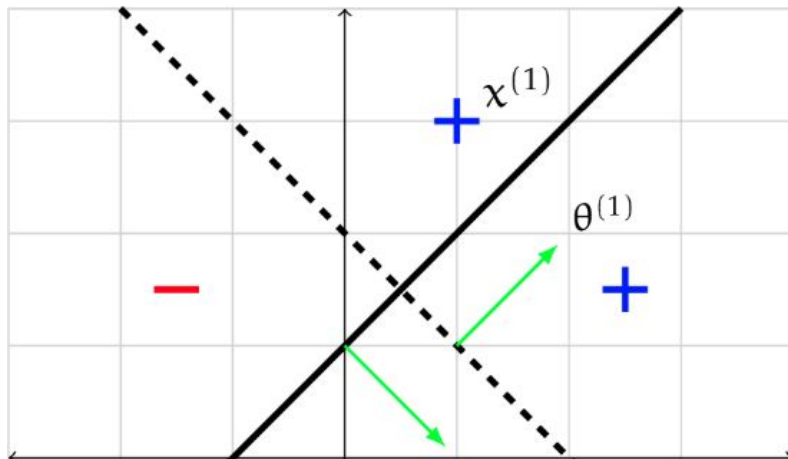


Introduction to Machine Learning

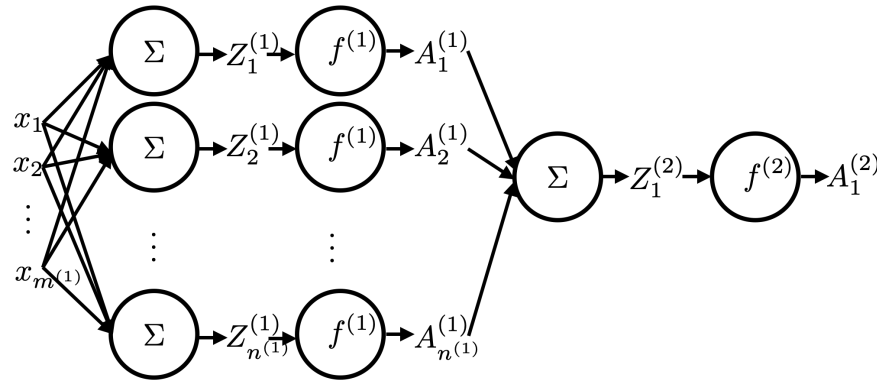


Convolutional Neural Networks

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?

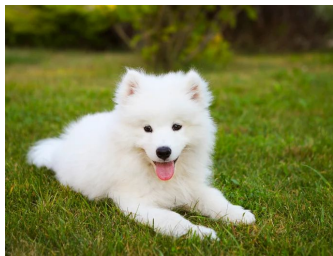
Universal Function Approximator:

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

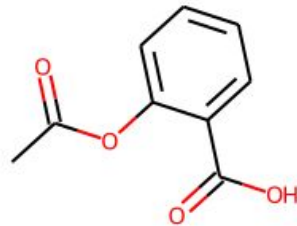
We want to inject structural knowledge about the input domain into our neural network.



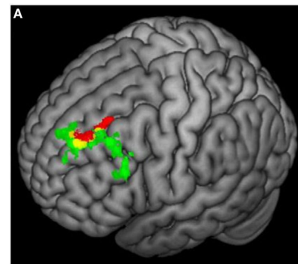
Image



Audio



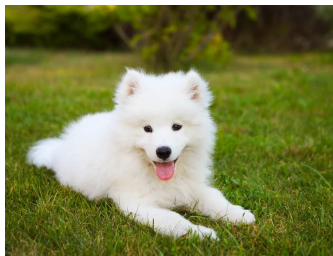
Molecules



fMRI

Neural Network Architectures

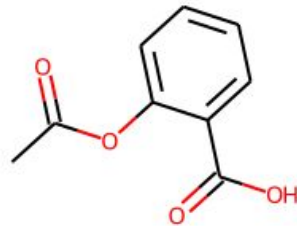
We want to inject structural knowledge about the input domain into our neural network.



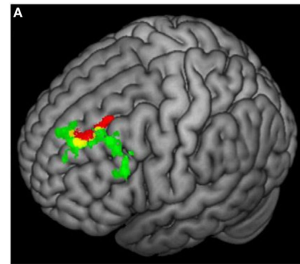
Image



Audio



Molecules

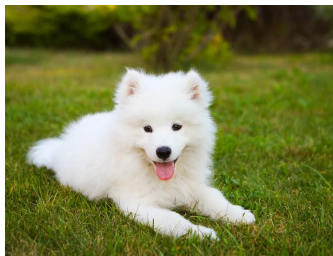


fMRI

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

Neural Network Architectures

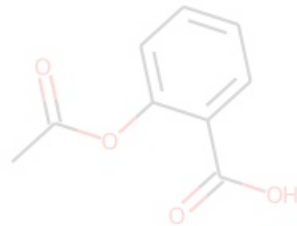
We want to inject structural knowledge about the input domain into our neural network.



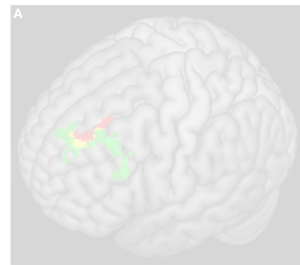
Image



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?

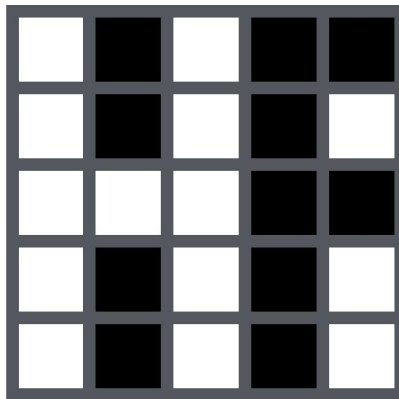
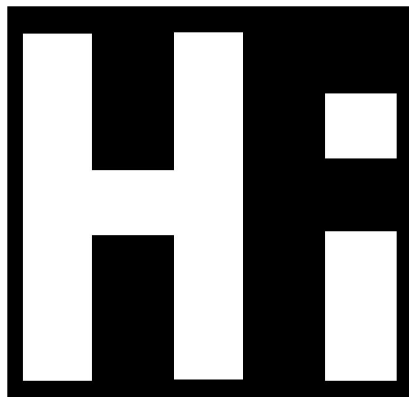


How do computers see?



62	62	63	64	65	66	67	68	69	70	71	72	72	73	73	73	72	72	71	70	69	67	66	66	65	63	62	61	60	6		
61	62	63	64	66	66	67	68	68	69	70	71	71	72	72	73	72	72	71	71	70	69	68	66	66	65	63	62	61	60	6	
61	62	63	64	66	66	68	68	69	70	71	72	73	73	73	72	72	71	71	69	68	67	66	66	65	65	64	63	62	61	60	6
61	63	64	64	66	67	68	68	69	70	71	71	73	73	74	73	73	71	70	69	68	66	66	65	64	63	62	61	61	6	6	
61	63	64	65	67	68	69	69	70	71	71	72	55	53	69	72	72	71	71	70	69	68	67	66	65	64	63	62	60	60	6	
63	64	65	66	67	68	69	69	70	71	72	42	4	5	11	48	72	71	71	69	69	68	67	66	65	64	62	62	60	59	5	
63	65	66	66	68	68	69	70	71	71	72	18	4	7	8	66	71	70	69	68	68	67	66	65	64	63	61	59	59	5	5	
63	65	67	67	68	69	69	70	71	71	72	64	4	27	24	54	33	29	52	64	68	68	67	66	65	64	63	62	61	59	58	5
64	65	66	66	68	69	70	71	41	24	24	12	17	24	48	60	37	43	36	52	66	68	67	66	65	64	63	61	60	59	58	5
65	66	67	67	68	69	71	49	6	6	6	5	34	36	12	47	34	17	29	54	43	63	67	66	65	64	63	62	60	59	58	5
64	65	66	66	68	69	38	6	6	5	5	7	16	19	4	47	44	27	24	40	67	66	66	65	65	64	63	61	60	59	58	5
63	64	65	65	67	30	6	6	5	5	5	6	8	9	20	27	51	78	41	44	66	65	65	65	65	64	63	62	60	59	58	5
63	64	65	65	34	5	5	5	5	5	5	4	19	6	7	54	64	20	59	65	65	64	64	64	63	62	61	60	59	57	5	
63	64	64	65	14	5	6	5	5	4	5	4	18	7	5	4	19	10	11	65	64	64	64	63	61	66	62	61	60	59	58	5
63	64	64	65	53	7	4	5	6	6	7	10	6	5	5	4	21	24	18	64	64	64	63	62	64	65	62	62	60	59	58	5
64	64	64	64	65	50	4	4	4	5	11	16	6	6	4	6	35	16	26	66	64	64	63	61	72	67	63	62	61	59	58	5
64	64	64	64	65	46	4	4	4	5	6	9	8	5	29	10	43	56	29	57	64	64	63	61	70	67	62	64	65	59	59	5
64	64	64	65	66	27	5	4	4	5	6	6	6	18	66	20	57	60	46	36	75	70	62	61	70	67	62	61	60	59	58	5
49	50	62	65	57	5	5	6	5	6	6	6	6	41	59	28	60	58	44	22	63	71	72	60	69	68	61	60	58	59	59	5
42	52	57	52	26	5	5	5	5	5	5	5	5	70	50	43	61	62	64	39	42	64	60	62	56	63	65	65	67	61	53	5
32	32	32	33	6	5	5	5	5	6	6	11	39	21	33	51	50	45	46	18	32	36	33	23	44	70	71	51	42	27	3	
50	50	51	39	5	5	5	5	6	6	6	42	69	28	34	42	39	43	37	26	29	40	26	29	26	35	42	35	33	18	1	
52	53	51	22	5	5	5	5	6	5	6	5	44	56	17	51	54	53	54	56	51	22	54	54	55	55	54	53	53	52	5	
54	54	53	8	5	5	5	5	6	5	6	13	52	42	21	51	54	51	49	49	50	22	41	45	42	42	41	40	41	44	43	4
52	52	54	36	8	5	5	6	6	5	6	26	55	32	32	54	53	51	51	51	51	44	25	51	51	49	49	50	49	48	46	4
54	54	52	53	30	7	5	6	5	6	40	54	29	52	51	53	56	55	52	52	51	38	52	52	50	49	46	46	45	45	4	
51	52	51	53	27	14	5	4	5	4	7	47	51	21	39	49	47	49	52	52	49	35	31	48	46	47	47	47	46	46	4	
48	50	51	53	25	14	17	8	4	4	17	46	40	18	43	47	46	49	52	54	53	53	54	18	50	49	46	47	47	47	4	
49	49	49	49	22	12	20	24	6	14	35	51	39	48	48	50	51	51	49	51	51	52	50	41	58	48	47	47	47	45	45	4
51	49	50	50	22	13	19	36	13	12	42	50	40	73	50	50	50	49	48	49	49	48	49	45	51	46	44	44	44	42	45	4
47	49	49	47	20	16	26	39	21	15	36	48	42	61	47	48	51	47	50	51	51	51	49	47	47	52	47	47	44	43	45	4

