# Convolutional neural networks

## Lecture 8

Changho Suh

January 24, 2024

# Conv layer and Pooling layer

# Outline

1. Will study further on "feature map".

2. Will study 2nd building block: **Pooling** layer

# Example: How filter works

Vertical filter

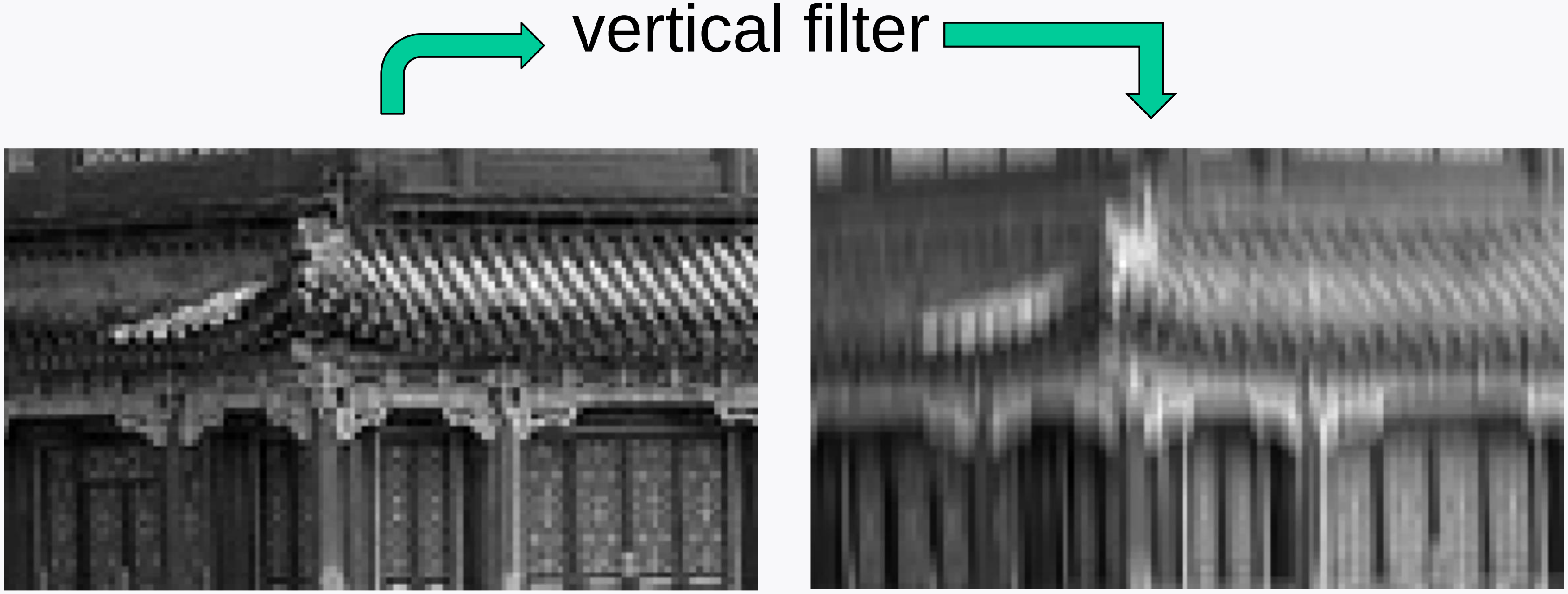
$f_w = 7$

$f_h = 7$

