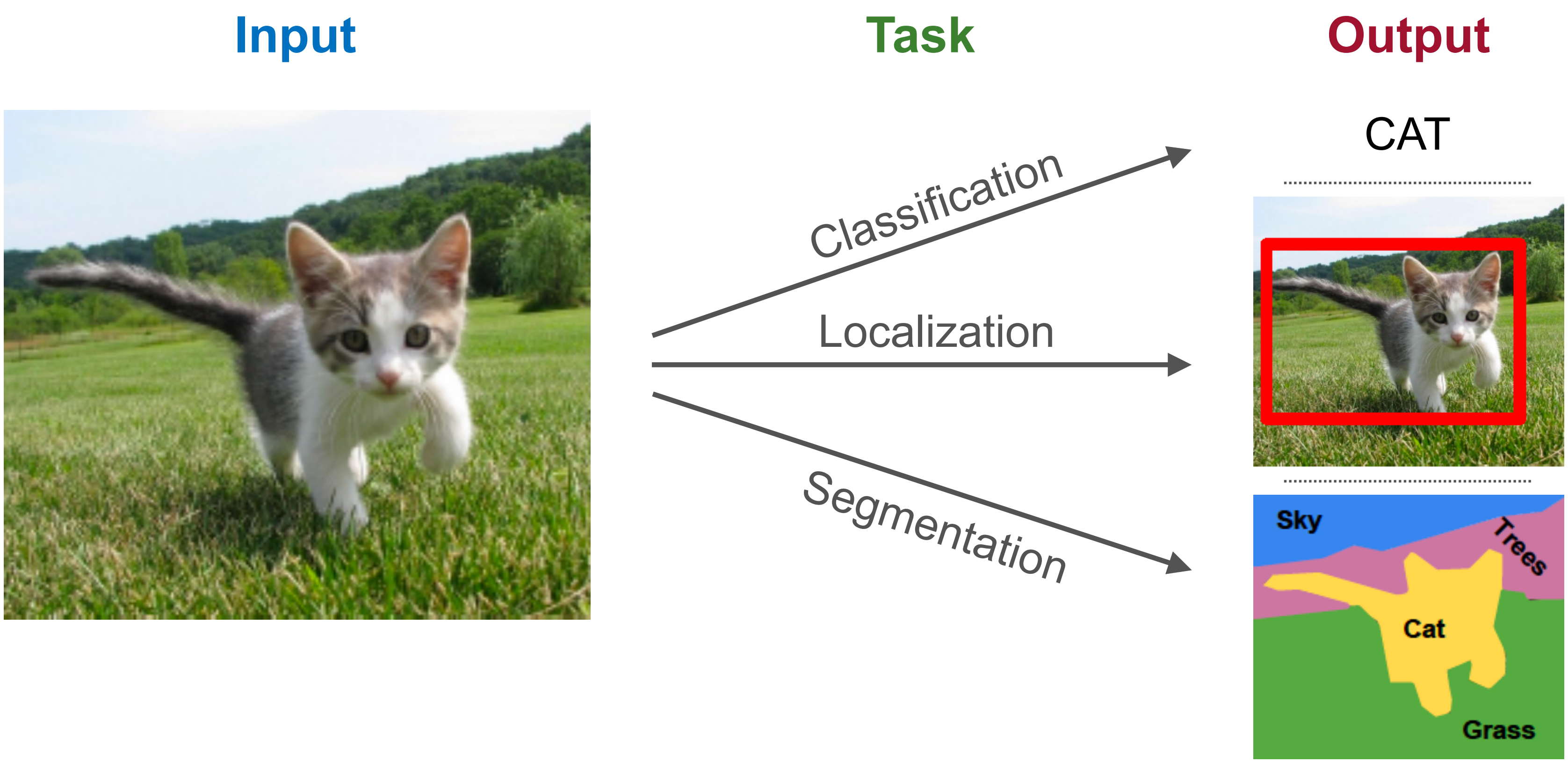
# Computer vision

Image-related tasks
Fully-connected layer
Bottleneck of standard networks

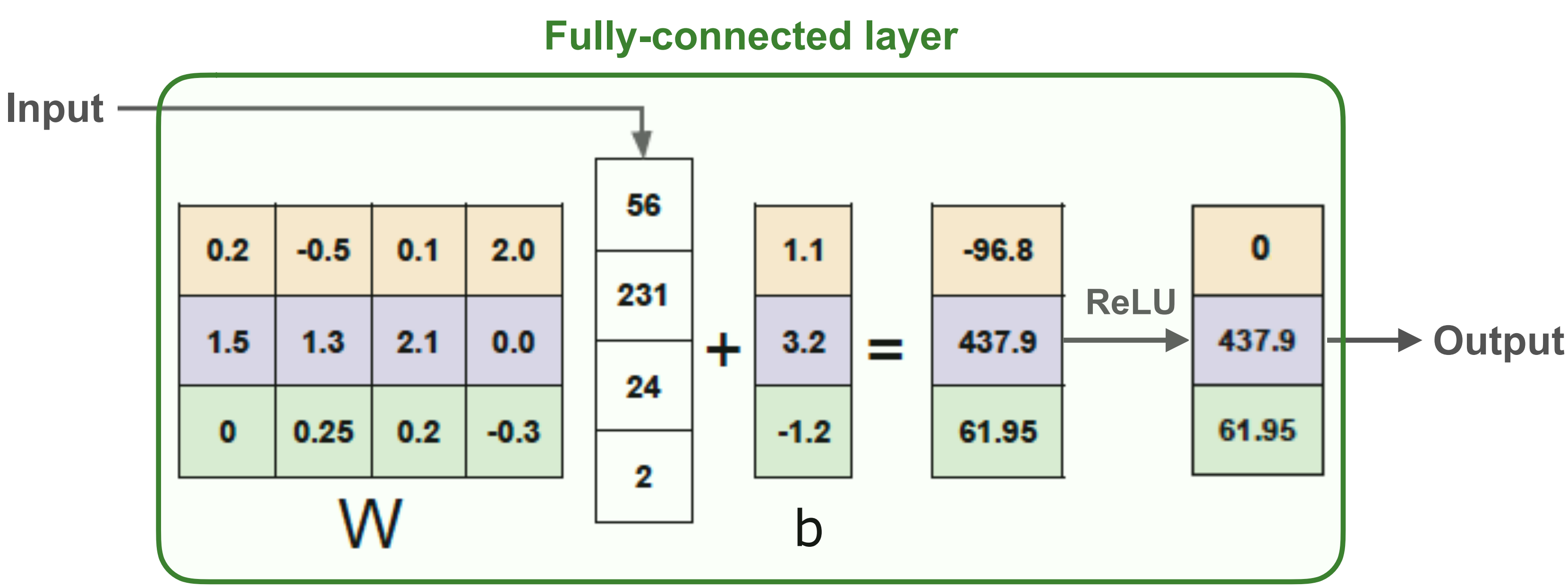
# Image-related tasks

- Neural networks can be used for many things…



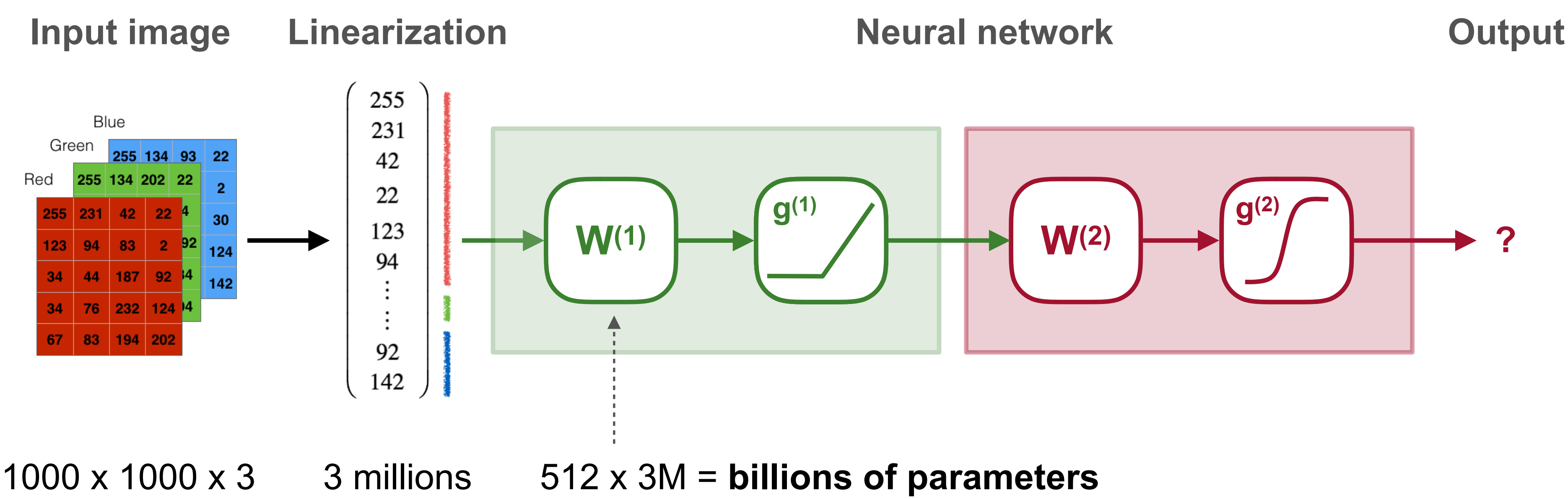
# Fully-connected layer

- Standard networks are made of **fully-connected layers**
  - *Each layer is a matrix multiplication*
  - *Each layer contains **"(input dim. + 1) x hidden dim."** parameters*



# Bottleneck of standard networks

- Fully-connected layers are **unsuitable for images**
  - *Too many parameters to train*
  - *Difficult to prevent overfitting*

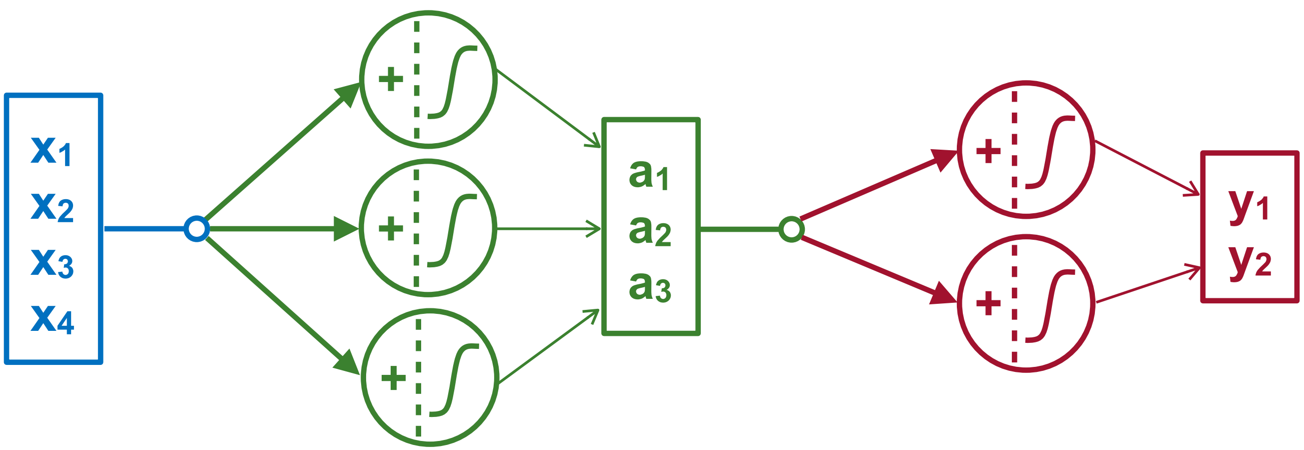


# Quiz

- Consider the two-layer network shown below.