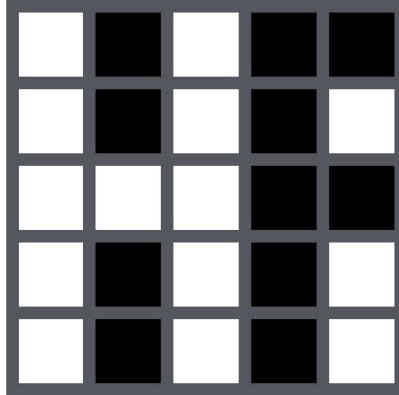
How do computers see?



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

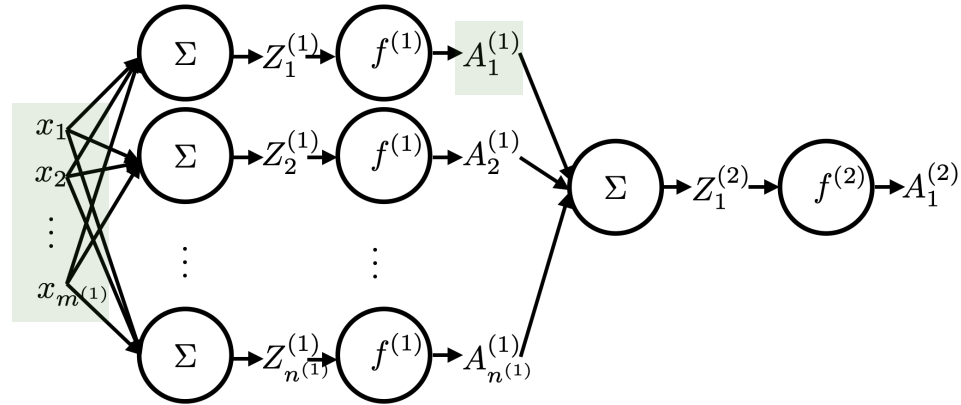
x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}

Logistic Regression/
Neural Network

Whether “H” is
in image or not

“Fully-Connected” Neural Networks

Every hidden unit is connected to the input!



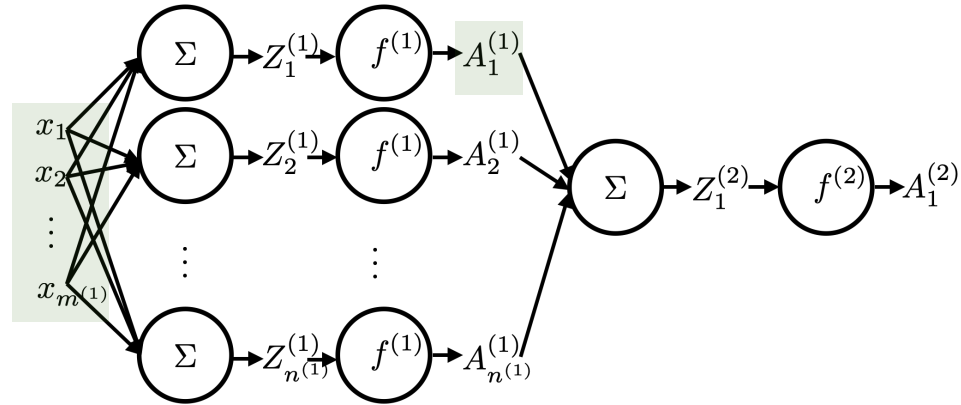
x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}

Logistic Regression/
Neural Network

Whether “H” is
in image or not

“Fully-Connected” Neural Networks

Every hidden unit is connected to the input!



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 x_7 is close to x_1 .

Image Structure

What do we know about images?

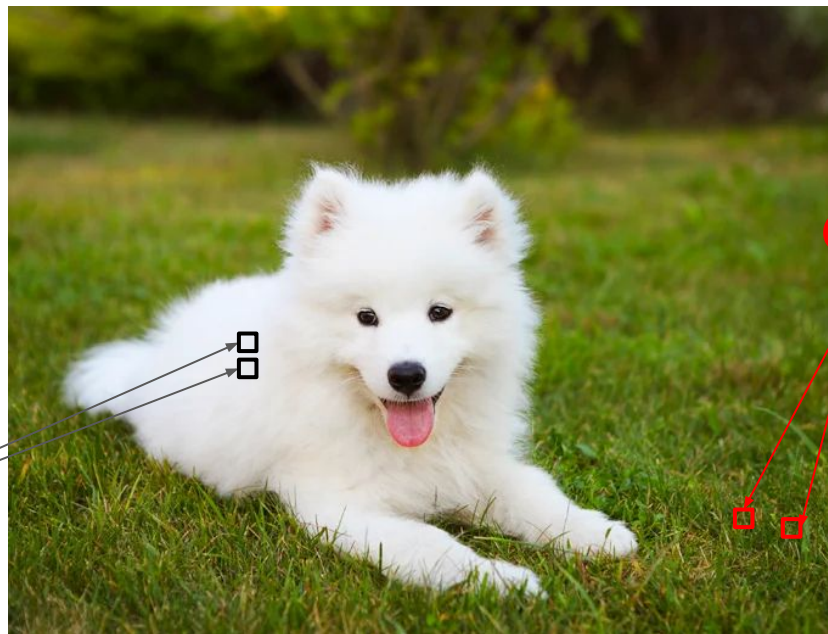


Image Structure

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)

Fur



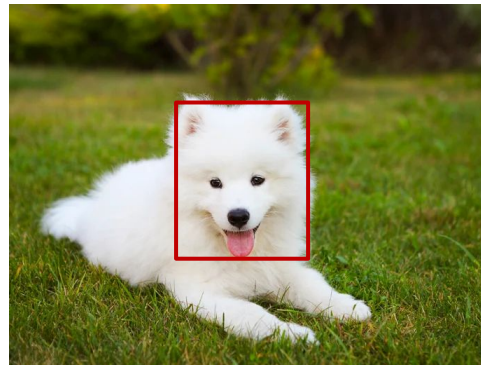
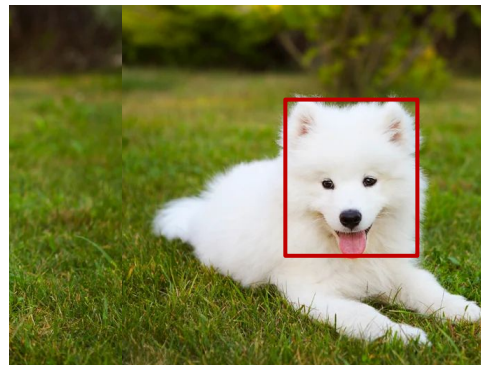
Grass

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

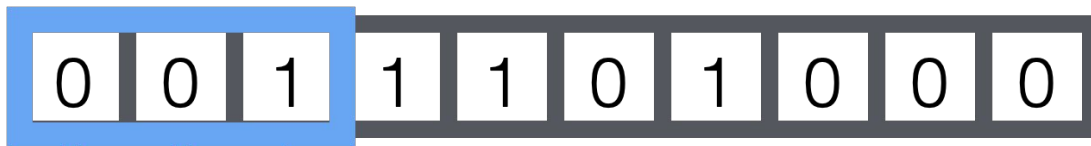


Convolutional Filter
(model parameters)

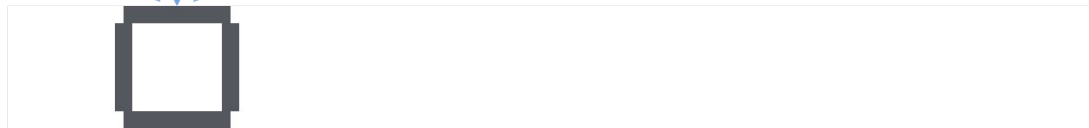


Convolutional Layer

1D sequence

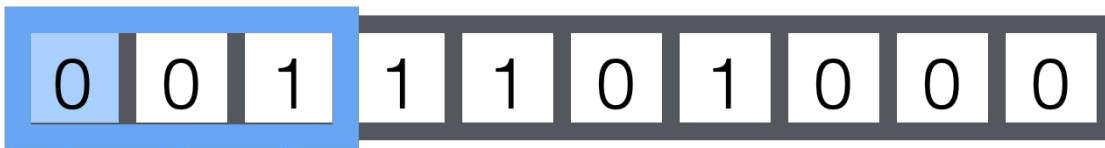


Convolutional Filter
(model parameters)



