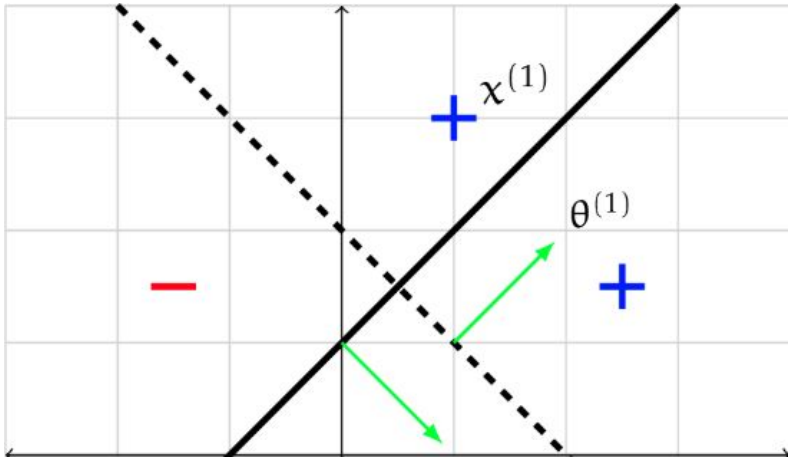
Introduction to Machine Learning

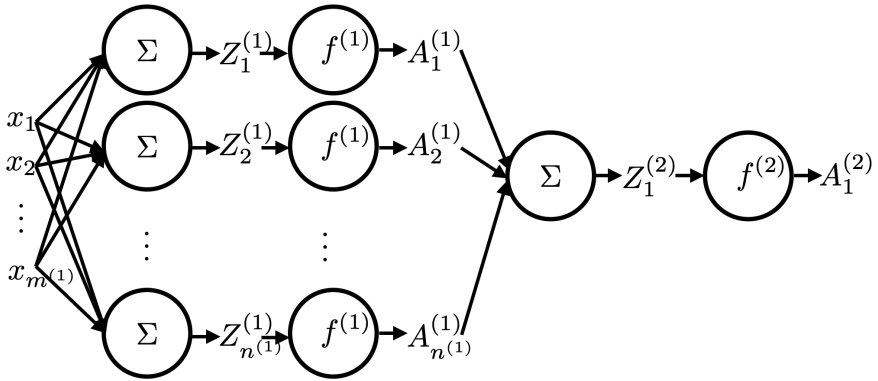


Convolutional Neural Networks

Neural Networks

Universal Function Approximator:

- Neural networks with large-enough width (and bounded depth) can approximate any “nice” function



(Analogous result for bounded width and large-enough depth)

Are we done?

Universal Function Approximator:

Let’s suppose we had an infinite compute: Can we just set the number of hidden units to 10^1000?

Are we done?

Universal Function Approximator:

Let’s suppose we had an infinite compute: Can we just set the number of hidden units to 10^1000?

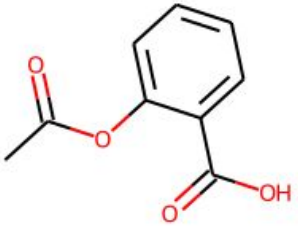
- The theorem says nothing about **sample efficiency**
- Sample efficiency (informal): Model A is more sample efficient than model B if A needs less training data to get the same test performance

Neural Network Architectures

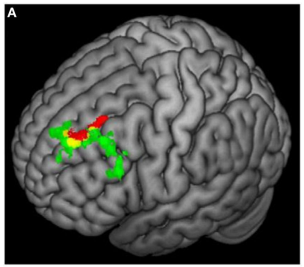
We want to inject structural knowledge about the input domain into our neural network.

Image

Audio



Molecules



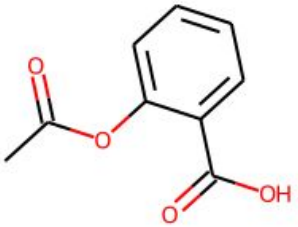
fMRI

Neural Network Architectures

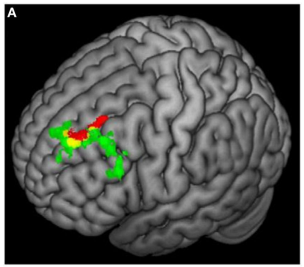
We want to inject structural knowledge about the input domain into our neural network.

Image

Audio



Molecules



fMRI

If the neural network is “aware” of input domain structure, then it can learn faster and generalize better.

Neural Network Architectures

We want to inject structural knowledge about the input domain into our neural network.

Image

Today

Audio

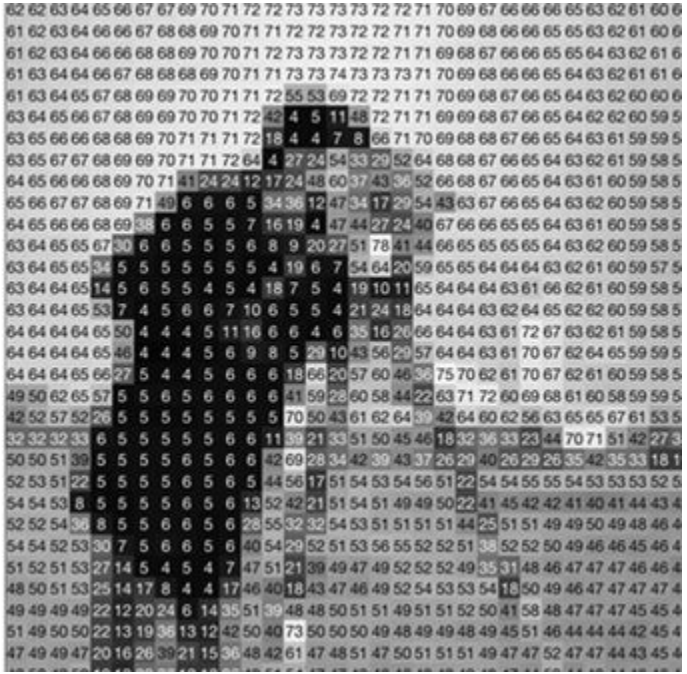
Molecules

fMRI

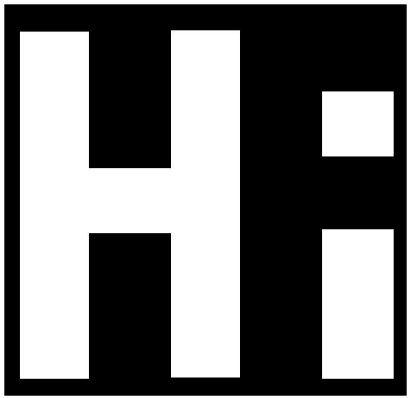
If the neural network is “aware” of input domain structure, then it can learn faster and generalize better.

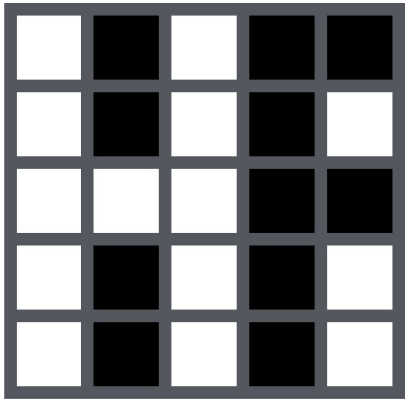
How do computers see?

How do computers see?



How do computers see?

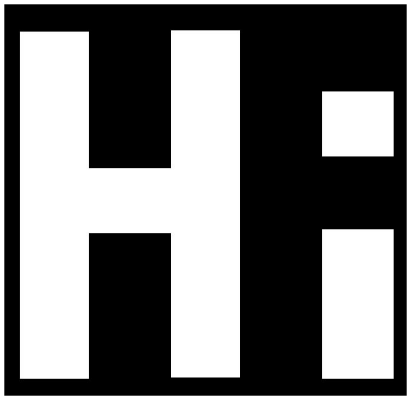


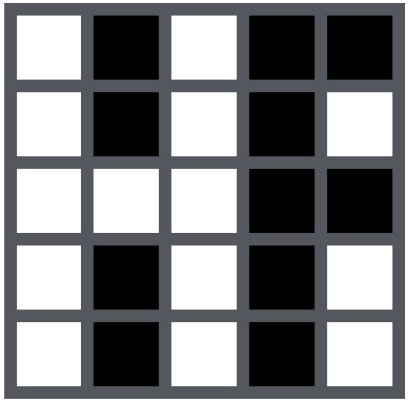


1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

“0” if black,
“1” if white

How do computers see?





1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

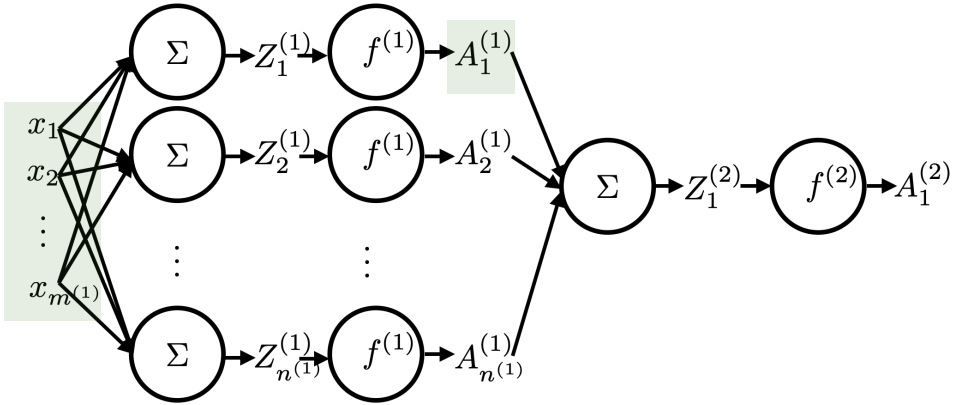
x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}

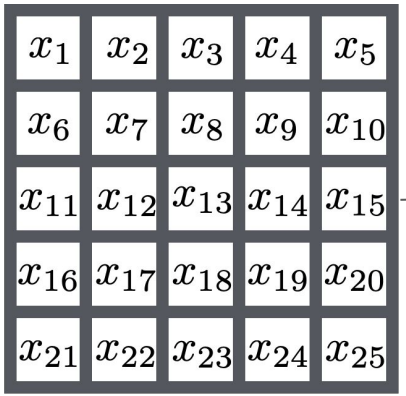
Logistic Regression/
Neural Network

Whether “H” is
in image or not

“Fully-Connected” Neural Networks

Every hidden unit is connected to the input!



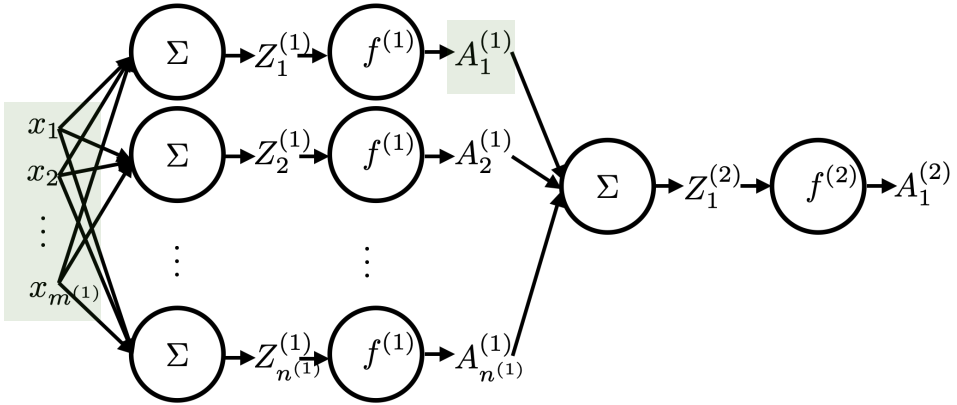


Logistic Regression/ Neural Network

Whether “H” is in image or not

“Fully-Connected” Neural Networks

Every hidden unit is connected to the input!



x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}

The model doesn’t know apriori that x7 is close to x1.

Image Structure

What do we know about images?

Image Structure

What do we know about images?

Spatial Locality: things nearby in 2D space are more likely to be similar (or have the same source of information)

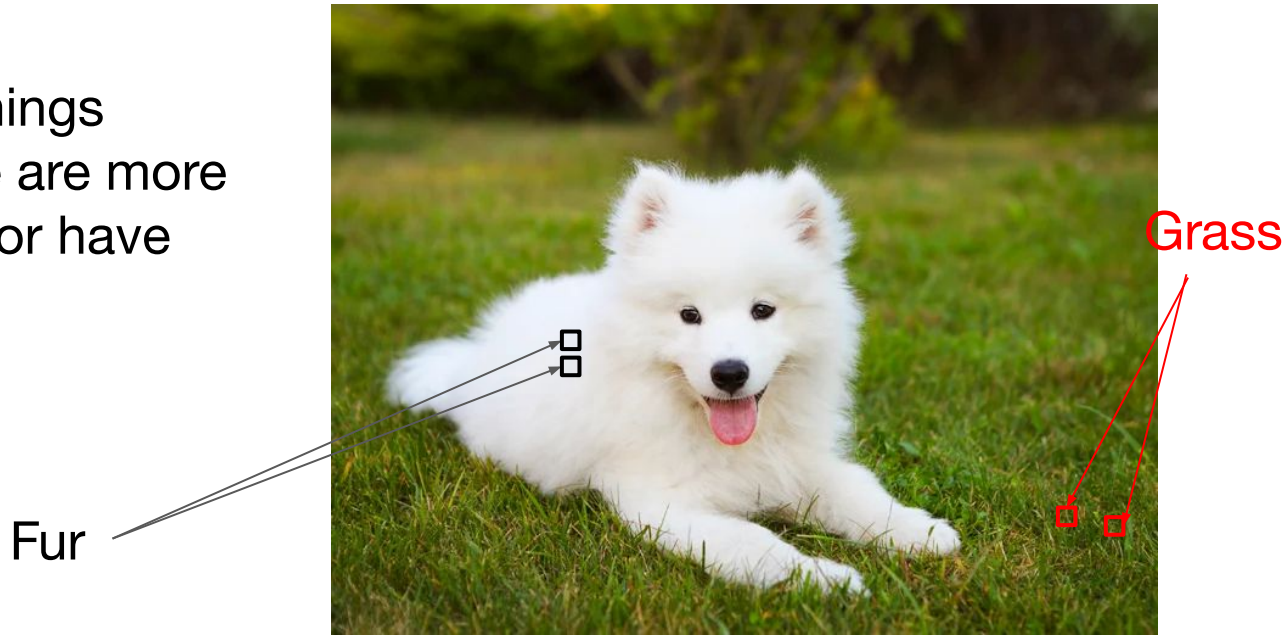
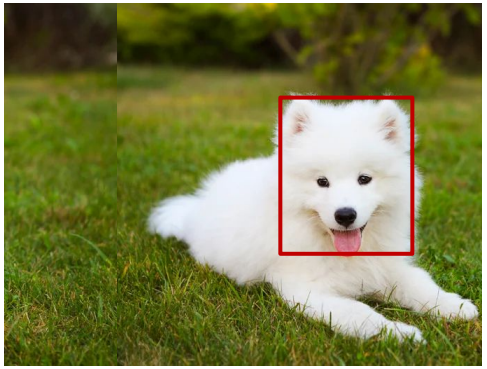
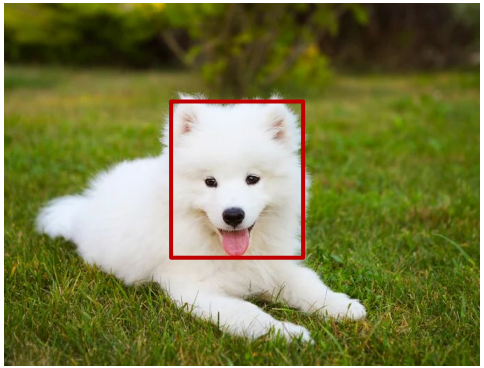


Image Structure

What do we know about images?

