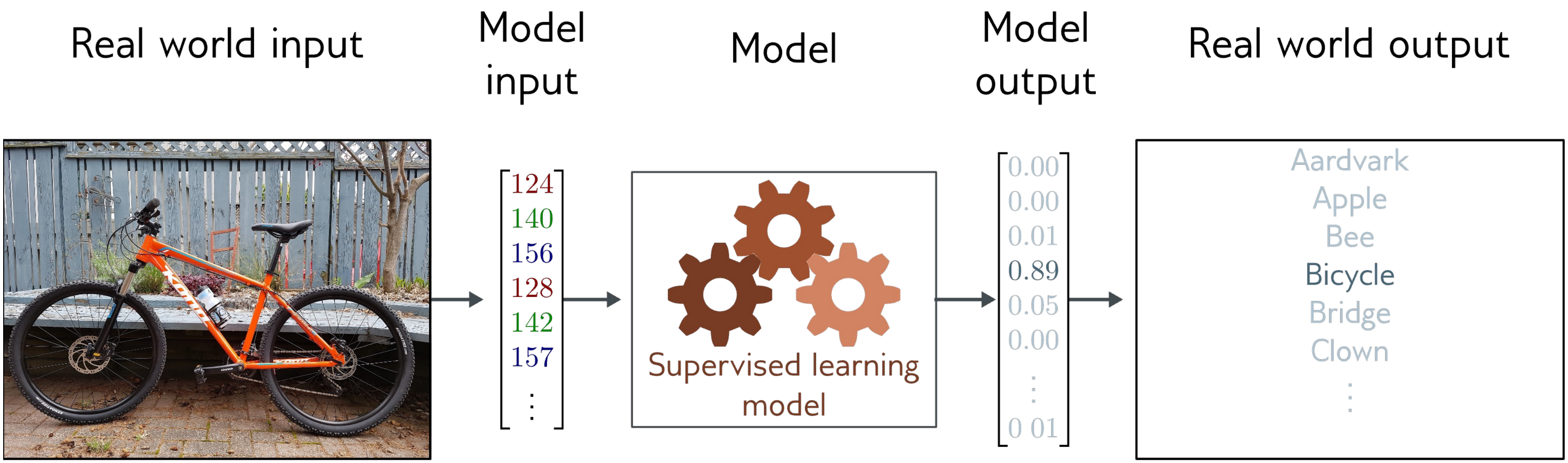
Convolutional Networks

DL4DS – Spring 2025

Convolutional networks

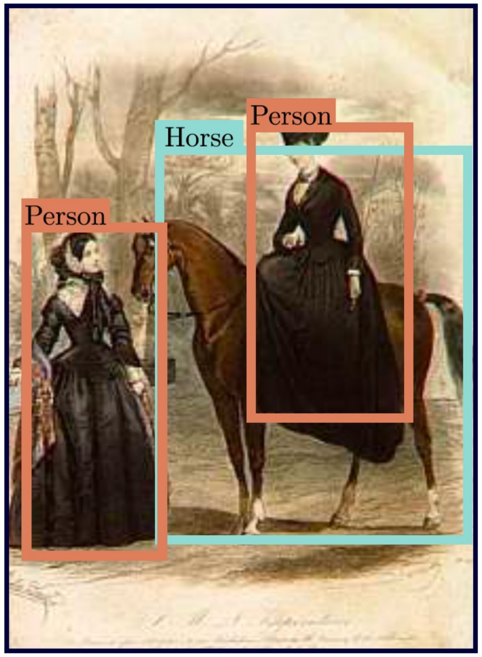
- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

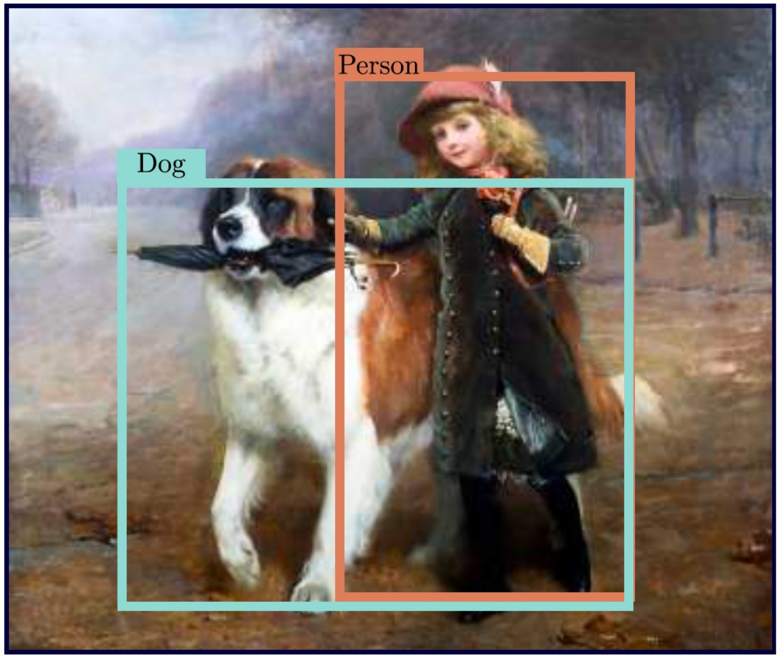
Image classification



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

Object detection (+ classification)





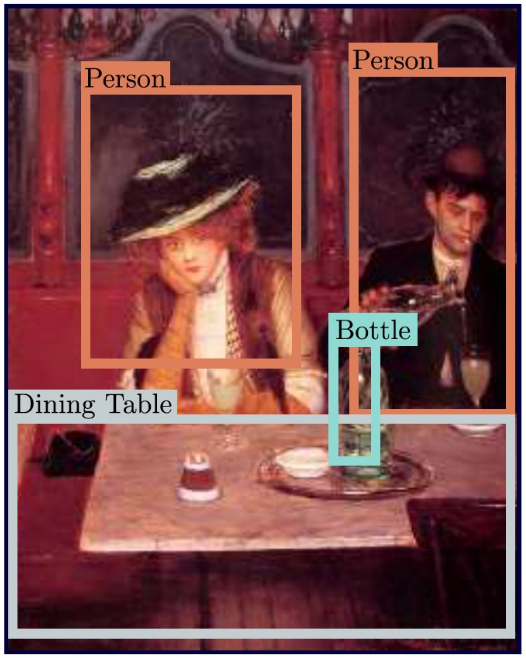
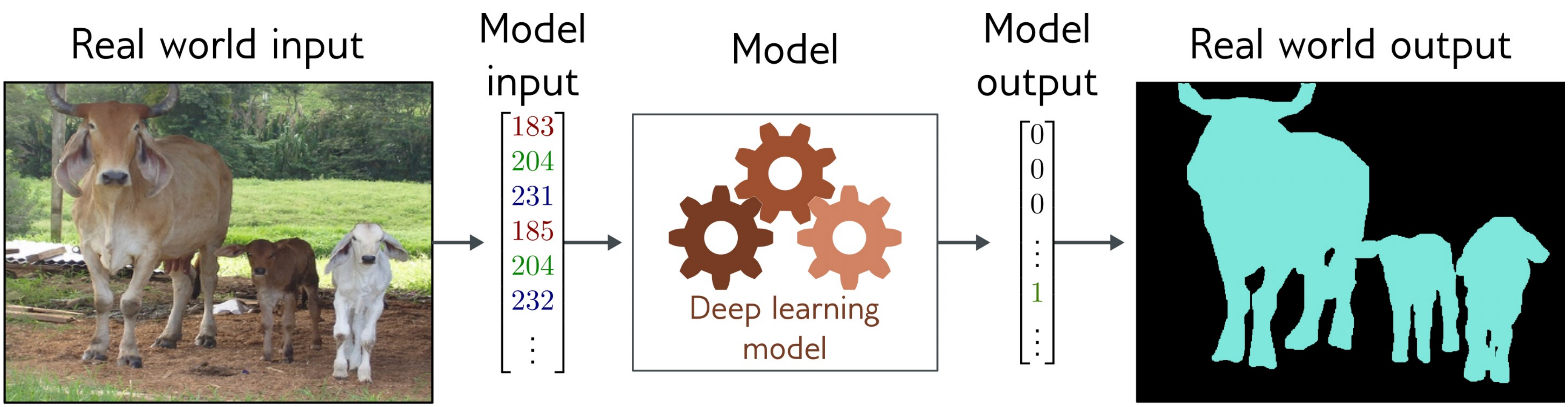


Image segmentation



- 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:

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

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:

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

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:

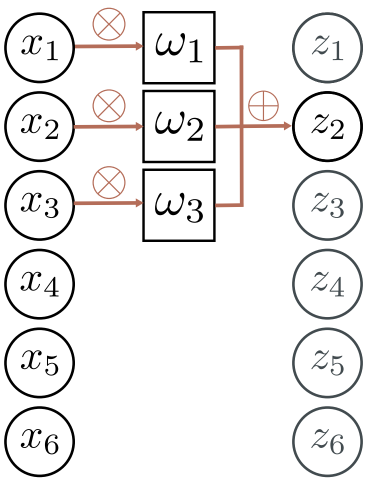
$$\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T$$

Kernel size = 3

** Not technically convolution because weights order is not reversed*

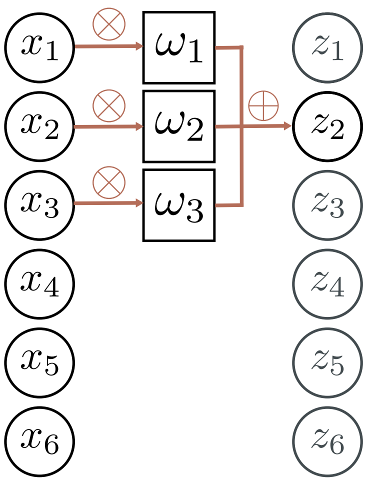
Convolution with kernel size 3

a)

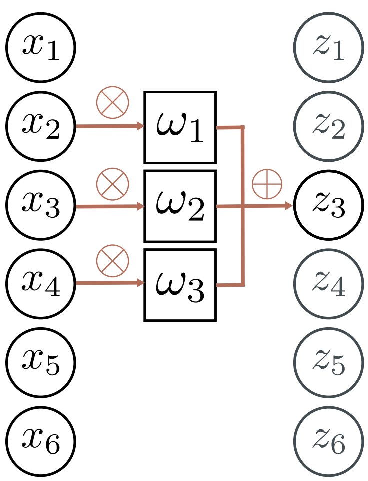


Convolution with kernel size 3

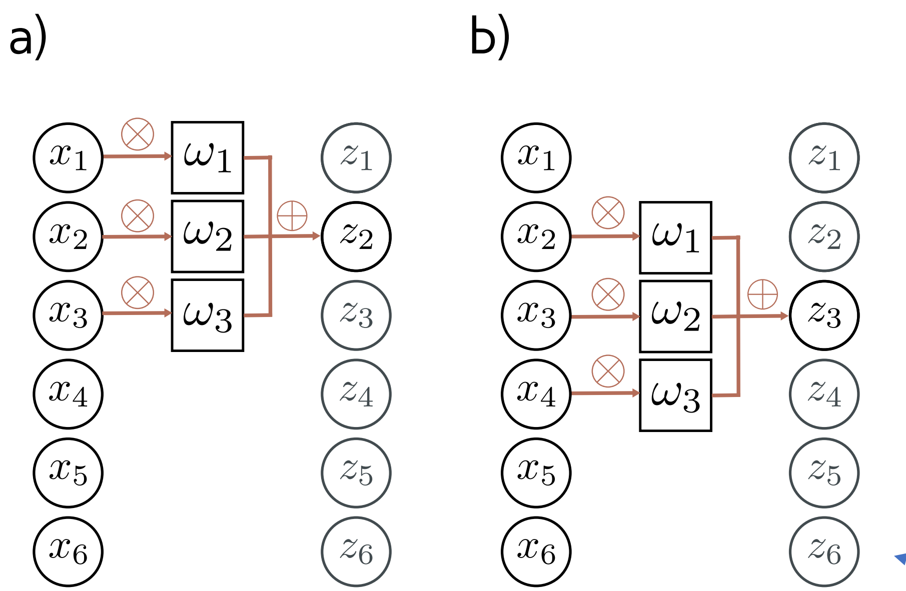
a)



b)



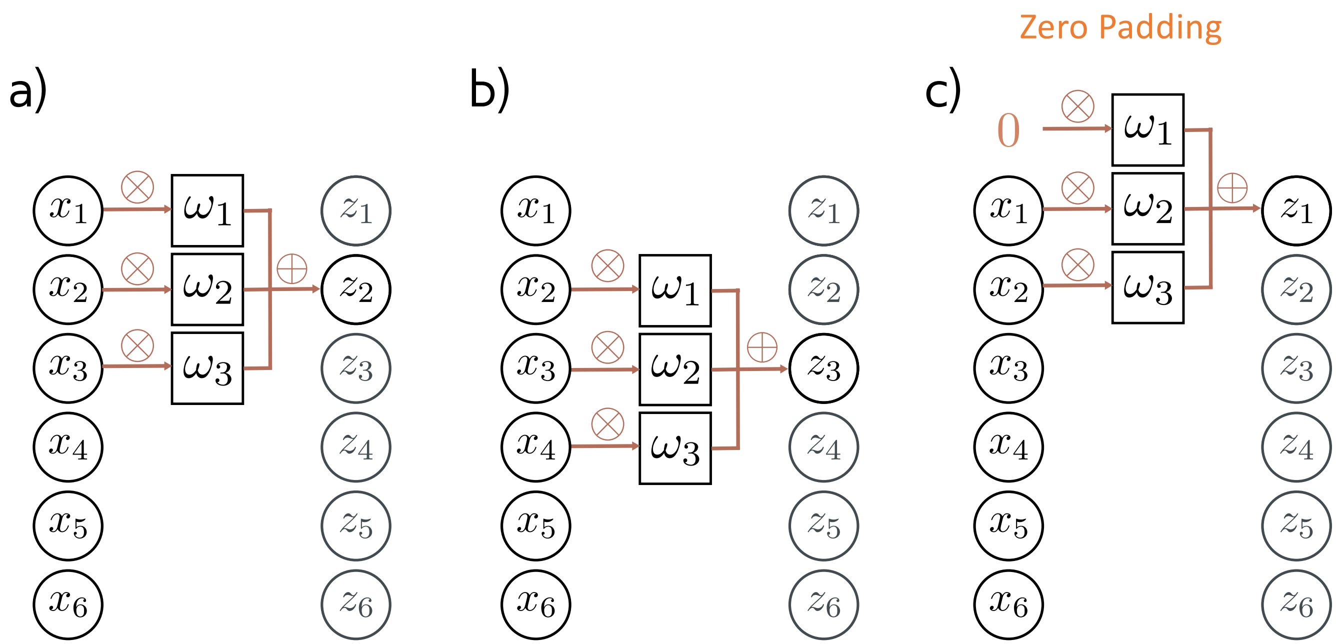
Convolution with kernel size 3



Equivariant to translation of input

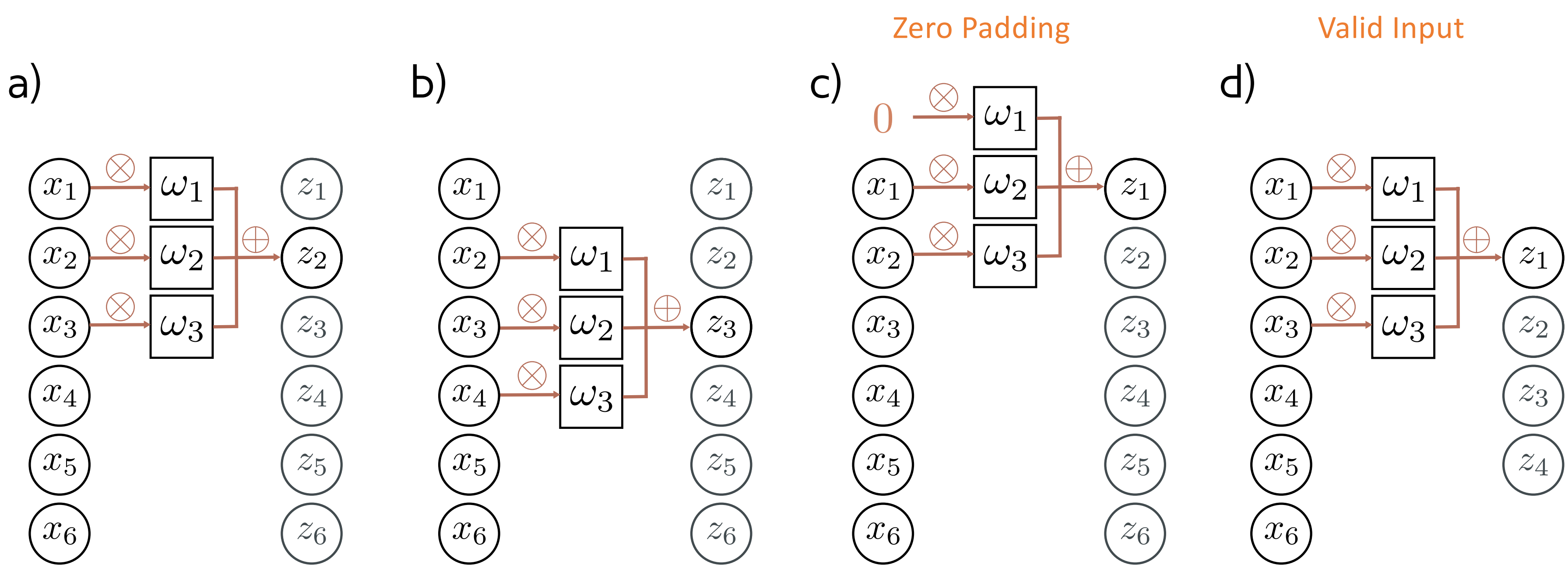
$$\mathbf{f}\left[\mathbf{t}[\mathbf{x}]\right] = \mathbf{t}\left[\mathbf{f}[\mathbf{x}]\right]$$

