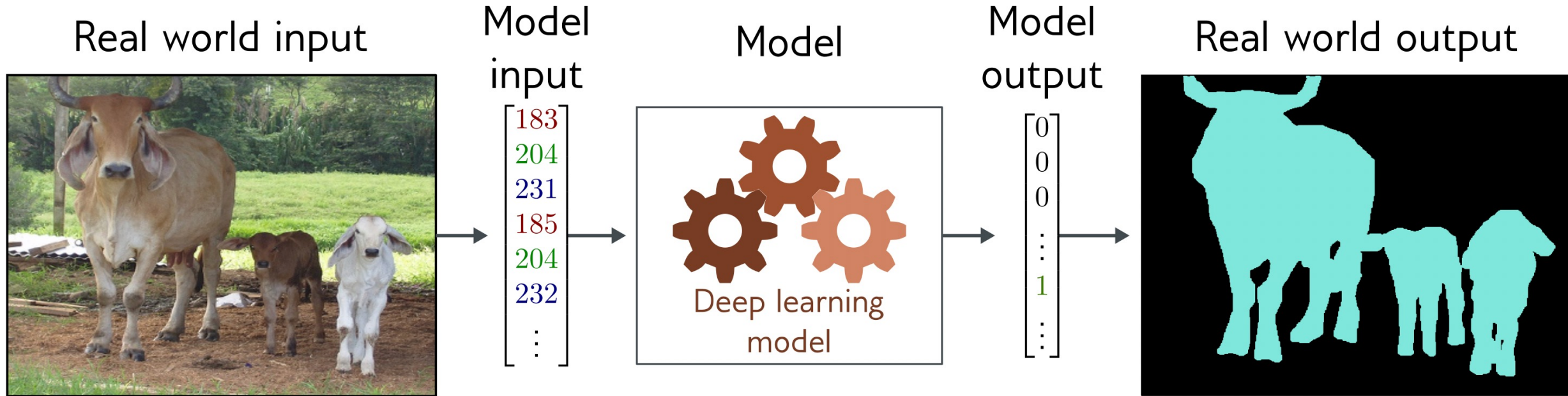


Image segmentation



- 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 \mathbf{x} :

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

- 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**:

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

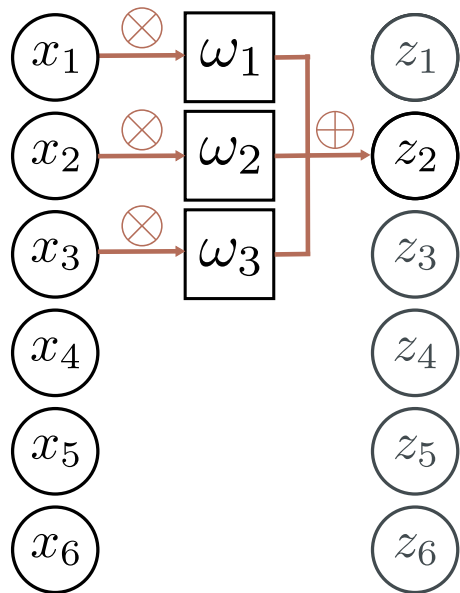
Kernel size = 3



* Not really technically convolution

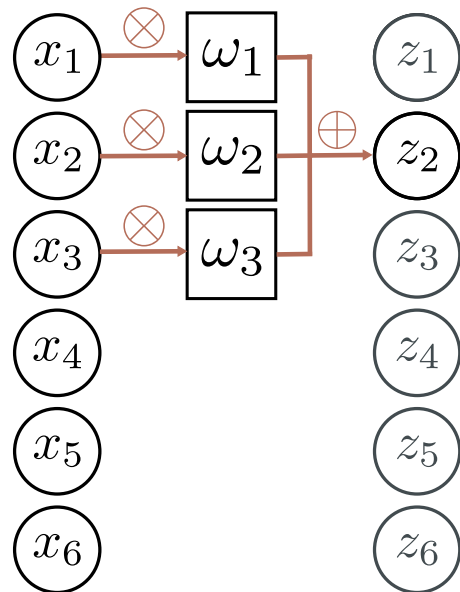
Convolution with kernel size 3

a)

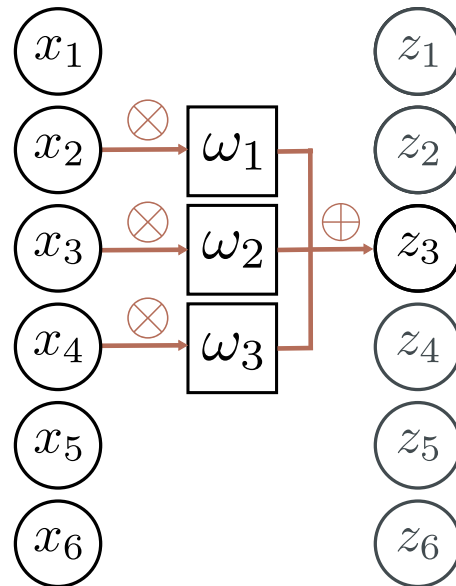


Convolution with kernel size 3

a)

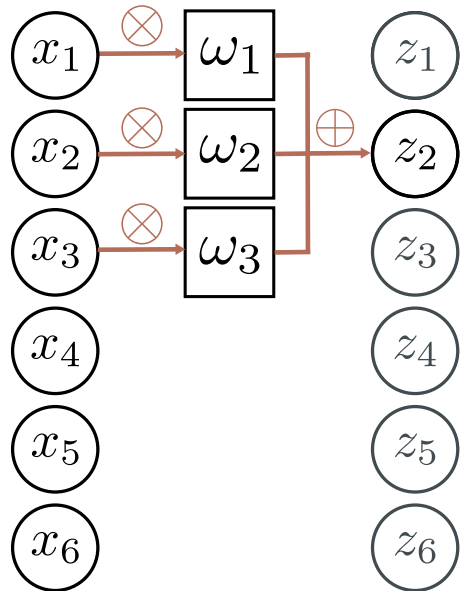


b)

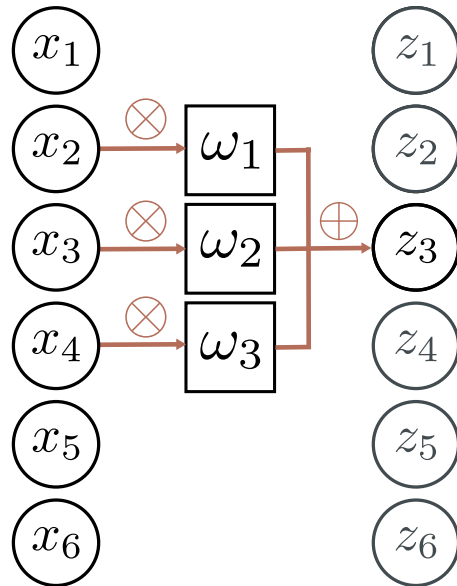


Convolution with kernel size 3

a)



b)

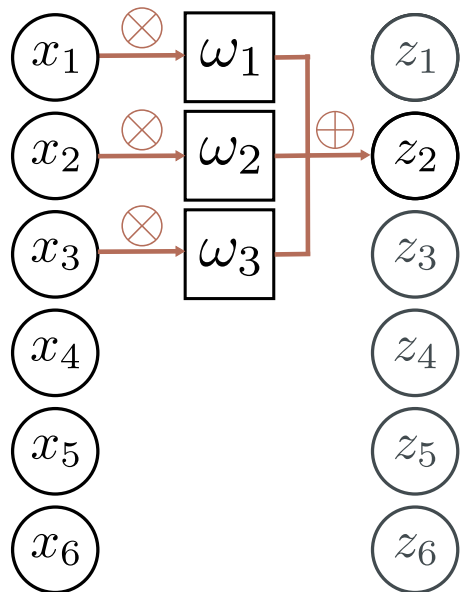


Equivariant to translation of input

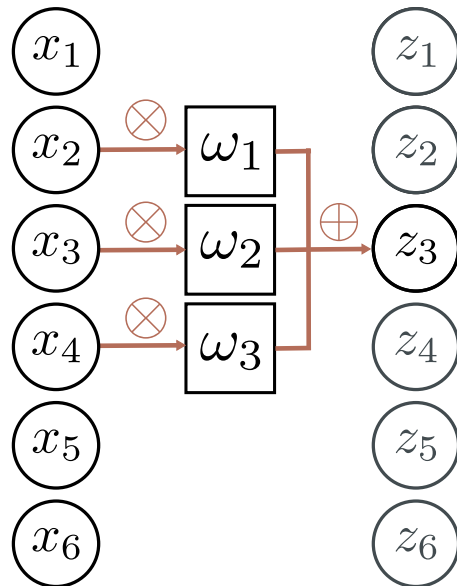
$$\mathbf{f}[\mathbf{t}[\mathbf{x}]] = \mathbf{t}[\mathbf{f}[\mathbf{x}]]$$

Zero padding

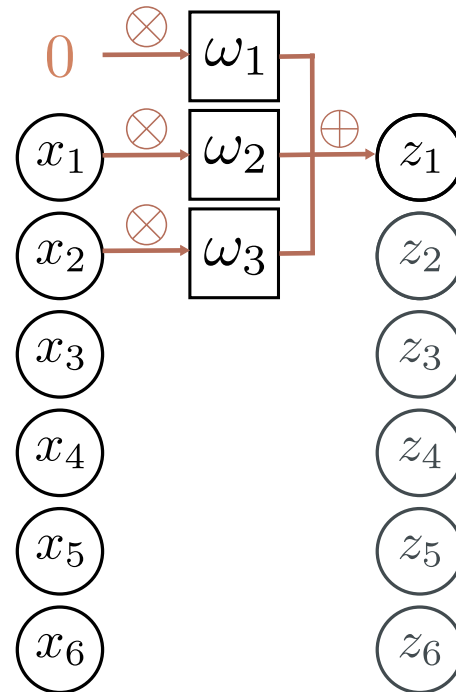
a)



b)

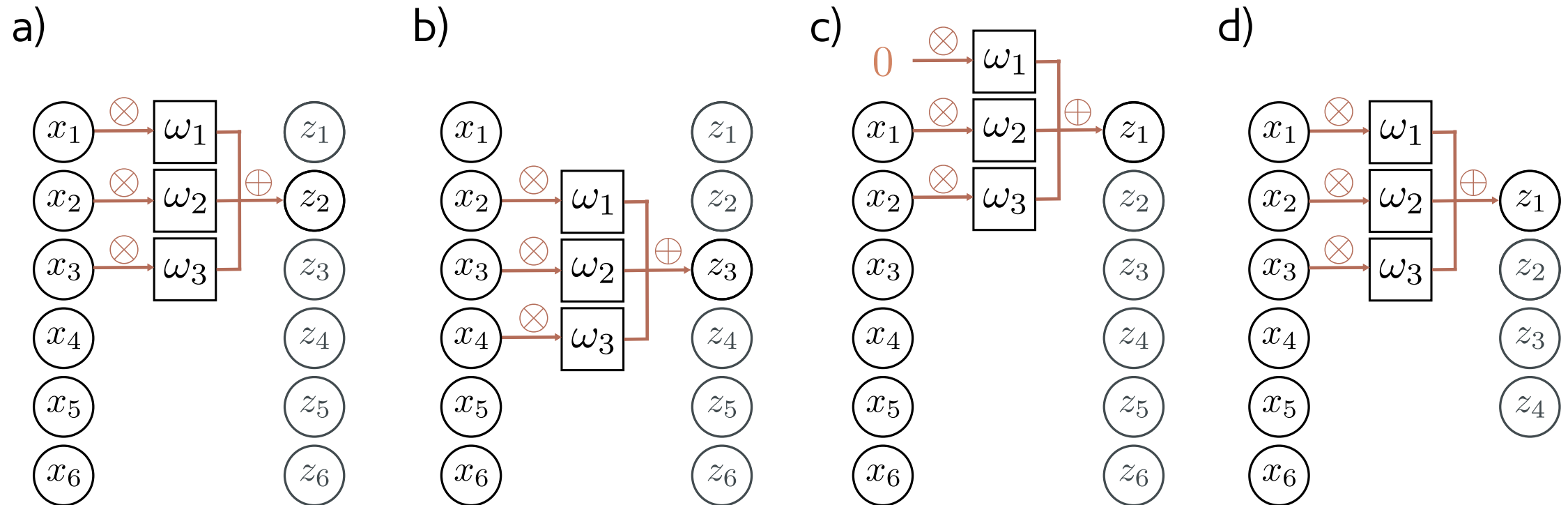


c)



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

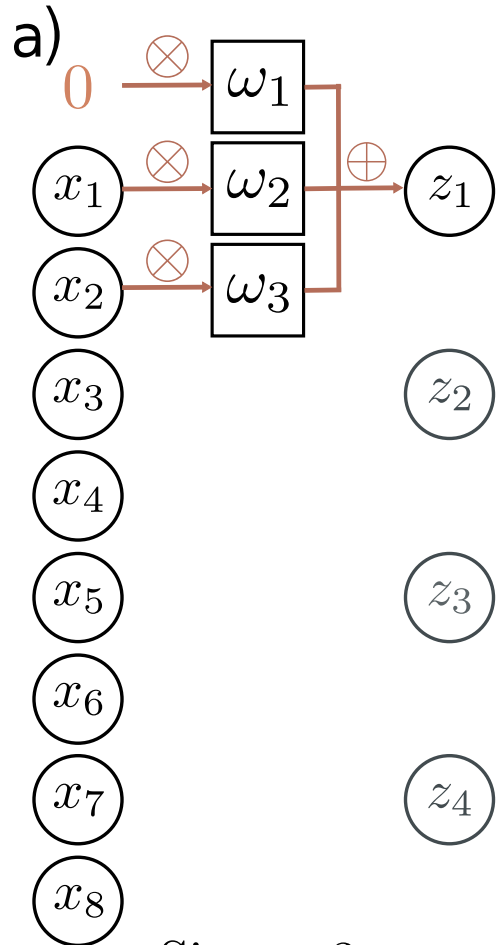
“Valid” convolutions



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

Stride, kernel size, and dilation

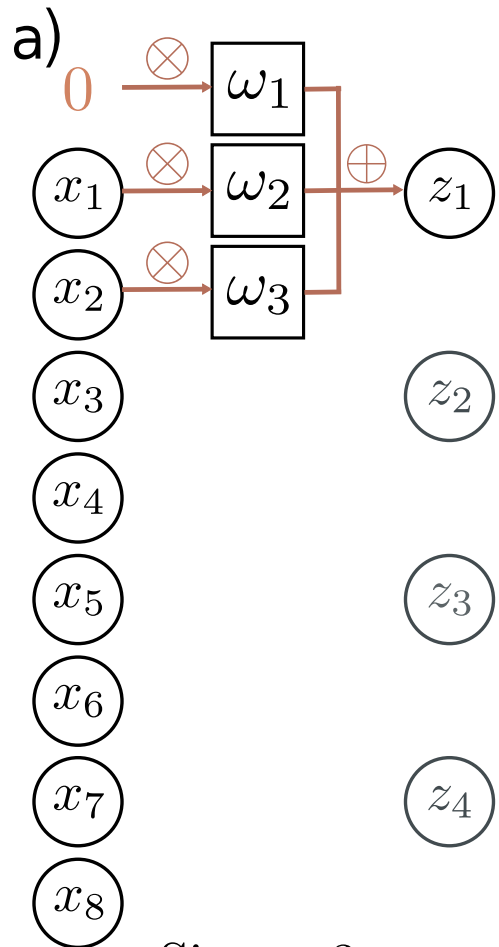
- **Stride** = shift by k positions for each output
 - Decreases size of output relative to input
- **Kernel size** = weight a different number of inputs for each output
 - Combine information from a larger area
 - But kernel size 5 uses 5 parameters
- **Dilated** or **atrous** convolutions = intersperse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