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

**Role:** Detect a *vertical line pattern*.

# Visualization



vertical filter

input

feature map

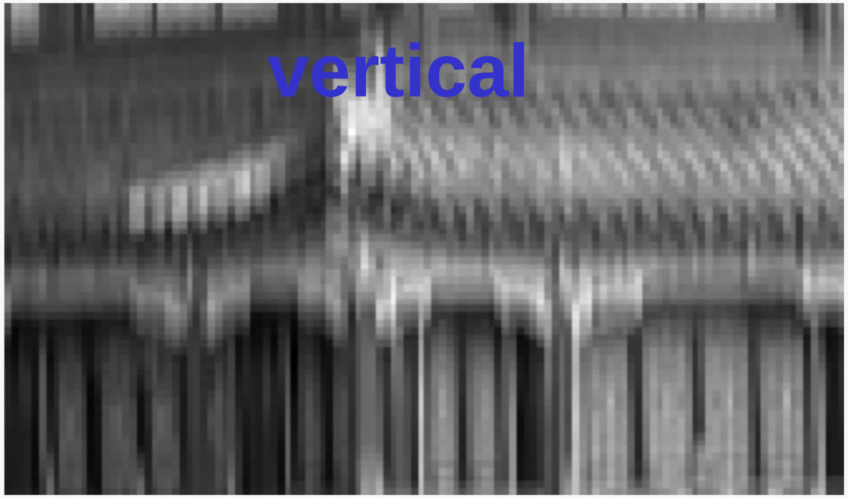
# Use of multiple filters

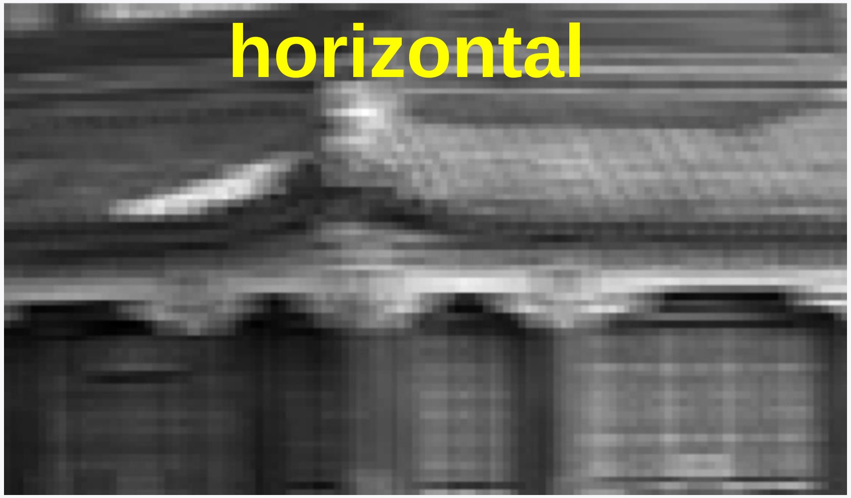
**Recall the role of a filter:** Detect a certain pattern.

