

# CSE 435

## Convolutional Neural Networks

Tanvir Azhar  
Lecturer, East Delta University

# Outline

- ① Motivation
- ② 2D Convolution in Traditional Computer Vision
- ③ Basic Convolution Operation
- ④ Pooling Operation
- ⑤ Variants of the Basic Convolution
- ⑥ The Neuro-scientific Basis for Convolutional Networks

# Machine Learning

The **Task** can be expressed an unknown target function:  $y = f(x)$

ML finds a Hypothesis (model),  $h(\cdot)$ , from hypothesis space  $\mathcal{H}$ , which approximates the unknown target function.

$$\hat{y} = h^*(x) \approx f(x)$$

The **Experience** is typically a data set,  $\mathcal{D}$ , of values

The **Performance** is typically numerical measure that determines how well the hypothesis matches the experience.

Nearly all machine learning algorithms can be described with the following fairly simple recipe:

- Dataset
- Cost function (Objective, loss)
- **Model**
- Optimization procedure

# Neural Network Resurgence

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition held between 2010 and 2017.

The datasets comprised approximately 1 million images and 1,000 object classes.

The annual challenge focuses on multiple tasks for image classification.

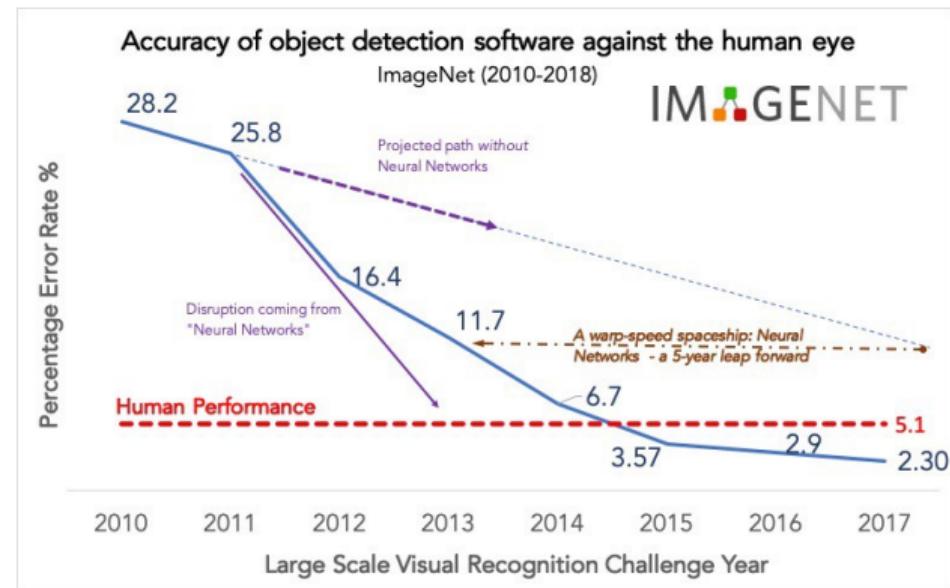


Image source: ImageNet

Alex Krizhevsky, et al. "ImageNet Classification with Deep Convolutional Neural Networks" developed a convolutional neural network that achieved top results on the ILSVRC-2010 and ILSVRC-2012 image classification tasks.

# Neural Network Resurgence

## AlexNET:

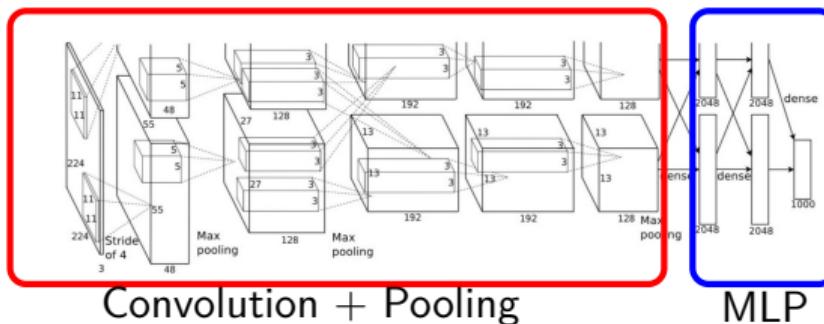


Image: [ImageNet Classification with Deep Convolutional Neural Networks](#)

Convolutions allows the network to have lots of neurons while keeping the number of actual parameters that need to be learned fairly small.

Convolutions has become very popular for verity of tasks including: Vision, NLP, Speech processing, etc.

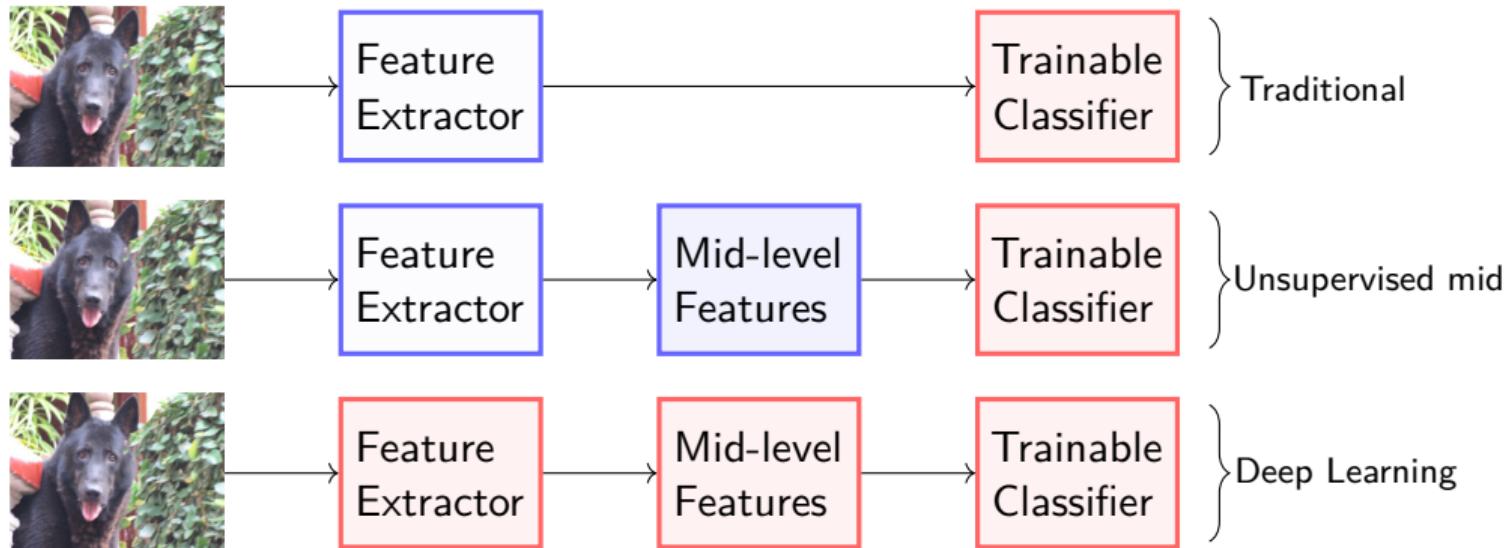
# Objective

- Understand the importance of convolutions neural networks.
- Understand variants of the basic convolution operation.

# Outline

- 1 Motivation
- 2 2D Convolution in Traditional Computer Vision
- 3 Basic Convolution Operation
- 4 Pooling Operation
- 5 Variants of the Basic Convolution
- 6 The Neuro-scientific Basis for Convolutional Networks

# Learning Hierarchical Representations

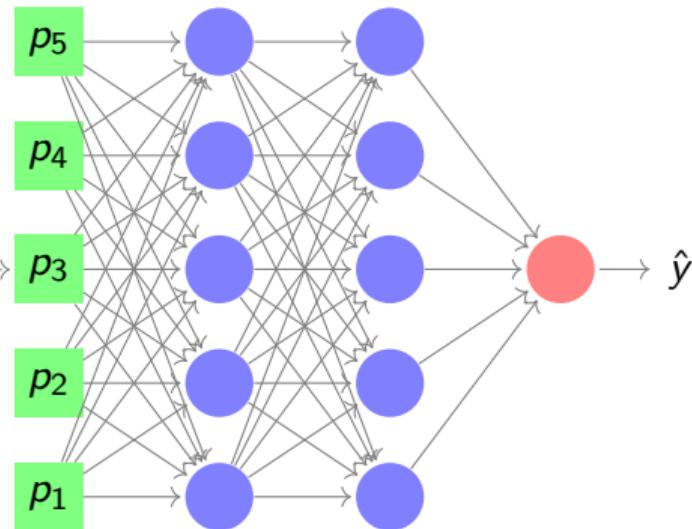


Handcrafted features are **time consuming, brittle and not scalable** in practice. DL learn underlying features directly from data.

# Apply Neural Networks to Images



Flatten



Each feature in the first hidden layer has a connection to each pixel.

In computer vision features are usually **local** in space and **same operation** is applied across different locations.