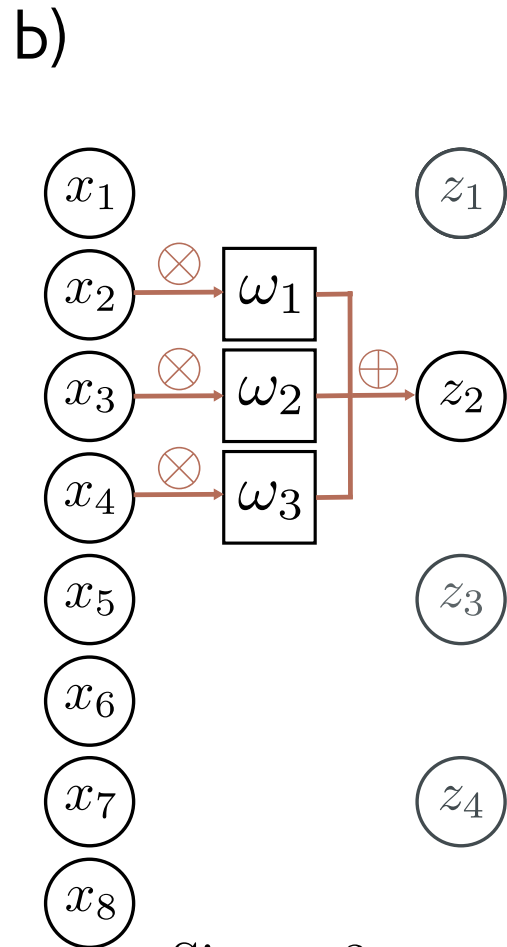
Size = 3

Stride = 2

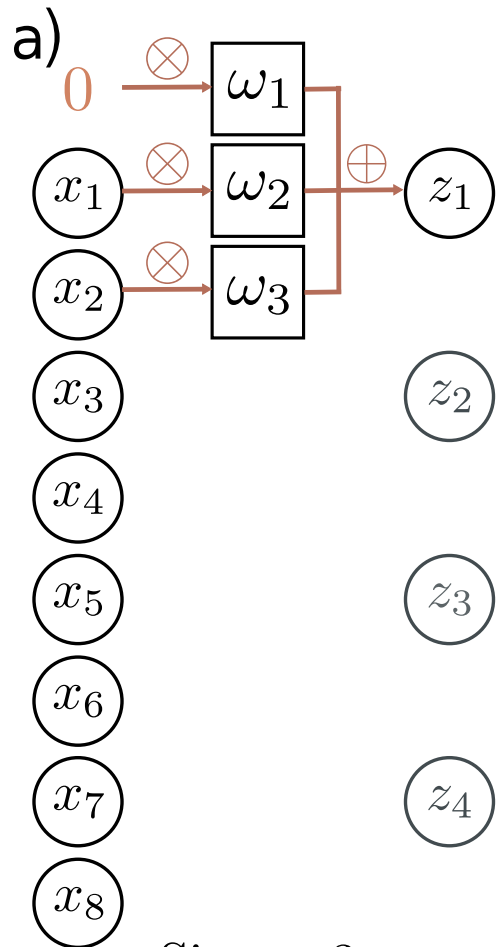
Dilation = 1



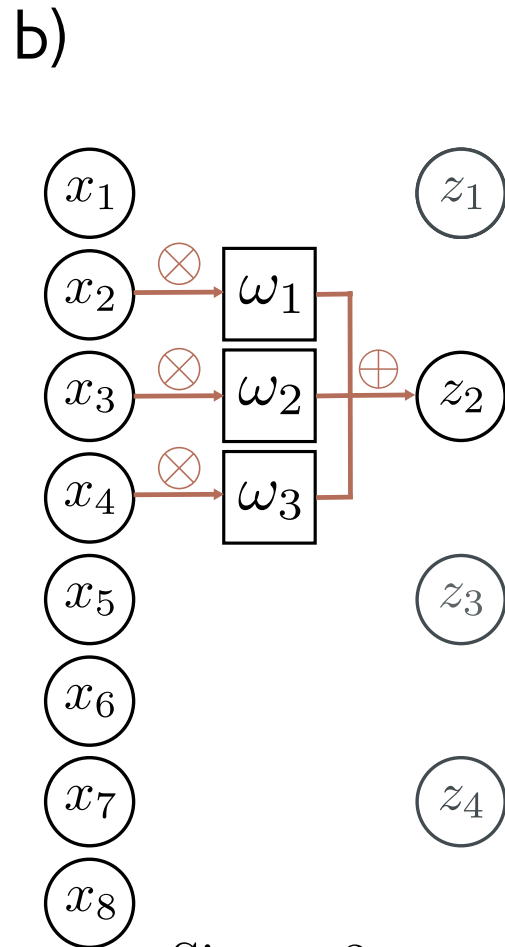
Size = 3
 Stride = 2
 Dilation = 1



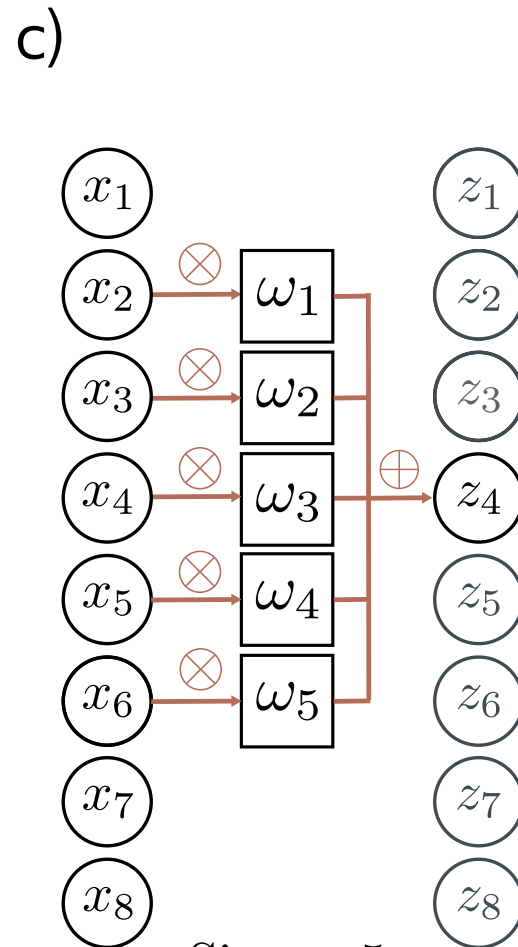
Size = 3
 Stride = 2
 Dilation = 1



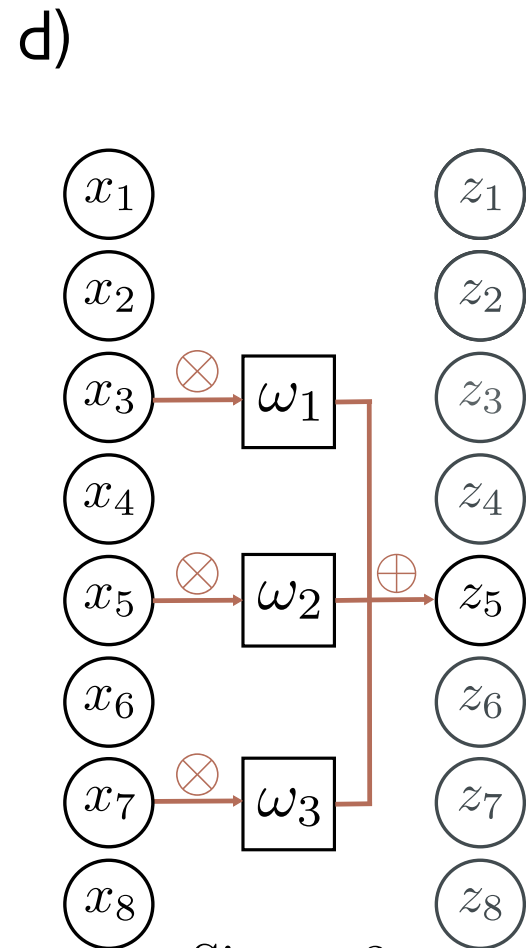
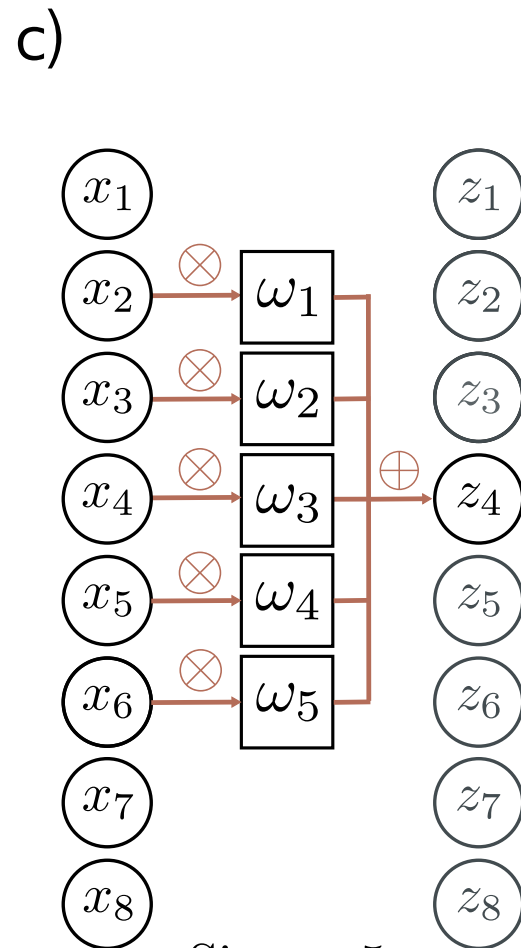
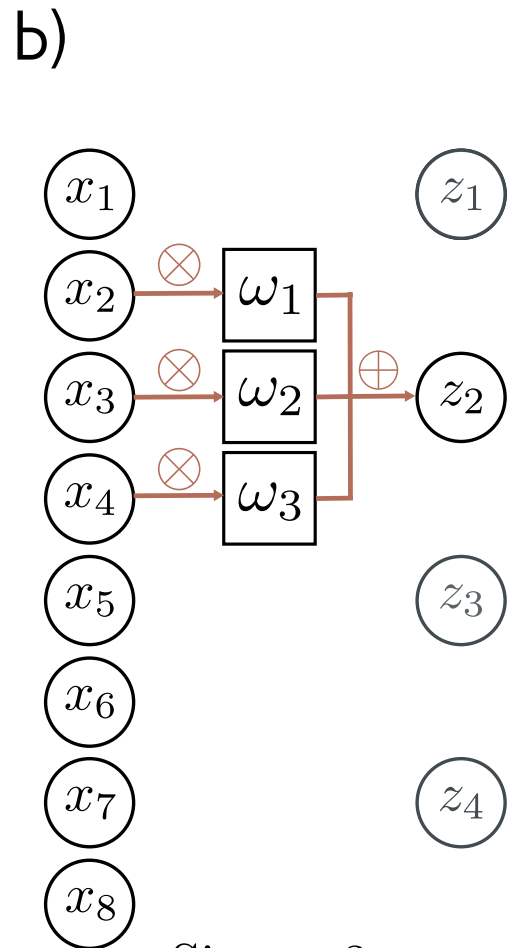
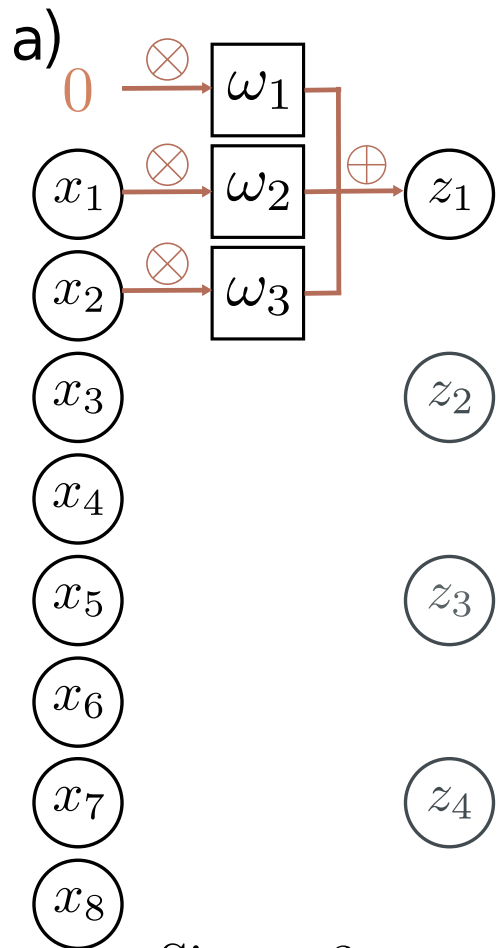
Size = 3
 Stride = 2
 Dilation = 1



Size = 3
 Stride = 2
 Dilation = 1



Size = 5
 Stride = 1
 Dilation = 1



Convolutional networks

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

Convolutional layer

$$\begin{aligned} h_i &= a [\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}] \\ &= a \left[\beta + \sum_{j=1}^3 \omega_j x_{i+j-2} \right] \end{aligned}$$

Special case of fully-connected network

Convolutional network:


$$\begin{aligned} h_i &= a [\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}] \\ &= a \left[\beta + \sum_{j=1}^3 \omega_j x_{i+j-2} \right] \end{aligned}$$

Fully connected network:

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


Special case of fully-connected network

Convolutional network:

$$h_i = a [\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}]$$
$$= a \left[\beta + \sum_{j=1}^3 \omega_j x_{i+j-2} \right]$$


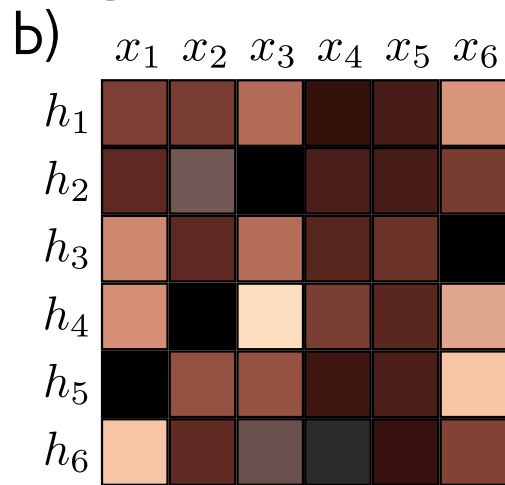
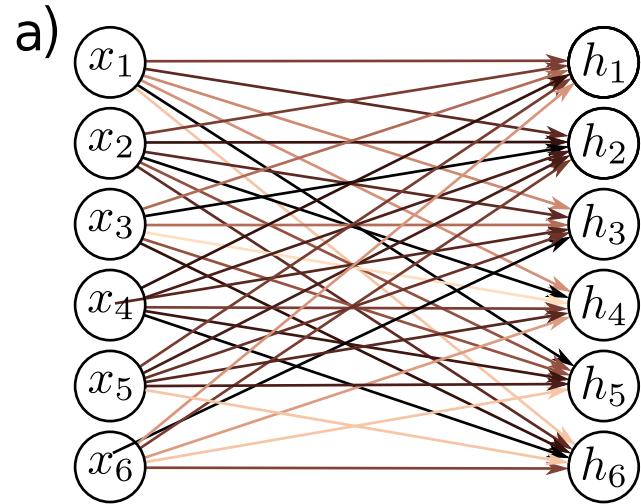
3 weights, 1 bias

Fully connected network:

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


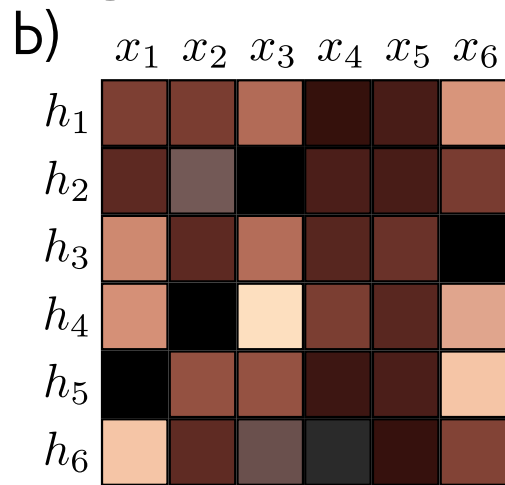
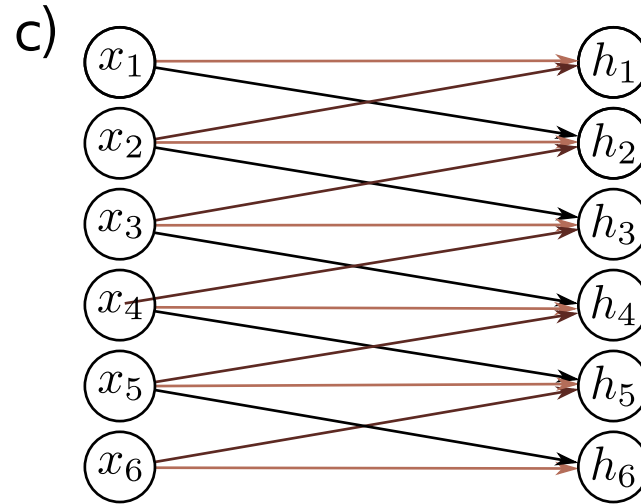
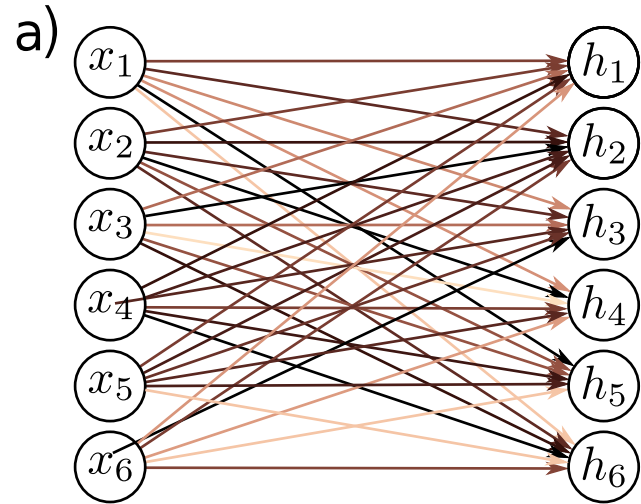
D^2 weights, D biases

Special case of fully-connected network

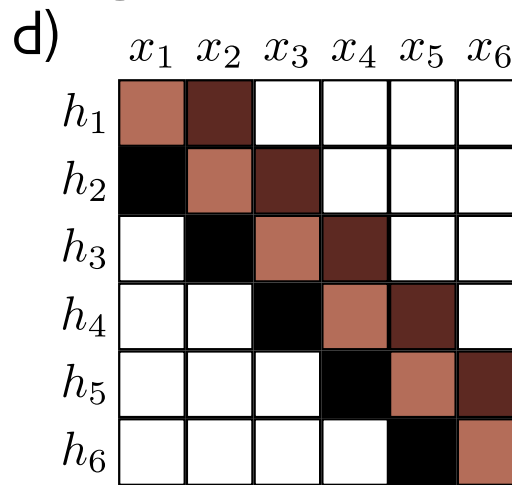


Fully connected network

Special case of fully-connected network

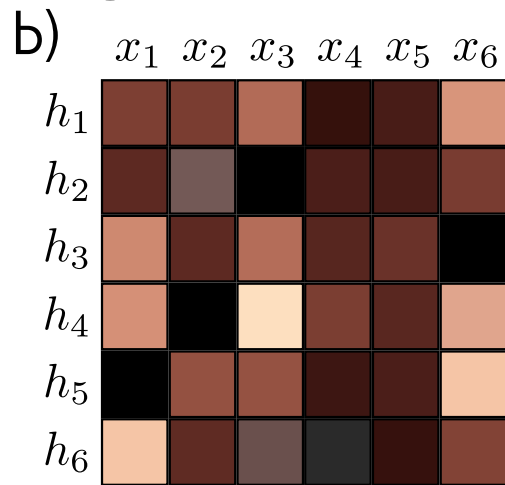
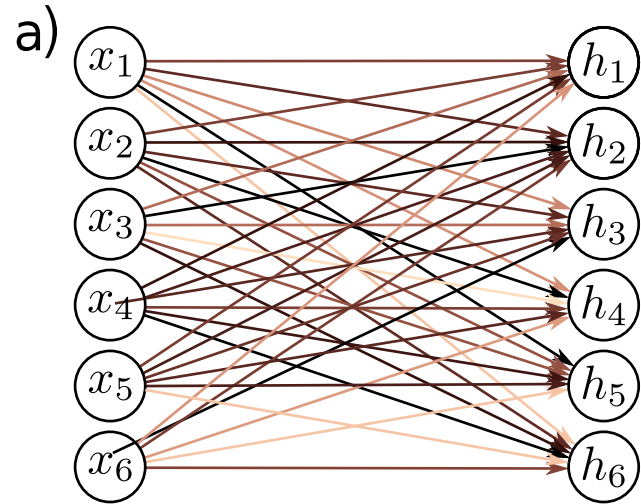


Fully connected network

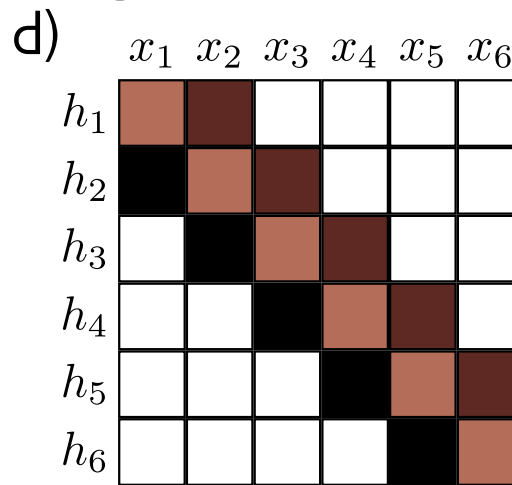
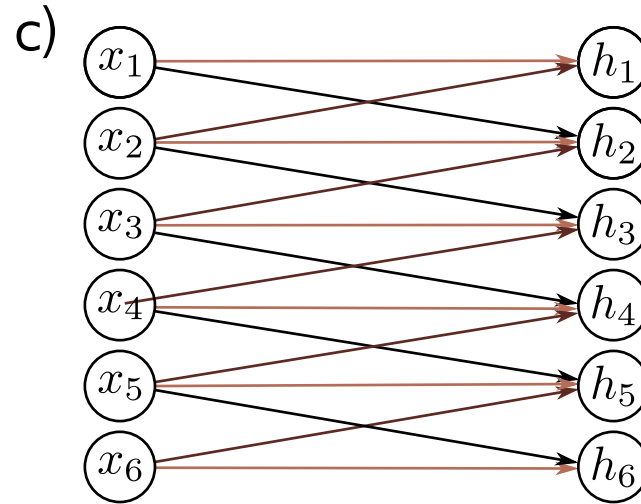


Convolution, kernel 3,
stride 1, dilation 1

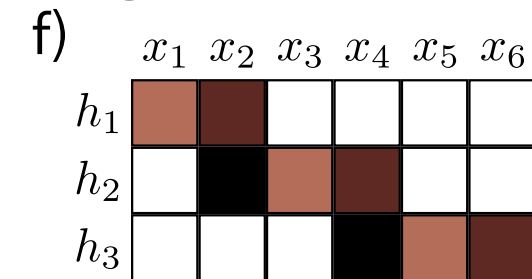
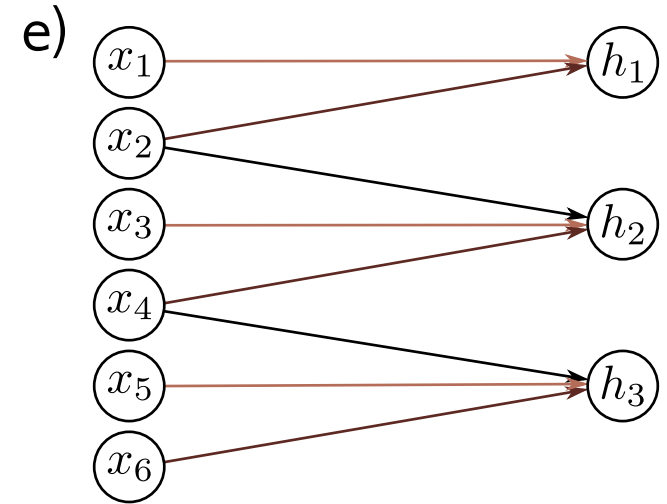
Special case of fully-connected network



Fully connected network



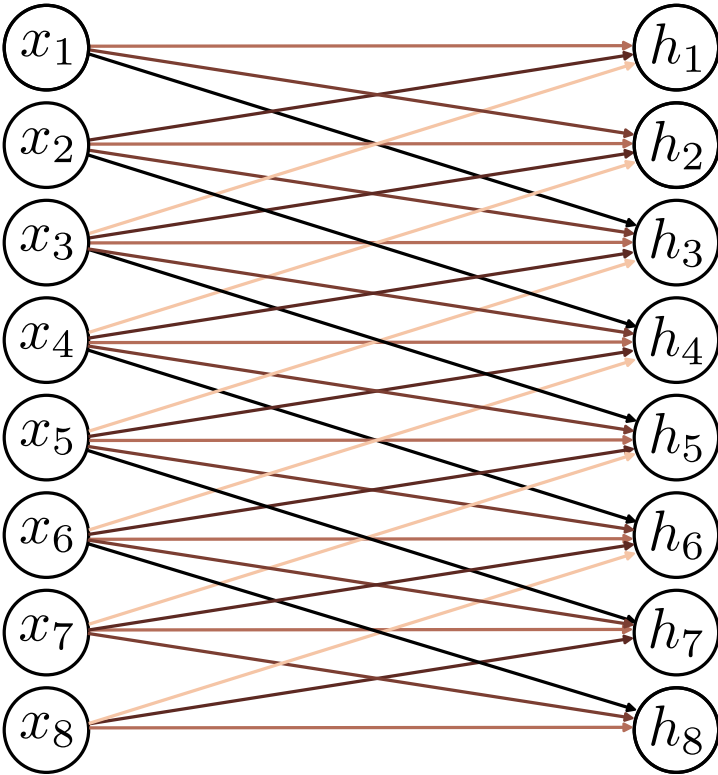
Convolution, size 3, stride 1,
dilation 1, zero padding



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

Question 1

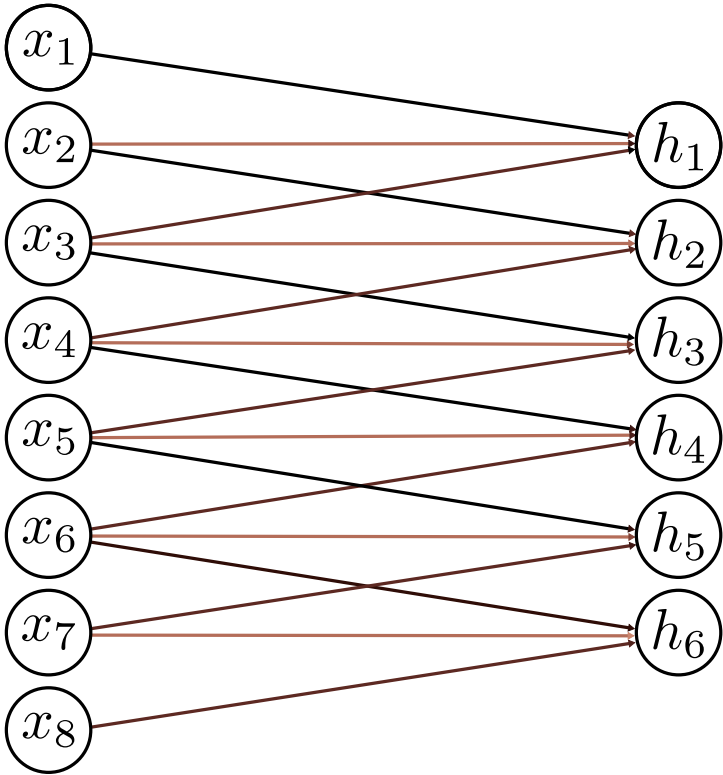
- Kernel size?
- Stride?
- Dilation?
- Zero padding / valid?



	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
h_1	Dark Brown	Dark Brown	Light Brown	White	White	White	White	White
h_2	Dark Brown	Dark Brown	Dark Brown	Light Brown	White	White	White	White
h_3	Black	Dark Brown	Dark Brown	Dark Brown	Light Brown	White	White	White
h_4	White	Black	Dark Brown	Dark Brown	Dark Brown	Light Brown	White	White
h_5	White	White	Black	Dark Brown	Dark Brown	Dark Brown	Light Brown	White
h_6	White	White	White	Black	Dark Brown	Dark Brown	Dark Brown	Light Brown
h_7	White	White	White	White	Black	Dark Brown	Dark Brown	Dark Brown
h_8	White	White	White	White	White	Black	Dark Brown	Dark Brown

Question 2

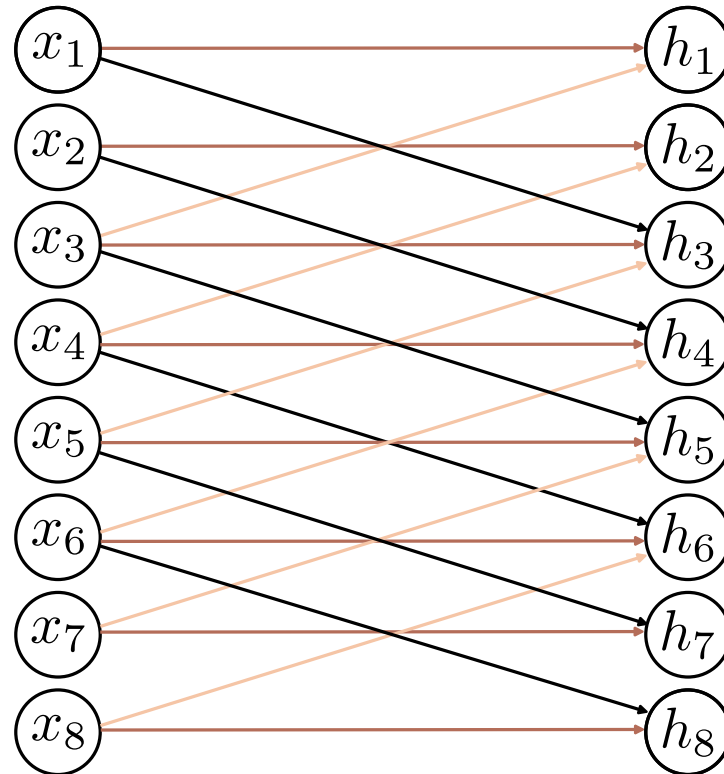
- Kernel size?
- Stride?
- Dilation?
- Zero padding / valid?



