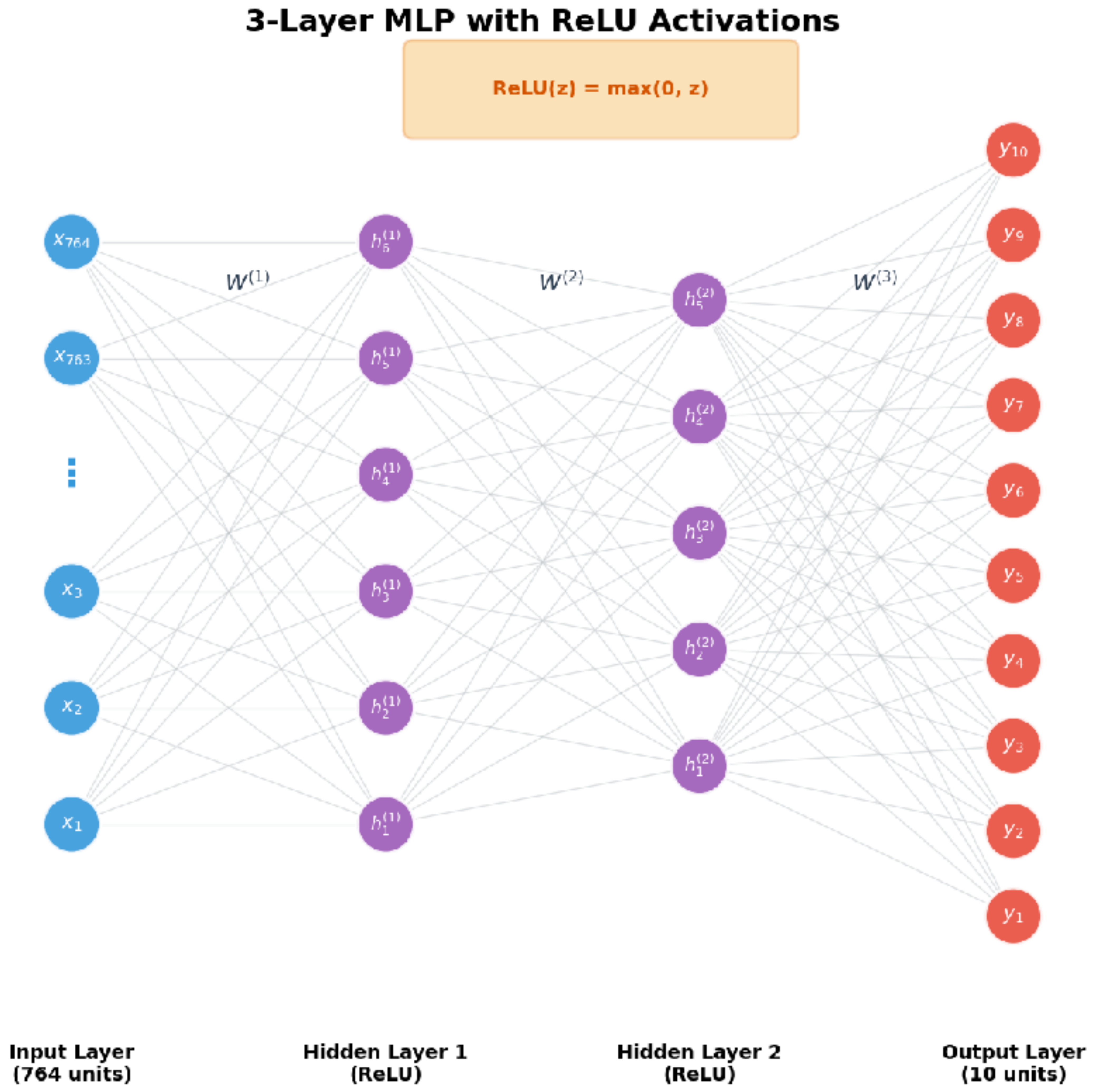
# Deep Learning (1470)