**Note:** May wish to detect multiple patterns.

**Hence:** Employ multiple filters, e.g., vertical filter, horizontal filter, edge-detect filter, …

# Visualization examples

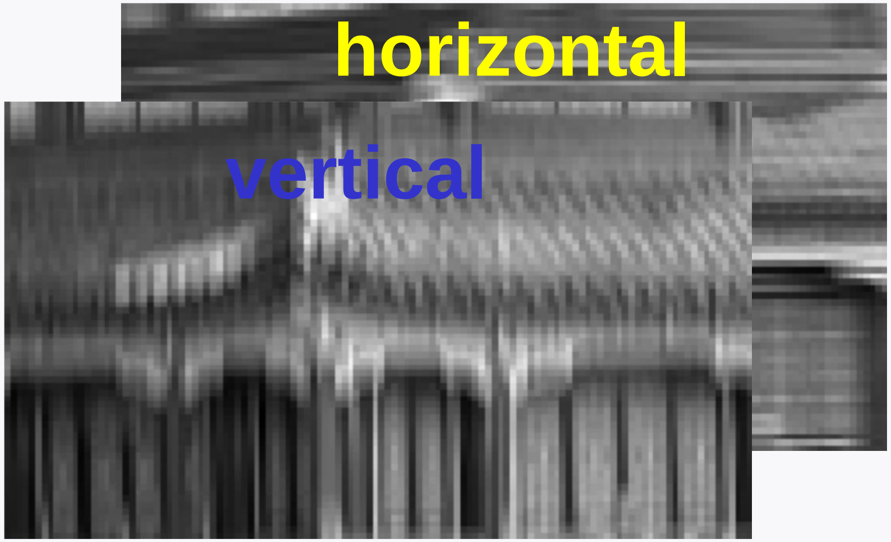




# Stacking multiple feature maps

filter

$w$ **3D** tensor



input $x$

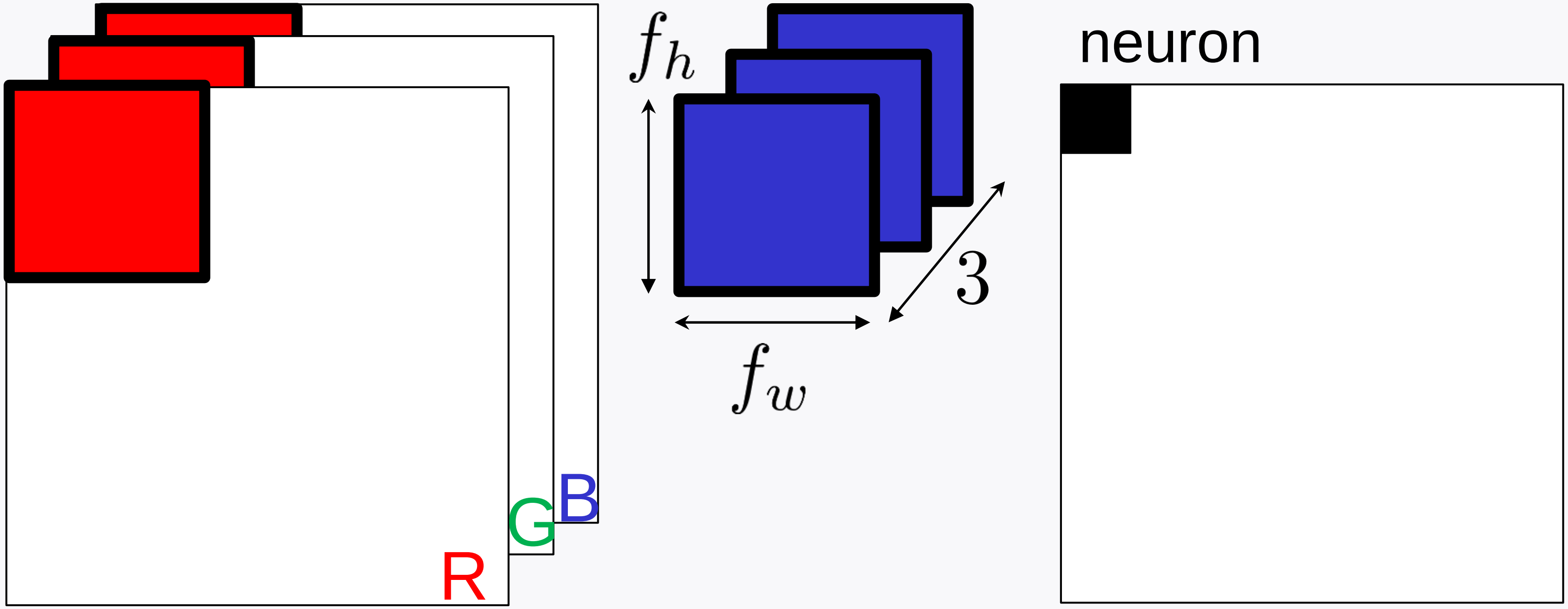
conv $z$

**3D** tensor

# Channels

$w$ **3D** tensor