Zero padding



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

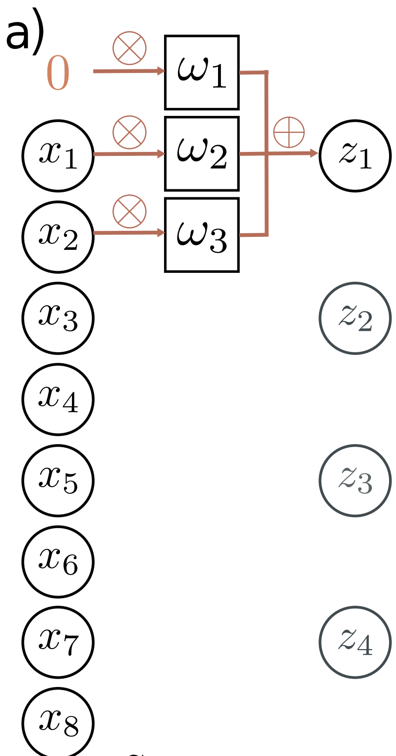
“Valid” convolutions



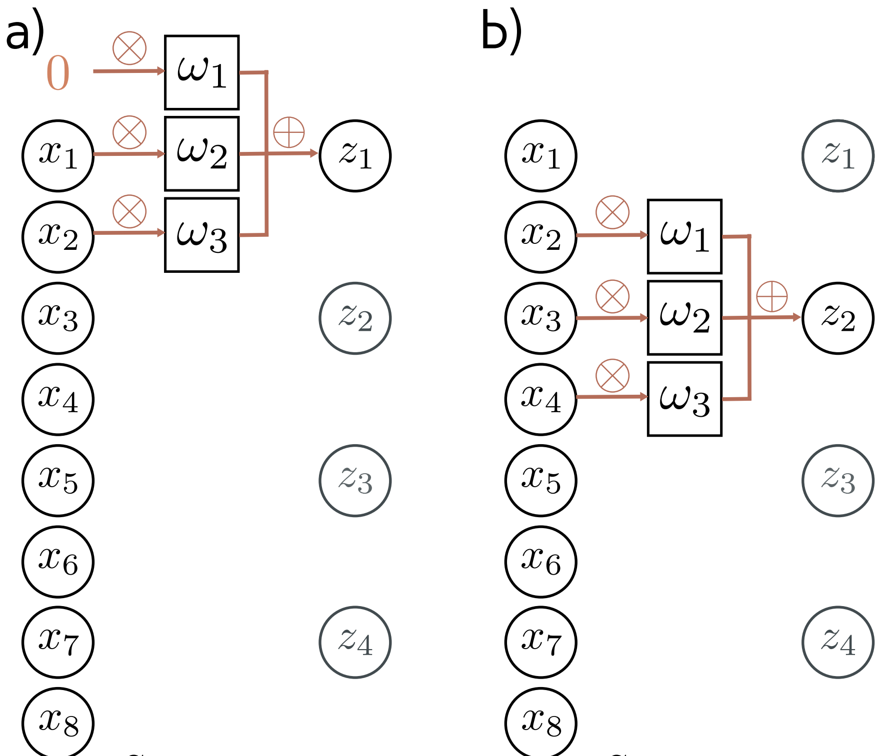
Only process positions where kernel falls in image (smaller output).

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

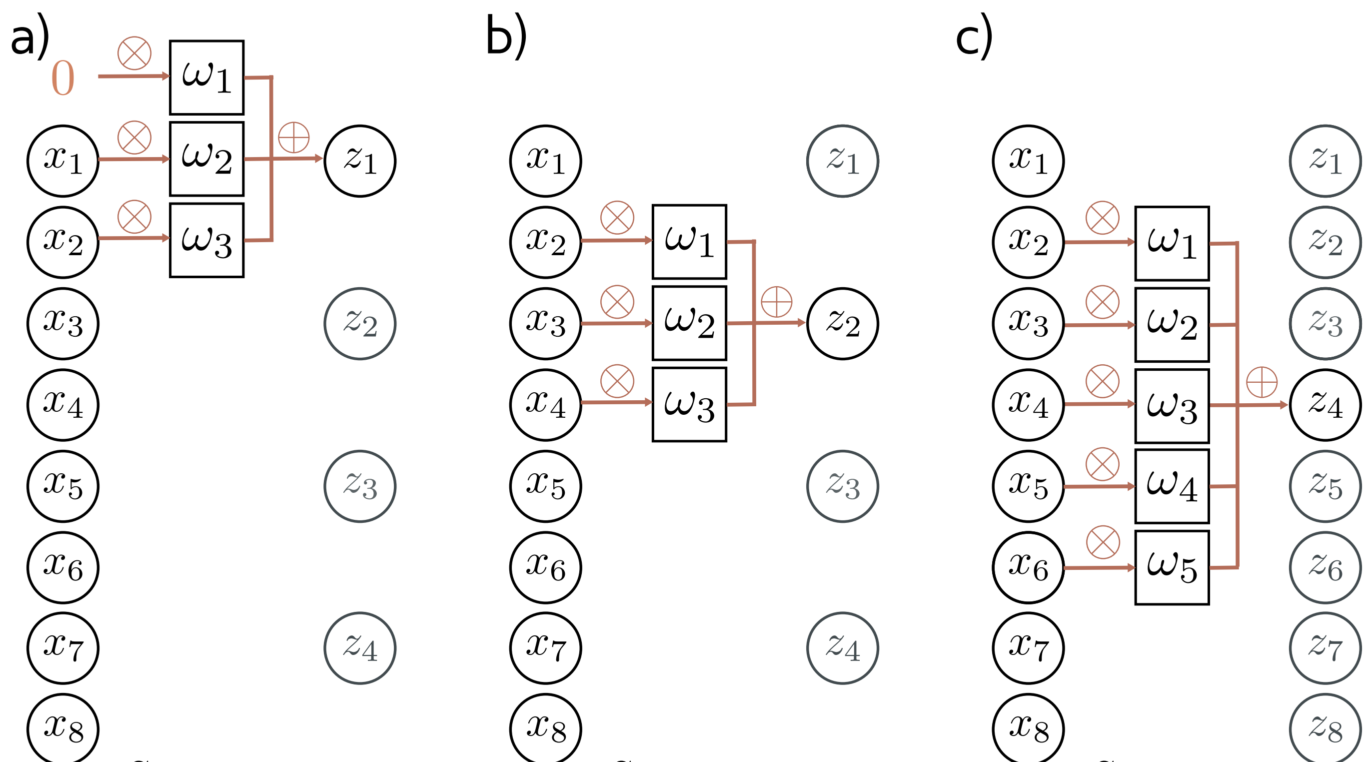


Size = 3
Stride = 2
Dilation = 1



Size = 3
Stride = 2
Dilation = 1

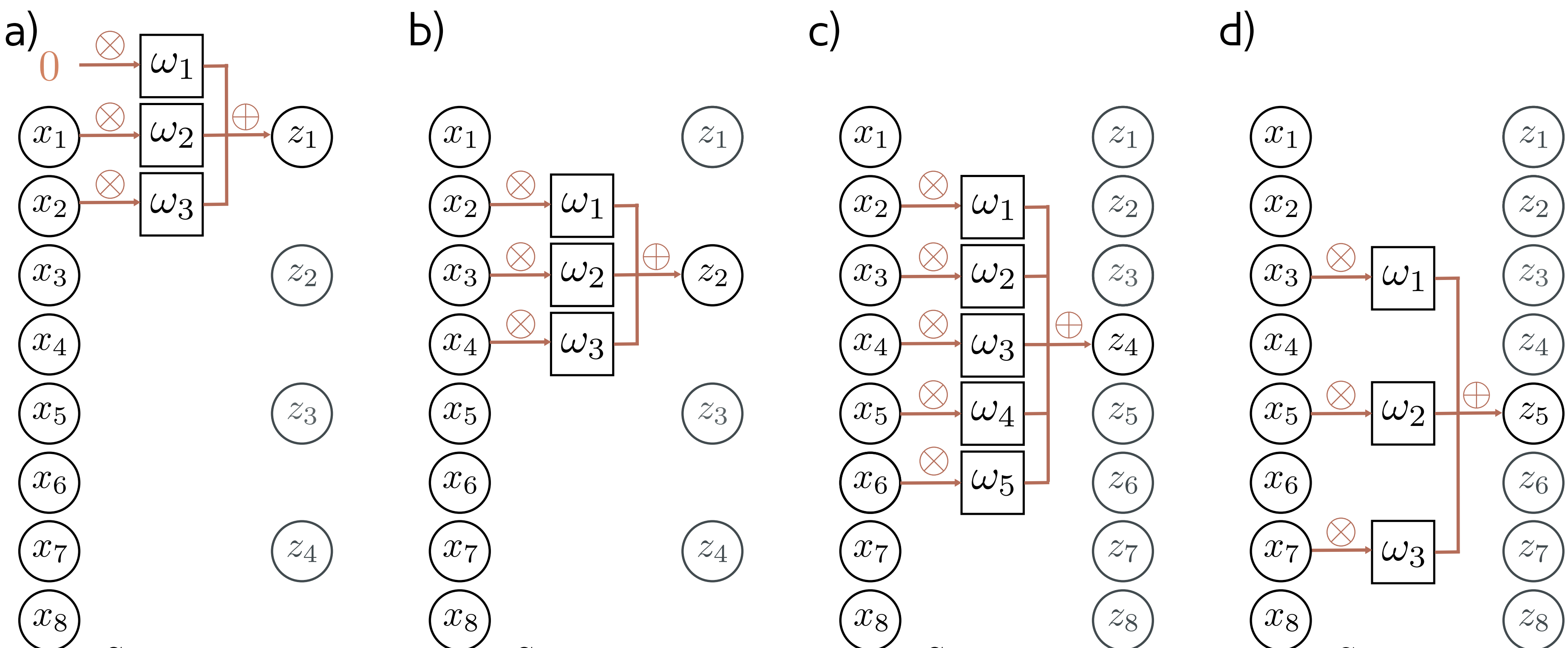
Size = 3
Stride = 2
Dilation = 1



Size = 3
Stride = 2
Dilation = 1

Size = 3
Stride = 2
Dilation = 1

Size = 5
Stride = 1
Dilation = 1



a) Size $= 3$
Stride $= 2$
Dilation $= 1$

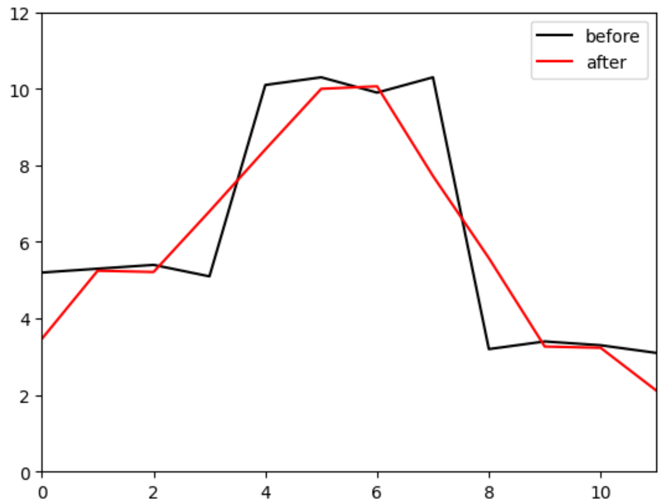
b) Size $= 3$
Stride $= 2$
Dilation $= 1$

c) Size $= 5$
Stride $= 1$
Dilation $= 1$

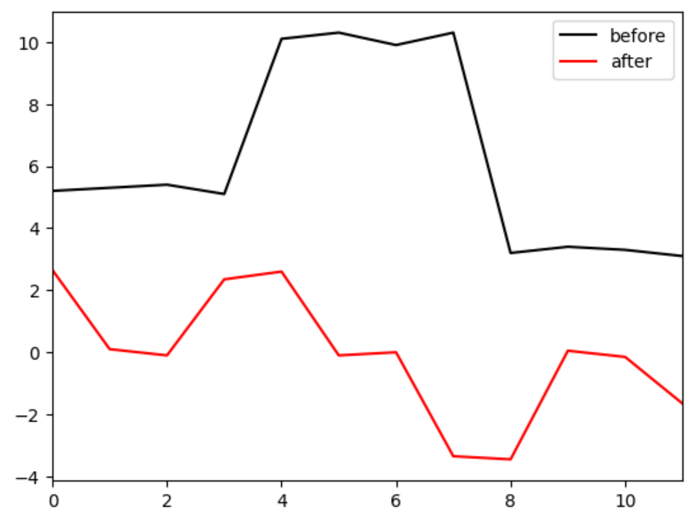
d) Size $= 3$
Stride $= 1$
Dilation $= 2$.

1-D Convolution Example

```
# Define a signal that we can apply convolution to
x = [5.2, 5.3, 5.4, 5.1, 10.1, 10.3, 9.9, 10.3, 3.2, 3.4, 3.3, 3.1]
```




```
omega = [0.33,0.33,0.33]
```




```
omega = [-0.5,0,0.5]
```

```
length=3, stride=1, dilation=1, zero padding
```

Convolutional networks

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

Convolutional layer
(size: 3, stride: 1, dilation: 1)

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

Special case of fully-connected network

Convolutional network:

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

Fully connected network:
(D inputs, D hidden units)

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

Special case of fully-connected network

Convolutional network:

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

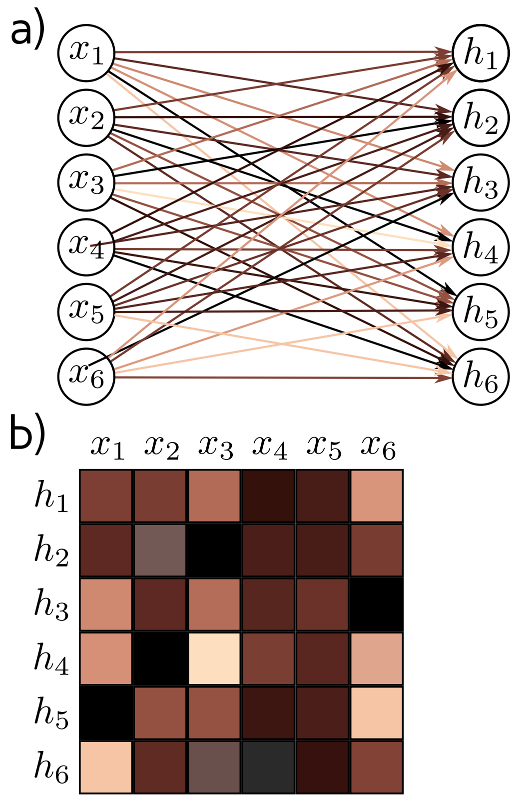
3 weights, 1 bias

Fully connected network:
(D inputs, D hidden units)

