

Deep Learning for Data Science

DS 542

Lecture 10
Convolutional Networks

Challenges Processing Images, Audio, Text, Video...

- Much bigger inputs?
- Variable size inputs?
- But some obvious structure to leverage?

Bigger Inputs

Original image size:

2560x1707x3=

13,109,760 values

Original image: kpmb.com

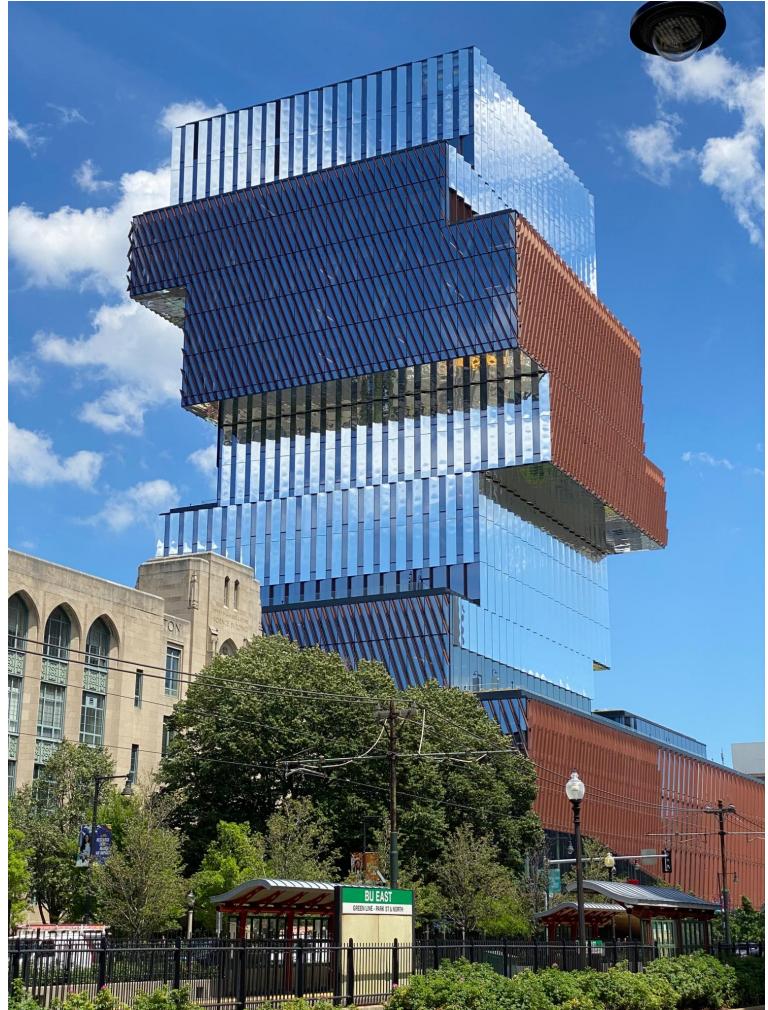


Variable Size

Original size: 3024x4032x3=
36,578,304 values

Vs previous 2560x1707x3
(not even same ratio)





Structure

←Original

3024x4032x3



Shrunk $\frac{1}{8}$ →

378x504x3

Data augmentation (from last time)

a) Original



b) Flip



c) Rotate and crop



d) Vertical stretch



e) Color balance



f) Blur



g) Vignette



h) Pincushion



Convolutional networks

Our first useful approach to these problems with large but structured inputs.

- Much bigger inputs? 
- Variable size inputs?  but often can rescale as workaround
- But some obvious structure to leverage? 

Spoiler: recognizing structure helps learn with bigger inputs.

Convolutional networks

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

Image classification

Real world input



Model
input

$$\begin{bmatrix} 124 \\ 140 \\ 156 \\ 128 \\ 142 \\ 157 \\ \vdots \end{bmatrix}$$

Model



Model
output

$$\begin{bmatrix} 0.00 \\ 0.00 \\ 0.01 \\ 0.89 \\ 0.05 \\ 0.00 \\ \vdots \\ 0.01 \end{bmatrix}$$

Real world output

Aardvark
Apple
Bee
Bicycle
Bridge
Clown
⋮

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

Object detection

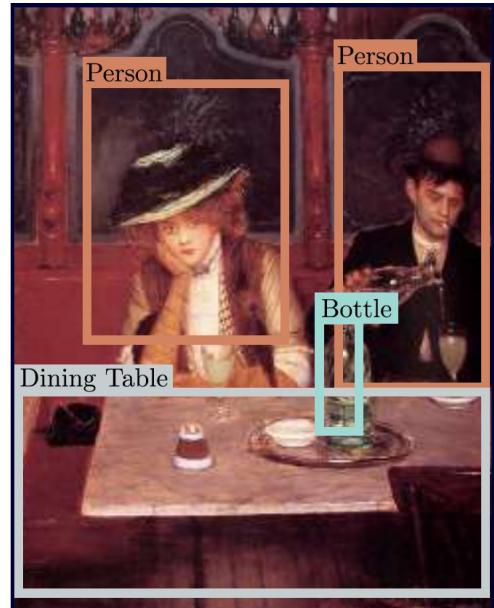
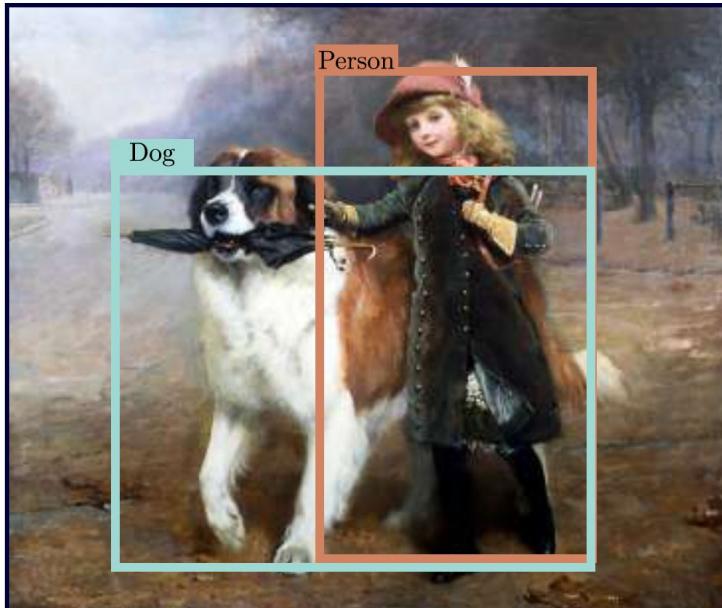
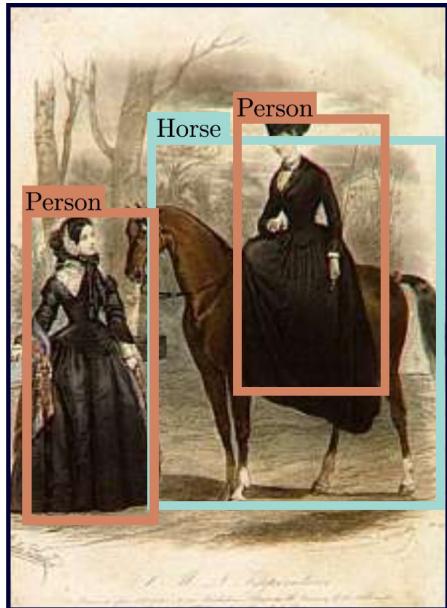
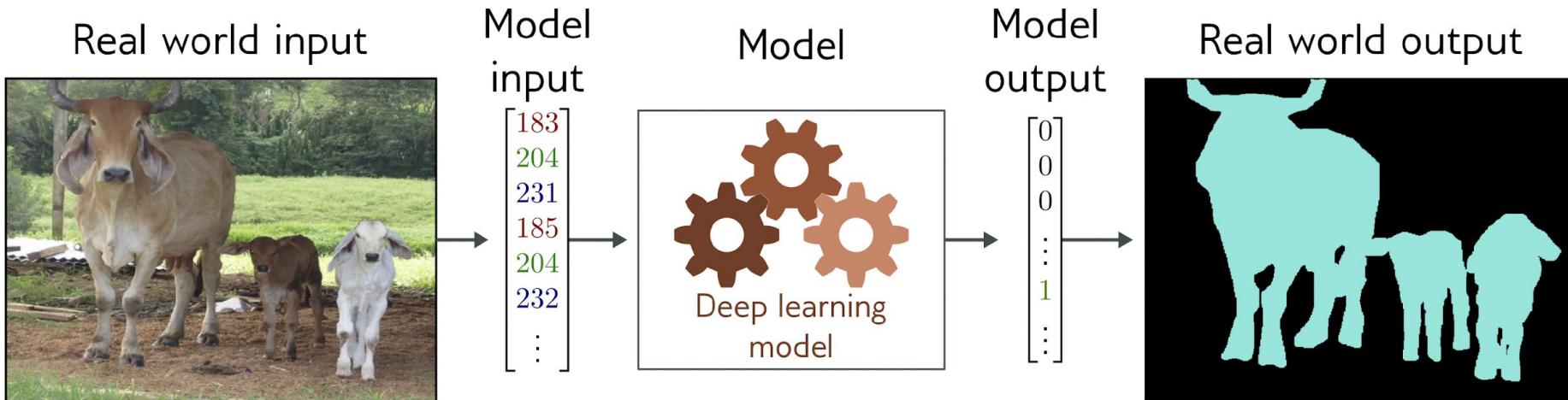


Image segmentation



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

Networks for images

Problems with fully-connected networks

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

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 = $150,520 \times 150,528$ 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

Opportunities for regularization?

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:

$$f[t[x]] = f[x]$$

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:

$$f[t[x]] = t[f[x]]$$

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:

- Convolutional **kernel** or **filter**:

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

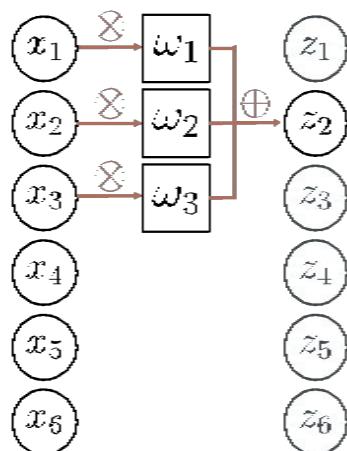
$$\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T$$

Kernel size =
3

* Not really technically
convolution

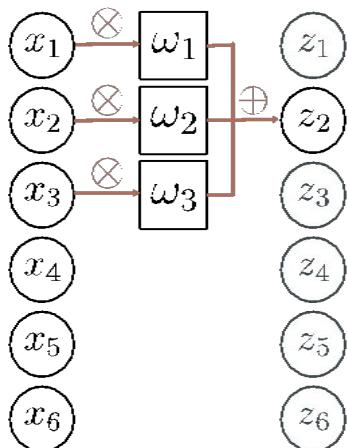
Convolution with kernel size 3

a)

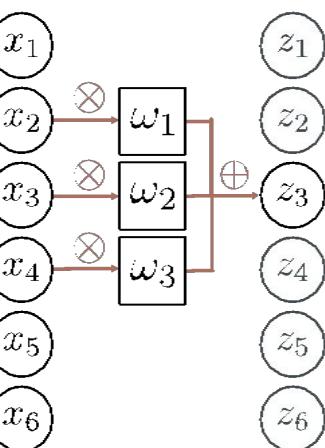


Convolution with kernel size 3

a)

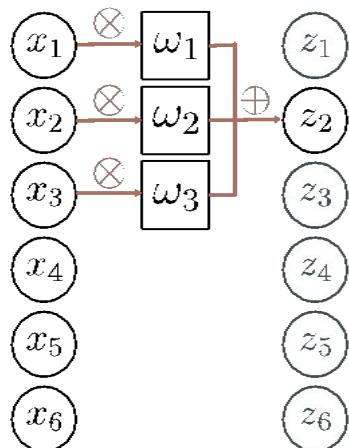


b)

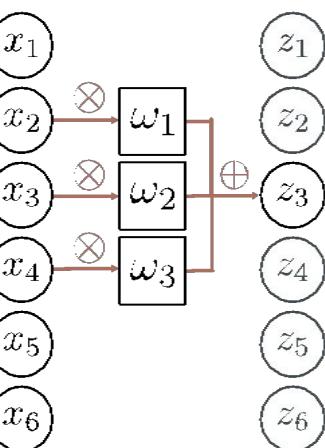


Convolution with kernel size 3

a)



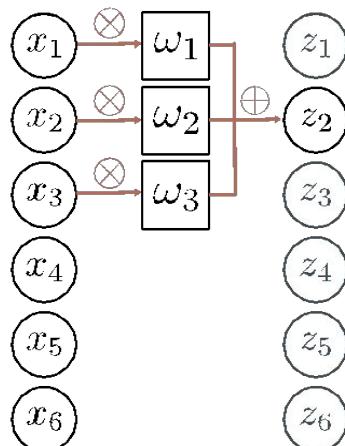
b)



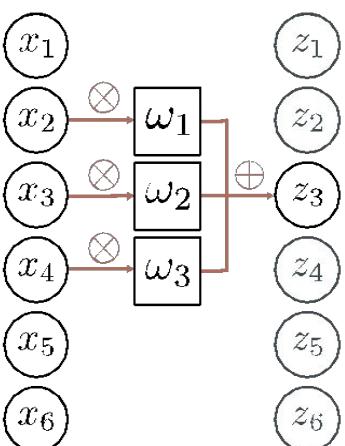
Equivariant to translation of
input $\mathbf{f}[\mathbf{t}[\mathbf{x}]] = \mathbf{t}[\mathbf{f}[\mathbf{x}]]$

Zero padding

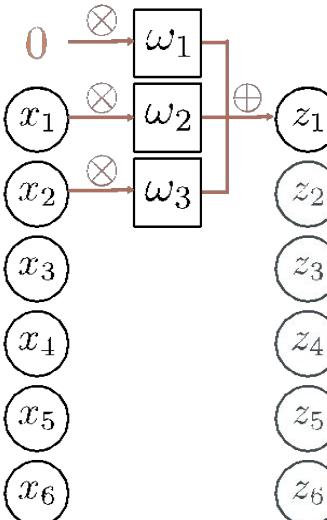
a)



b)