filter

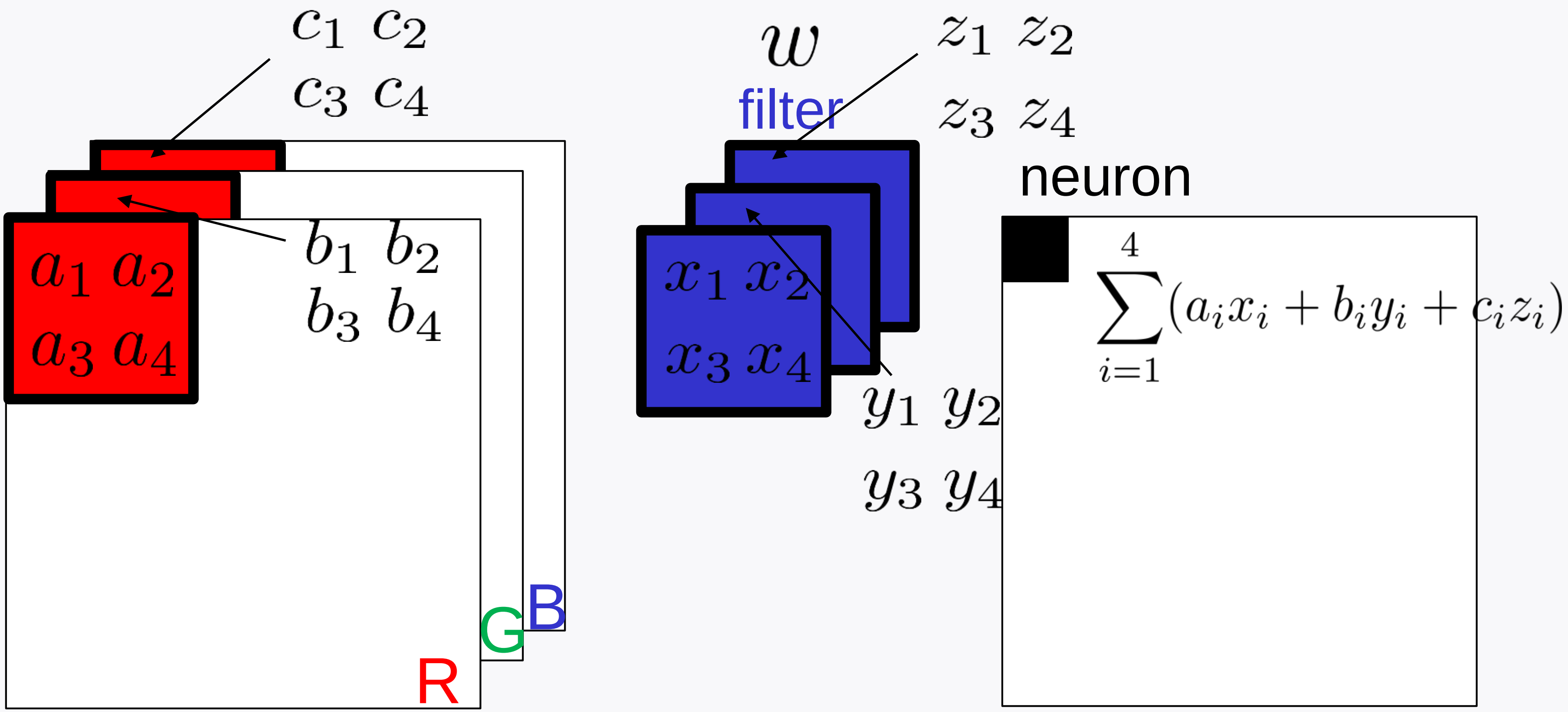


input $x$

**3D** tensor

conv $z$

# Computation of a neuron value

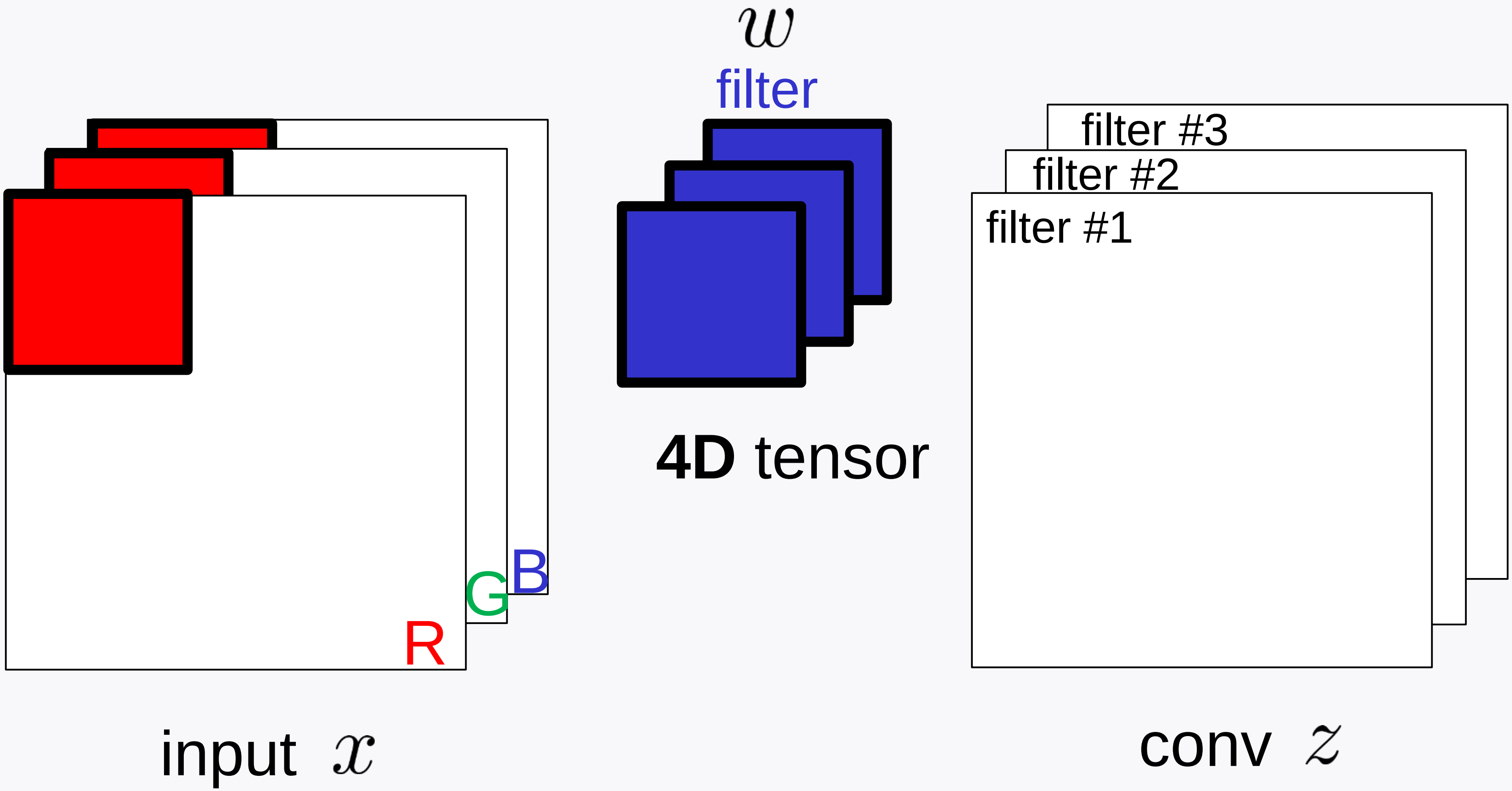


$$\sum_{i=1}^{4}(a_i x_i + b_i y_i + c_i z_i)$$

input $x$

conv $z$

# Dimension of weights?



**4D** tensor

# Pooling layer

# Serious memory issue

For training: Employ **backpropagation**.

**Backpropagation** requires:

*Caching everything* computed before.

Each conv layer requires ~ 10MB for one example.

Suppose a training batch contains **100 examples.**

→ Requires ~ 1GB per conv layer.

→ Requires much more with many layers.

# Motivates us to reduce complexity

This is where **Pooling** layer kicks in.

**Role** of **Pooling** layer:

*Downsample* to reduce # parameters and therefore memory size.