Convolutional Layer

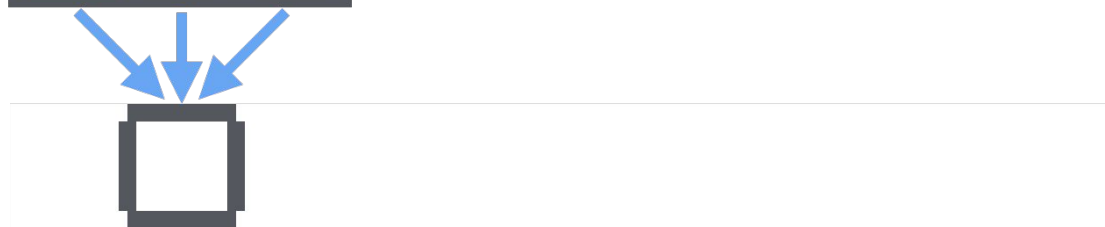
1D sequence



Convolutional Filter
(model parameters)

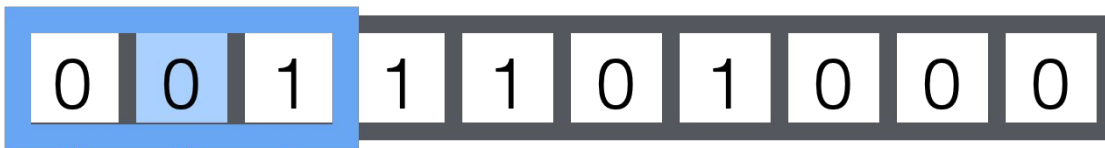


$$0 * -1$$



Convolutional Layer

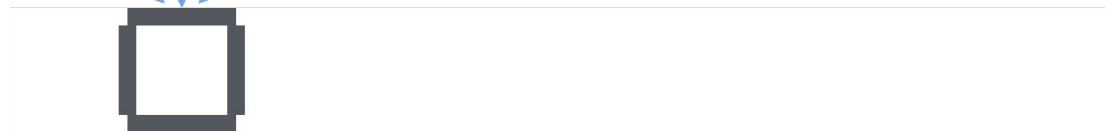
1D sequence



Convolutional Filter
(model parameters)

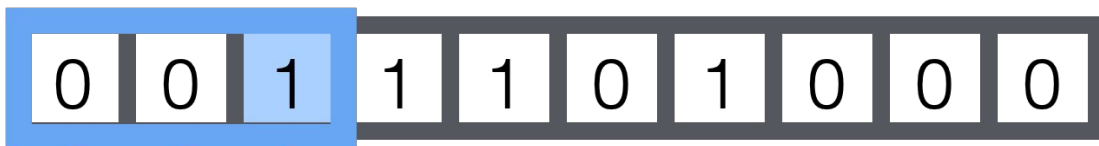


$$0 * -1 + 0 * 1$$



Convolutional Layer

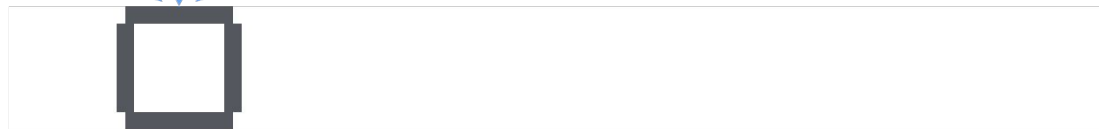
1D sequence



Convolutional Filter
(model parameters)

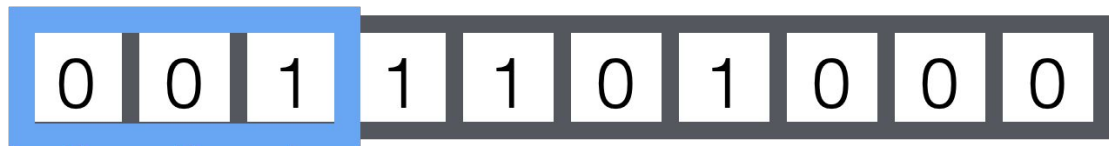


$$0 * -1 + 0 * 1 + 1 * -1$$



Convolutional Layer

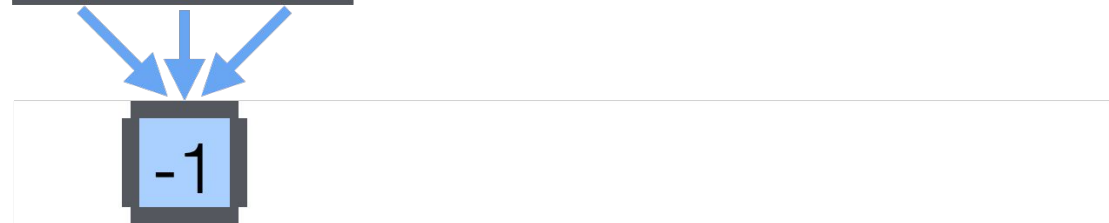
1D sequence



Convolutional Filter
(model parameters)



$$0 * -1 + 0 * 1 + 1 * -1 = -1$$

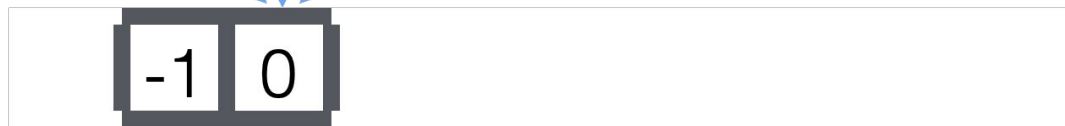


Convolutional Layer

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)



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

Convolutional Layer

1D sequence



Convolutional Filter
(model parameters)



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

Convolutional Layer

1D sequence



Convolutional Filter
(model parameters)



Filters can have different
widths

Convolutional Layer

1D sequence



Convolutional Filter
(model parameters)



Filters can have different
widths

Convolutional Layer

1D sequence



Convolutional Filter
(model parameters)



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

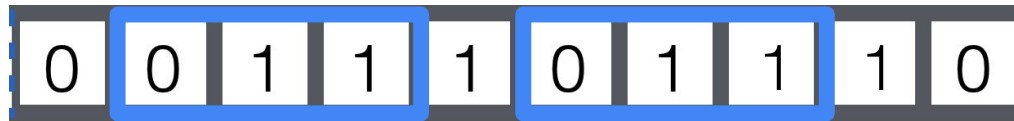
Convolutional Layer: Translational Invariance

0	0	1	1	1	0	1	1	1	0
---	---	---	---	---	---	---	---	---	---

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

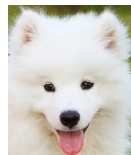
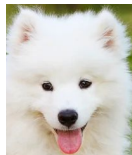
-1	0	-1	0	-2	0	-1	0
----	---	----	---	----	---	----	---

Convolutional Layer: Translational Invariance



Same local region =>
same output!

Convolutional Layer: Translational Invariance



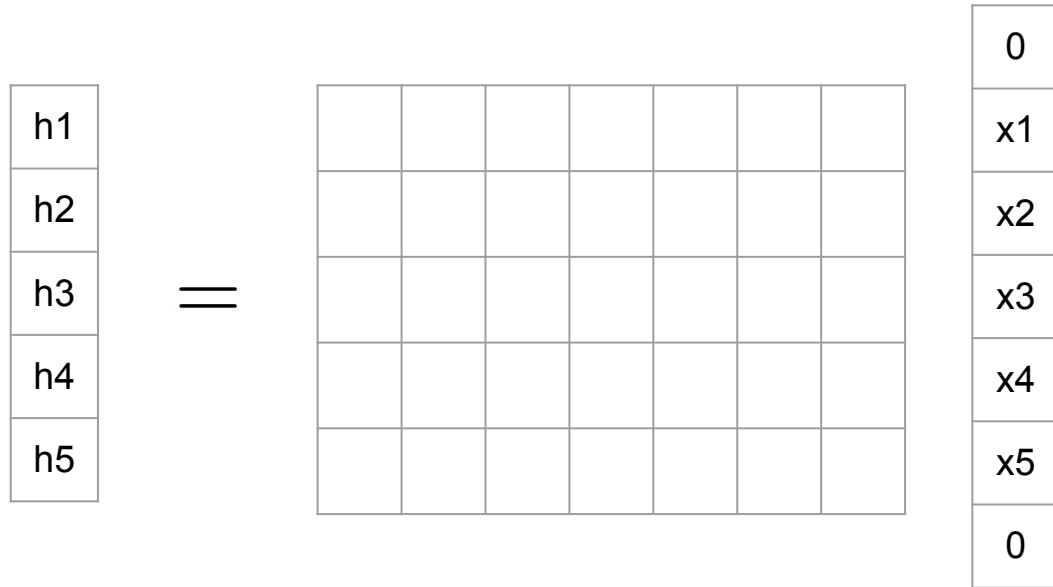
Doesn't matter where
the feature occurs



Same local region =>
same output!

Convolution as Matrix Multiplication

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



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

The diagram illustrates the construction of a 5x5 matrix. On the left is a 5x3 matrix with rows labeled h1, h2, h3, h4, h5 and columns labeled w1, w2, w3. In the middle is an equals sign. To the right of the equals sign is a 5x5 matrix. The first three columns of this 5x5 matrix are identical to the 5x3 matrix on the left. The fourth and fifth columns are filled with zeros. To the right of the 5x5 matrix is a 5x2 matrix with rows labeled x1, x2, x3, x4, x5 and columns labeled w1, w2. The first column of this 5x2 matrix contains zeros, and the second column contains the labels x1, x2, x3, x4, x5.