	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
h_1	Black	Light Brown	Dark Brown	White	White	White	White	White
h_2	White	Black	Light Brown	Dark Brown	White	White	White	White
h_3	White	White	Black	Light Brown	Dark Brown	White	White	White
h_4	White	White	White	Black	Light Brown	Dark Brown	White	White
h_5	White	White	White	White	Black	Light Brown	Dark Brown	White
h_6	White	White	White	White	White	Black	Light Brown	Dark Brown

Question 3

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



	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
h_1	Dark Red	White	Light Orange	White	White	White	White	White
h_2	White	Dark Red	White	Light Orange	White	White	White	White
h_3	Black	White	Dark Red	White	Light Orange	White	White	White
h_4	White	Black	White	Dark Red	White	Light Orange	White	White
h_5	White	White	Black	White	Dark Red	White	Light Orange	White
h_6	White	White	White	Black	White	Dark Red	White	Light Orange
h_7	White	White	White	White	Black	White	Dark Red	White
h_8	White	White	White	White	White	Black	White	Dark Red

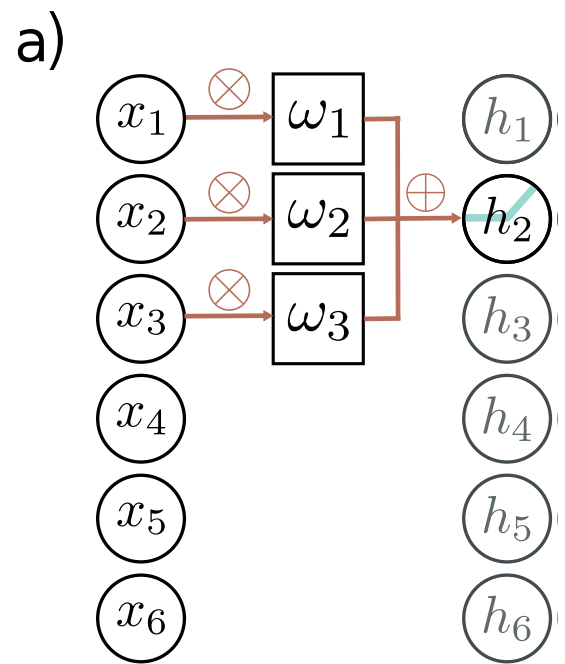
Convolutional networks

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

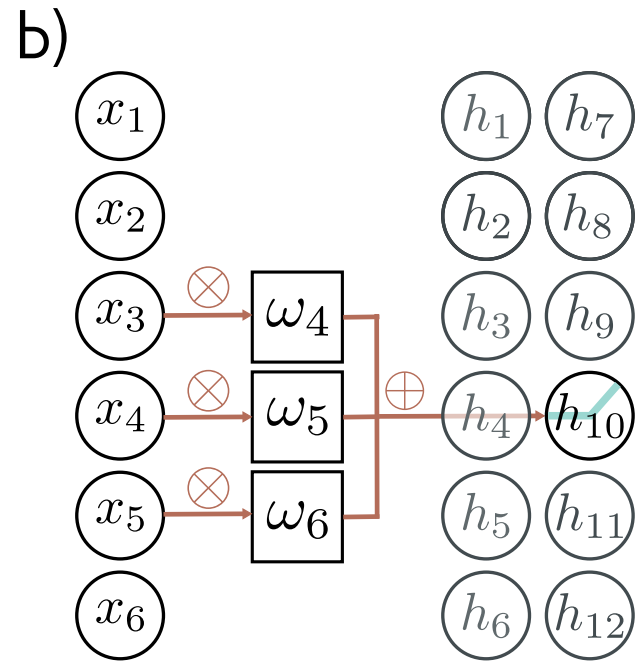
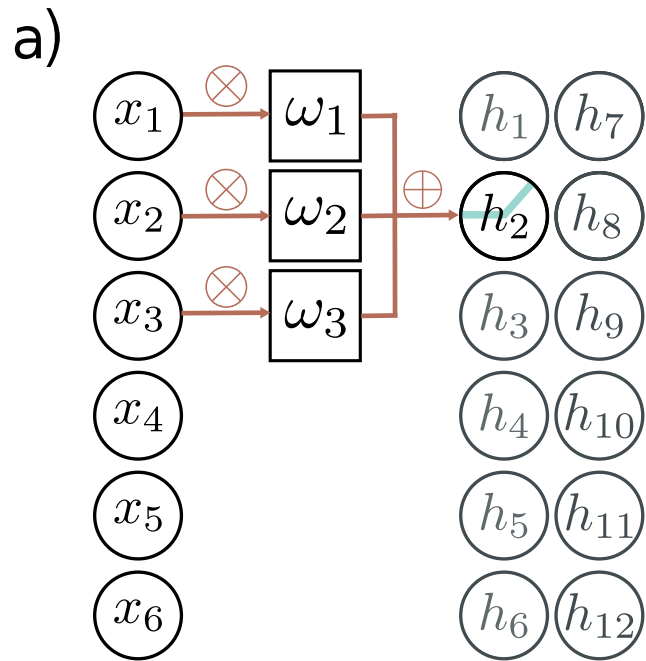
Channels

- 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**

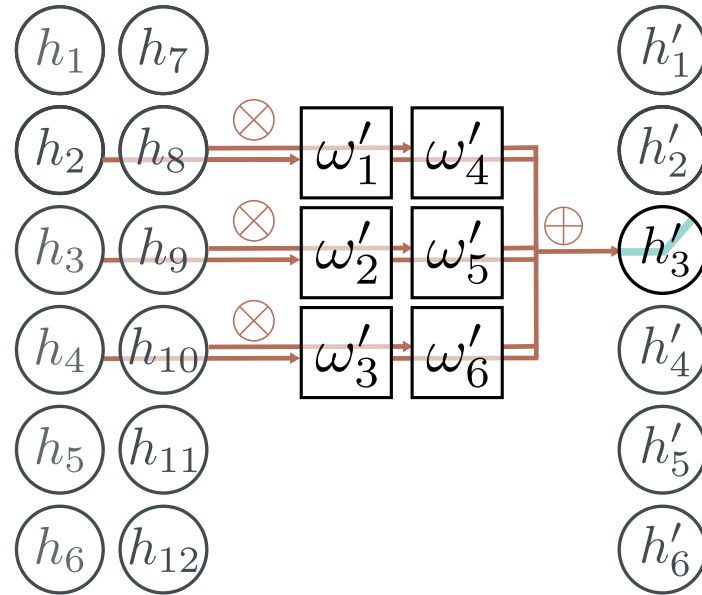
Two output channels, one input channel



Two output channels, one input channel



Two input channels, one output channel



How many parameters?

- If there are C_i input channels and kernel size K

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

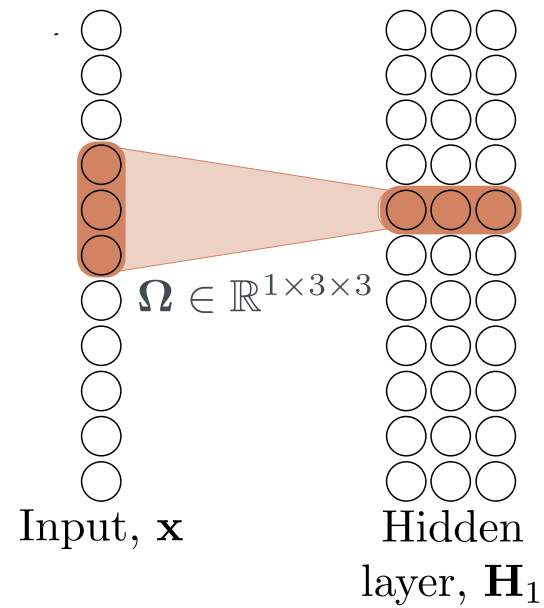
- If there are C_i input channels and C_o output channels

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

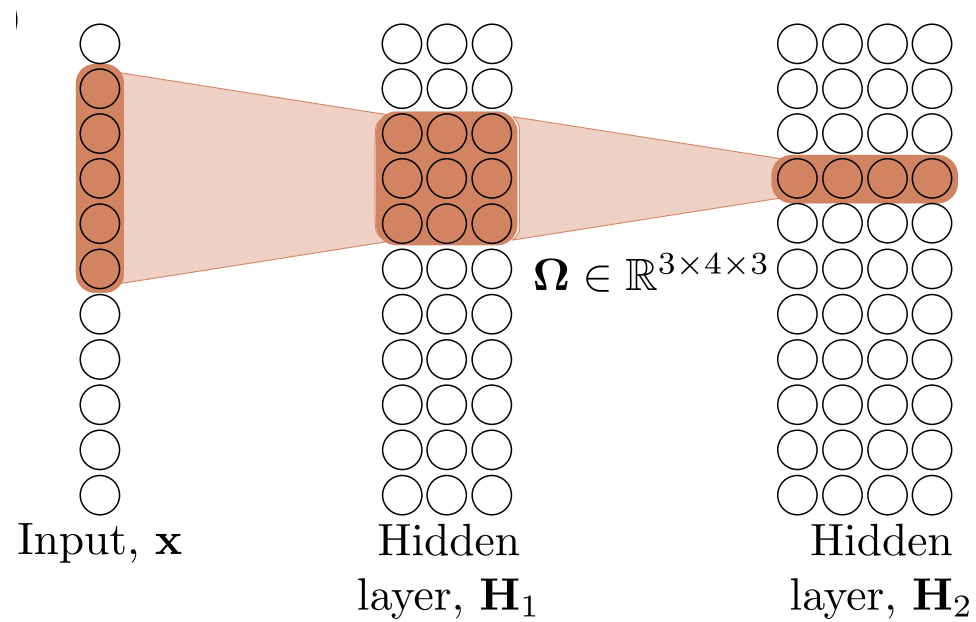
Convolutional networks

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

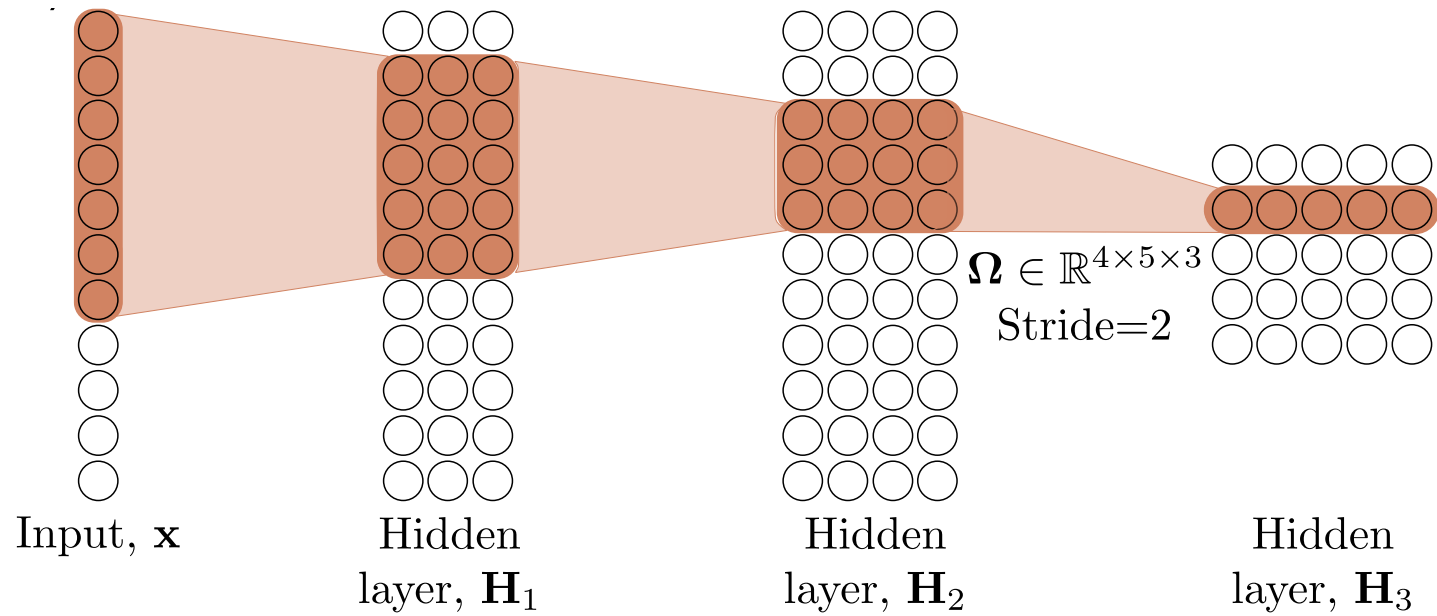
Receptive fields



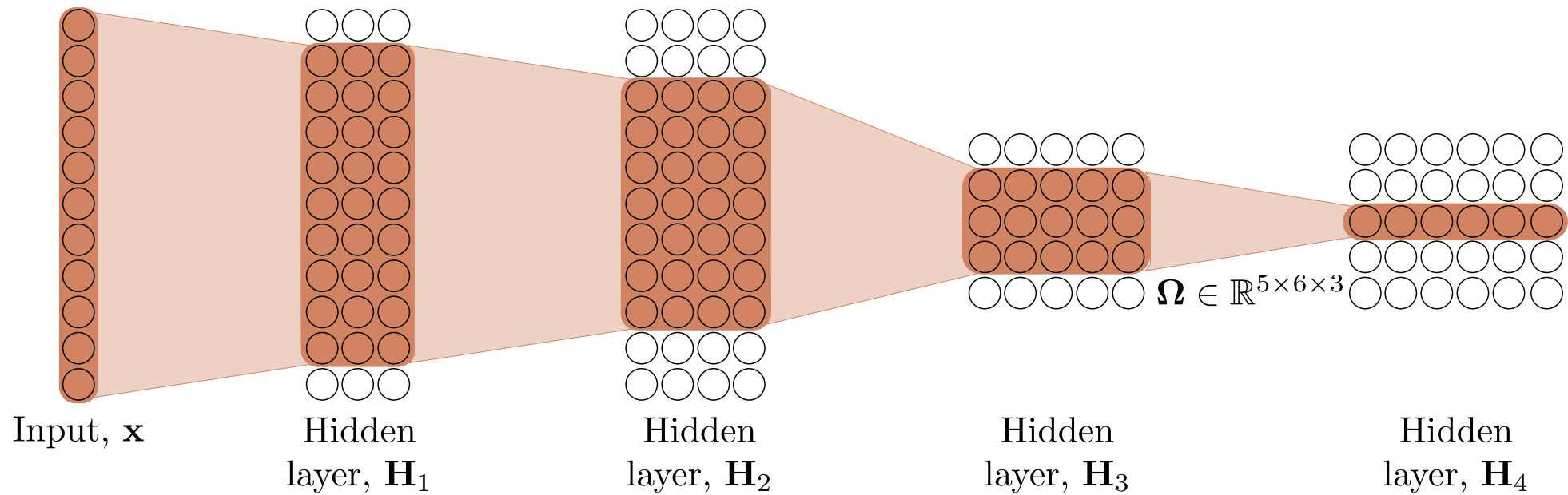
Receptive fields



Receptive fields



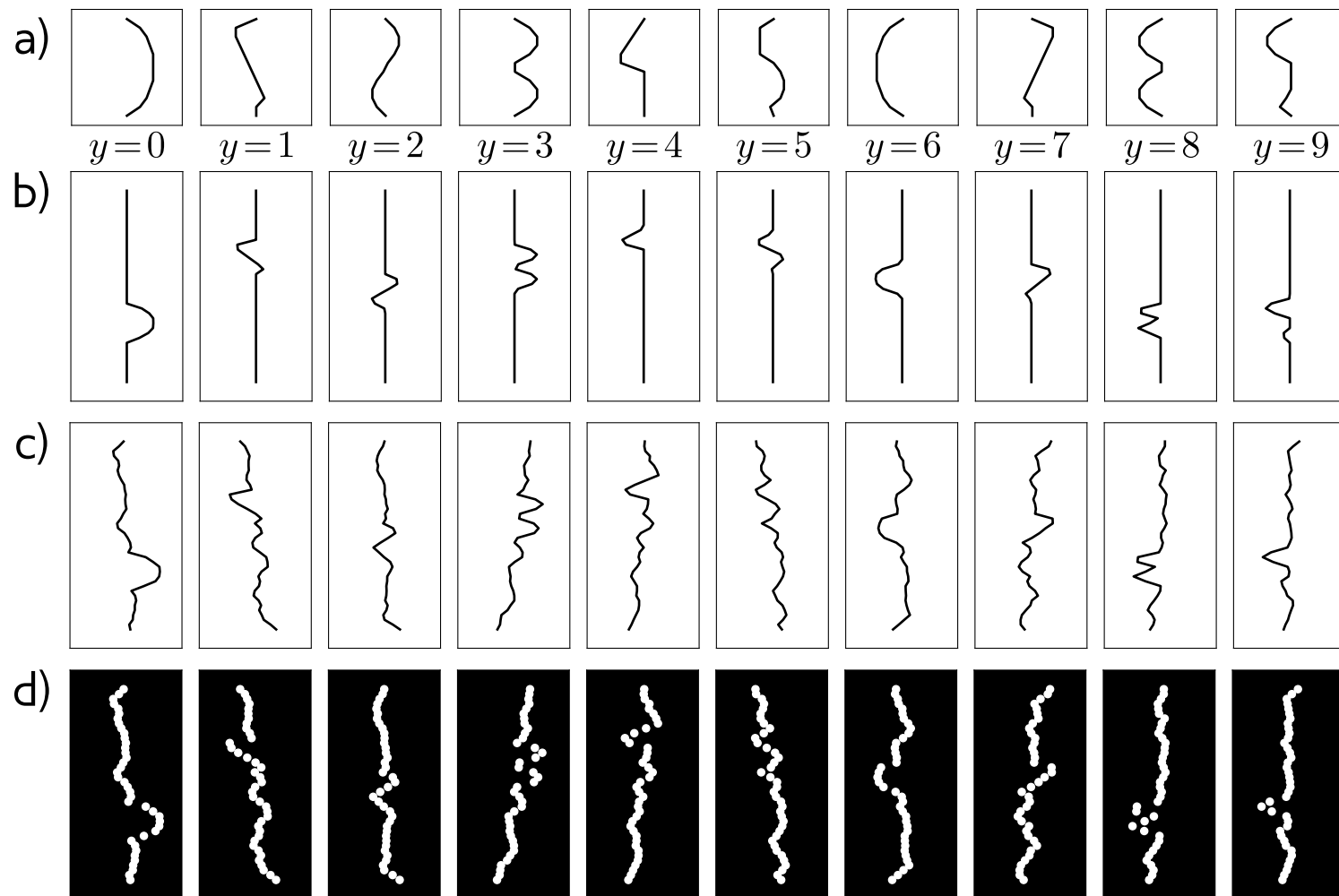
Receptive fields



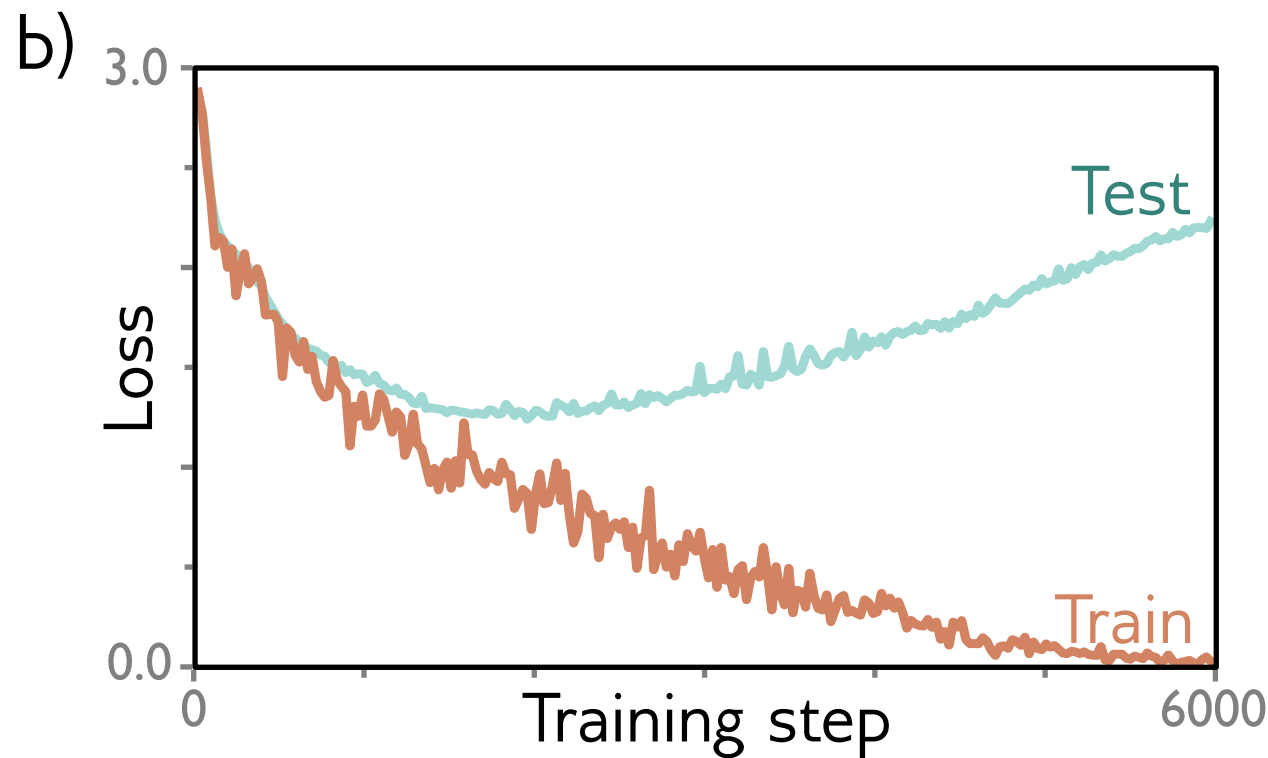
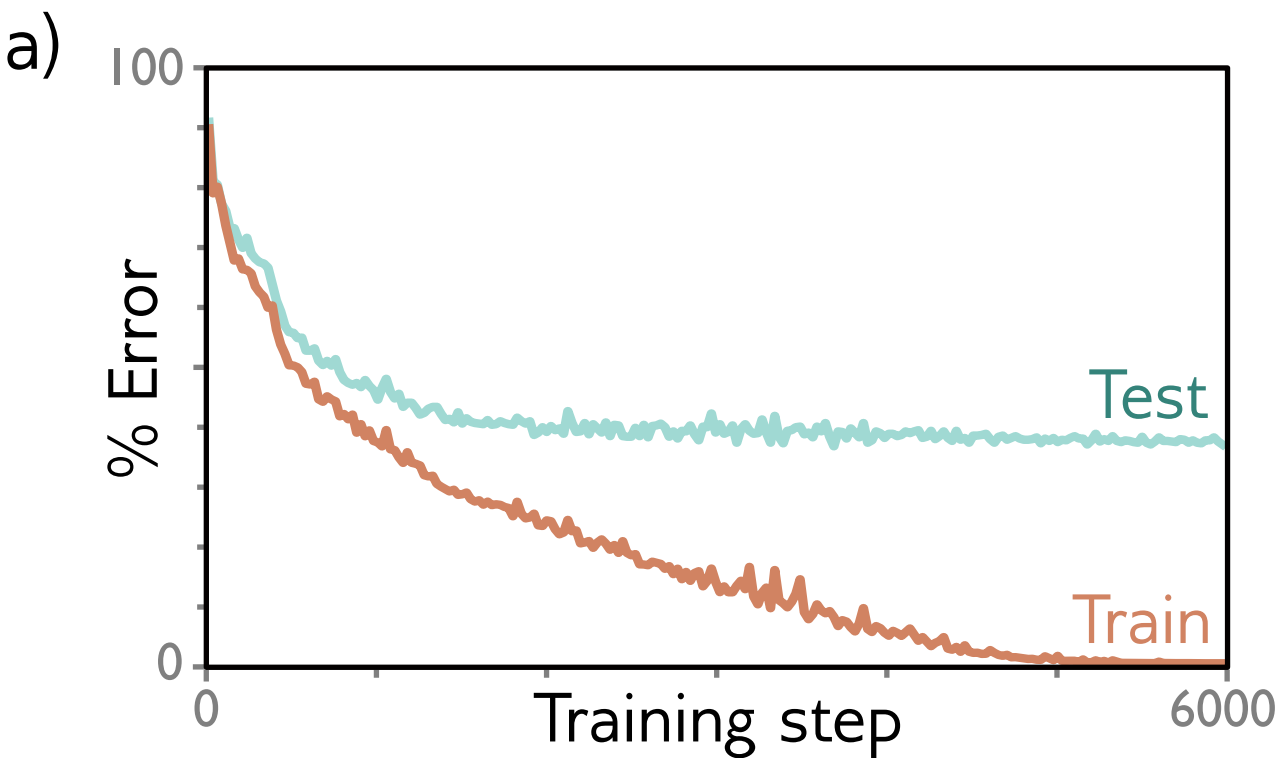
Convolutional networks