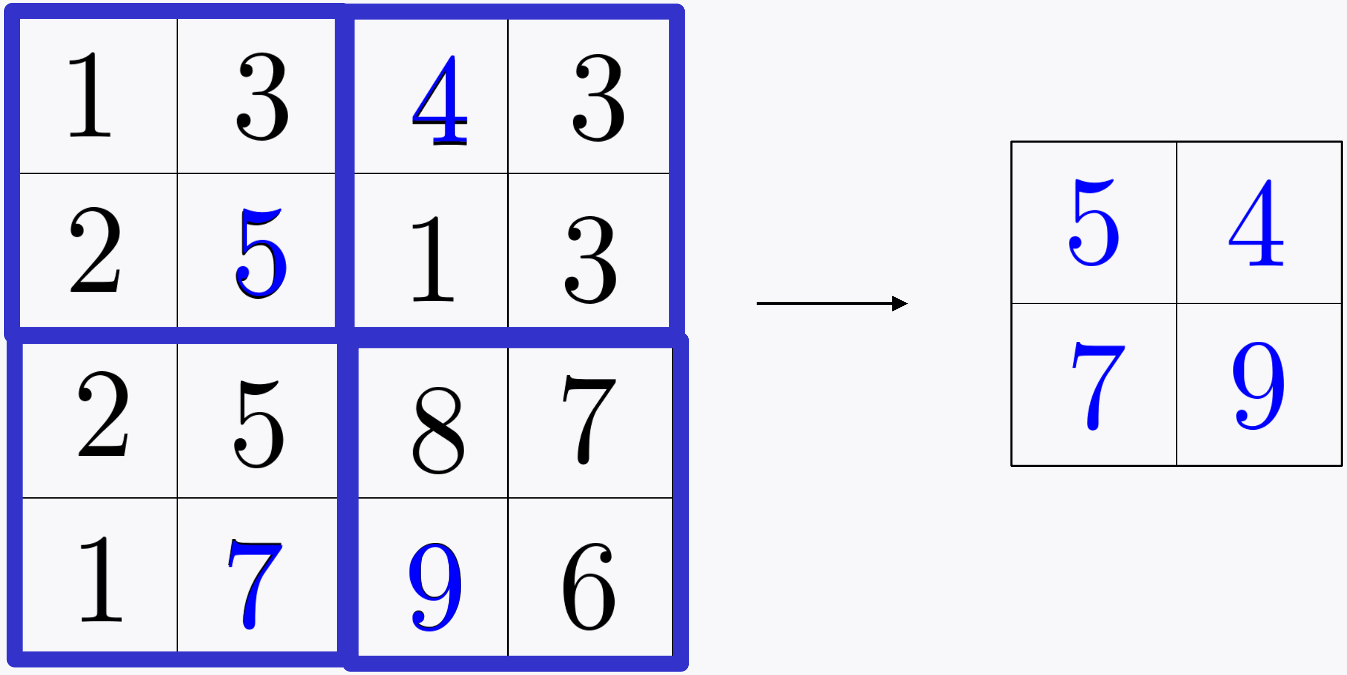
Two types of **Pooling:**

1. **Max Pooling** (most common)
2. **Average Pooling**

# Max Pooling

pooling filter (or kernel)

Suppose: stride=2



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

|   |   |
|---|---|
| 5 | 4 |
| 7 | 9 |

A feature map

max pooling

# Visualization example

2*2 pooling filter
stride=2

input

max pooling

**Note:** Works on each channel independently!

# Typical CNN architecture

Repeat the following stack module:

stack [Conv] + [ReLU] + [Pooling]