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

D^2 weights, D biases

Special case of fully-connected network



6 inputs to each hidden unit

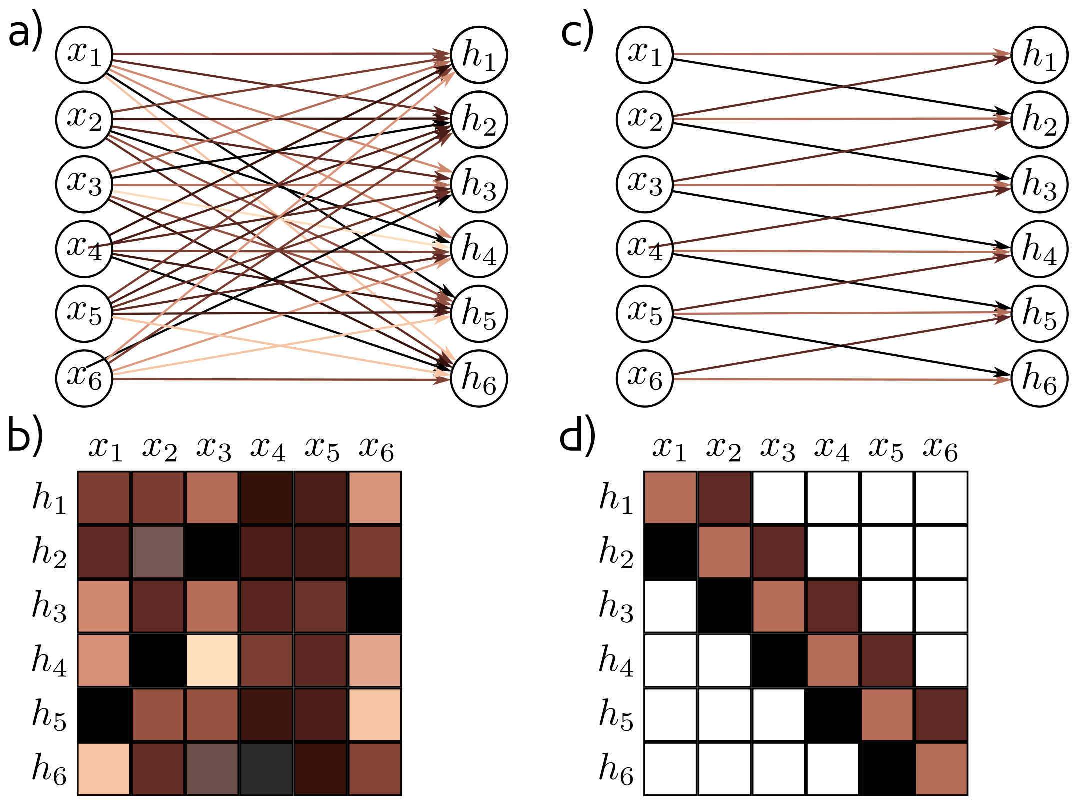
Weight Matrix

Fully connected network

Bias is implied

Special case of fully-connected network

Weight Matrices



Fully connected network

Convolution, kernel 3, stride 1, dilation 1

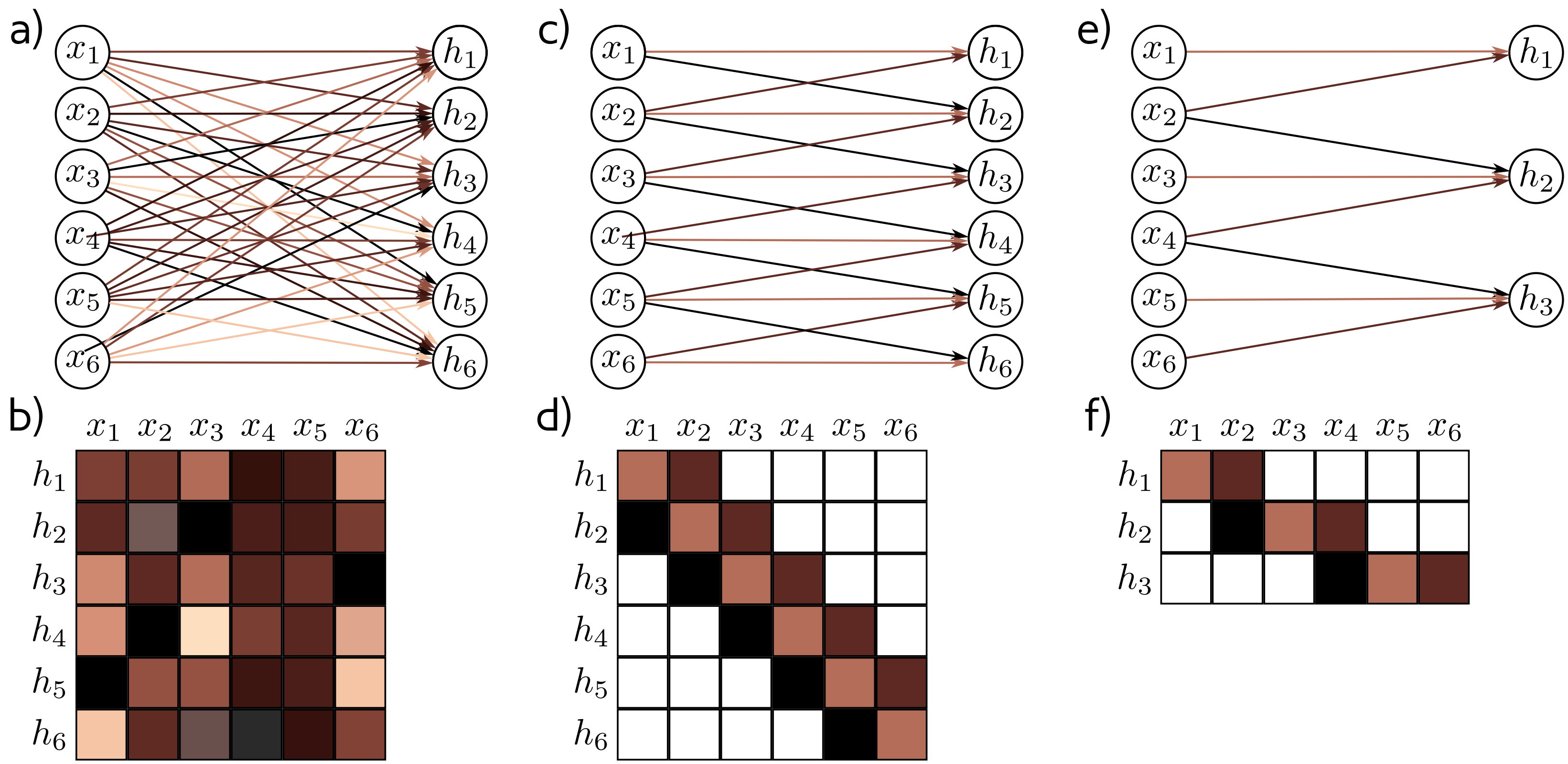
3 inputs to each hidden unit

Bias is implied

Special case of fully-connected network

Bias is implied

Weight Matrices



Fully connected network

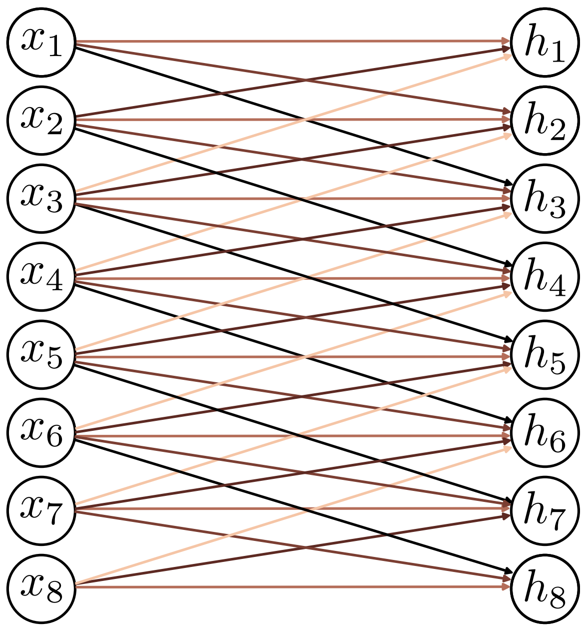
Convolution, size 3, stride 1, dilation 1, zero padding

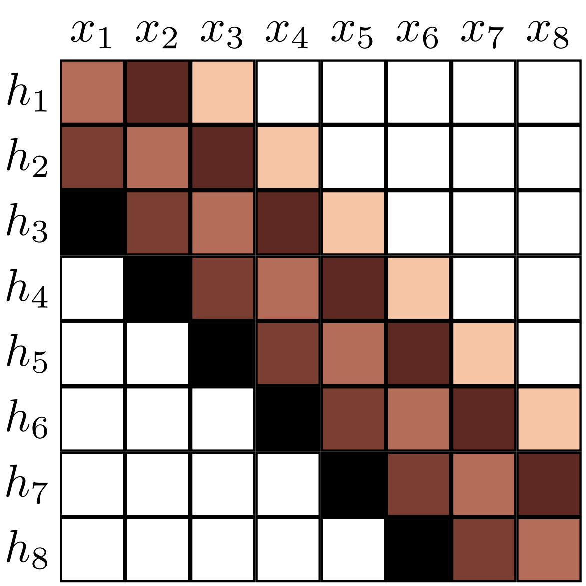
Convolution, size 3, stride 2, dilation 1, zero padding

Question 1

Bias is implied

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





slido

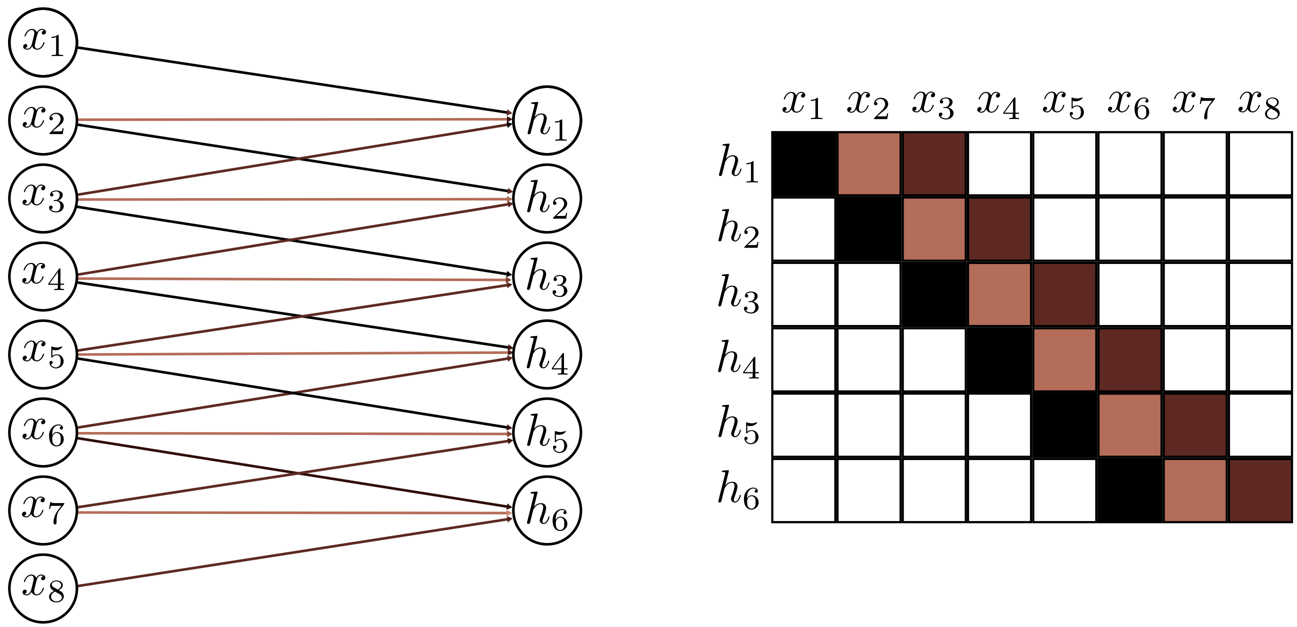
Convolution Configuration

ⓘ Start presenting to display the poll results on this slide.

Question 2

Bias is implied

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



slido

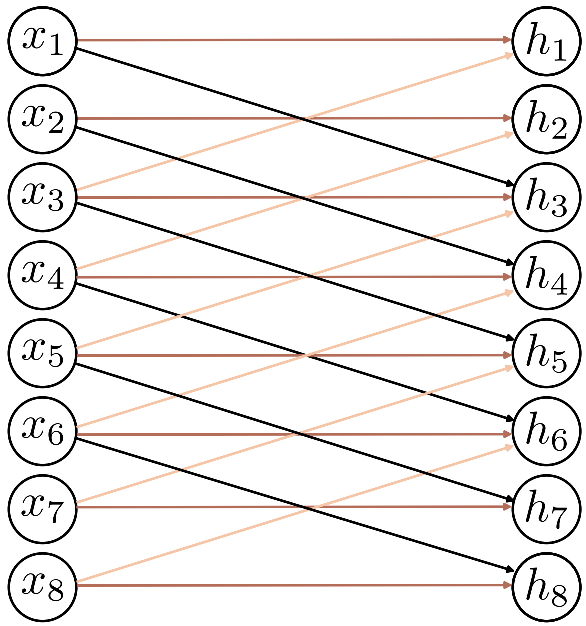
Conv Config 2

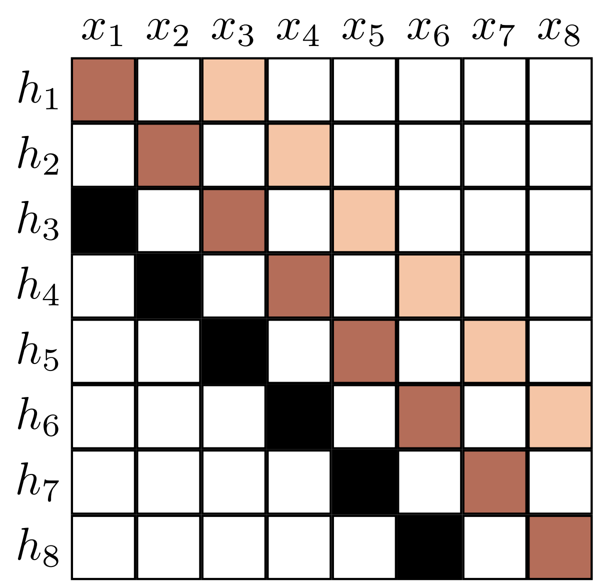
ⓘ Start presenting to display the poll results on this slide.

Question 3

Bias is implied

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





slido

Conv Config 3

ⓘ Start presenting to display the poll results on this slide.

Convolutional networks

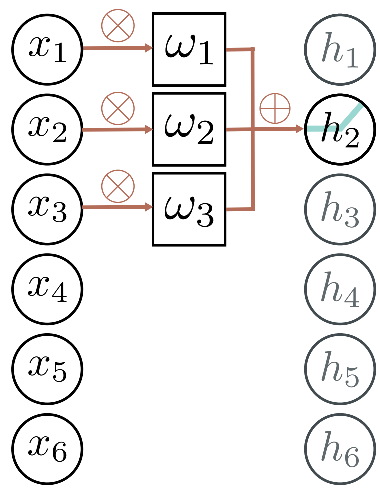
- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

Channels

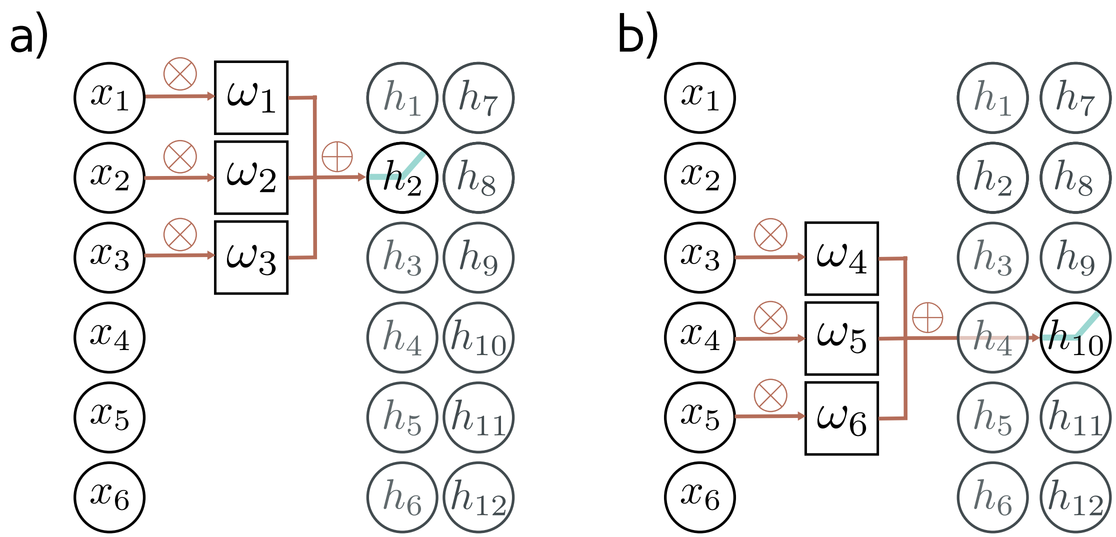
- The convolutional operation averages together the inputs
- Plus passes through ReLU function
- Result is loss of information
- Solution:
  - apply several convolutions and stack them in channels
  - Sometimes also called feature maps

Two output channels, one input channel

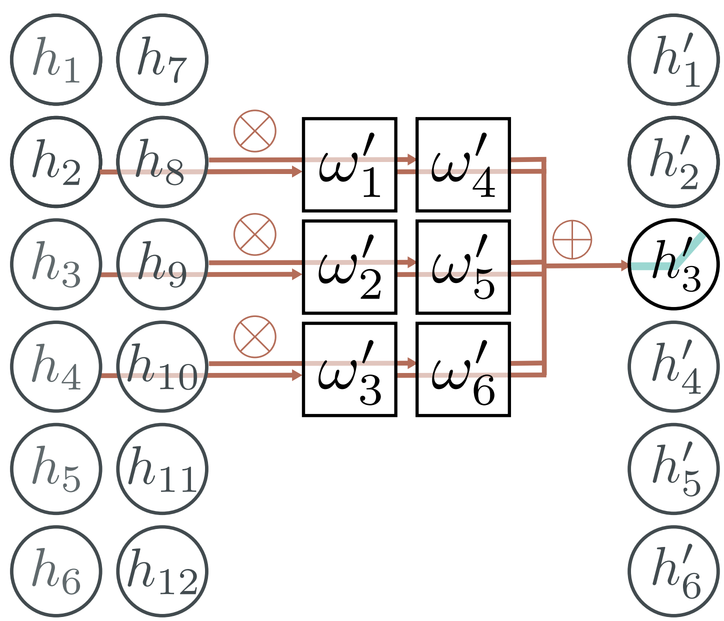
a)



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

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

- If there are C_i input channels and C_o output channels

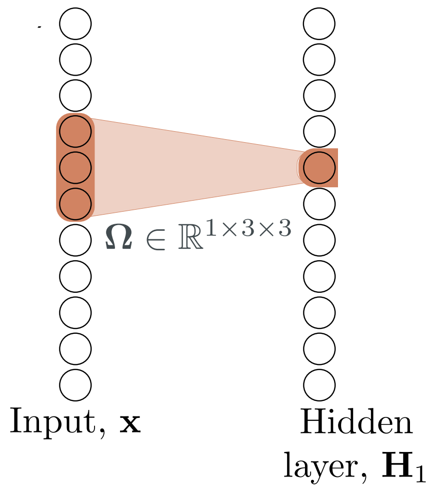
$$\boldsymbol{\Omega} \in \mathbb{R}^{C_i \times C_o \times K} \qquad \boldsymbol{\beta} \in \mathbb{R}^{C_o}$$

Convolutional networks

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

Receptive fields

$\mathbb{R}^{C_i \times C_o \times K}$

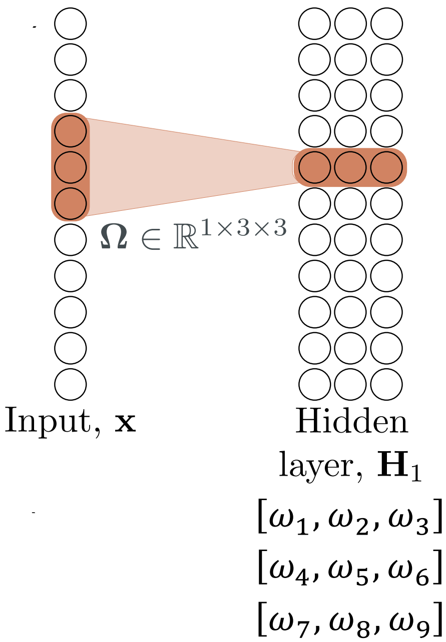


$[\omega_1, \omega_2, \omega_3]$

Indicates how many neighboring pixels influence the current output.