## Randall Balestriero

Class 7: Convolutions

# Multilayer Perceptrons

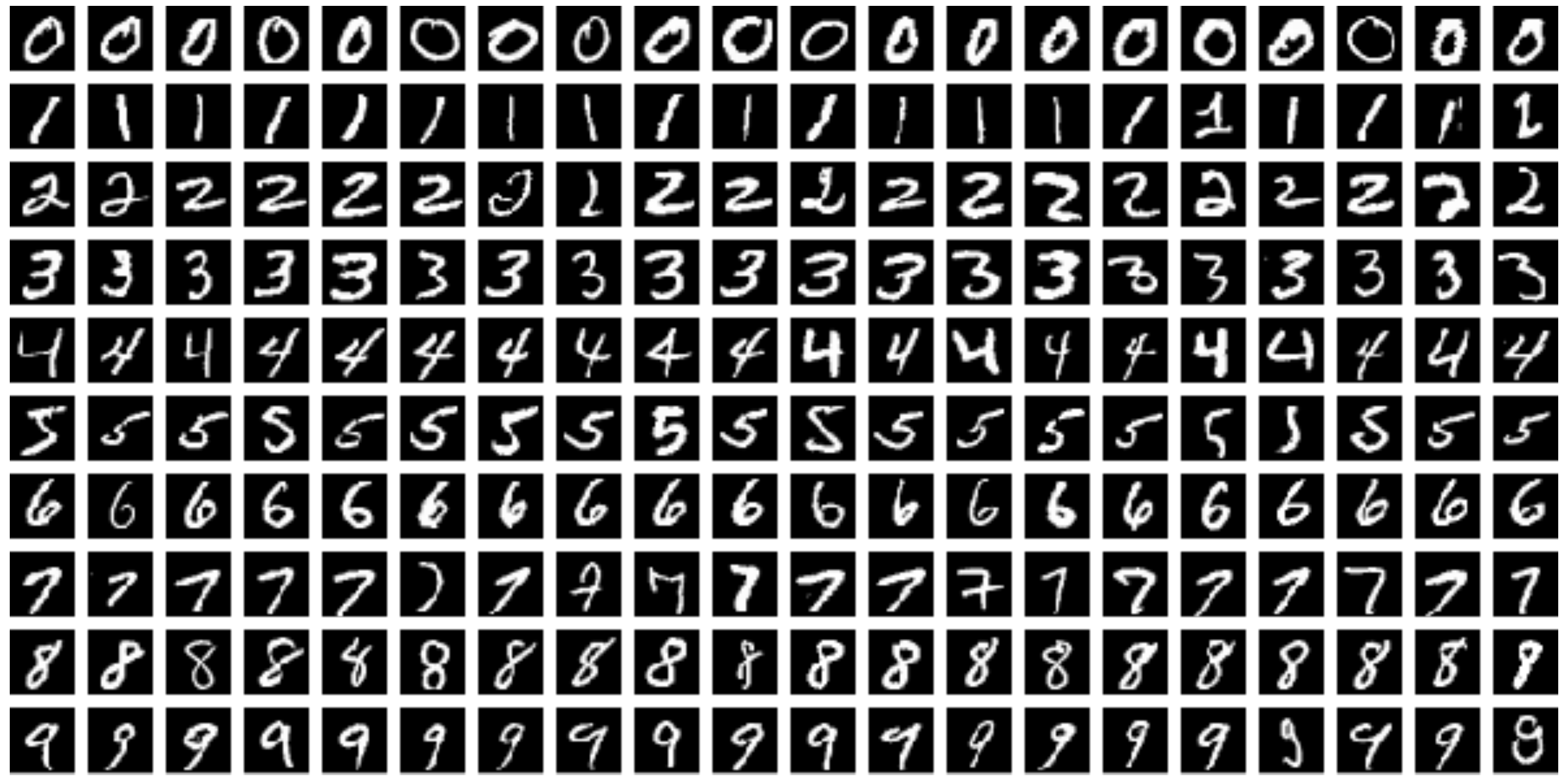
## And why we need better

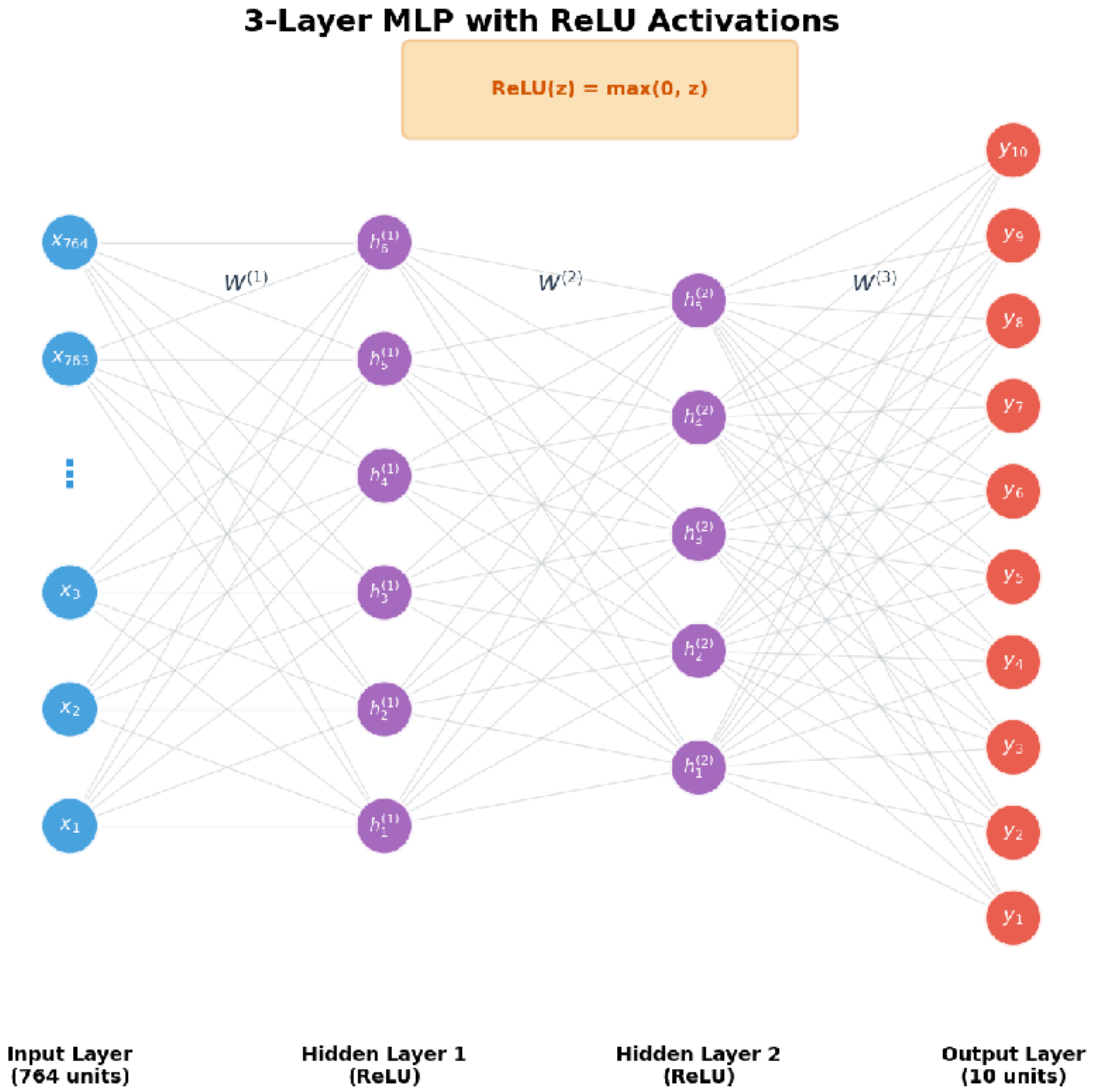


# Multilayer Perceptrons

## And why we need better

Extracting local features
at each layer is enough!