  1) *How many parameters have the hidden layer?*

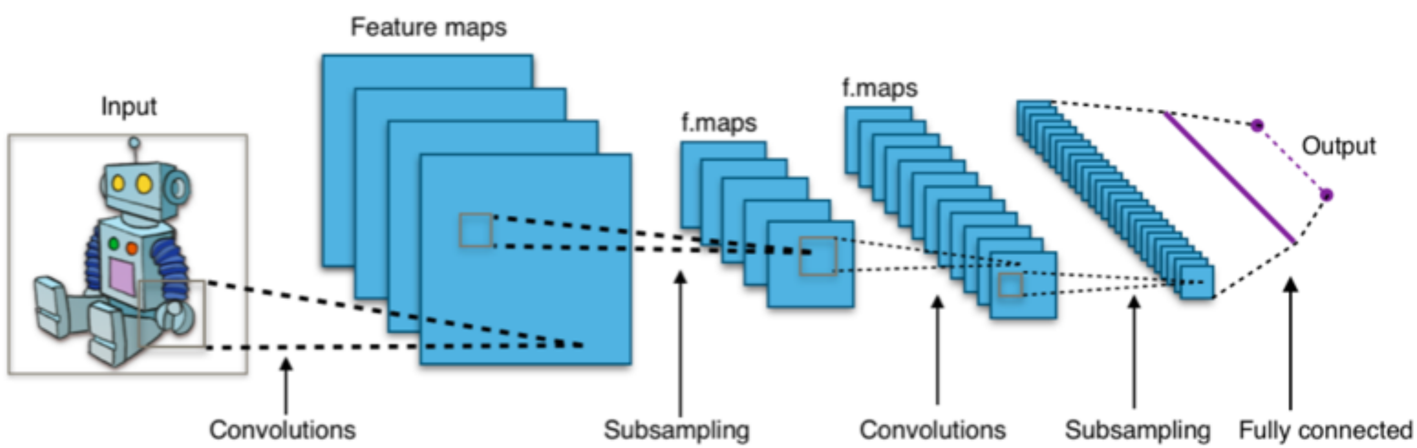
  2) *How many parameters have the output layer?*



# What we have seen so far…

- Standard networks are limited to **low-dimensional data**
  - *Too many parameters when the input is high-dimensional*
  - *How to deal with high-resolution images?*

- Solution for image inputs ➜ **Convolutional networks**



# Convolution

Convolution in 1D/2D/3D

Edge detection

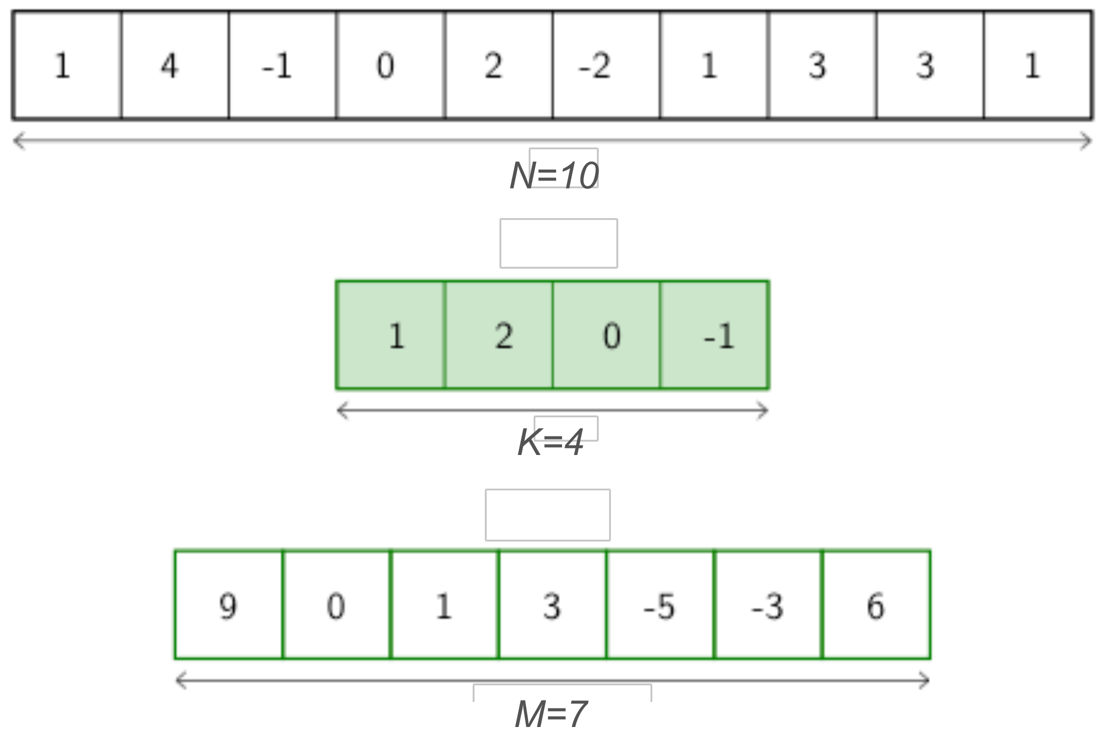
# Convolution 1D (1/2)

- Convolution with **vectors**

***Input***
*(vector of size **N**)*

***Kernel***
*(vector of size **K<N**)*

***Output***
*(vector of size **N-K+1**)*



# Convolution 1D (2/2)

- Visual explanation of convolution in 1D
  - *At each step, the kernel is multiplied to a chunk of the input…*
  - *… and the resulting coefficients are summed*

# Convolution 2D (1/2)

- Convolution with **images**

***Input***

*(size $N_1$ x $N_2$)*

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

*

***Kernel***

*(size $K_1$ x $K_2$)*

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

=

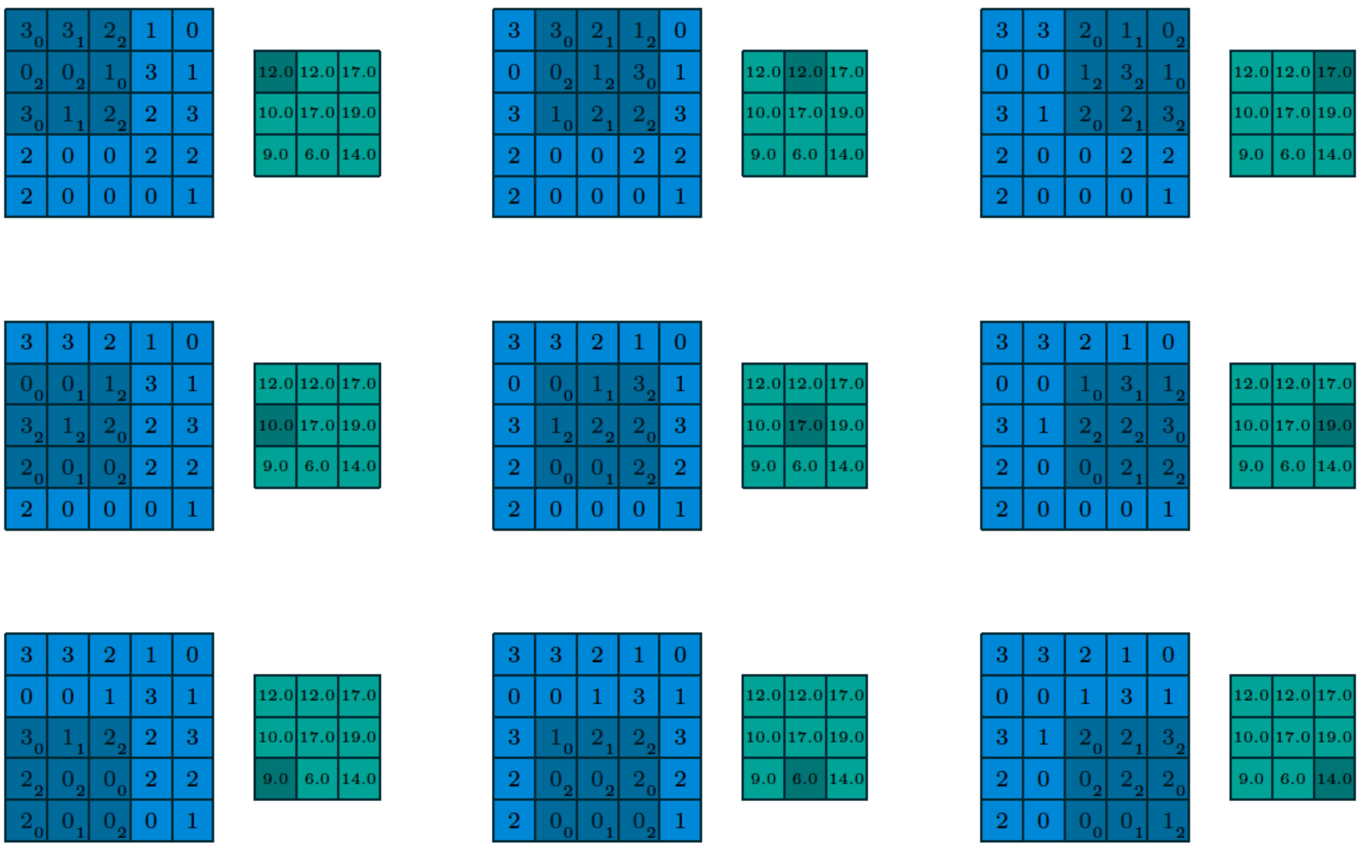
***Output***

*(size $N_1$-$K_1$+1 x $N_2$-$K_2$+1)*

| | | | |
|---|---|---|---|
| -5 | -4 | 0 | 8 |
| -10 | -2 | 2 | 3 |
| 0 | -2 | -4 | -7 |
| -3 | -2 | -3 | -16 |

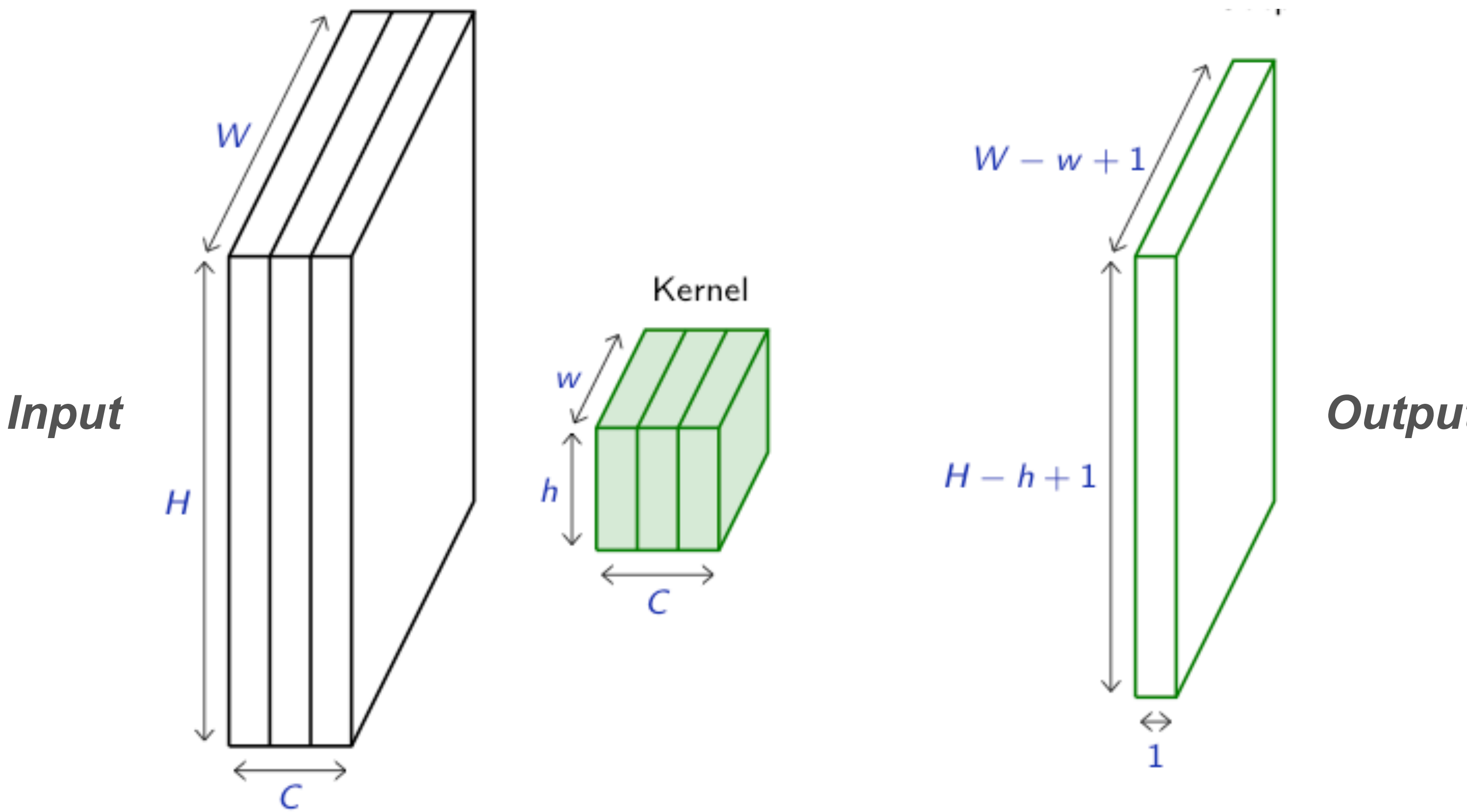
# Convolution 2D (2/2)