# Spacial Relationship

- Natural language Processing (NLP):

```
The sole importance of the crossing of the Berezina lies in the fact  
that it was a bold and indubitably proved the fallacy of all the plans for  
cutting off the enemy's retreat and the soundness of the only possible  
line of action--the one Kutuzov and the general mass of the army  
demanded--namely, simply to follow the enemy up. The French crowd fled  
at a continually increasing speed and all its energy was directed to  
reaching its goal. The hillside was covered with animals and with people  
who blocked its path. This was shown not so much by the arrangements it  
made for crossing as by what took place at the bridges. When the bridges  
broke down, unarmed soldiers, people from Moscow and women with children  
who were with the French transport, all--carried on by vis inertiae--  
pressed forward into boats and into the ice-covered water and did not,  
surrender.
```

```
Cell sensitive to position in line:  
Cell that robustly activates inside if statements:  
static int _dequeue_signal(struct sigpending *pending, sigset_t *mask,  
    sginfo_t *info)  
{  
    int sig = next_signal(pending, mask);  
    if (sig){  
        if (current->notifier){  
            if (!sgimember(current->notifier->mask, sig)){  
                if (((current->notifier)-(current->notifier_data)) <  
                    clear_thread_flag(TIF_SIGPENDING);  
            }  
        }  
        collect_signal(sig, pending, info);  
    }  
    return sig;  
}
```

Image: [The Unreasonable Effectiveness of Recurrent Neural Networks](#)

- Speech Recognition (Voice to text):

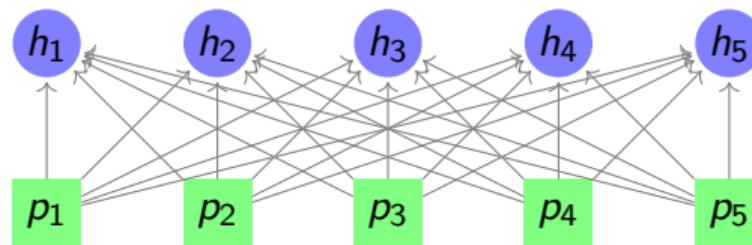
- “We have one **hour** before **our** appointment with the real estate agent.”
- “There is no **right** way to **write** a great novel”

- Not all tasks have such relationships. e.g: Predicting house prices using some attributed like “number of rooms”, “suburb”, etc.

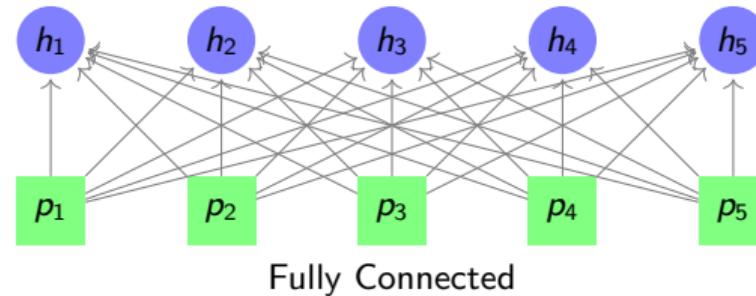
# Apply Neural Networks to Images

In our ML model we would like to have these two properties.

- Feature extraction usually happens locally - sparse connectivity.
- In feature extraction the same operation is applied at different locations - parameter sharing.

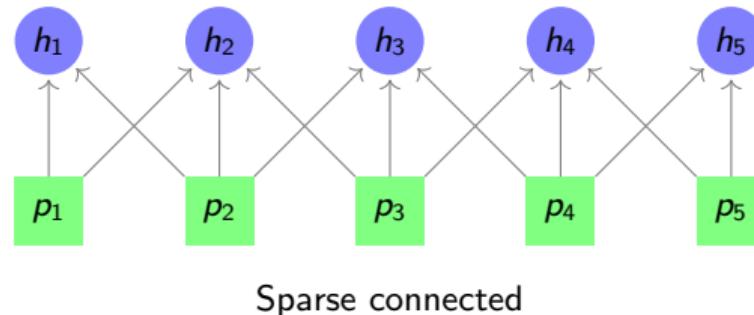
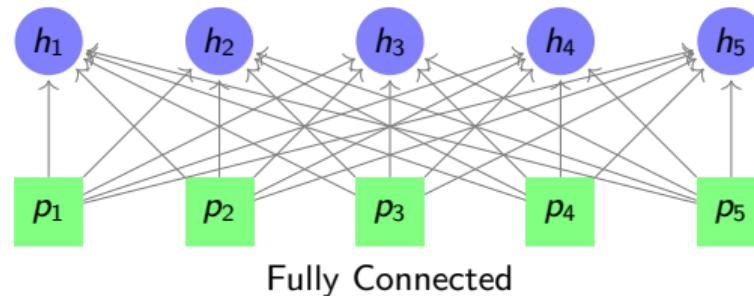


- Feature extraction usually happens locally - **sparse connectivity**.
- In feature extraction the same operation is applied at different locations - parameter sharing.



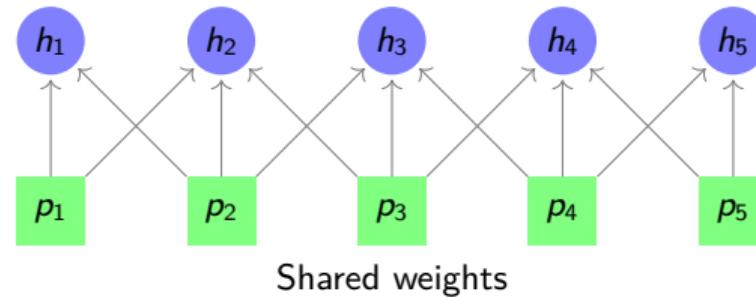
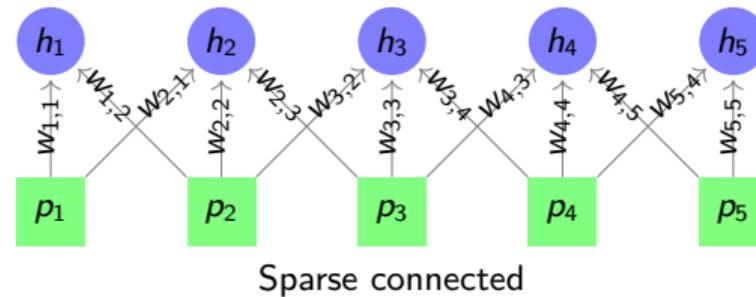
# Sparse Connectivity

- Feature extraction usually happens locally - **sparse connectivity**.
- In feature extraction the same operation is applied at different locations - parameter sharing.



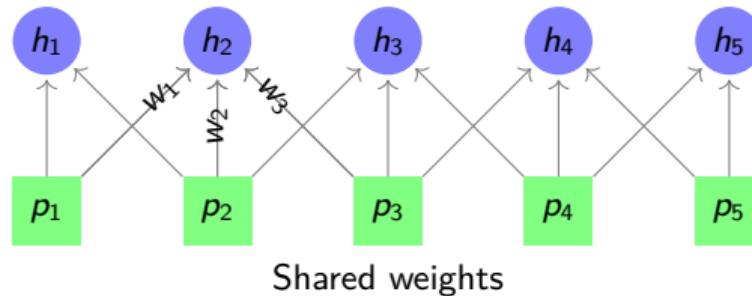
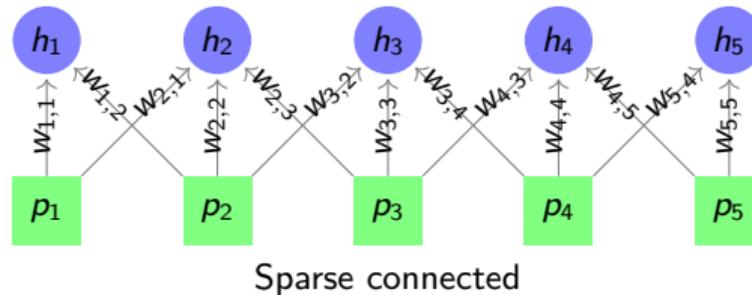
# Parameter Sharing ✓

- Feature extraction usually happens locally - **sparse connectivity**.
- In feature extraction the same operation is applied at different locations - **parameter sharing**.



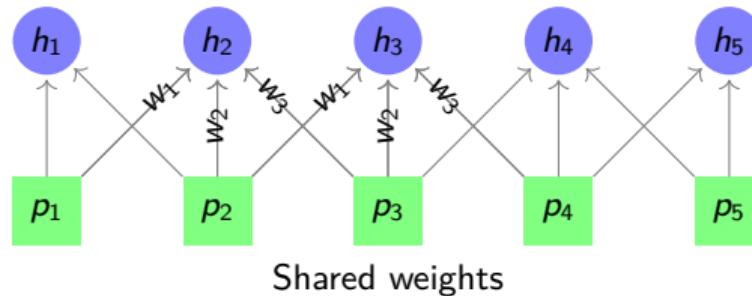
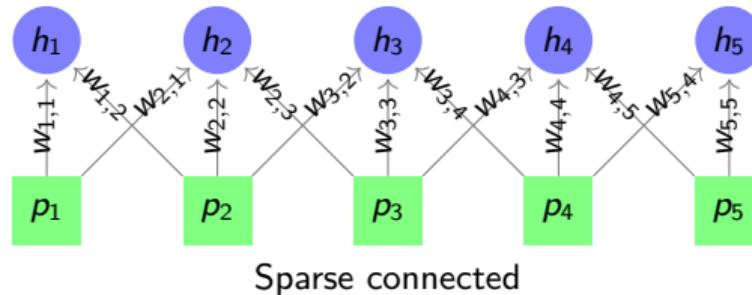
# Parameter Sharing

- Feature extraction usually happens locally - **sparse connectivity**.
- In feature extraction the same operation is applied at different locations - **parameter sharing**.