Receptive fields

$\mathbb{R}^{C_i \times C_o \times K}$

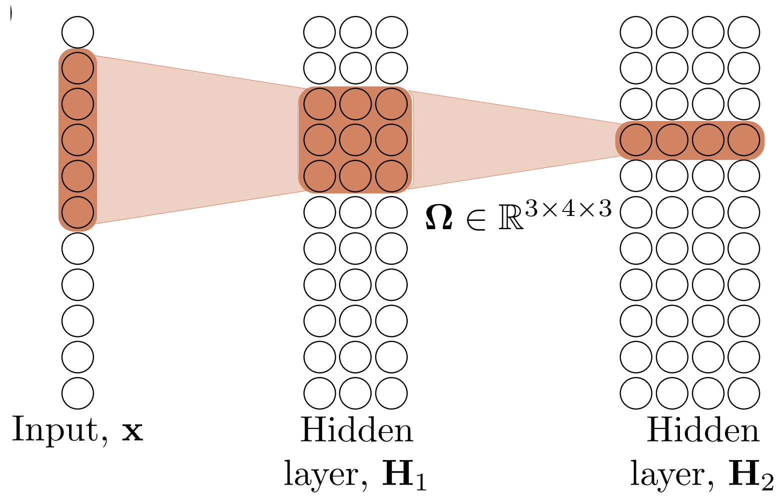


Receptive field only dependent on filter size, not number of channels

Each channel has a different set of filter weights.

Receptive fields

$\mathbb{R}^{C_i \times C_o \times K}$



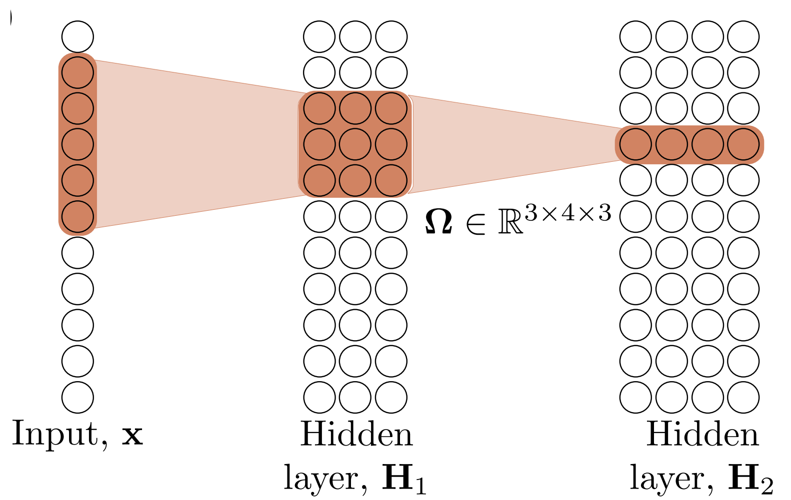
$[\omega_1, \omega_2, \omega_3]$
$[\omega_4, \omega_5, \omega_6]$
$[\omega_7, \omega_8, \omega_9]$

$[\omega_1, \omega_2, \omega_3]$
$[\omega_4, \omega_5, \omega_6]$
$[\omega_7, \omega_8, \omega_9]$
$[\omega_{10}, \omega_{11}, \omega_{12}]$

Influence compounds in subsequent layers.

Receptive fields

$\mathbb{R}^{C_i \times C_o \times K}$



$[\omega_1, \omega_2, \omega_3]$
$[\omega_4, \omega_5, \omega_6]$
$[\omega_7, \omega_8, \omega_9]$

$[\omega_1, \omega_2, \omega_3]$
$[\omega_4, \omega_5, \omega_6]$
$[\omega_7, \omega_8, \omega_9]$
$[\omega_{10}, \omega_{11}, \omega_{12}]$

The area of support is equivalent to the area of support of convolution.

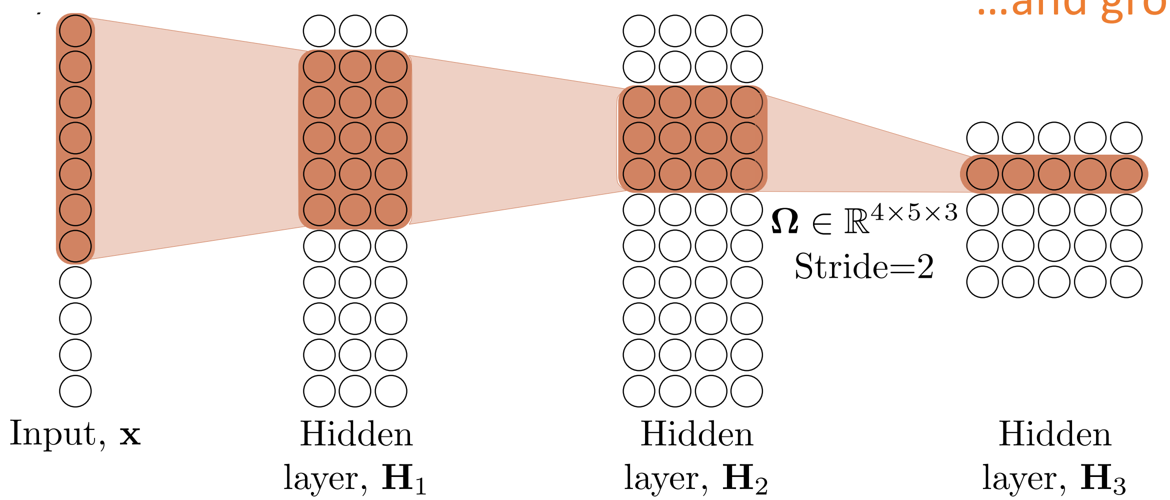
$$[\omega_1, \omega_2, \omega_3] \otimes [\omega_4, \omega_5, \omega_6] \\ = [\omega_1, \omega_2, \omega_3, \omega_4, \omega_5]$$

Receptive fields

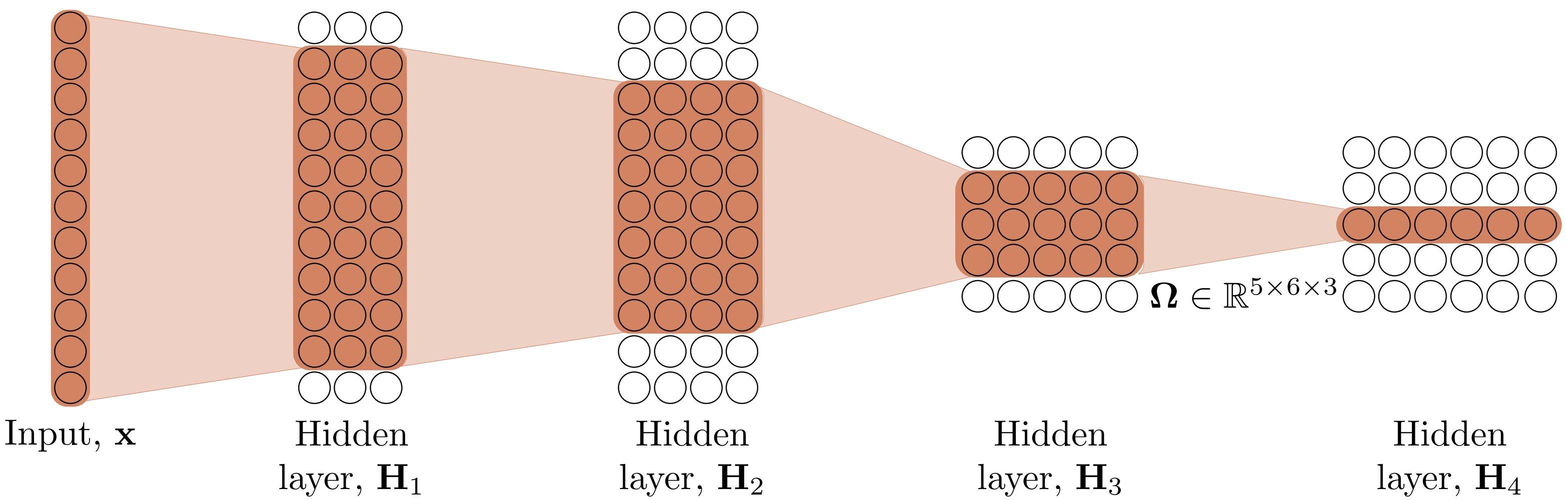
$\mathbb{R}^{C_i \times C_o \times K}$

$$[\omega_1, \omega_2, \omega_3, \omega_4, \omega_5] \otimes [\omega_1, \omega_2, \omega_3] = [\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7]$$

…and grows with each layer



Receptive fields $\mathbb{R}^{C_i \times C_o \times K}$

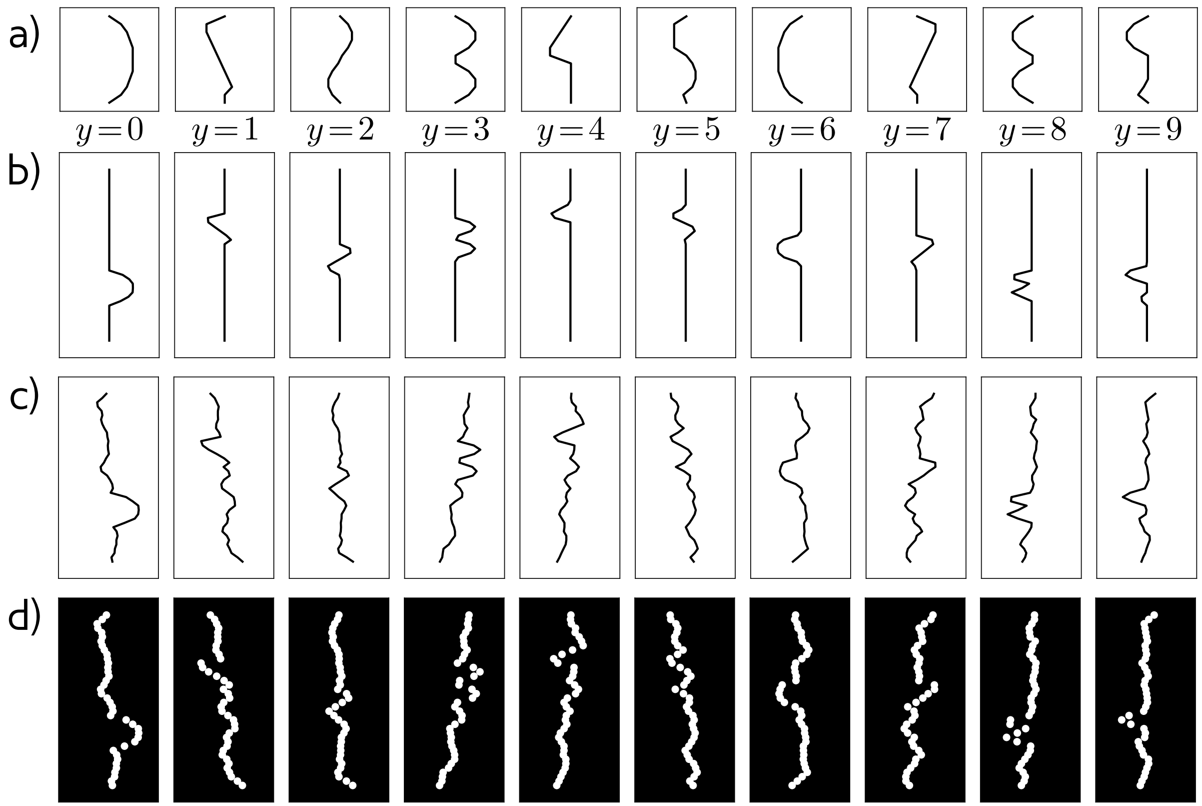


Reminder: Receptive field only dependent on filter size, not number of channels.

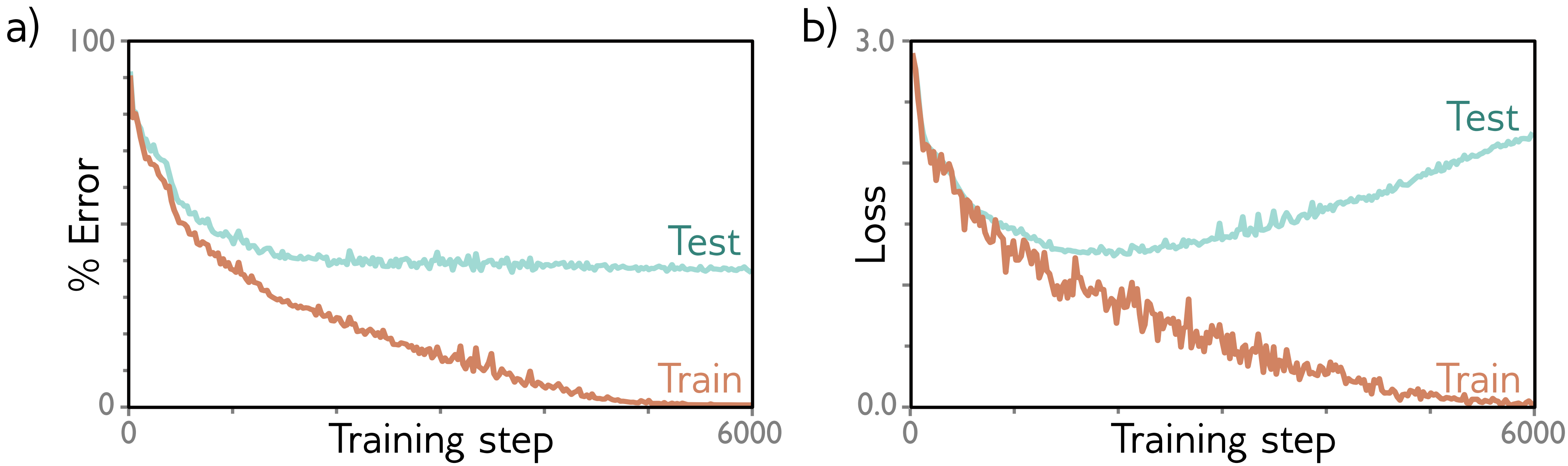
Convolutional networks

- 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



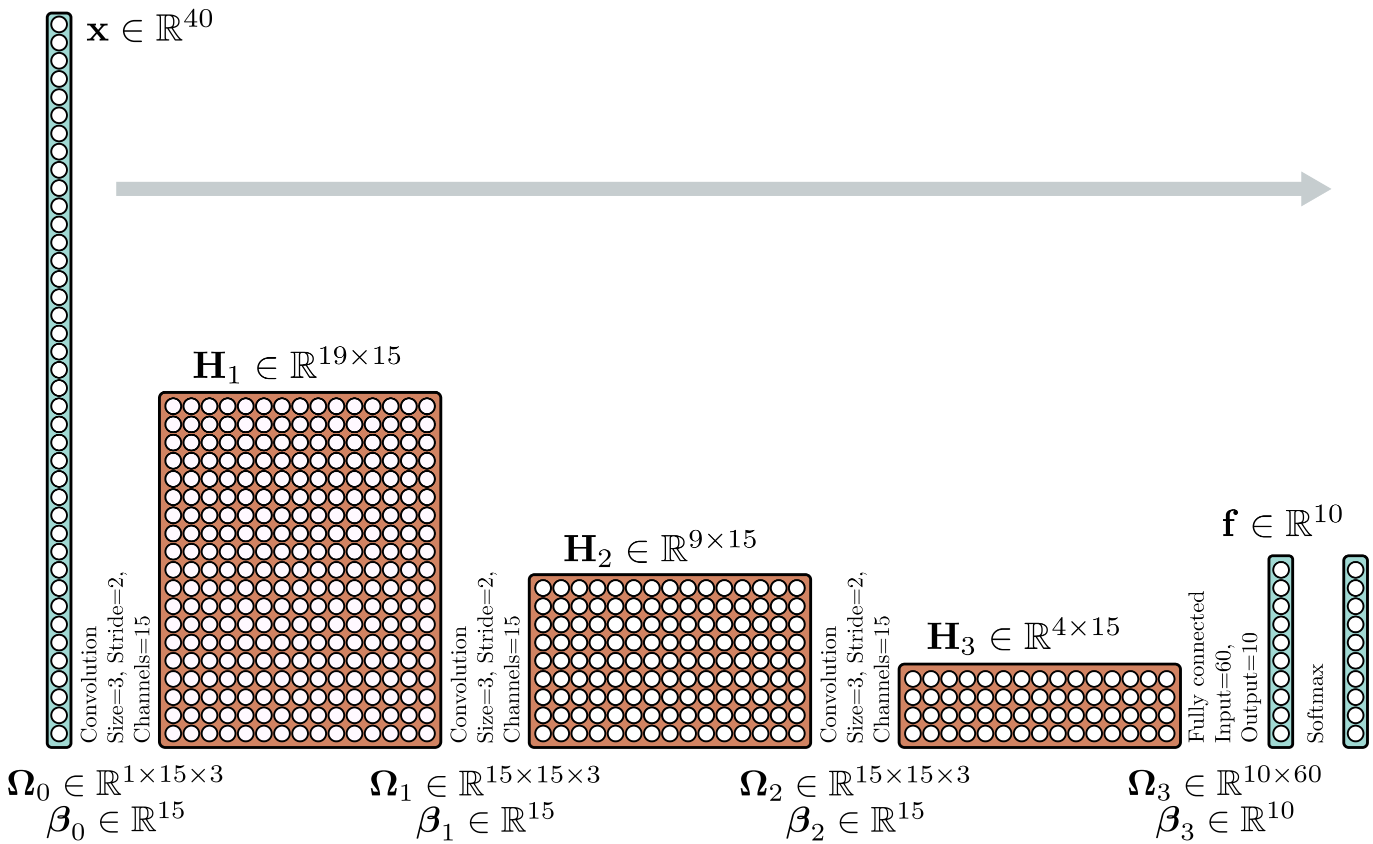
Total parameters = 150,185

Convolutional network

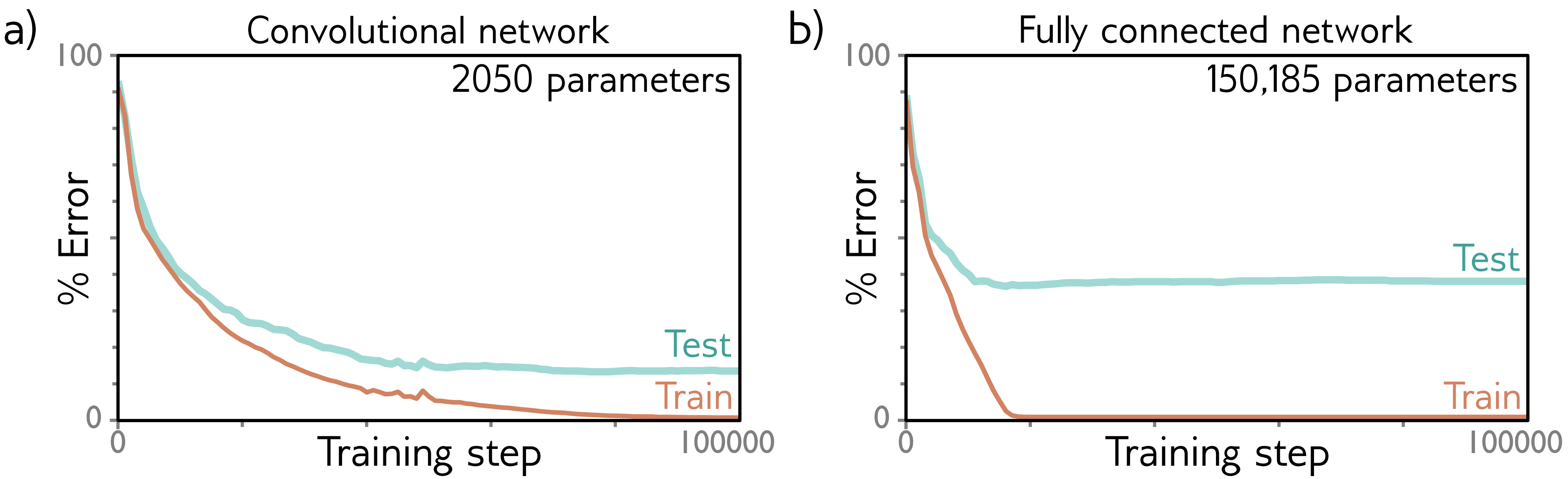
- Four hidden layers
- Three convolutional layers
- One fully-connected layer
- Softmax at end
- Total parameters = 2050
- Trained for 100,000 steps with SGD, LR = 0.01, batch size 100