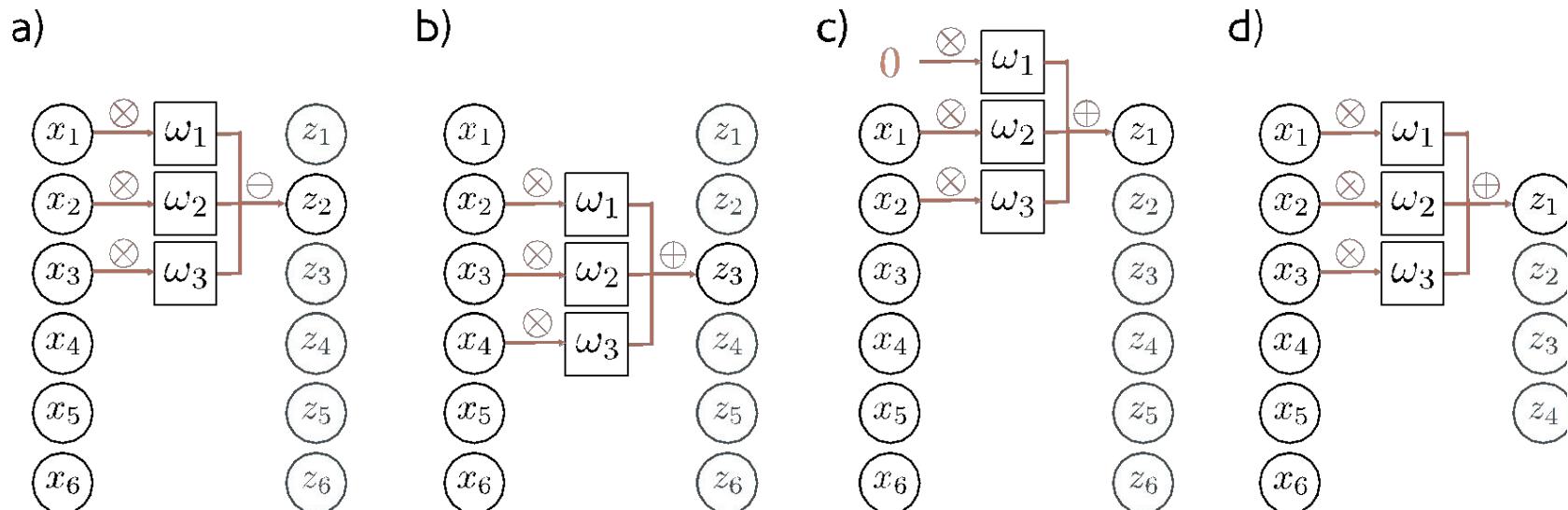


c)



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

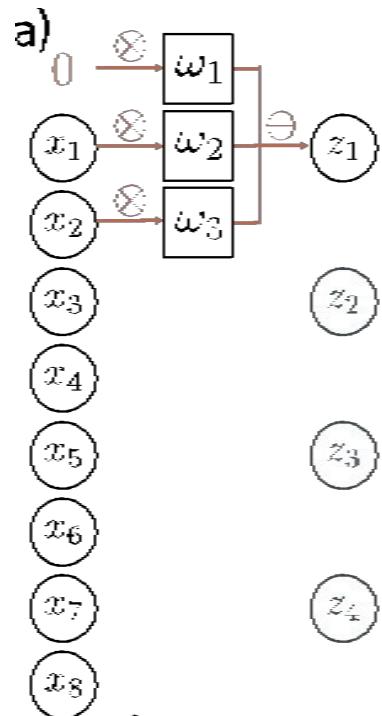
“Valid” convolutions



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

Stride, kernel size, and dilation

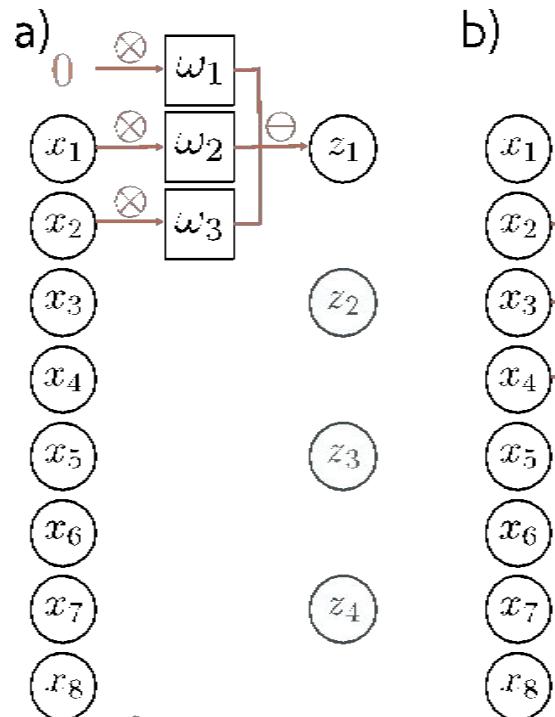
- **Stride** = shift by k positions for each output
 - Decreases size of output relative to input
- **Kernel size** = weight a different number of inputs for each output
 - Combine information from a larger area
 - But kernel size 5 uses 5 parameters
- **Dilated or atrous convolutions** = intersperse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



Size = 3

Stride = 2

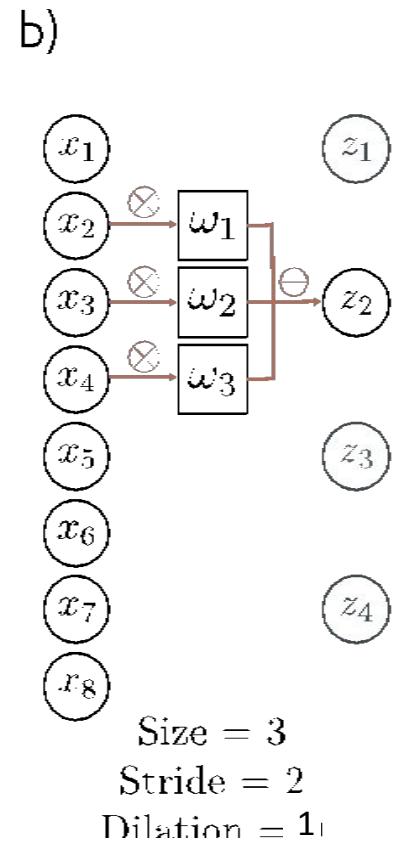
Dilation = 1

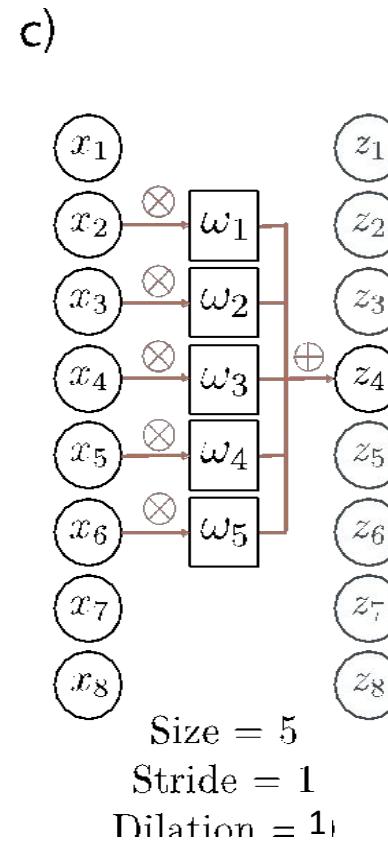
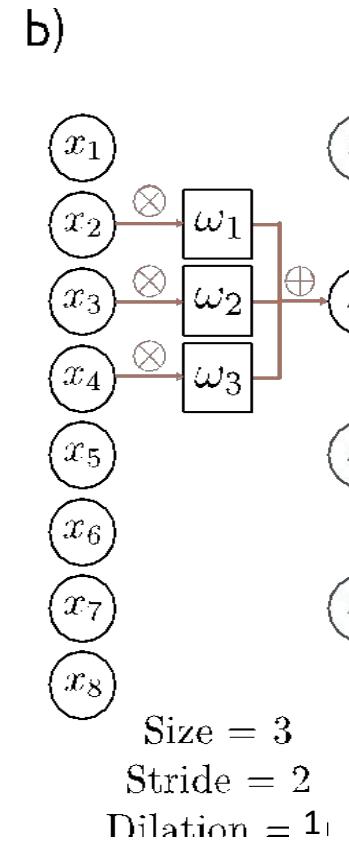
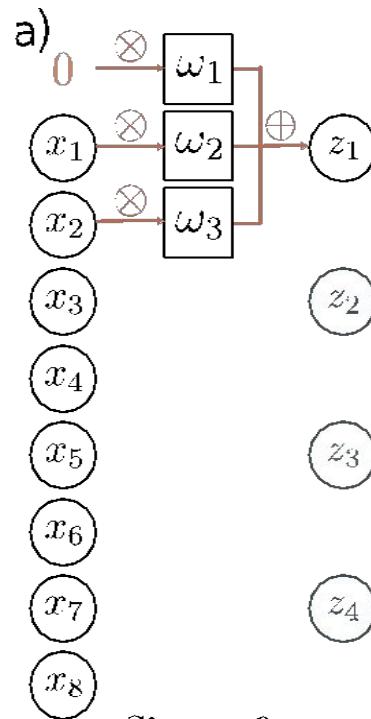


Size = 3

Stride = 2

Dilation = 1





Size = 3

Stride = 2

Dilation = 1

Size = 3

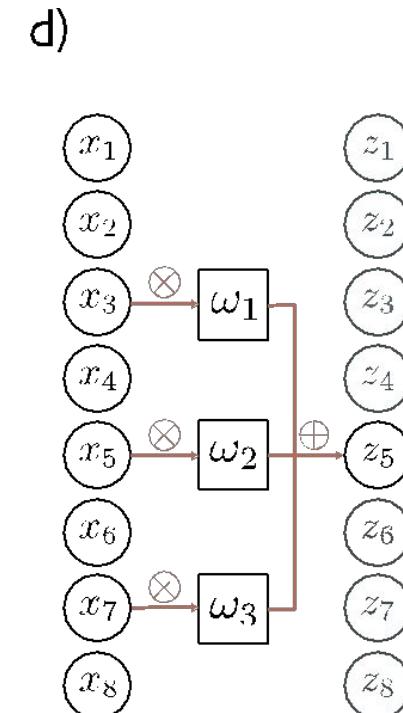
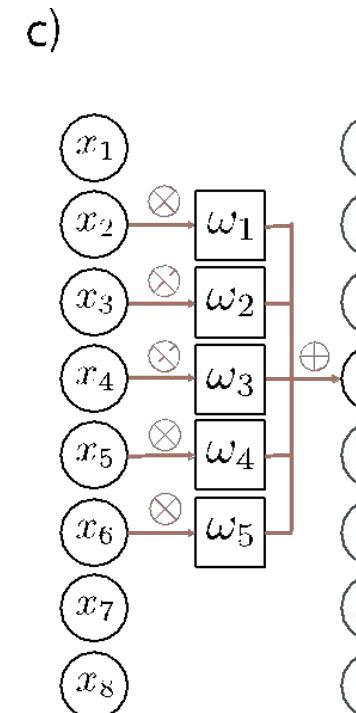
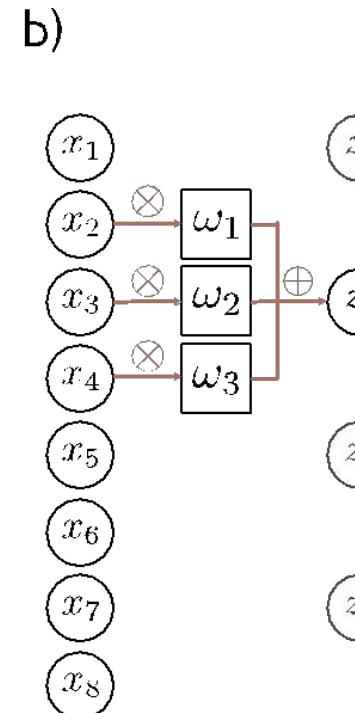
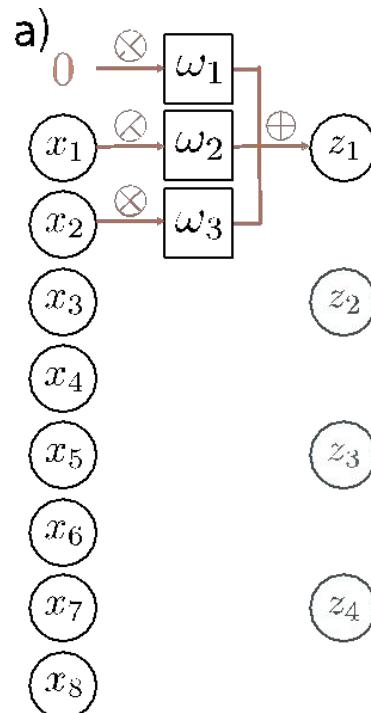
Stride = 2

Dilation = 1

Size = 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

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

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

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

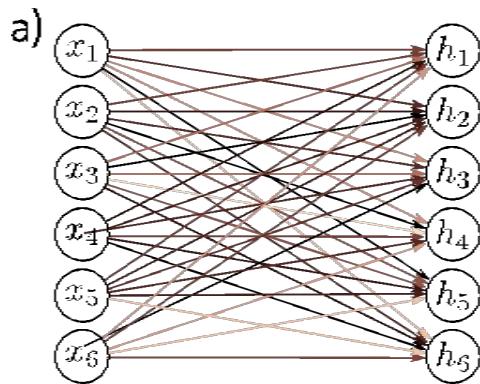
Fully connected
network:

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



D^2 weights, D biases

Special case of fully-connected network



b)

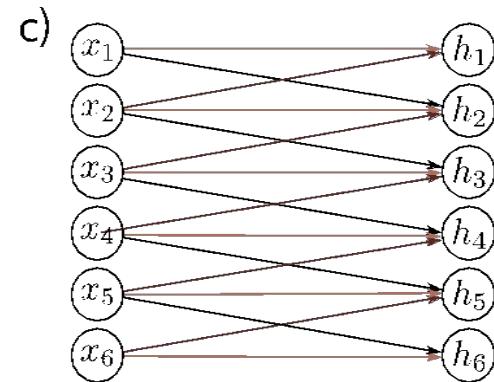
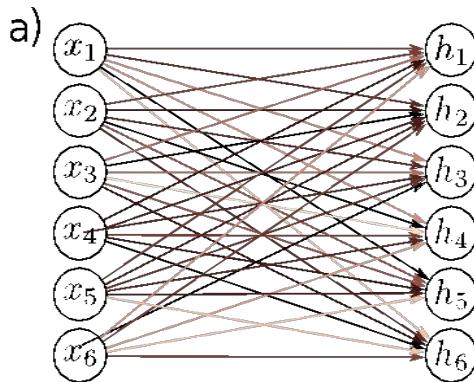
	x_1	x_2	x_3	x_4	x_5	x_6
h_1						
h_2						
h_3						
h_4						
h_5						
h_6						