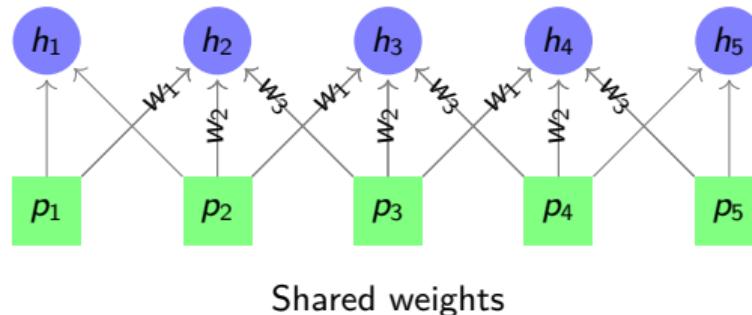
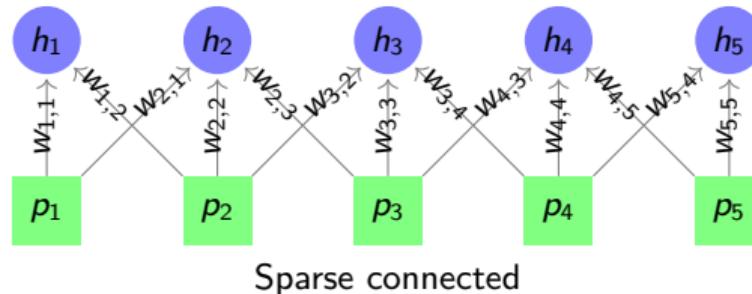
# Parameter Sharing

- Feature extraction usually happens locally - **sparse connectivity**.
- In feature extraction the same operation is applied at different locations - **parameter sharing**.



# Parameter Sharing

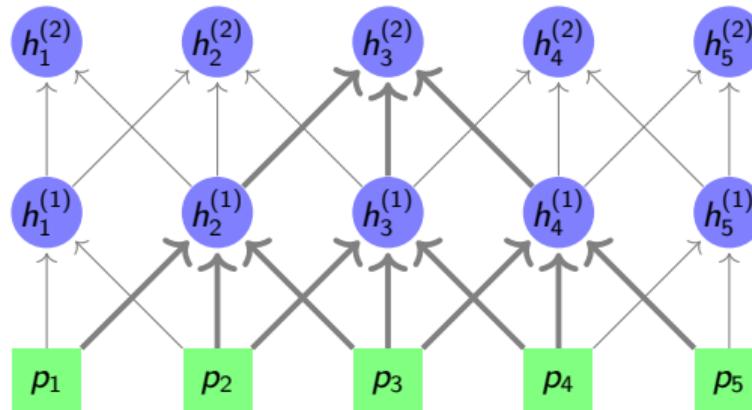
- Feature extraction usually happens locally - **sparse connectivity**.
- In **feature extraction** the **same operation** is applied at **different locations** - **parameter sharing**.



# 1D Convolution

Both ideas, sparse connectivity & parameter sharing can be achieved with convolutions.

Convolutions can also be implemented in a hierarchy, where each layer act on the features extracted by the layer below.



note that this network will have only 6 weights and 2 biases; compared to  $(25+5) + (25 + 5)$  parameters in a fully connected network with the same number of neurons.

# Outline

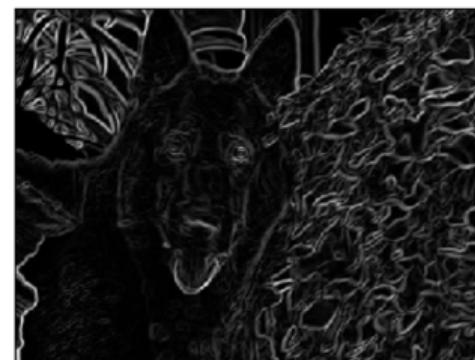
- 1 Motivation
- 2 2D Convolution in Traditional Computer Vision
- 3 Basic Convolution Operation
- 4 Pooling Operation
- 5 Variants of the Basic Convolution
- 6 The Neuro-scientific Basis for Convolutional Networks



# 2D Convolution in Traditional Computer Vision



Input Image



Edge Features

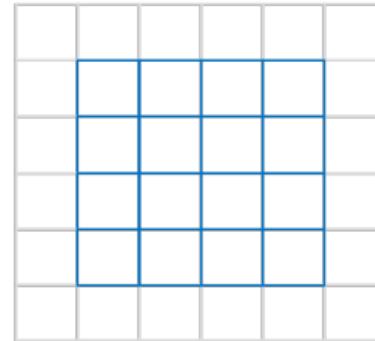
Extracting Edge Features

# 2D Convolution in Traditional Computer Vision

0.8	0.74	0.6	0.5	0.47	0.55
0.81	0.73	0.58	0.45	0.41	0.46
0.81	0.73	0.58	0.43	0.35	0.38
0.81	0.73	0.58	0.43	0.31	0.31
0.82	0.73	0.61	0.44	0.29	0.27
0.83	0.75	0.62	0.44	0.27	0.25