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

MNIST 1D Dataset



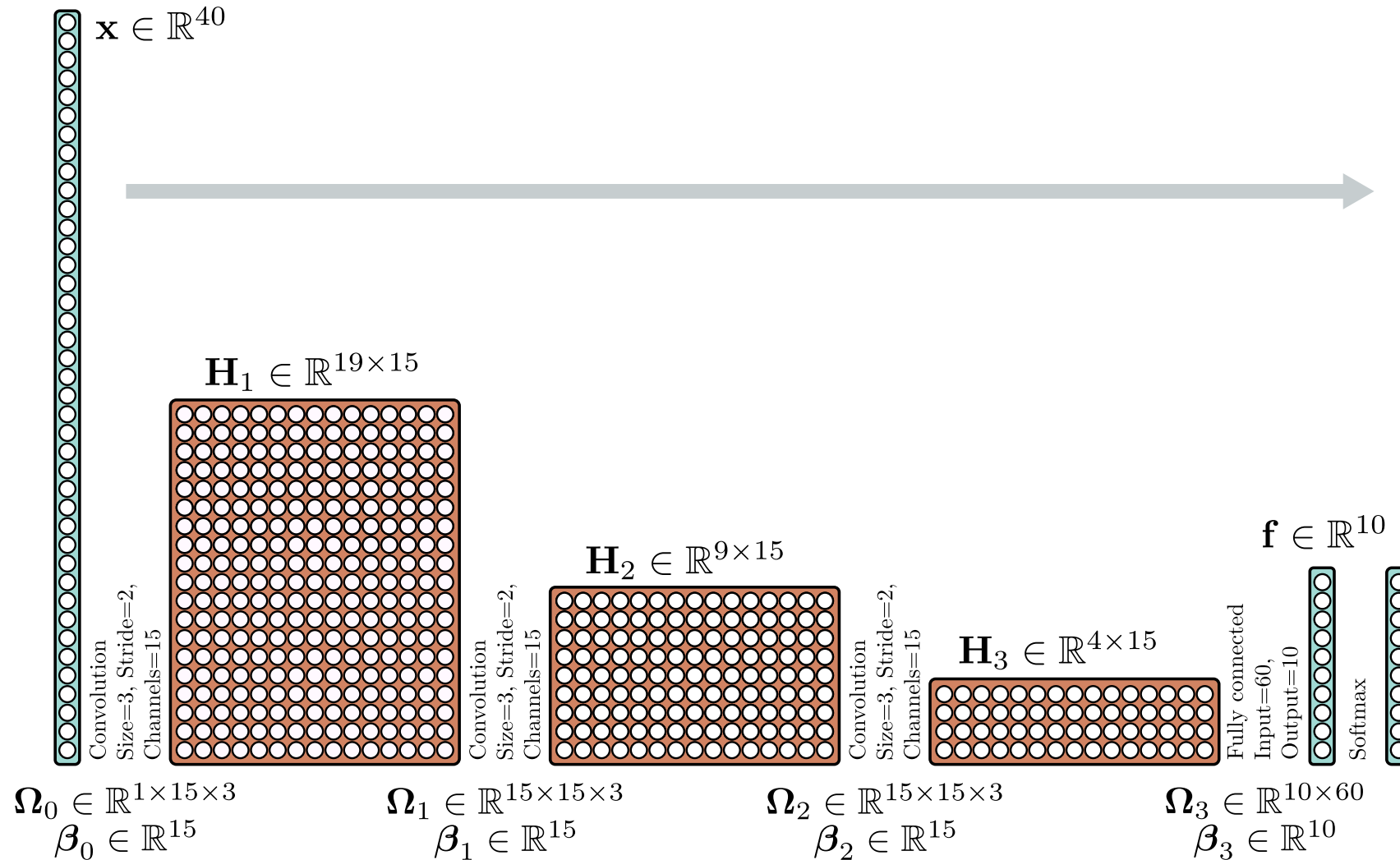
MNIST-1D results for fully-connected network



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

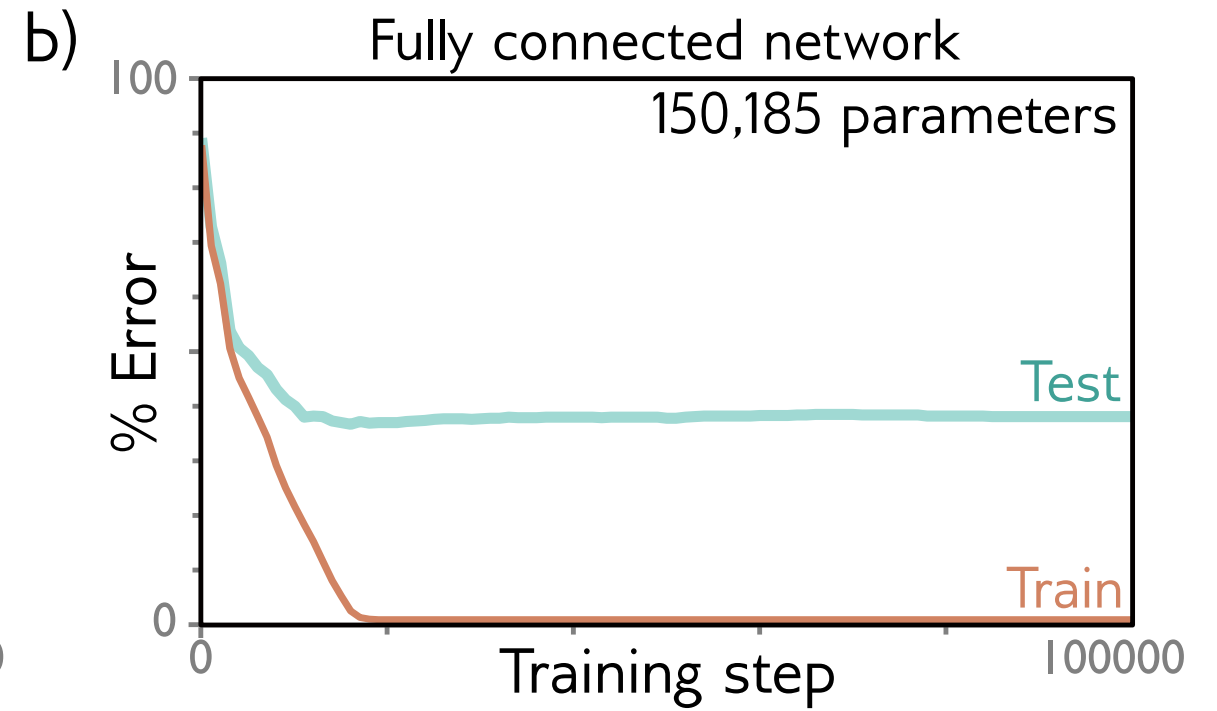
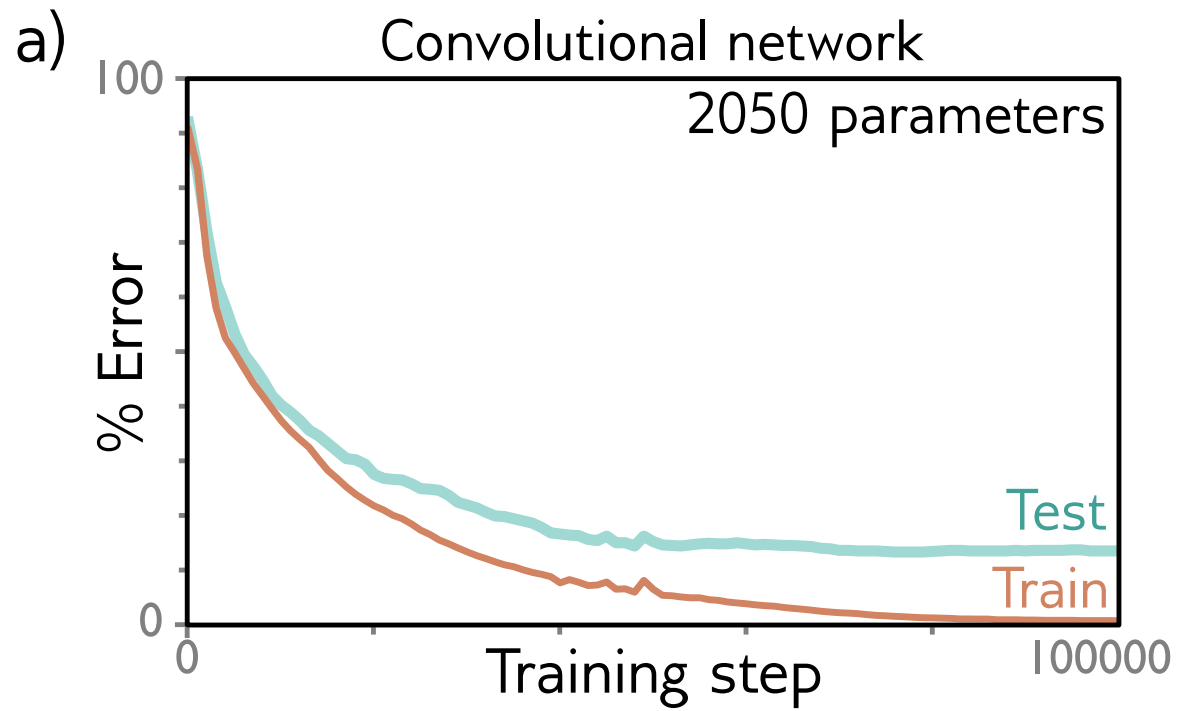
MNIST-1D convolutional network



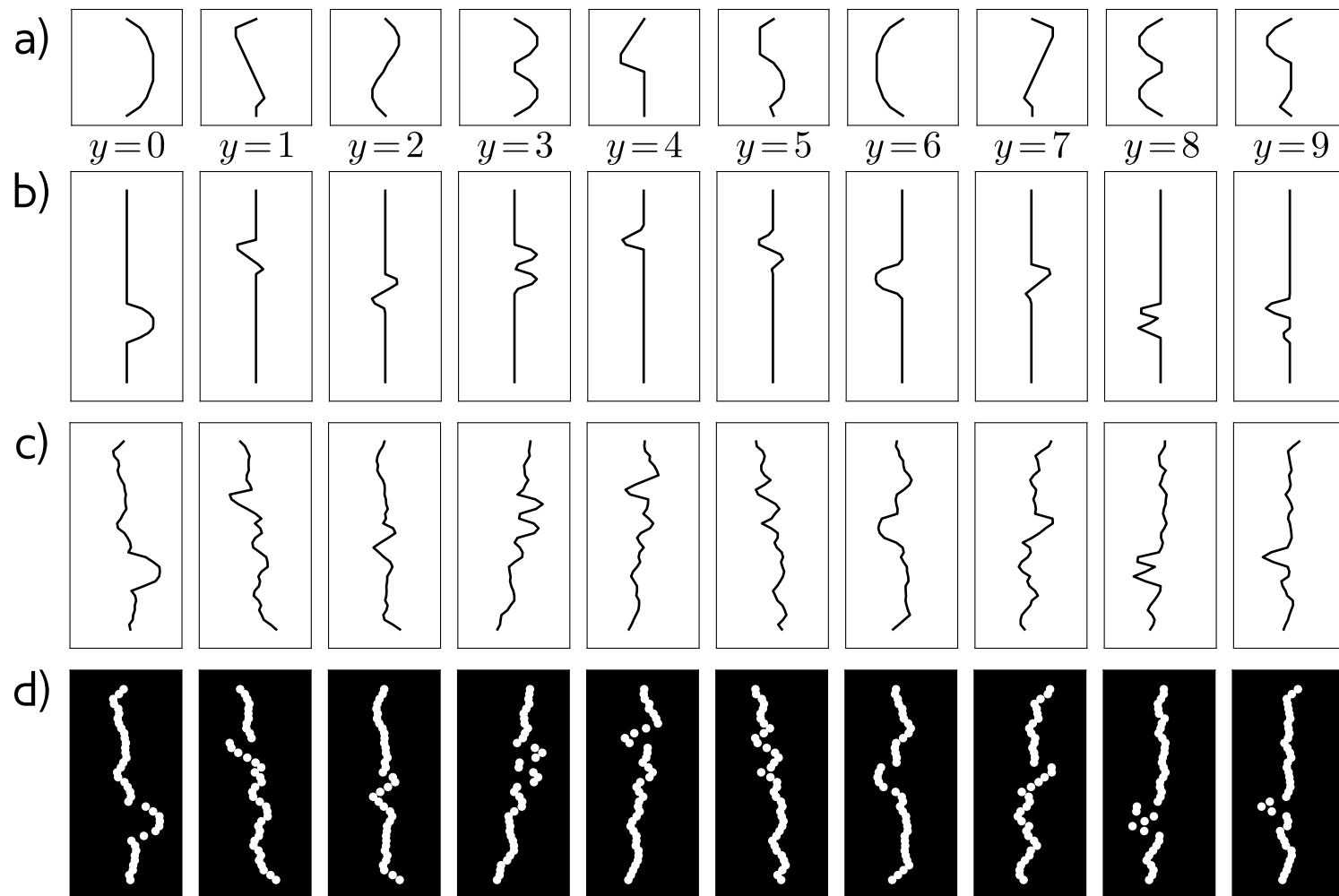
Fully connected network

- 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/output 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

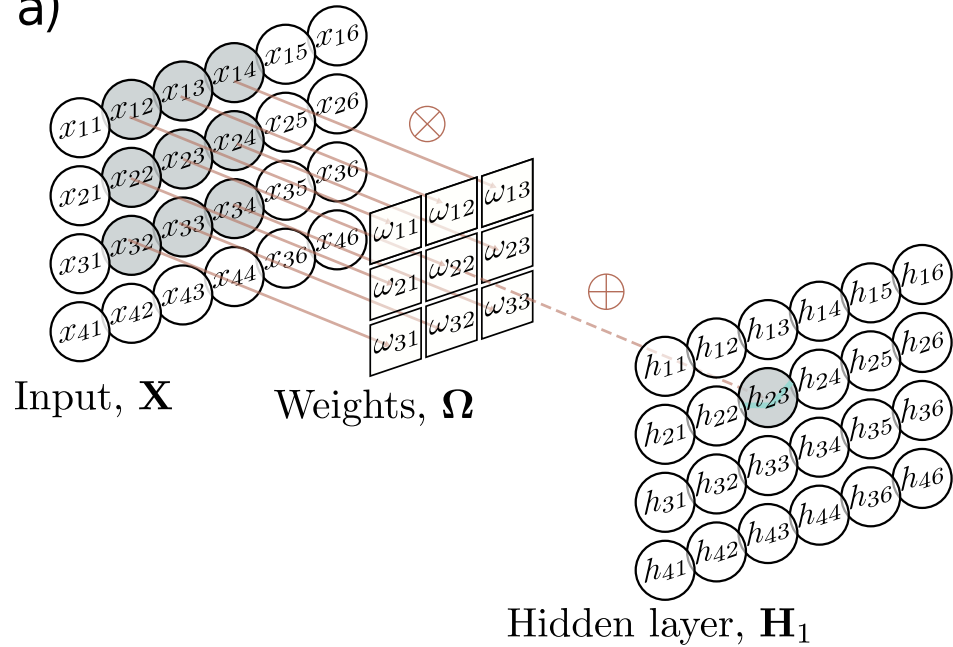
2D Convolution

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

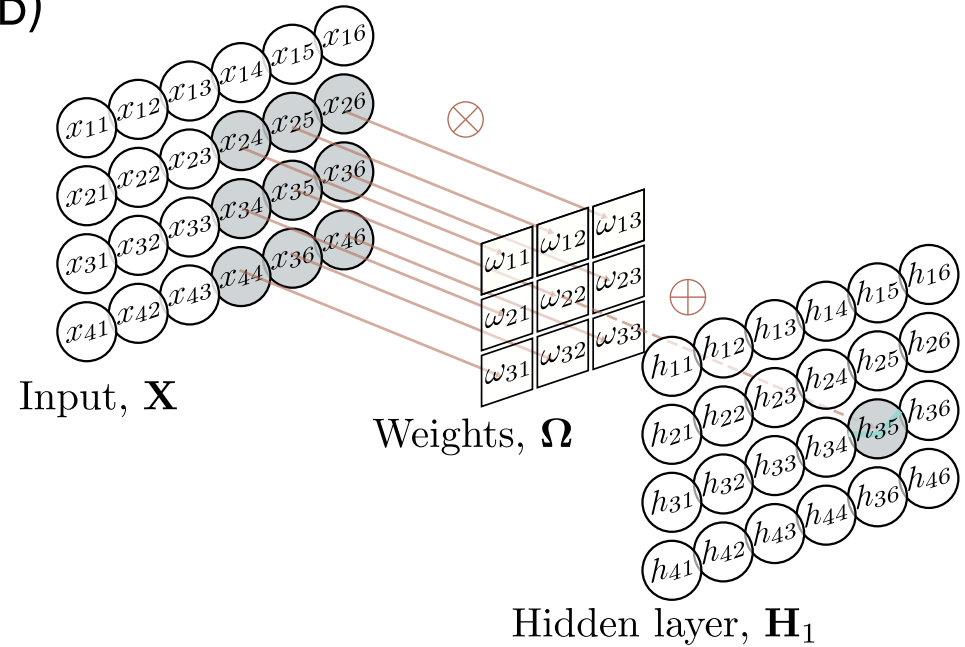
$$h_{i,j} = a \left[\beta + \sum_{m=1}^3 \sum_{n=1}^3 \omega_{m,n} x_{i+m-2, j+n-2} \right]$$

2D Convolution

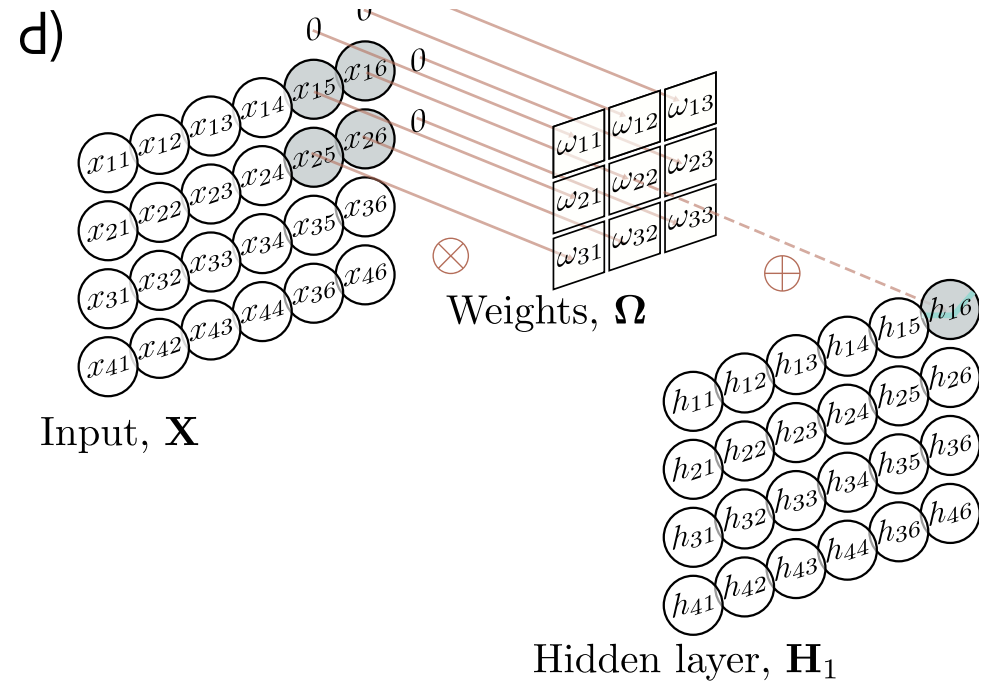
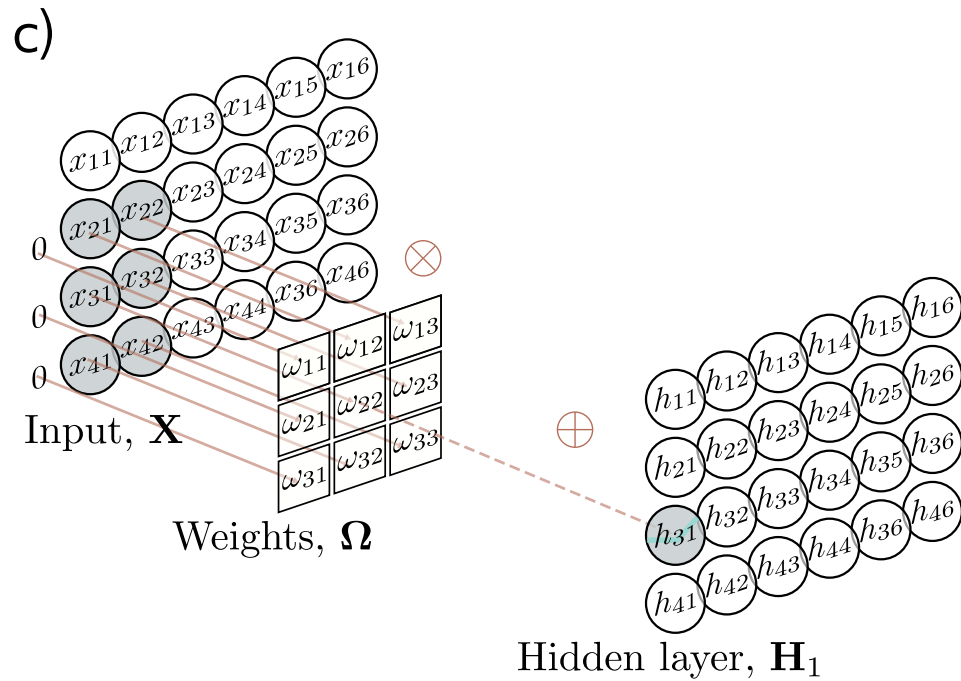
a)