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Sequential                               [100, 10]                 --
├─Conv1d: 1-1                            [100, 15, 19]             60
├─ReLU: 1-2                              [100, 15, 19]             --
├─Conv1d: 1-3                            [100, 15, 9]              690
├─ReLU: 1-4                              [100, 15, 9]              --
├─Conv1d: 1-5                            [100, 15, 4]              690
├─ReLU: 1-6                              [100, 15, 4]              --
├─Flatten: 1-7                           [100, 60]                 --
├─Linear: 1-8                            [100, 10]                 610
==========================================================================================
Total params: 2,050
Trainable params: 2,050
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 1.07
==========================================================================================
Input size (MB): 0.02
Forward/backward pass size (MB): 0.39
Params size (MB): 0.01
Estimated Total Size (MB): 0.42
==========================================================================================
```

MNIST-1D convolutional network



Performance



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

2D Convolution

Convolution #2

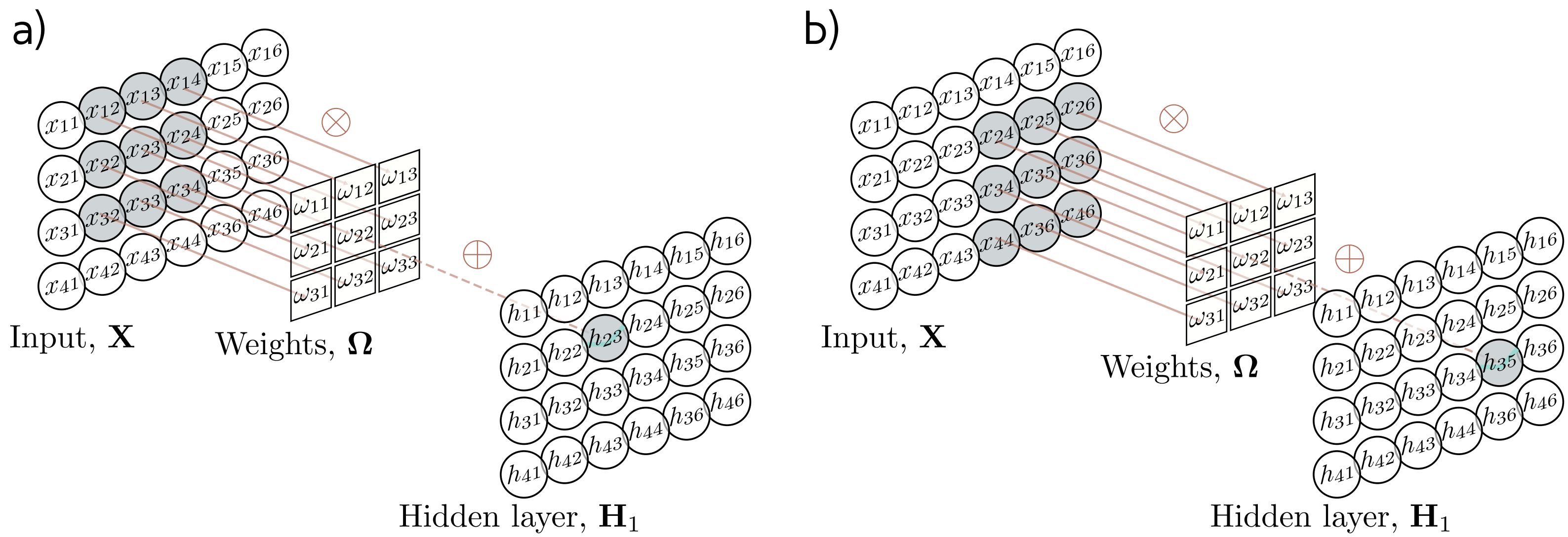
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

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

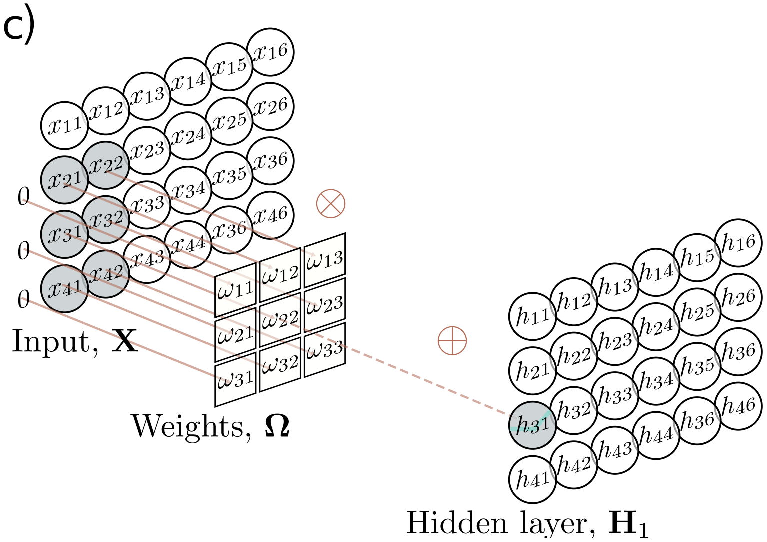