Weight Matrix

Fully connected
network

Bias is implied

Special case of fully-connected network



b)

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

Fully connected
network

d)

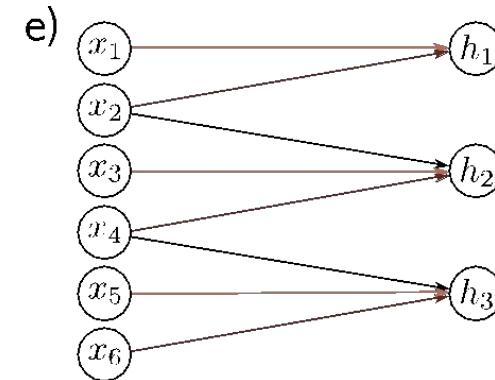
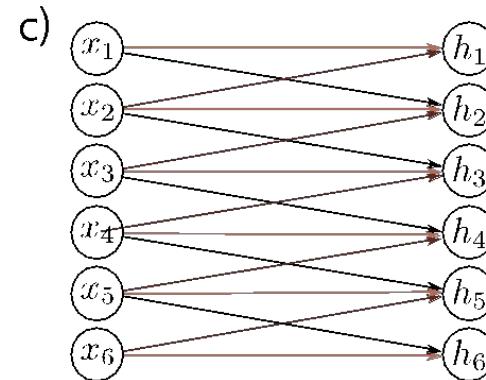
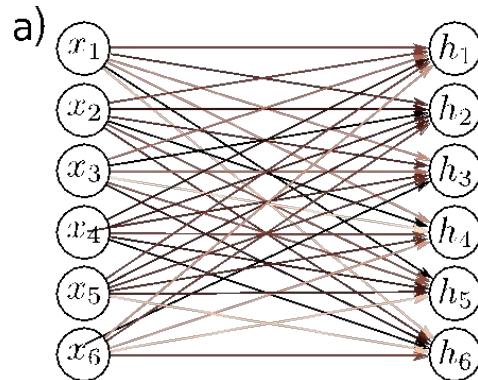
	x_1	x_2	x_3	x_4	x_5	x_6
h_1	Light Brown	Dark Brown	White	White	White	White
h_2	Black	White	Light Brown	Dark Brown	White	White
h_3	White	Light Brown	Dark Brown	Dark Brown	White	White
h_4	White	White	White	Light Brown	Dark Brown	Dark Brown
h_5	White	White	White	Black	Light Brown	Dark Brown
h_6	White	White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

Convolution, kernel 3,
stride 1, dilation 1

Bias is implied

Special case of fully-connected network

Bias is implied



Weight Matrices

b)

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

Fully connected
network

d)

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

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

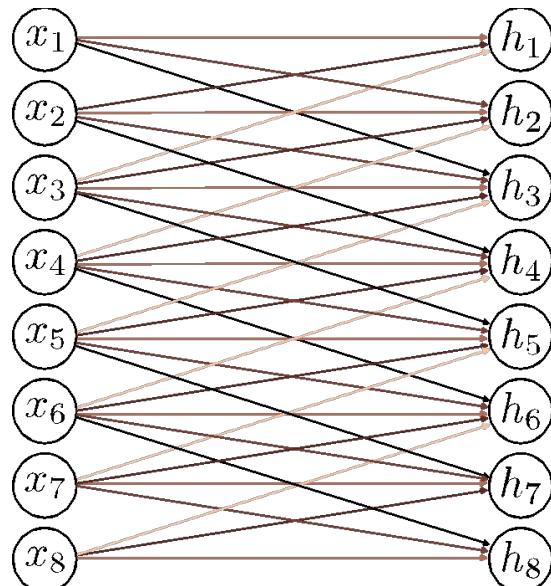
f)

	x_1	x_2	x_3	x_4	x_5	x_6
h_1	Light Brown	Dark Brown	White	White	White	White
h_2	White	Black	Light Brown	Dark Brown	White	White
h_3	White	White	White	Black	Light Brown	Dark Brown
	White	White	White	White	Black	Light Brown

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

Question 1

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

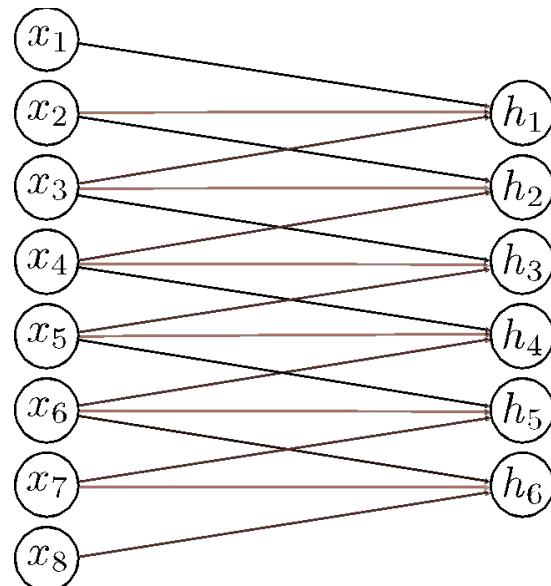


	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
h_1	■							
h_2		■						
h_3			■					
h_4				■				
h_5					■			
h_6						■		
h_7							■	
h_8								■

Bias is implied

Question 2

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

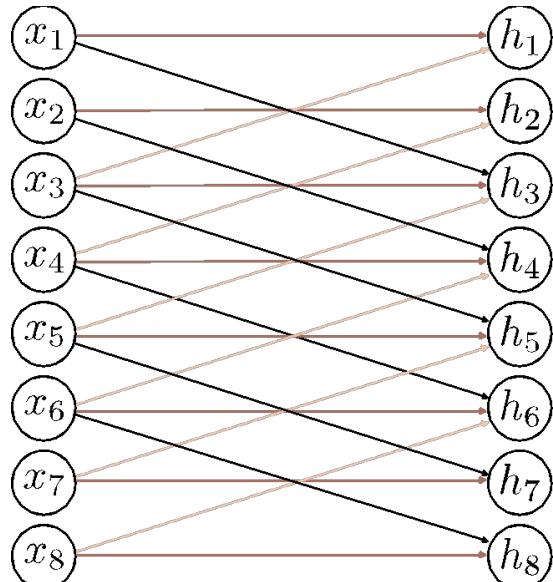


	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
h_1	Black	Brown	Brown					
h_2		Black	Brown	Brown				
h_3			Black	Brown	Brown			
h_4				Black	Brown			
h_5					Black	Brown		
h_6						Black	Brown	

Bias is implied

Question 3

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



Bias is implied

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
h_1	■							
h_2		■						
h_3			■		■		■	
h_4				■		■		■
h_5					■			■
h_6						■		■
h_7							■	■
h_8								■

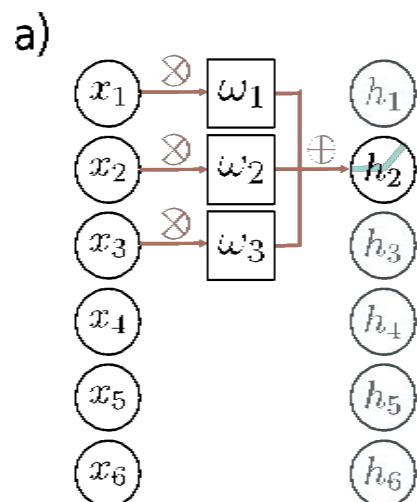
Convolutional networks

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

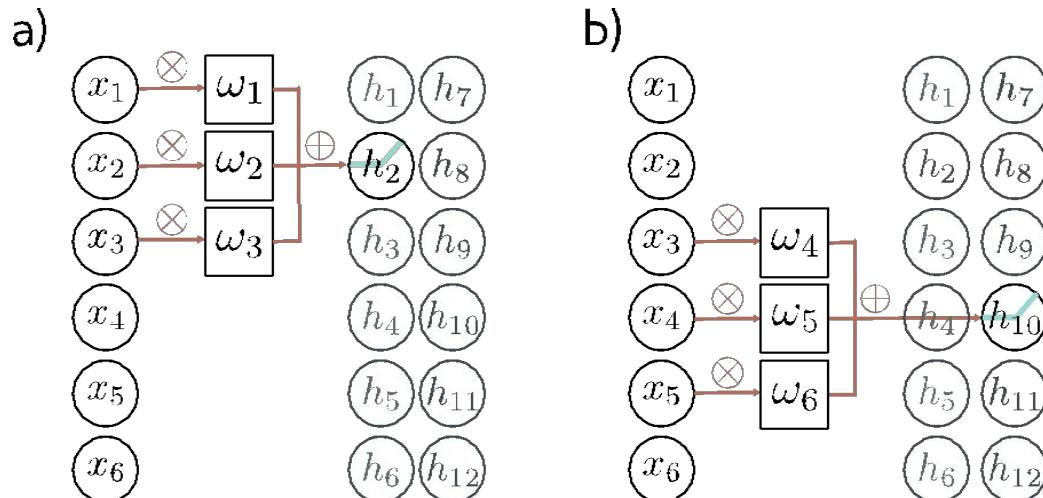
Channels

- 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**
 - Similar motivations to having multiple units in a hidden layer

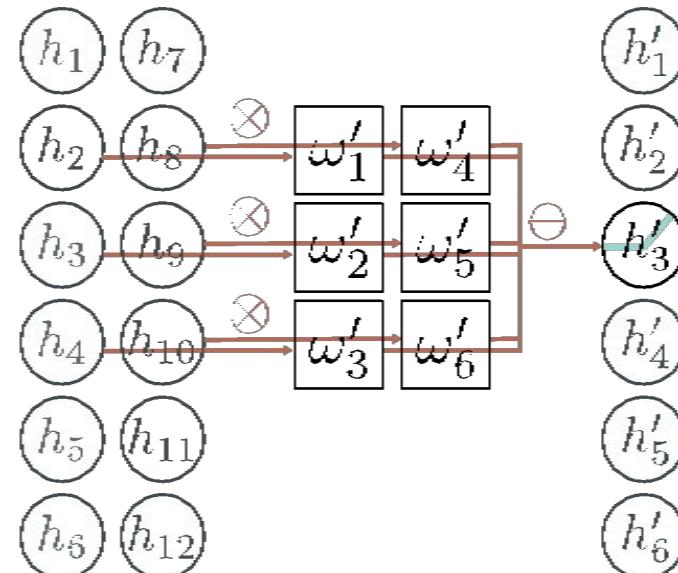
Two output channels, one input channel



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

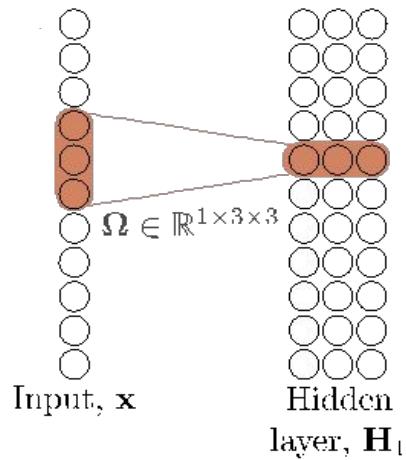
$$\Omega \in \mathbb{R}^{C_i \times C_o \times K} \quad \beta \in \mathbb{R}^{C_o}$$

Convolutional networks

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

Receptive fields

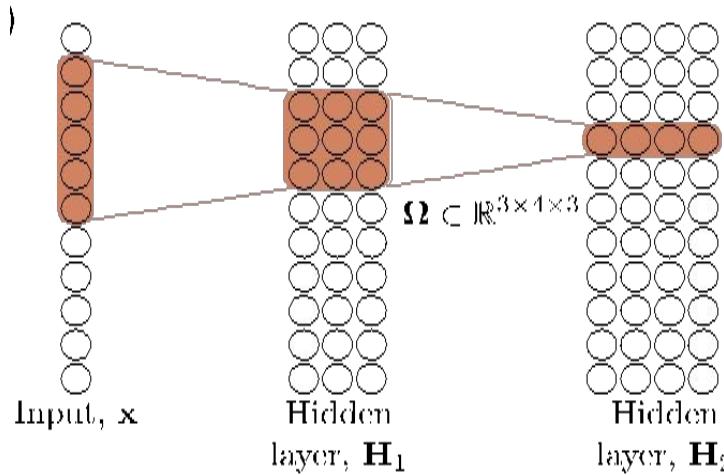
$$\mathbb{R}^{C_i \times C_o \times K}$$



Receptive fields

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

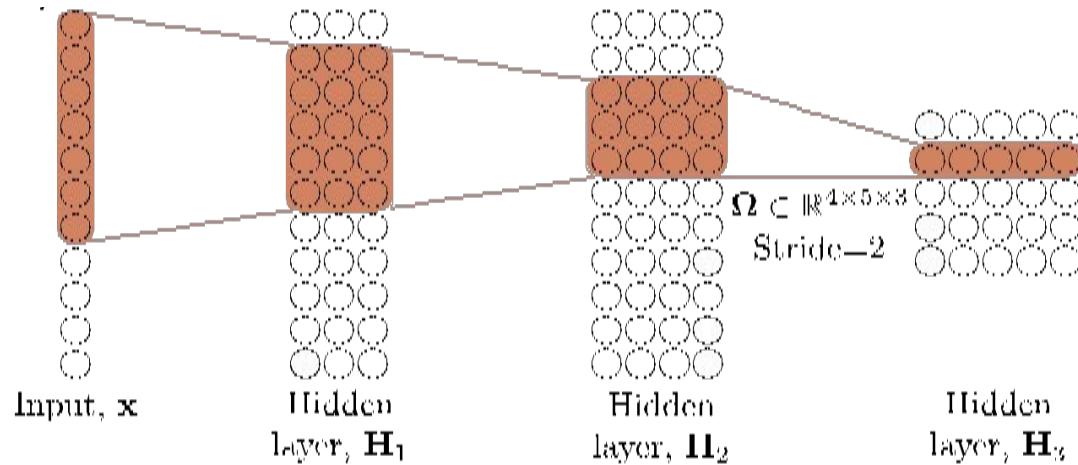
$$[\omega_1, \omega_2, \omega_3] \otimes [\omega_1, \omega_2, \omega_3] = [\omega_1, \omega_2, \omega_3, \omega_4, \omega_5]$$