Translation Invariance: The same pattern of pixels has the same “interpretation” no matter where it occurs in the image.





Still a “samoyed” even though absolute position is different

Convolutional Layer

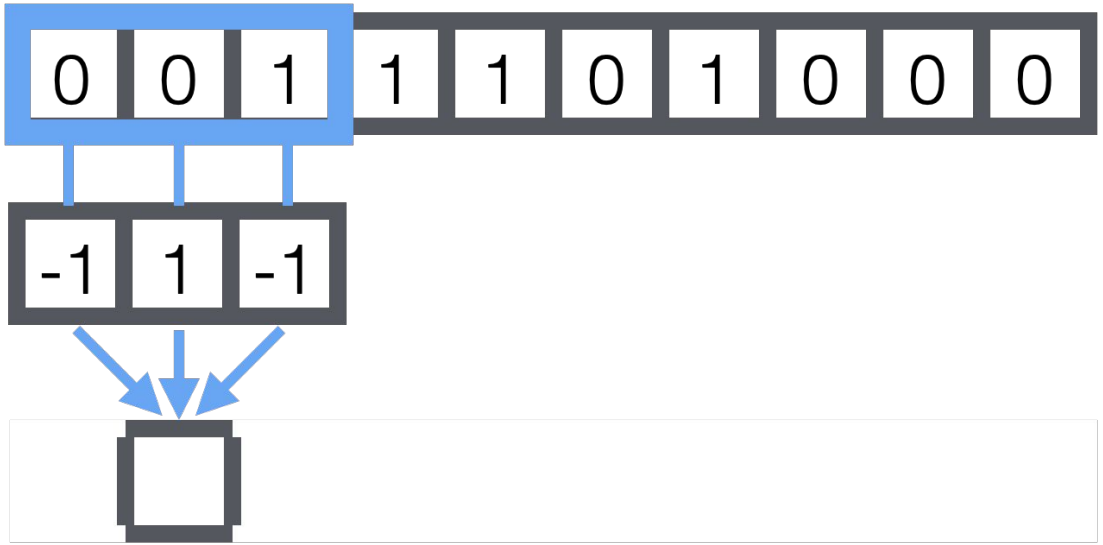
1D sequence

Convolutional Filter
(model parameters)

Convolutional Layer

1D sequence

Convolutional Filter
(model parameters)



Convolutional Layer

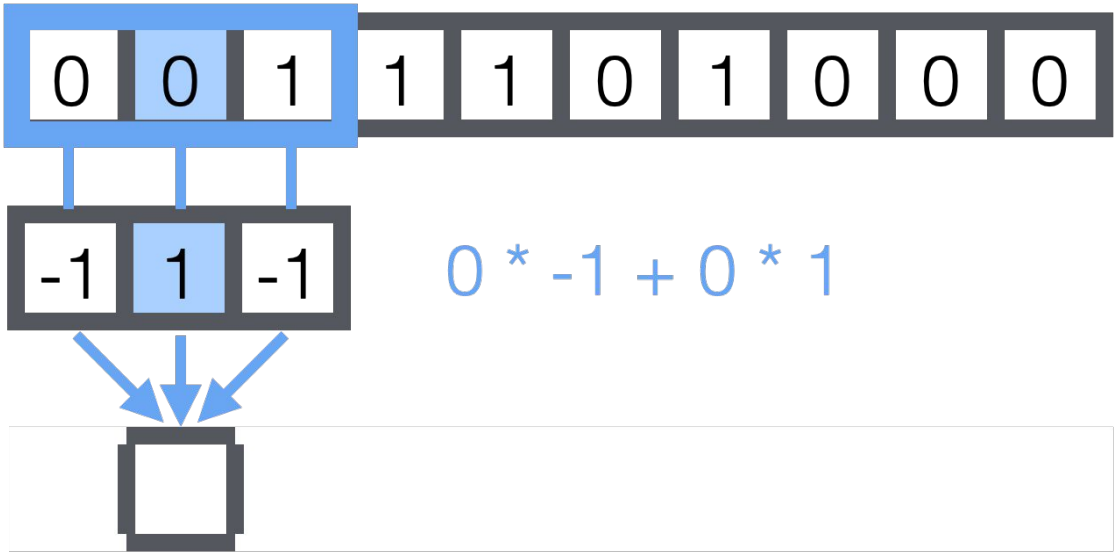
1D sequence

Convolutional Filter
(model parameters)

Convolutional Layer

1D sequence

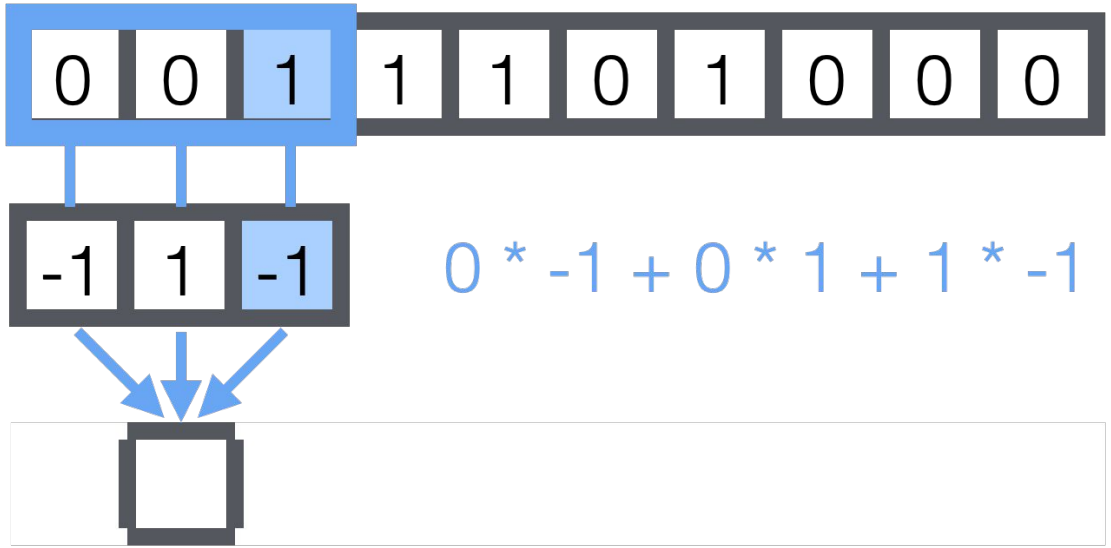
Convolutional Filter
(model parameters)



Convolutional Layer

1D sequence

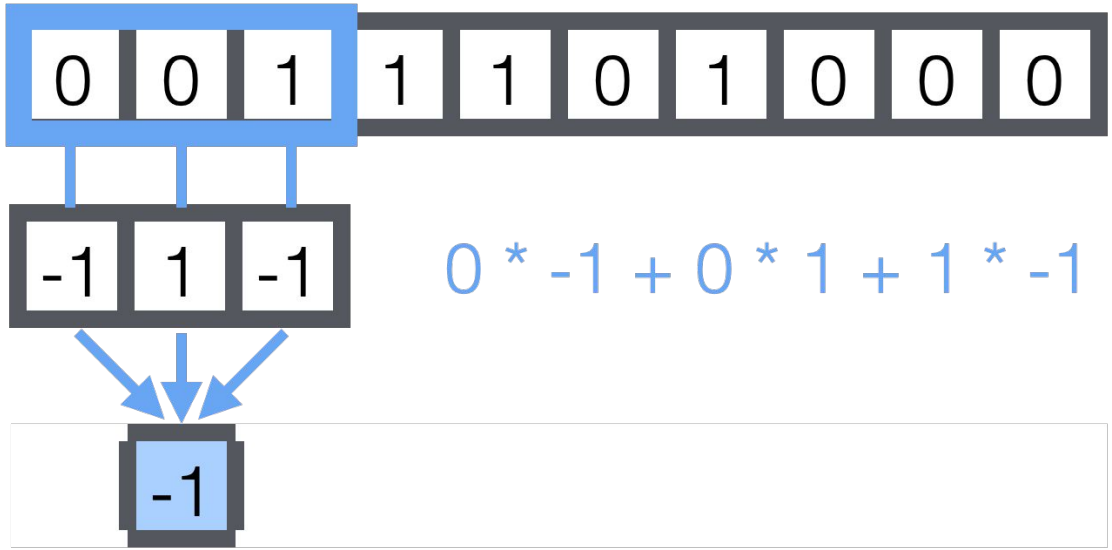
Convolutional Filter
(model parameters)



Convolutional Layer

1D sequence

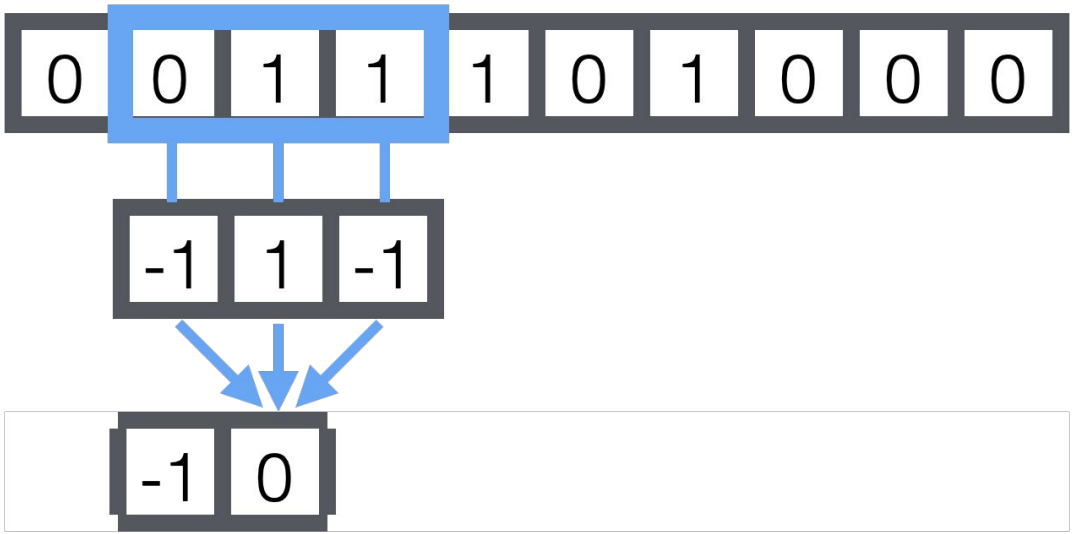
Convolutional Filter
(model parameters)



Convolutional Layer

1D sequence

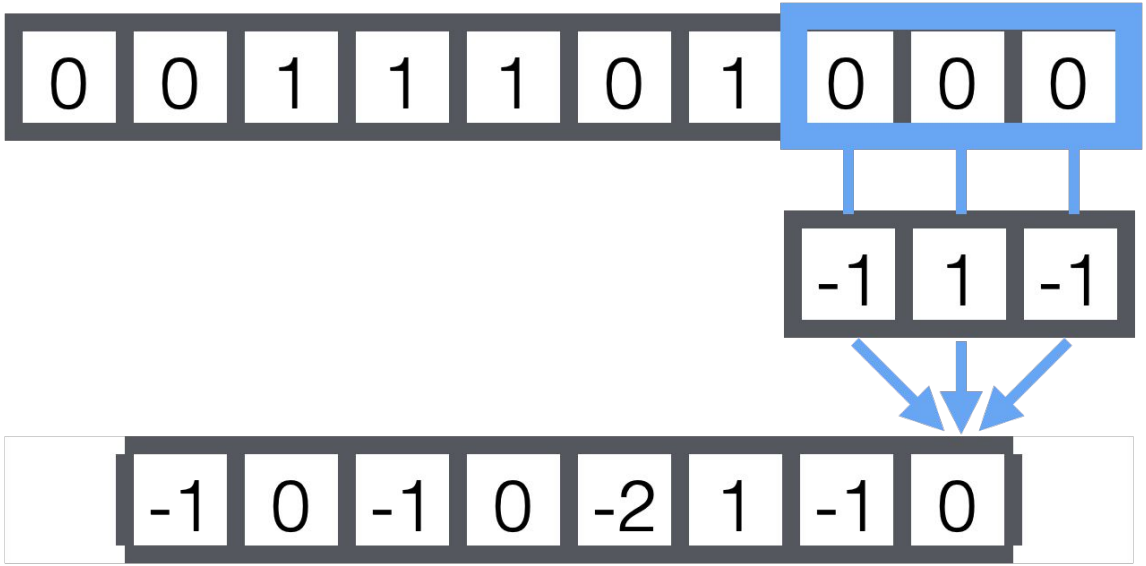
Convolutional Filter
(model parameters)



Convolutional Layer

1D sequence

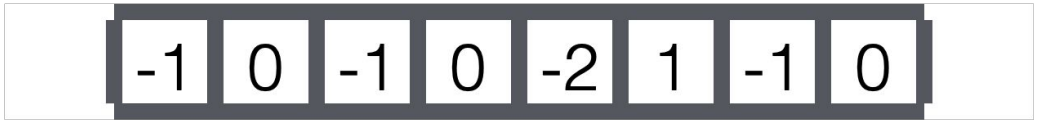
Convolutional Filter
(model parameters)



Convolutional Layer

1D sequence

Convolutional Filter
(model parameters)



(Optional) Padding to
preserve sequence length

Convolutional Layer

1D sequence

Convolutional Filter
(model parameters)

Preactivation Layer

(Optional) Padding to
preserve sequence length

Convolutional Layer

1D sequence

Convolutional Filter
(model parameters)

with bias +1

Convolutional filter can have biases (as in fully connected layers)

Convolutional Layer

1D sequence

Convolutional Filter
(model parameters)

with bias *b*

Filter weights (and bias)
will be learned via
gradient descent

Convolutional Layer

1D sequence

Convolutional Filter
(model parameters)

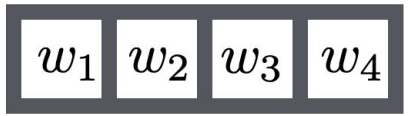
with bias *b*

Filters can have different widths

Convolutional Layer

1D sequence

Convolutional Filter
(model parameters)



with bias *b*

Filters can have different widths

Convolutional Layer

1D sequence

Convolutional Filter
(model parameters)

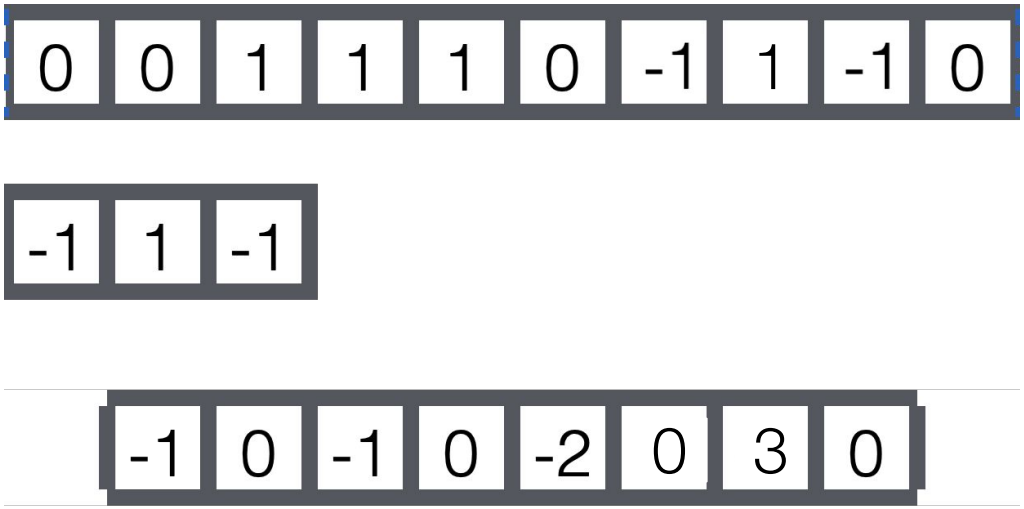
with bias +1

Preactivation Layer

ReLU nonlinearity

Convolutional Layer: Spatial Locality

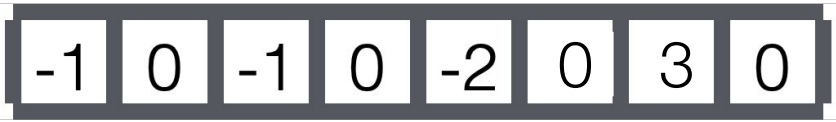
What is this filter doing?