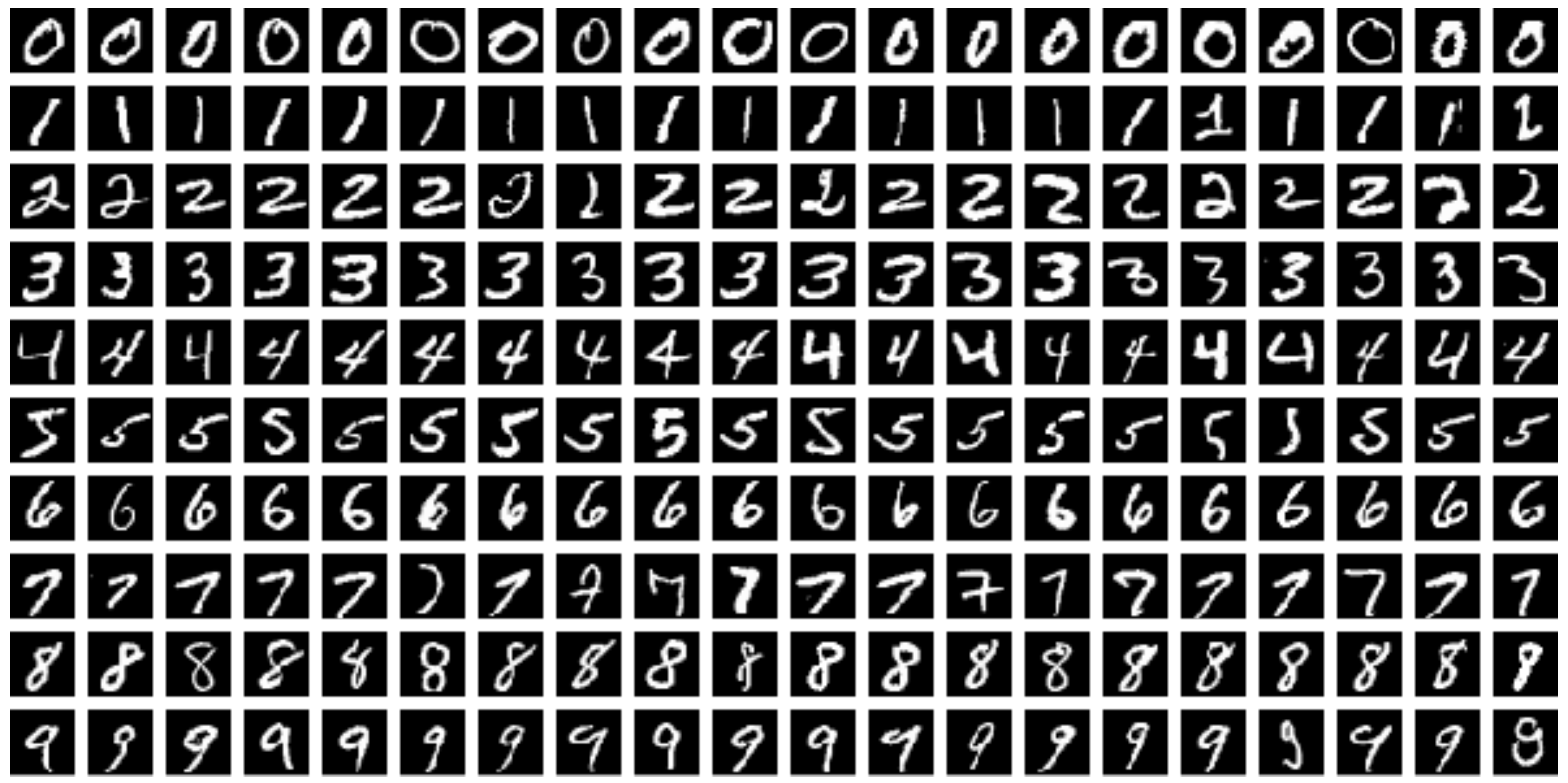




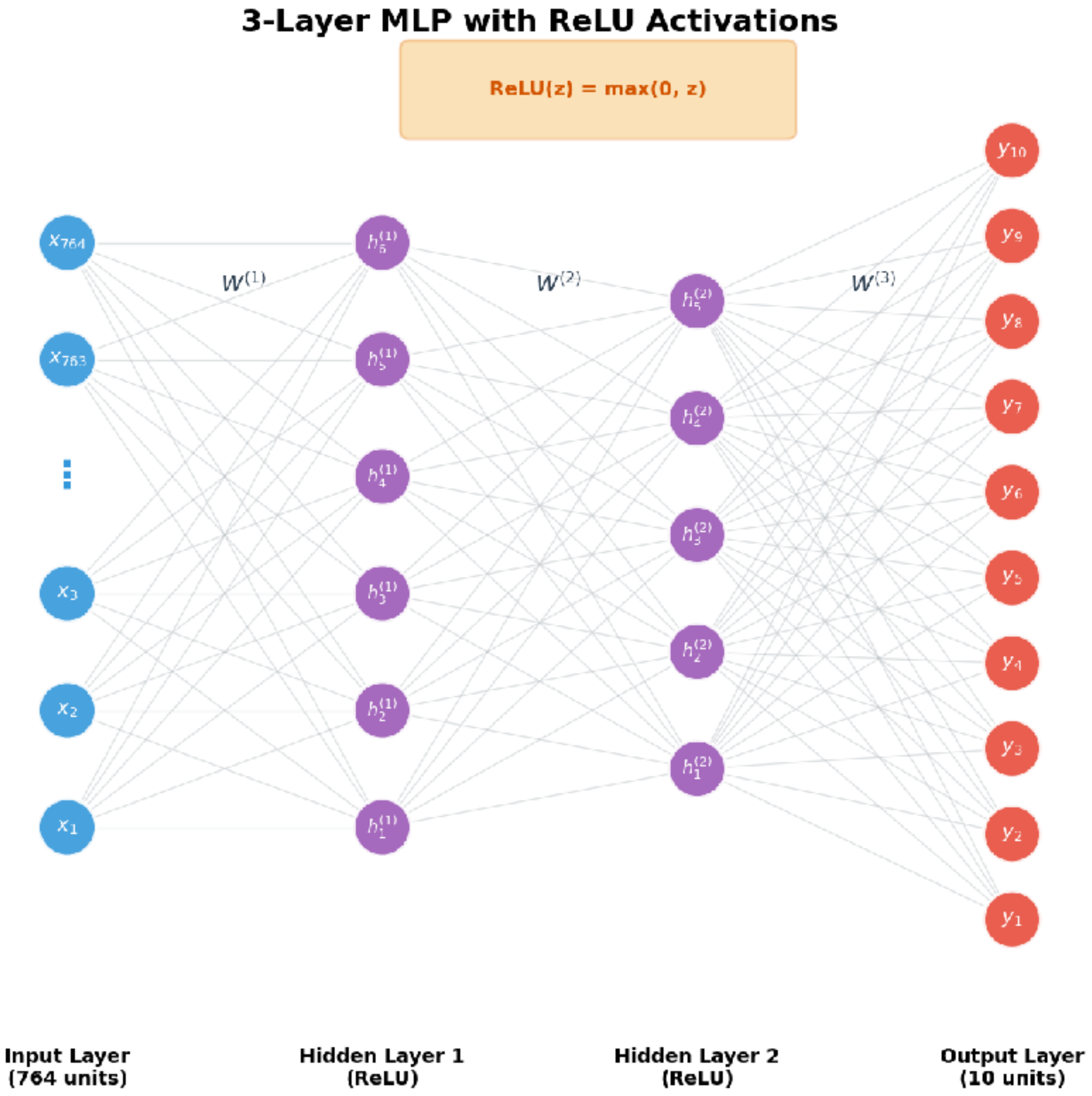
# Multilayer Perceptrons

## And why we need better

Extracting local features
at each layer is enough!



Not only local features… but
translation invariant features!



# The Main Building Block: Convolution

- A convolution is a linear operator
- Convolution is an operation that takes two inputs

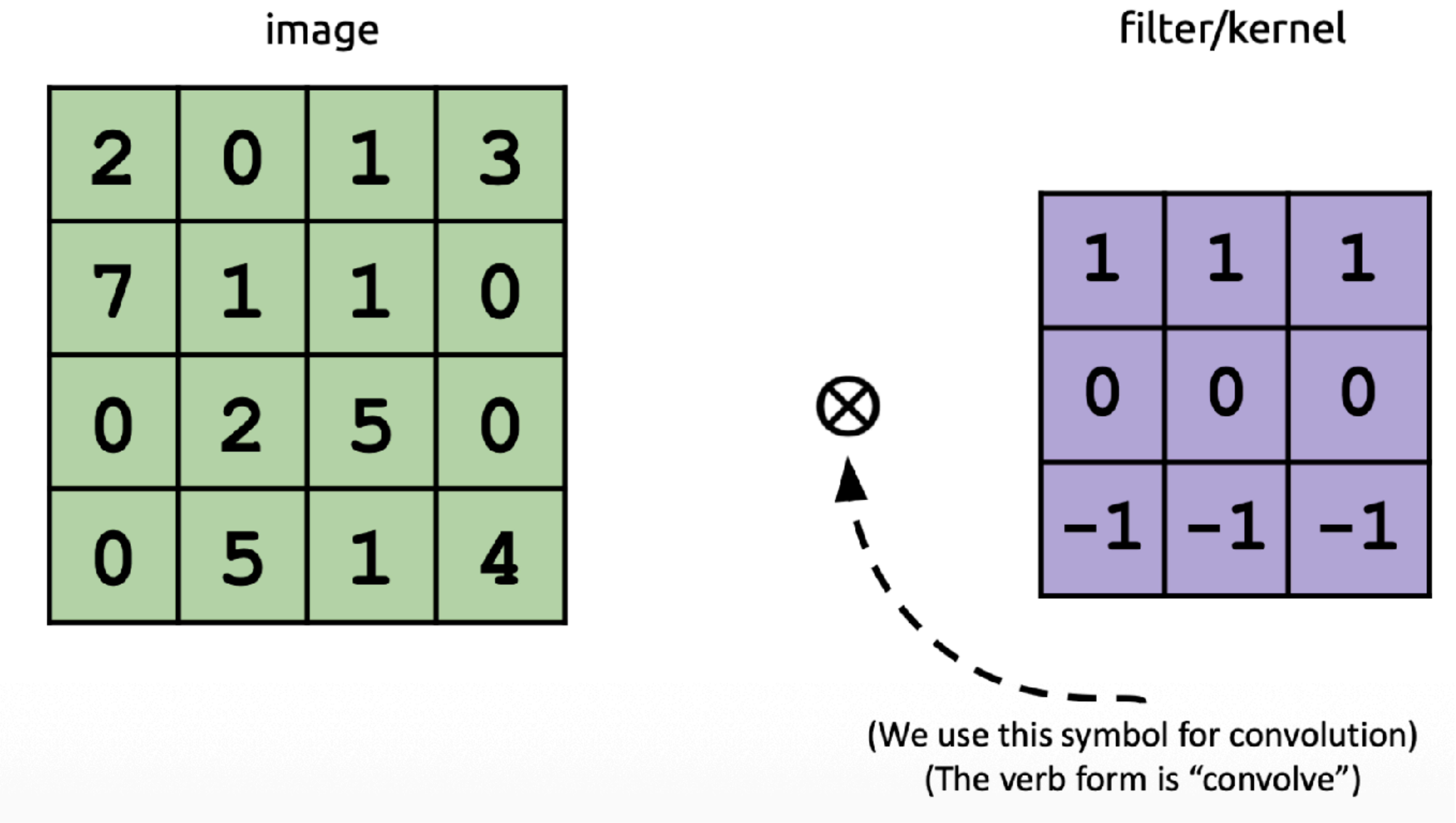
(1) An image (2D – B/W)

(2) A filter (also called a kernel)

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

2D array of numbers; could be any values

# What Convolution Does (Visually)



image

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

$\otimes$

filter/kernel

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

(We use this symbol for convolution)
(The verb form is “convolve”)