Receptive fields

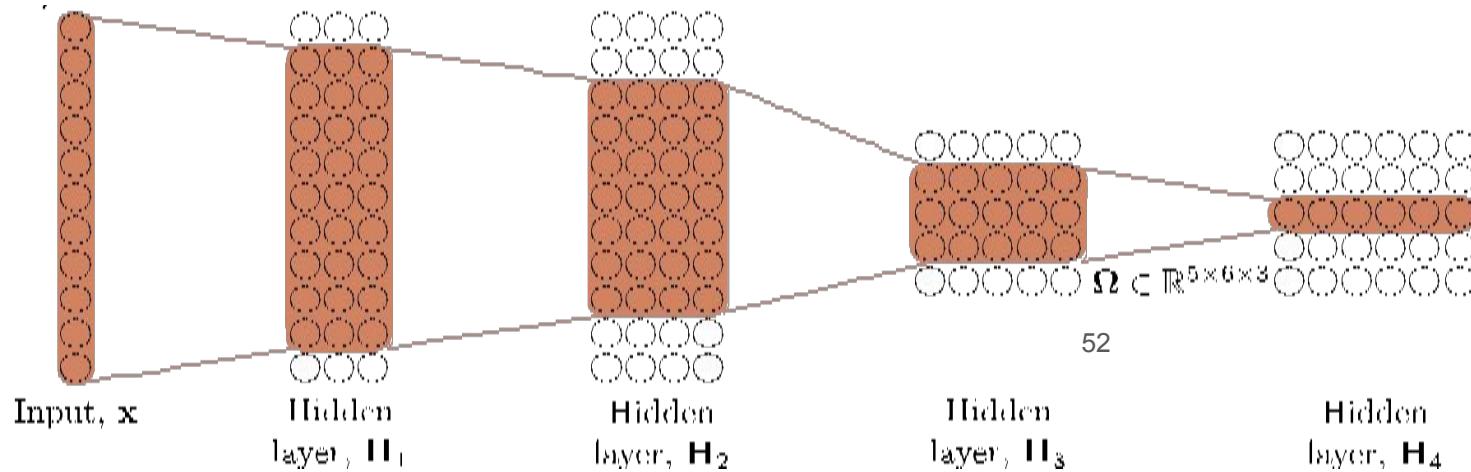
$$\mathbb{R}^{C_i \times C_o \times K}$$

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



Receptive fields

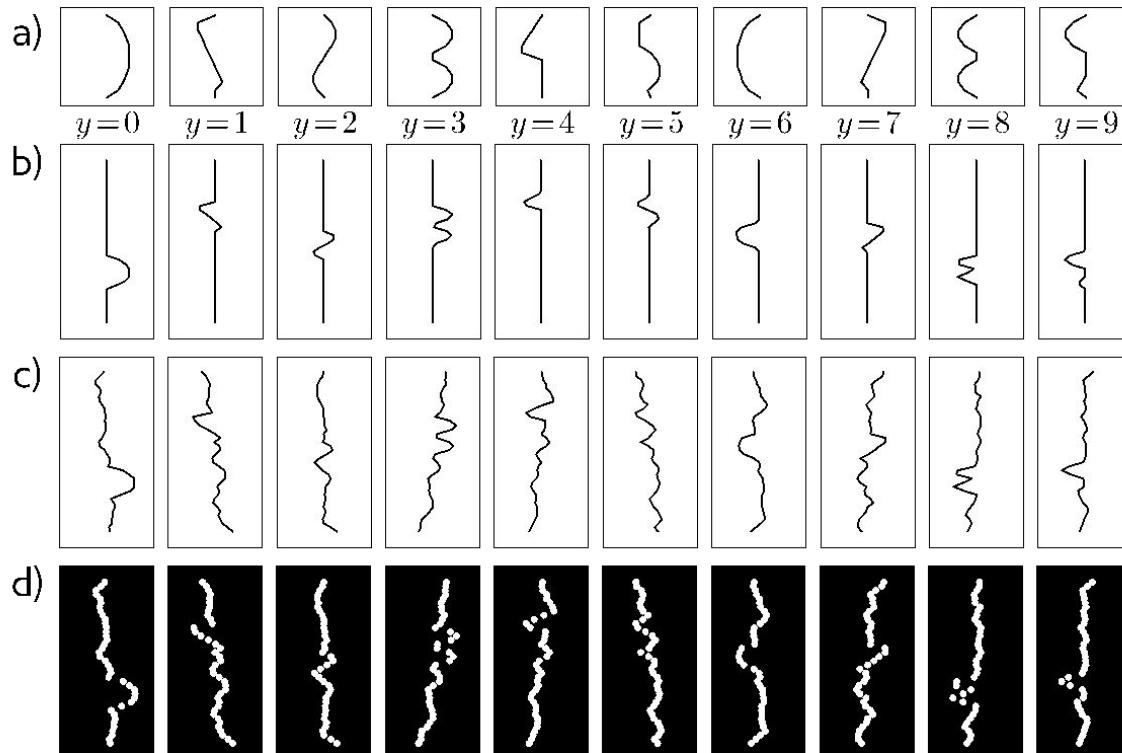
$$\mathbb{R}^{C_i \times C_o \times K}$$



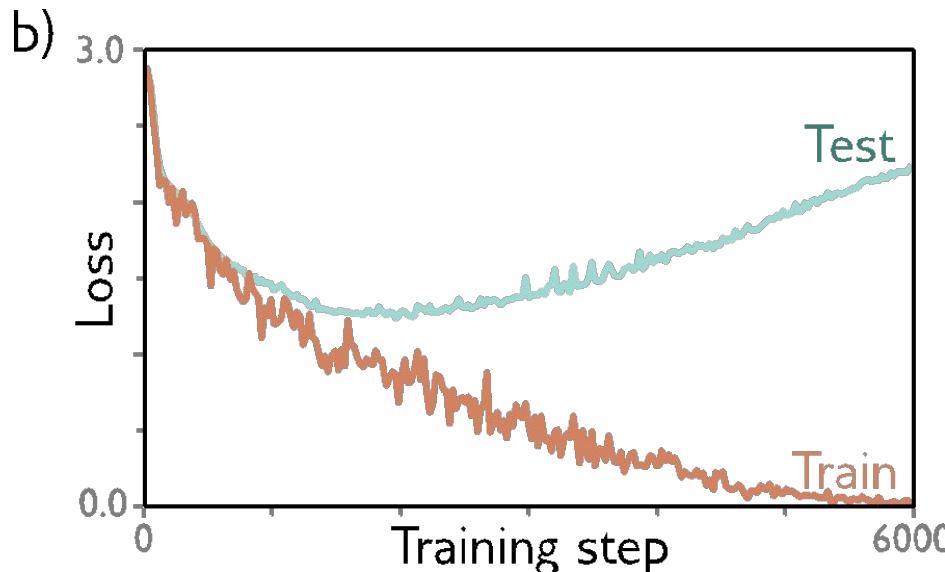
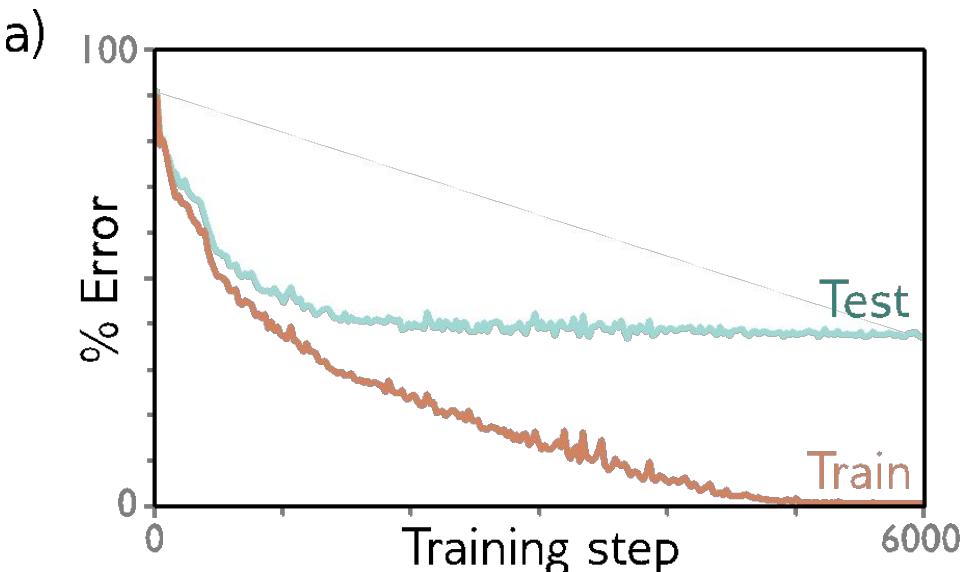
Convolutional networks

- 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



Fully connected network

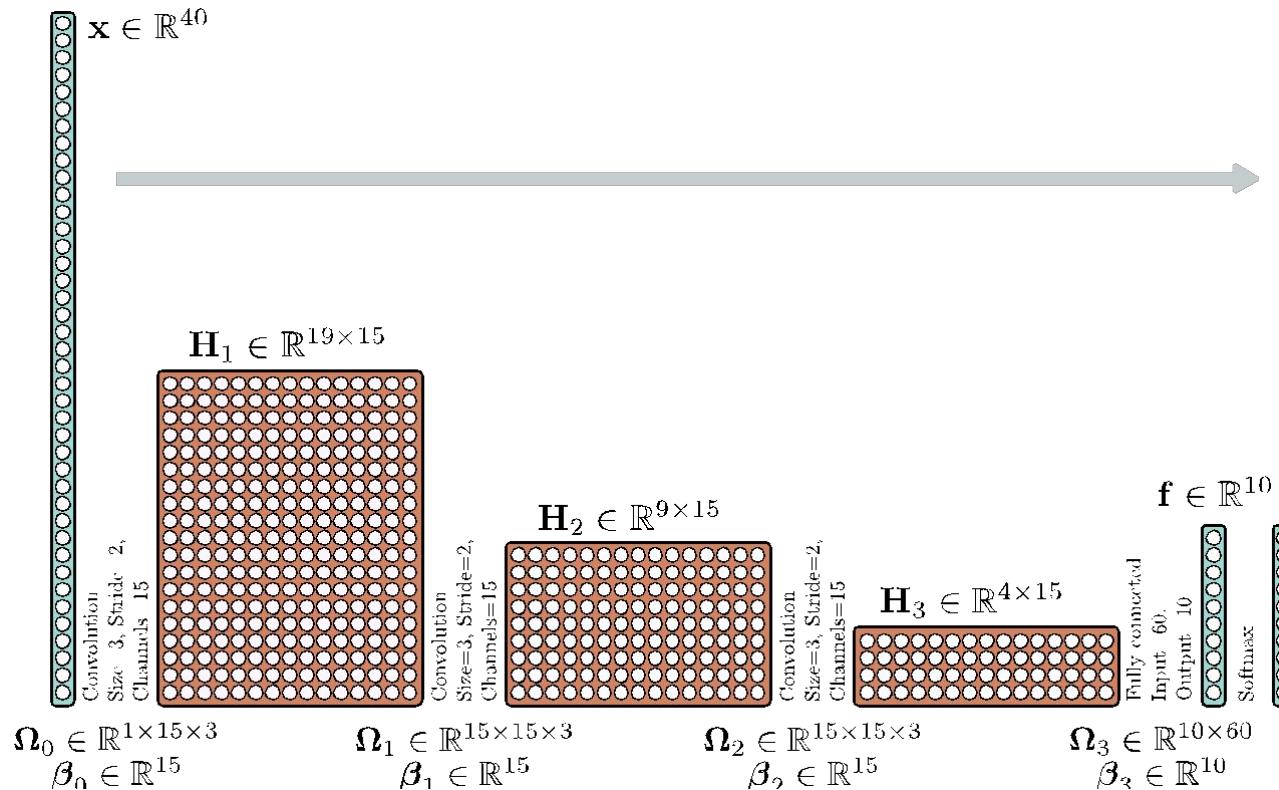
- Exactly same number of layers and hidden units
- All fully-connected layers
- Total parameters = 150,185

Convolutional network

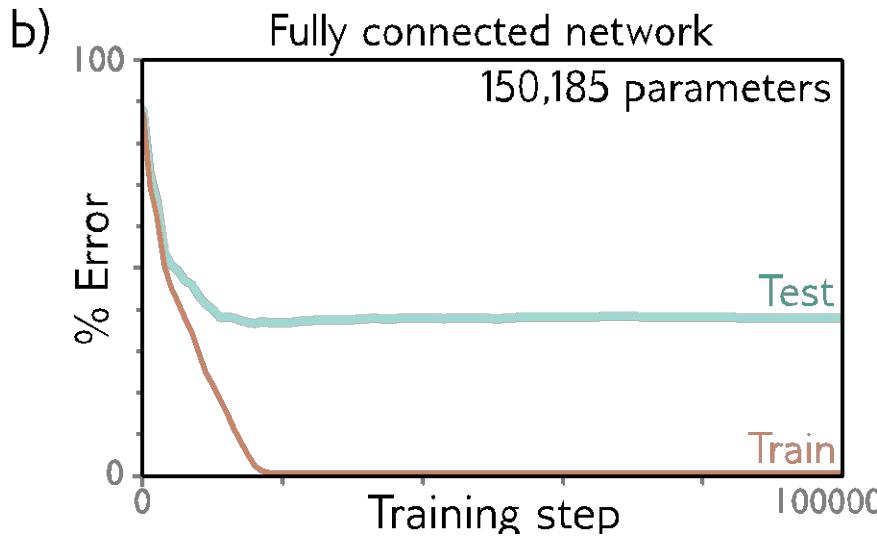
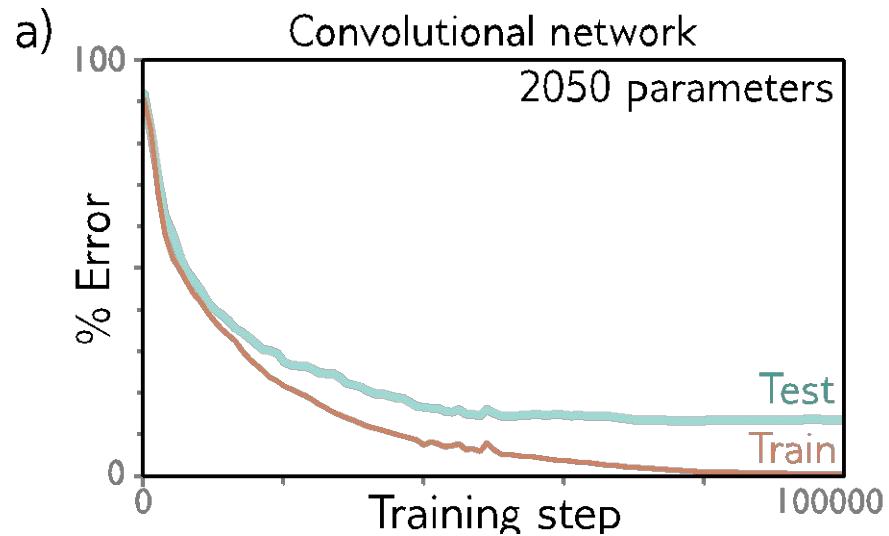
- Four hidden layers
- Three convolutional layers
- One fully-connected layer
- Softmax at end
- Total parameters = 2050
- Trained for 100,000 steps with SGD,
LR = 0.01, batch size 100