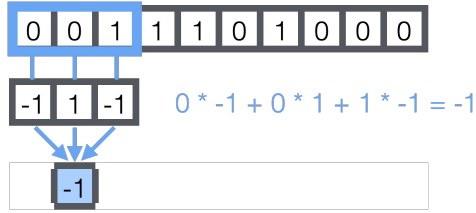
h1
h2
h3
h4
h5

=

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	0
x1	x2
x3	x4
x5	0

Convolution as Matrix Multiplication



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

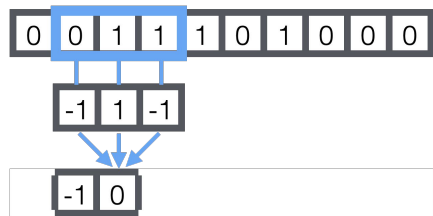
h_1
h_2
h_3
h_4
h_5

$=$

w_1	w_2	w_3	0	0	0	0
0	w_1	w_2	w_3	0	0	0
0	0	w_1	w_2	w_3	0	0
0	0	0	w_1	w_2	w_3	0
0	0	0	0	w_1	w_2	w_3

0
x_1
x_2
x_3
x_4
x_5
0

Convolution as Matrix Multiplication



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

h1
h2
h3
h4
h5

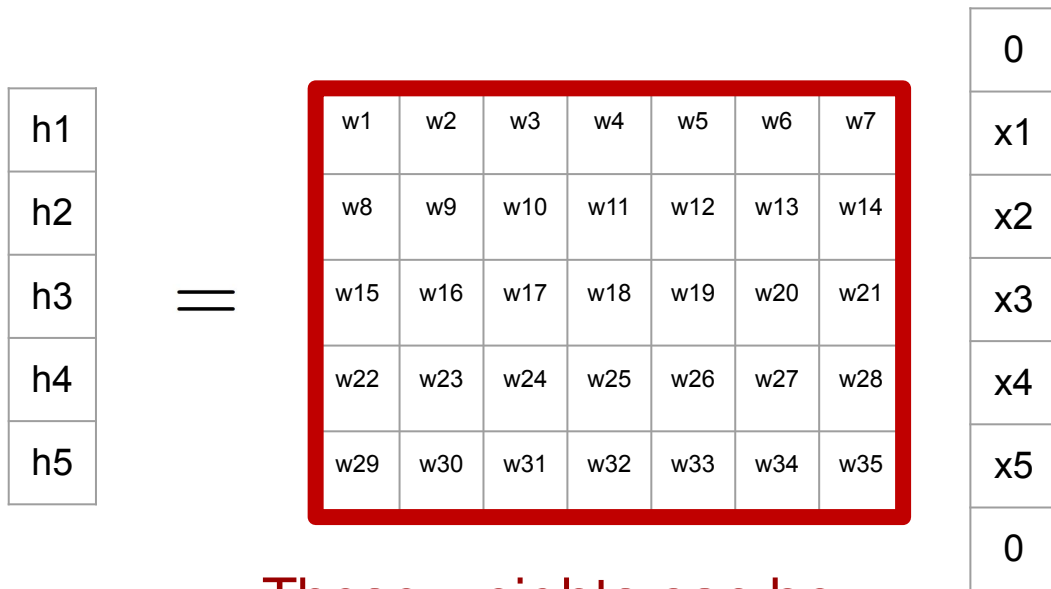
=

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

A 2D
image:

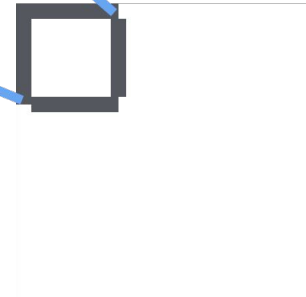
1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

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

-1

After
convolution:



Convolutions on Images

A 2D
image:

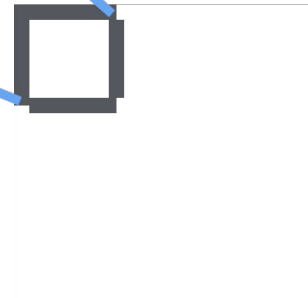
1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

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

$$-1 + 0$$

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

$$\begin{aligned} & -1 + 0 + -1 \\ & + -1 + 0 + -1 \\ & + -1 + -1 + -1 \\ & = -7 \end{aligned}$$



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

After
convolution:

-7	-2
----	----

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

After
convolution:

-7	-2	-4
----	----	----

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

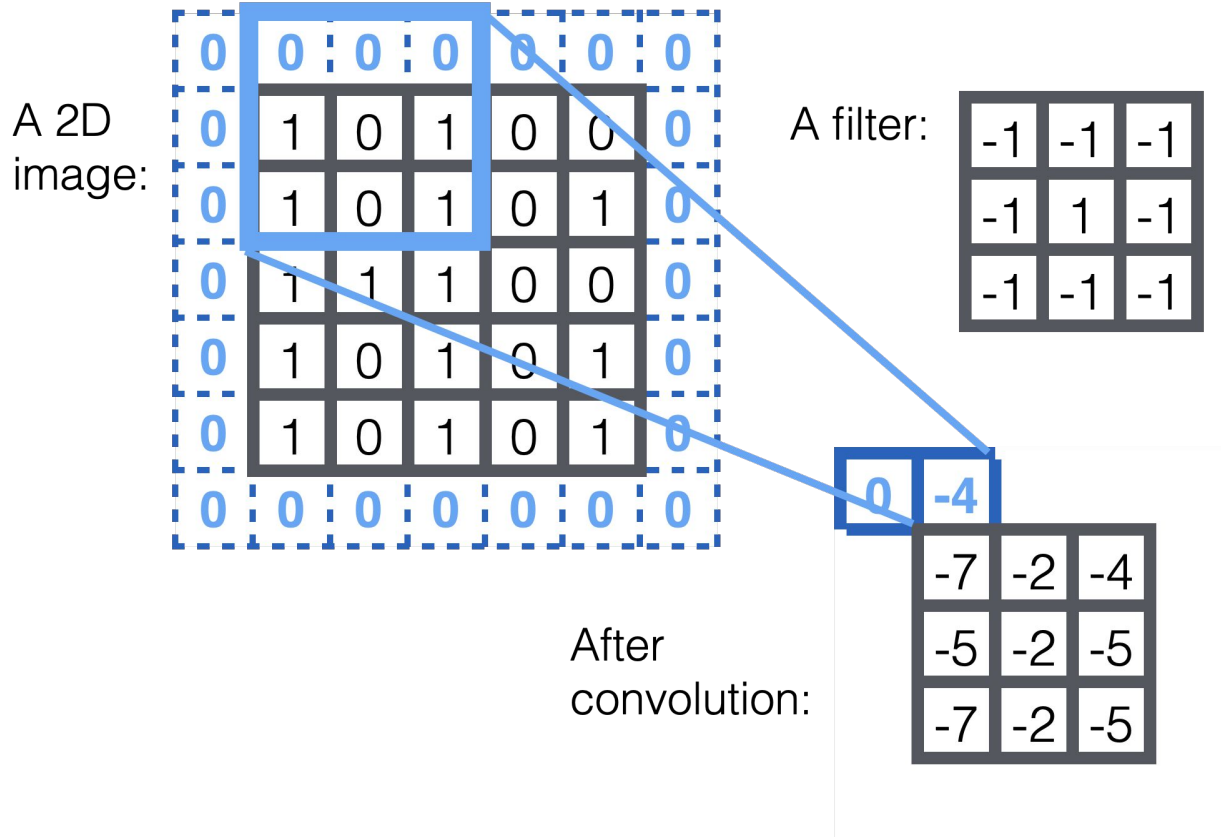
A filter:

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

After
convolution:

-7	-2	-4
-5	-2	-5
-7	2	-5

Convolutions on Images



Convolutions on Images

A 2D
image:

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

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:

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

A filter:

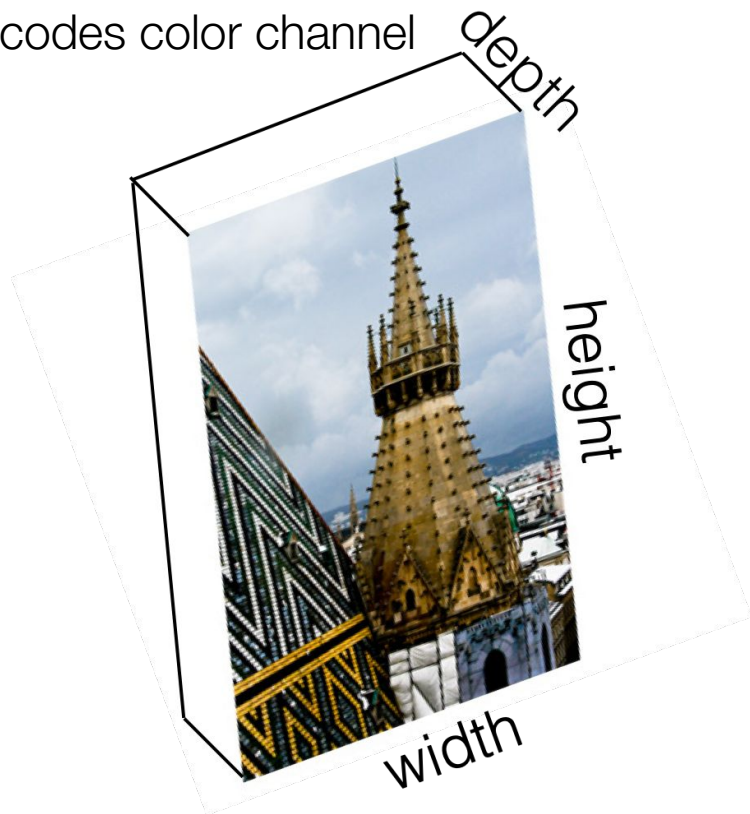
-1	-1	-1
-1	1	-1
-1	-1	-1

After
convolution
& ReLU:

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

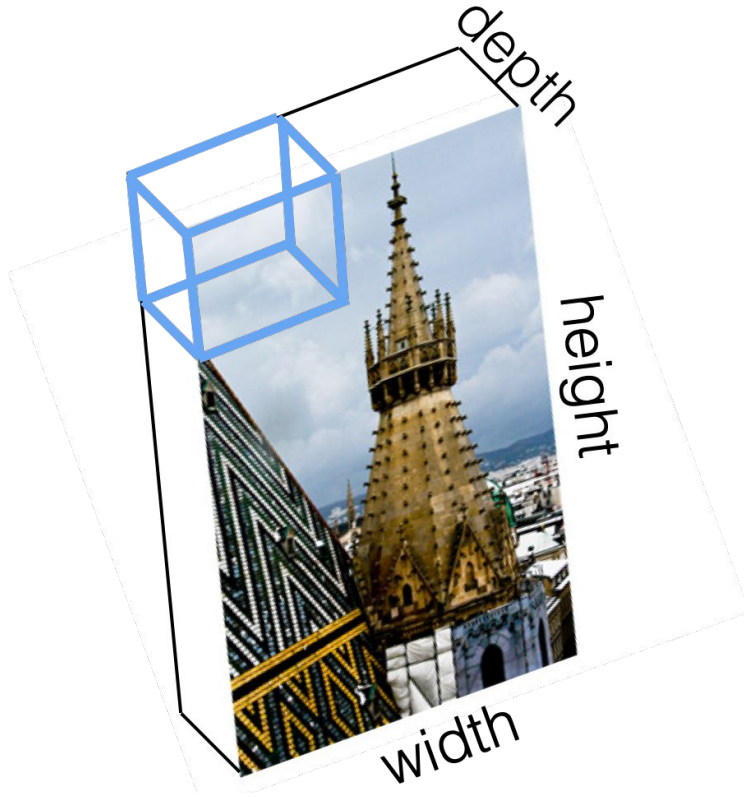
Convolutions on Images

Encodes color channel

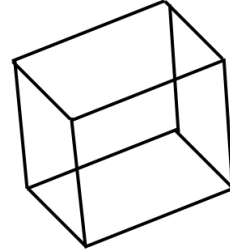


- Tensor: generalization of a matrix
 - E.g. 1D: vector, 2D: matrix

Convolutions on Images

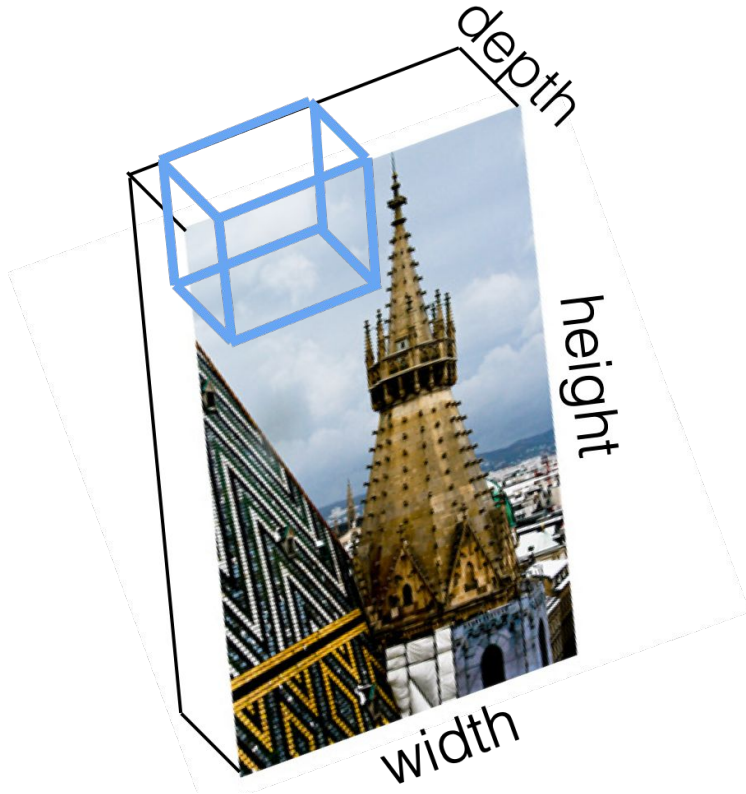


A filter:

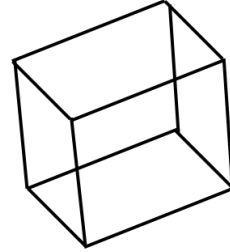


- Tensor: generalization of a matrix
 - E.g. 1D: vector, 2D: matrix

Convolutions on Images

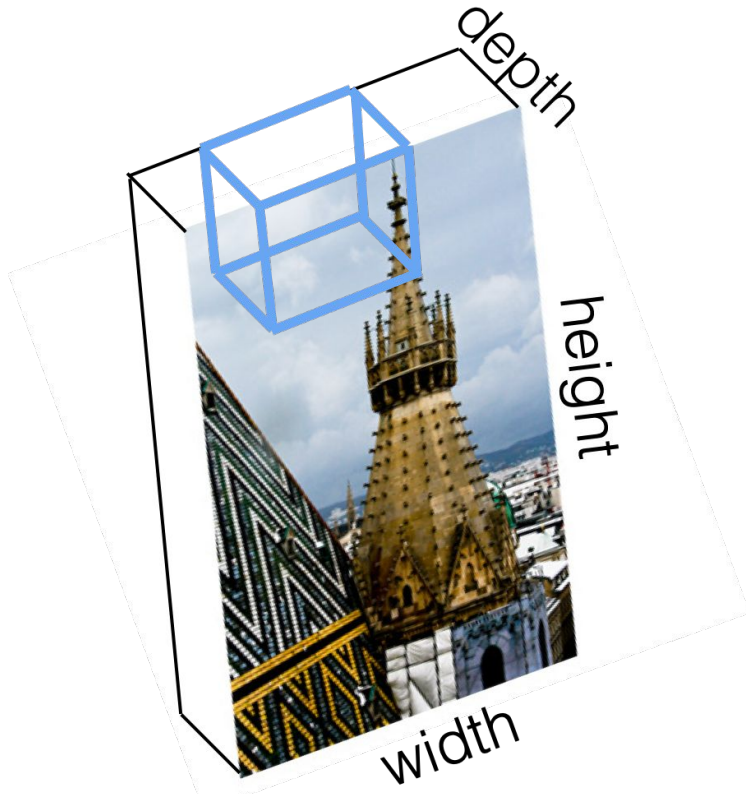


A filter:

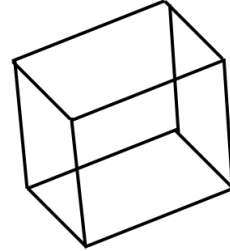


- Tensor: generalization of a matrix
 - E.g. 1D: vector, 2D: matrix

Convolutions on Images

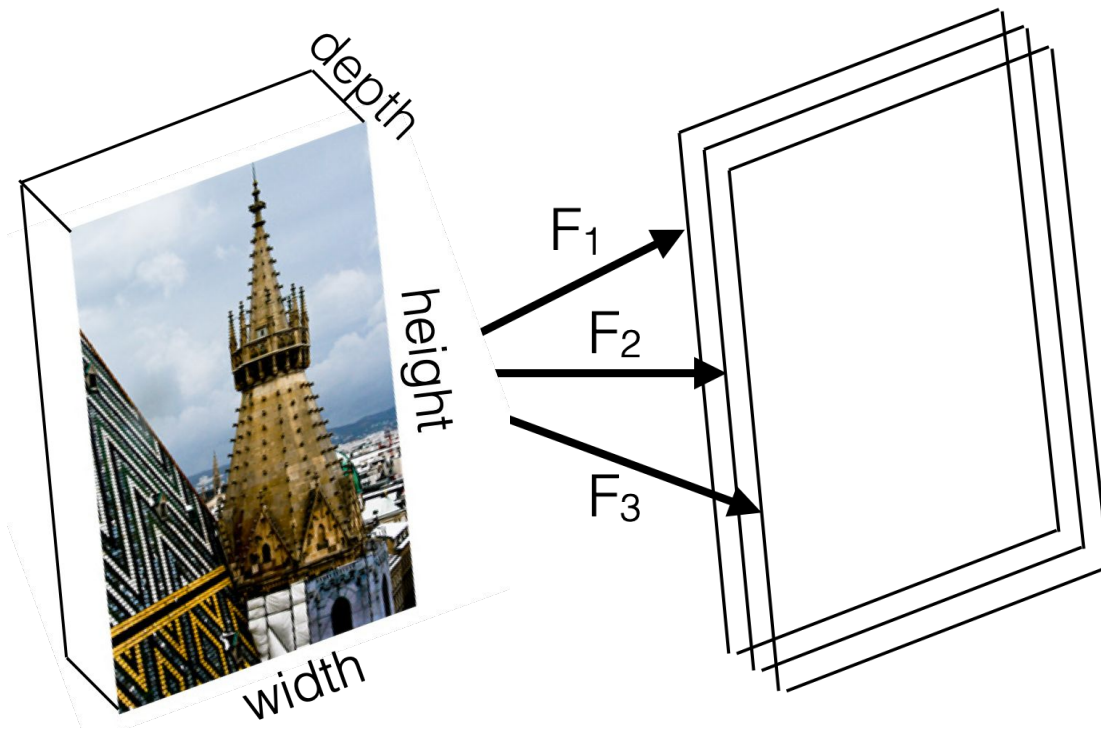


A filter:



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

Convolutions on Images



- Input : height x width x depth
- Each filter 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