\*

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

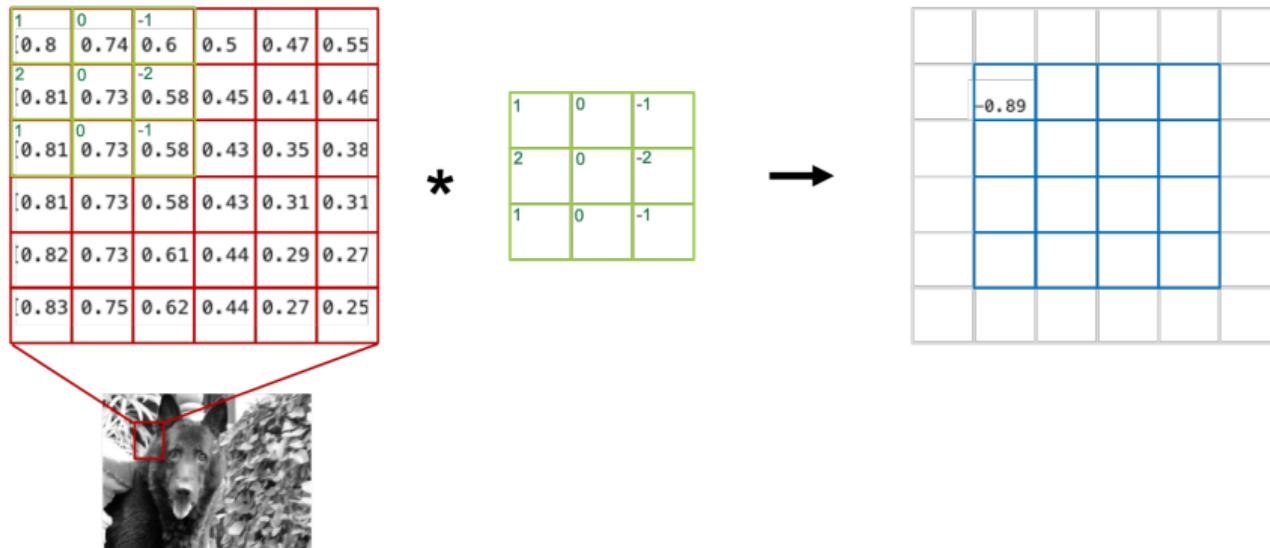


Sobel Filter  
Kernel



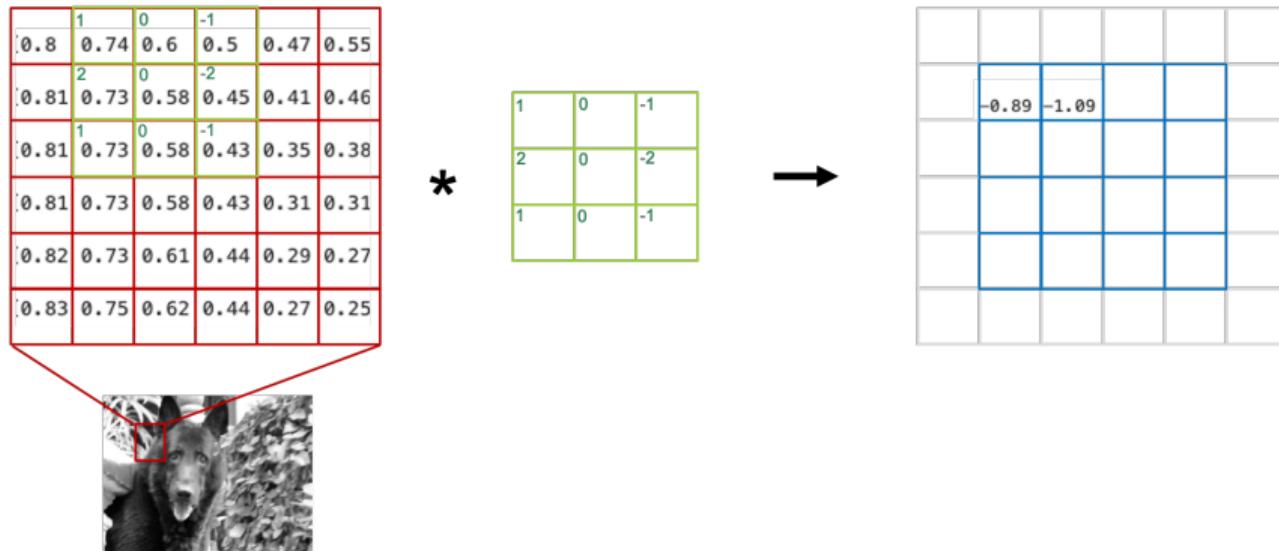
Sobel operator is human engineered

# 2D Convolution in Traditional Computer Vision



$$I_o(i, j) = \sum_{p=-1}^1 \sum_{q=-1}^1 w_{p,g} \times I_{in}(i - p, j - q)$$

# 2D Convolution in Traditional Computer Vision



$$I_o(i, j) = \sum_{p=-1}^1 \sum_{q=-1}^1 w_{p,q} \times I_{in}(i-p, j-q)$$

# 2D Convolution in Traditional Computer Vision

0.8	0.74	0.6	0.5	0.47	0.55
0.81	0.73	0.58	0.45	0.41	0.46
0.81	0.73	0.58	0.43	0.35	0.38
0.81	0.73	0.58	1	0	-1
0.82	0.73	0.61	2	0	-2
0.83	0.75	0.62	1	0.44	0.27

\*

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



-0.89	-1.09	-0.71	0.02	
-0.92	-1.18	-0.91	-0.2	
-0.9	-1.2	-1.11	-0.44	
-0.87	-1.21	-1.26	-0.63	



$$I_o(i, j) = \sum_{p=-1}^1 \sum_{q=-1}^1 w_{p,q} \times I_{in}(i-p, j-q)$$

# Outline

- 1 Motivation
- 2 2D Convolution in Traditional Computer Vision
- 3 Basic Convolution Operation**
- 4 Pooling Operation
- 5 Variants of the Basic Convolution
- 6 The Neuro-scientific Basis for Convolutional Networks

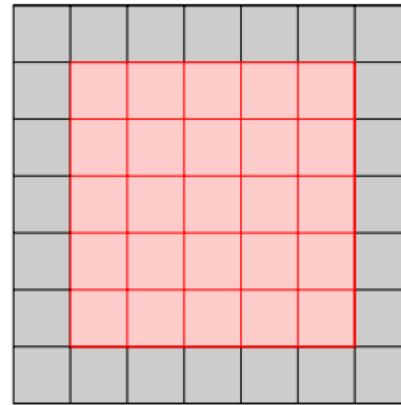
# 2D Convolution

1	6	2	8	1	2	7
1	6	2	8	1	2	5
0	5	8	1	5	7	1
1	7	1	3	5	8	0
5	2	4	4	5	8	4
8	2	3	7	3	8	2
1	2	3	6	5	9	6

Image

$$* \begin{matrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{matrix} =$$

Weight Filter



Output

The weights  $[w_{11}, w_{12}, \dots, w_{33}]$  are learned from data. For this example lets assume all weights are:  $[w_{11}, w_{12}, \dots, w_{33}] = [1, 1, \dots, 1]$ .

# 2D Convolution

1	6	2	8	1	2	7
1	6	2	8	1	2	5
0	5	8	1	5	7	1
1	7	1	3	5	8	0
5	2	4	4	5	8	4
8	2	3	7	3	8	2
1	2	3	6	5	9	6

Image

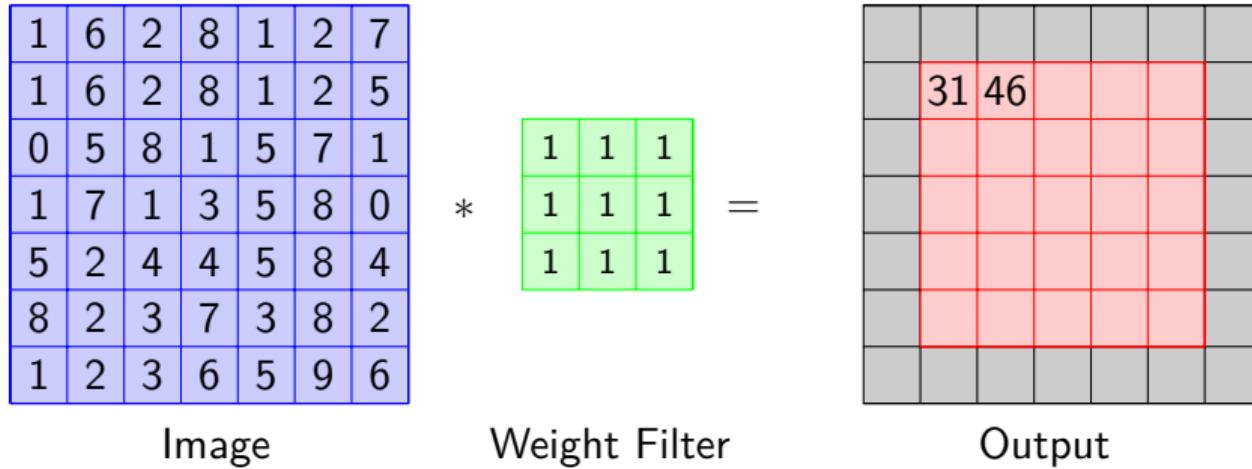
1	1	1
1	1	1
1	1	1

\* =


Output

The weights  $[w_{11}, w_{12}, \dots, w_{33}]$  are learned from data. For this example lets assume all weights are:  $[w_{11}, w_{12}, \dots, w_{33}] = [1, 1, \dots, 1]$ .

# 2D Convolution



The weights  $[w_{11}, w_{12}, \dots, w_{33}]$  are learned from data. For this example lets assume all weights are:  $[w_{11}, w_{12}, \dots, w_{33}] = [1, 1, \dots, 1]$ .

# 2D Convolution

1	6	2	8	1	2	7
1	6	2	8	1	2	5
0	5	8	1	5	7	1
1	7	1	3	5	8	0
5	2	4	4	5	8	4
8	2	3	7	3	8	2
1	2	3	6	5	9	6

\*

1	1	1
1	1	1
1	1	1

=

			31	46	36	

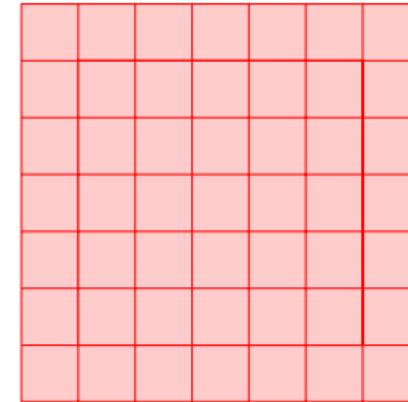
Image                  Weight Filter                  Output

The weights  $[w_{11}, w_{12}, \dots, w_{33}]$  are learned from data. For this example lets assume all weights are:  $[w_{11}, w_{12}, \dots, w_{33}] = [1, 1, \dots, 1]$ .

# Padding ✓

0	0	0	0	0	0	0	0	0	0
0	1	6	2	8	1	2	7	0	0
0	1	6	2	8	1	2	5	0	0
0	0	5	8	1	5	7	1	0	0
0	1	7	1	3	5	8	0	0	0
0	5	2	4	4	5	8	4	0	0
0	8	2	3	7	3	8	2	0	0
0	1	2	3	6	5	9	6	0	0
0	0	0	0	0	0	0	0	0	0

$$\begin{matrix} * & \begin{matrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{matrix} & = & \begin{matrix} \text{red grid} \end{matrix} \end{matrix}$$

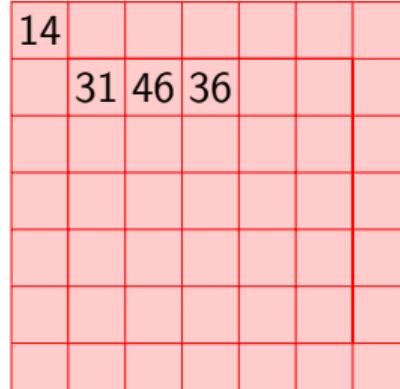


The weights  $[w_{11}, w_{12}, \dots, w_{33}]$  are learned from data. For this example lets assume all weights are:  $[w_{11}, w_{12}, \dots, w_{33}] = [1, 1, \dots, 1]$ .