- Visual explanation of convolution in 2D



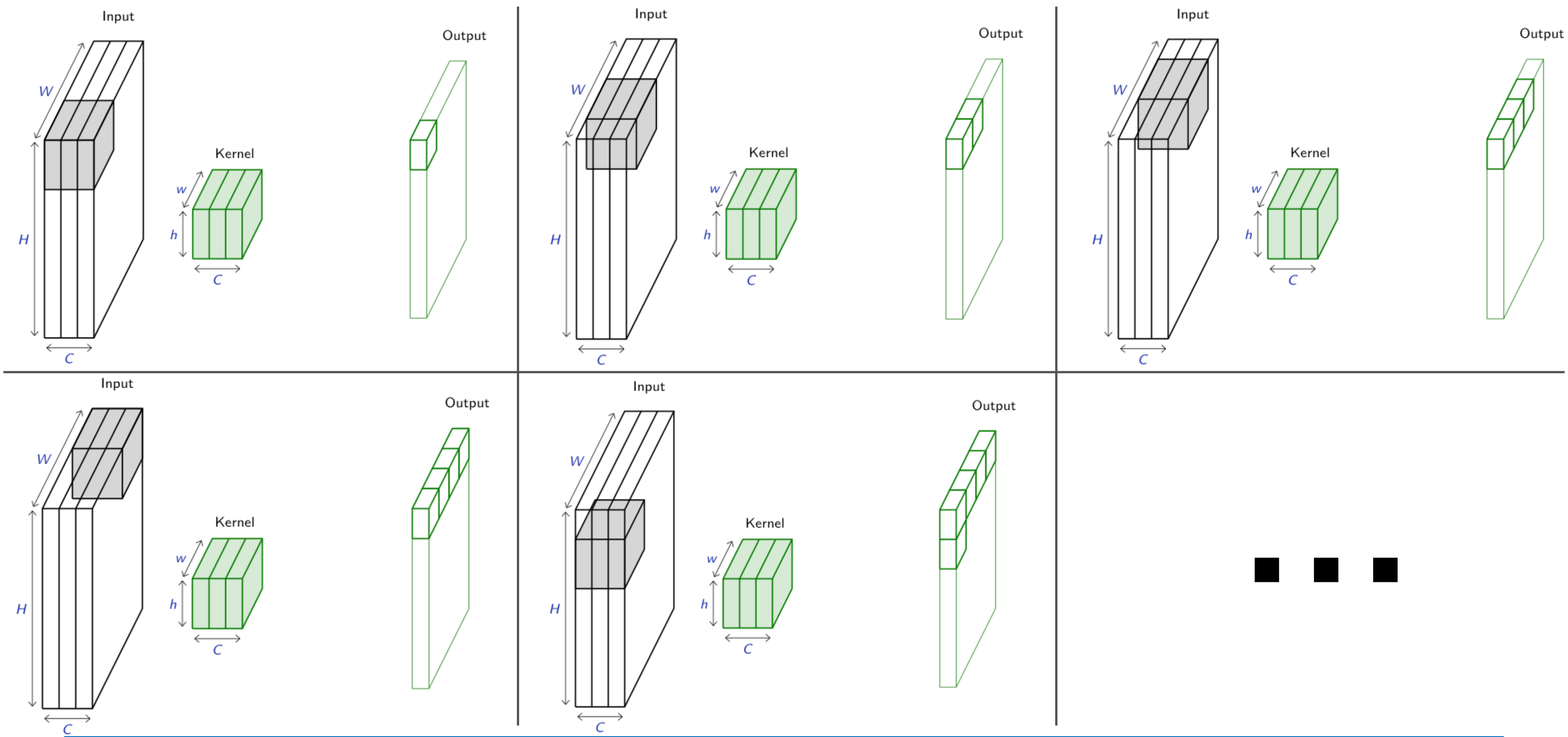
# Convolution 3D (1/2)

- Convolution with **multi-channel images**



# Convolution 3D (2/2)

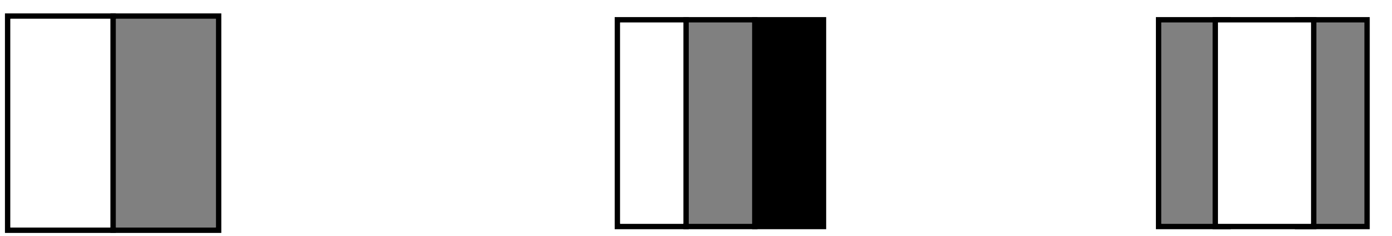
- Visual explanation of convolution in 3D



# Edge detection (1/3)

- **Vertical edge detection**

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$



# Edge detection (2/3)

- **Horizontal edge detection**

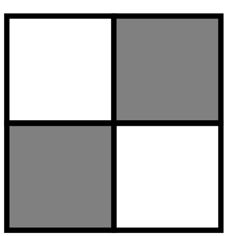
| | | | | | |
|---|---|---|---|---|---|
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 0 | 0 | 0 | 10 | 10 | 10 |
| 0 | 0 | 0 | 10 | 10 | 10 |
| 0 | 0 | 0 | 10 | 10 | 10 |

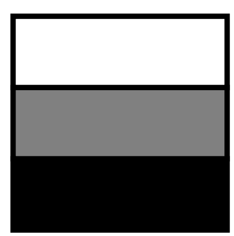
$*$

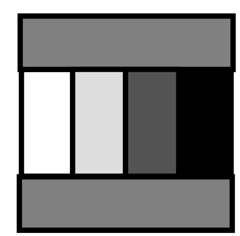
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

$=$

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 30 | 10 | -10 | -30 |
| 30 | 10 | -10 | -30 |
| 0 | 0 | 0 | 0 |



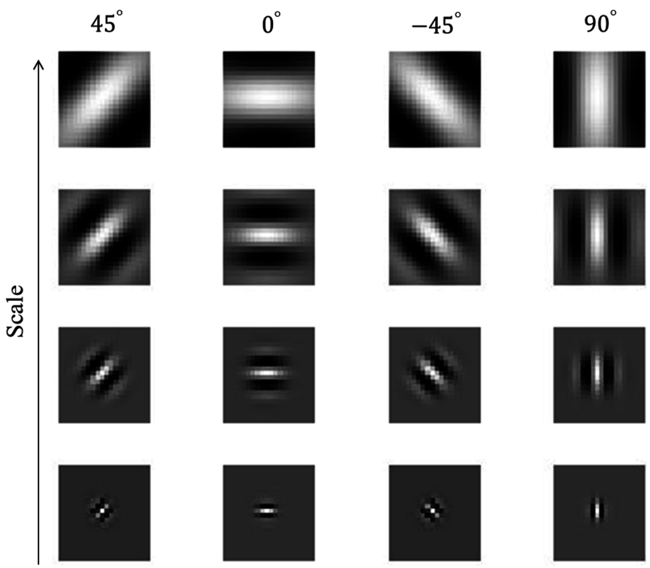




# Edge detection (3/3)

- **Putting all together**
  - *A different kernel for each orientation and scale*

**Kernels**




Convolutions

**Output**

(Combination of all the convolved images)

# Quiz

1) What is the output size of the following convolutions?
   - ***10x12** matrix convolved by **5x5** kernel*
   - ***15x15x10** volume convolved by **3x3x3** kernel*

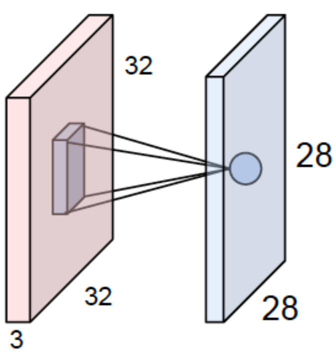
2) Compute the following convolution.

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
3 & 0 & 1 & 2 & 7 & 4 \\ \hline
1 & 5 & 8 & 9 & 3 & 1 \\ \hline
2 & 7 & 2 & 5 & 1 & 3 \\ \hline
0 & 1 & 3 & 1 & 7 & 8 \\ \hline
4 & 2 & 1 & 6 & 2 & 8 \\ \hline
2 & 4 & 5 & 2 & 3 & 9 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
1 & 1 & 1 \\ \hline
1 & 1 & 1 \\ \hline
1 & 1 & 1 \\ \hline
\end{array}
\;=\;
\begin{array}{|c|c|c|c|}
\hline
\quad & \quad & \quad & \quad \\ \hline
\quad & \quad & \quad & \quad \\ \hline
\quad & \quad & \quad & \quad \\ \hline
\quad & \quad & \quad & \quad \\ \hline
\end{array}
$$

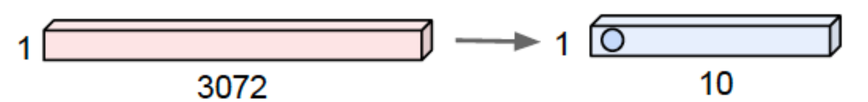
# What we have seen so far…

- **Convolution ➜** spatially-invariant operation
  - *The same weights are used for computing the output coefficients*
- **Matrix multiplication ➜** general operation
  - *Different weights are used for computing the output coefficients*