$$h_{i,j} = \mathrm{a}\left[\beta + \sum_{m=1}^{3}\sum_{n=1}^{3}\omega_{m,n}x_{i+m-2,j+n-2}\right]$$

2D Convolution

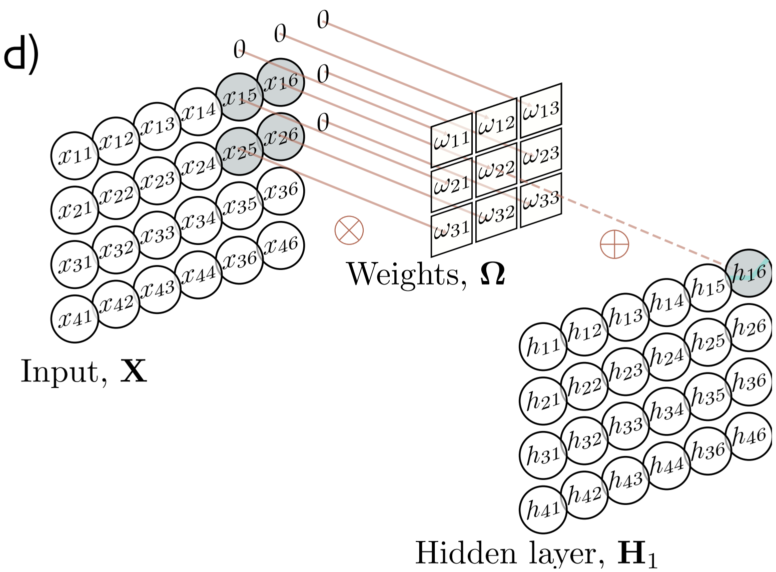


2D Convolution with Zero Padding

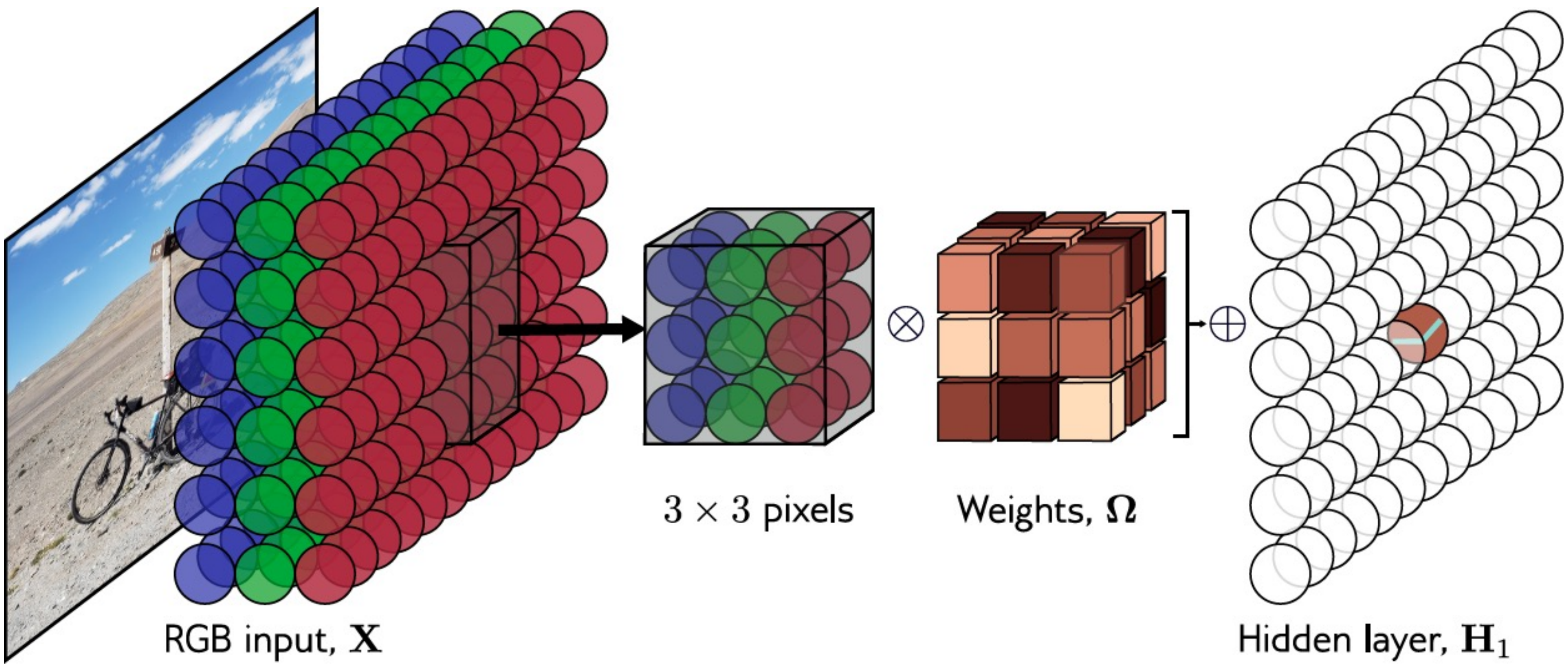
c)



d)



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 x K

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

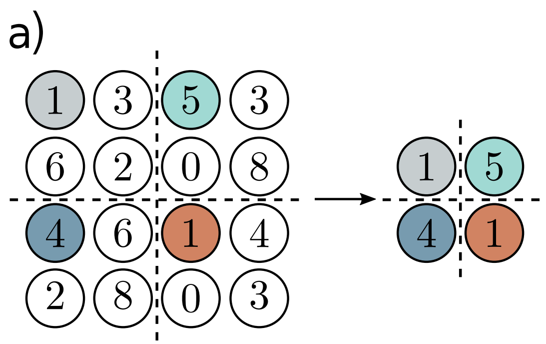
- If there are C_i input channels and C_o output channels

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

Convolution #2

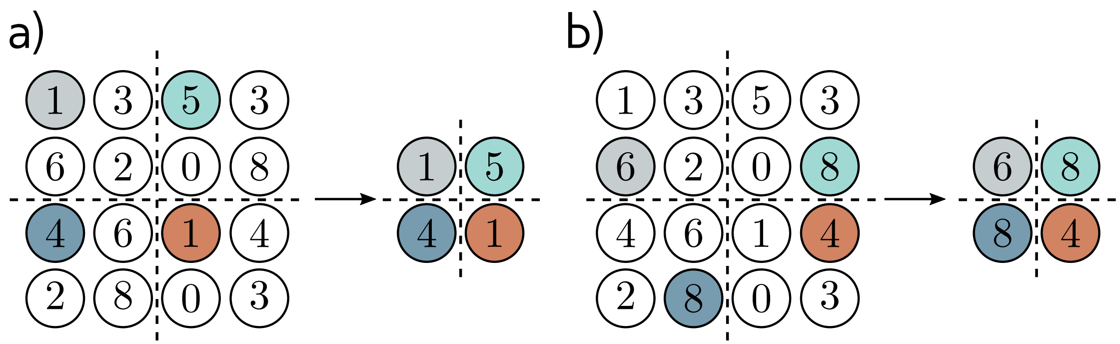
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

Downsampling



Sample every other position (equivalent to stride two)

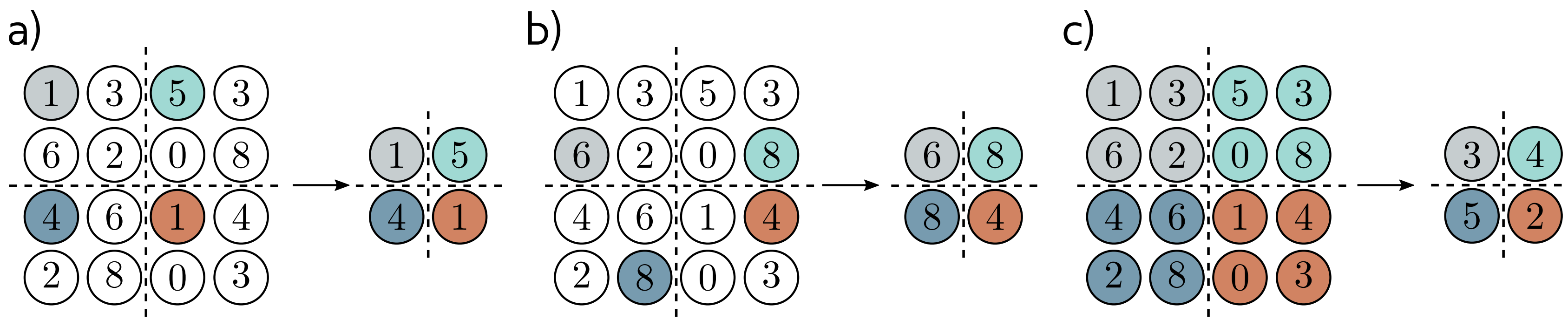
Downsampling



Sample every other position (equivalent to stride two)

Max pooling (partial invariance to translation)

Downsampling



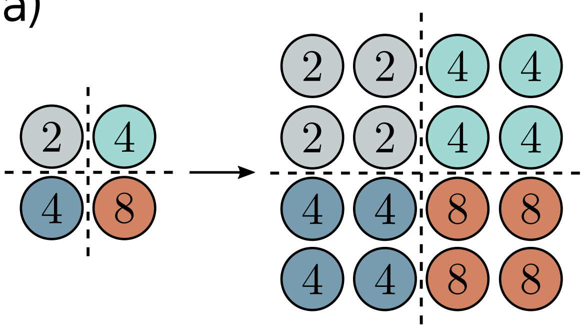
Sample every other position (equivalent to stride two)

Max pooling (partial invariance to translation)

Mean pooling

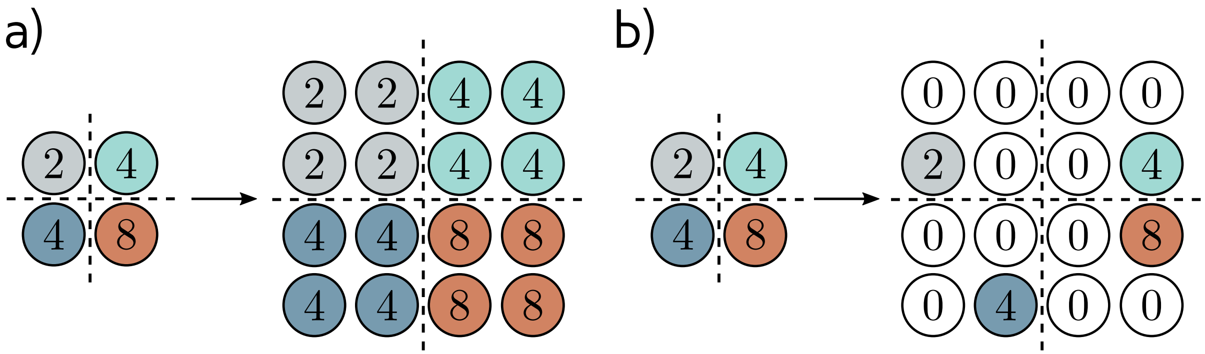
Upsampling

a)



Duplicate

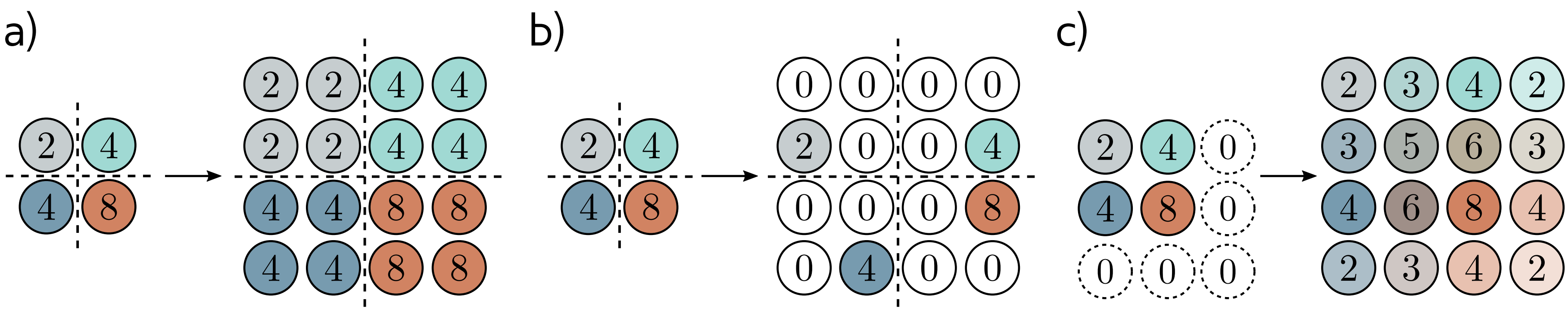
Upsampling



Duplicate

Max-upsampling

Upsampling

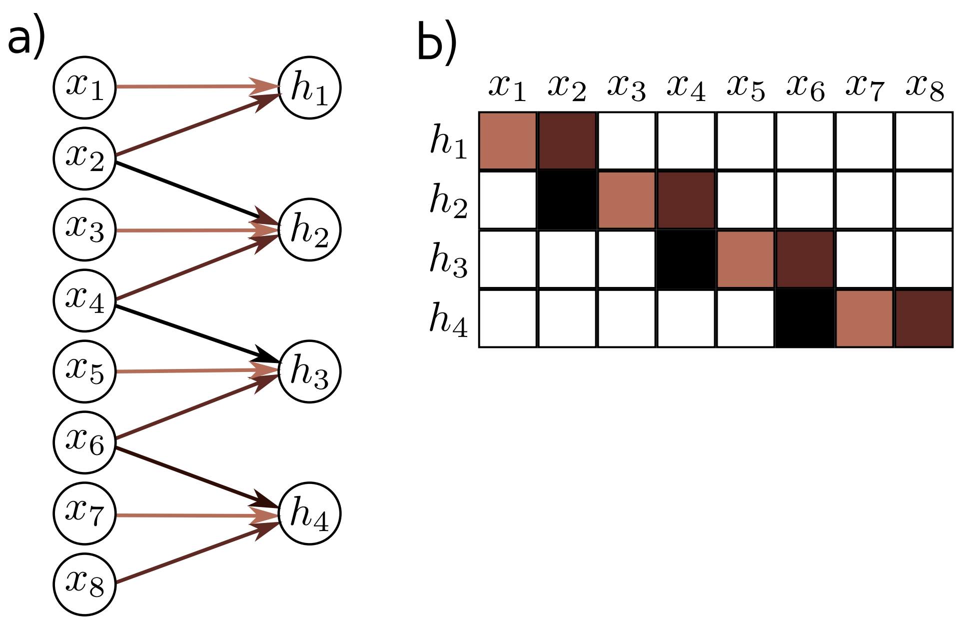


Duplicate

Max-upsampling

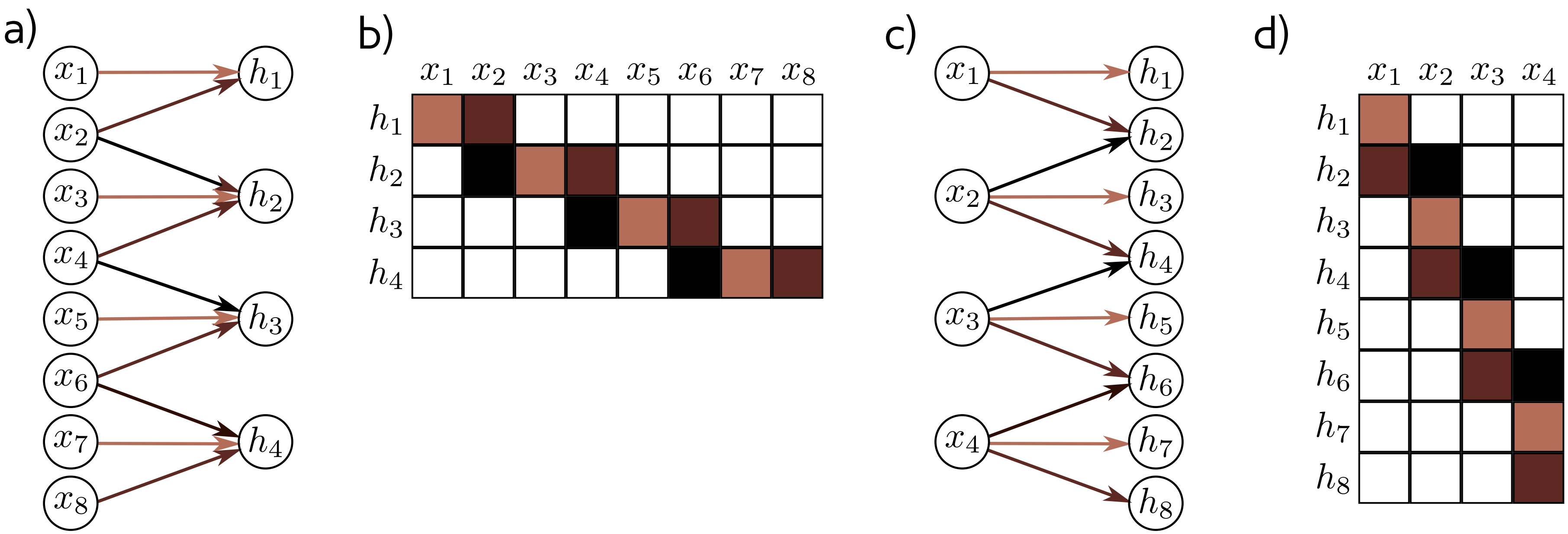
Bilinear interpolation

Transposed convolutions



Kernel size 3, Stride 2 convolution

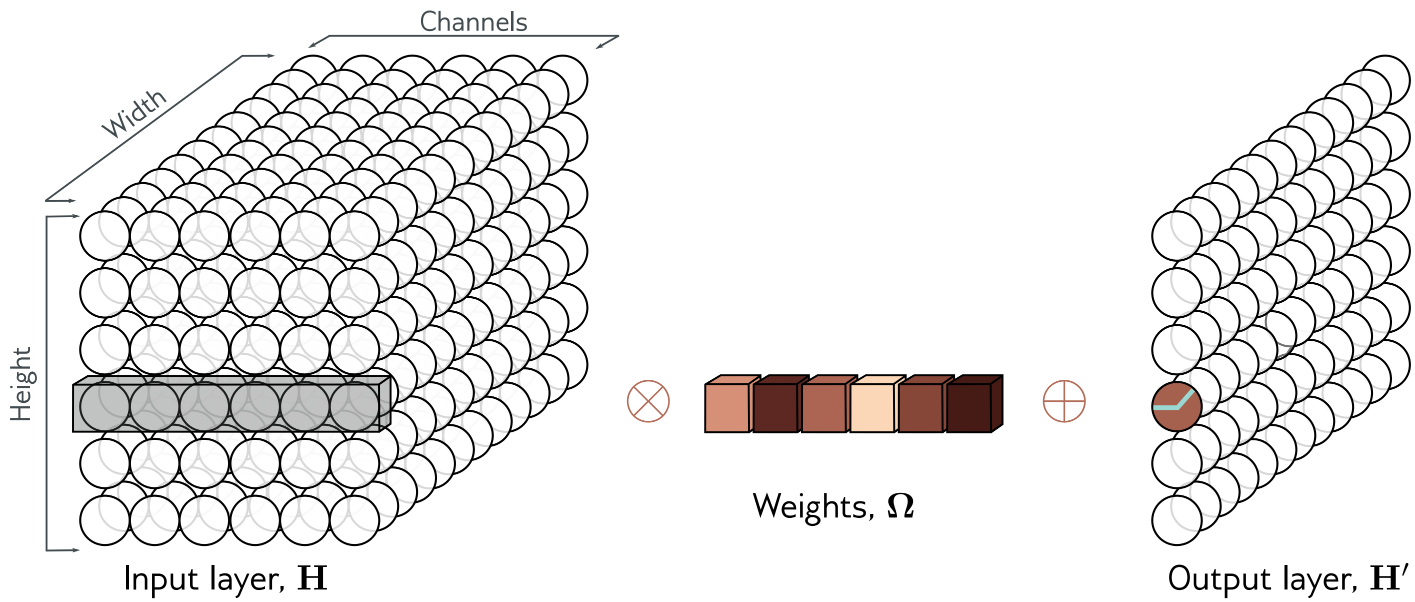
Transposed convolutions



Kernel size 3, Stride 2 convolution