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/output mappings, all of which are plausible

2D Convolution

Convolution #2

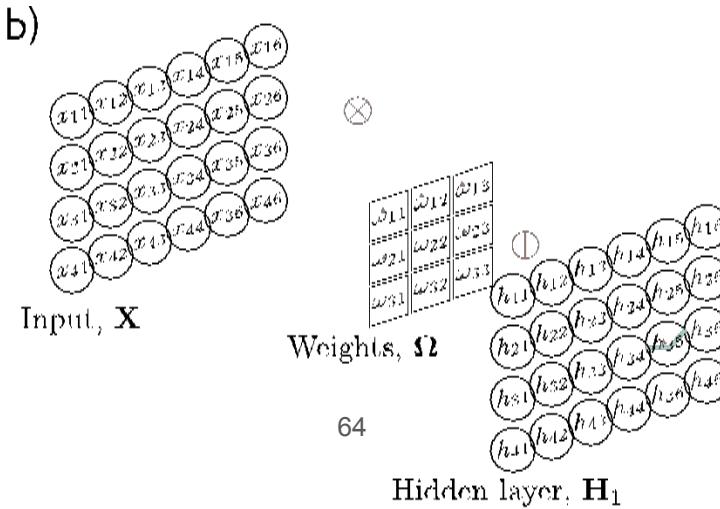
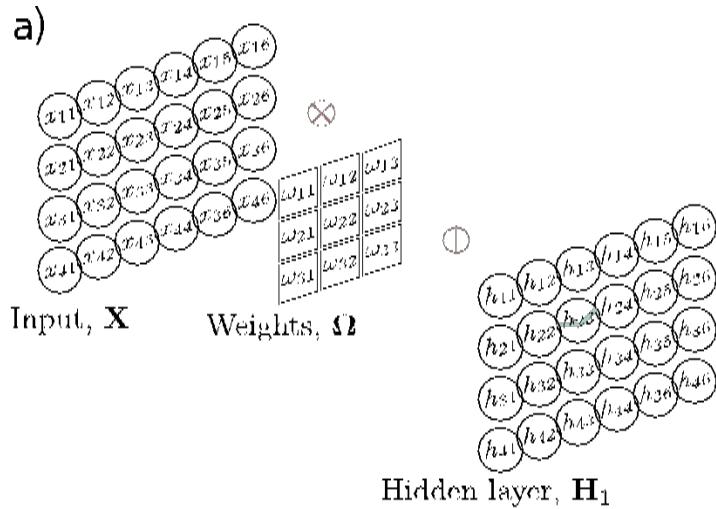
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

2D Convolution

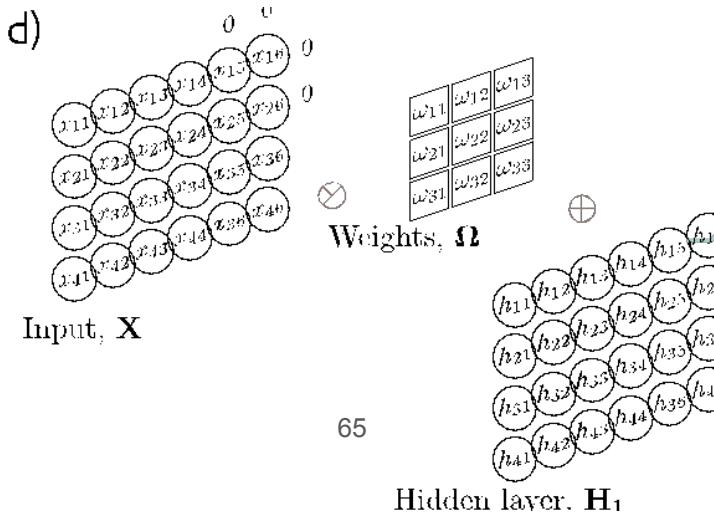
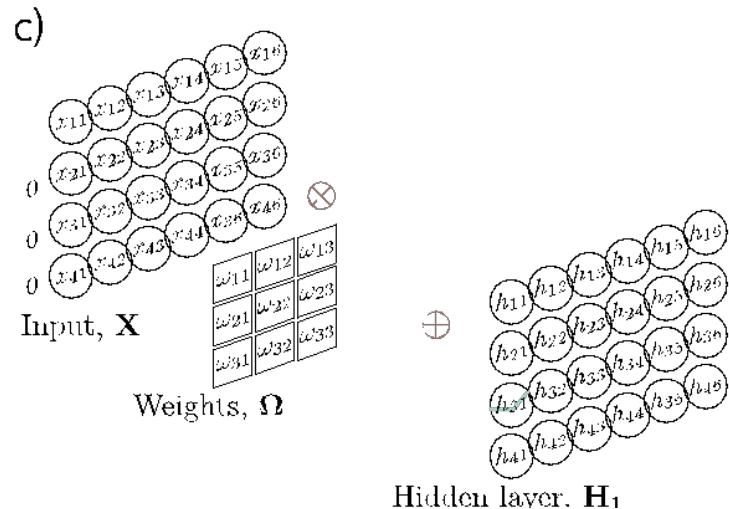
- Convolution in 2D
 - Weighted sum over a $K \times K$ region
 - $K \times K$ weights
- Build into a convolutional layer by adding bias and passing through activation function

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

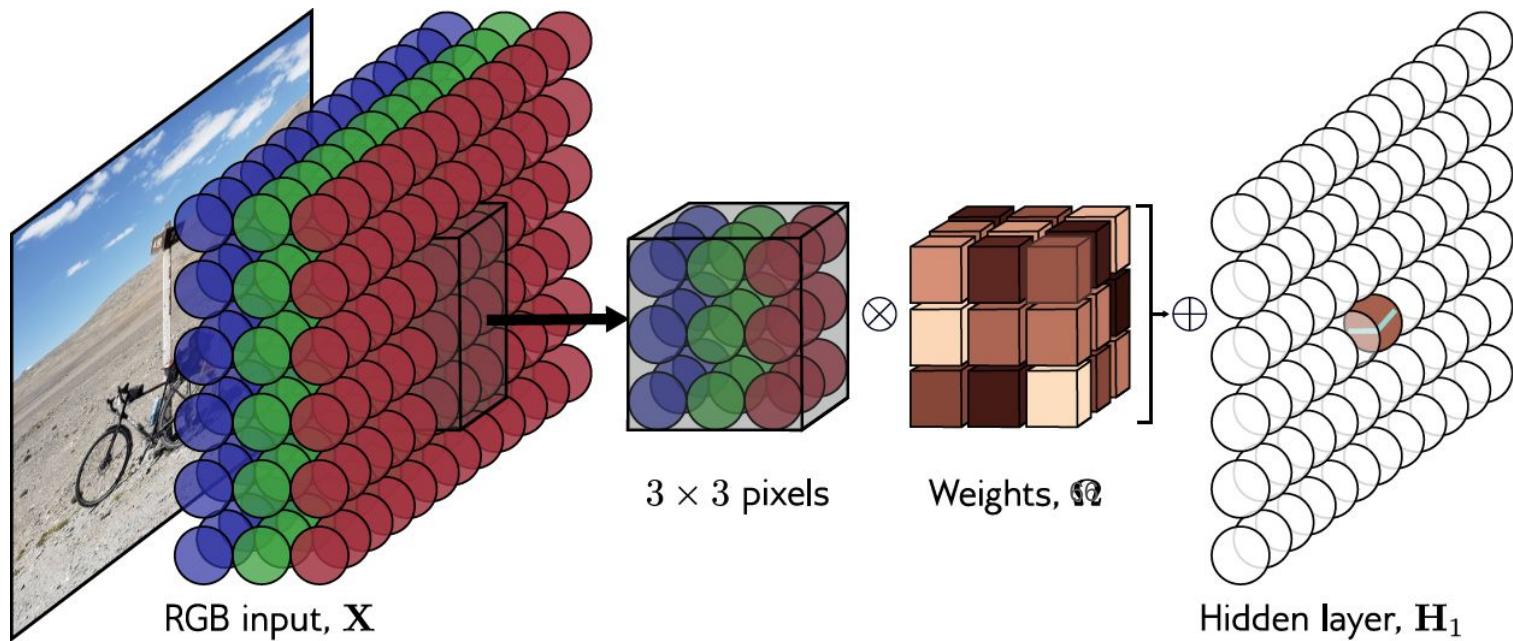
2D Convolution



2D Convolution with Zero Padding



Channels in 2D convolution



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

How many parameters?

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

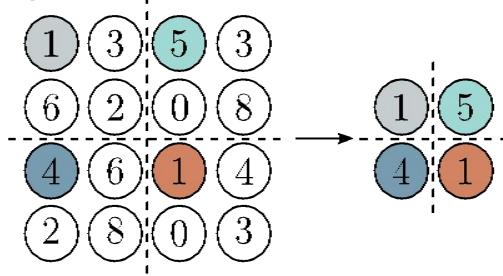
$$\omega \in \mathbb{R}^{C_i \times C_o \times K \times K} \quad \beta \in \mathbb{R}^{C_o}$$

Convolution #2

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

Downsampling

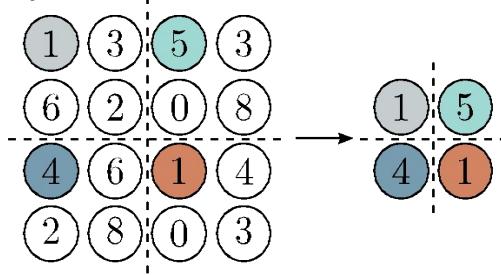
a)



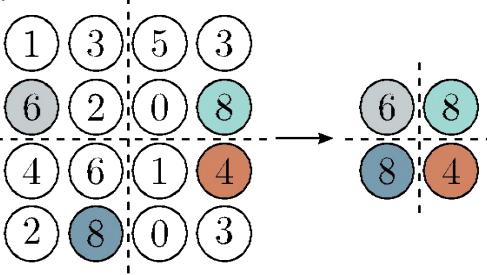
Sample every other
position (equivalent to
stride two)

Downsampling

a)



b)

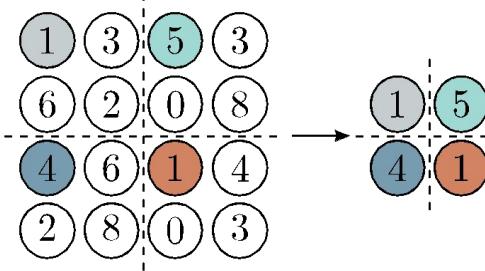


Sample every other
position (equivalent to
stride two)

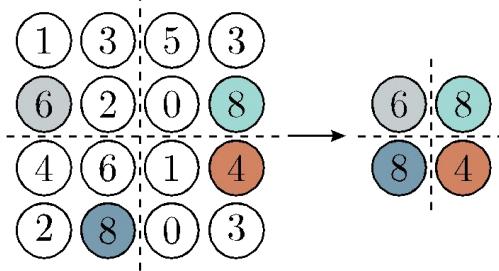
Max pooling
(partial invariance to
translation)

Downsampling

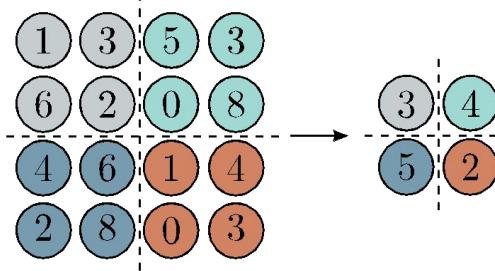
a)



b)



c)



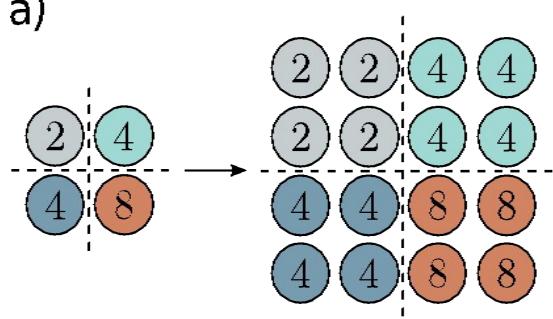
Sample every other
position (equivalent to
stride two)

Max pooling
(partial invariance to
translation)

Mean pooling

Upsampling

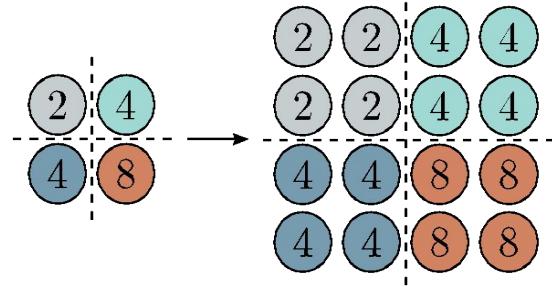
a)



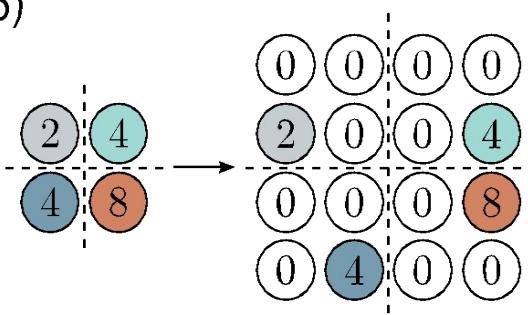
Duplicate

Upsampling

a)