**Convolution**
*(5x5 kernel)*




**Matrix multiplication**
*(10x3072 matrix)*



# Convolutional layer

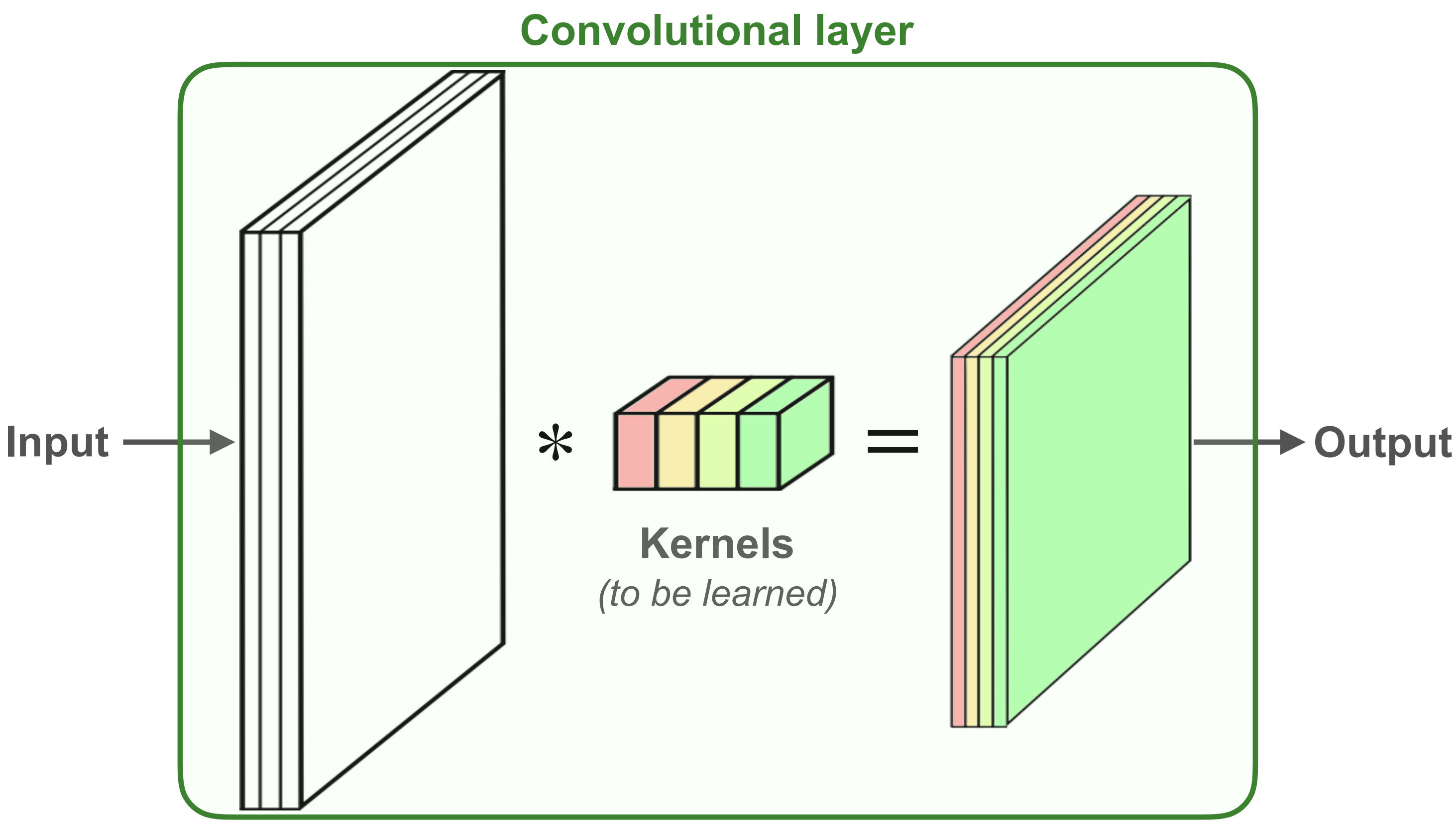
Multiple kernels

Hyper-parameters

Padding & Stride

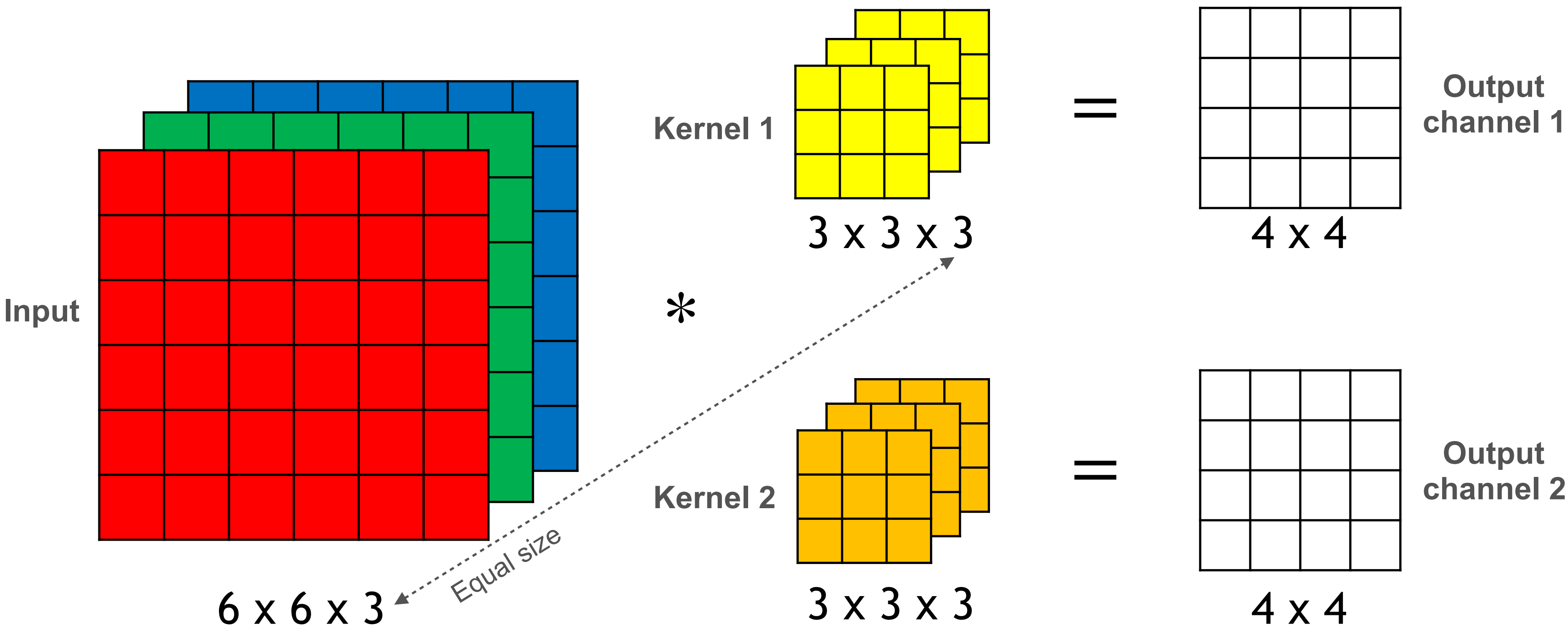
# Multiple kernels (1/2)

- The input is convolved with **multiple kernels**



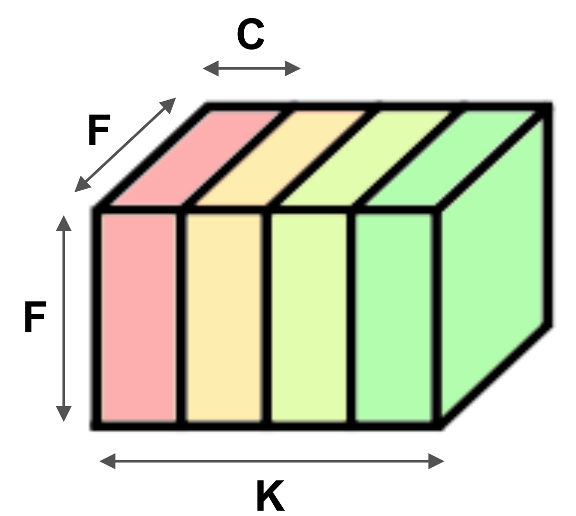
# Multiple kernels (2/2)

- Each convolution produces a channel for the output
  - *The **kernel depth** must be equal to the number of **input channels***



# Hyper-parameters

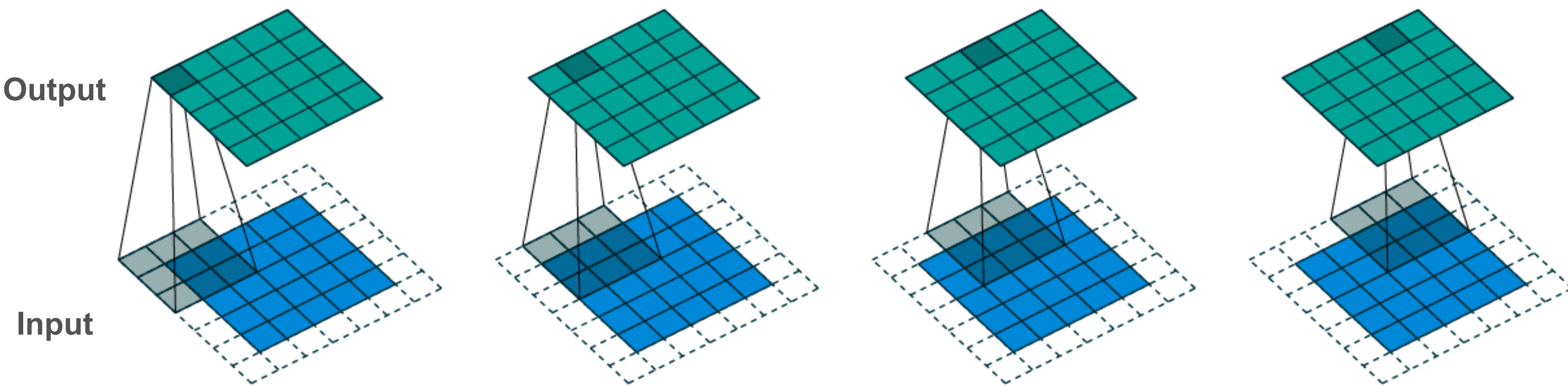
- Main hyper-parameters
  - ***F* ➜ *Spatial size of kernels***
  - ***K* ➜ *Number of kernels***
- Other hyper-parameters
  - ***P* ➜ *Padding***
  - ***S* ➜ *Stride***
- Fixed value
  - ***C* ➜ *Kernel depth*** *(equal to the number of input channels)*



# Padding & stride (1/3)

- The input can be **padded** with zero rows/columns
  - *The output size is extended by **P > 0***

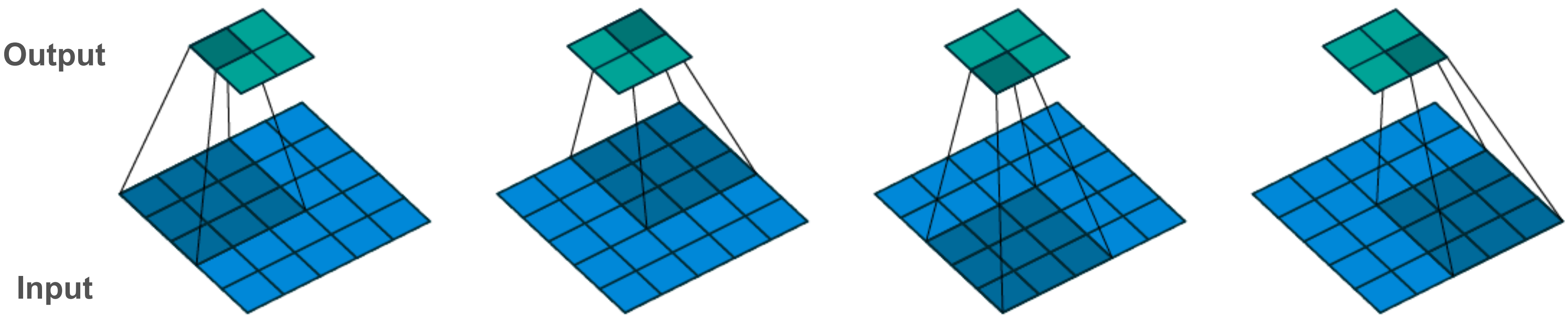
Example with **P=1**



# Padding & stride (2/3)

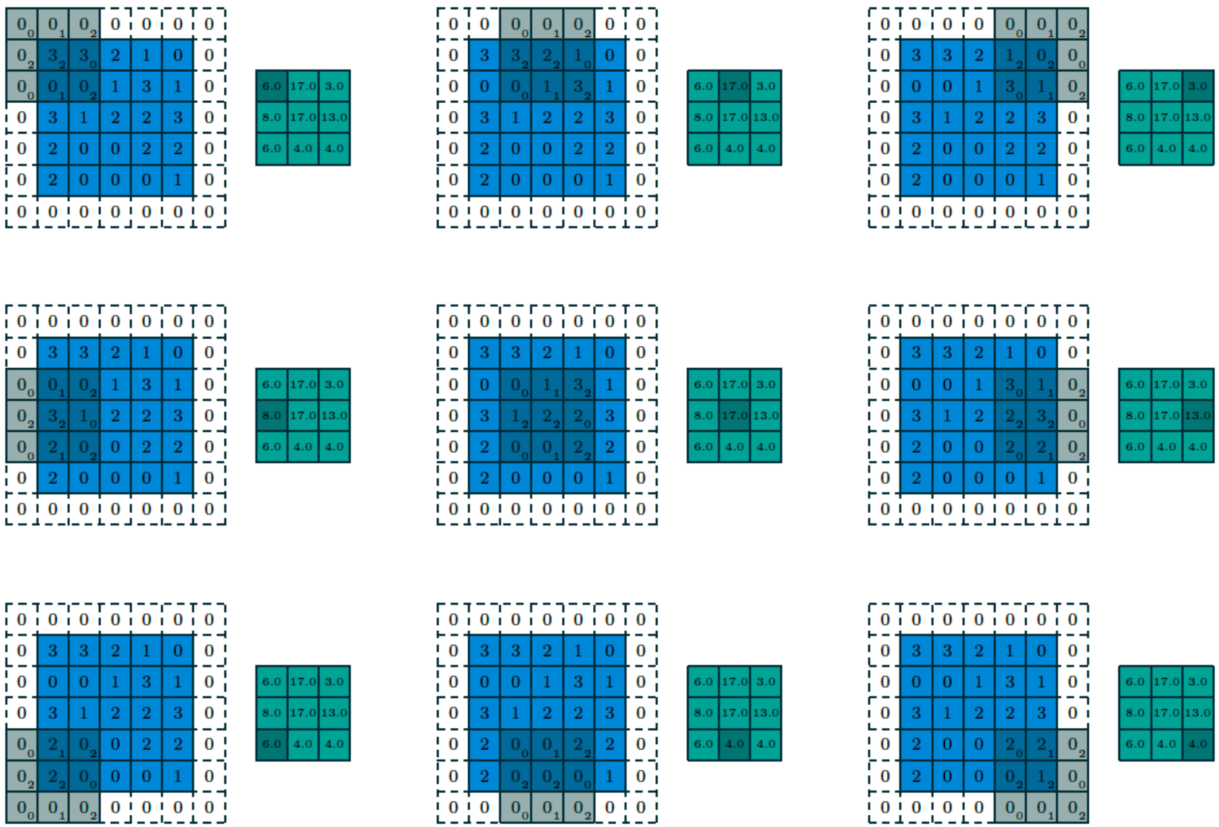
- The output can be **downsampled** along the rows/columns
  - *The output size is reduced by a factor **S > 1***