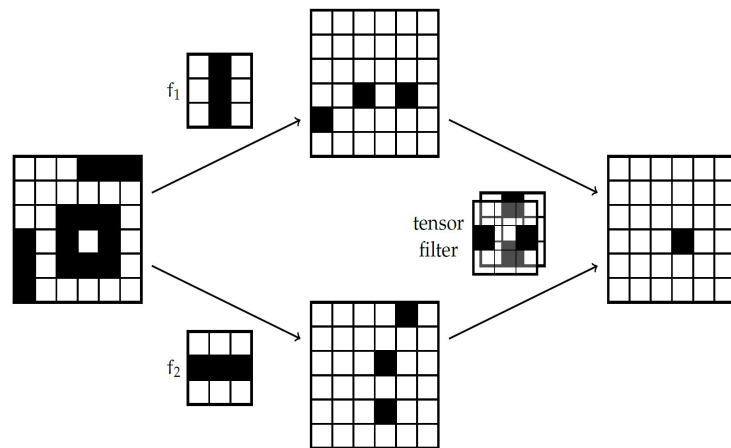
Input: $n^{l-1} \times n^{l-1} \times m^{l-1}$

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

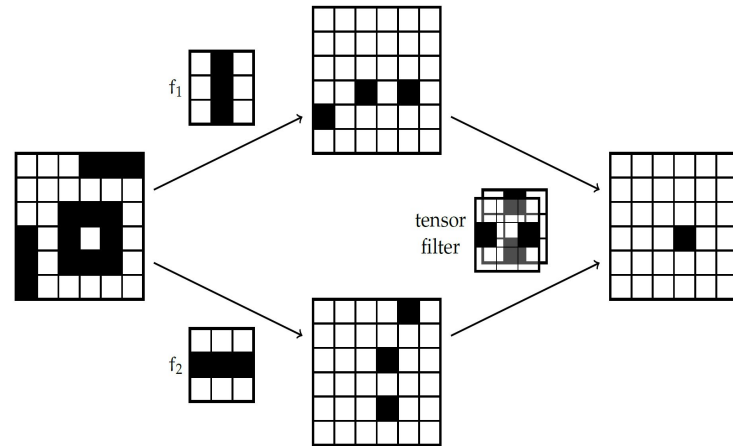
Input: $n^{l-1} \times n^{l-1} \times m^{l-1}$

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:

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”)

After max
pooling:

0

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”)

After max
pooling:

0	0
---	---

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”)

After max
pooling:

0	0	1
---	---	---

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”)

After max
pooling:

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

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”)

After max
pooling:

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

“Stride” 1 pooling

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”)
- E.g. stride 3

After max
pooling:

0	1
1	0

“Stride” 3 pooling

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”)
- E.g. stride 3

After max
pooling.

0	
1	0

“Stride” 3 pooling

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”)
- E.g. stride 3

After max
pooling.

0	1
1	0

“Stride” 3 pooling

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”)
- E.g. stride 3

After max
pooling:

0	1
1	0

“Stride” 3 pooling

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”)
- E.g. stride 3

After max
pooling:

0	1
1	0

“Stride” 3 pooling

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”)
- E.g. stride 3

After max
pooling:

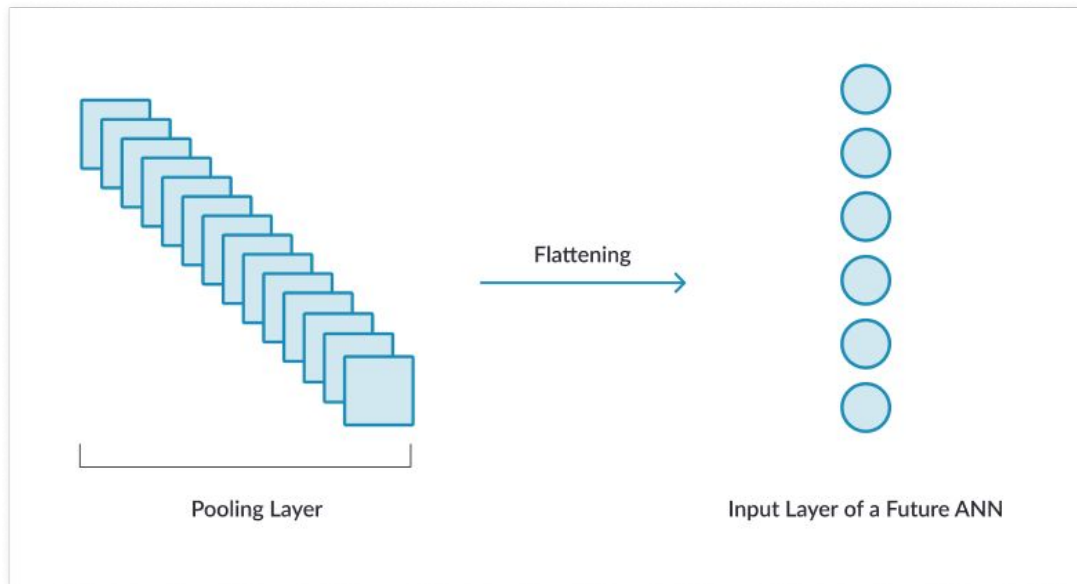
0	1
1	0

“Stride” 3 pooling

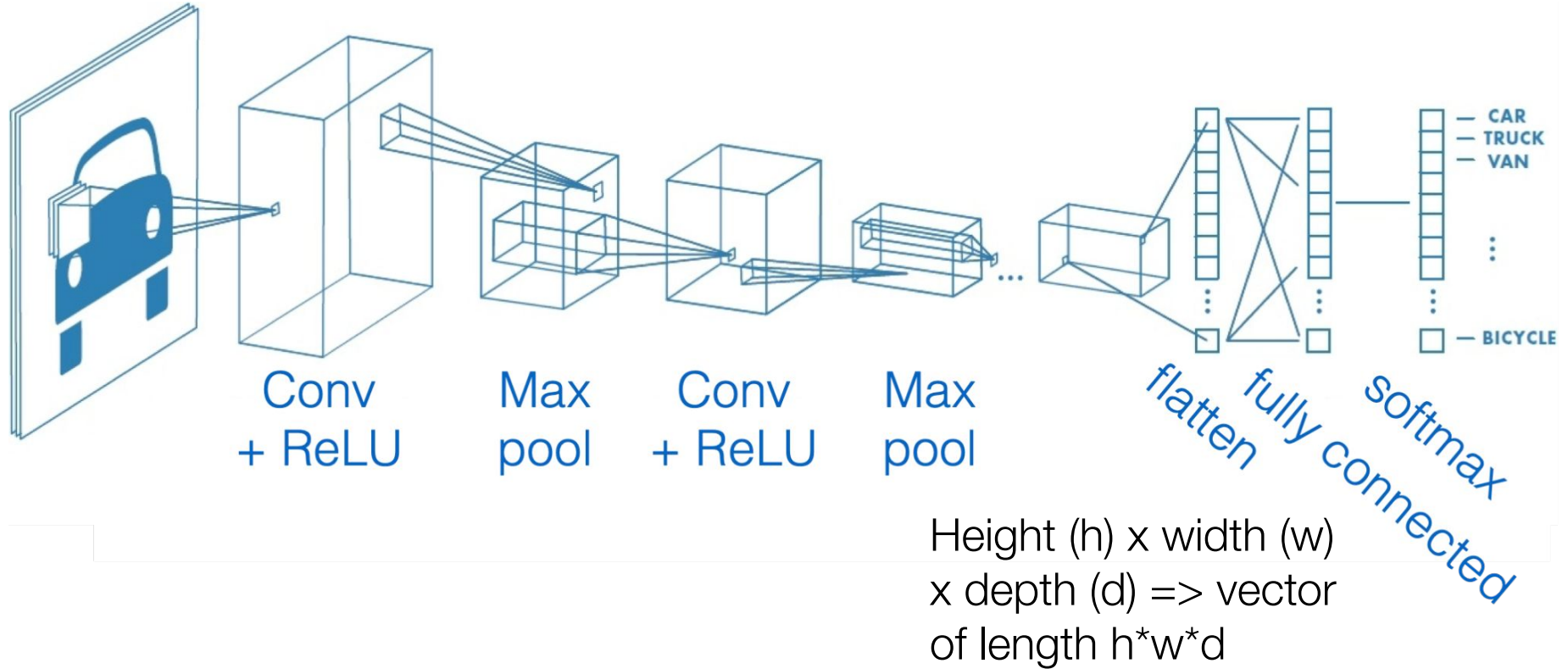
No parameters in
max pooling layer!

Flatten

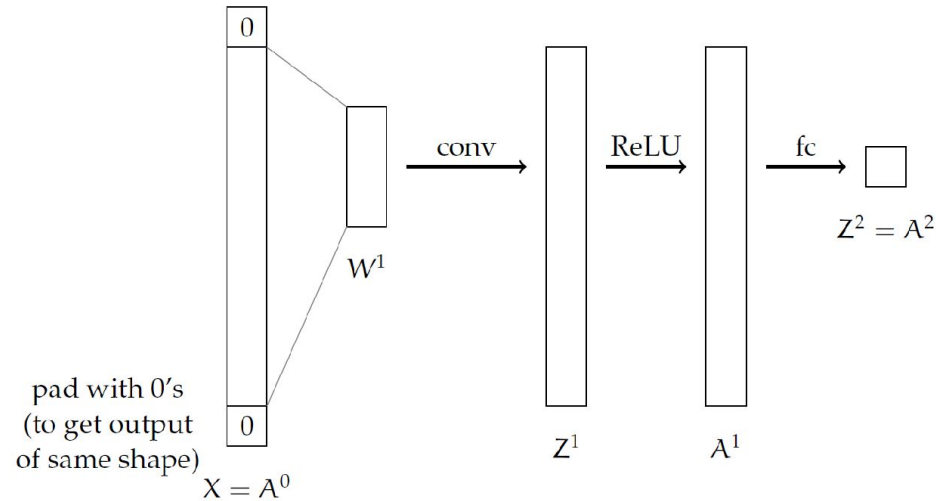
Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier



Putting it all together



Simple CNN: Forward Pass



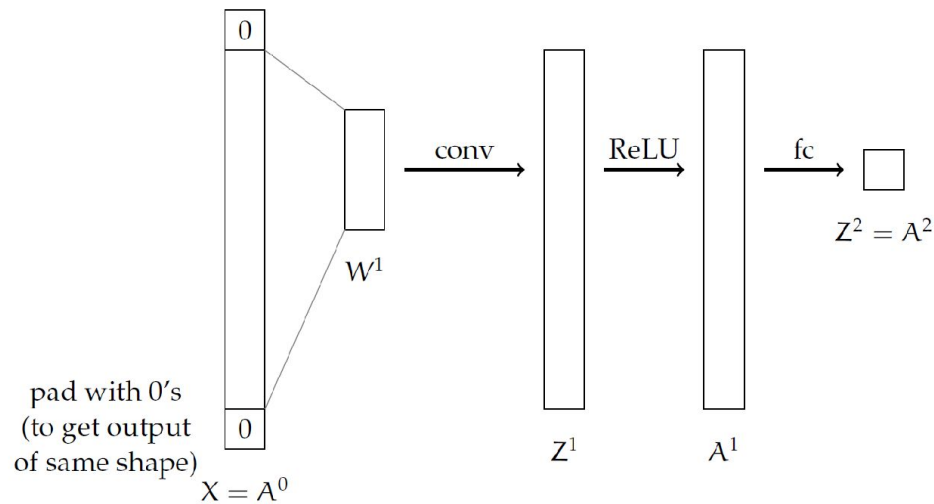
$$Z_i^1 = W^{1T} \cdot A_{[i-\lfloor k/2 \rfloor : i + \lfloor k/2 \rfloor]}^0$$

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

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

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

Simple CNN: Backpropagation



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

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

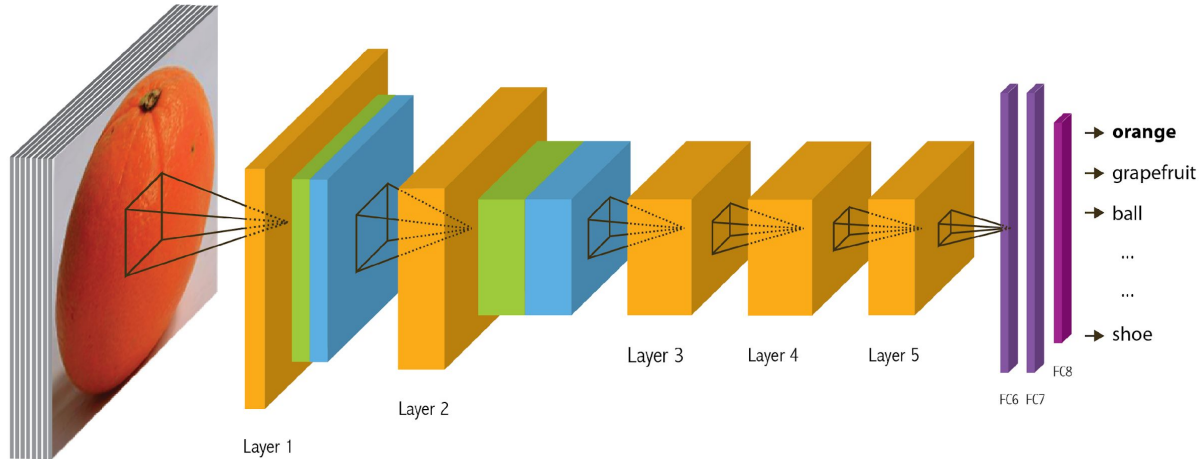
$$A^2 = W^{2T} A^1$$

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

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

Deep Learning Today

Used both for analysis and synthesis



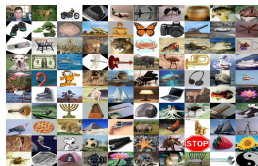
Input image

Advances in Dataset Collection

COIL-20



Caltech 101



2 year
old kid



IMAGENET
(2009)



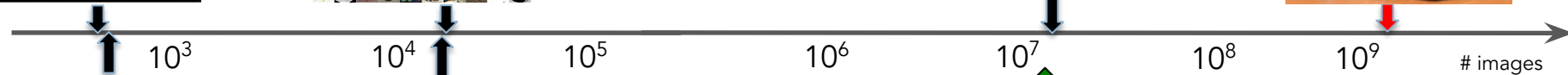
Caltech-4 (2003)



PASCAL (2005)



Places
(2013)

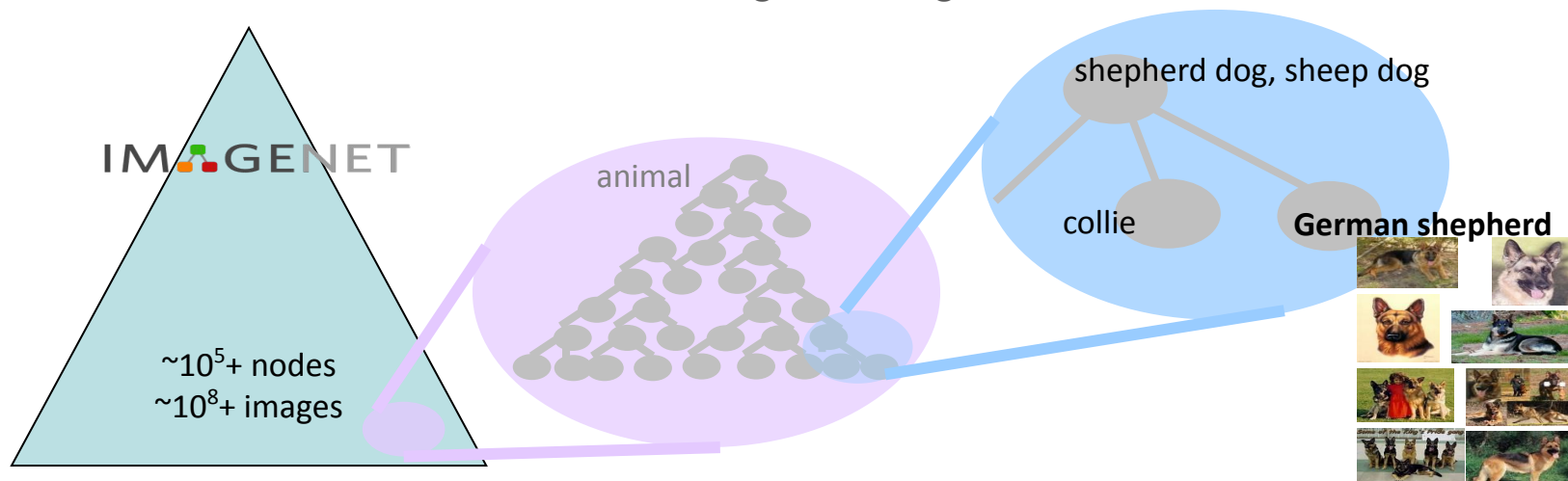


IMAGENET

An **ontology of images** based on WordNet

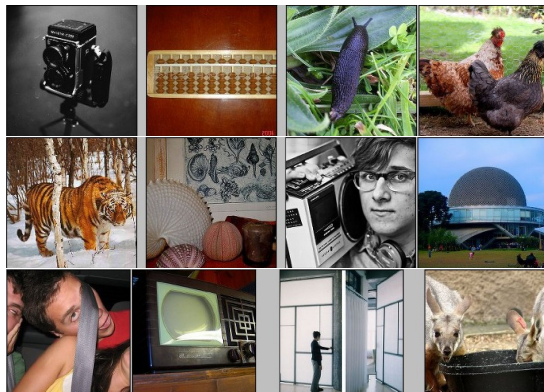
ImageNet currently has

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



Application to ImageNet

IMAGENET



[Deng et al. CVPR 2009]

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

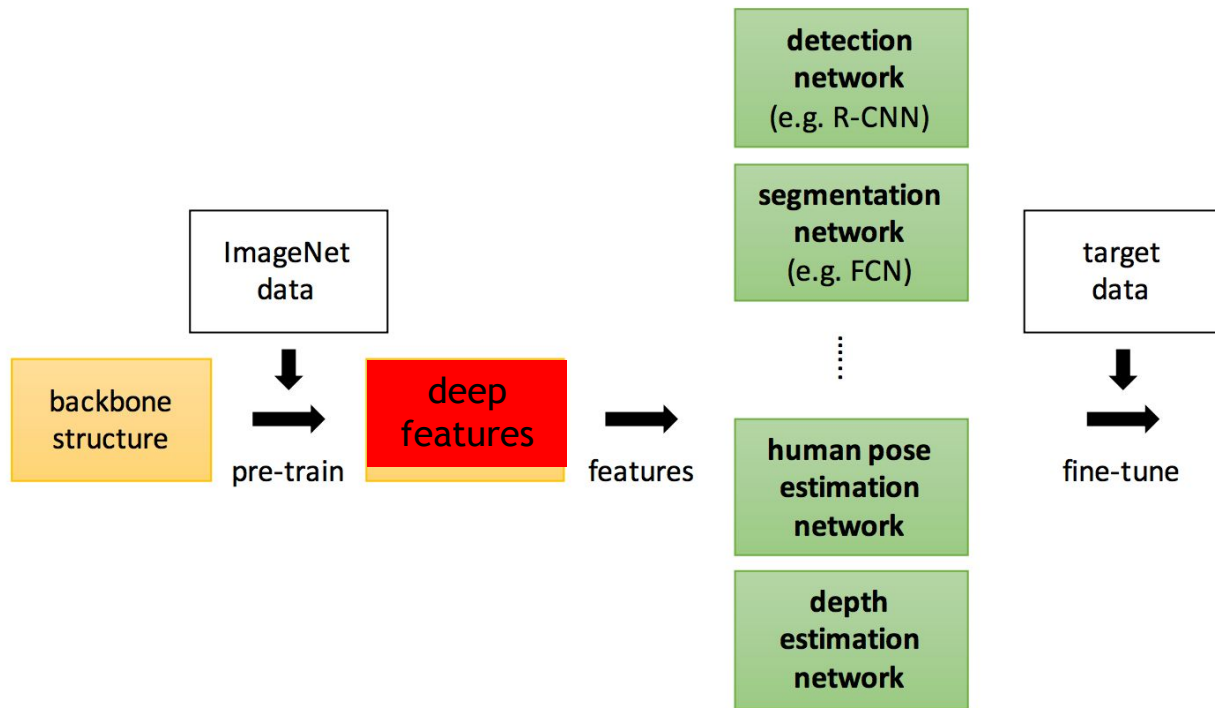
ImageNet Classification with Deep Convolutional Neural Networks [NIPS 2012]