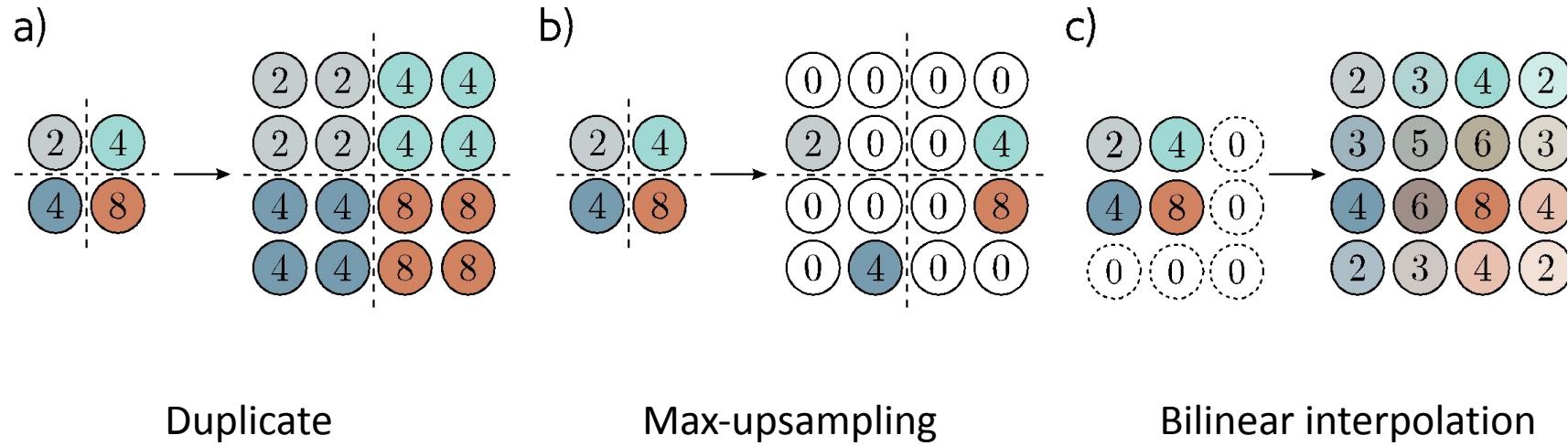
b)



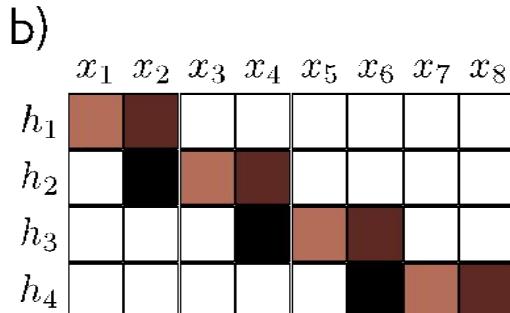
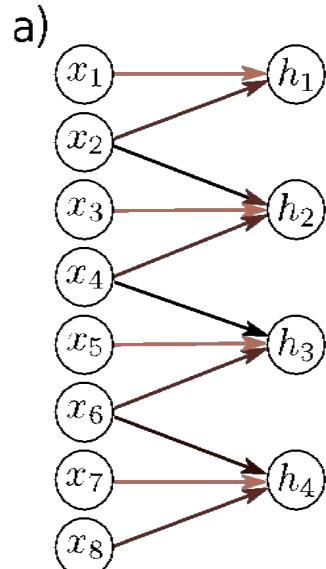
Duplicate

Max-upsampling

Upsampling

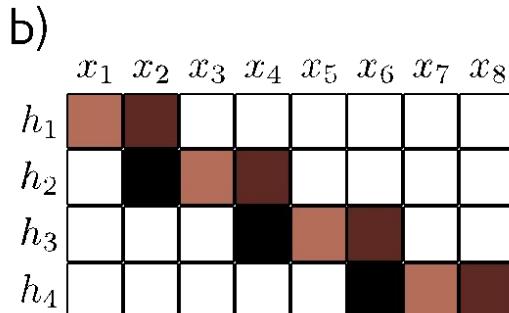
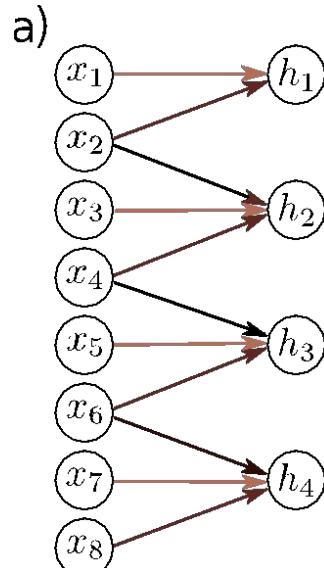


Transposed convolutions

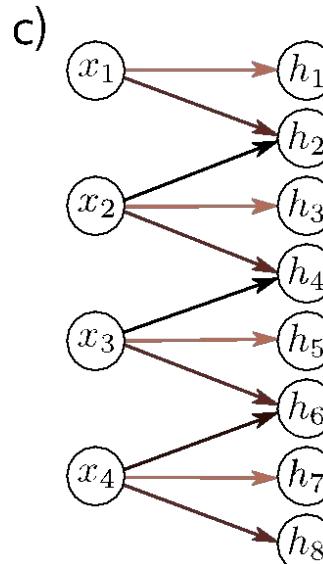


Kernel size 3, Stride 2
convolution

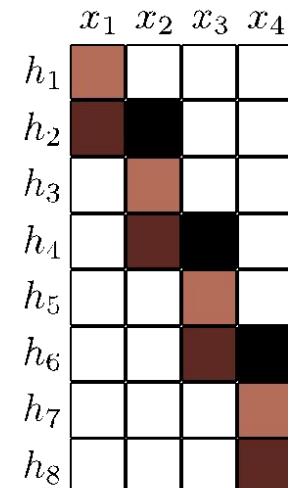
Transposed convolutions



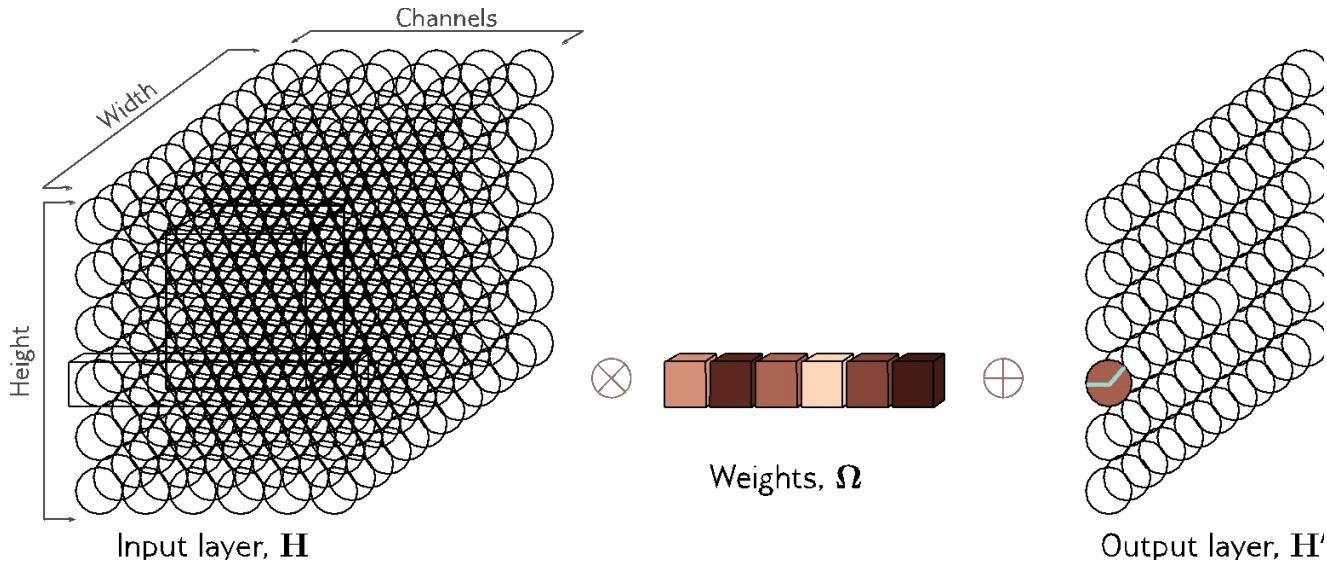
Kernel size 3, Stride 2
convolution



Transposed
convolution



1x1 convolution

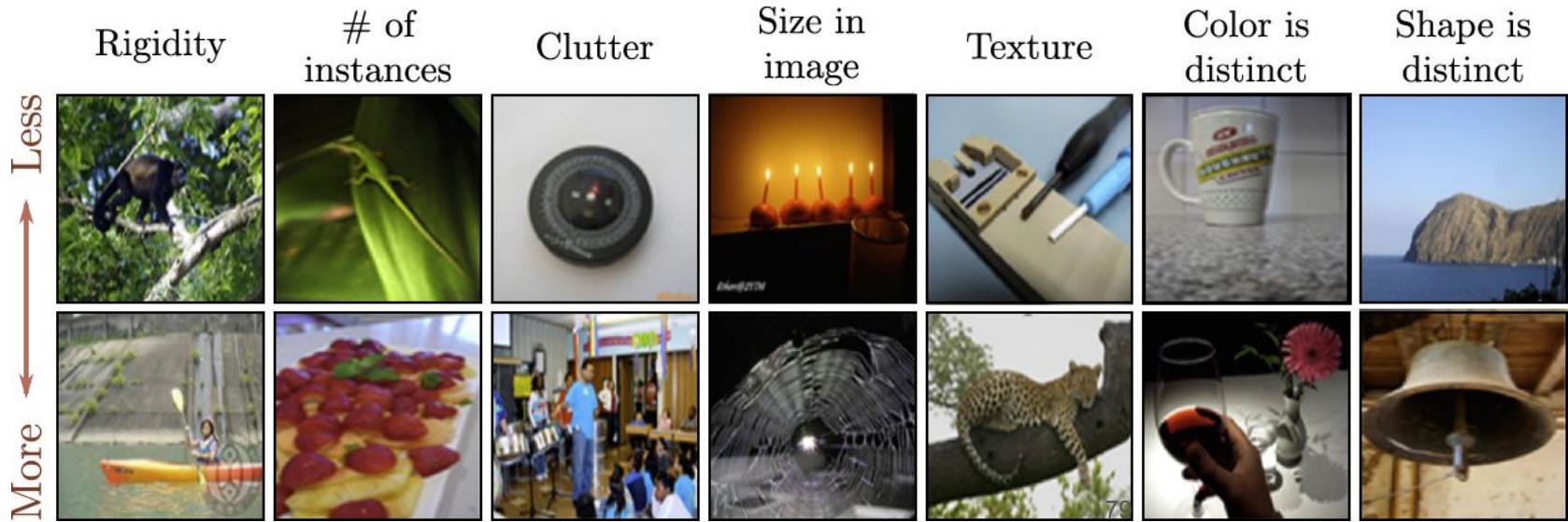


- Mixes channels
- Can change number of channels
- Equivalent to running same fully connected network at each position

Convolution #2

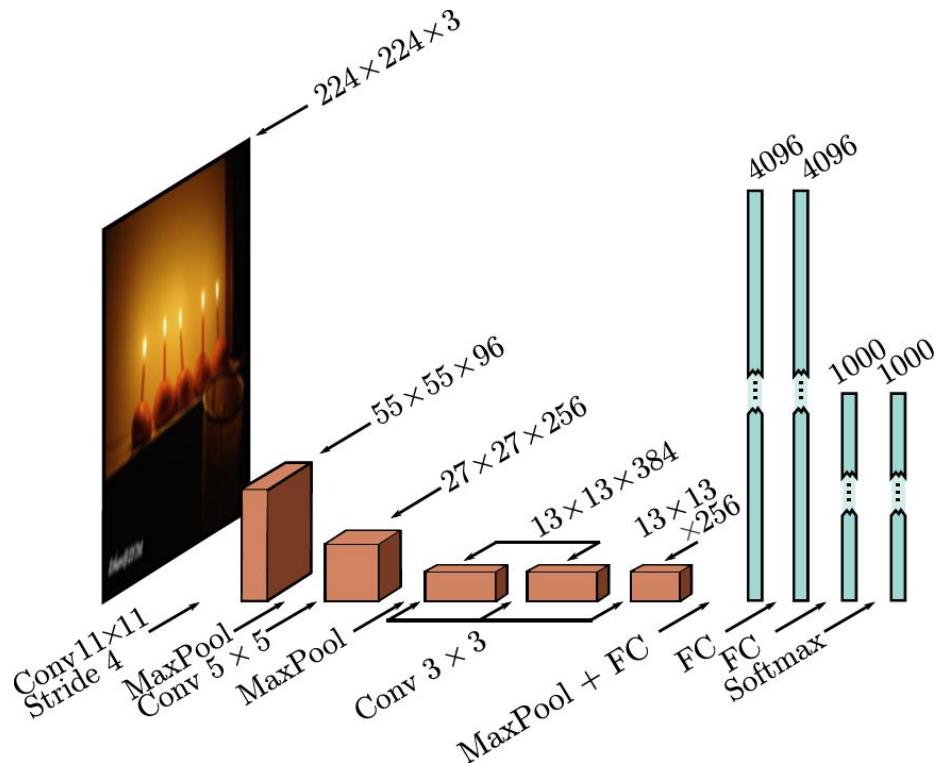
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- **Image classification**
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

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
parameters are in fully connected layers