Example with **S=2**



# Padding & stride (3/3)

- Example with **padding** **(P=1)** and **stride** **(S=2)**

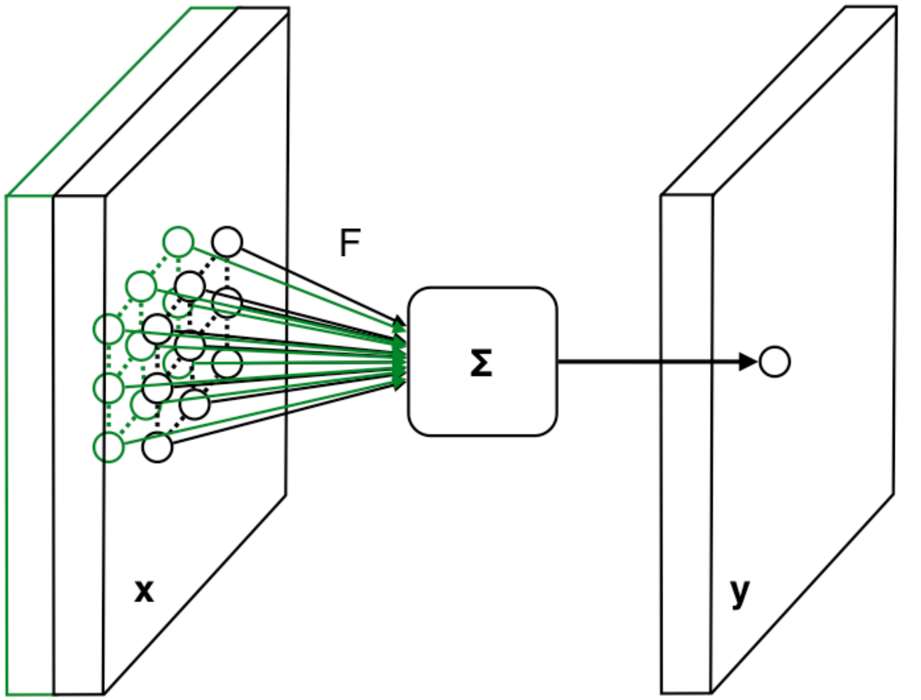


# Neural interpretation

- A convolution is a neuron with limited reception field
  - *The field limit is defined by the kernel size*
  - *The same neuron is "fired" over multiple areas from the input.*

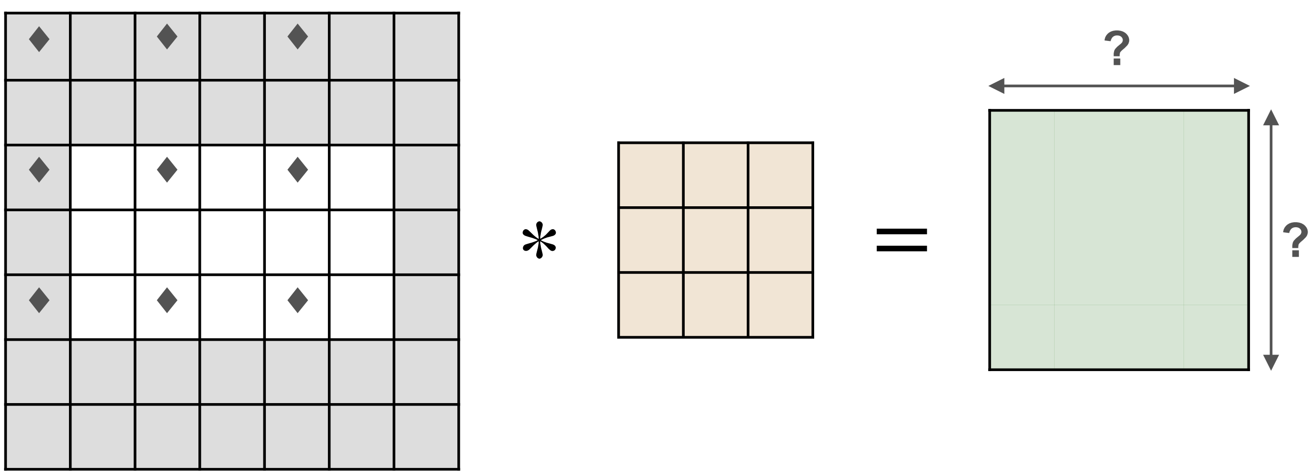
**Local**
Filters look locally

**Translation invariant**
Filters act the same everywhere



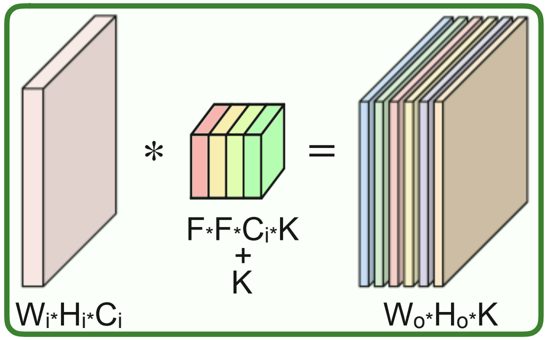
# Quiz

- Compute the output shape of a convolutional layer with
  - ***Input*** ➜ *3 x 5 x C*
  - ***Kernel*** ➜ *3 x 3 x C x 2*
  - ***Padding*** ➜ *2 x 1*
  - ***Stride*** ➜ *2 x 2*



# What we have seen so far…

- **Convolutional layer**
  - The input is a volume of size $\mathbf{W_i}$ x $\mathbf{H_i}$ x $\mathbf{C_i}$
  - Four hyper-parameters are required
    - ***K*** ➜ *Number of kernels*
    - ***F*** ➜ *Spatial size*
    - ***S*** ➜ *Stride*
    - ***P*** ➜ *Padding*



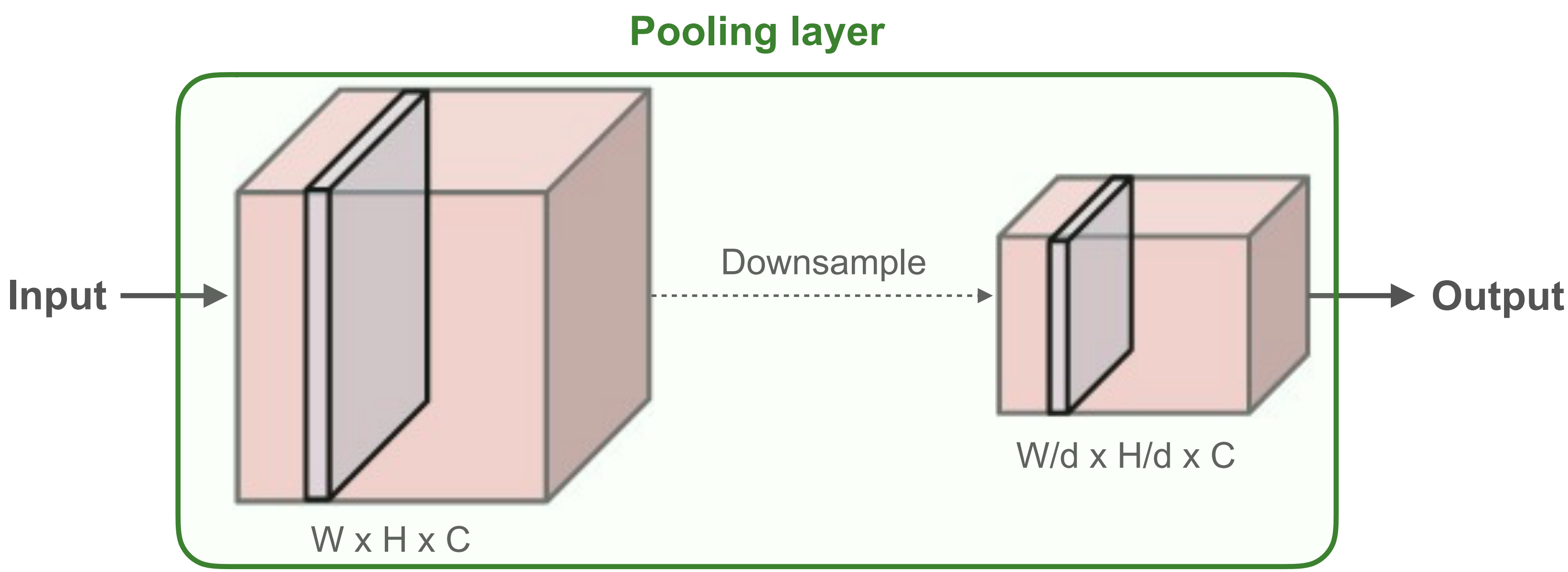

  - The output is a volume of size $\mathbf{W_o}$ x $\mathbf{H_o}$ x $\mathbf{C_o}$
    - $W_o = (W_i - F + 2P) / S + 1$
    - $H_o = (H_i - F + 2P) / S + 1$
    - $C_o = K$
  - The number of parameters to be learned is $\mathbf{(F \times F \times C_i + 1) \times K}$