# What Convolution Does (Visually)

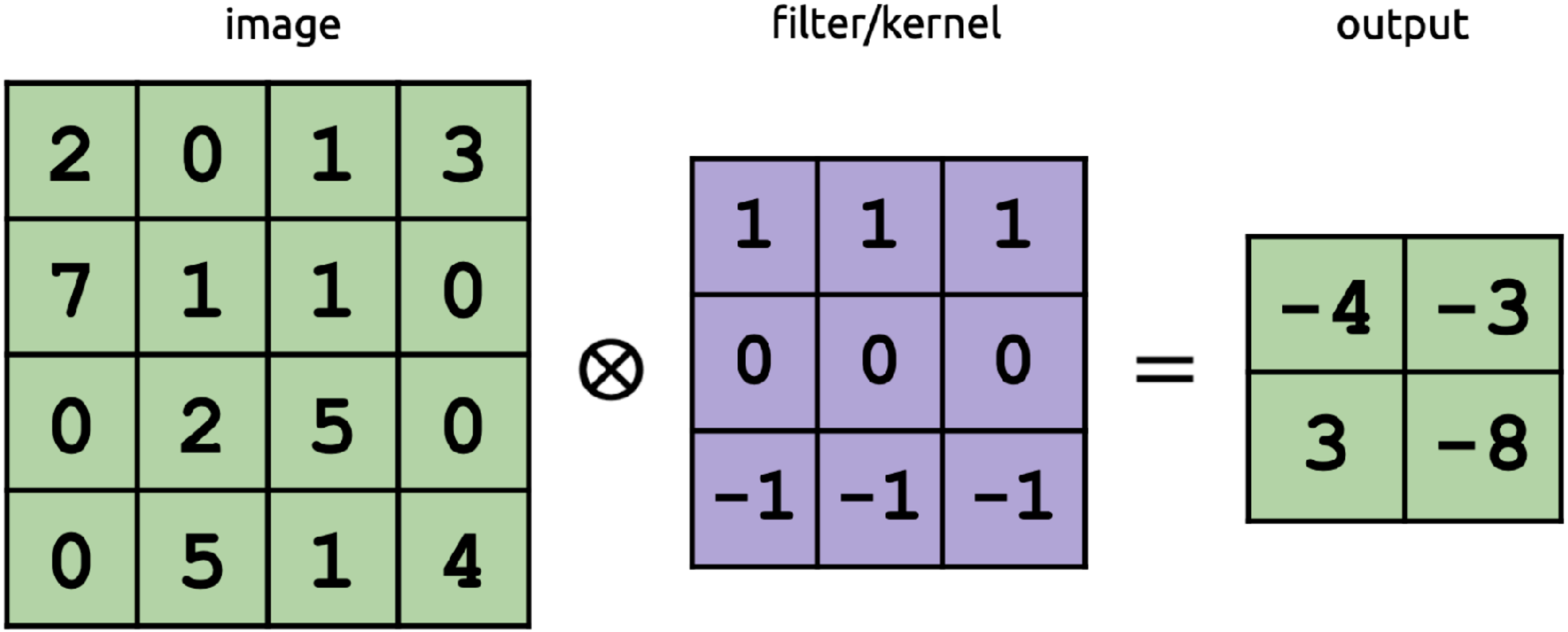
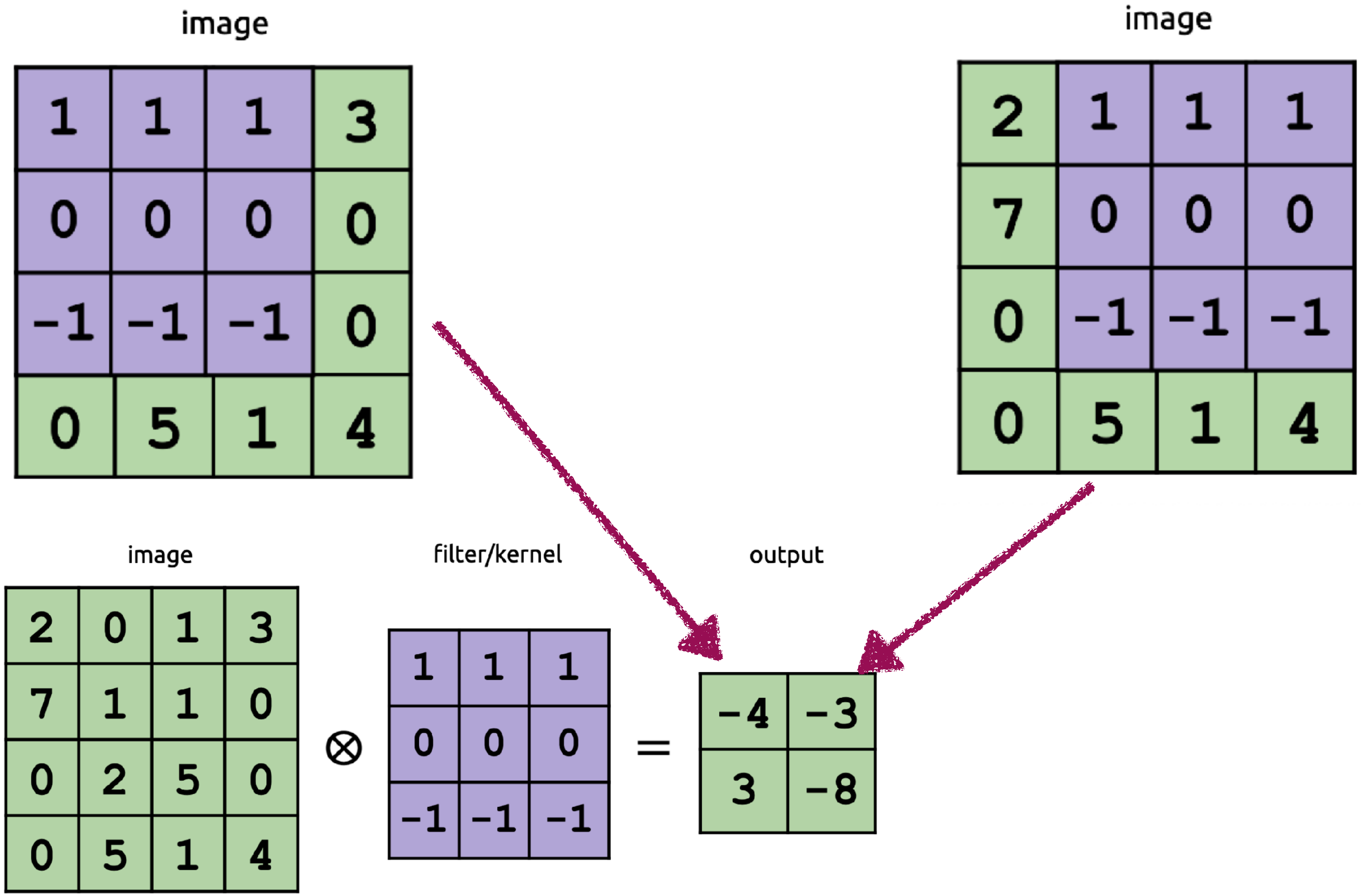