Convolutional Layer: Spatial Locality

What is this filter doing?

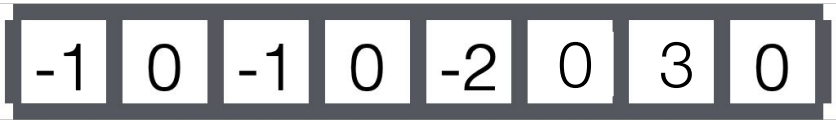
Looking for length-3 sequences things that it has high dot product with.



Convolutional Layer: Spatial Locality

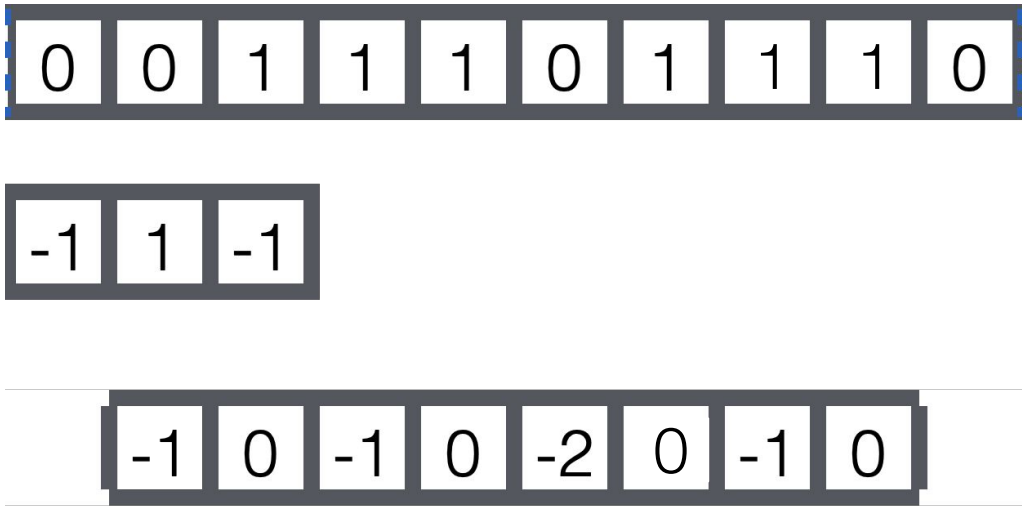
What is this filter doing?

Looking for length-3 sequences things that it has high dot product with.

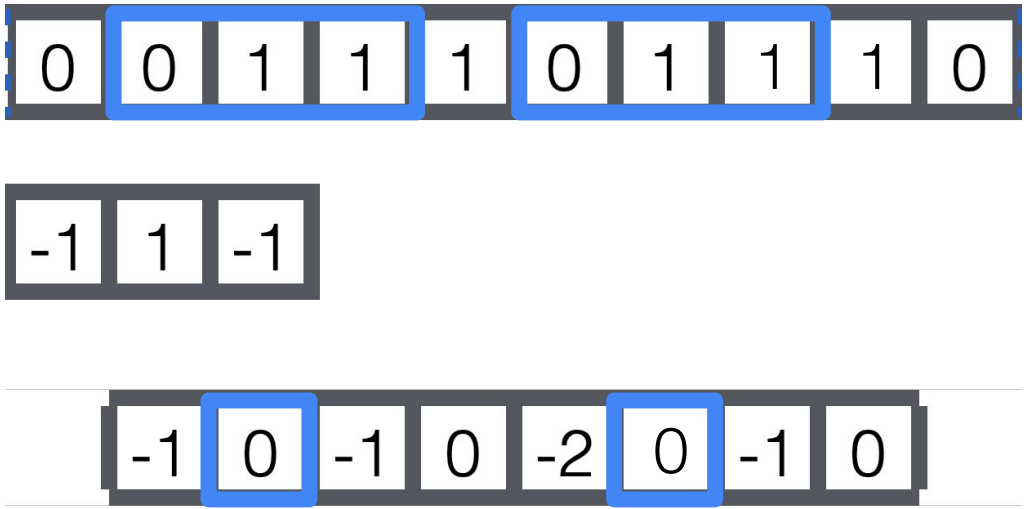


Looks for **local** information!
Spatial locality is encoded into the model

Convolutional Layer: Translational Invariance

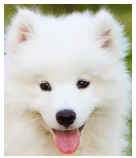


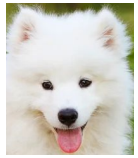
Convolutional Layer: Translational Invariance



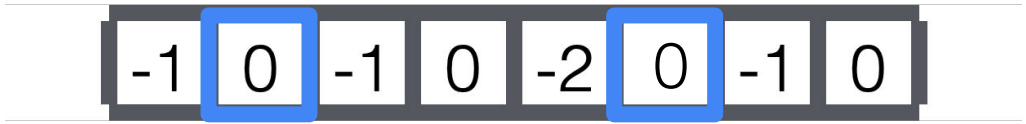
Same local region =>
same output!

Convolutional Layer: Translational Invariance





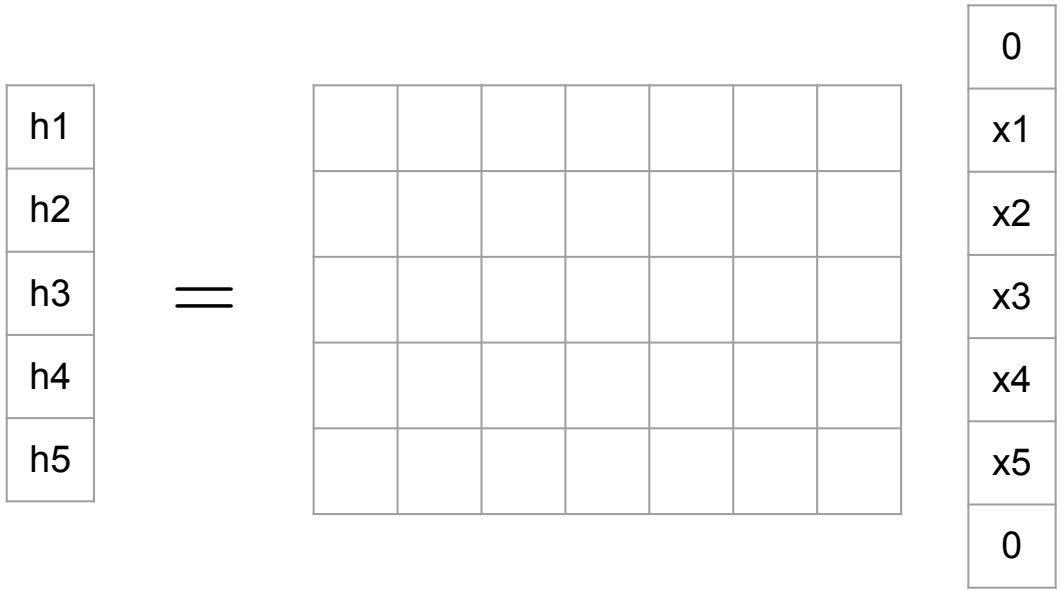
Doesn't matter where the feature occurs



Same local region => same output!

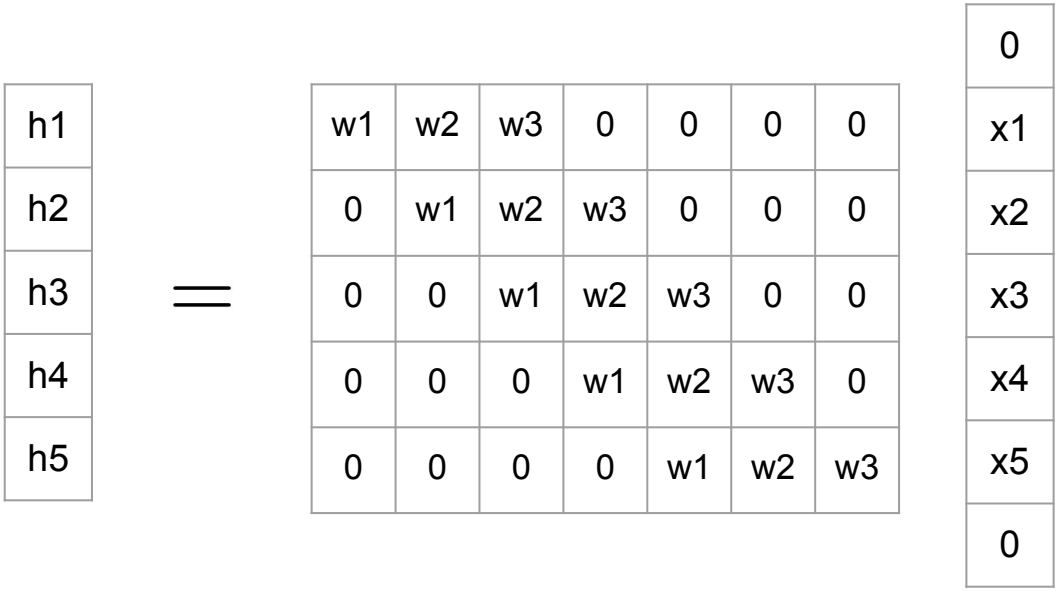
Convolution as Matrix Multiplication

$$\mathbf{h} = \mathbf{W}\mathbf{x}$$

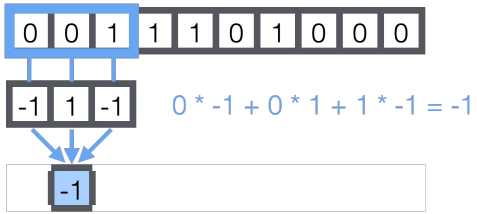


Convolution as Matrix Multiplication

$$\mathbf{h} = \mathbf{W}\mathbf{x}$$



Convolution as Matrix Multiplication



$$\mathbf{h} = \mathbf{W}\mathbf{x}$$

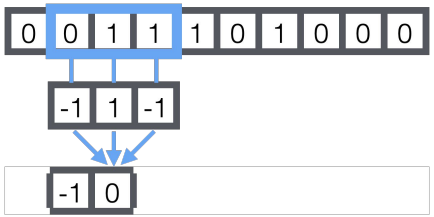
h1
h2
h3
h4
h5

=

w1	w2	w3	0	0	0	0
0	w1	w2	w3	0	0	0
0	0	w1	w2	w3	0	0
0	0	0	w1	w2	w3	0
0	0	0	0	w1	w2	w3

0
x1
x2
x3
x4
x5
0

Convolution as Matrix Multiplication



$$\mathbf{h} = \mathbf{W}\mathbf{x}$$

h1
h2
h3
h4
h5

=

w1	w2	w3	0	0	0	0
0	w1	w2	w3	0	0	0
0	0	w1	w2	w3	0	0
0	0	0	w1	w2	w3	0
0	0	0	0	w1	w2	w3

0
x1
x2
x3
x4
x5
0

Fully Connected Layer

$$\mathbf{h} = \mathbf{W}\mathbf{x}$$

These weights can be anything!