# Pooling layer

Downsampling
Max-pooling
Translation invariance

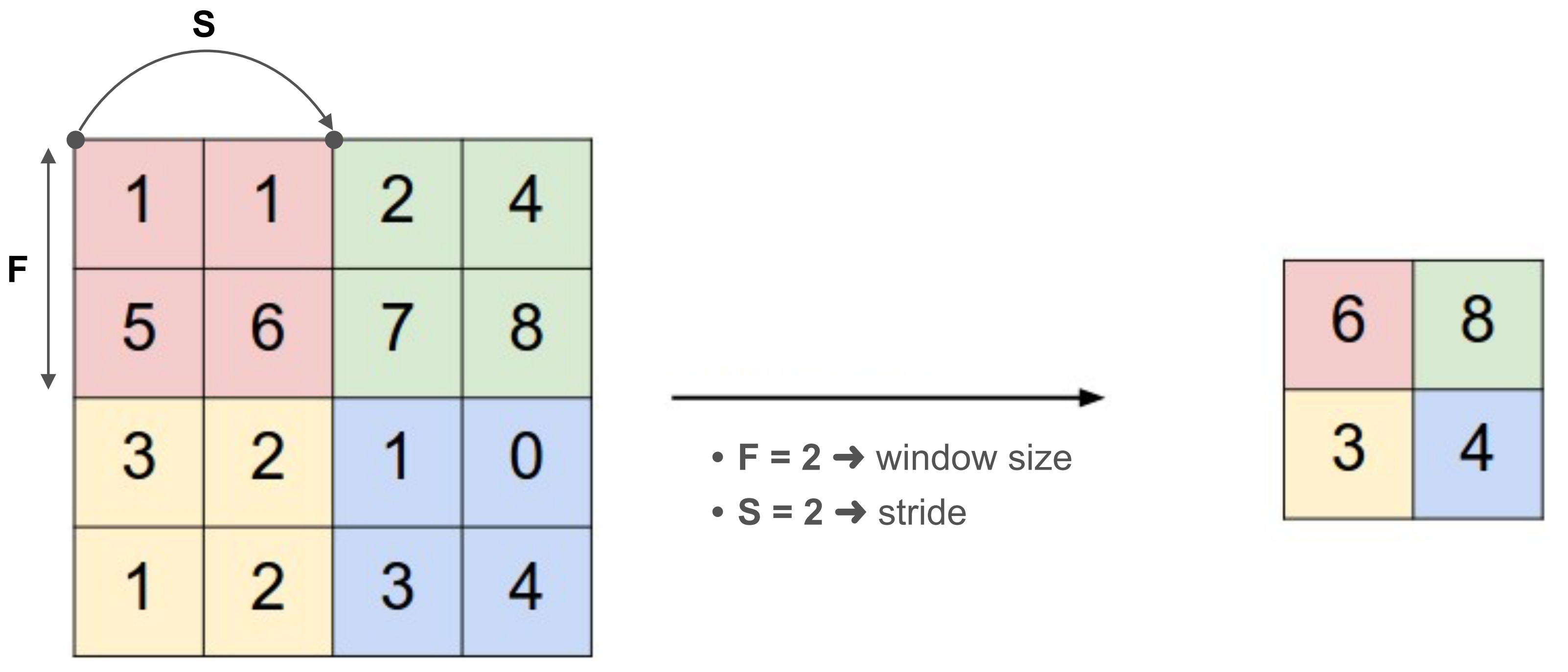
# Downsampling

- The **pooling layer** reduces the spatial size of the input
  - *It operates over each channel map independently*
  - *It has no parameter to be learned*



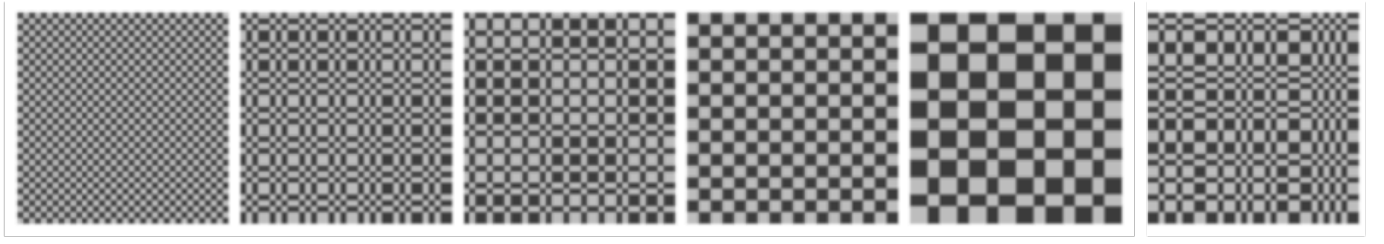
# Max-pooling (1/2)

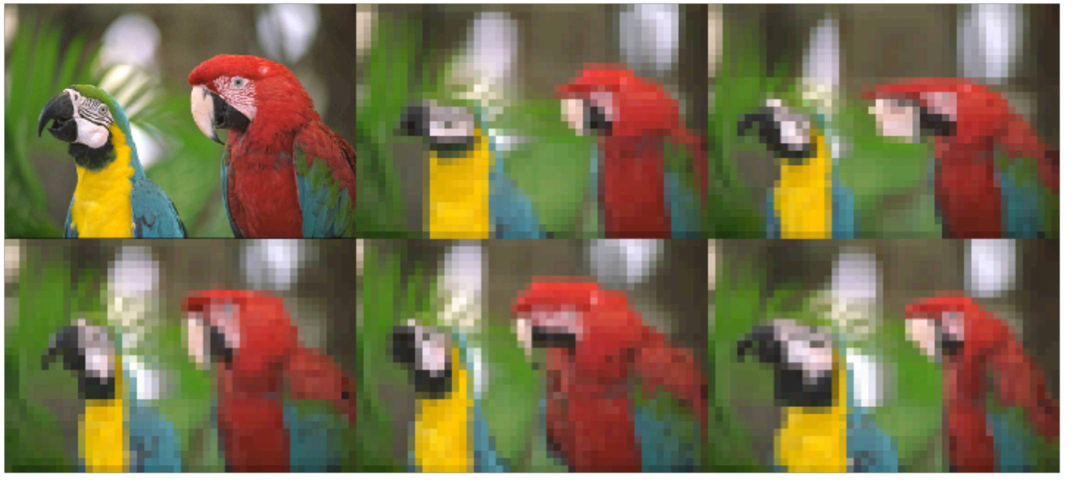
- The most-common variant is **max-pooling**
  - *It computes the max over a sliding window*
  - ***Hyper-parameters*** ➜ *window size (**F**) & stride (**S**)*



# Max-pooling (2/2)

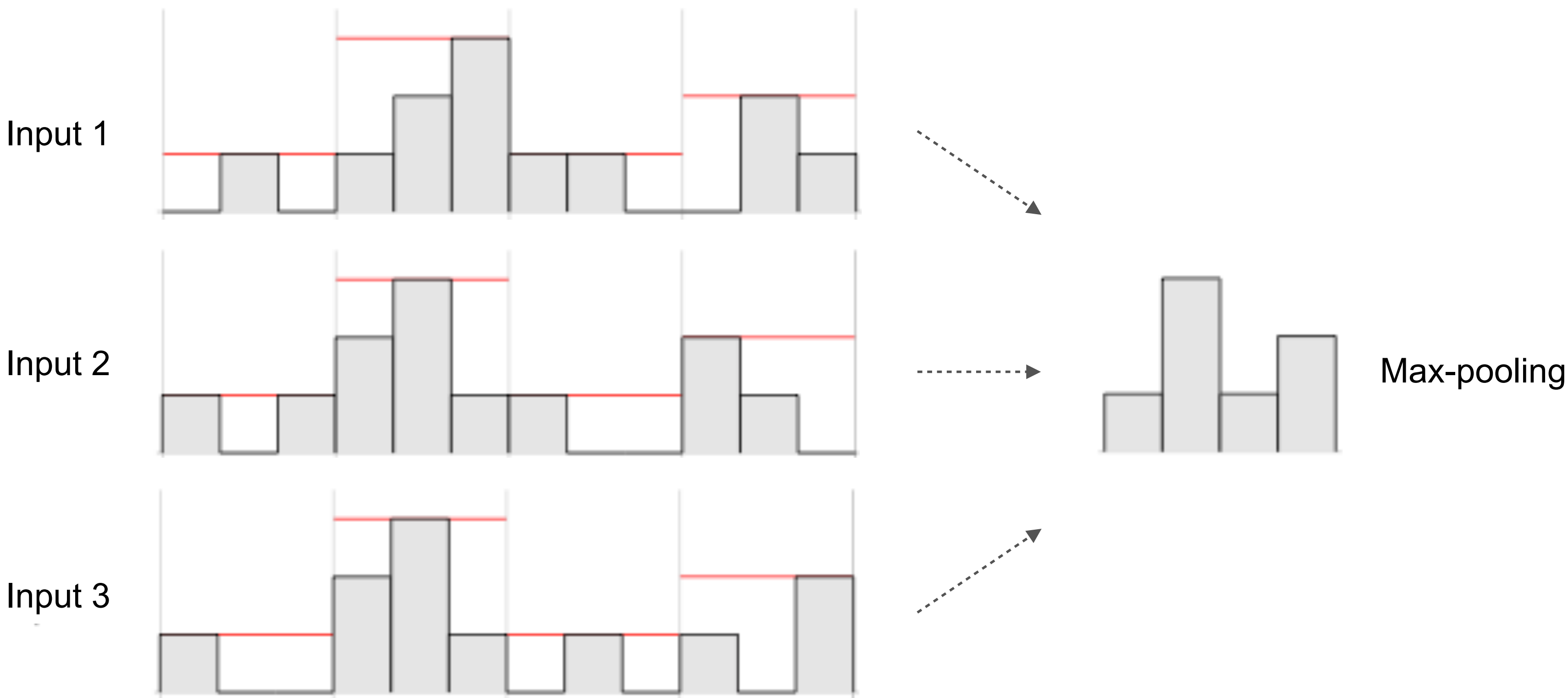
- **Stochastic pooling** ➜ Random pooling mask at each pass





# Translation invariance

- Pooling and convolution are **translation invariant**
  - *A spatial shift in the input doesn't change the output*



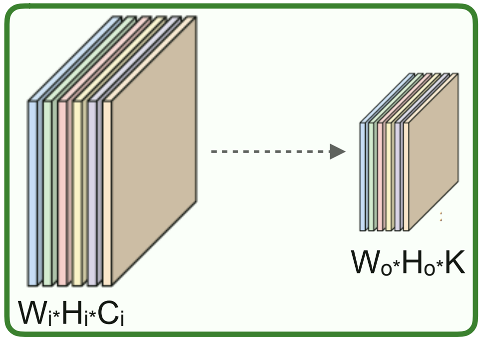
# Quiz

1) Because pooling layers do not have parameters, they do not affect the gradient calculation.
   A. *True*
   B. *False*

2) You have an input volume that is **32x32x16**, and apply max-pooling with a **stride of 2** and a **window size of 2**. What is the output volume?
   A. *16x16x8*
   B. *32x32x8*
   C. *16x16x16*
   D. *15x15x16*

# What we have seen so far…

- **Pooling layer**
  - The input is a volume of size $\mathbf{W_i}$ x $\mathbf{H_i}$ x $\mathbf{C_i}$
  - Two hyper-parameters are required
    - ***F*** ➜ *Window size*
    - ***S*** ➜ *Stride*
  - The output is a volume of size $\mathbf{W_o}$ x $\mathbf{H_o}$ x $\mathbf{C_o}$
    - $W_o = (W_i - F) / S + 1$
    - $H_o = (H_i - F) / S + 1$
    - $C_o = C_i$
  - No parameters to be learned




- **Note ➜** Pooling can be replaced with a stride in convolutional layers!

# Convolutional network

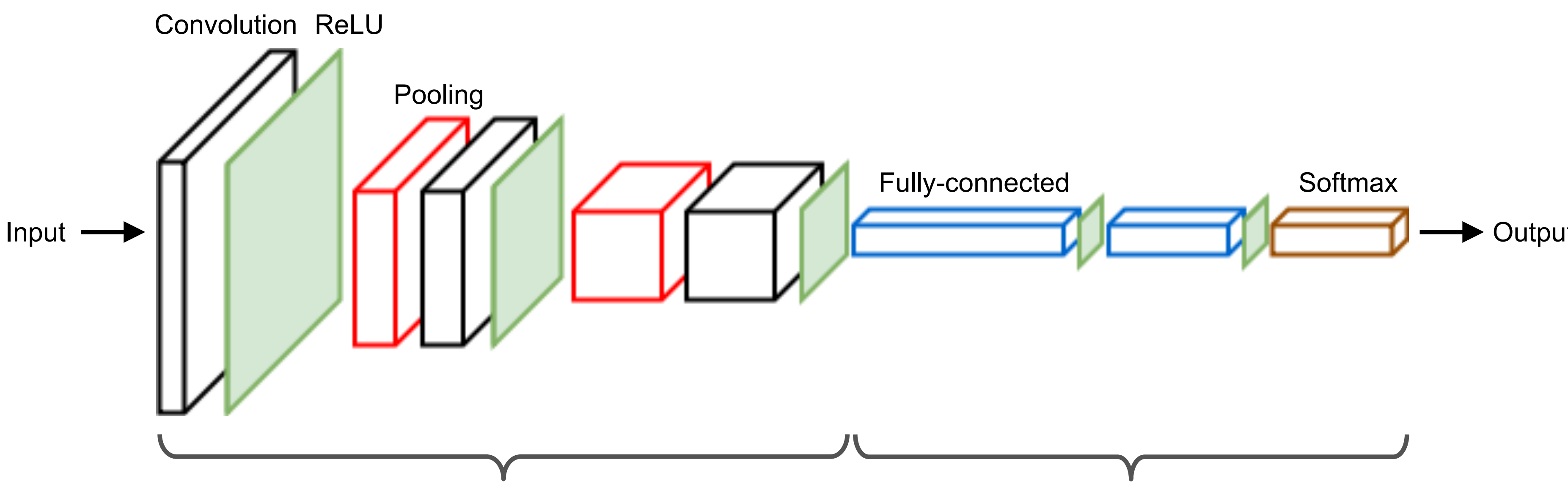
Building blocks

Classic architectures

Modern architectures

# Building blocks (1/2)

- Architecture of a **convolutional neural network**
  - ***Base*** ➜ *Convolution + pooling*
  - ***Head*** ➜ *Fully-connected + output layer*