✓ Padding will enable to get a output that is the same size of the input. the most common type of padding is zero padding.

# Padding

0	0	0	0	0	0	0	0	0	0
0	1	6	2	8	1	2	7	0	0
0	1	6	2	8	1	2	5	0	0
0	0	5	8	1	5	7	1	0	0
0	1	7	1	3	5	8	0	0	0
0	5	2	4	4	5	8	4	0	0
0	8	2	3	7	3	8	2	0	0
0	1	2	3	6	5	9	6	0	0
0	0	0	0	0	0	0	0	0	0

\*  $\begin{matrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{matrix}$  = 

The weights  $[w_{11}, w_{12}, \dots, w_{33}]$  are learned from data. For this example lets assume all weights are:  $[w_{11}, w_{12}, \dots, w_{33}] = [1, 1, \dots, 1]$ . Padding will enable to get a output that is the same size of the input. the most common type of padding is zero padding.

# Stride ✓

stride = 2

The diagram shows a matrix multiplication operation. On the left, there is an input matrix with 8 rows and 7 columns. A green border highlights the first two rows. A blue border highlights the first three columns. A handwritten note "stride = 2" is written above the first two rows. The input matrix contains the following values:

1	6	2	8	1	2	7
1	6	2	8	1	2	5
0	5	8	1	5	7	1
1	7	1	3	5	8	0
5	2	4	4	5	8	4
8	2	3	7	3	8	2
1	2	3	6	5	9	6

In the middle, there is a weight matrix with 3 rows and 3 columns. A green border highlights the first row. The weight matrix contains the following values:

$w_{11}$	$w_{12}$	$w_{13}$
$w_{21}$	$w_{22}$	$w_{23}$
$w_{31}$	$w_{32}$	$w_{33}$

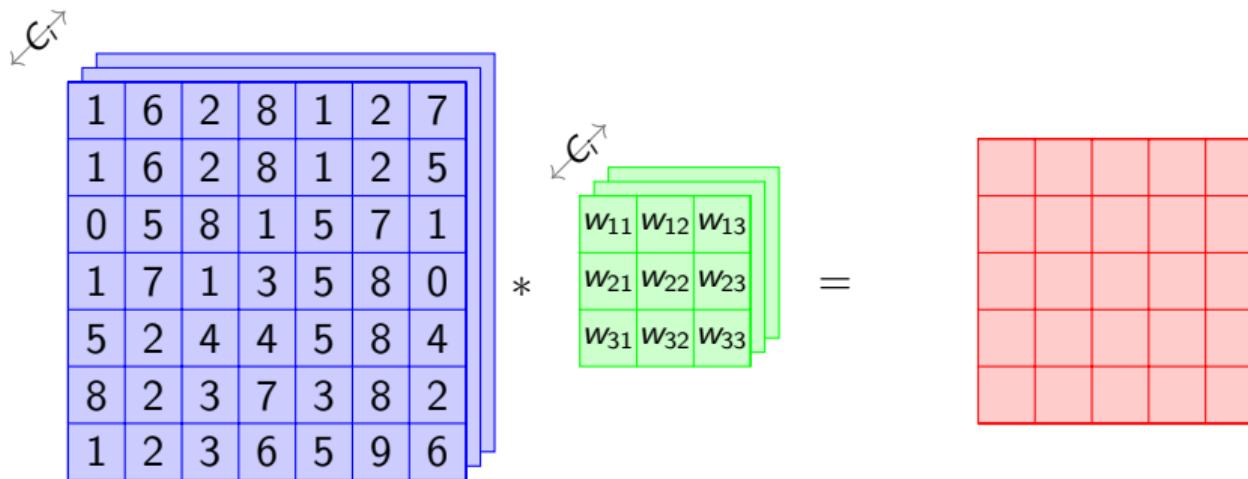
To the right of the weight matrix is an equals sign (=). To the right of the equals sign is a 3x2 output matrix. A red border highlights the first row. The output matrix contains the following values:

31	36

The weights  $[w_{11}, w_{12}, \dots, w_{33}]$  are learned from data. For this example lets assume all weights are:  $[w_{11}, w_{12}, \dots, w_{33}] = [1, 1, \dots, 1]$ . The stride of 2 will be used.

Stride will downsize the output by a factor related to the stride value.

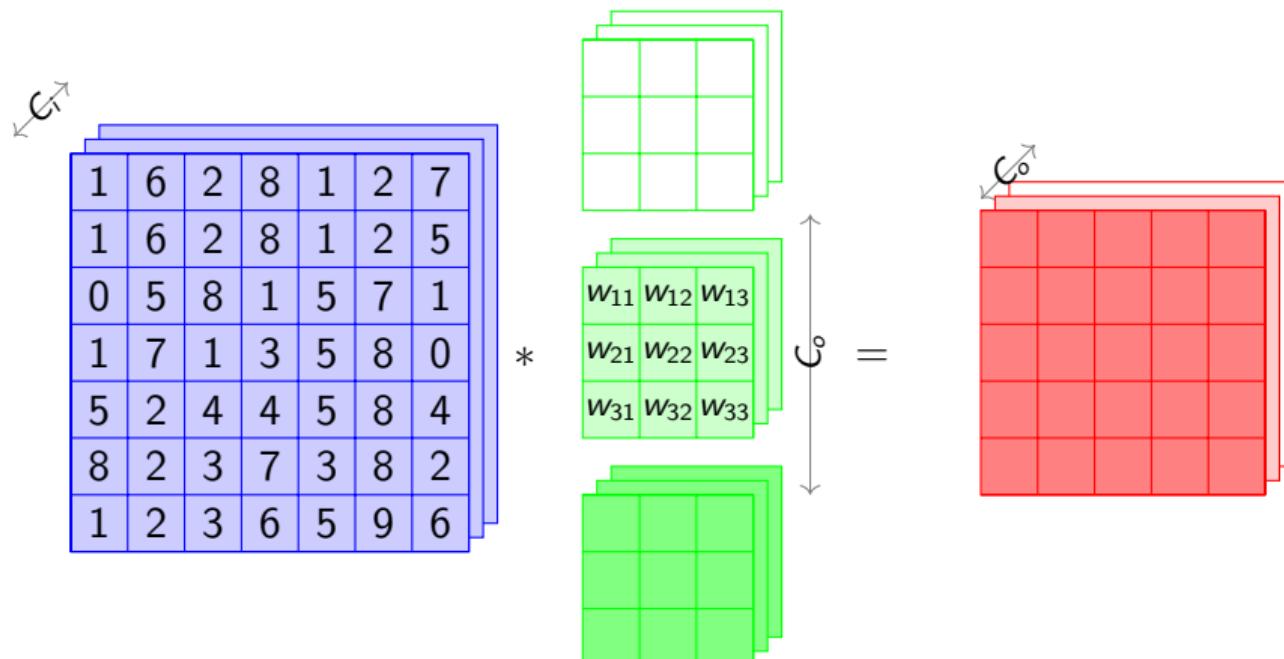
## 2D Convolution Multiple Input Channels



The convolution operation can be easily extended to handle multiple input channels.

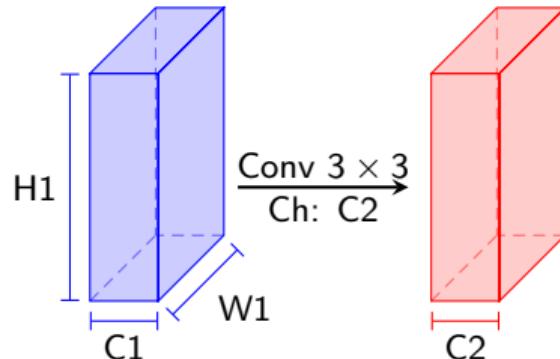
The number of channels in the weights filter *should* be equal to the number of channels in the input.

# Multiple Filters



We can have multiple weights filters. The number of output channels will now be equal to the number of weight filters ( $C_0$ ).