Transposed convolution

1x1 convolution



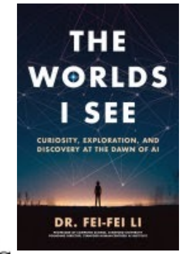
- Mixes channels
- Can change number of channels
- Equivalent to running same fully connected network at each position

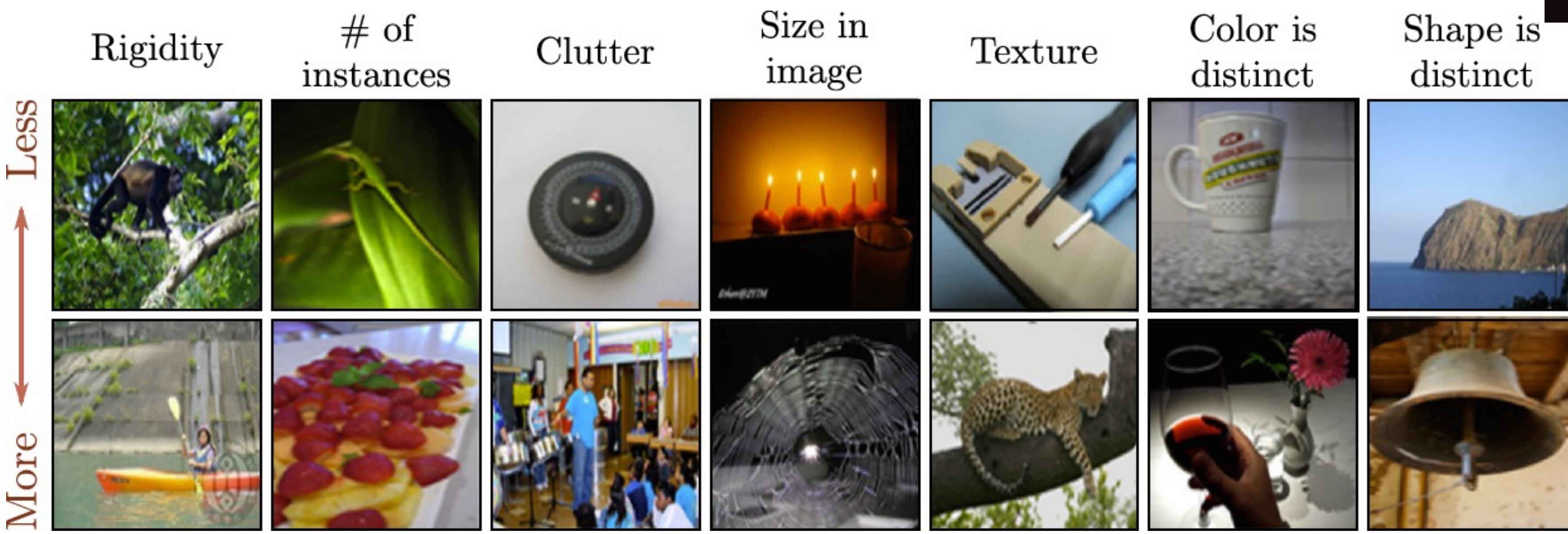
Convolution #2

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

ImageNet 1K database

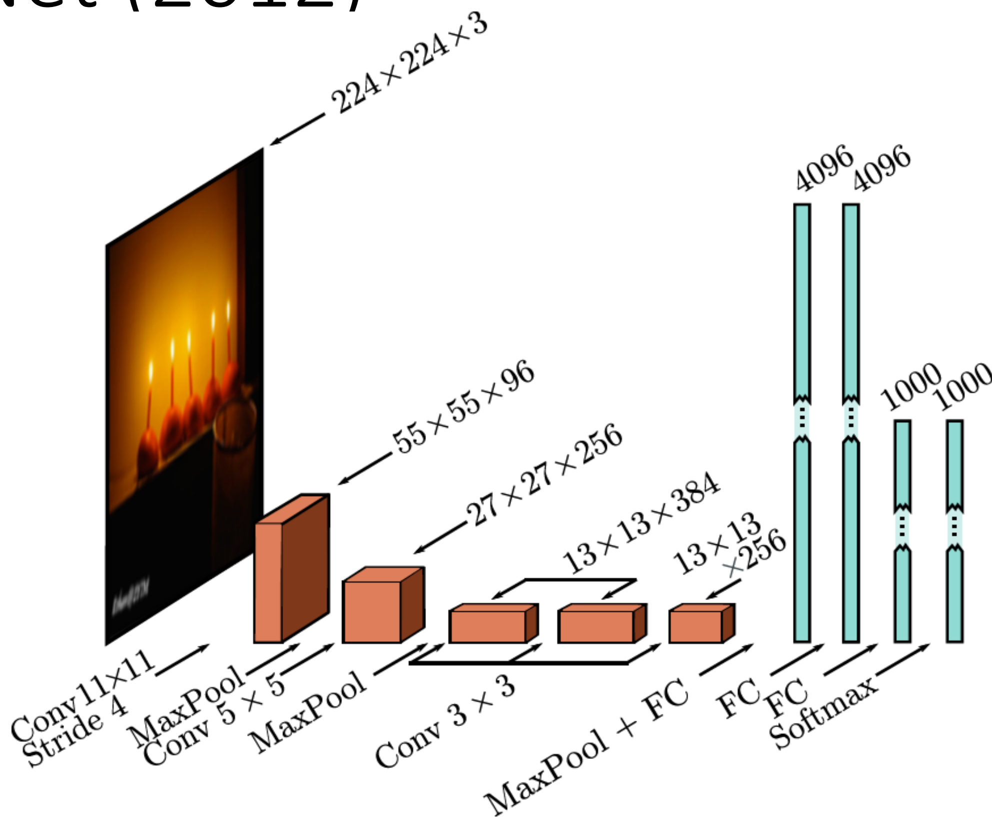
Fei-Fei Li





- 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

A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2012, doi: 10.1145/3065386.

AlexNet (2012)

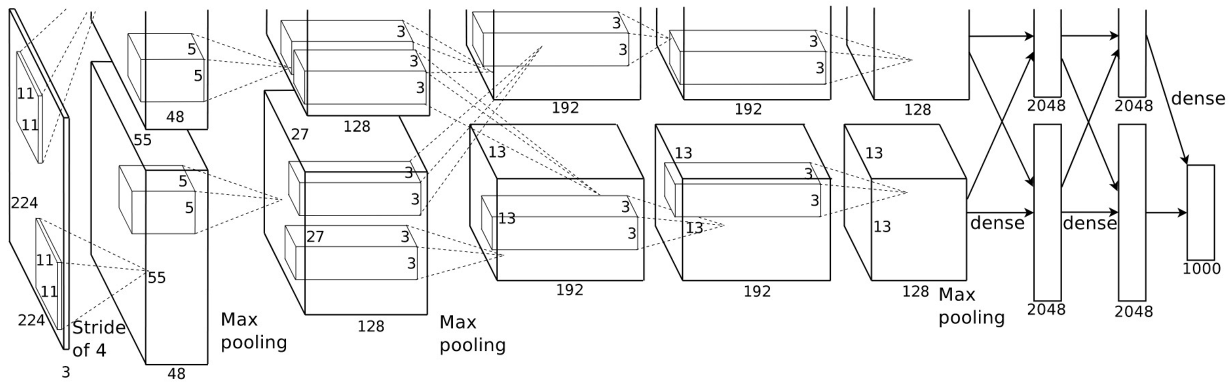


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Won the 2012 Large-Scale Vision Recognition Challenge (ILSVRC) by a big margin.

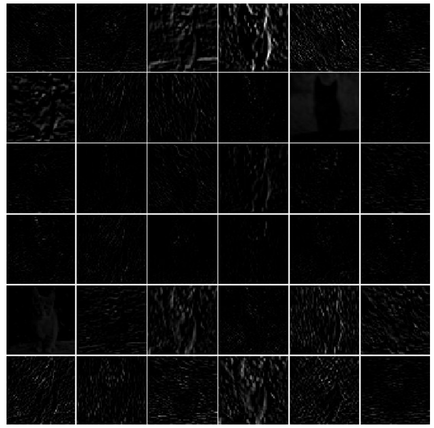
Took between five and six days to train on two GTX 580 3GB GPUs with manually optimized compute kernels.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2012, doi: 10.1145/3065386.

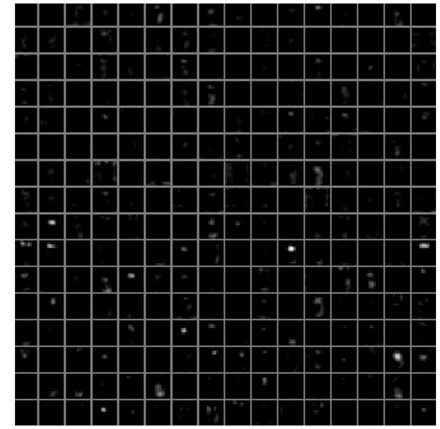
AlexNet

Cat image input
(not actual image)

Activations
(feature maps)



…

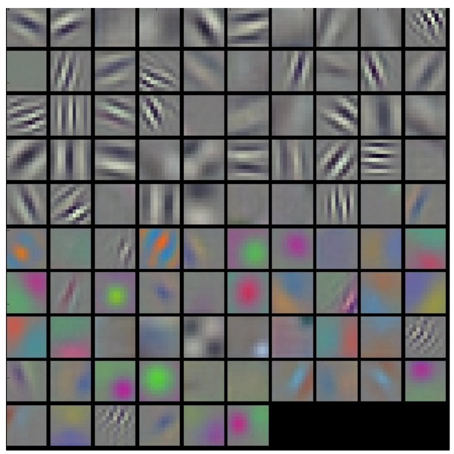


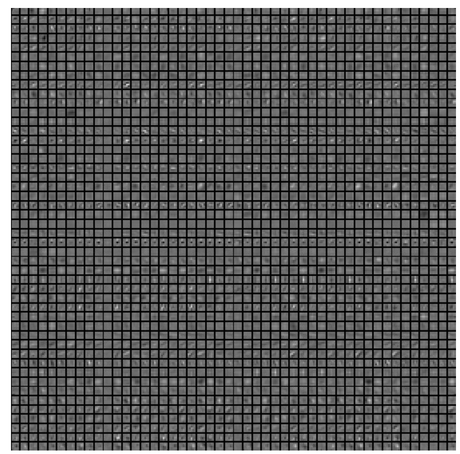
1st Layer

2nd Layer

5th Layer

Filter Kernels



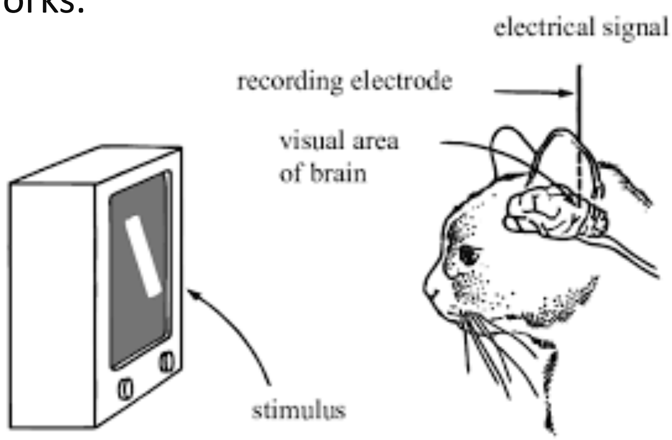


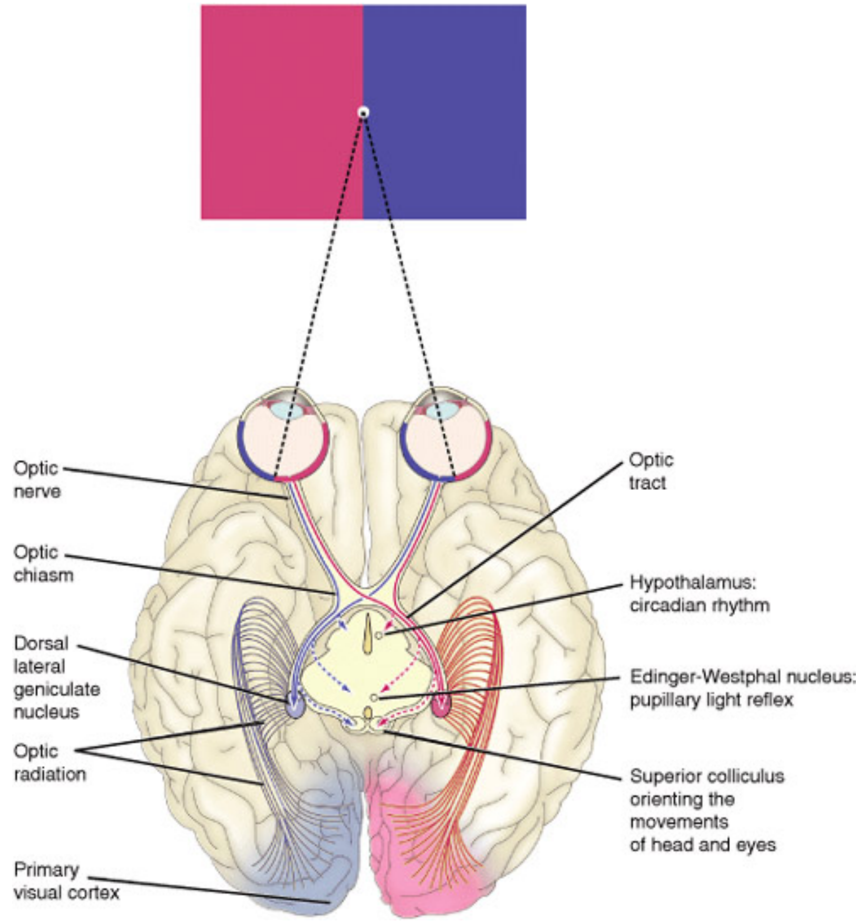
…

https://cs231n.github.io/understanding-cnn/

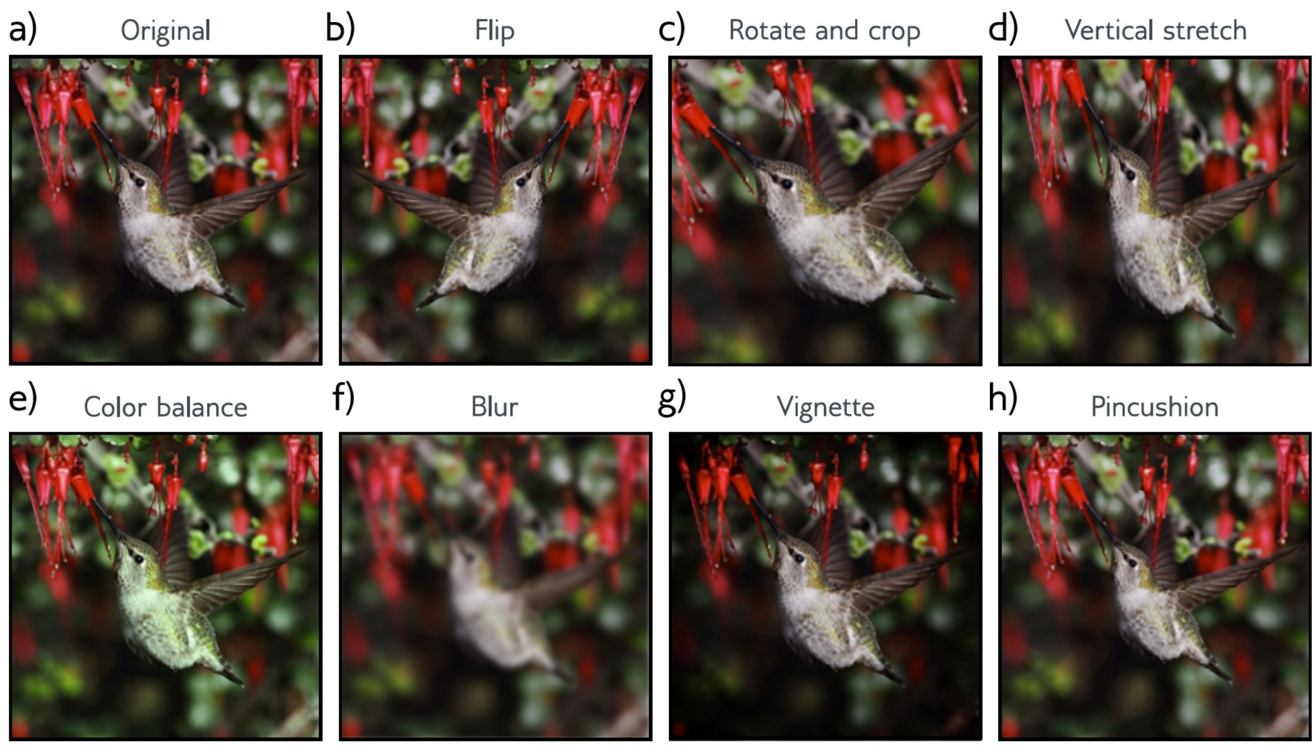
~1959: Hubel & Wiesel – Visual cortex and receptive fields