**Feature extractor** (base)

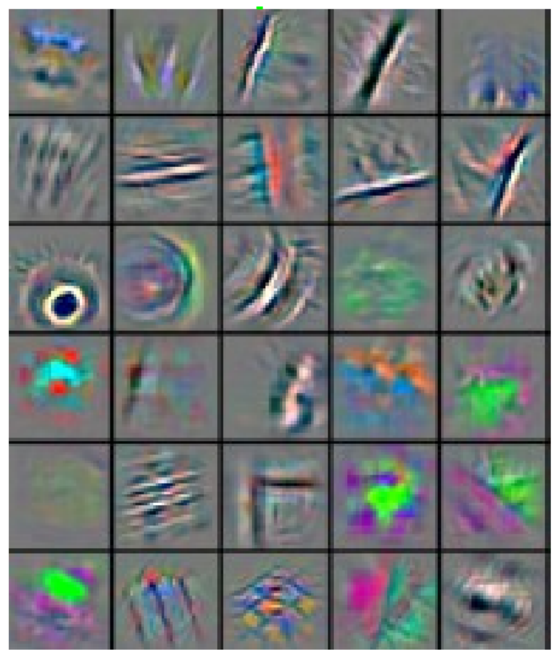
**Classifier** (head)

# Building blocks (2/2)

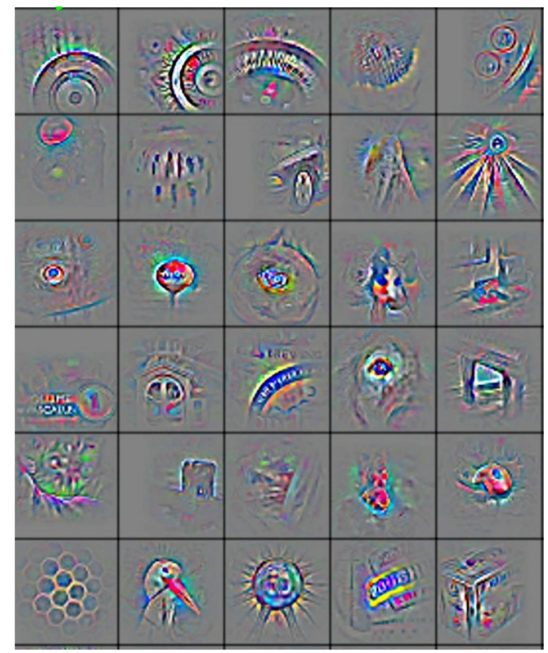
- ConvNets learn spatial **hierarchies of patterns**

**Low-level patterns**
*(conv. layers in the front)*

**Mid-level patterns**
*(conv. layers in the middle)*

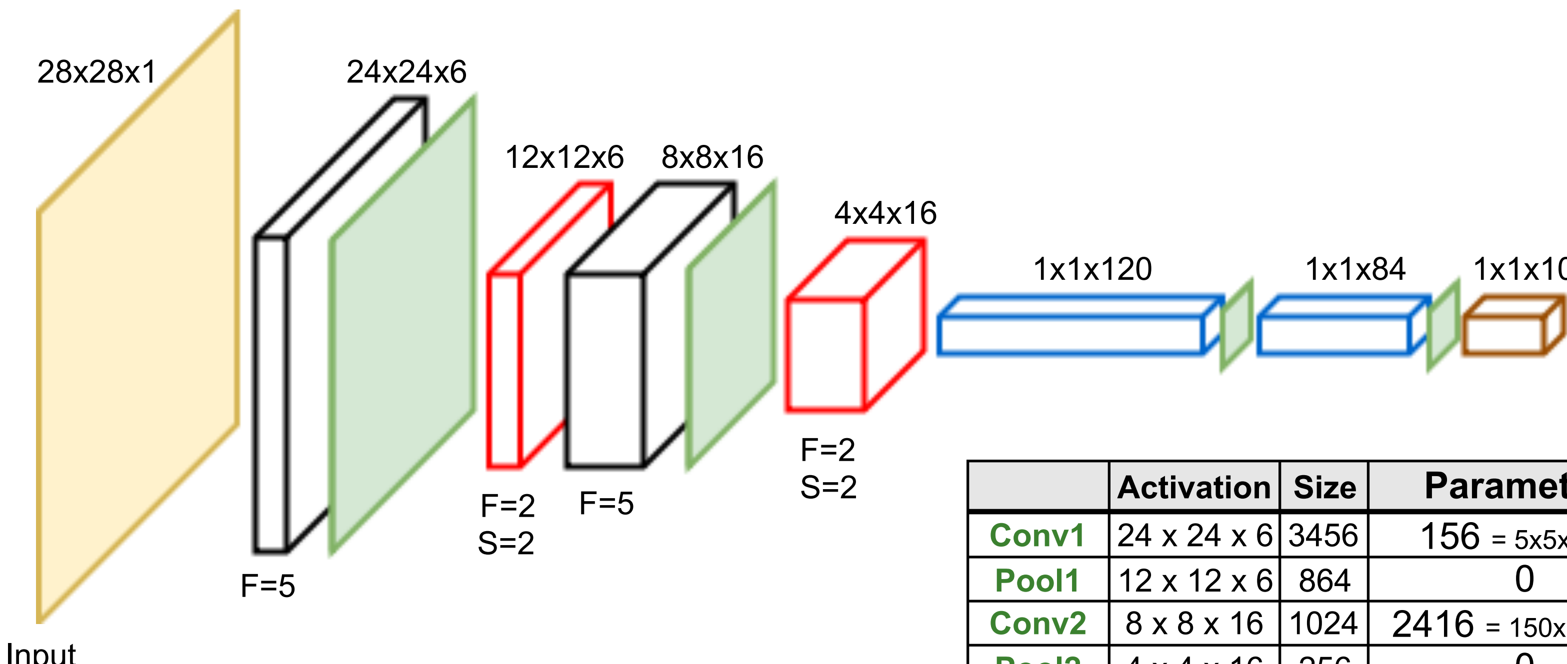


**High-level patterns**
*(conv. layers in the end)*



# Classic architectures (1/3)

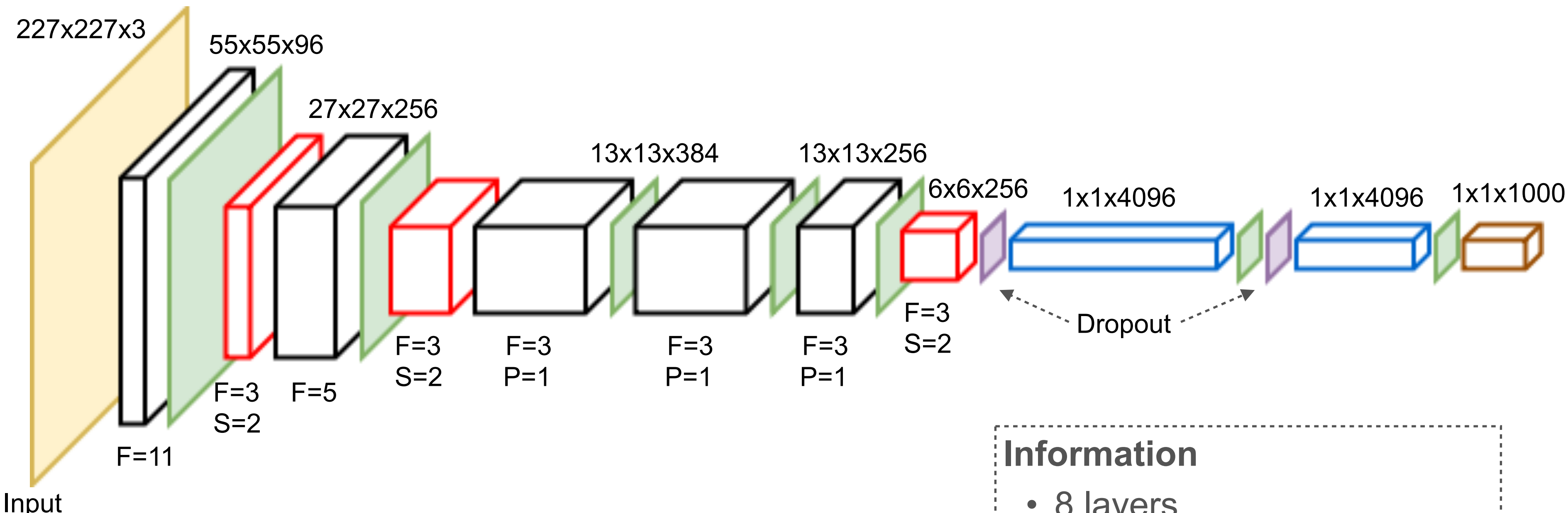
- **LeNet-5** (1998) ➜ *0.05 millions of parameters*




| | Activation | Size | Parameters |
|---|---|---|---|
| **Conv1** | 24 x 24 x 6 | 3456 | 156 = 5x5x6 + 6 |
| **Pool1** | 12 x 12 x 6 | 864 | 0 |
| **Conv2** | 8 x 8 x 16 | 1024 | 2416 = 150x16 + 16 |
| **Pool2** | 4 x 4 x 16 | 256 | 0 |
| **Flatten** | 1 x 1 x 256 | 256 | 0 |
| **FC1** | 1 x 1 x 120 | 120 | 30840 = (256+1)x120 |
| **FC2** | 1 x 1x 84 | 84 | 10164 = (120+1)x84 |
| **Softmax** | 1 x 1 x 10 | 10 | 850 = (84+1)x10 |

# Classic architectures (2/3)

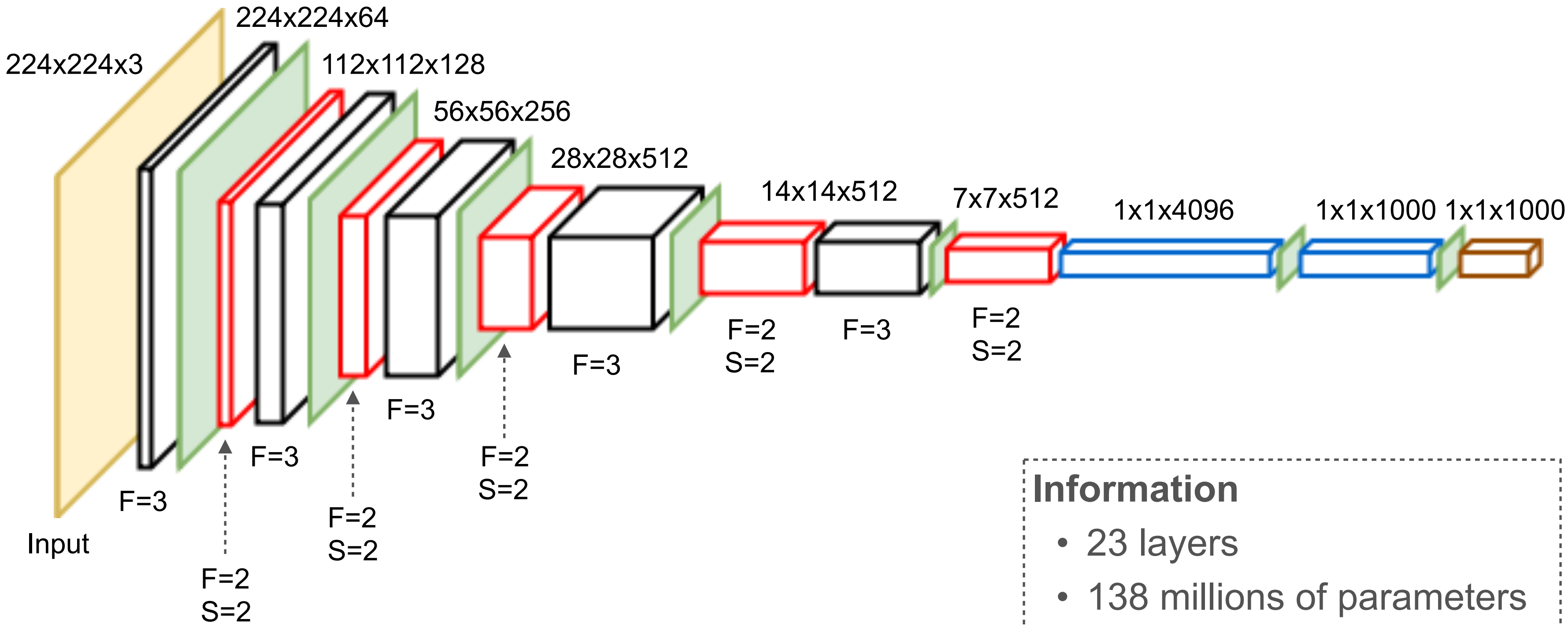
- **AlexNet** (2012)