image filter/kernel output

$$\begin{bmatrix} 2 & 0 & 1 & 3 \\ 7 & 1 & 1 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 5 & 1 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -4 & -3 \\ 3 & -8 \end{bmatrix}$$

# What Convolution Does (Visually)



image

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 3 |
| 0 | 0 | 0 | 0 |
| -1 | -1 | -1 | 0 |
| 0 | 5 | 1 | 4 |

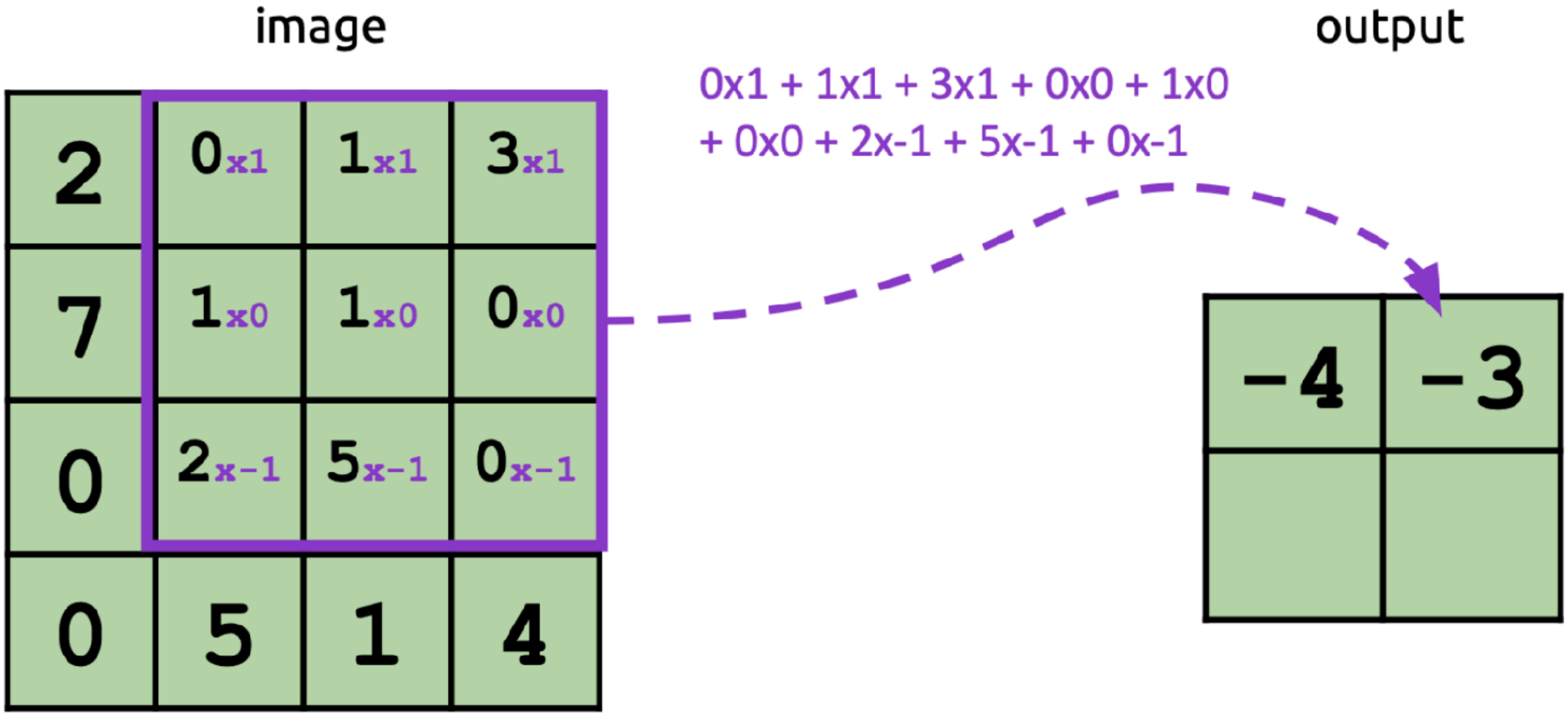
image

| | | | |
|---|---|---|---|
| 2 | 1 | 1 | 1 |
| 7 | 0 | 0 | 0 |
| 0 | -1 | -1 | -1 |
| 0 | 5 | 1 | 4 |

image, filter/kernel, output

$$\begin{bmatrix} 2 & 0 & 1 & 3 \\ 7 & 1 & 1 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 5 & 1 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -4 & -3 \\ 3 & -8 \end{bmatrix}$$