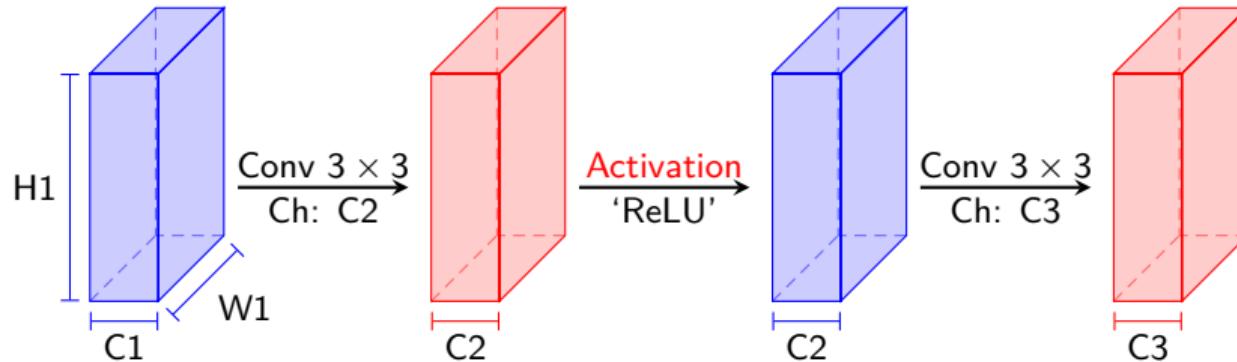
# Multiple Layers



We can represent a 2D convolution in tensor representation. The input to the convolution is a tensor of size  $[B, H_1, W_1, C_1]$  and the output is tensor of size  $[B, H_1, W_1, C_2]$ .  $B$  is the batch size (if padding is 'same').

```
tf.keras.layers.Conv2D(  
    filters, kernel_size, strides=(1, 1), padding='valid', data_format=None,  
    dilation_rate=(1, 1), groups=1, activation=None, use_bias=True,  
    kernel_initializer='glorot_uniform', bias_initializer='zeros',  
    kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None,  
    kernel_constraint=None, bias_constraint=None, **kwargs  
)
```

# Multiple Layers



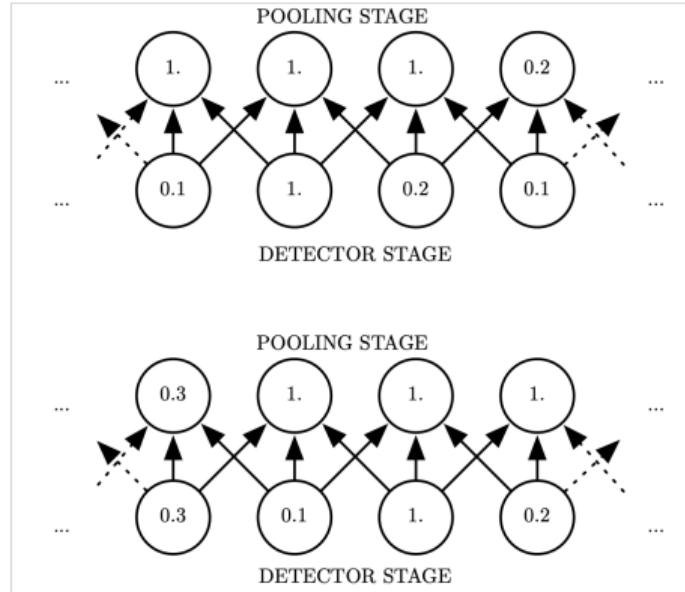
Convolutions can be stacked one after the other.

```
tf.keras.layers.Conv2D(  
    filters, kernel_size, strides=(1, 1), padding='valid', data_format=None,  
    dilation_rate=(1, 1), groups=1, activation=None, use_bias=True,  
    kernel_initializer='glorot_uniform', bias_initializer='zeros',  
    kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None,  
    kernel_constraint=None, bias_constraint=None, **kwargs  
)
```

# Outline

- 1 Motivation
- 2 2D Convolution in Traditional Computer Vision
- 3 Basic Convolution Operation
- 4 Pooling Operation
- 5 Variants of the Basic Convolution
- 6 The Neuro-scientific Basis for Convolutional Networks

# Invariance to Translation



# Invariance to Translation

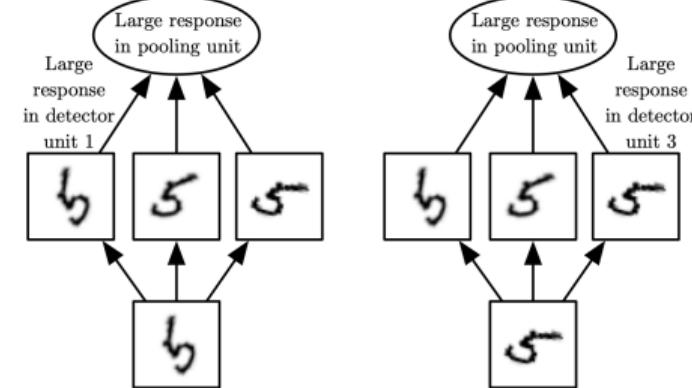
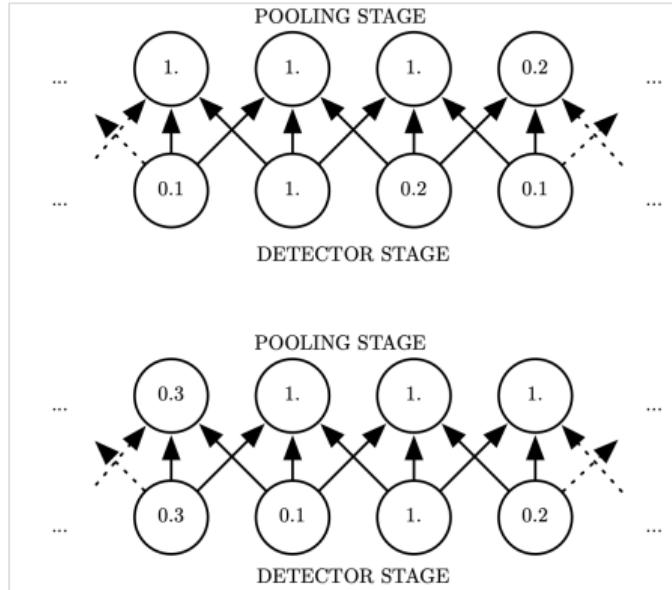


Image: Goodfellow, 2016

We would like our feature representations, not to change minimally when the input is shifted or rotated slightly.

# Pooling

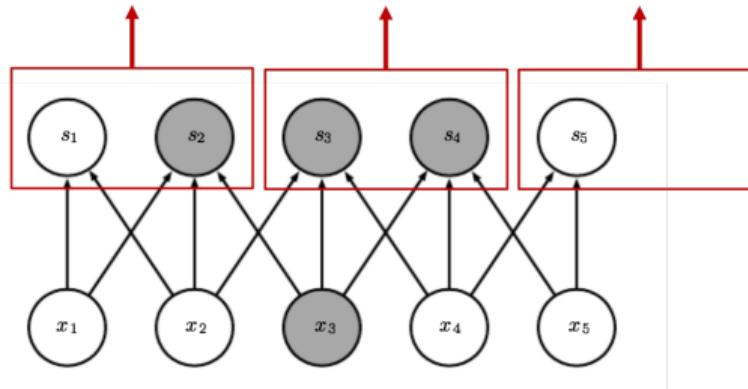


Image: Goodfellow, 2016



Pooling help reduce redundant information and provide some level of invariance to translations.

Pooling can use simple mathematical functions like max, sum , etc. “max-pooling” is the most common.

# 2D Pooling

1	6	2	8	1	2	7
1	6	2	8	1	2	5
0	5	8	1	5	7	1
1	7	1	3	5	8	0
5	2	4	4	5	8	4
8	2	3	7	3	8	2
1	2	3	6	5	9	6

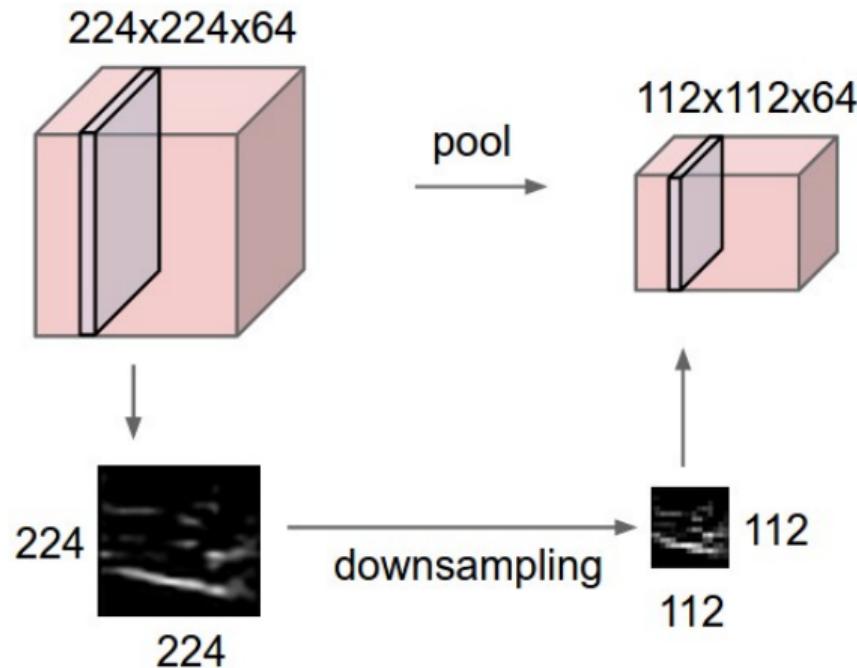
=

6	8	2
7		

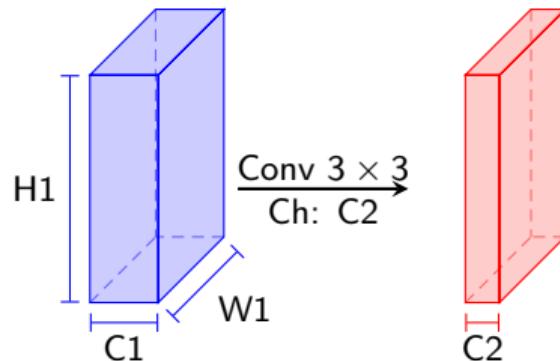
In this example we use 2D pooling with  $2 \times 2$  and stride 2.