**Information**
- 8 layers
- 60 millions of parameters
- 84% accuracy on ImageNet

# Classic architectures (3/3)

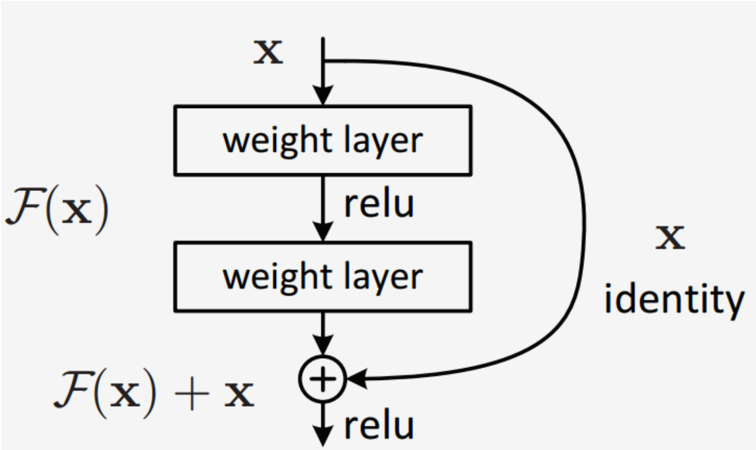
- **VGG-16** (2015)



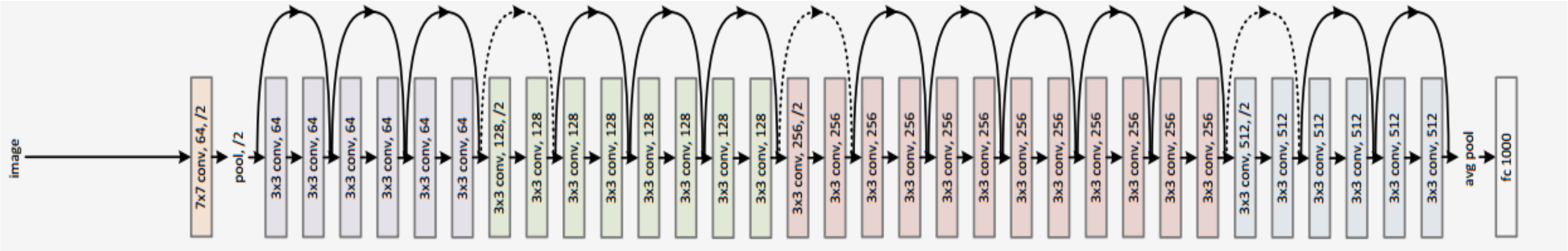

**Information**
- 23 layers
- 138 millions of parameters
- 90% accuracy on ImageNet

# Modern architectures (1/2)

- **Residual network** (2016)

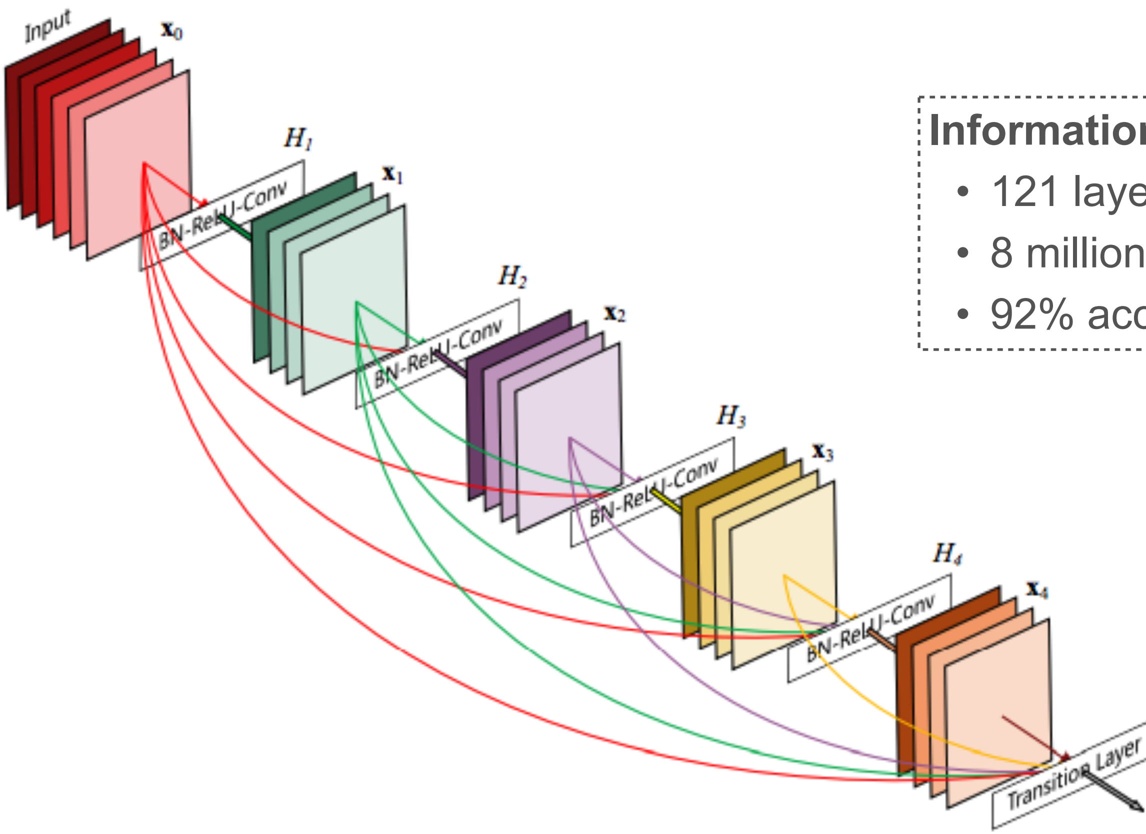



**Information**
- 168 layers
- 25 millions of parameters
- 93% accuracy on ImageNet



# Modern architectures (2/2)

- **Dense network** (2017)




**Information**
- 121 layers
- 8 millions of parameters
- 92% accuracy on ImageNet

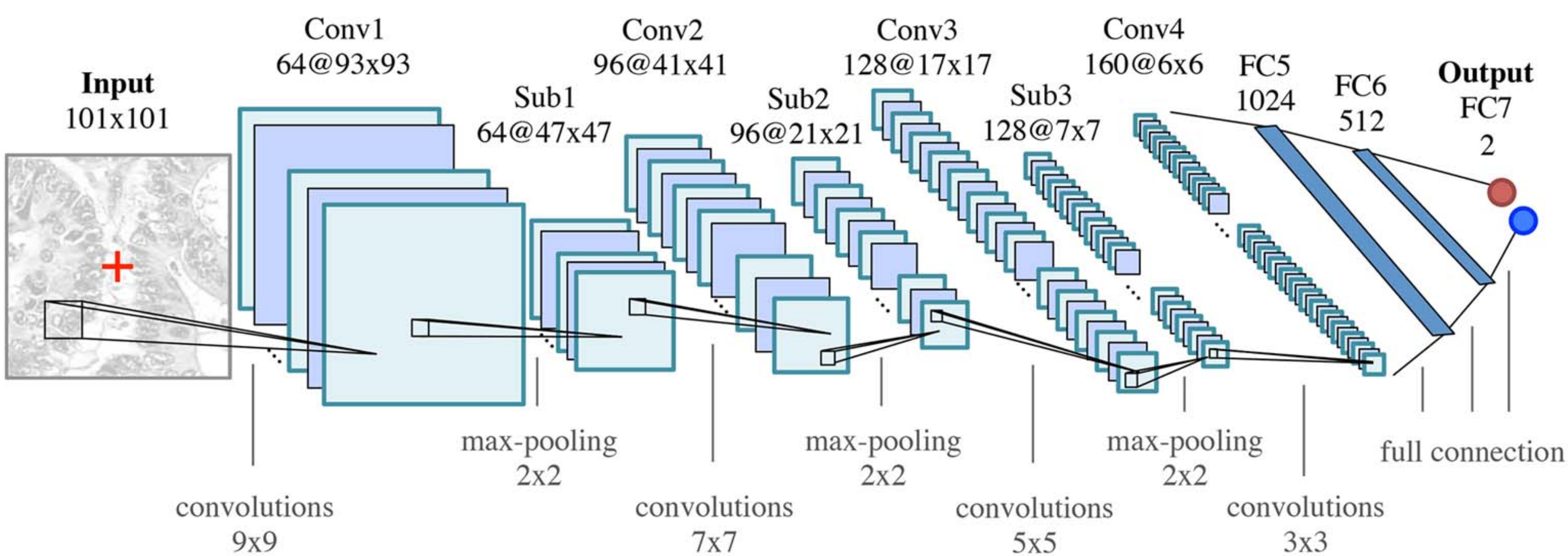
# Quiz

1) Which of the following do you typically see as you move to deeper layers in a ConvNet? *(**W**x**H**x**C** is the layers' output size)*
   A. ***W*** *and* ***H*** *decreases, while* ***C*** *also decreases*
   B. ***W*** *and* ***H*** *increases, while* ***C*** *decreases*
   C. ***W*** *and* ***H*** *decreases, while* ***C*** *increases*
   D. ***W*** *and* ***H*** *increases, while* ***C*** *also increases*

2) Which of the following do you typically see in a ConvNet?
   A. *Multiple CONV layers follows by a POOL layer*
   B. *Multiple POOL layers follows by a CONV layer*
   C. *FC layers in the* ***last*** *few layers*
   D. *FC layers in the* ***first*** *few layers*

# What we have seen so far…

- ConvNets make the assumption that the **inputs are images**
  - ***New layers*** ➜ *Convolution & Pooling*
  - ***Architecture*** ➜ *Feature extractor + Classifier*



# Practical advice

- **Use whatever works best on ImageNet**
  - If you're feeling a bit of a fatigue in thinking about the architectural decisions, you'll be pleased to know that in 90% or more of applications you should not have to worry about these. Instead of rolling your own architecture for a problem, you should look at whatever architecture currently works best on ImageNet, download a pretrained model and fine-tune it on your data. You should rarely ever have to train a CNN from scratch or design one from scratch.

- **Prefer a stack of small CONV layers to one large CONV layer**
  - Suppose that you stack three 3x3 CONV layers on top of each other (with non-linearities in between, of course). In this arrangement, each neuron on the first CONV layer has a 3x3 view of the input volume. A neuron on the second CONV layer has a 3x3 view of the first CONV layer, and hence by extension a 5x5 view of the input volume. Similarly, a neuron on the third CONV layer has a 3x3 view of the 2nd CONV layer, and hence a 7x7 view of the input volume. Suppose that instead of these three layers of 3x3 CONV, we only wanted to use a single CONV layer with 7x7 receptive fields. These neurons would have a receptive field size of the input volume that is identical in spatial extent (7x7). However, the neurons would be computing a linear function over the input, while the three stacks of CONV layers contain non-linearities that make their features more expressive.