# What Convolution Does (Visually)

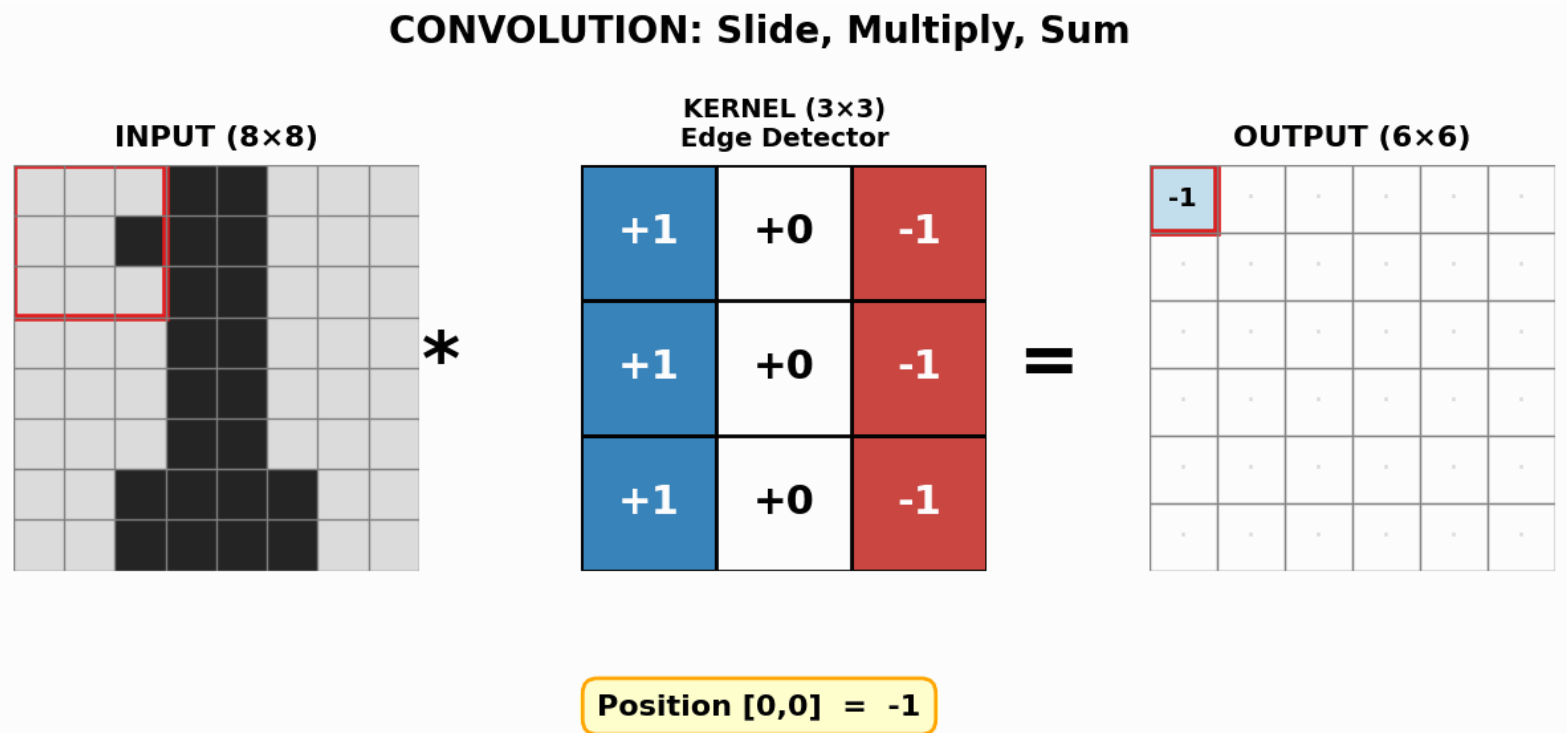


|  |  |  |  |
|---|---|---|---|
| 2 | $0_{x1}$ | $1_{x1}$ | $3_{x1}$ |
| 7 | $1_{x0}$ | $1_{x0}$ | $0_{x0}$ |
| 0 | $2_{x-1}$ | $5_{x-1}$ | $0_{x-1}$ |
| 0 | 5 | 1 | 4 |