Pooling is done each channel separately.

## 2D Pooling



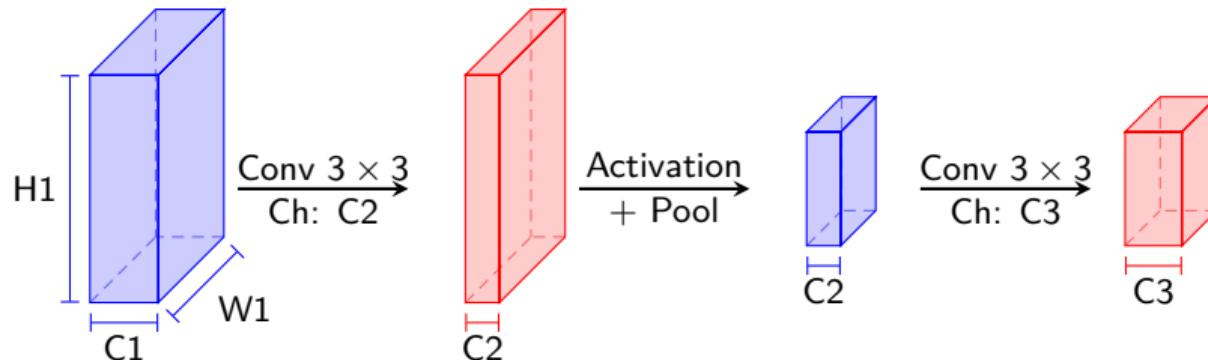
# Pooling & Convolutions



Convolutions can be combined with pooling to construct a chain of layers.

```
| tf.keras.layers.MaxPool2D(  
|     pool_size=(2, 2), strides=None, padding='valid', data_format=None, **kwargs  
| )
```

# Pooling & Convolutions

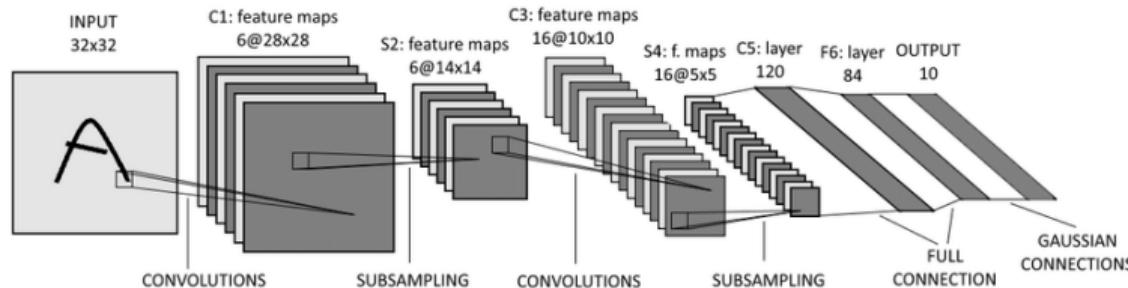


Convolutions can be combined with pooling to construct a chain of layers.

```
| tf.keras.layers.MaxPool2D(  
|     pool_size=(2, 2), strides=None, padding='valid', data_format=None, **kwargs  
| )
```

# LeNet Architecture

"LeNet is a classic example of convolutional neural network to successfully predict handwritten digits." [LeNet]



```
model = tf.keras.Sequential()
model.add(Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='tanh', input_shape=input_shape, padding="valid"))
model.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
model.add(Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))
model.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
model.add(Flatten())
model.add(Dense(120, activation='tanh'))
model.add(Dense(84, activation='tanh'))
model.add(Dense(10, activation='softmax'))
```

# Outline

- 1 Motivation
- 2 2D Convolution in Traditional Computer Vision
- 3 Basic Convolution Operation
- 4 Pooling Operation
- 5 Variants of the Basic Convolution**
- 6 The Neuro-scientific Basis for Convolutional Networks

# Strided Convolutions ✓

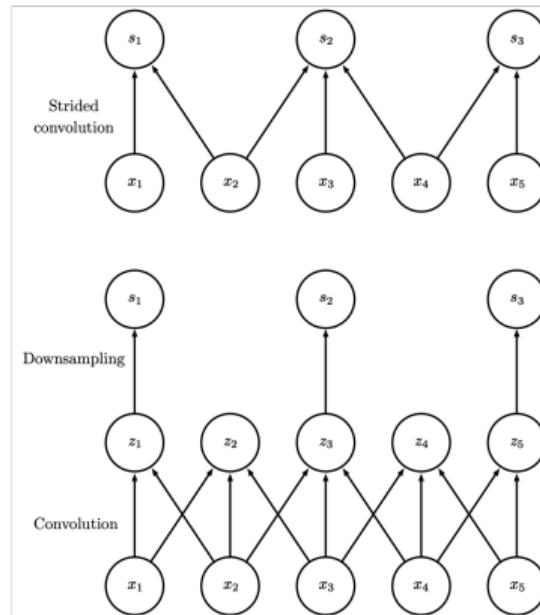
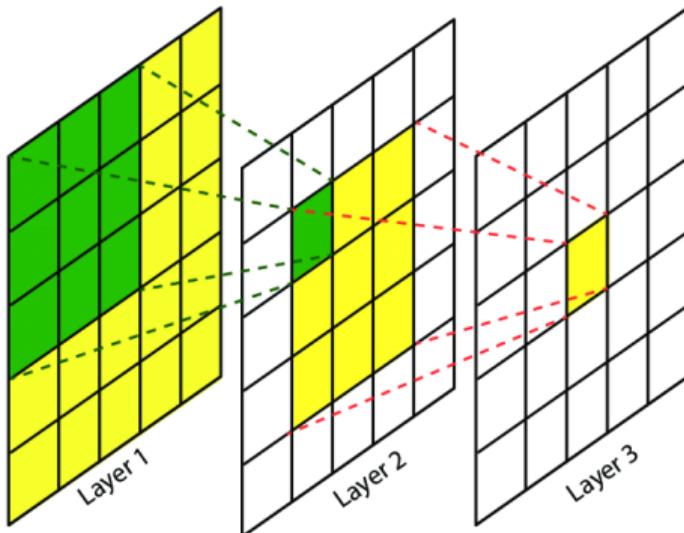


Image: Goodfellow, 2016

Can replace convolution + pooling with strided convolutions.

# Receptive Field ✓



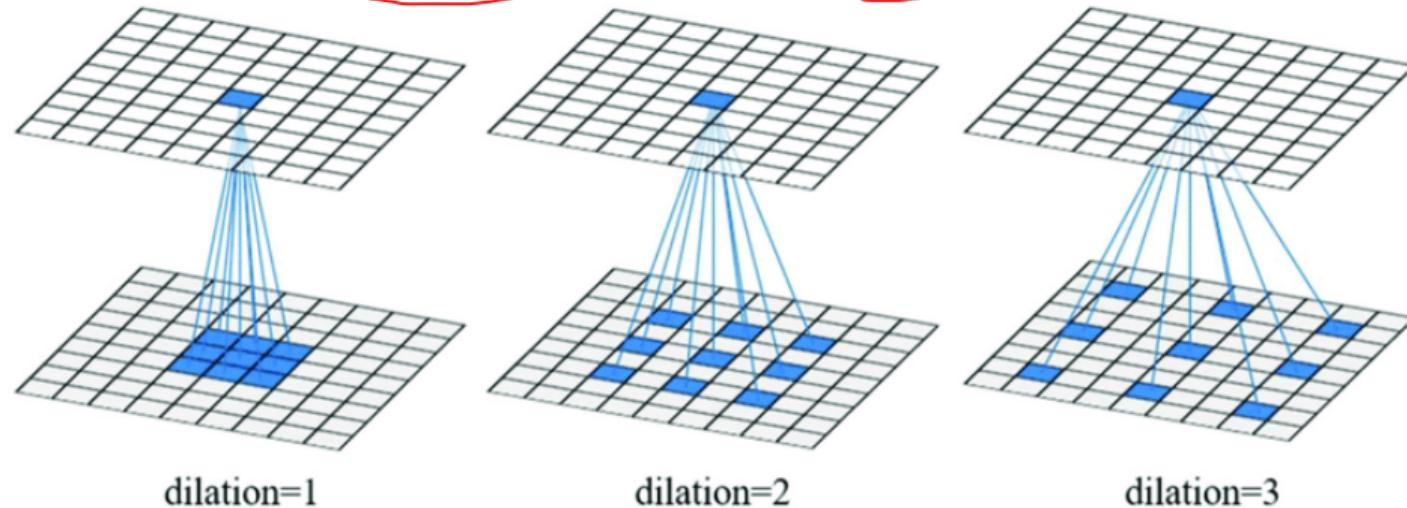
Assume 3 by 3 convolutions in each layer

✓ Receptive field of the convolutional network, is defined as the size of the region in the input that produces the feature.