Convolution as Matrix Multiplication

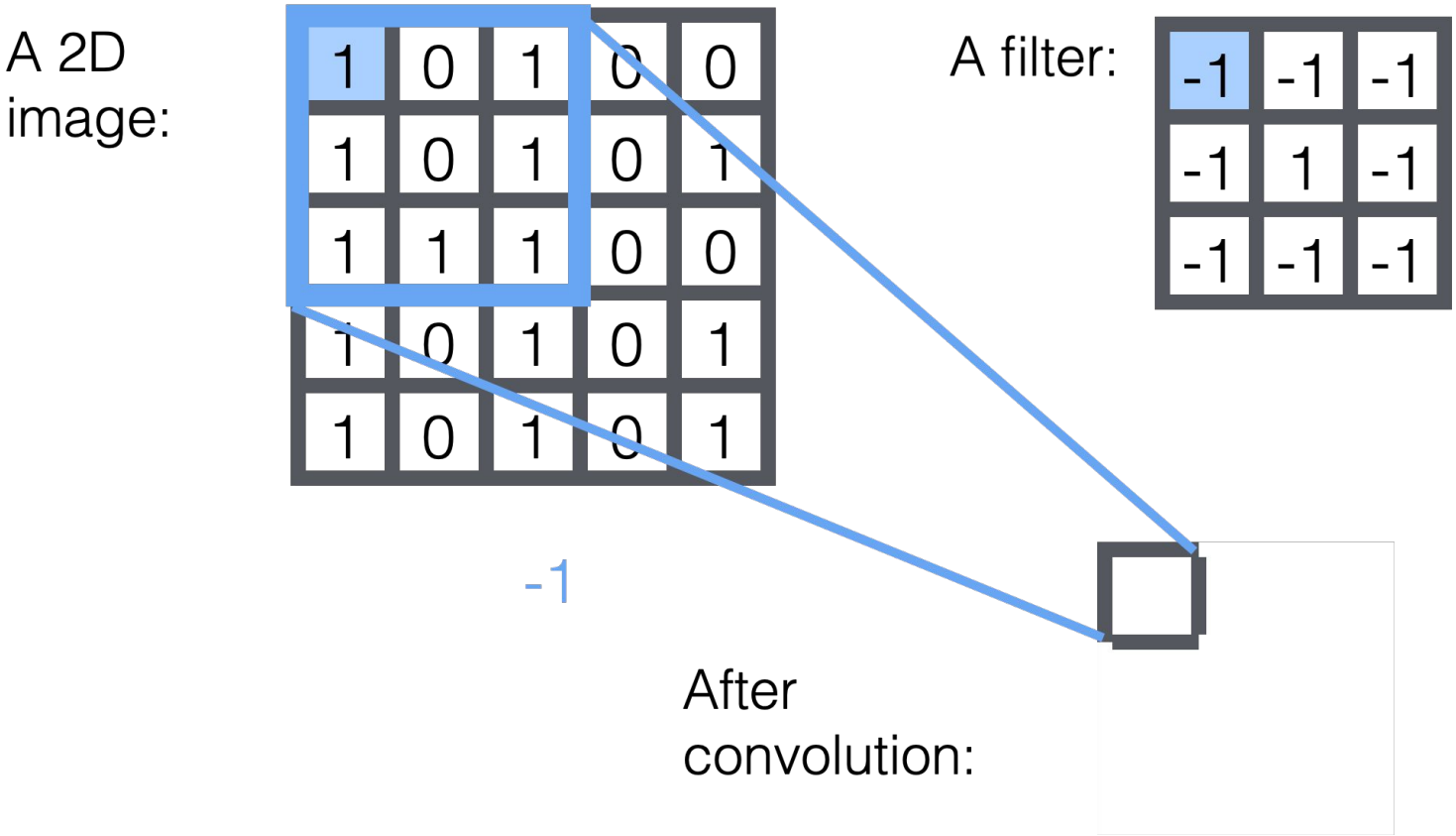
- 1D Convolution is just matrix multiplication with a structured matrix!

- Convolutional layers used shared weights to enforce spatial locality and translation invariance.

- They are less flexible of a model than fully connected layers (i.e., parameters).

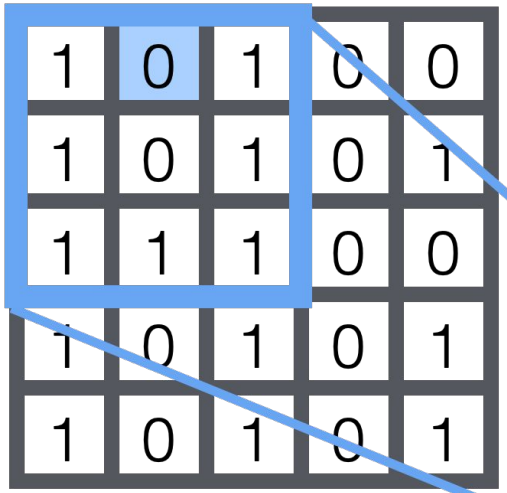
- When would convolutional layers be a bad idea??

Convolutions on Images

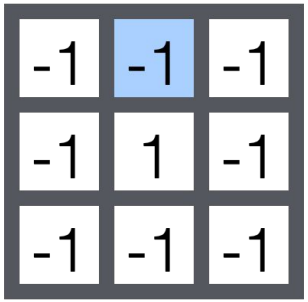


Convolutions on Images

A 2D image:



A filter:



-1 + 0

After convolution:

Convolutions on Images

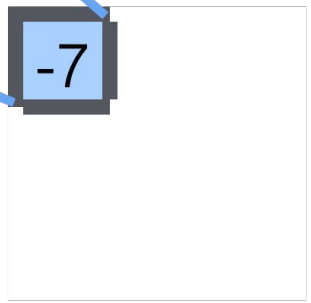
A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

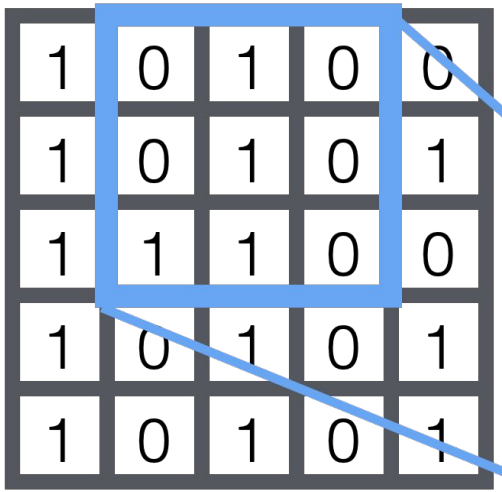
-1	-1	-1
-1	1	-1
-1	-1	-1

-1 + 0 + -1
+ -1 + 0 + -1
+ -1 + - 1 + -1
= -7

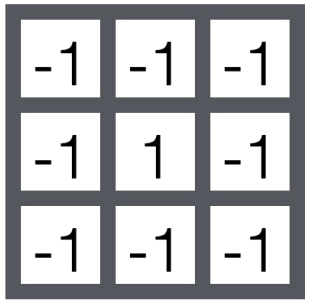


Convolutions on Images

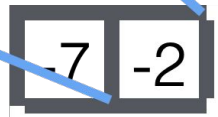
A 2D image:



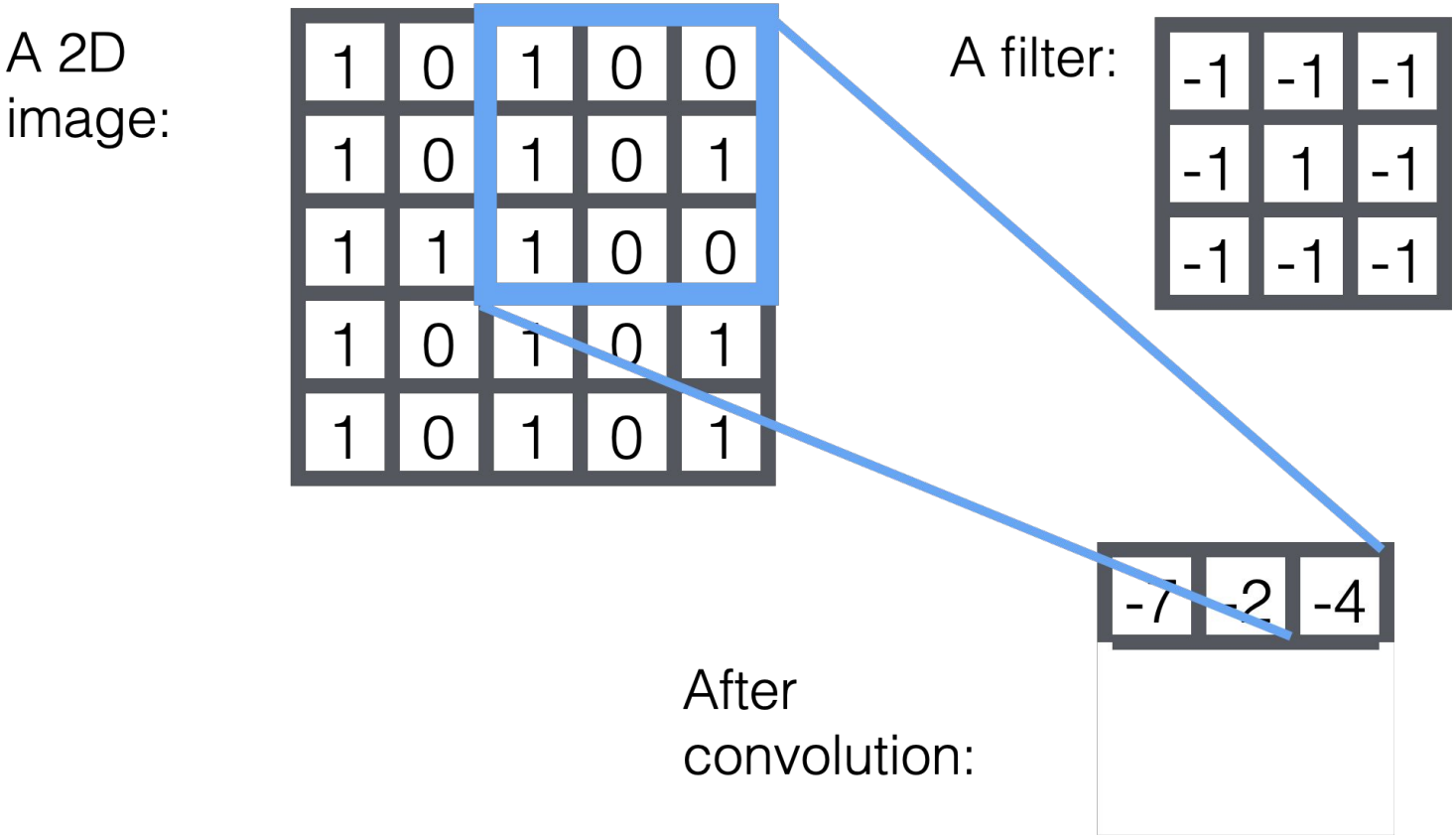
A filter:



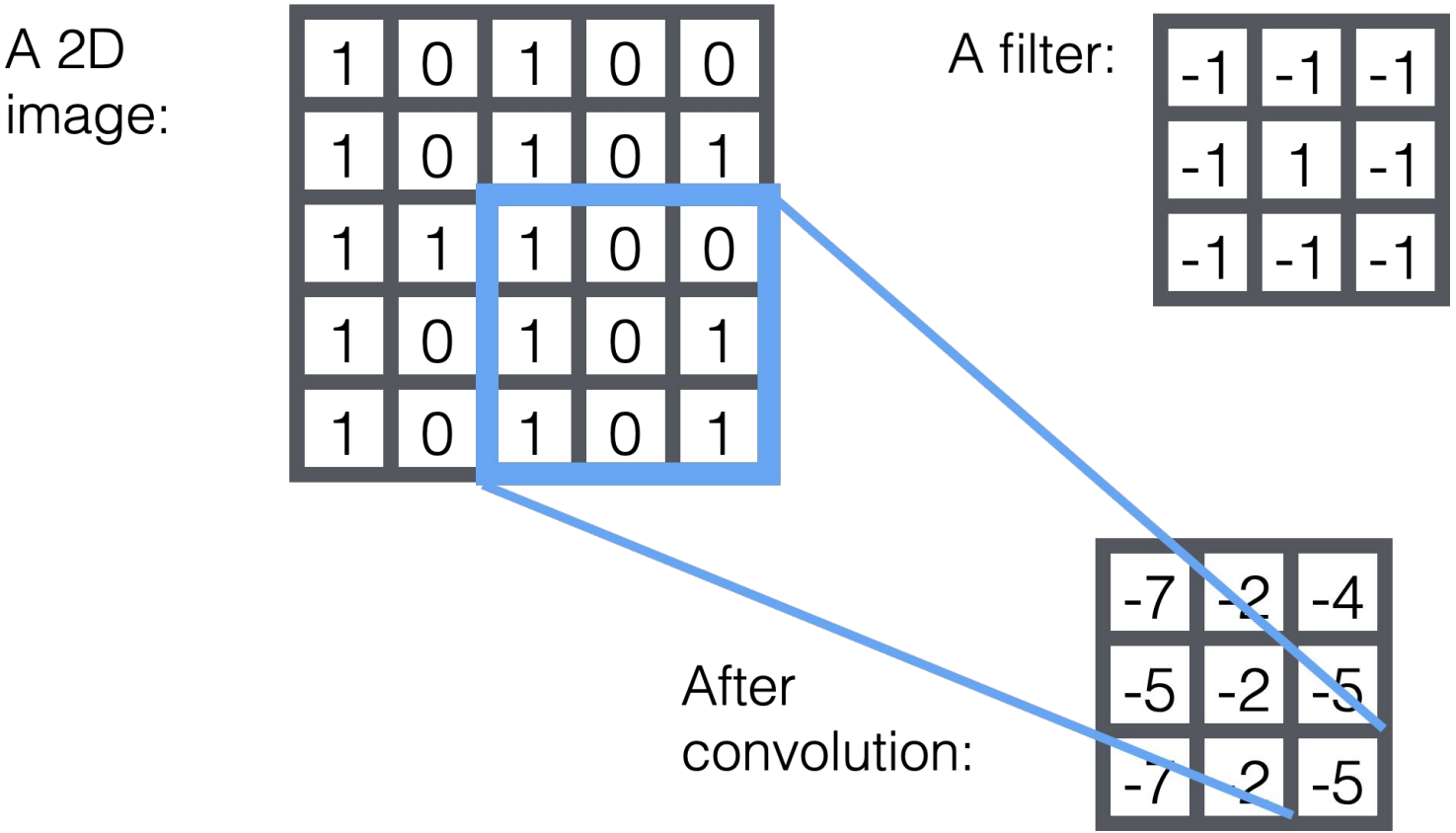
After convolution:



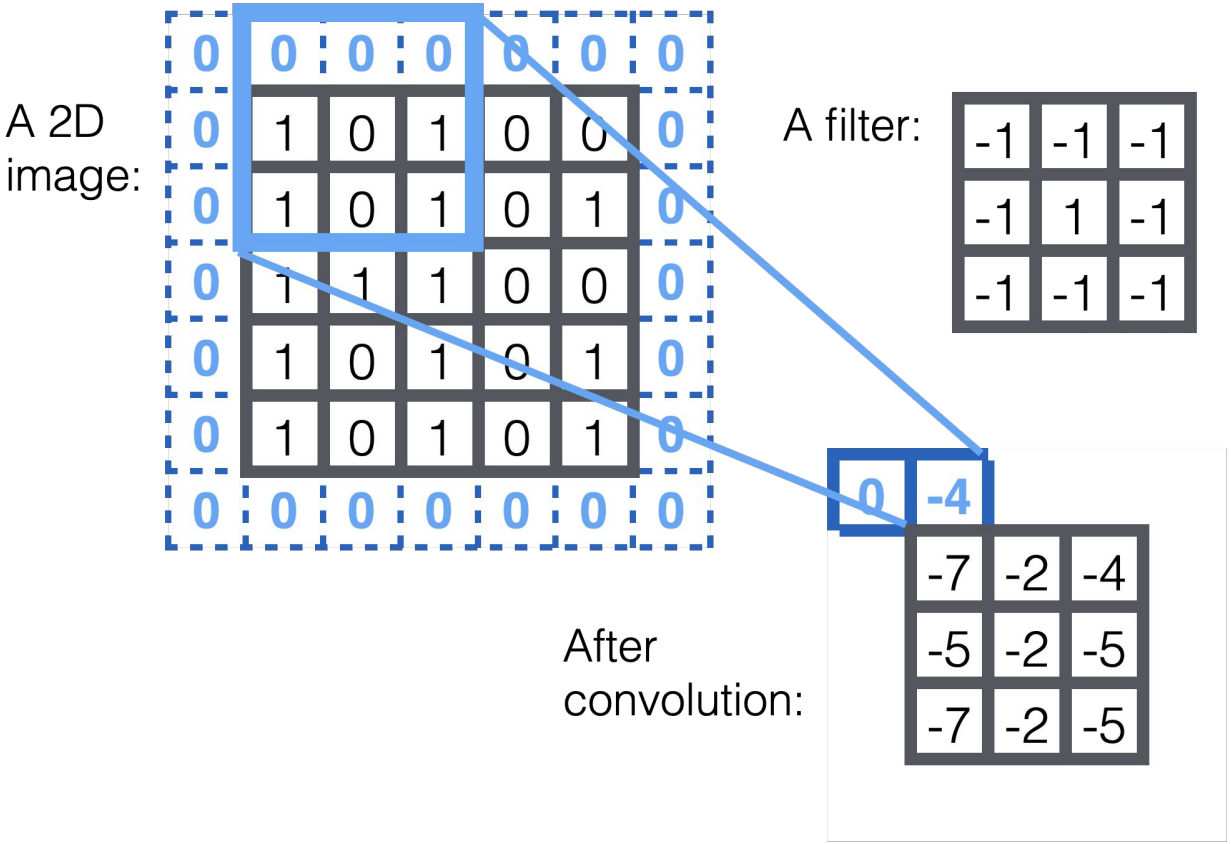
Convolutions on Images



Convolutions on Images

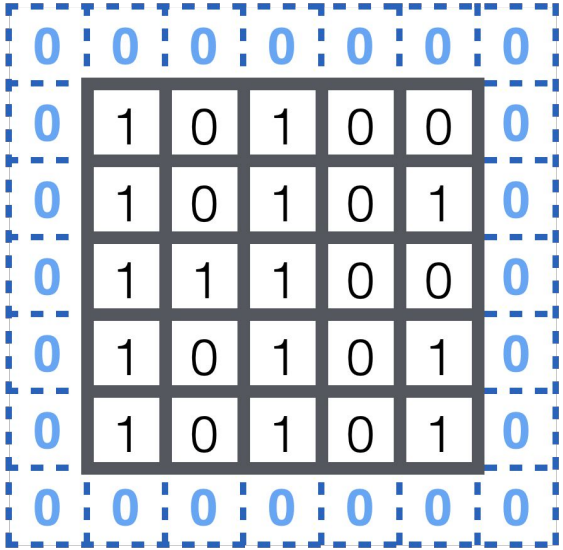


Convolutions on Images



Convolutions on Images

A 2D image:



A filter:

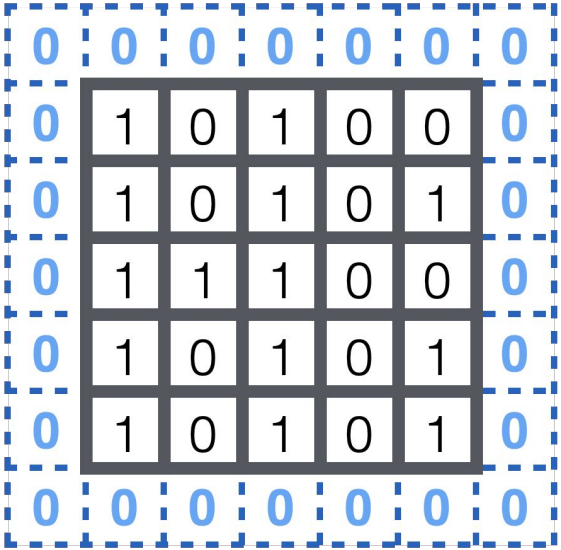
-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

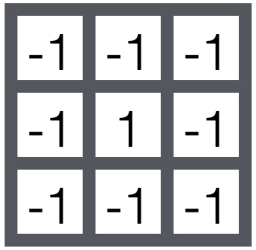
0	-4	0	-3	-1
-2	-7	-2	-4	1
-2	-5	-2	-5	-2
-2	-7	-2	-5	0
0	-4	0	-4	0

Convolutions on Images

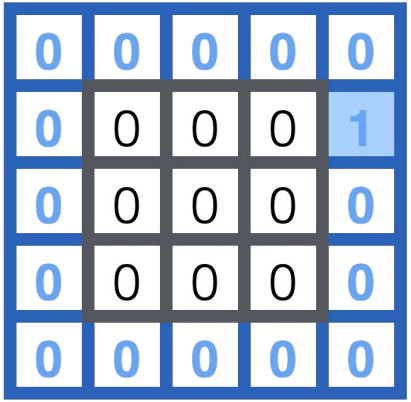
A 2D image:



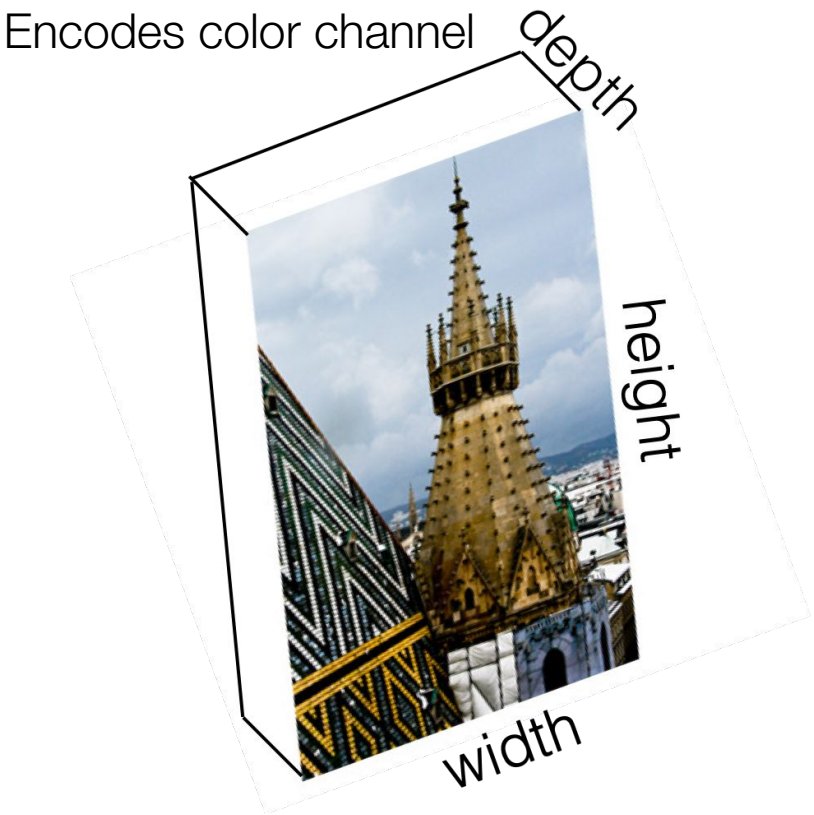
A filter:



After convolution & ReLU:

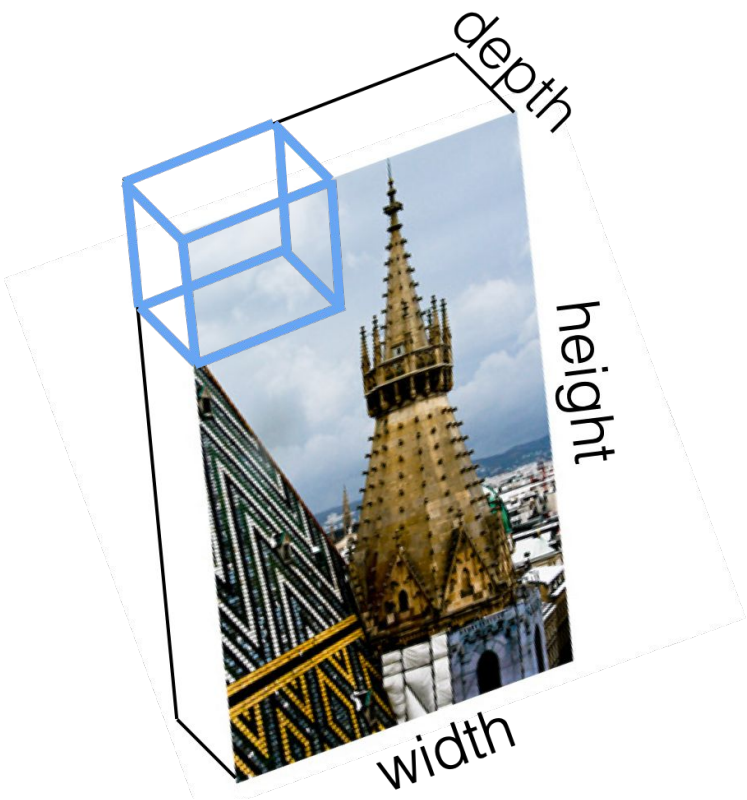


Convolutions on Images

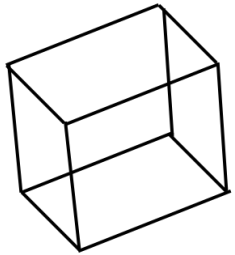


- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

Convolutions on Images

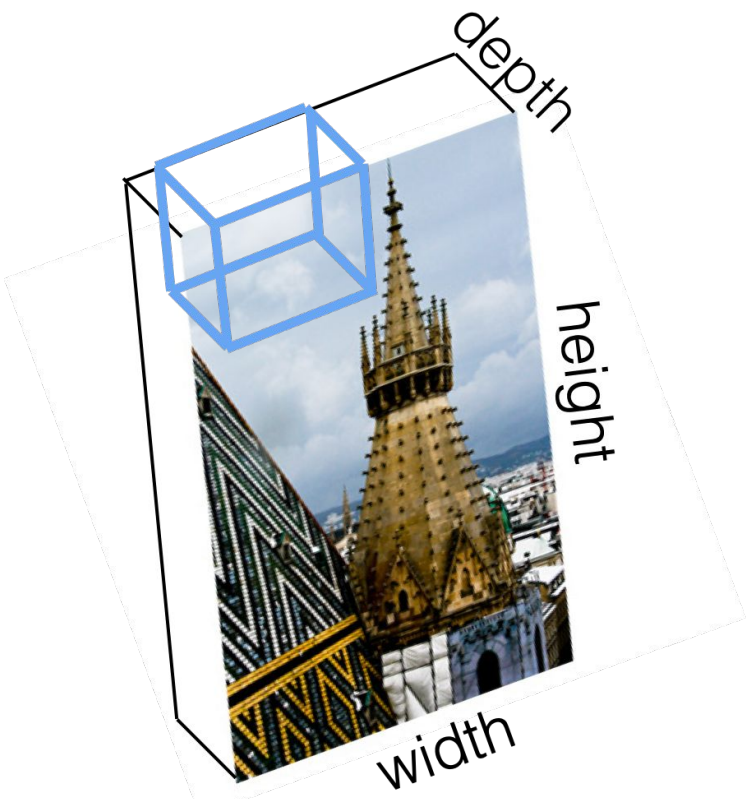


A filter:

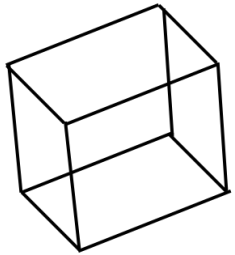


- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

Convolutions on Images

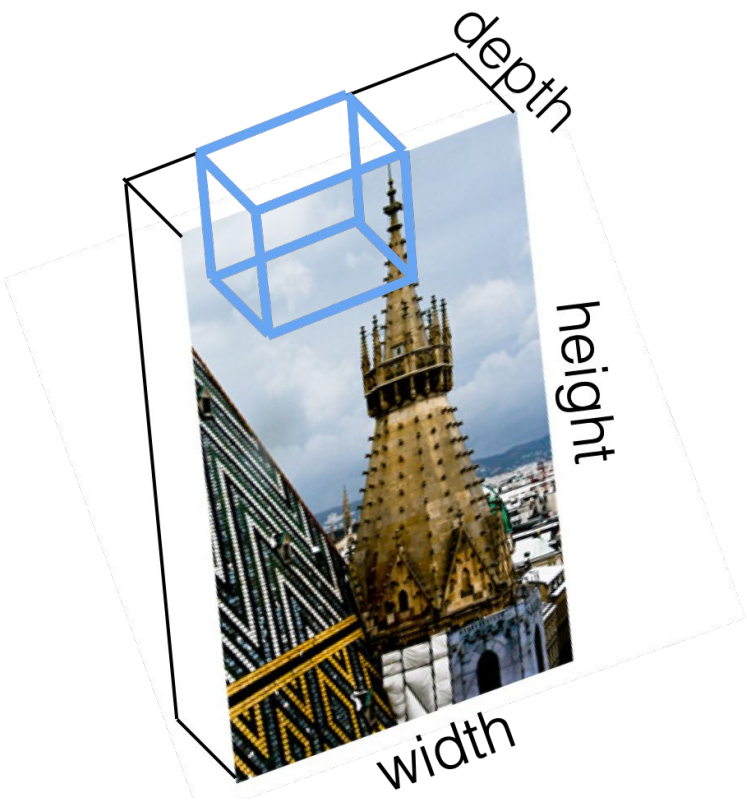


A filter:

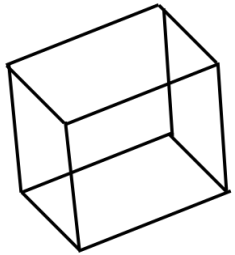


- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

Convolutions on Images

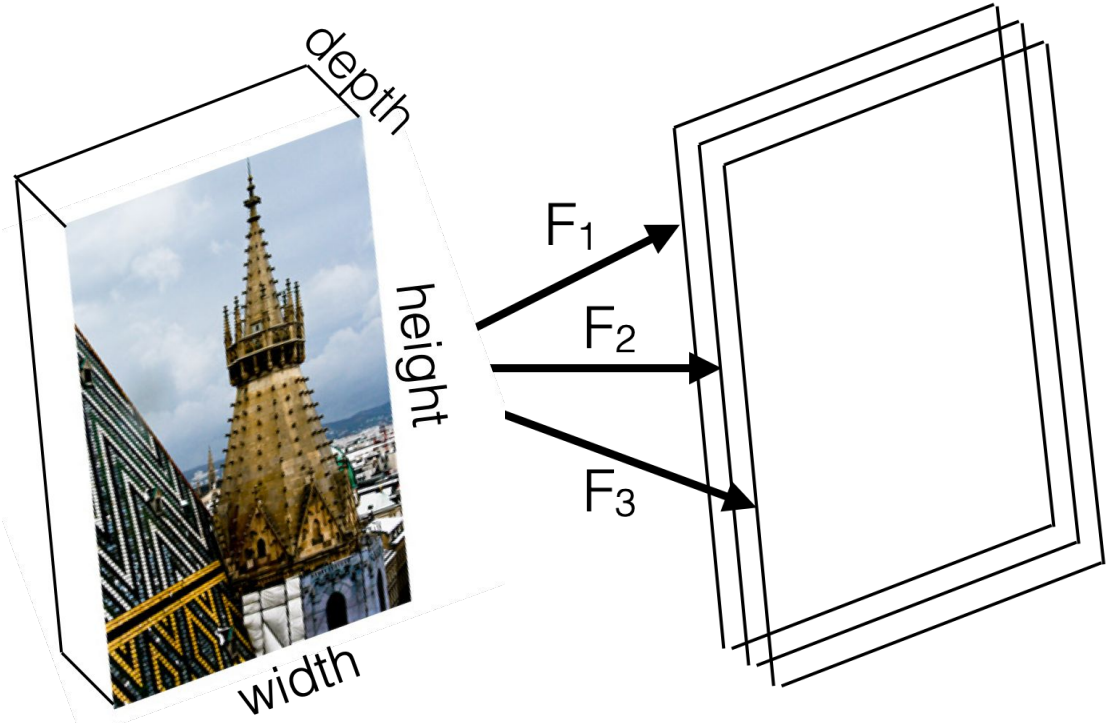


A filter:



- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

Convolutions on Images



- Input : height x width x depth
- Each fiter F produces height x width “hidden layer”
- Multiple filters (“filter bank”) produce a layer of dimension height x width x number of filters

Specifying Each Layer

Denote current layer with l

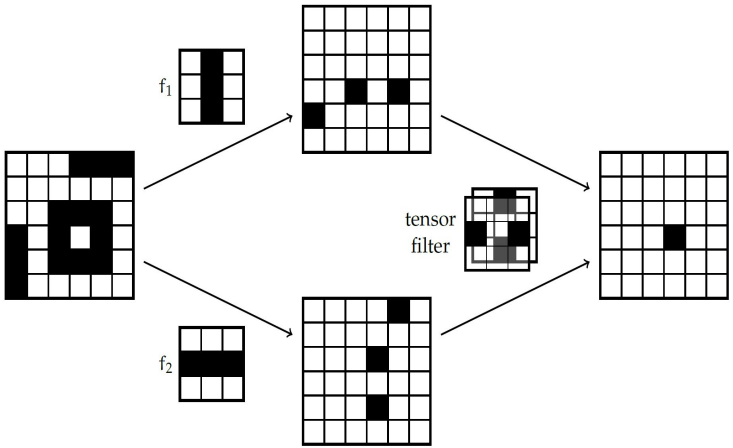
Input: $n^{l-1} \times n^{l-1} \times m^{l-1}$

Number of filters: m^l

Size of filters: $k^l \times k^l \times m^{l-1}$

Stride: s^l

Output: $n^l \times n^l \times m^l$, where $n^l = n^{l-1} / s^l$



Study Question

1) How many weights are in a convolutional layer specified as below?
2) If we used a fully-connected layer with the same size inputs and outputs, how many weights would it have?

Denote current layer with l

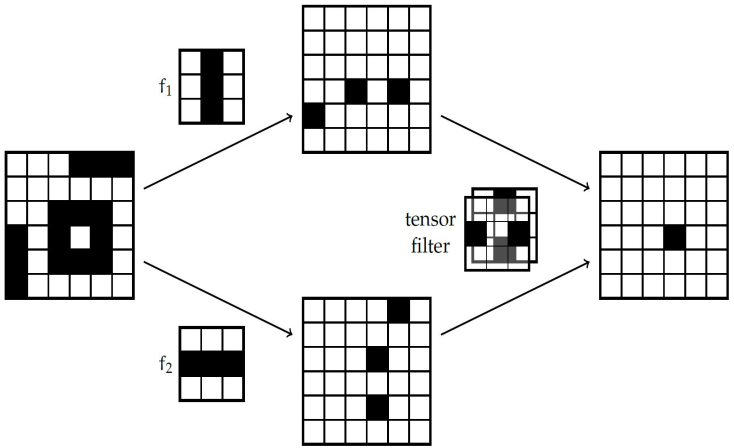
Input: $n^{l-1} \times n^{l-1} \times m^{l-1}$

Number of filters: m^l

Size of filters: $k^l \times k^l \times m^{l-1}$

Stride: s^l

Output: $n^l \times n^l \times m^l$, where $n^l = n^{l-1} / s^l$



Max Pooling

Output from the convolutional layer & ReLU:

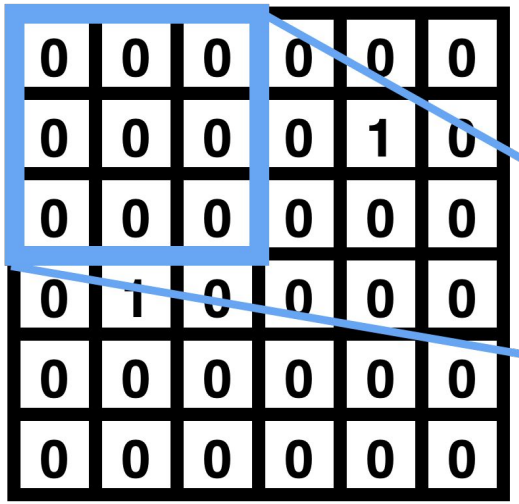
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

Max Pooling

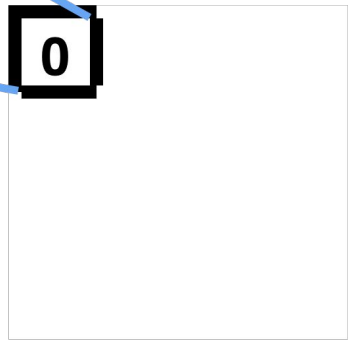
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

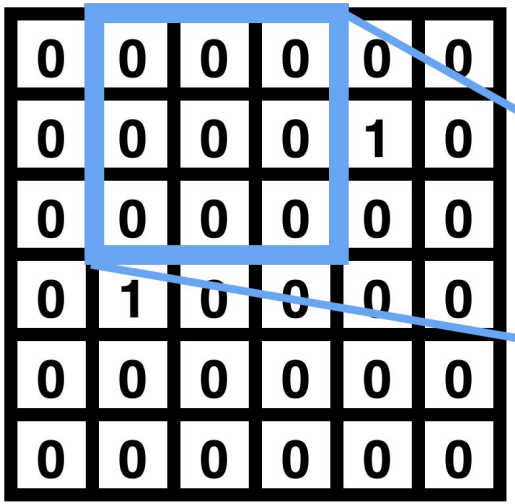
- E.g. size 3x3 (“size 3”)

After max pooling:



Max Pooling

Output from the convolutional layer & ReLU:



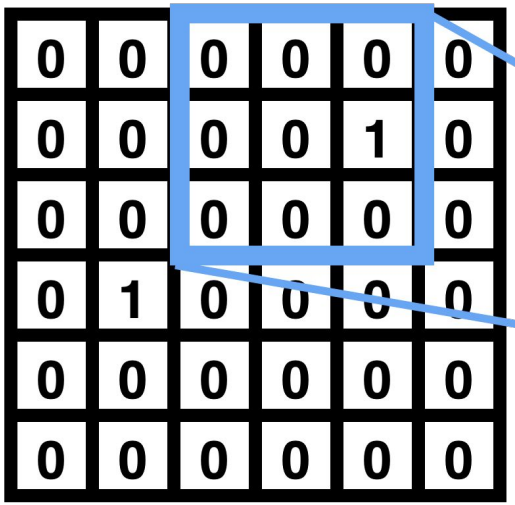
Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

After max pooling:

Max Pooling

Output from the convolutional layer & ReLU:



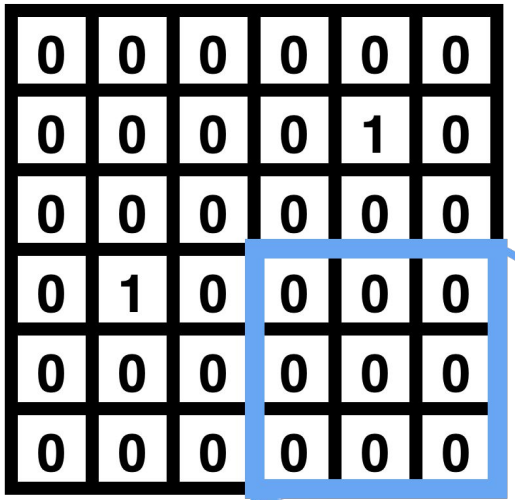
Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

After max pooling:

Max Pooling

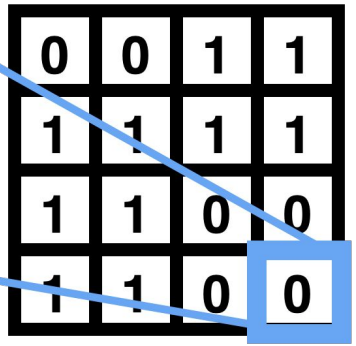
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

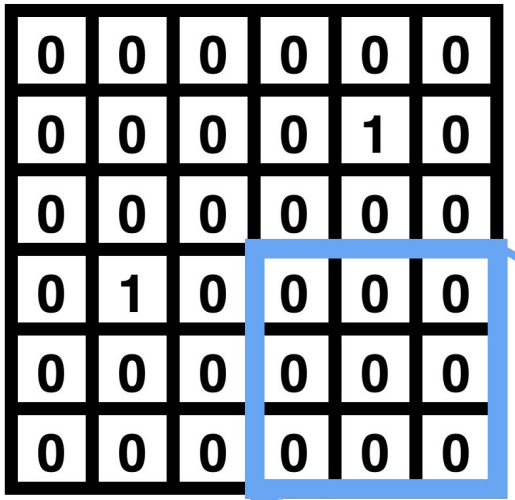
- E.g. size 3x3 (“size 3”)

After max pooling:



Max Pooling

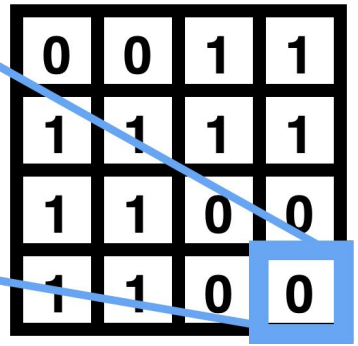
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

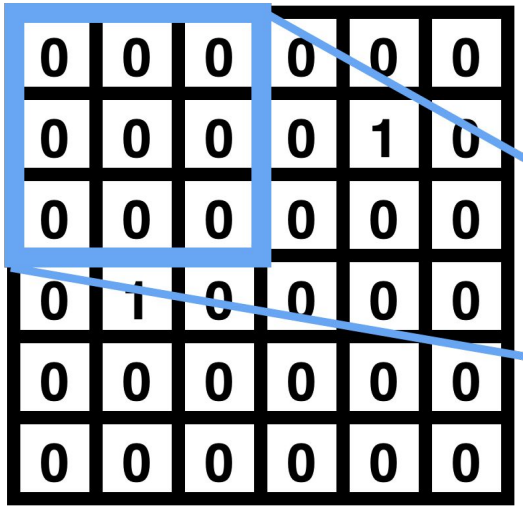
After max pooling:



“Stride” 1 pooling

Max Pooling

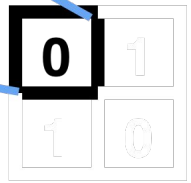
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

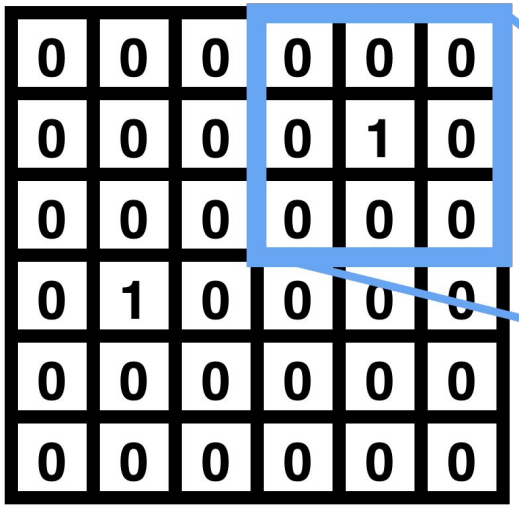
After max pooling:



“Stride” 3 pooling

Max Pooling

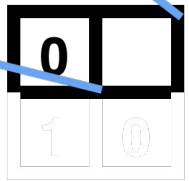
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

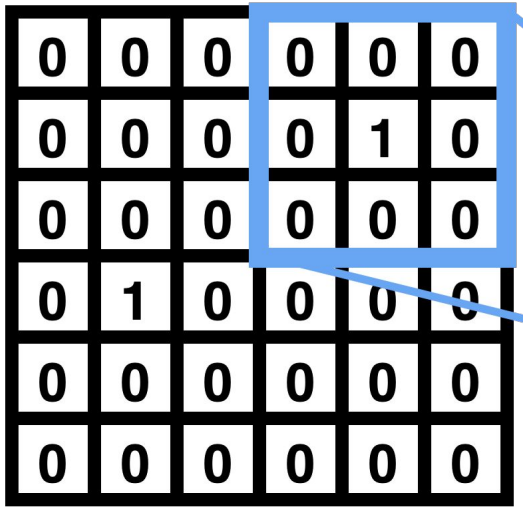
After max pooling:



“Stride” 3 pooling

Max Pooling

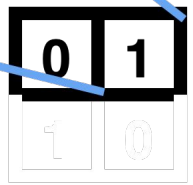
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

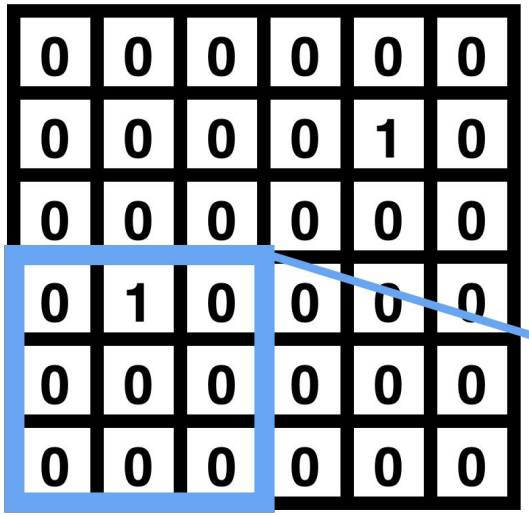
After max pooling:



“Stride” 3 pooling

Max Pooling

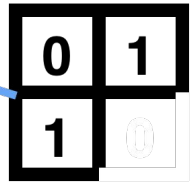
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

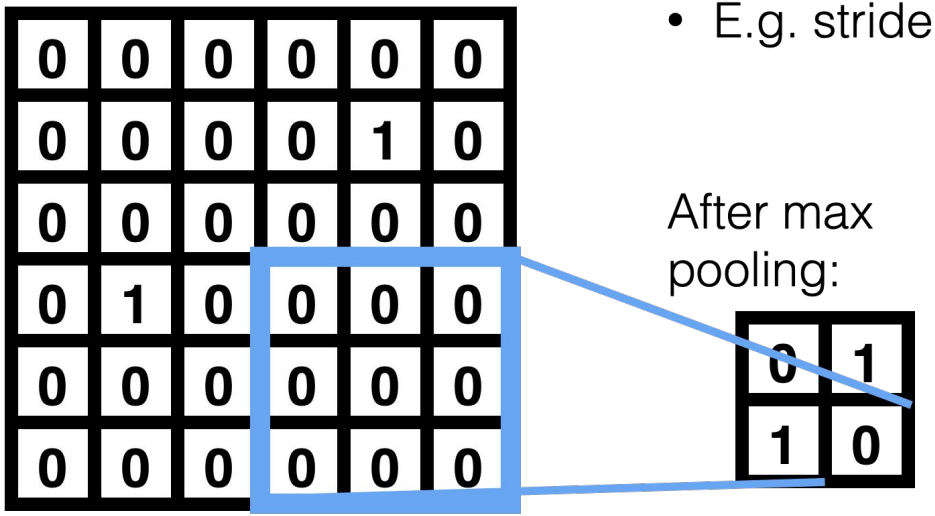
After max pooling:



“Stride” 3 pooling

Max Pooling

Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

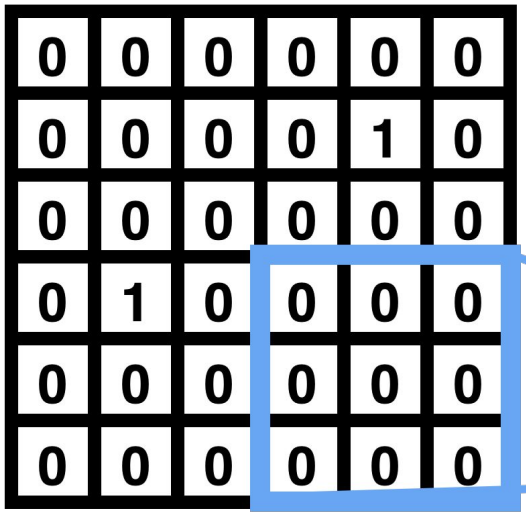
- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

“Stride” 3 pooling

Max Pooling

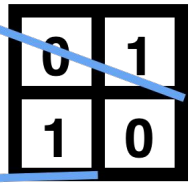
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

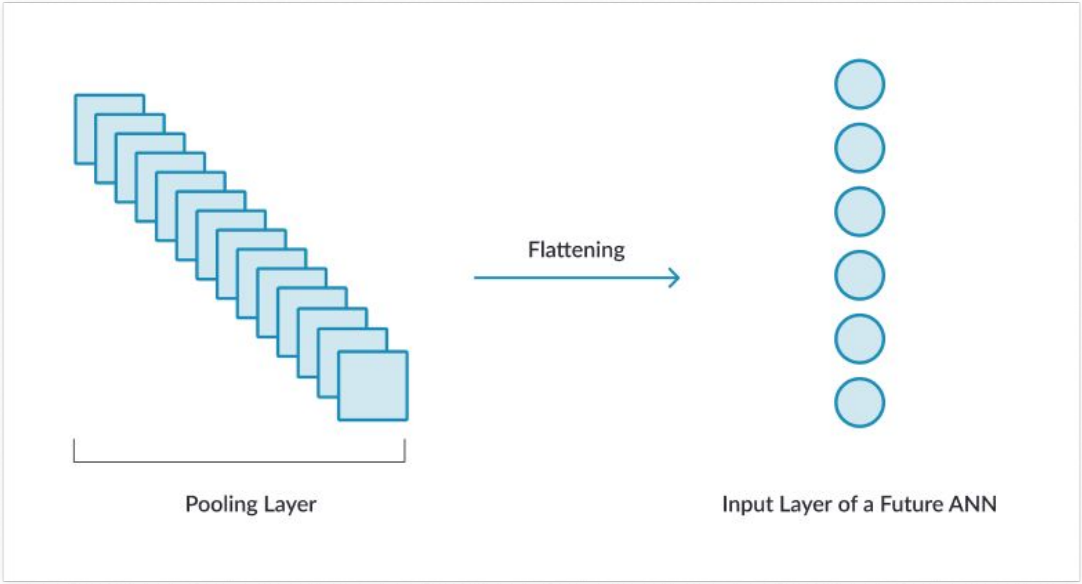


“Stride” 3 pooling

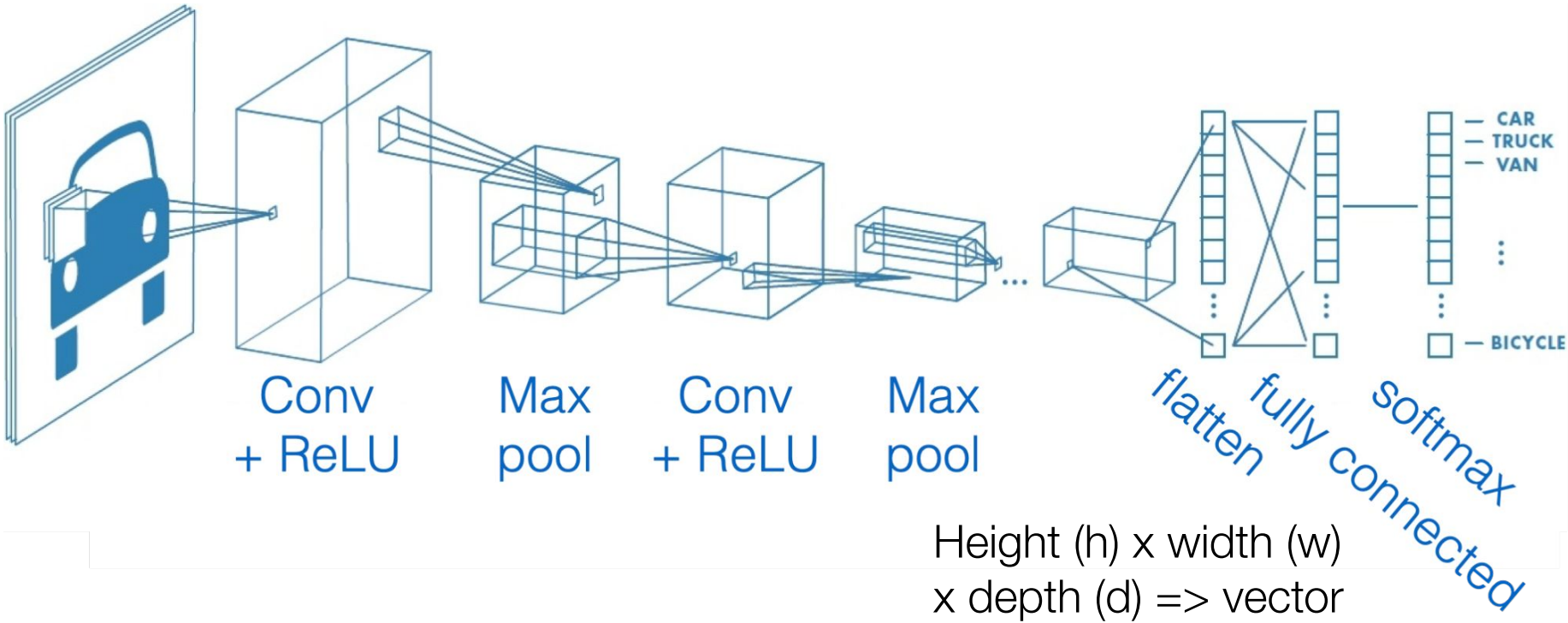
No parameters in max pooling layer!

Flatten

Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier

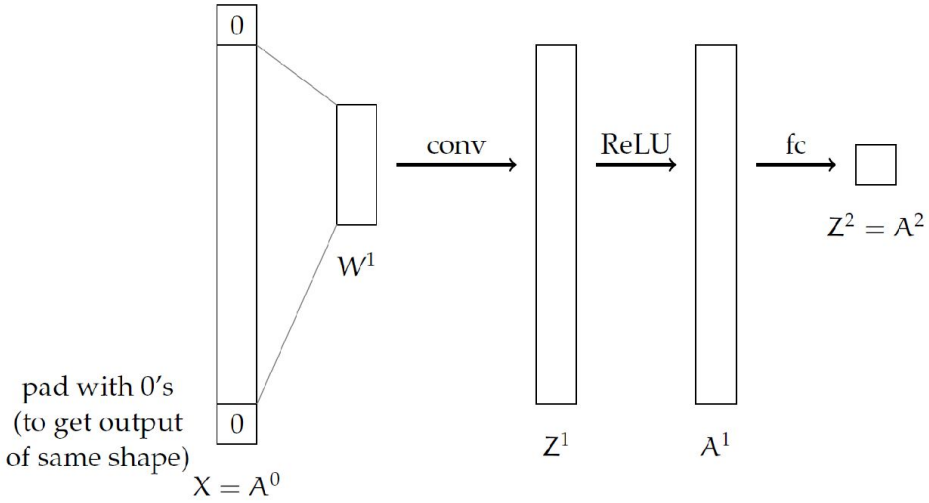


Putting it all together



Height (h) x width (w) x depth (d) => vector of length h*w*d

Simple CNN: Forward Pass



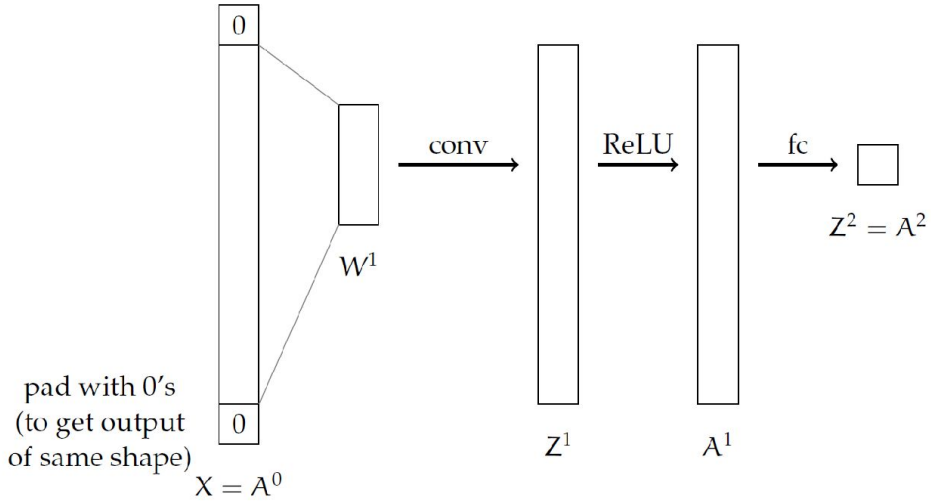
$$Z_i^1 = {W^1}^T \cdot A^0_{[i-\lfloor k/2 \rfloor : i+\lfloor k/2 \rfloor]}$$

$$A^1 = \mathrm{ReLU}(Z^1)$$

$$A^2 = {W^2}^T A^1$$

$$L(A^2, y) = (A^2 - y)^2$$

Simple CNN: Backpropagation

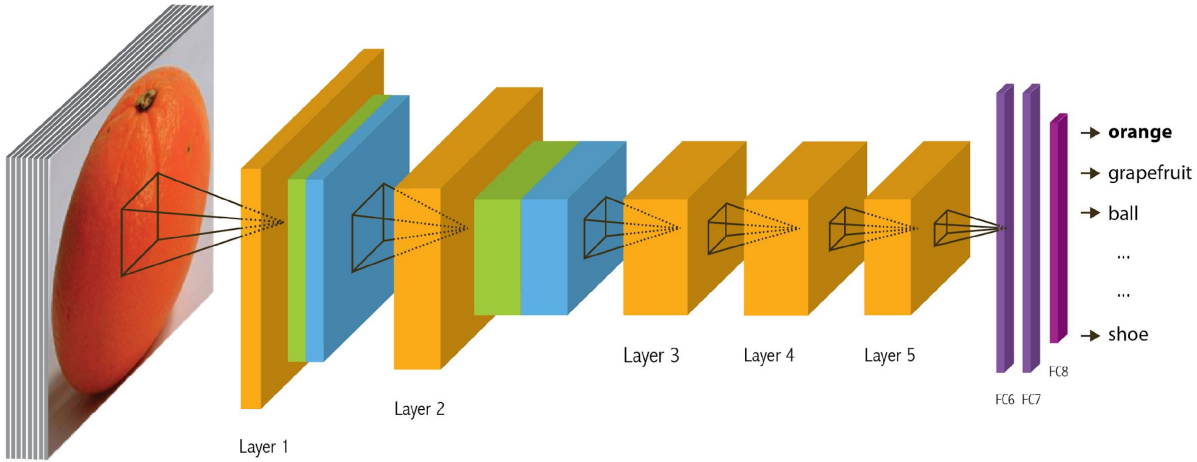


$$Z_i^1 = {W^1}^T \cdot A^0_{[i-\lfloor k/2 \rfloor : i+\lfloor k/2 \rfloor]}$$
$$A^1 = \mathrm{ReLU}(Z^1)$$
$$A^2 = {W^2}^T A^1$$
$$L(A^2, y) = (A^2 - y)^2$$

$$\frac{\partial \mathrm{loss}}{\partial W^1} = \frac{\partial Z^1}{\partial W^1} \cdot \frac{\partial A^1}{\partial Z^1} \cdot \frac{\partial \mathrm{loss}}{\partial A^1}$$

Deep Learning Today

Used both for analysis and synthesis



Input image

Advances in Dataset Collection

COIL-20

Caltech 101

2 year old kid

IMAGENET (2009)

10^3 10^4 10^5 10^6 10^7 10^8 10^9 # images

Caltech-4 (2003)

PASCAL (2005)

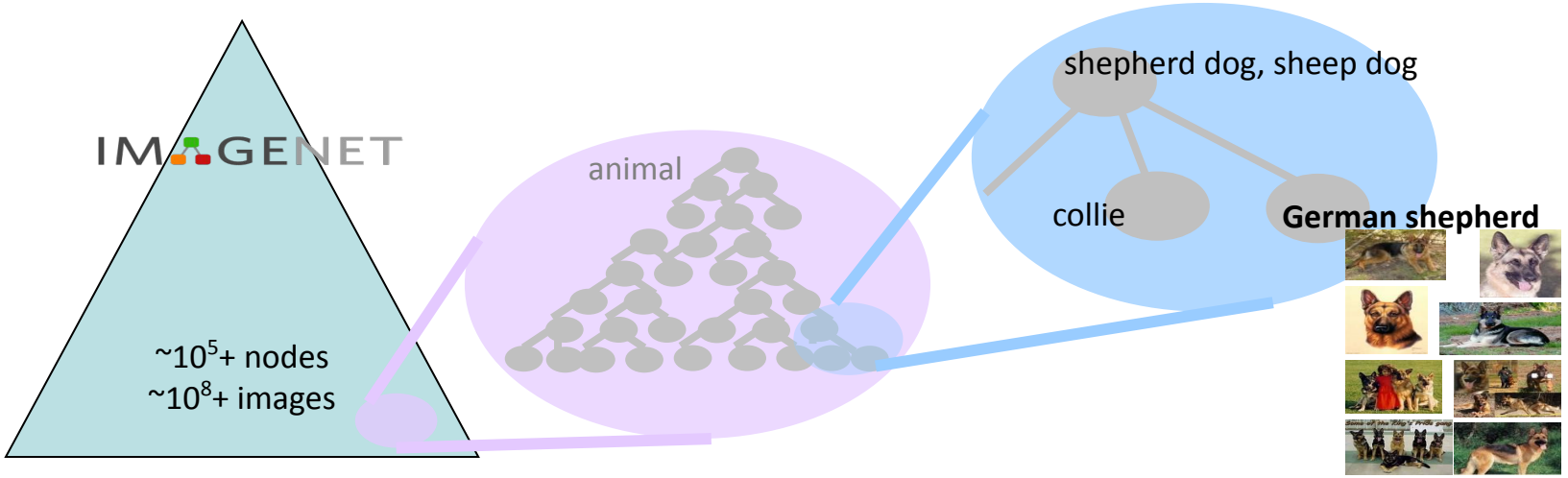
Places (2013)