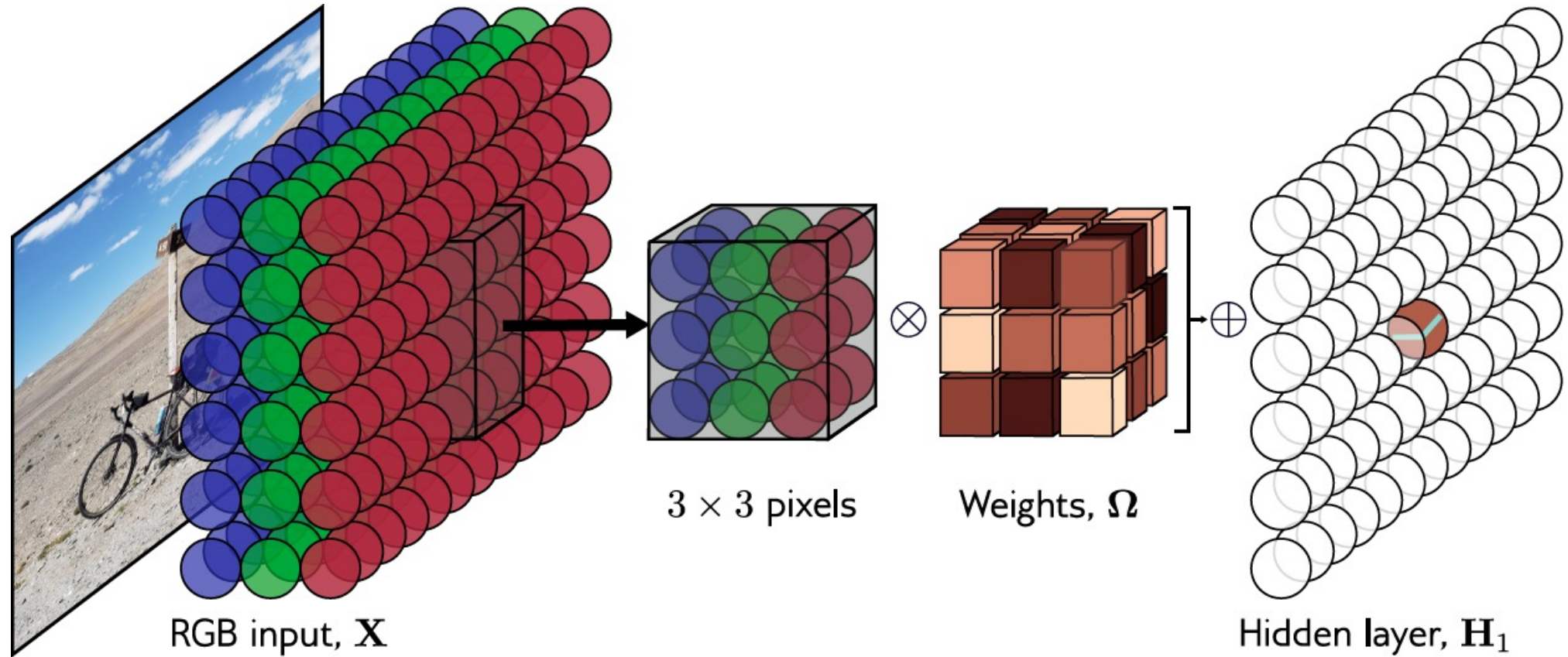
b)



2D Convolution



Channels in 2D convolution



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

How many parameters?

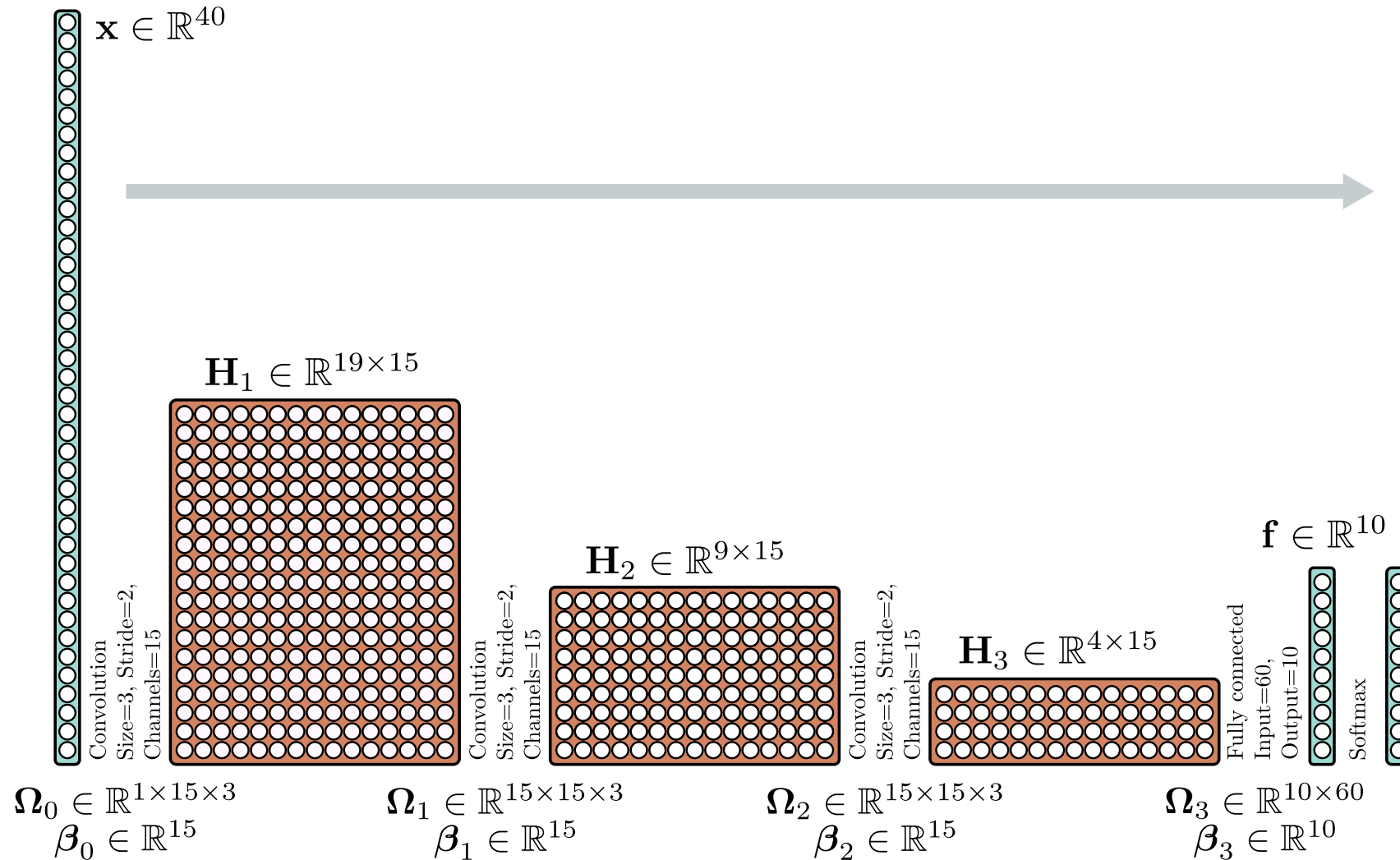
- If there are C_i input channels and kernel size $K \times K$

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

- If there are C_i input channels and C_o output channels

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

MNIST-1D convolutional network

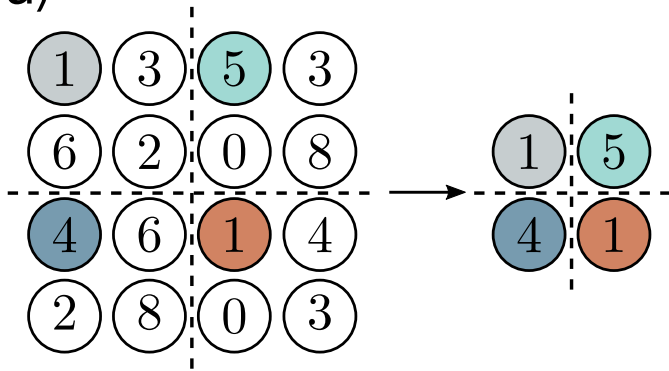


Convolution #2

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

Downsampling

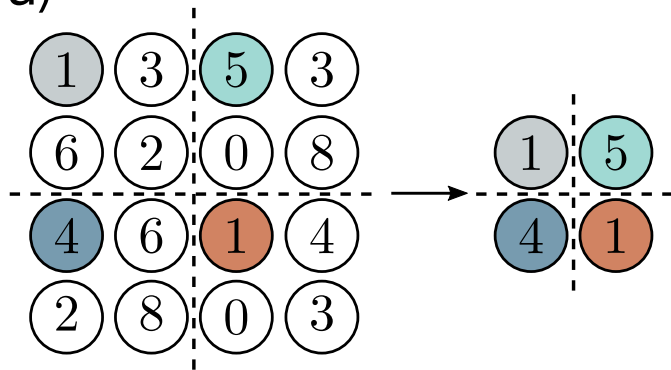
a)



Sample every other
position (equivalent to
stride two)

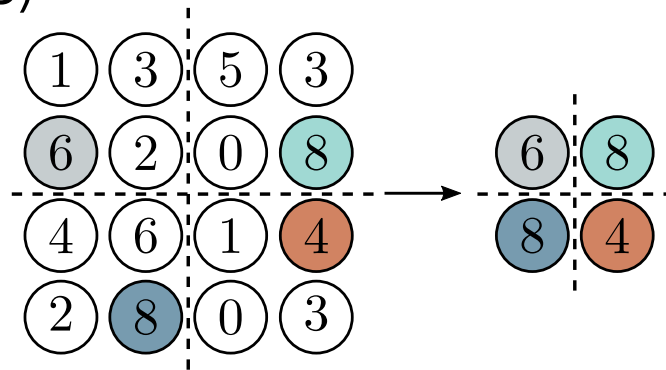
Downsampling

a)



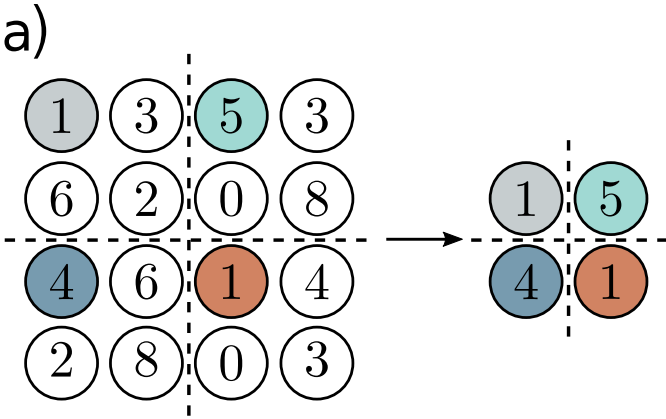
Sample every other
position (equivalent to
stride two)

b)

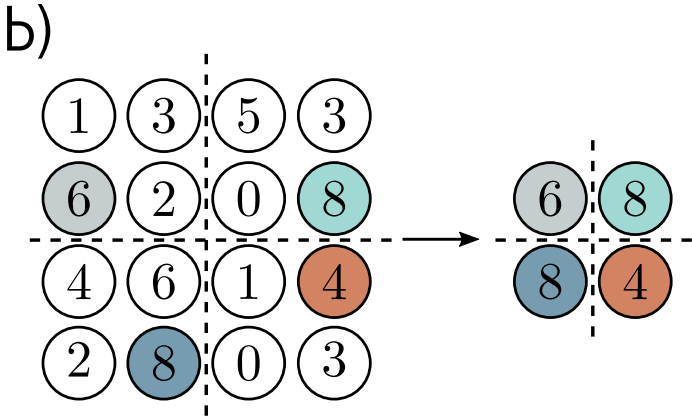


Max pooling
(partial invariance to
translation)

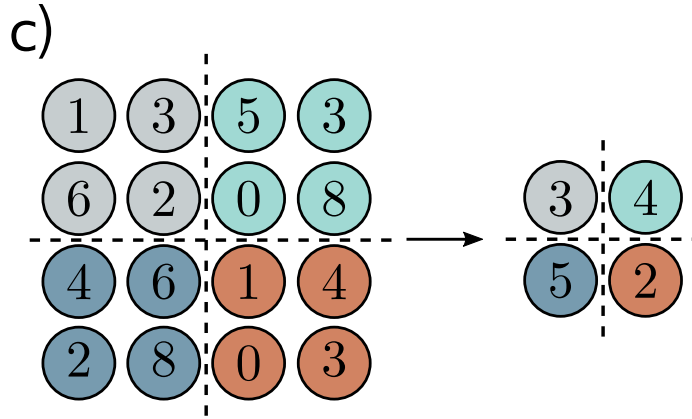
Downsampling



Sample every other position (equivalent to stride two)



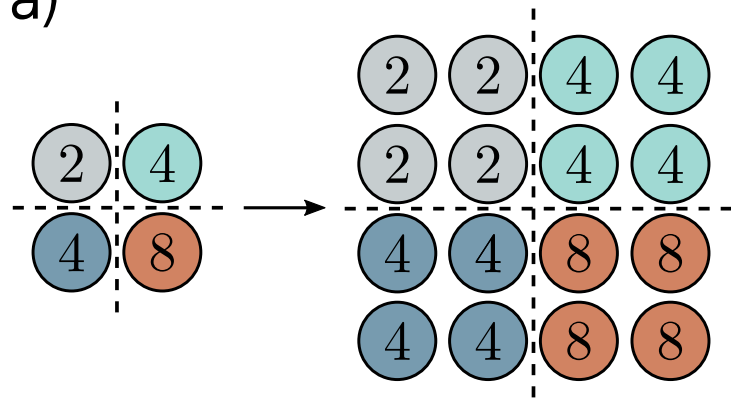
Max pooling (partial invariance to translation)



Mean pooling

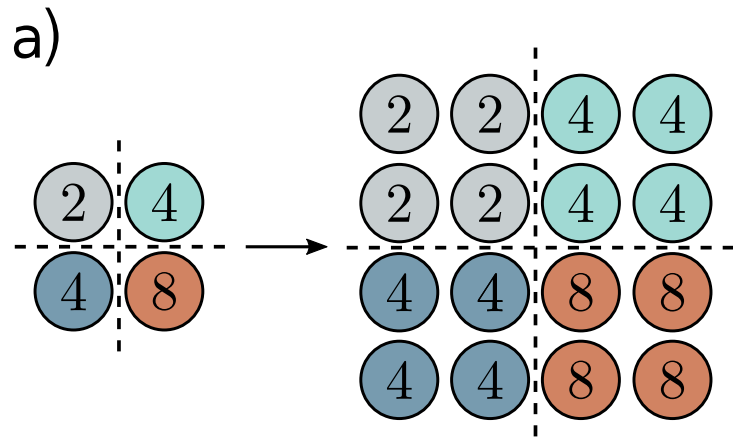
Upsampling

a)

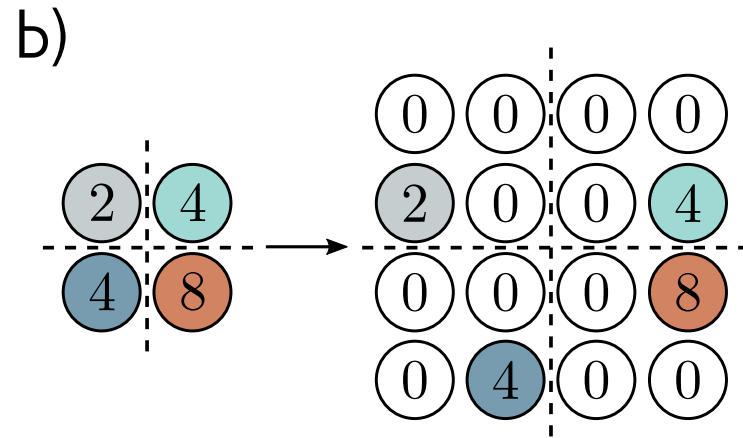


Duplicate

Upsampling

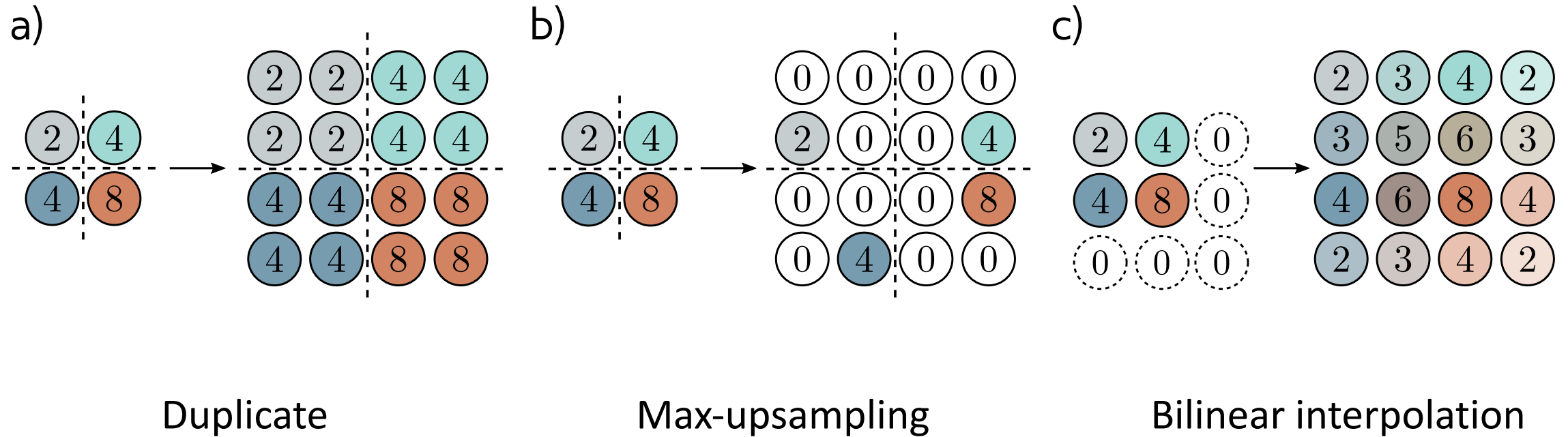


Duplicate

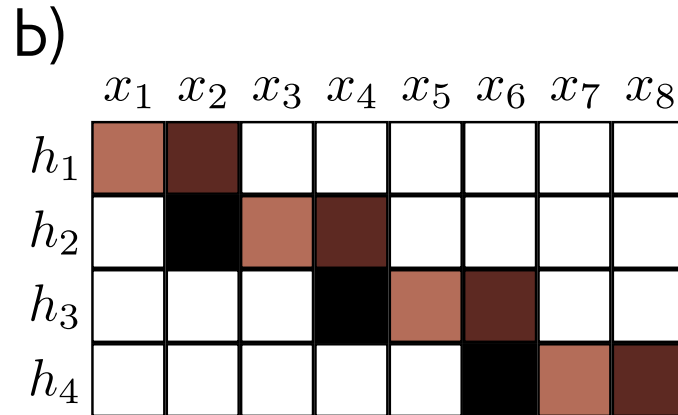
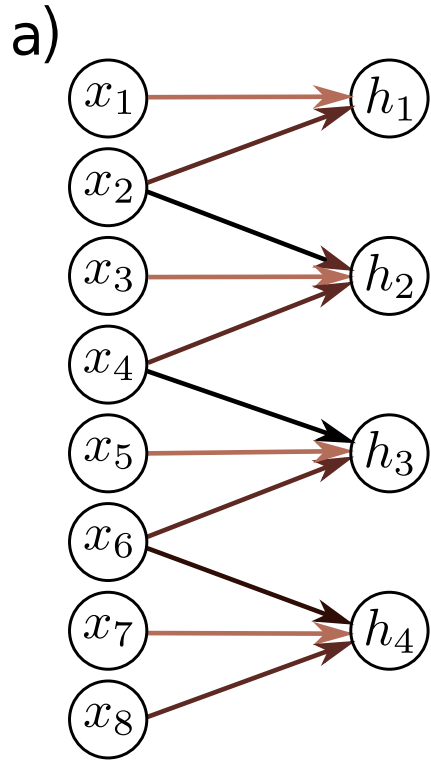


Max-upsampling

Upsampling

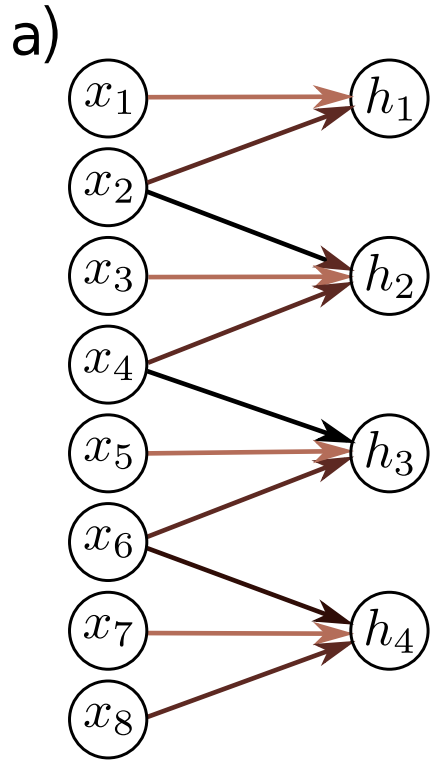


Transposed convolutions

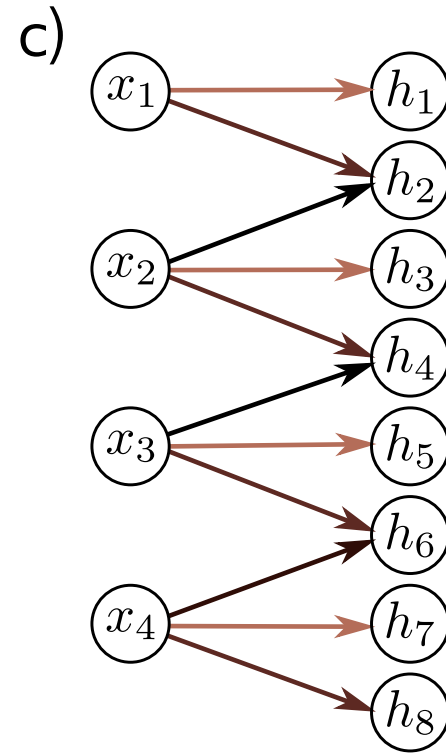
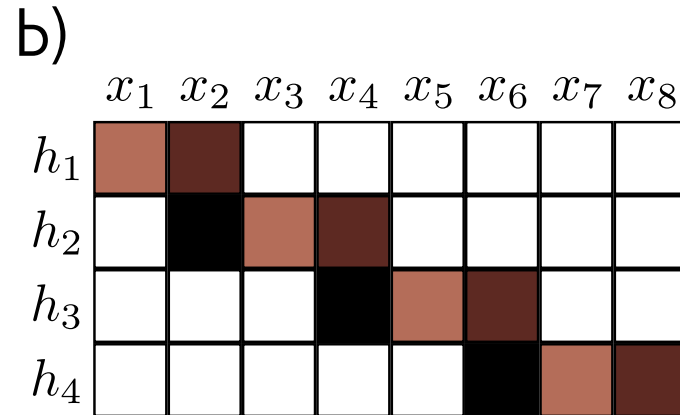