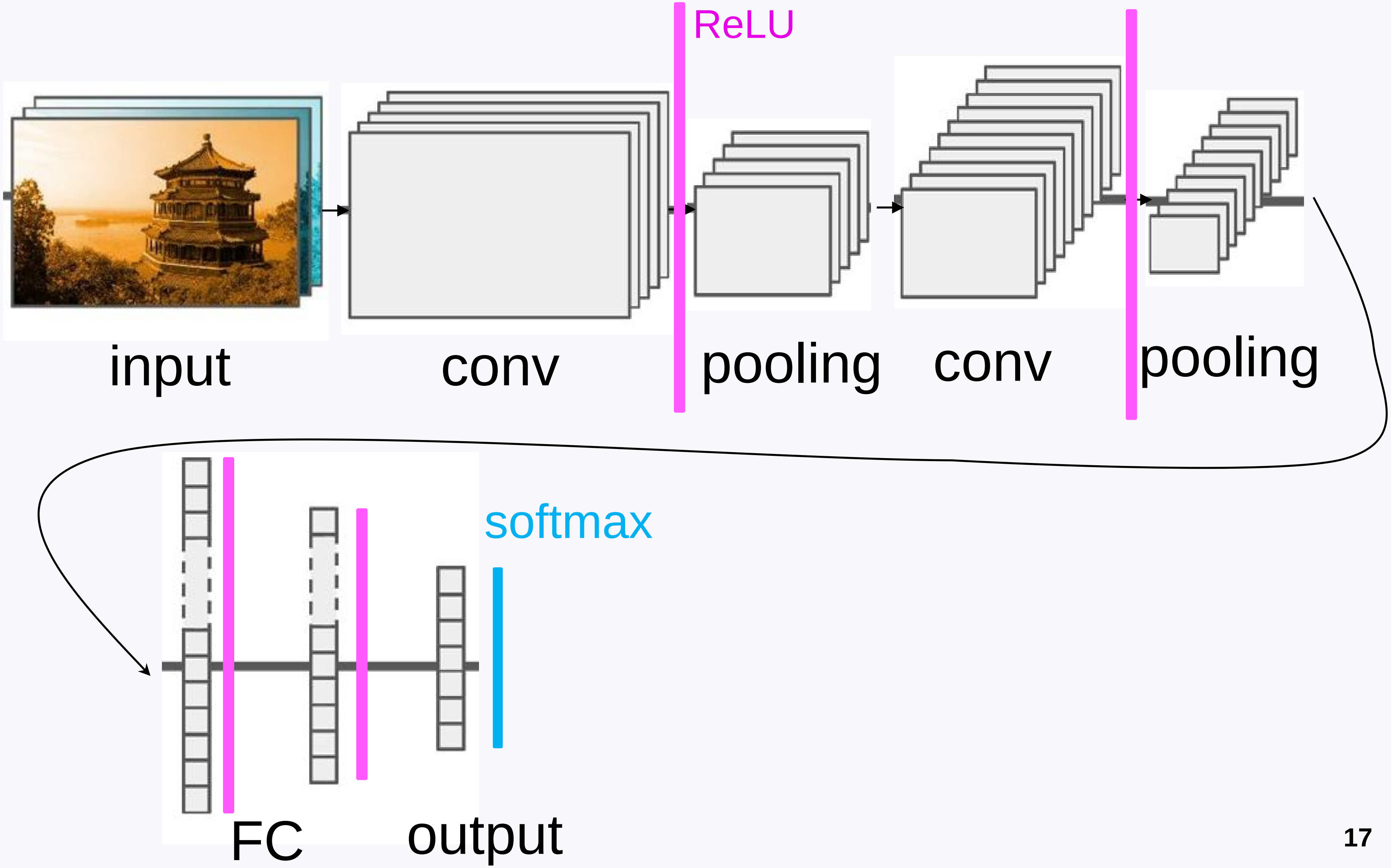
As the network gets deeper:

1. Feature map size gets smaller;
2. # of feature maps gets bigger.

At the end of the stacks:

Fully-connected (FC) layers
+ output layer (e.g., softmax)

# Typical CNN architecture in picture



ReLU

input

conv

pooling

conv

pooling

softmax

FC

output

# Two popular CNNs

1. **AlexNet** (2012)

Alex Krizhevsky

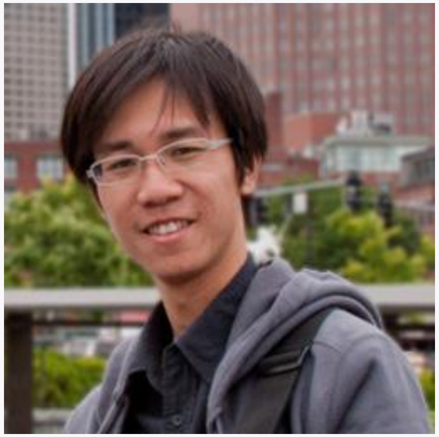
Ilya Sutskever

Geoffrey Hinton

Anchored the deep learning revolution.

2. **ResNet** (2015)

Currently the most powerful & arguably the simplest.



Kaiming He

# Look ahead

Will study details on AlexNet & ResNet.