IMAGENET

An ontology of images based on WordNet

ImageNet currently has

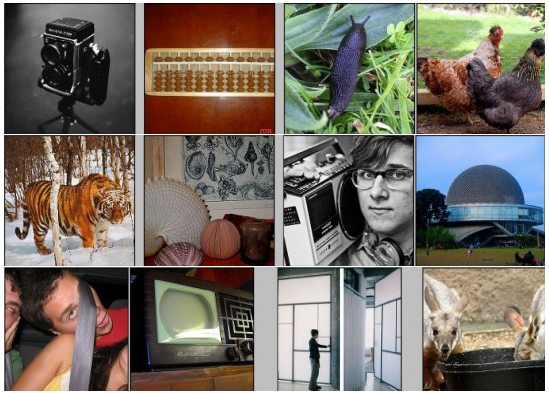
- 13,000+ categories of visual concepts
- 10 million human-cleaned images (~700im/categ)
- 1/3+ is released online @ www.image-net.org



Deng, Dong, Socher, Li & Fei-Fei, CVPR 2009

Application to ImageNet

IMAGENET



[Deng et al. CVPR 2009]

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk

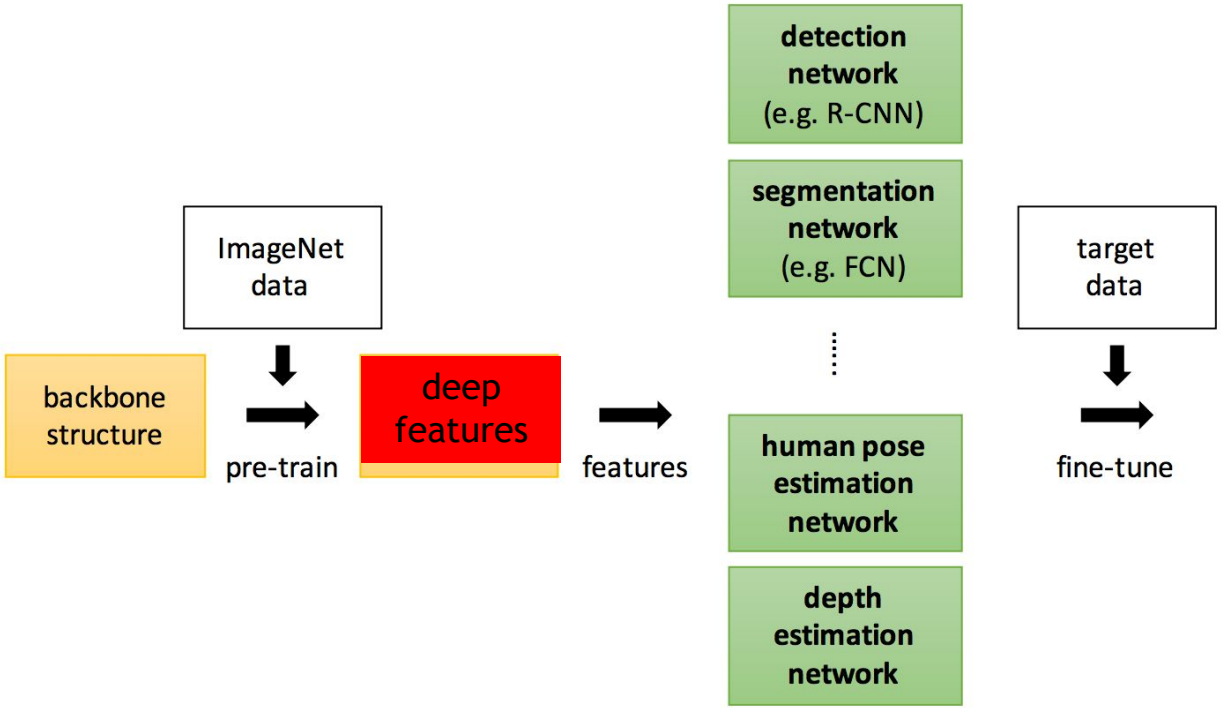
ImageNet Classification with Deep Convolutional Neural Networks [NIPS 2012]

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

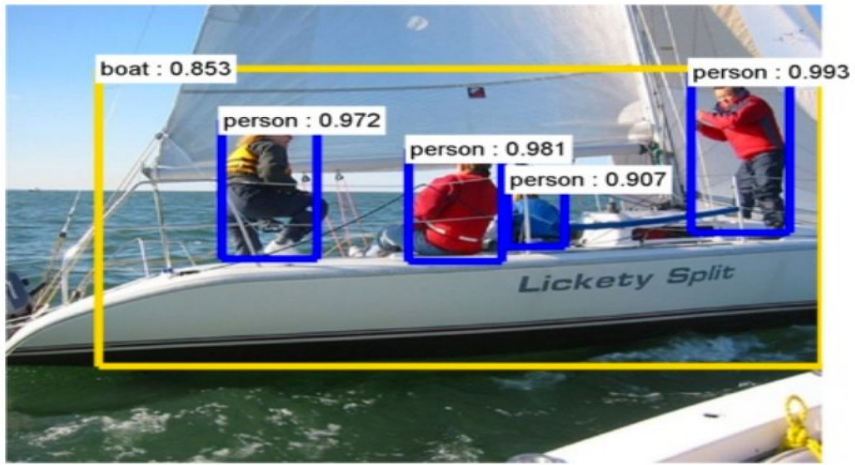
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Solving Different Tasks



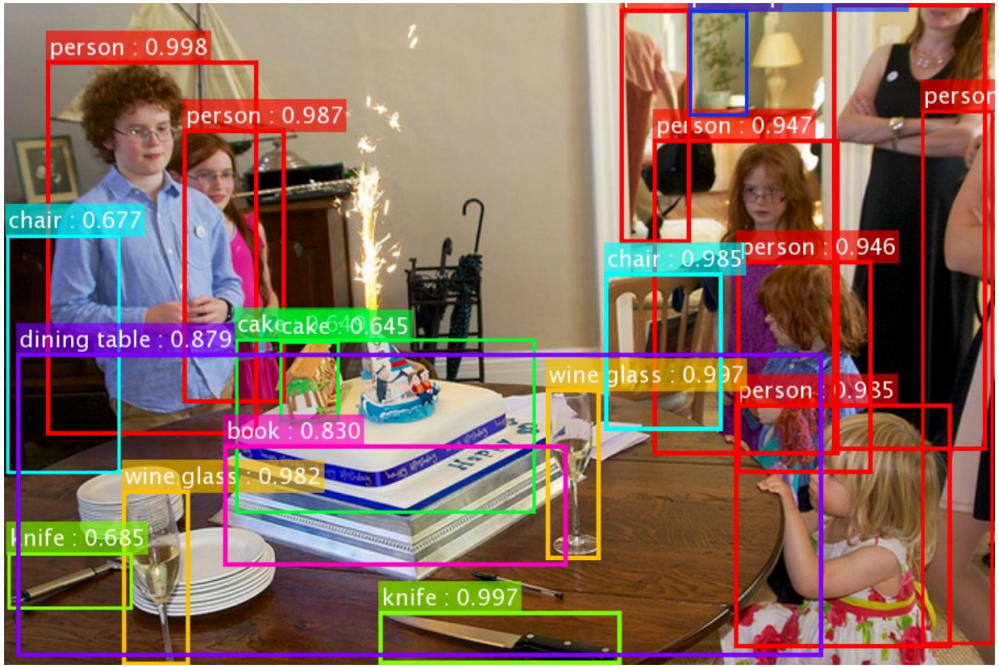
Slide Credit: http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Object Detection: What and Where



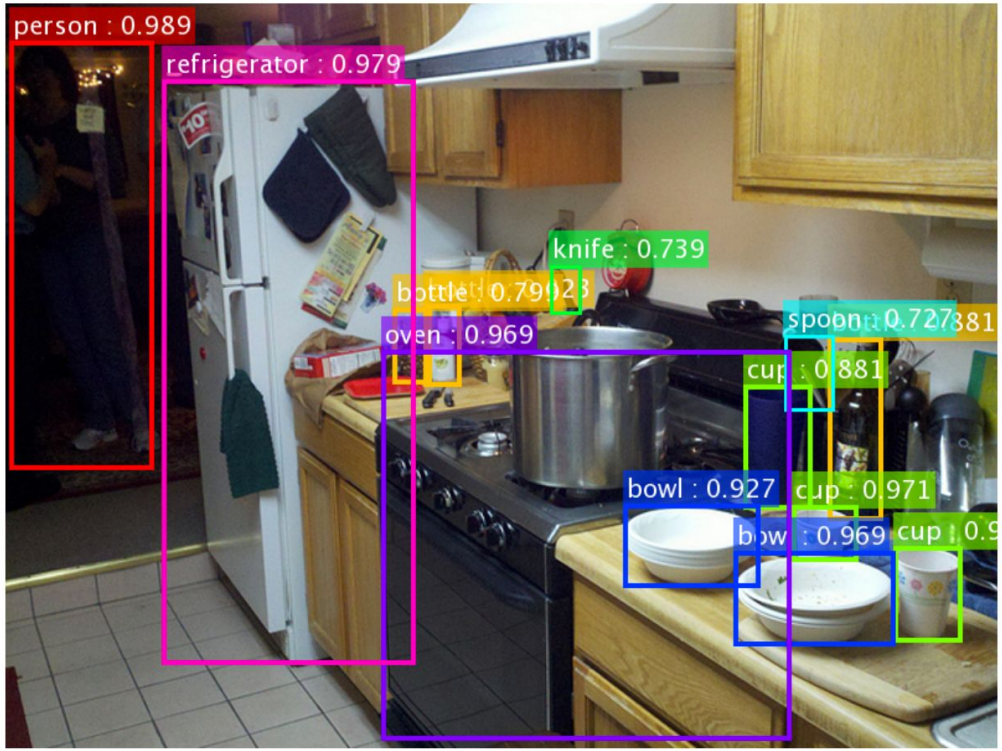
Slide Credit: http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Object Detection Results



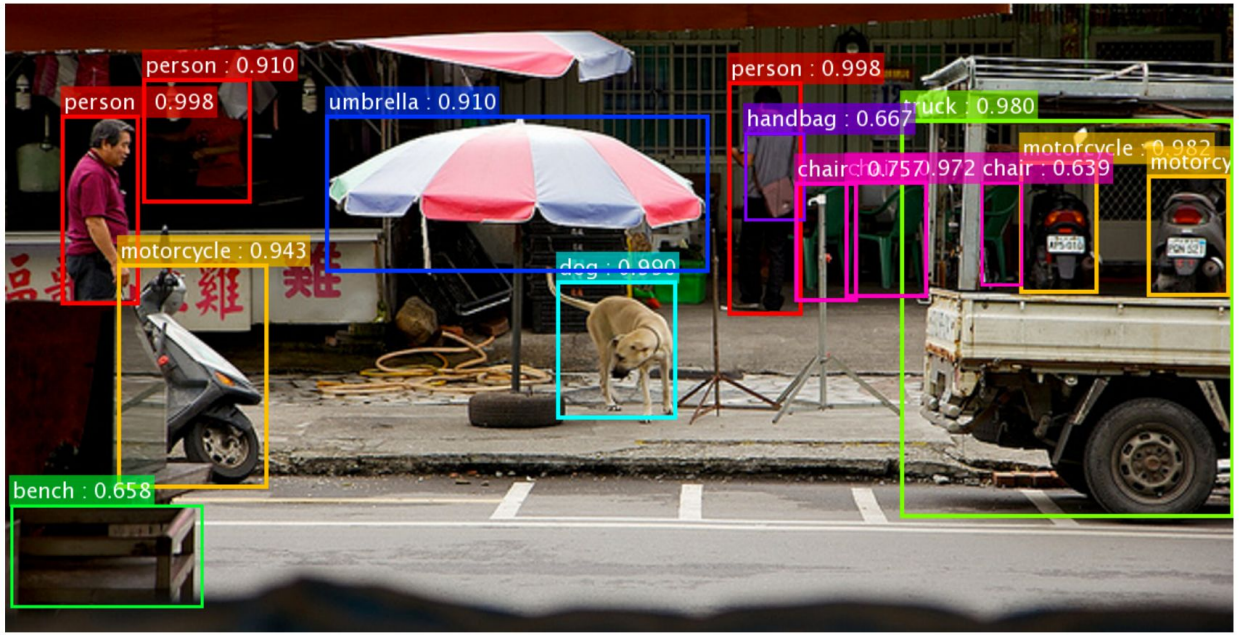
Slide Credit: http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Object Detection Results



Slide Credit: http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Object Detection Results

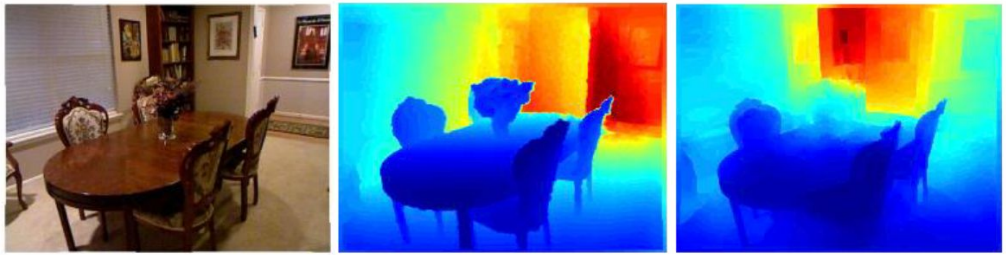


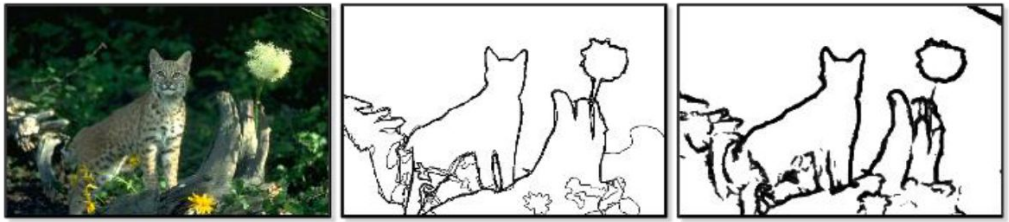
Slide Credit: http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Segmentation, Depth, & More

semantic segmentation

monocular depth estimation (Liu et al. 2015)





boundary prediction (Xie & Tu 2015)