Kernel size 3, Stride 2 convolution

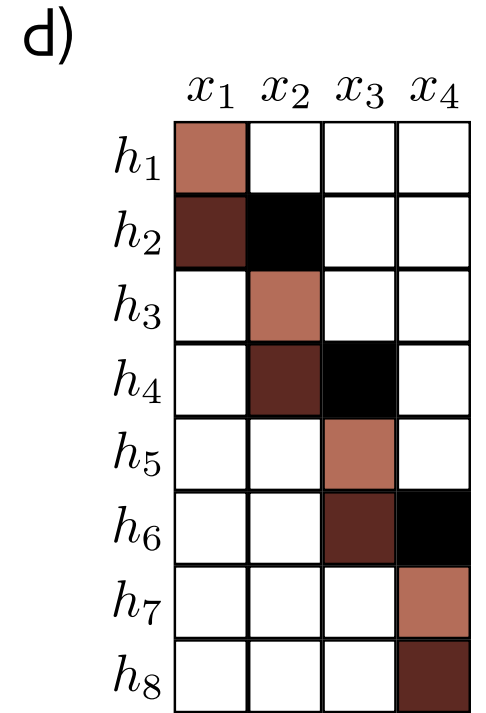
Transposed convolutions



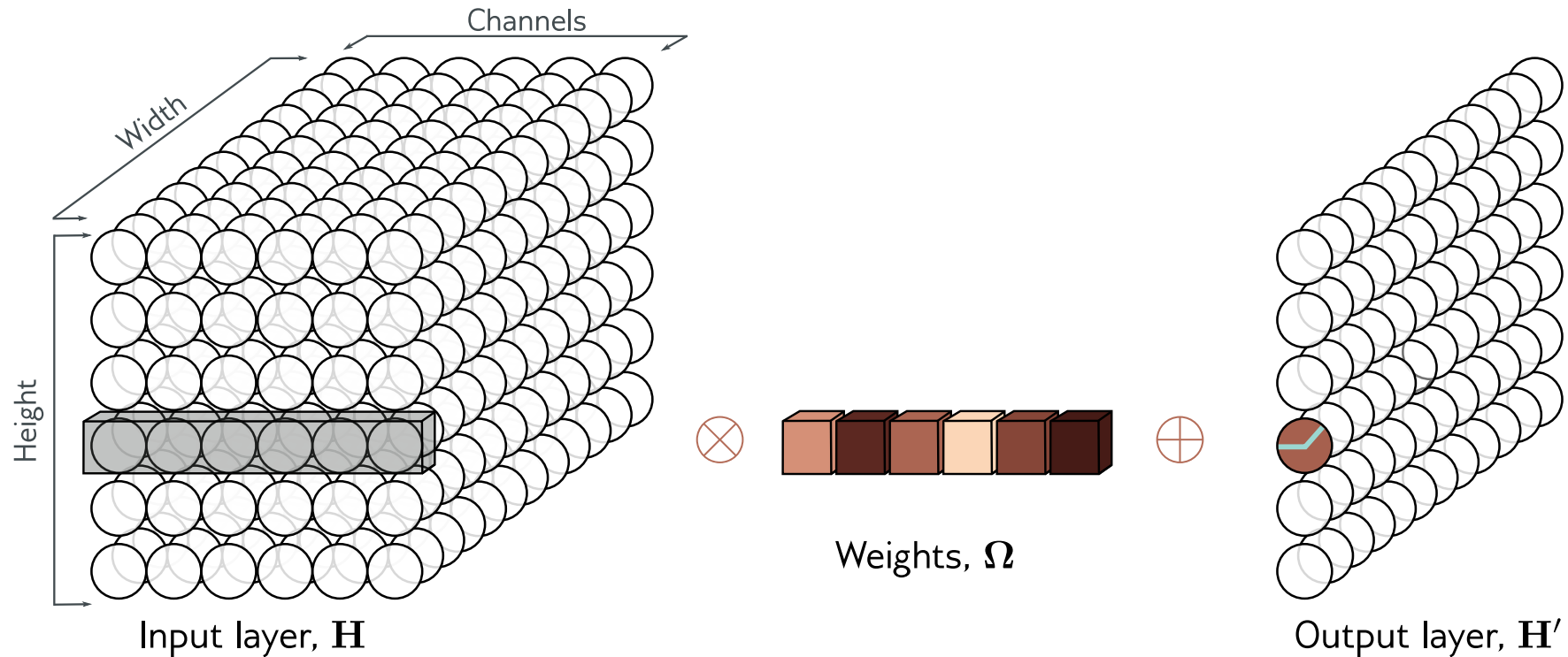
Kernel size 3, Stride 2 convolution



Transposed convolution



1x1 convolution

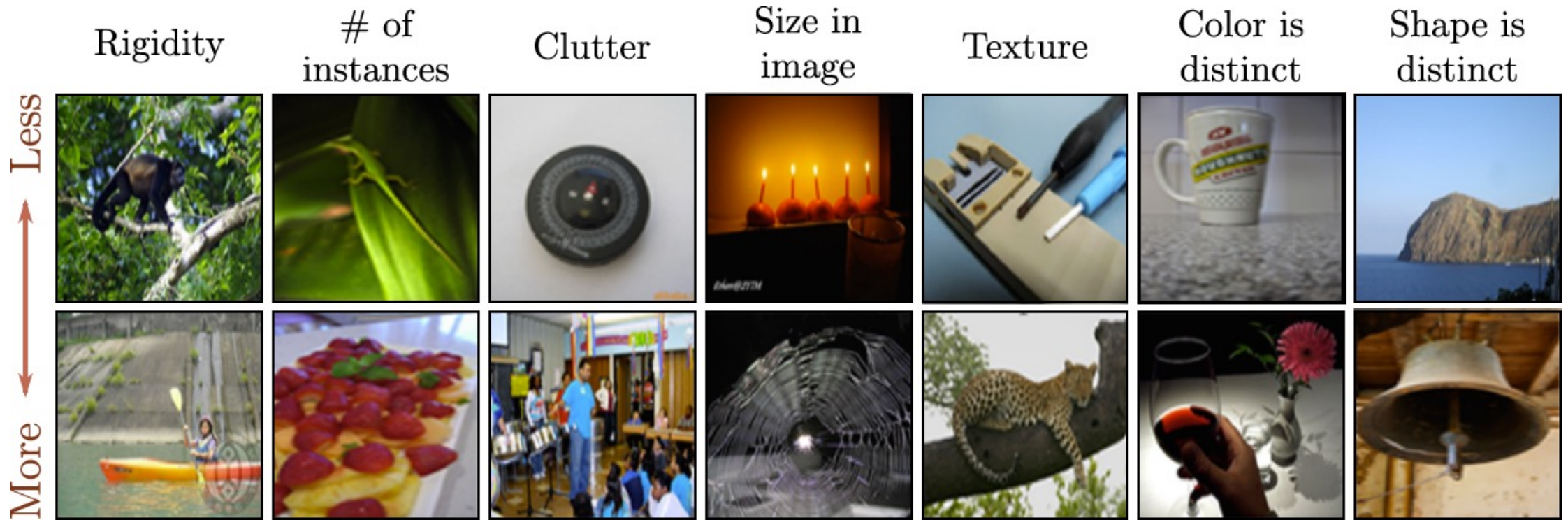


- Mixes channels
- Can change number of channels
- Equivalent to running same fully connected network at each position

Convolution #2

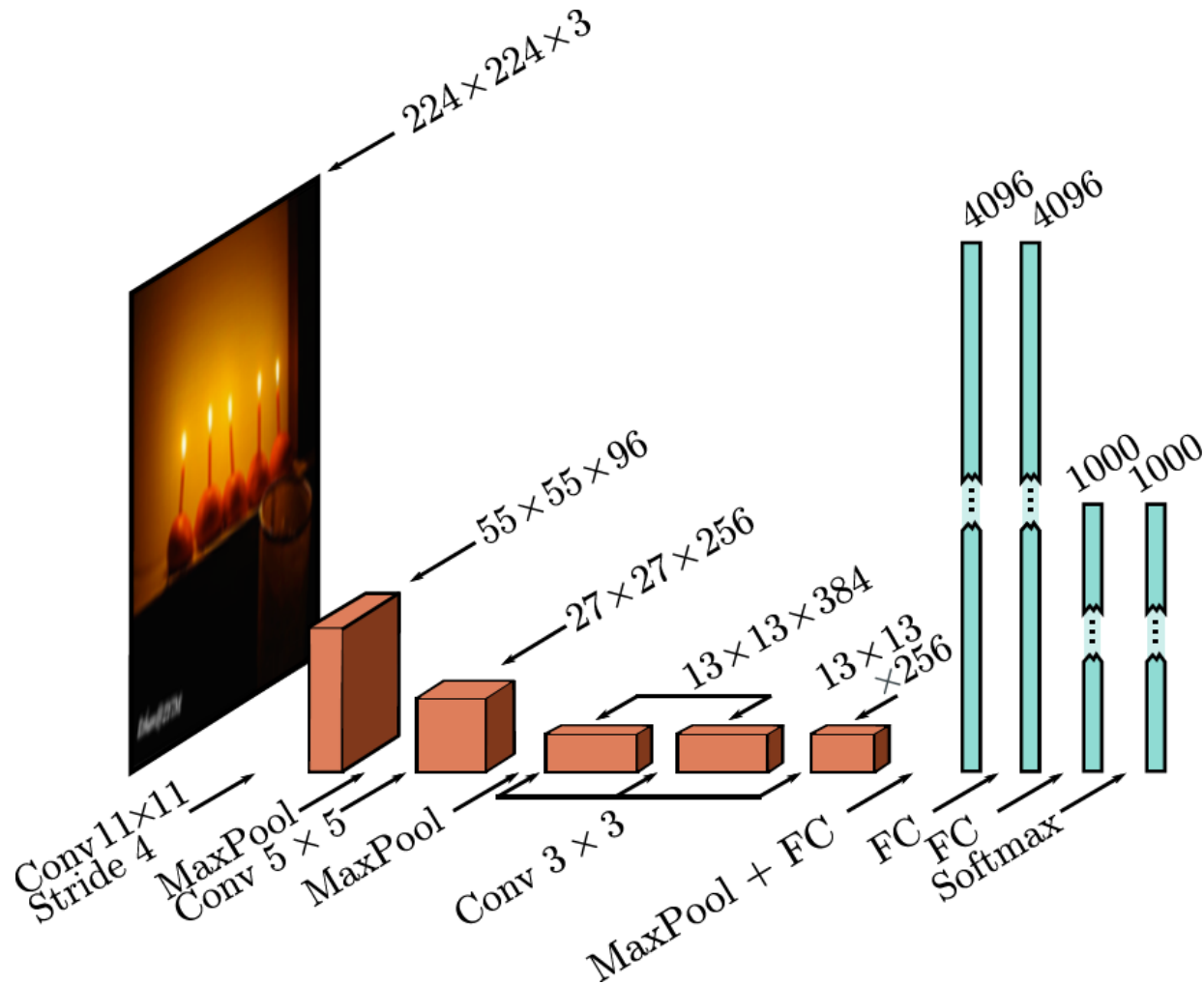
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

ImageNet database



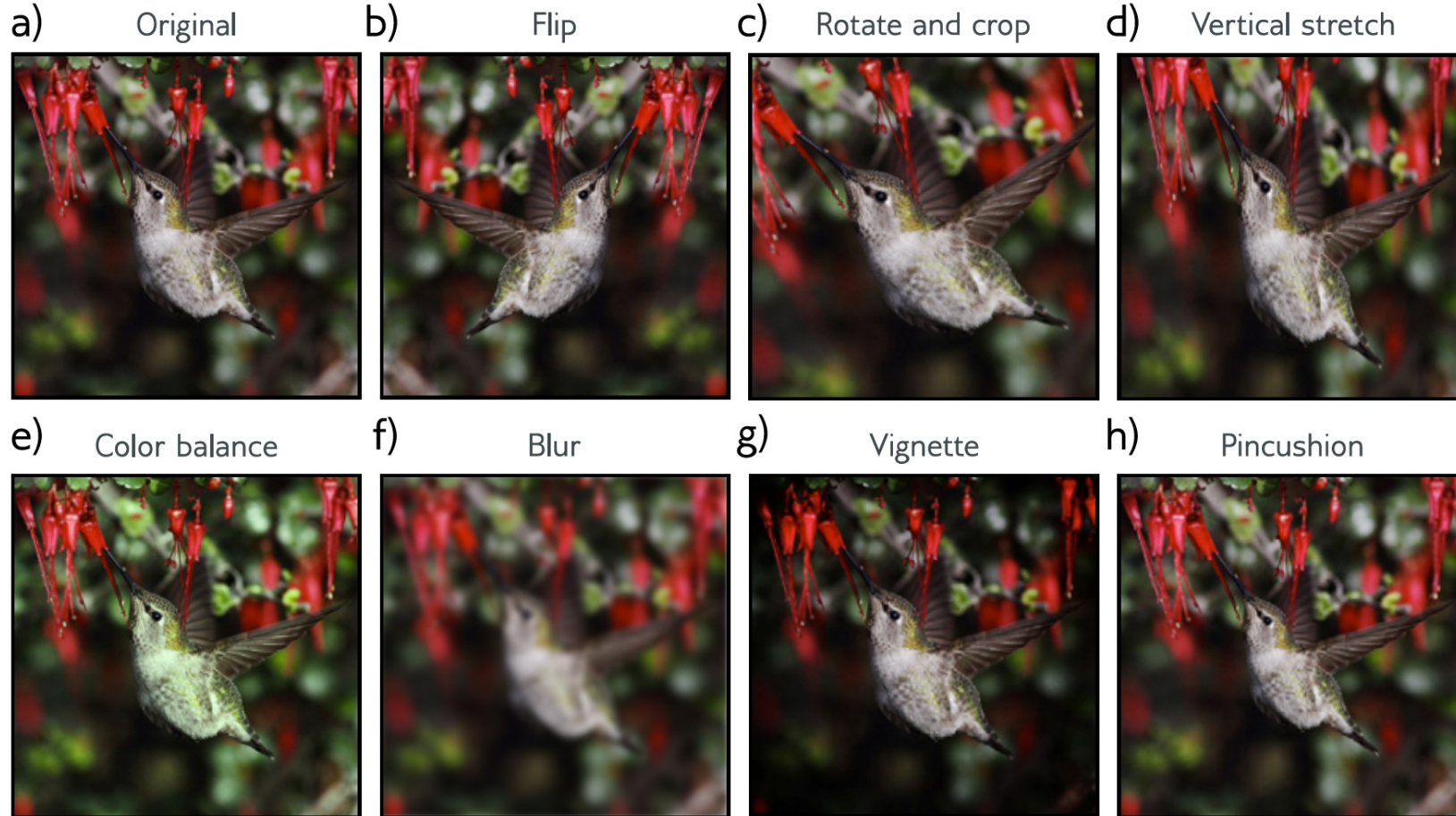
- 224 x 224 images
- 1,281,167 training images, 50,000 validation images, and 100,000 test images
- 1000 classes

AlexNet (2012)



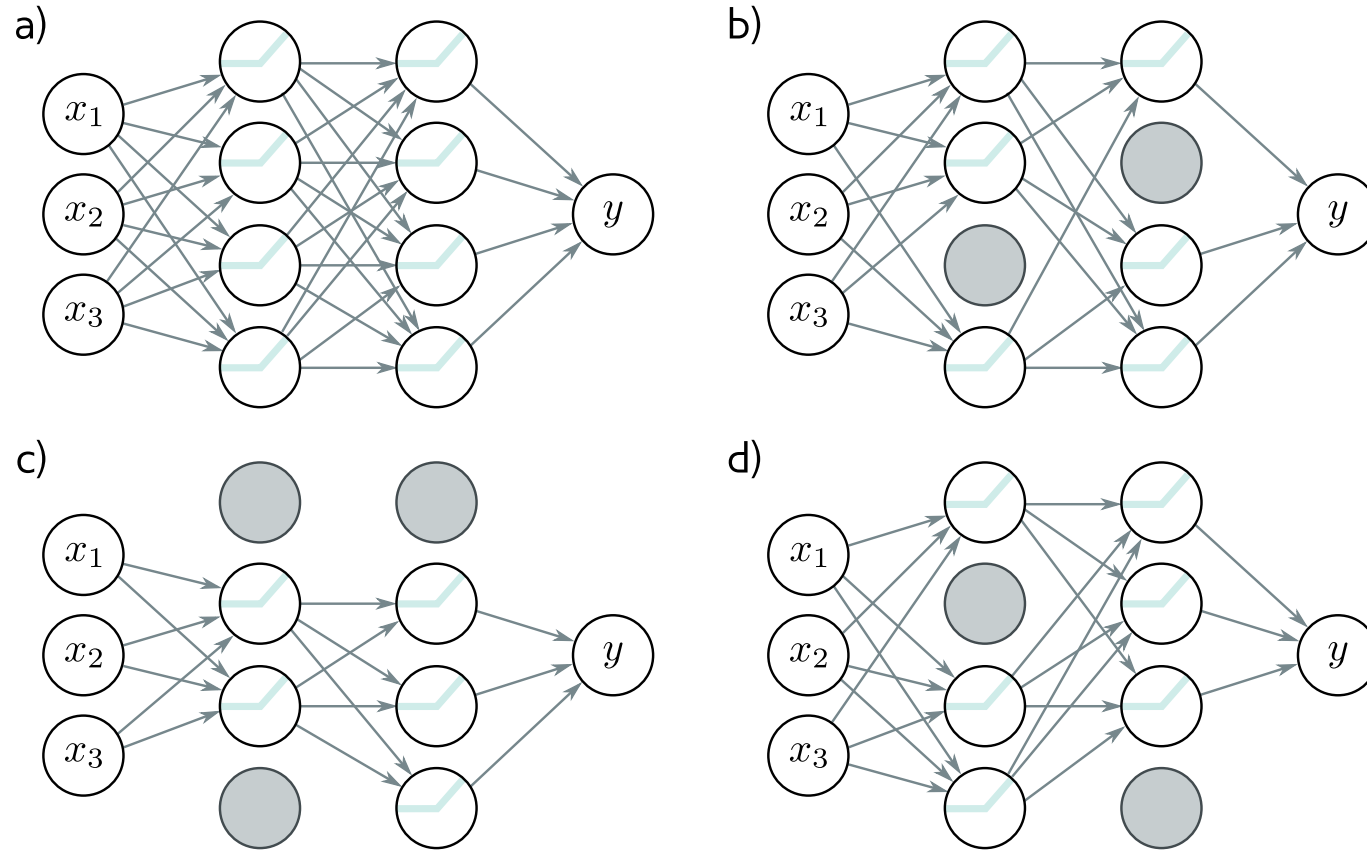
Almost all the 60 million parameters are in fully connected layers

Data augmentation



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

Dropout

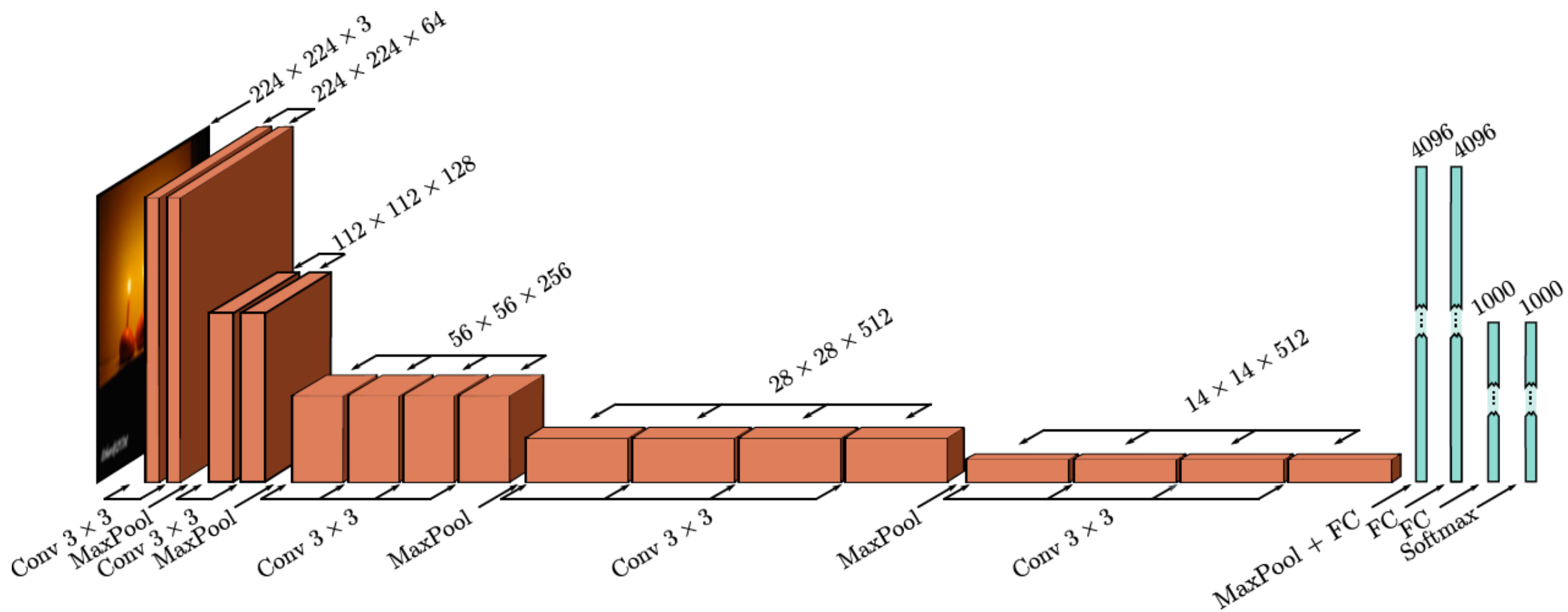


- Dropout was applied in the fully connected layers

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.