David Hubel and Torsten Wiesel discovered how neurons in the visual cortex respond to specific patterns of light, such as edges and orientations. Their work on receptive fields provided key insights into hierarchical processing in vision, influencing the design of modern convolutional neural networks.



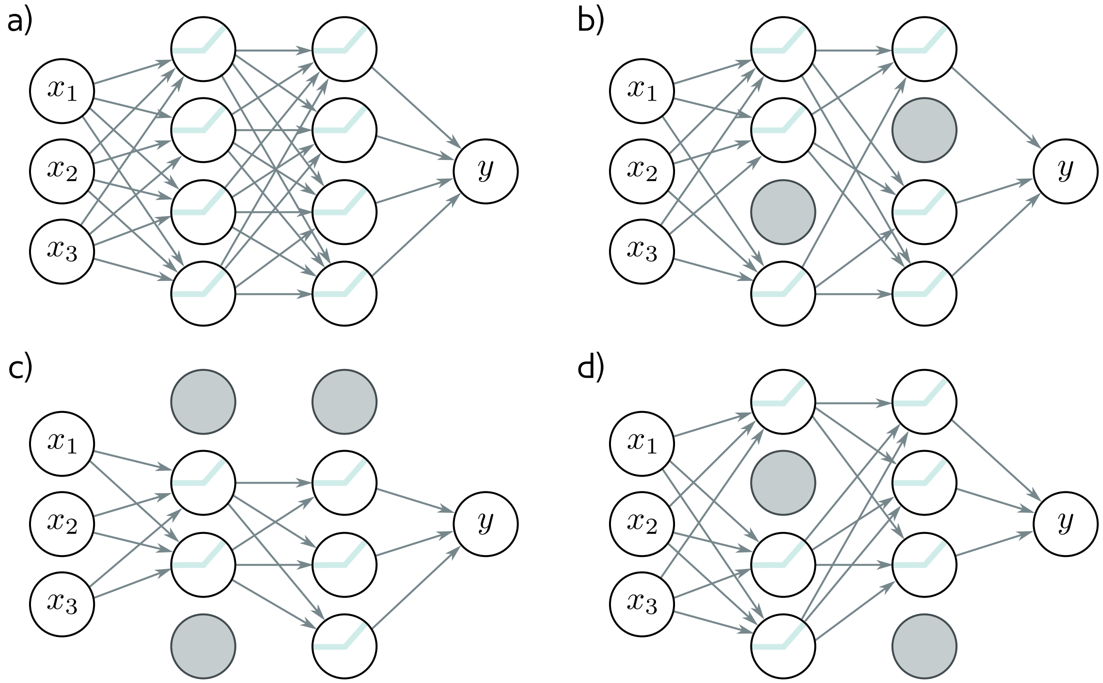


Data augmentation



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

Dropout

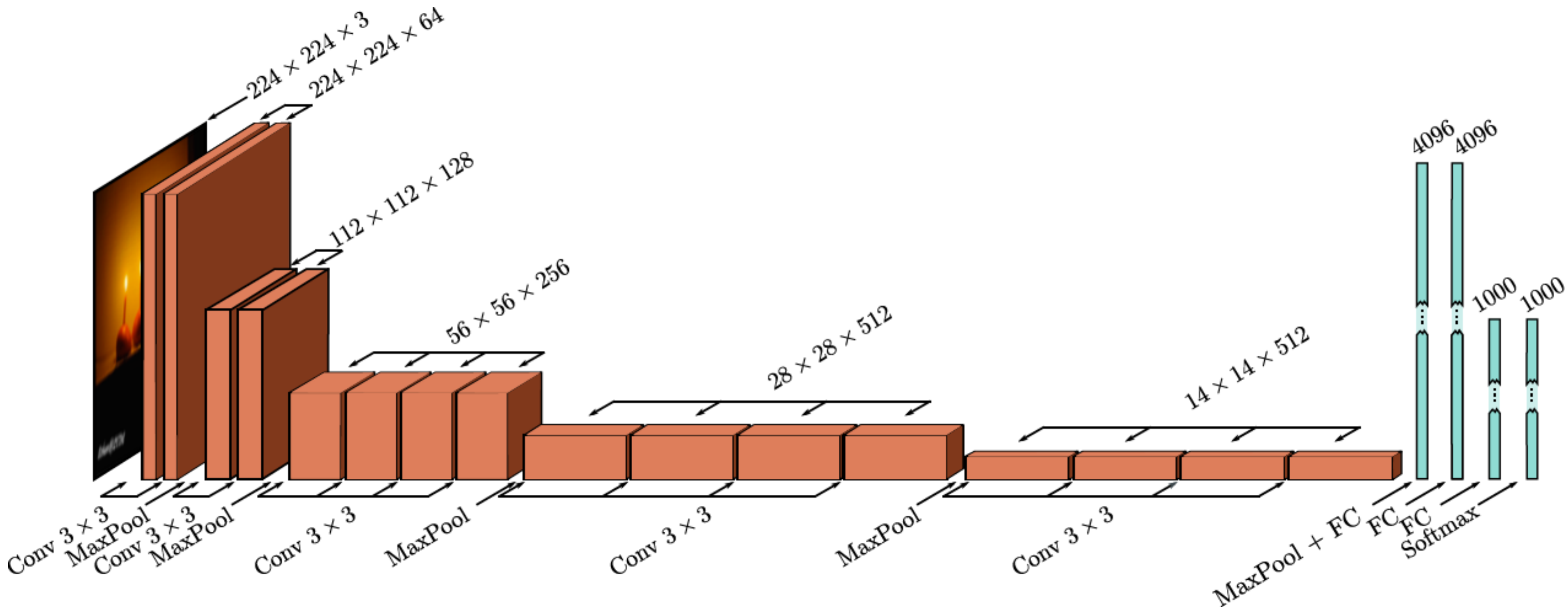


- Dropout was applied in the fully connected layers

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)

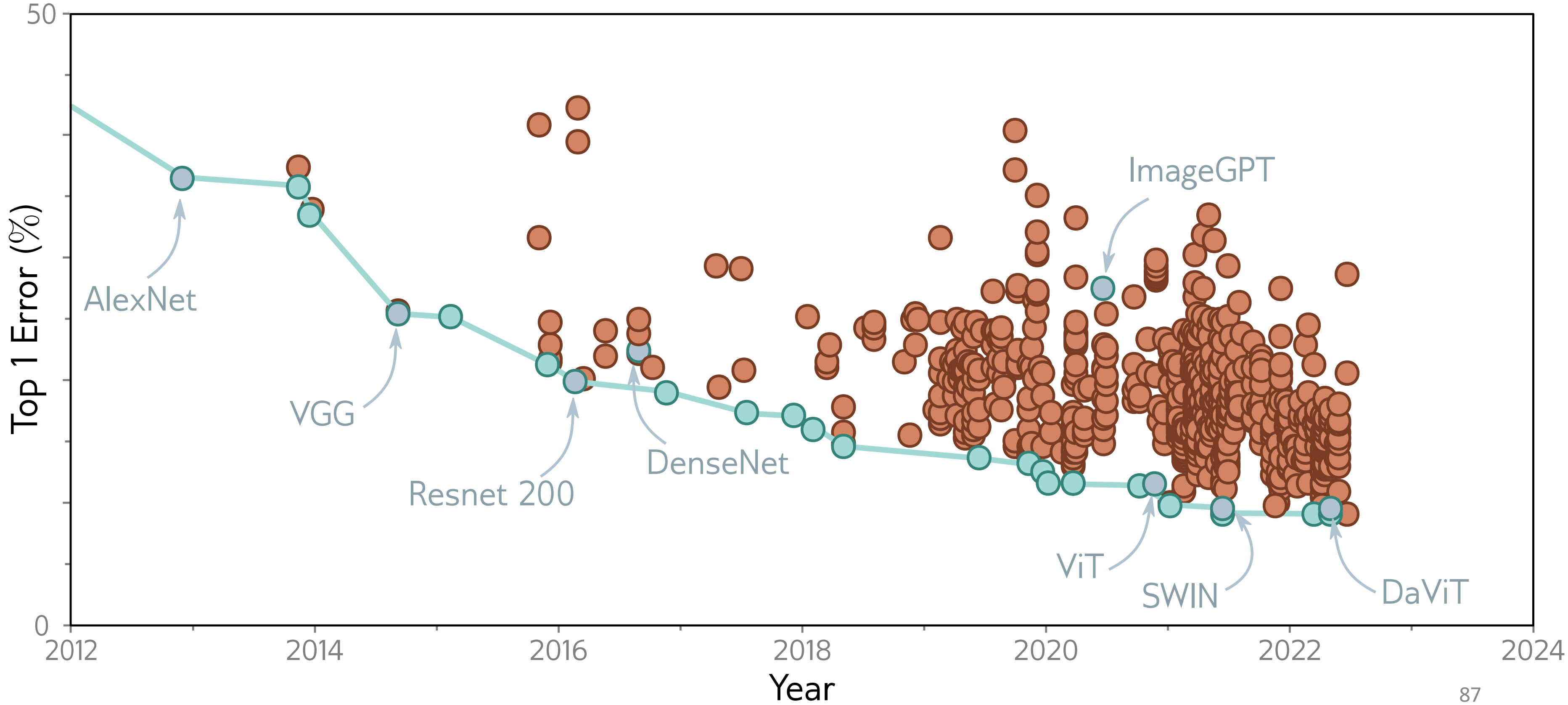


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
4096
4096
1000
1000
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool + FC
FC
FC
Softmax

Details

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

ImageNet History



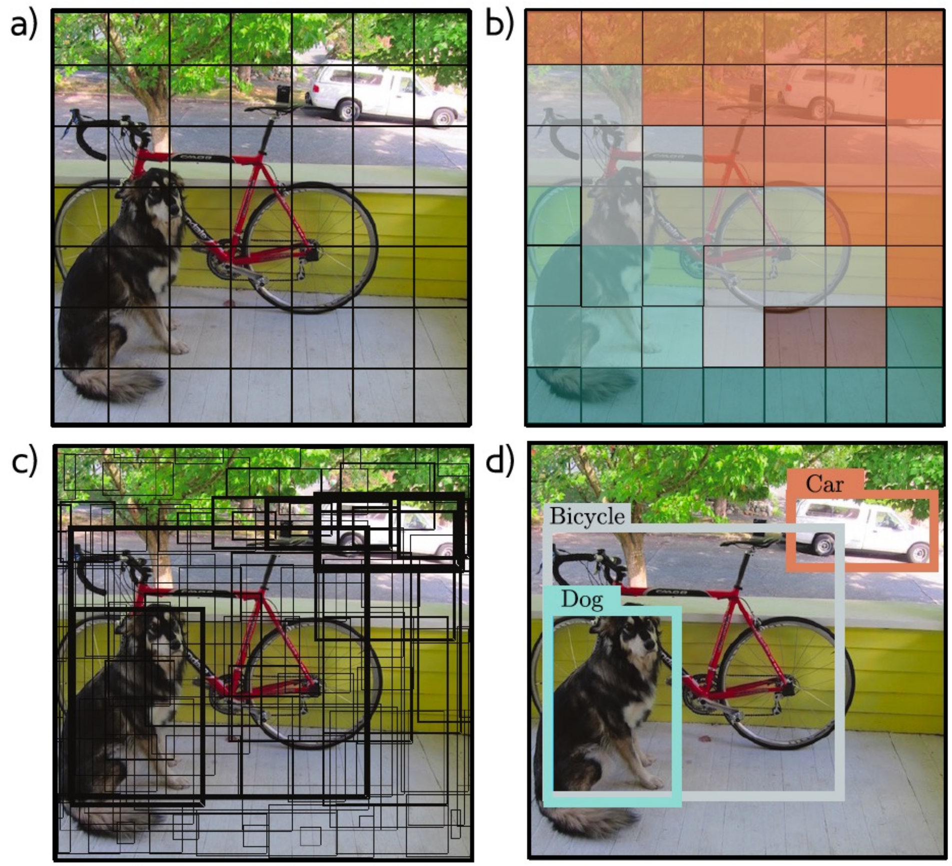
Convolution #2

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

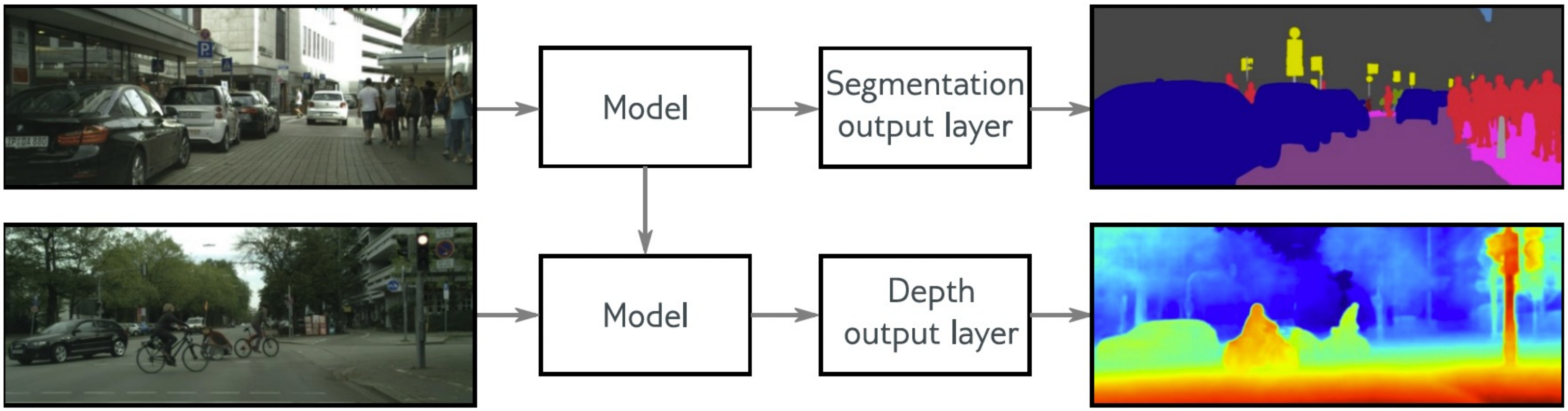
You Only Look Once (YOLO)

- 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 – (non maximum suppression)

Object detection (YOLO)

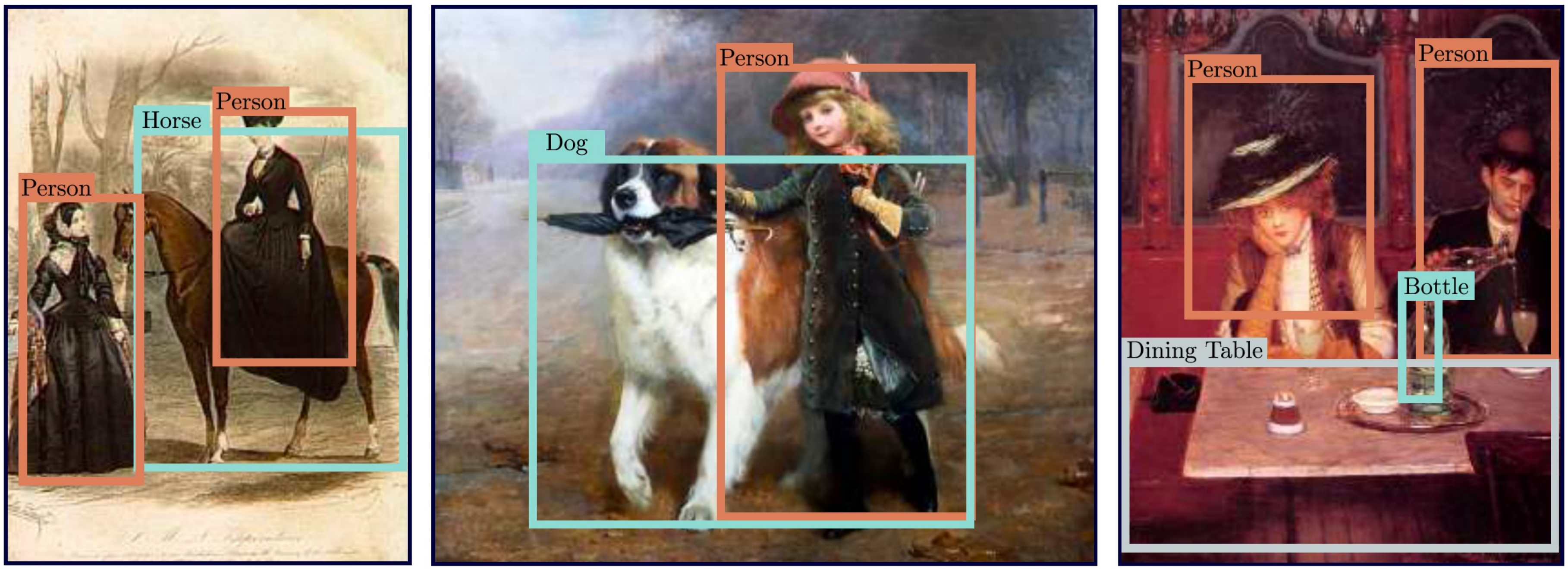


Transfer learning



Transfer learning from ImageNet classification

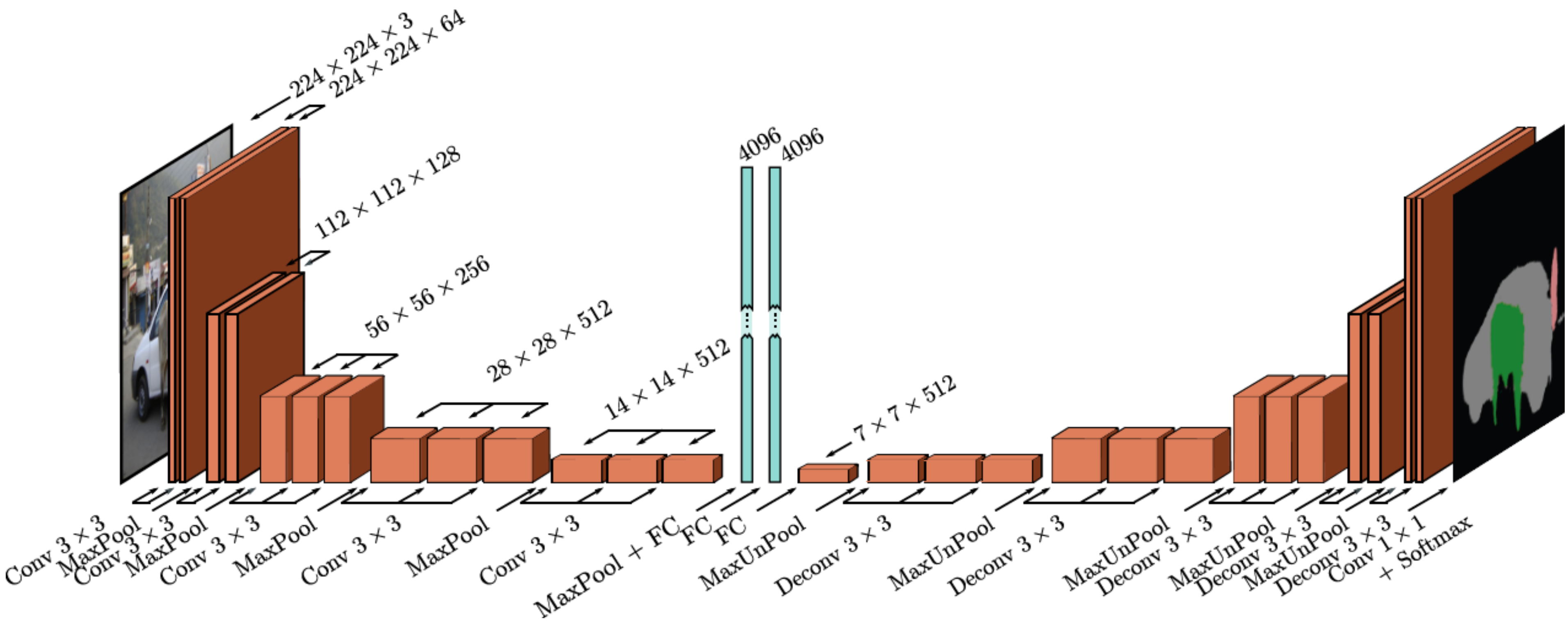
Results


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

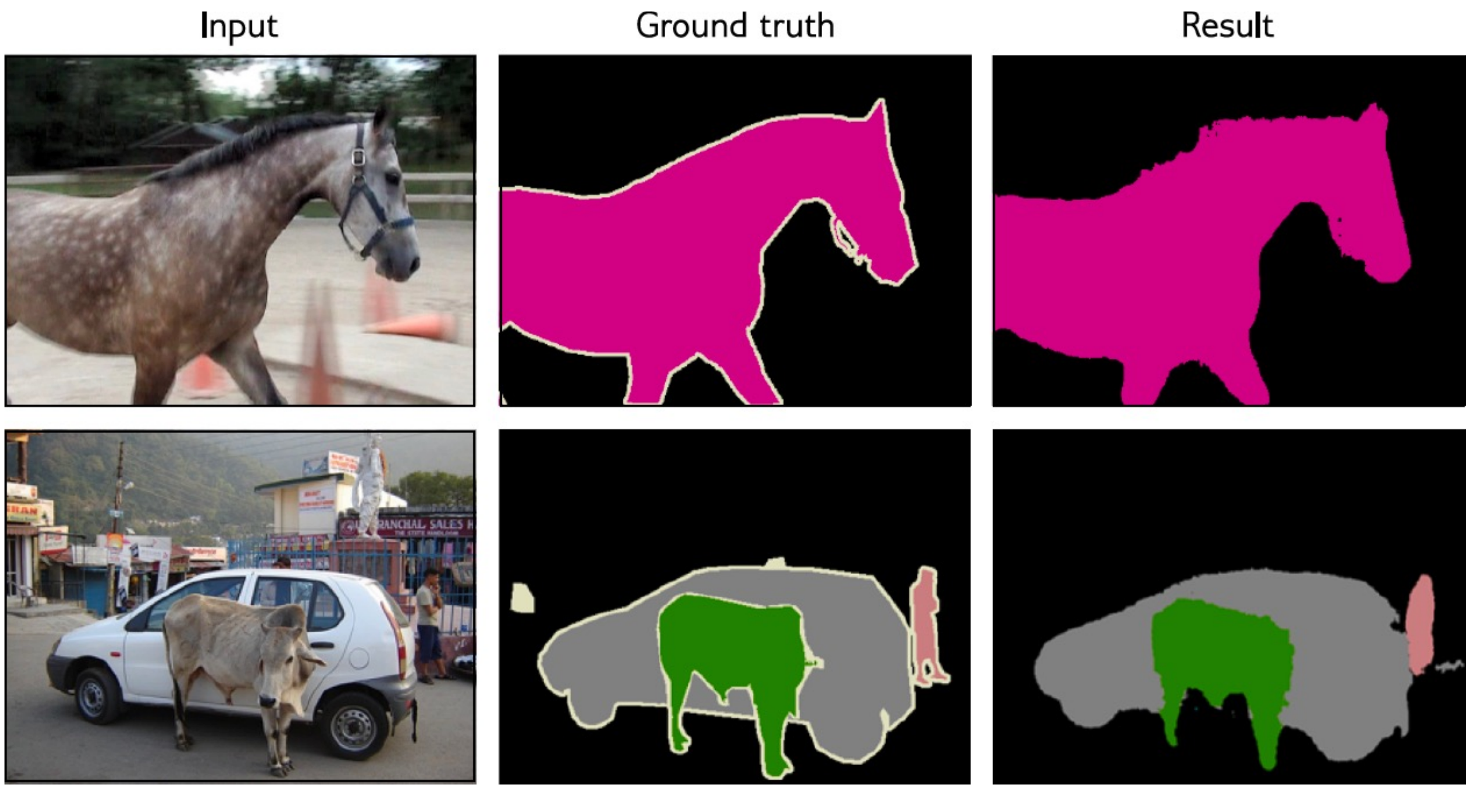
Semantic Segmentation (2015)

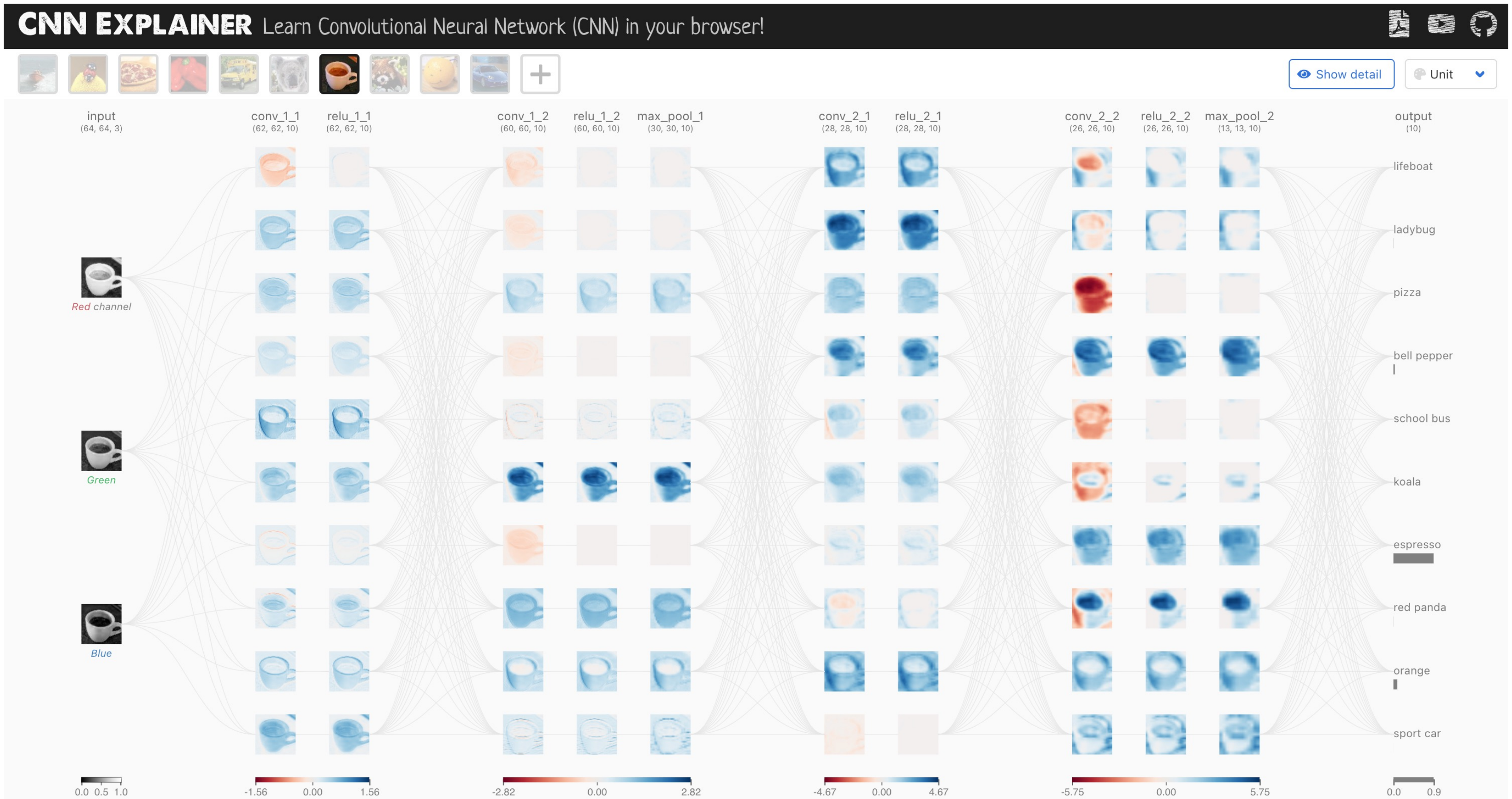


Encoder

Decoder

Semantic segmentation results





https://poloclub.github.io/cnn-explainer/

Feedback?

Link