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

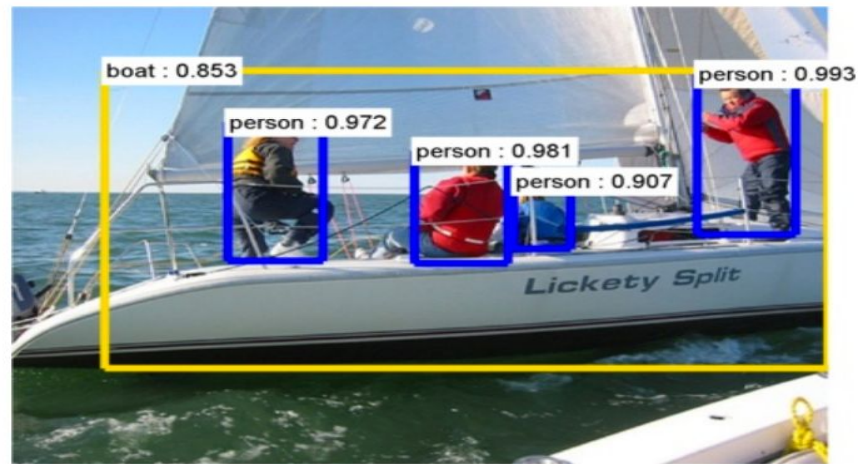
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

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

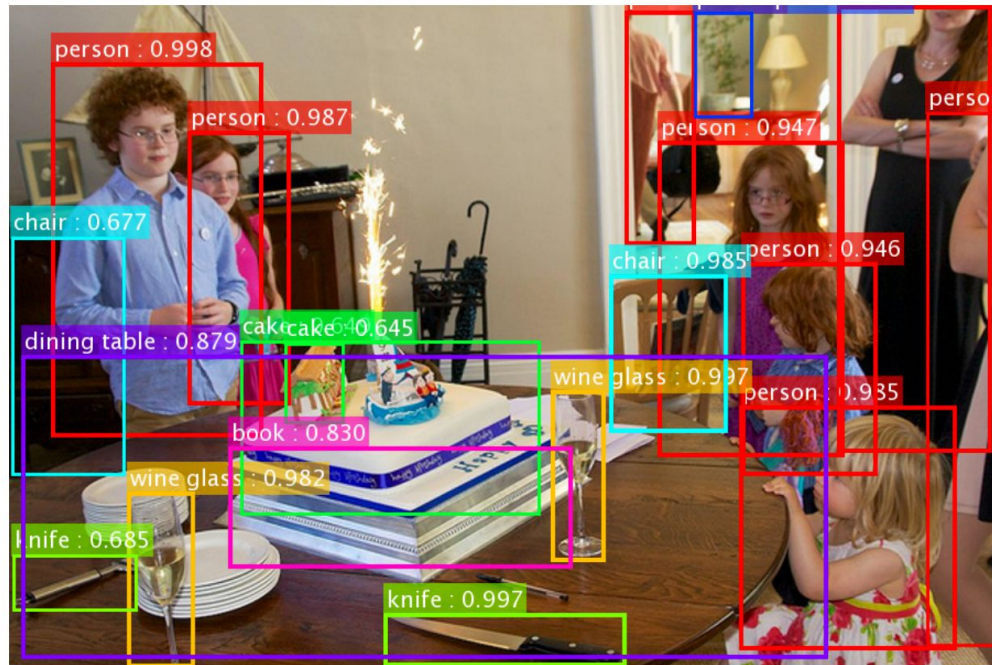
Solving Different Tasks



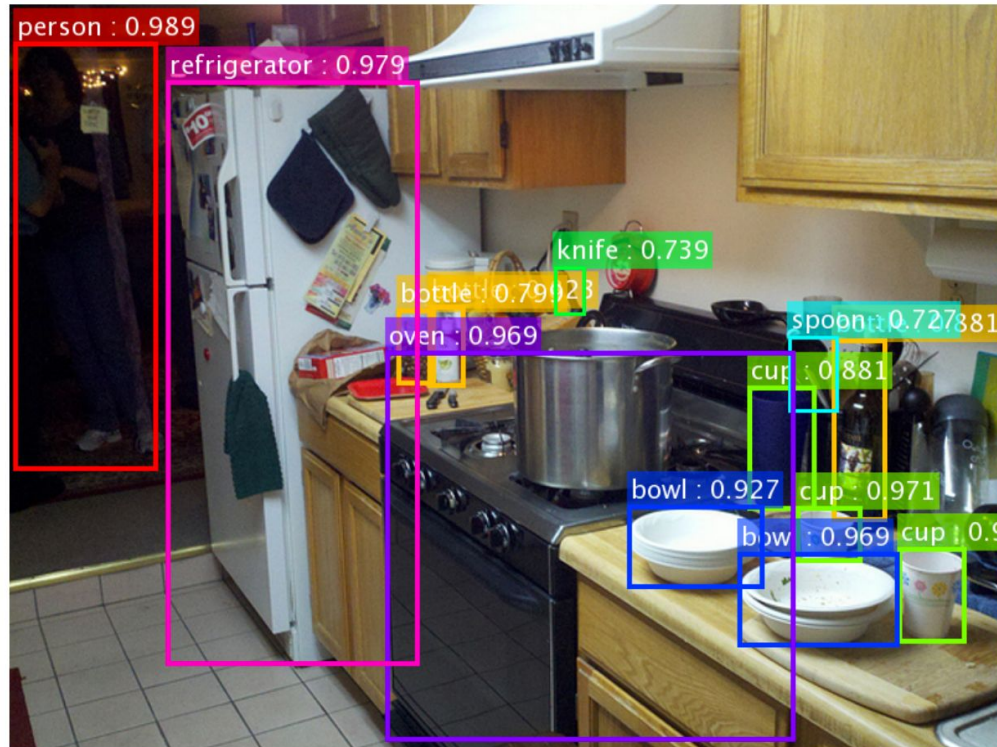
Object Detection: What and Where



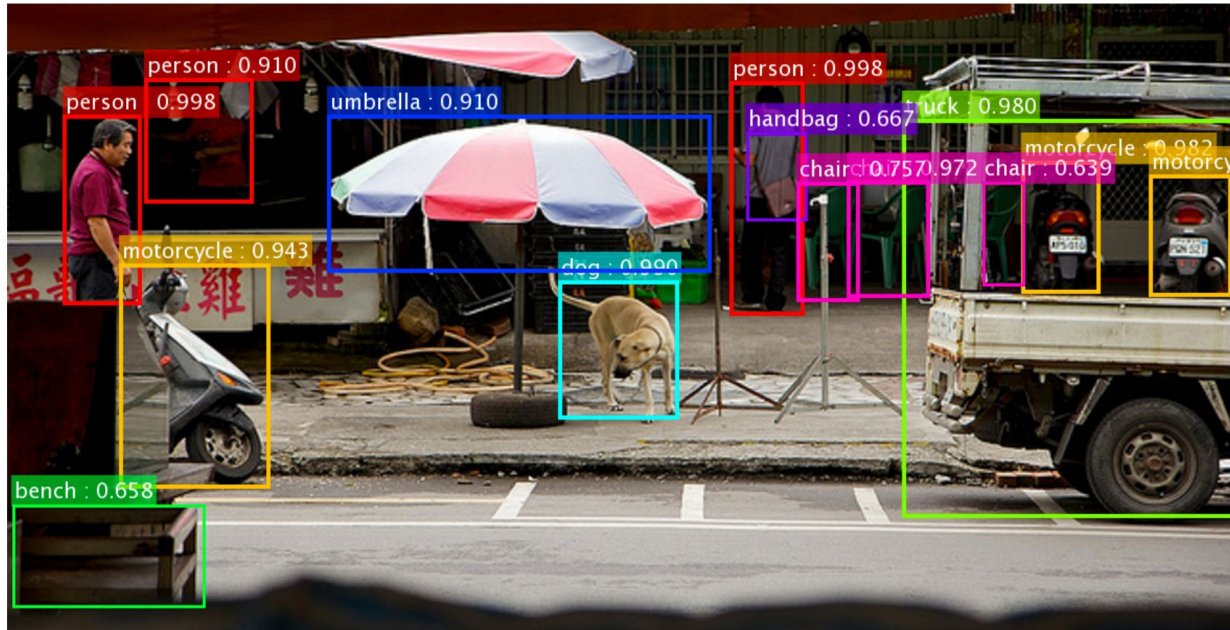
Object Detection Results



Object Detection Results



Object Detection Results

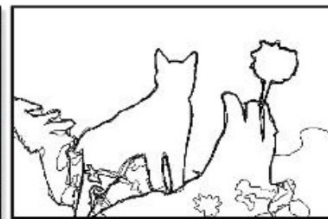
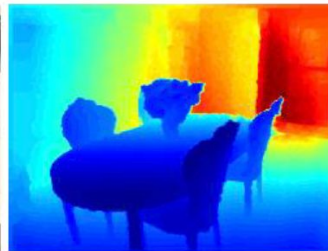


Segmentation, Depth, & More

semantic
segmentation



monocular depth estimation (Liu et al. 2015)



boundary prediction (Xie & Tu 2015)