✓ Receptive field increases with the number of layers (depth).

## Dilated Convolutions

What if you want to increase the receptive field without having so many layers?

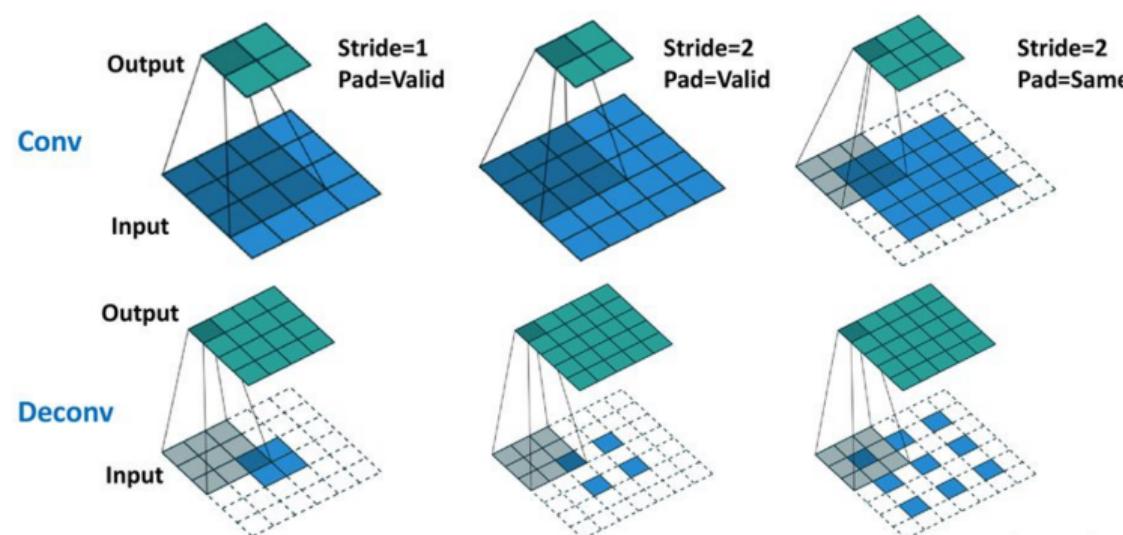


Dilated Convolutions delivers a wider field of view (receptive field) at the same computational cost. Also known as Atrous convolutions.

# Transpose Convolution

Convolution + pooling (or strided convolutions) are usually used to reduce the output tensor width and height in subsequent layers of a network.

What if you want to increase the output tensor width and height? You can use transpose convolution also known as deconvolution (e.g. Image segmentation).



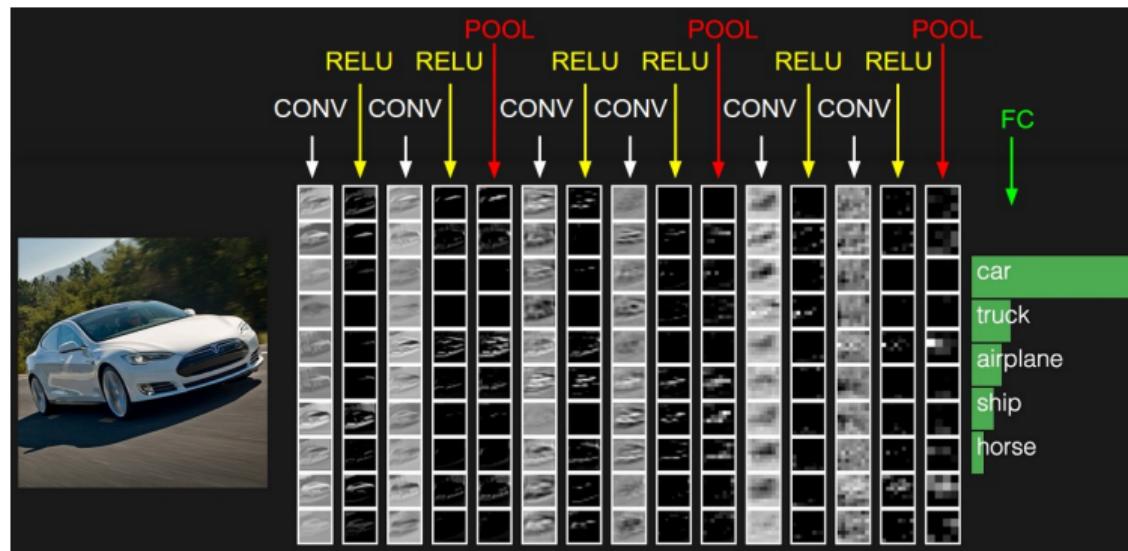
# Outline

- 1 Motivation
- 2 2D Convolution in Traditional Computer Vision
- 3 Basic Convolution Operation
- 4 Pooling Operation
- 5 Variants of the Basic Convolution
- 6 The Neuro-scientific Basis for Convolutional Networks

## Feature Maps

Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future

Visualizing and Understanding Deep Neural Networks by Matt Zeiler



# Gabor Kernels

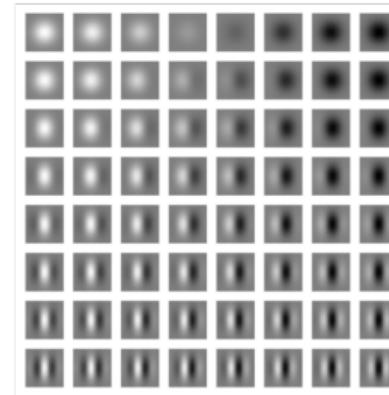
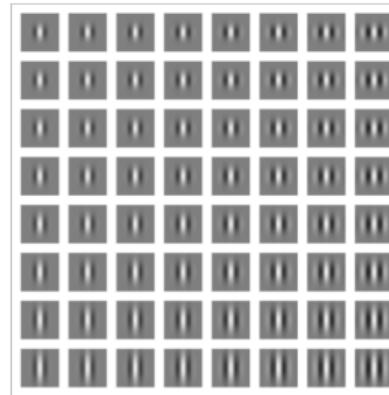
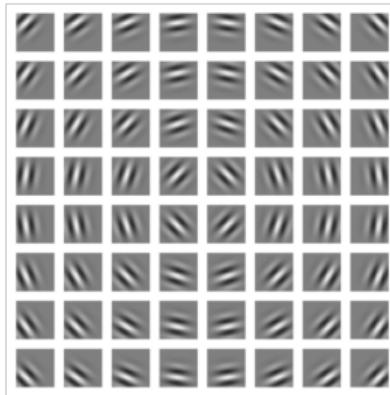


Image: Goodfellow, 2016

Gabor functions with a variety of parameter settings.

# Gabor-like Learned Kernels

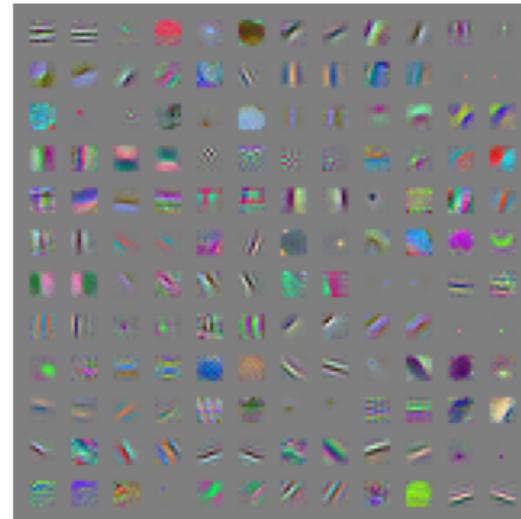
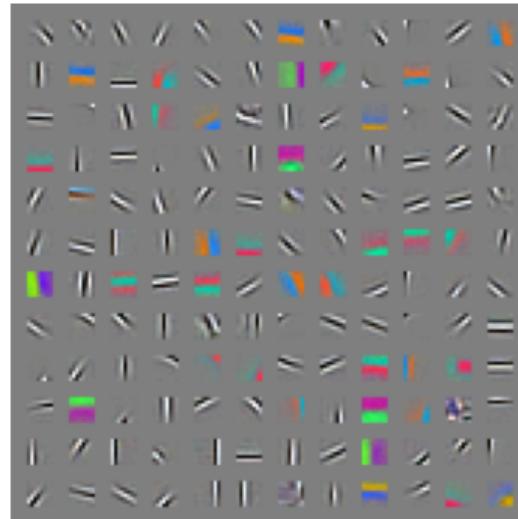


Image: Goodfellow, 2016

Many machine learning algorithms learn features that detect edges or specific colors of edges when applied to natural images. These feature detectors are reminiscent of the **Gabor functions** known to be present in the primary visual cortex.

# Summary

- ① Main components of CNN and why they work.
- ② Extensions of the basic convolution operation.

## Next week:

- ① Popular Convolutional neural network architectures.