Data augmentation

a) Original



b) Flip



c) Rotate and crop



d) Vertical stretch



e) Color balance



f) Blur



g) Vignette

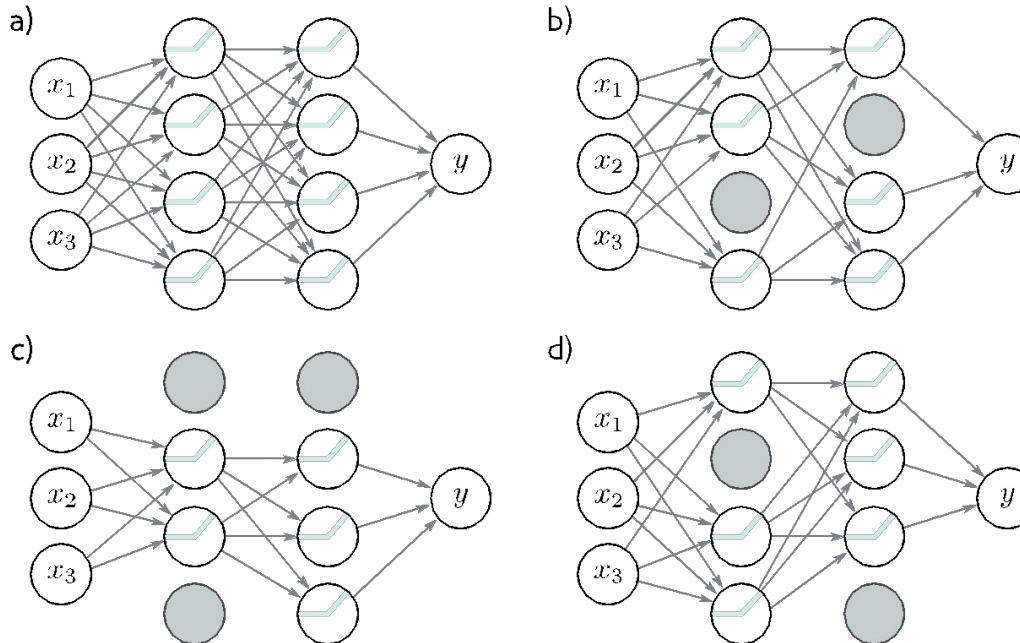


h) Pincushion



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

Dropout

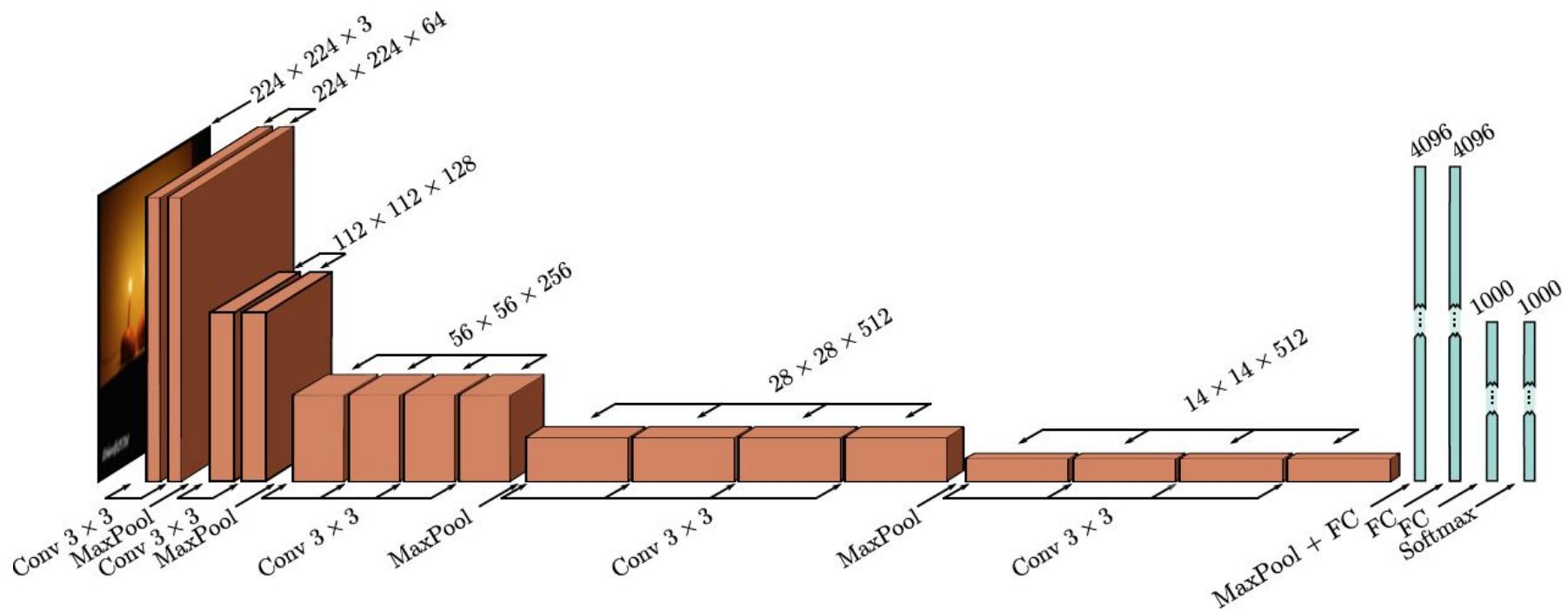


- Dropout was applied in the fully connected layers

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.

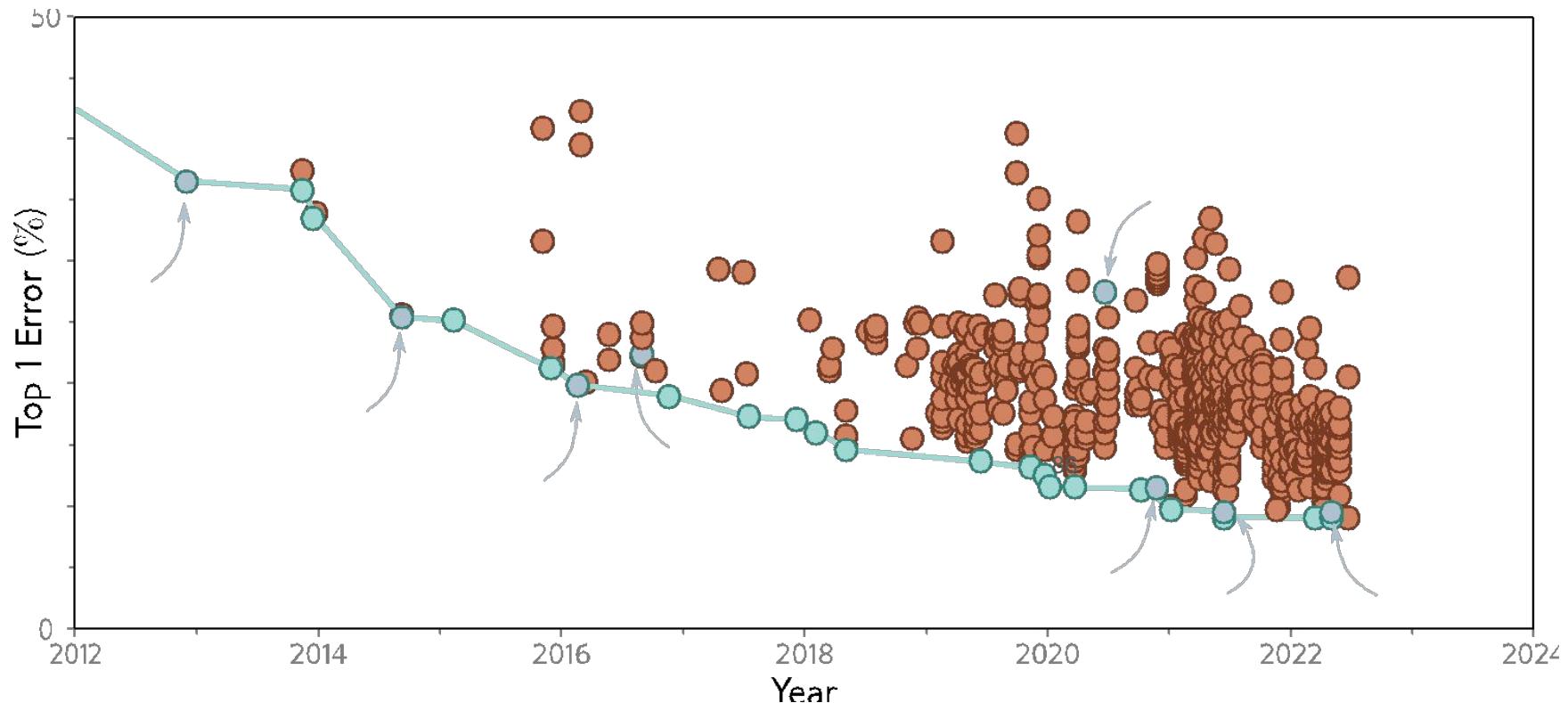
VGG (2015)



Details

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

ImageNet History

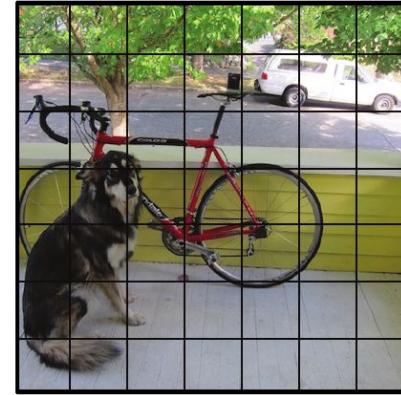


Convolution #2

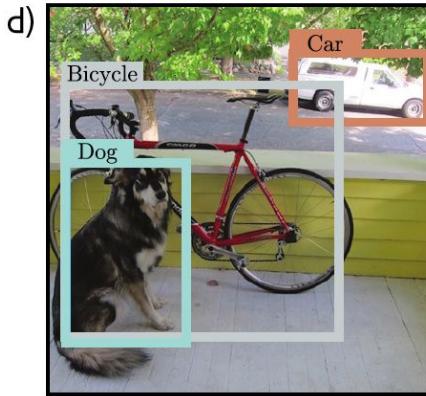
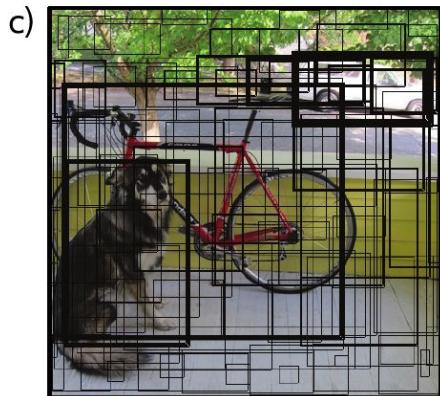
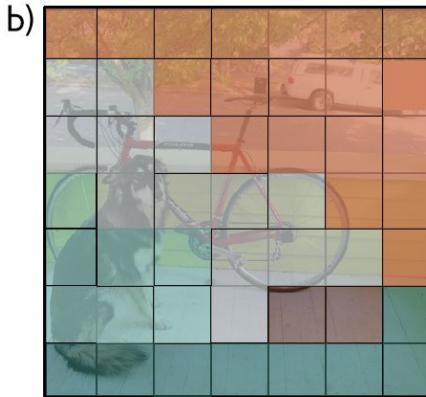
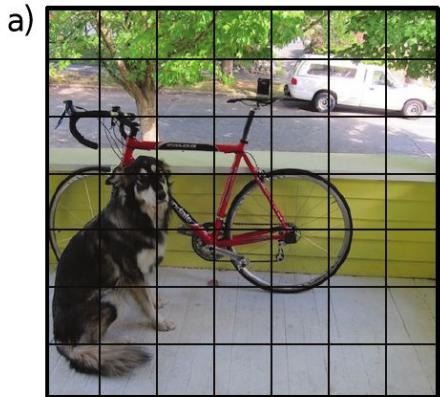
- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

You Only Look Once (YOLO)

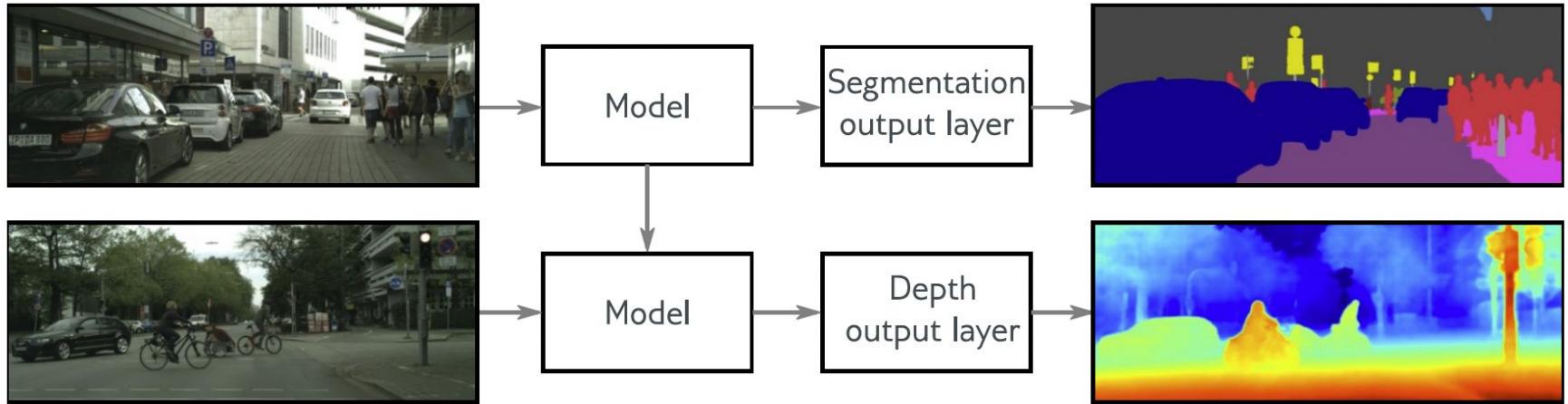
- Network similar to VGG (448x448 input)
- 7×7 grid of locations
- Predict class at each location
- Predict 2 bounding boxes at each location
 - Five parameters –x,y, height, width, and confidence
- Momentum, weight decay, dropout, and data augmentation
- Heuristic at the end to threshold and decide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



Object detection (YOLO)