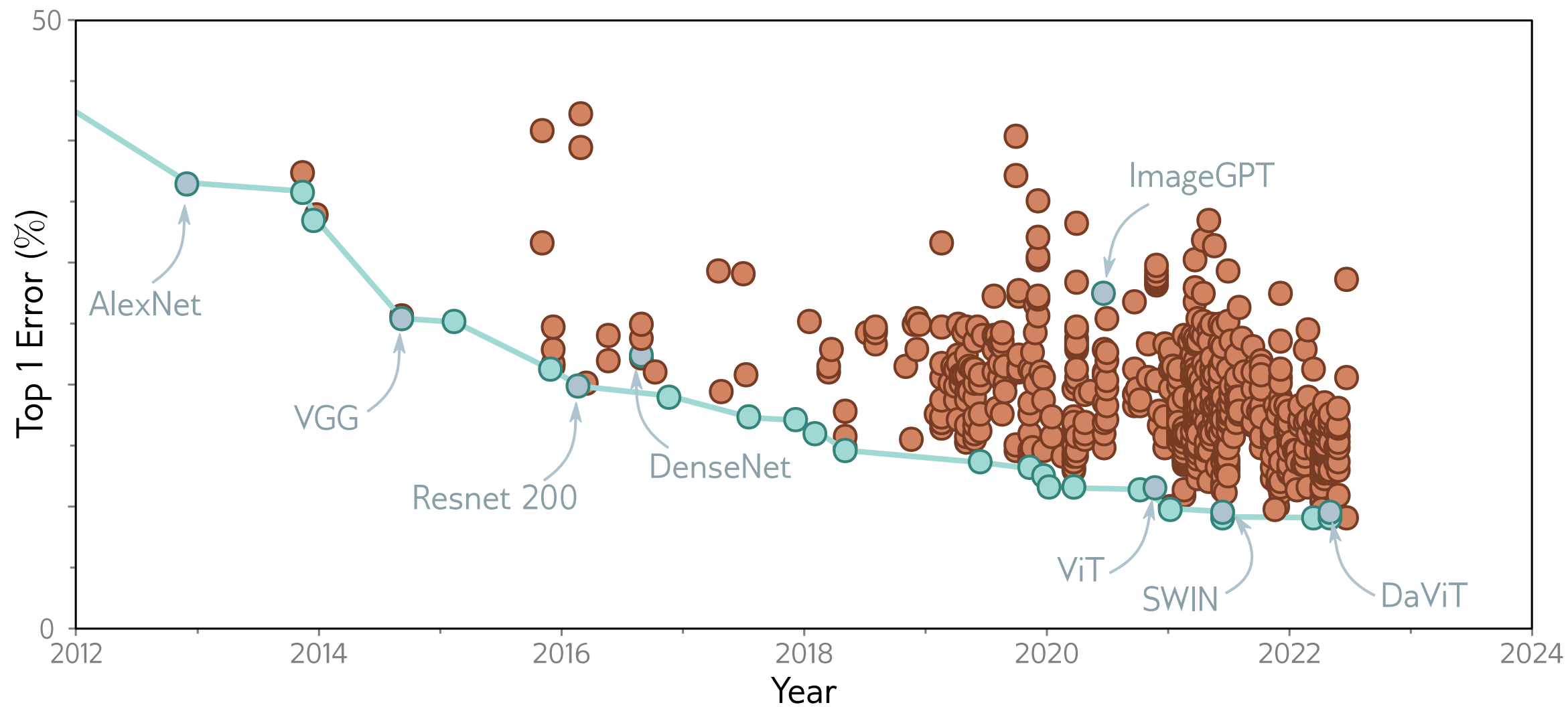
VGG (2015)



Details

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

ImageNet History

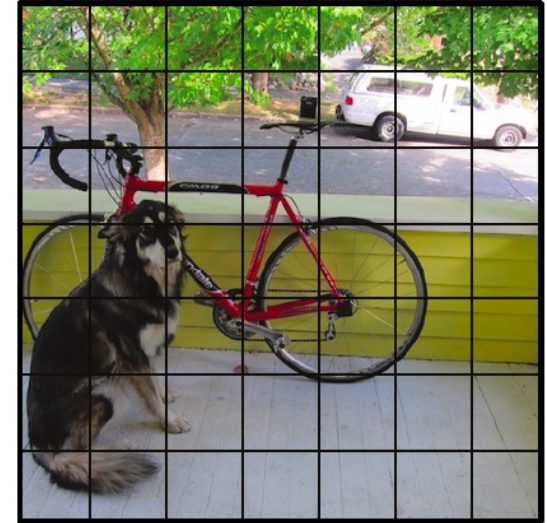


Convolution #2

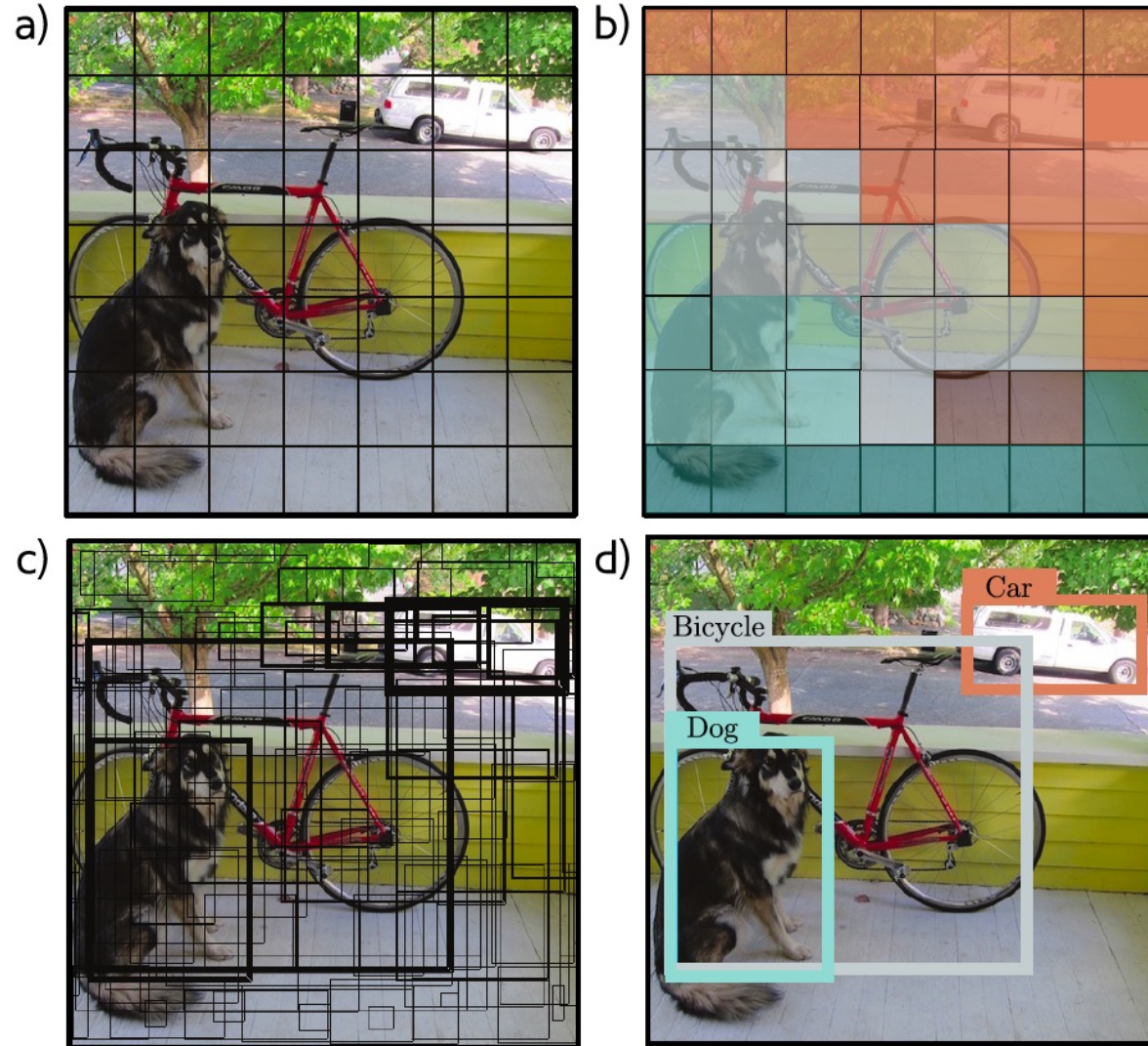
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

You Only Look Once (YOLO)

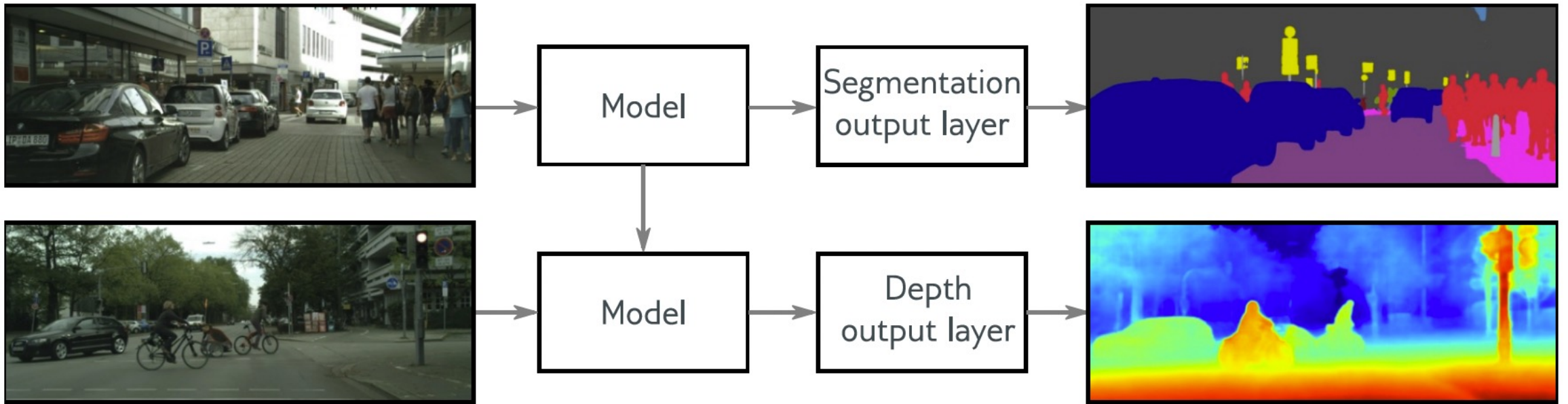
- Network similar to VGG (448x448 input)
- 7×7 grid of locations
- Predict class at each location
- Predict 2 bounding boxes at each location
 - Five parameters –x,y, height, width, and confidence
- Momentum, weight decay, dropout, and data augmentation
- Heuristic at the end to threshold and decide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



Object detection (YOLO)

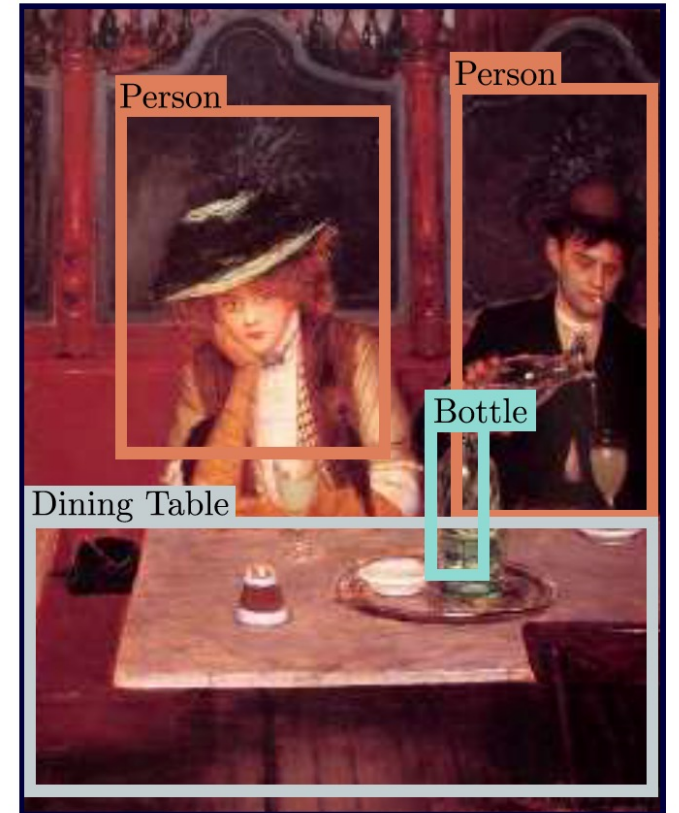
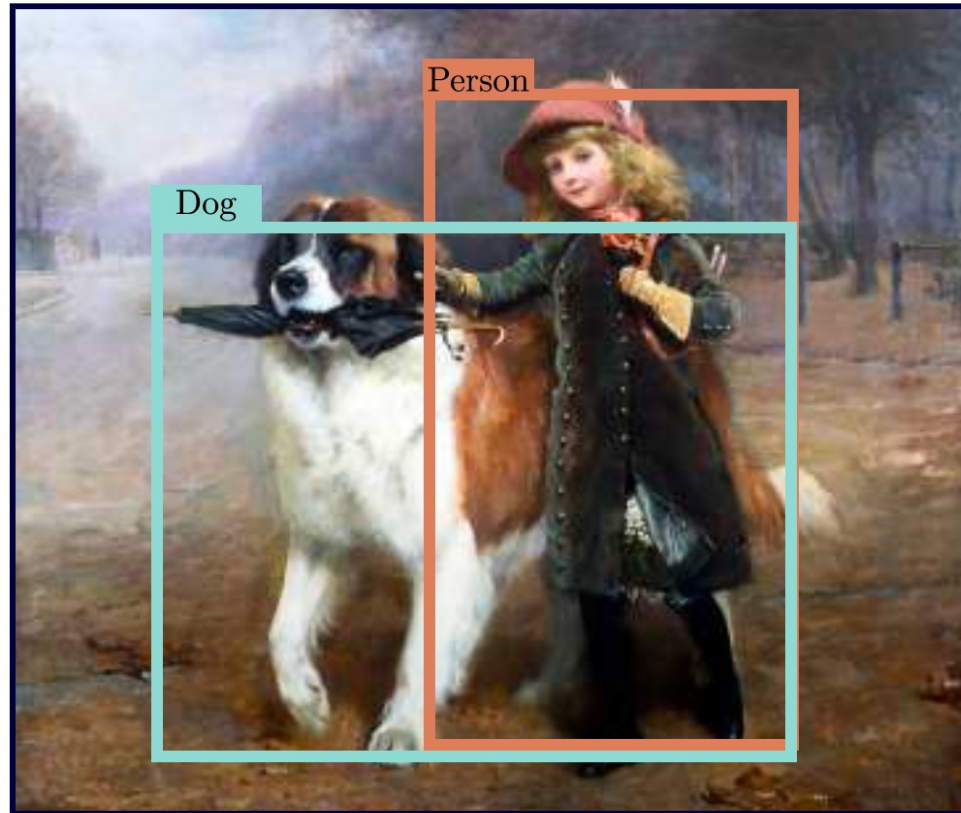
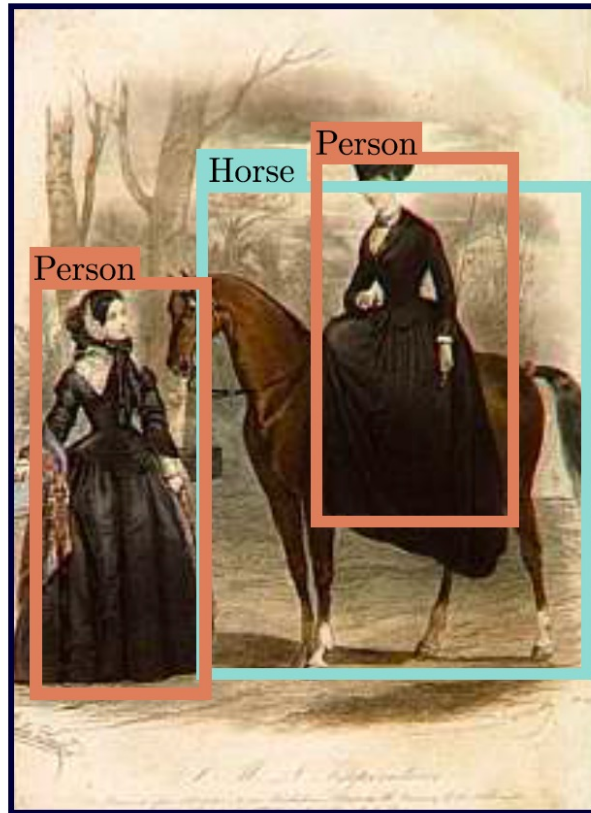


Transfer learning



Transfer learning from ImageNet classification

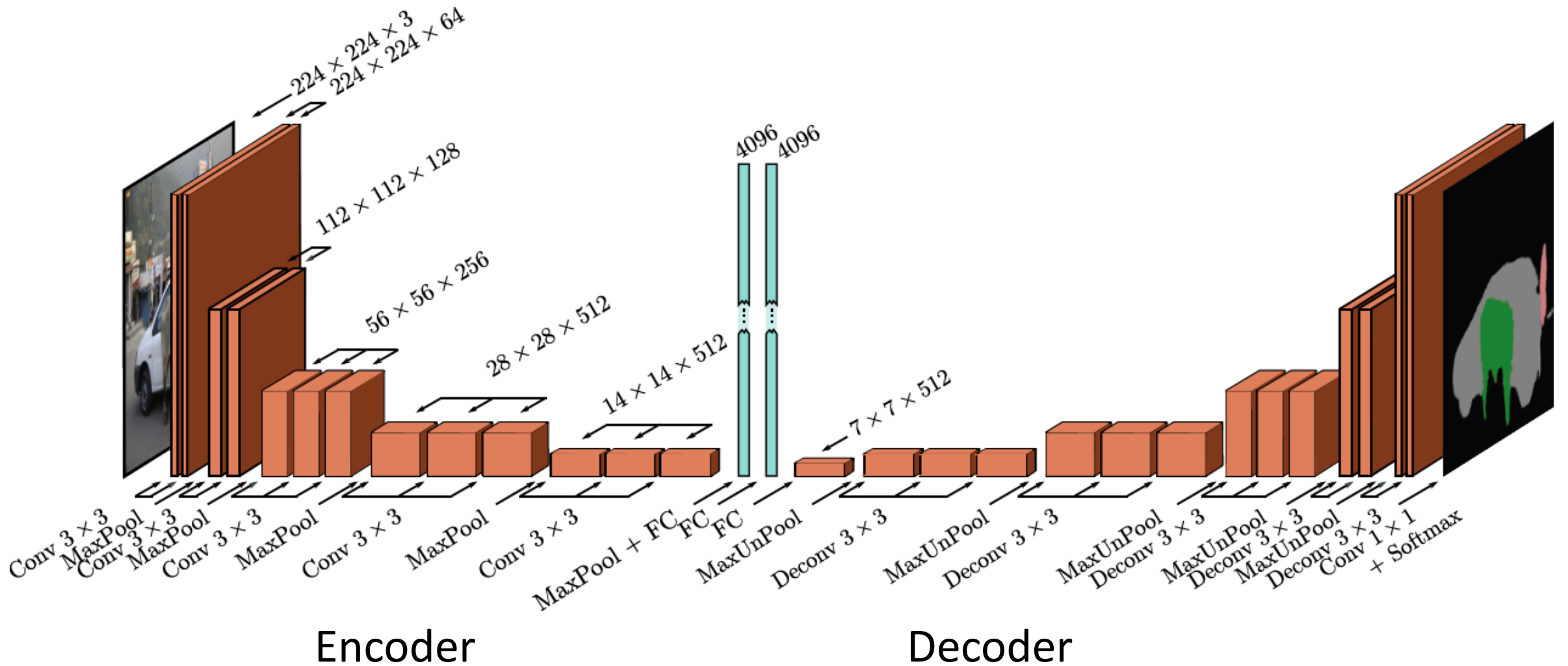
Results



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)

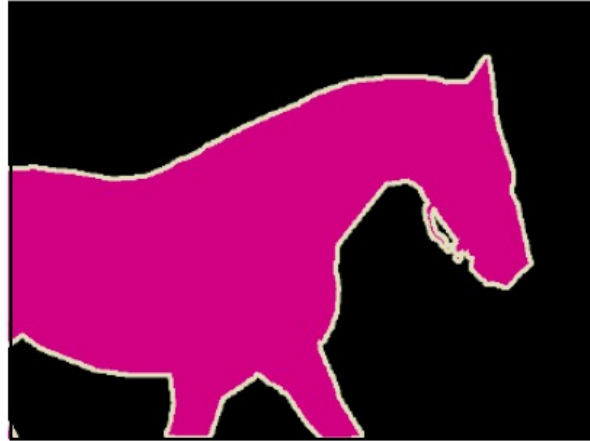


Semantic segmentation results

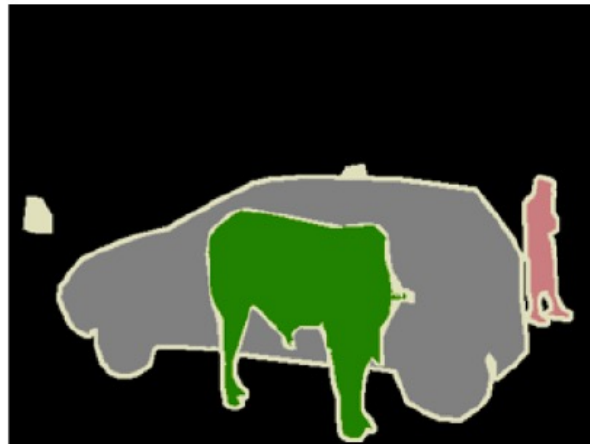
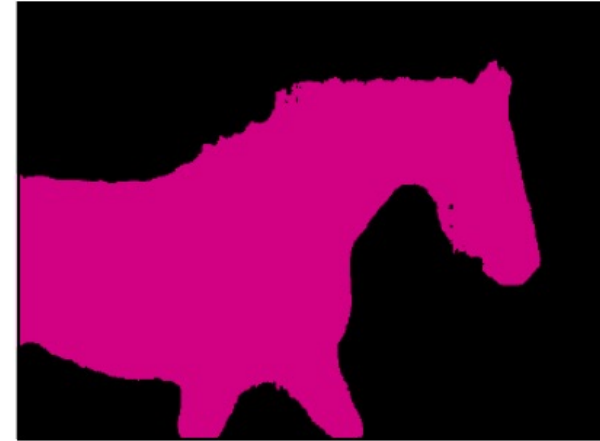
Input