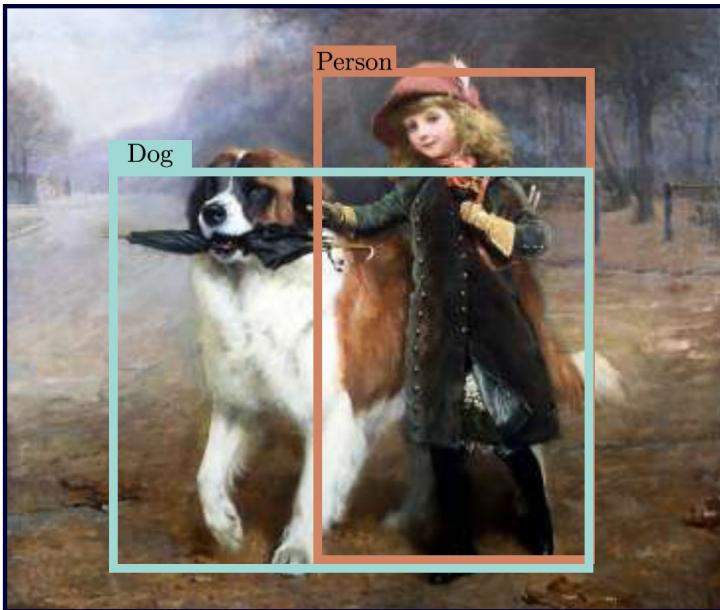
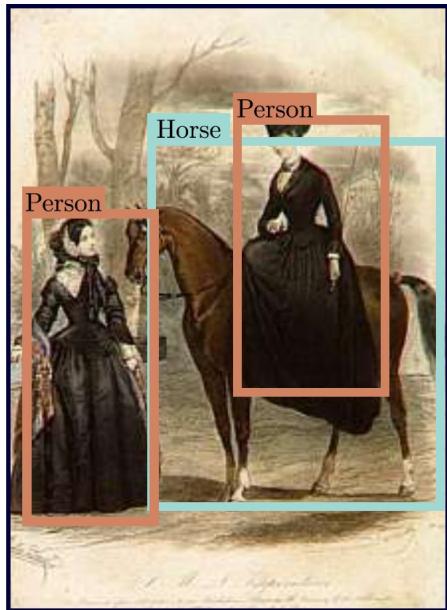


Transfer learning



Transfer learning from ImageNet classification

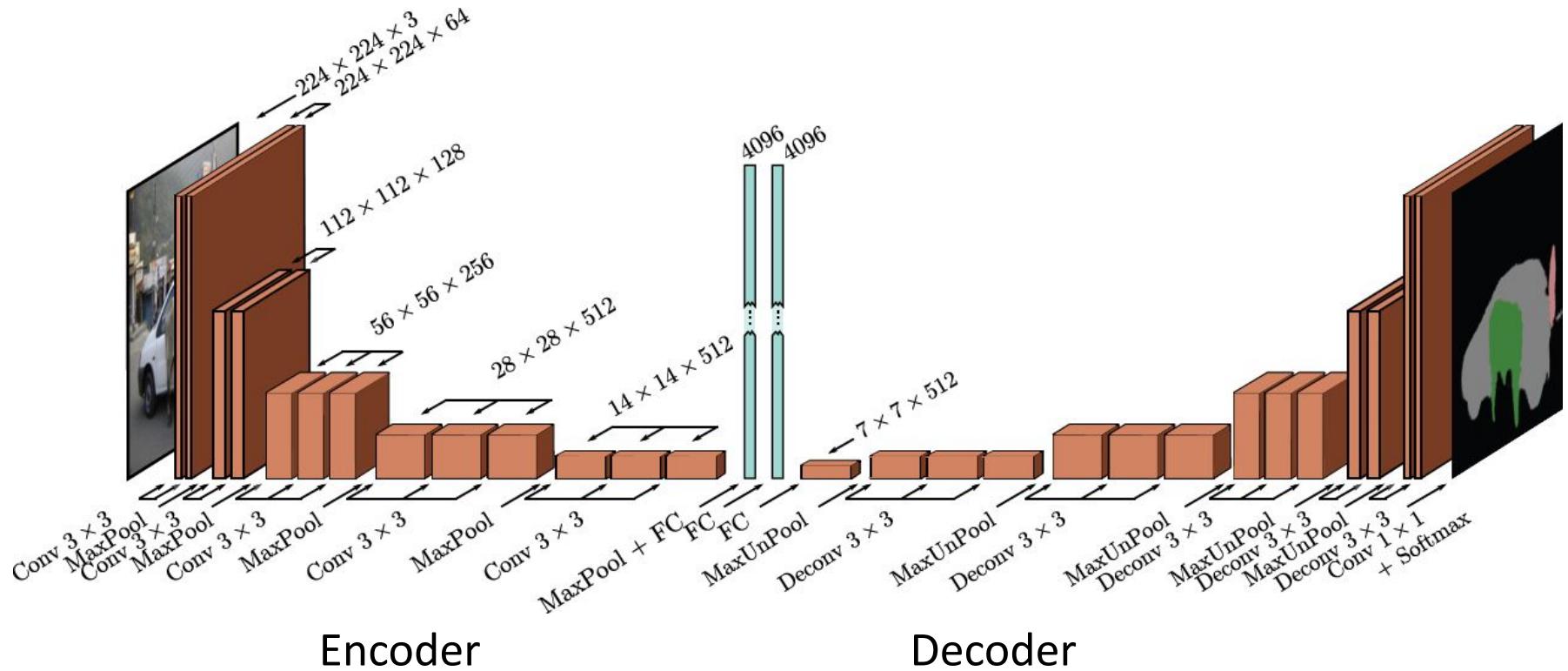
Results



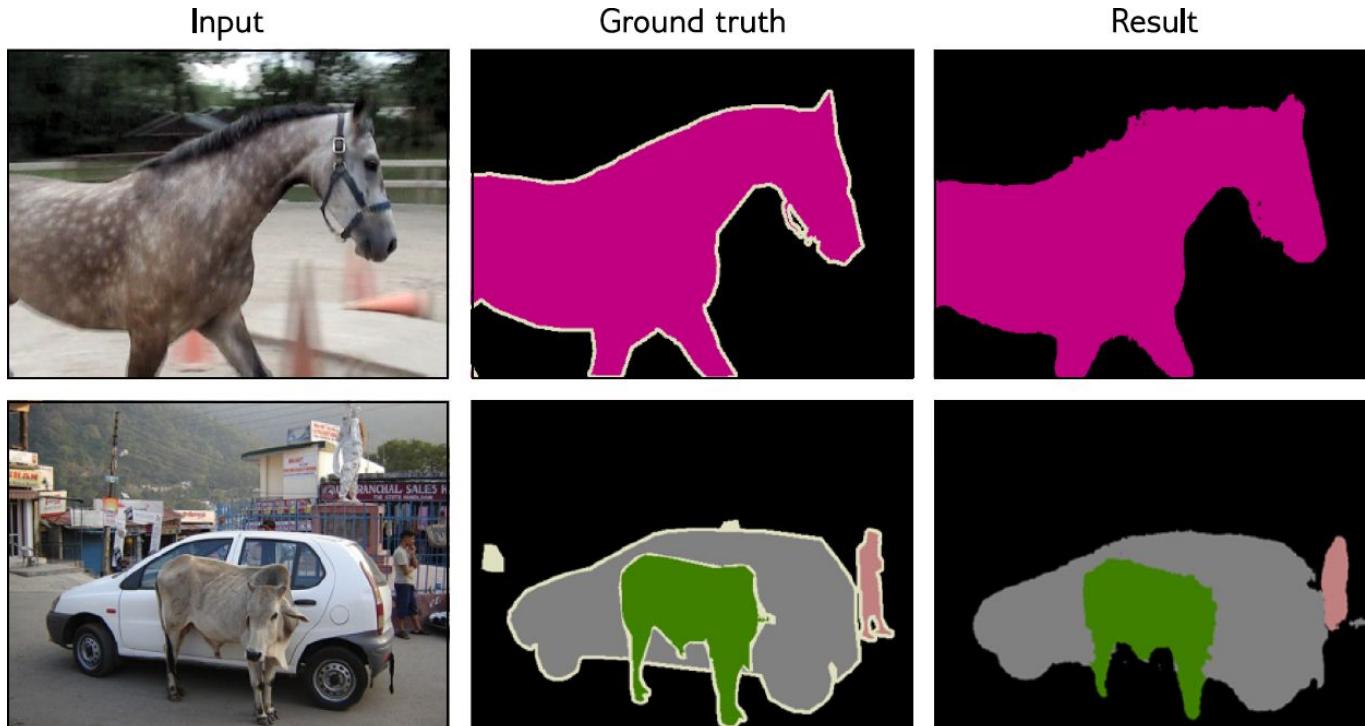
Convolution #2

- 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

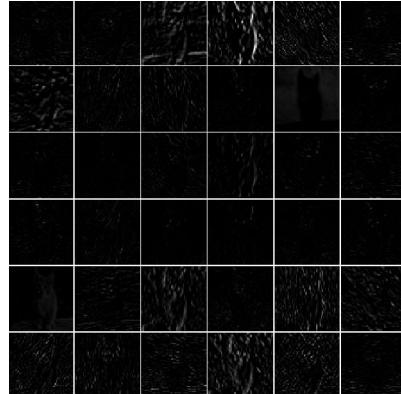


AlexNet



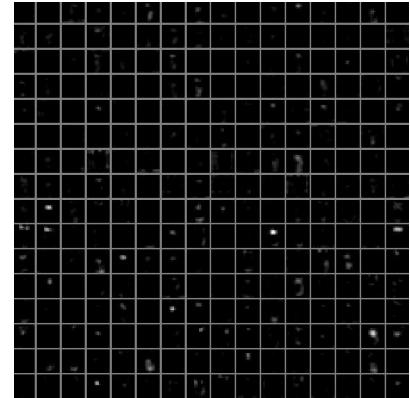
Cat image input
(not actual image)

Activations
(feature maps)



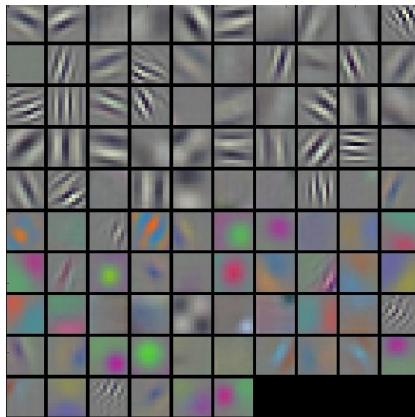
1st Layer

▪ ▪ ▪



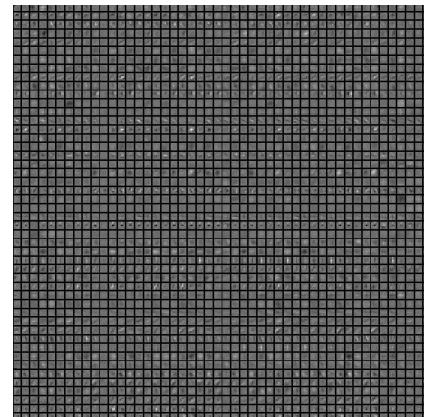
5th Layer

Filter Kernels



2nd Layer

▪ ▪ ▪

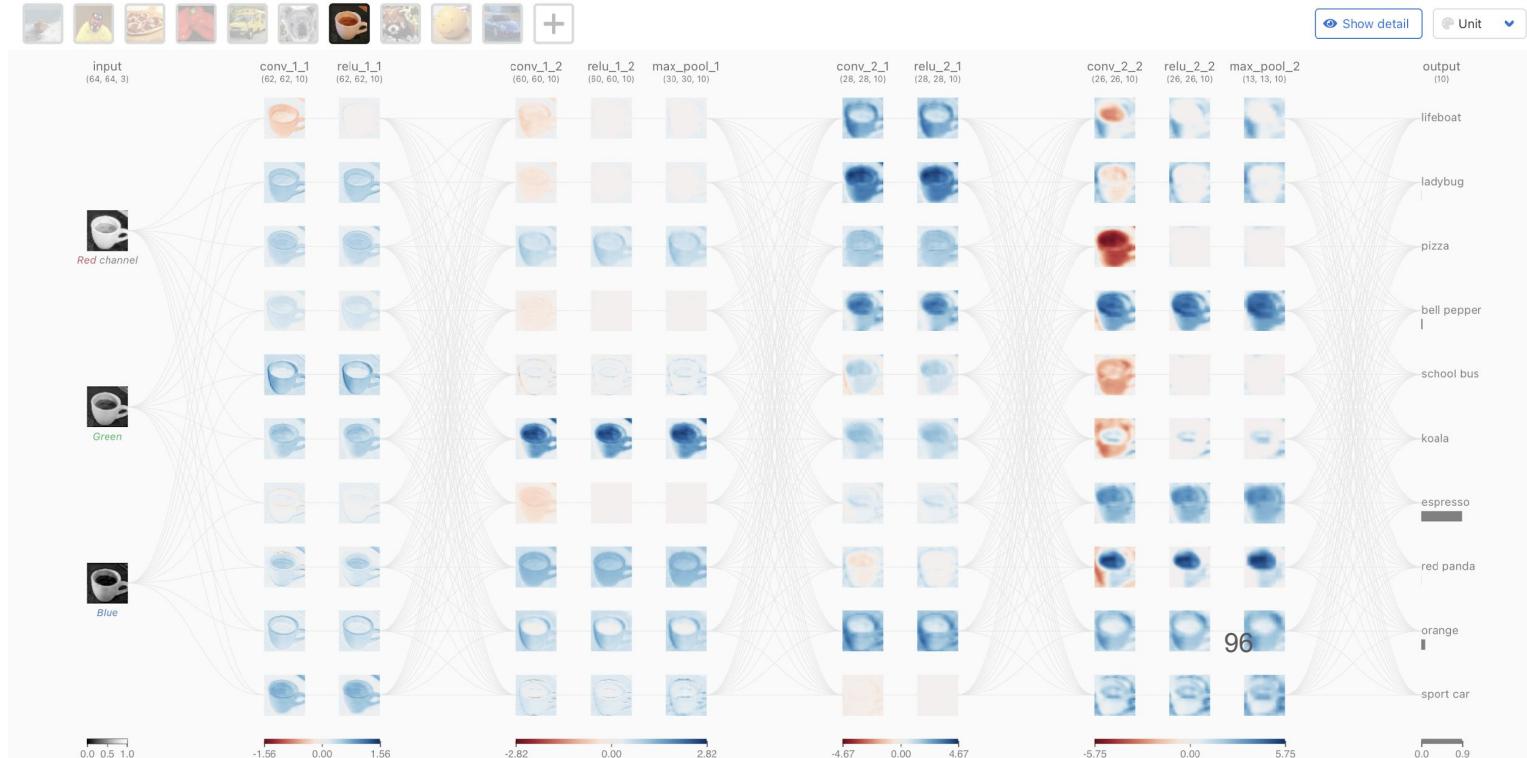


CNN EXPLAINER

Learn Convolutional Neural Network (CNN) in your browser!



Show detail Unit



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

Other Approaches

After midterm:

- Recurrent neural networks
 - Repeatedly run same network on small chunks to update persistent state.
- Attention / Transformers
 - Run same network on small chunks of input, combined all of them via weighted averages...
 - Originally designed for text, but also state of the art for some vision problems now

My favorite, but not a topic for this course

- Neural fields
 - Big network re-parameterizing inputs as functions with small inputs and outputs

Feedback?