Ground truth



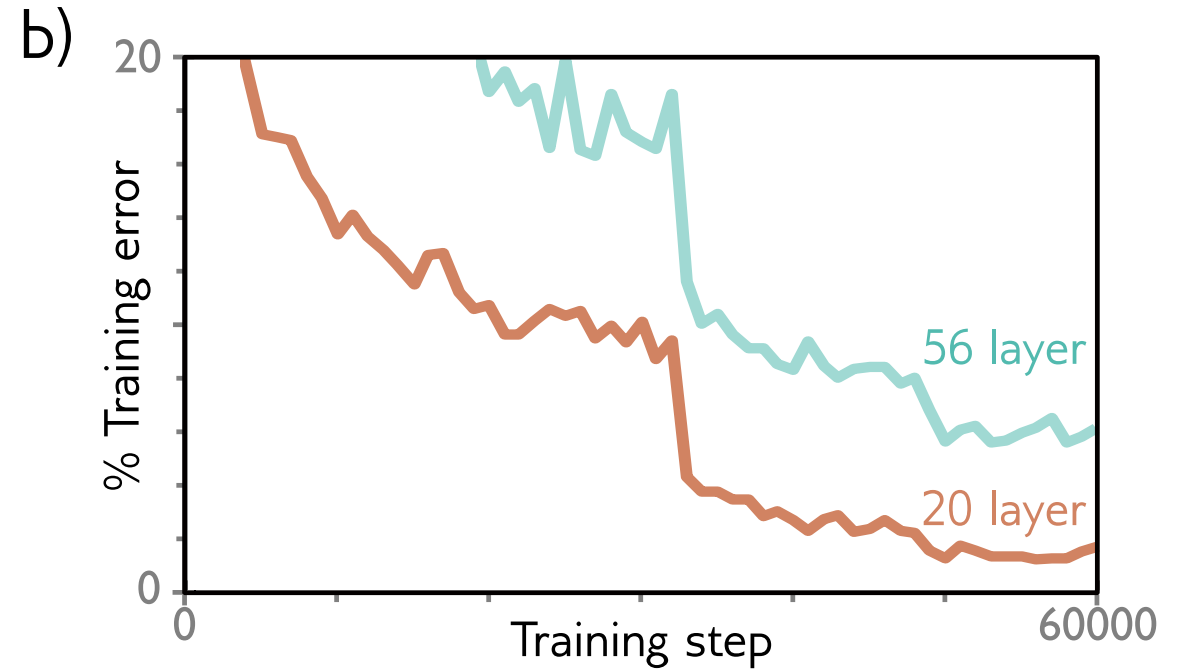
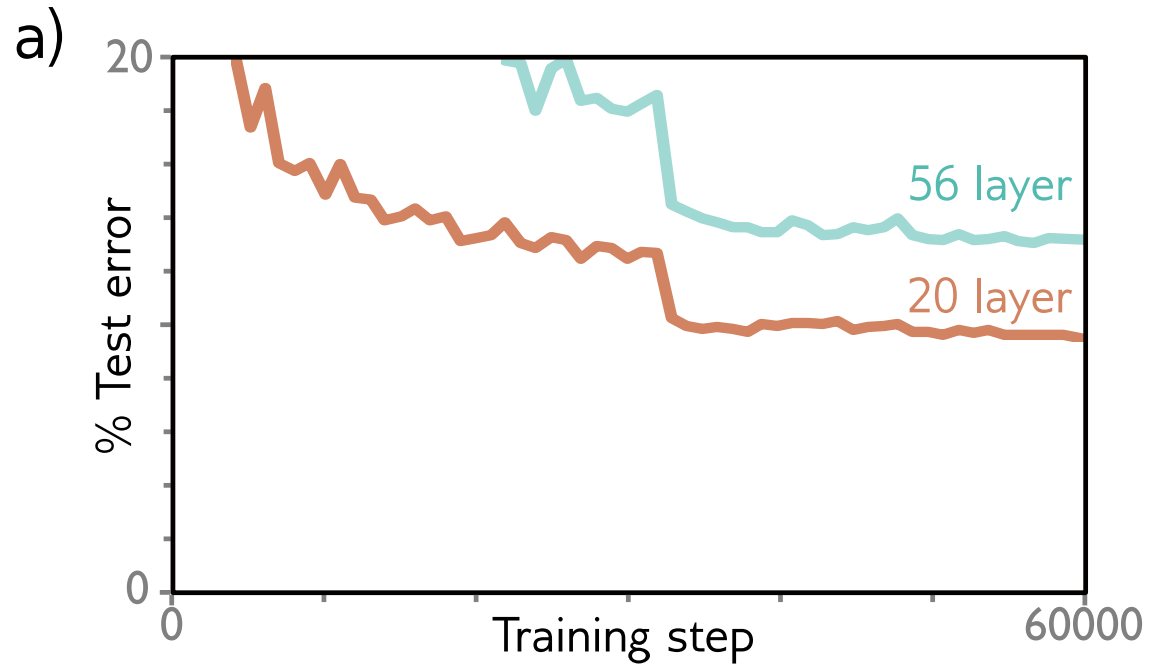
Result



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



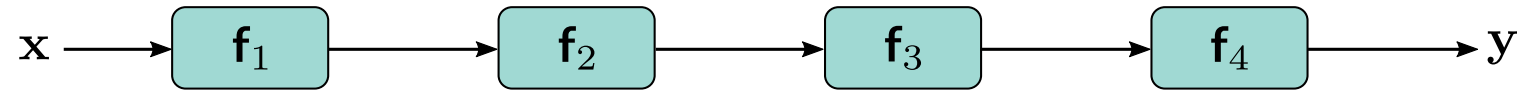
Regular network:

$$\mathbf{h}_1 = \mathbf{f}_1[\mathbf{x}, \phi_1]$$

$$\mathbf{h}_2 = \mathbf{f}_2[\mathbf{h}_1, \phi_2]$$

$$\mathbf{h}_3 = \mathbf{f}_3[\mathbf{h}_2, \phi_3]$$

$$\mathbf{y} = \mathbf{f}_4[\mathbf{h}_3, \phi_4]$$



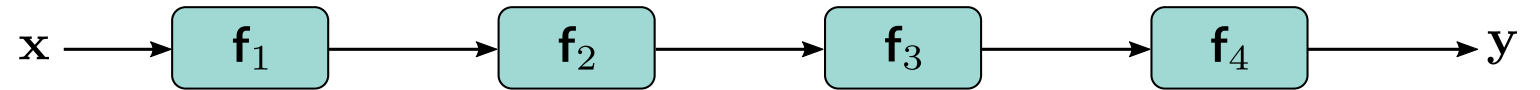
Regular network:

$$\mathbf{h}_1 = \mathbf{f}_1[\mathbf{x}, \phi_1]$$

$$\mathbf{h}_2 = \mathbf{f}_2[\mathbf{h}_1, \phi_2]$$

$$\mathbf{h}_3 = \mathbf{f}_3[\mathbf{h}_2, \phi_3]$$

$$\mathbf{y} = \mathbf{f}_4[\mathbf{h}_3, \phi_4]$$



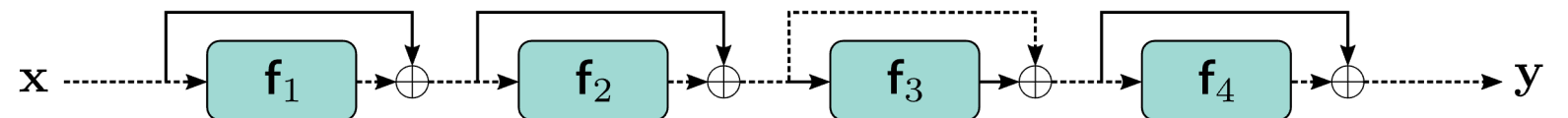
Residual network (2016):

$$\mathbf{h}_1 = \mathbf{x} + \mathbf{f}_1[\mathbf{x}, \phi_1]$$

$$\mathbf{h}_2 = \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \phi_2]$$

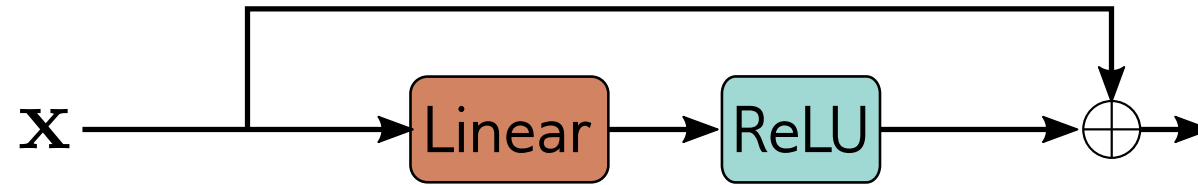
$$\mathbf{h}_3 = \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \phi_3]$$

$$\mathbf{y} = \mathbf{h}_3 + \mathbf{f}_4[\mathbf{h}_3, \phi_4]$$

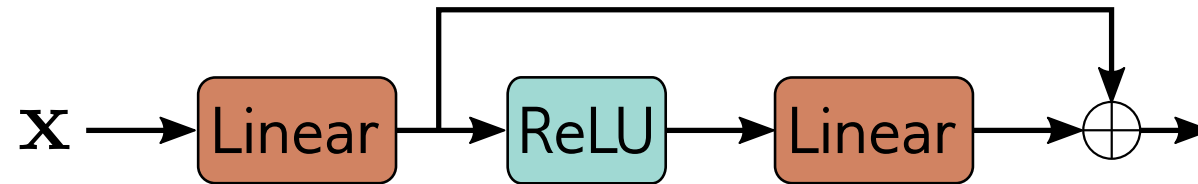


Order of operations is important

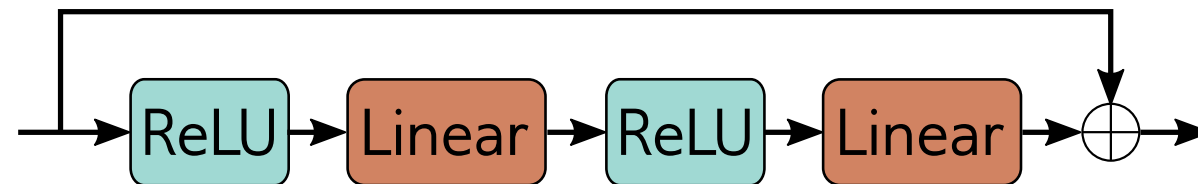
a)



b)



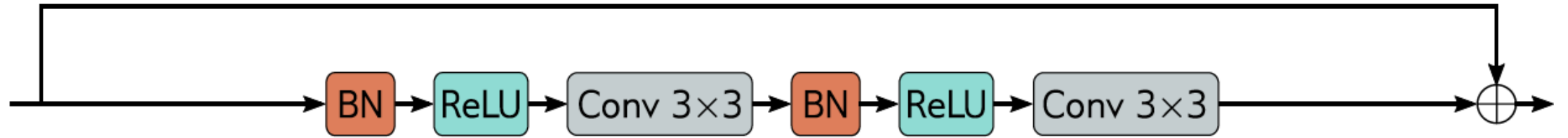
c)



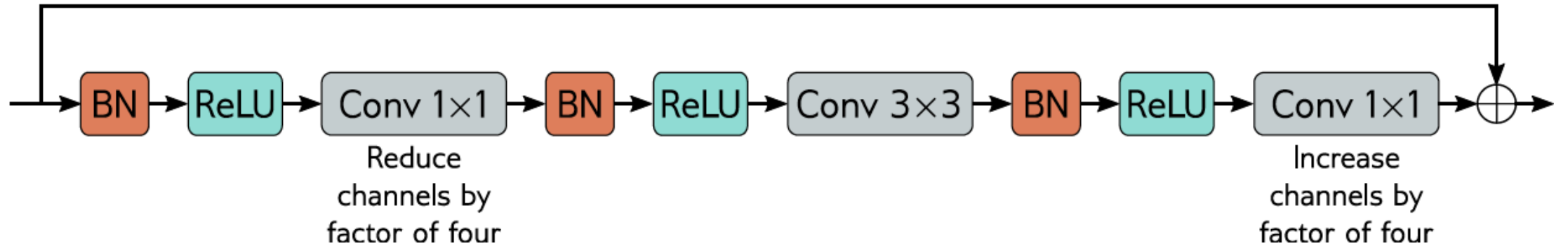
+ Batch Norm

Resnet Block

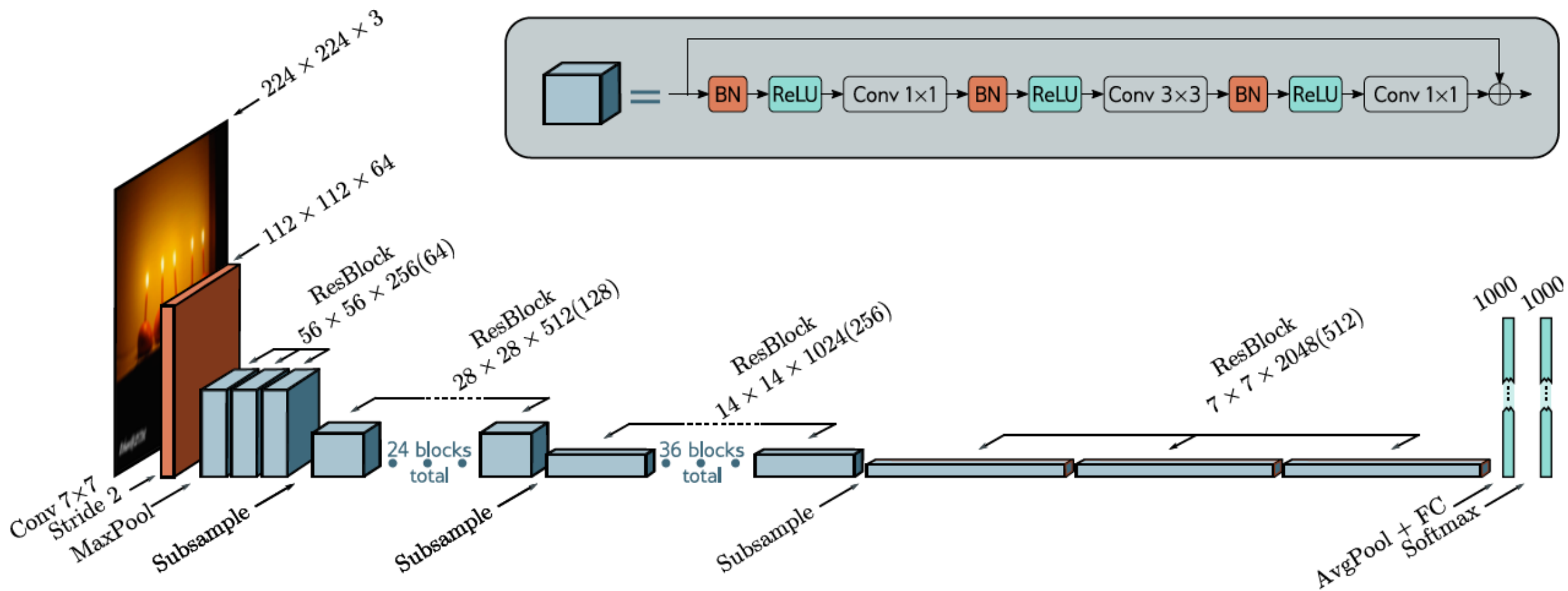
a)



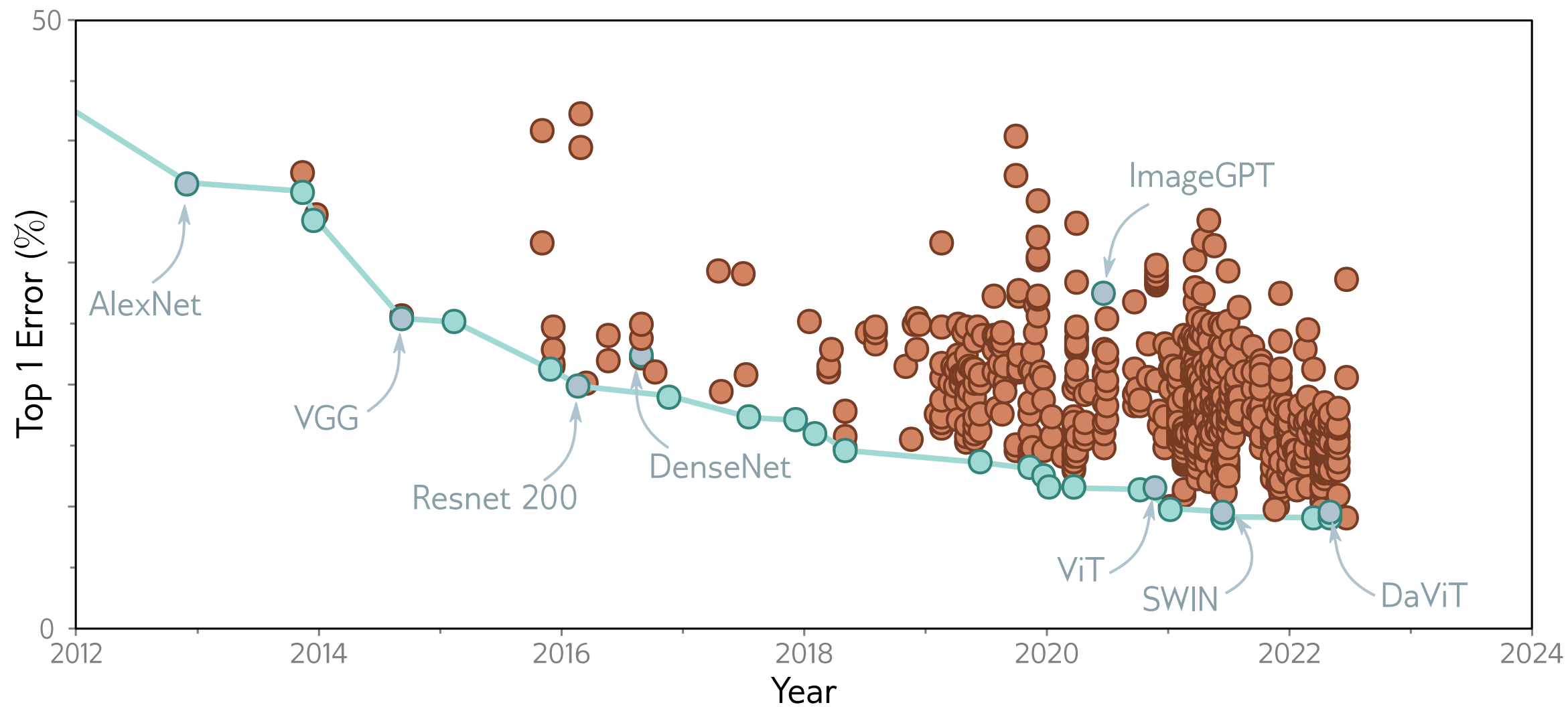
b)



Resnet 200 (2016)



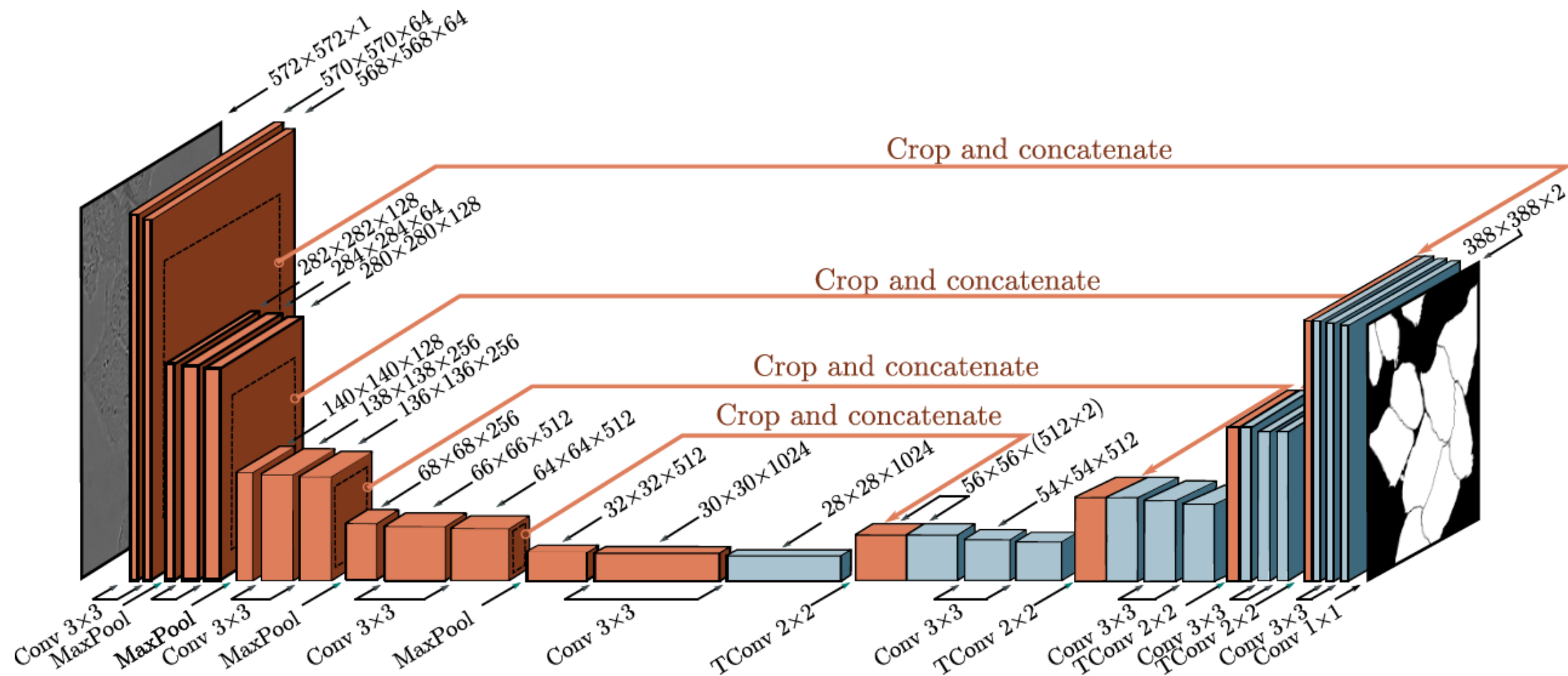
ImageNet History



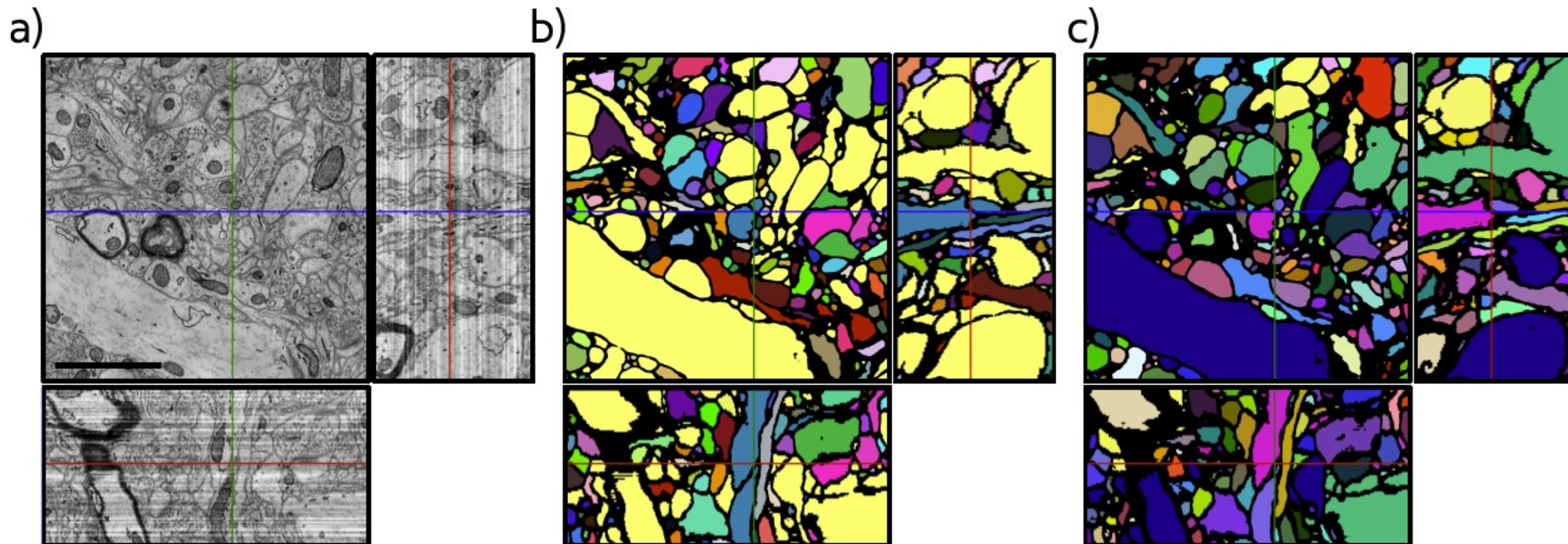
Convolution #2

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

U-Net (2016)



U-Net Results



Stacked hourglass networks (2016)