0x1 + 1x1 + 3x1 + 0x0 + 1x0
+ 0x0 + 2x-1 + 5x-1 + 0x-1

|  |  |
|---|---|
| -4 | -3 |
|  |  |

# Example

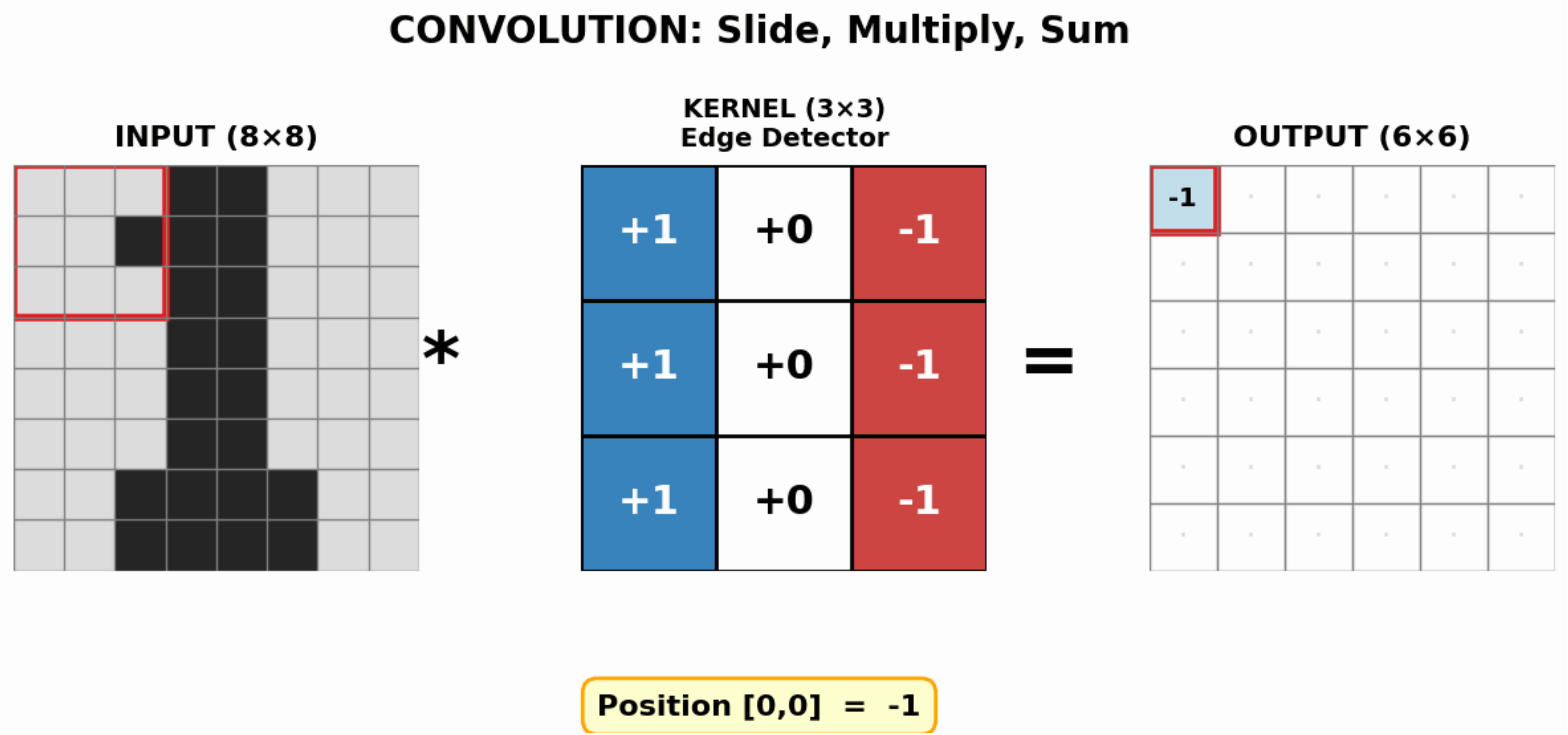


CONVOLUTION: Slide, Multiply, Sum

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

Position [0,0] = -1

# Example



CONVOLUTION: Slide, Multiply, Sum

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

Position [0,0] = -1

# This was all a lie!

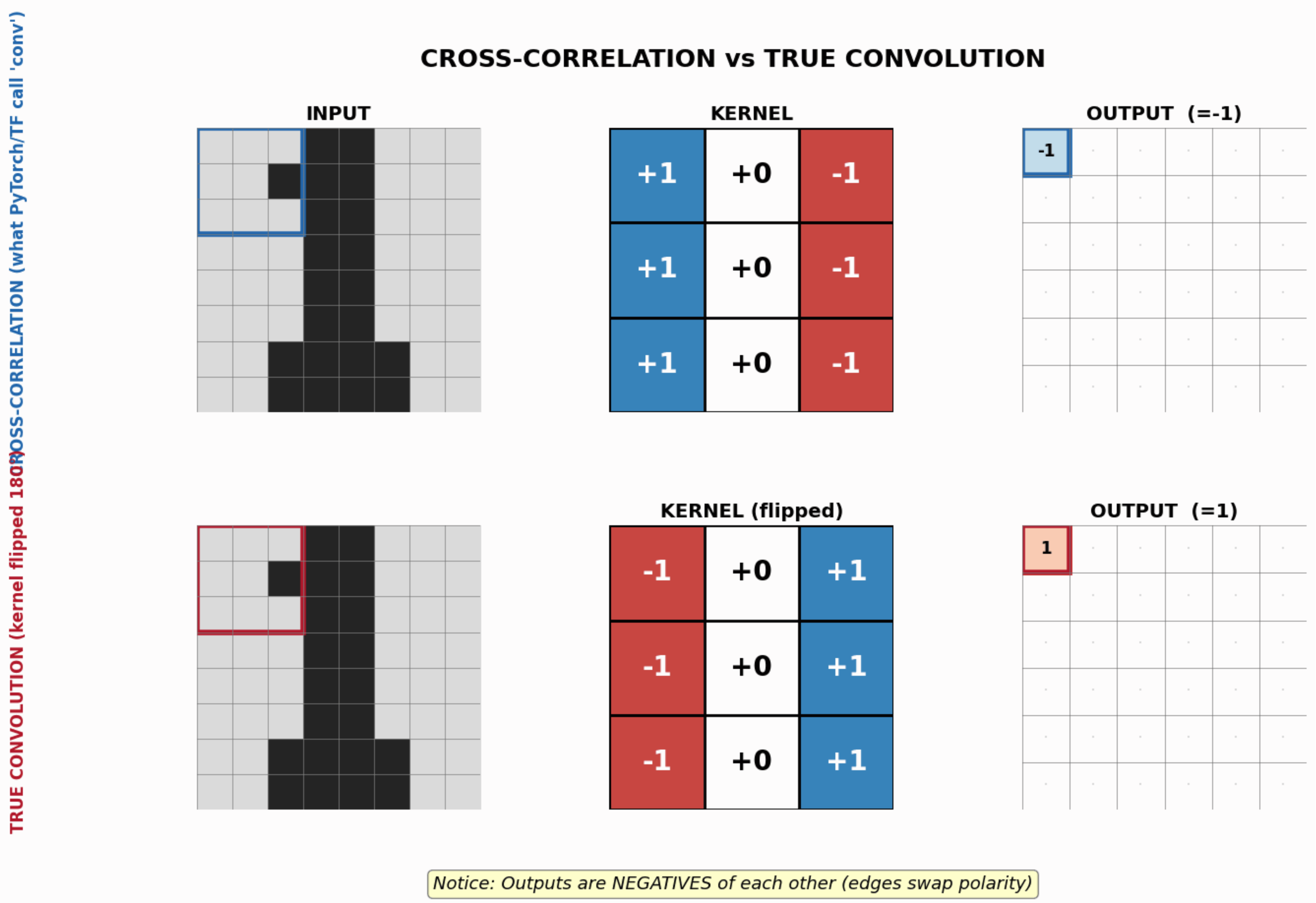
- All those example are actually ``cross-correlation”, not ``convolution”
- It is what is implemented in all those frameworks and called ``convolution”
- True convolution is almost that, but you need to flip the filter before applying

# This was all a lie!

- All those example are actually ``cross-correlation”, not ``convolution”
- It is what is implemented in all those frameworks and called ``convolution”
- True convolution is almost that, but you need to flip the filter before applying

Exercise: think why when starting from random init, it doesn’t matter!

# Example


CROSS-CORRELATION vs TRUE CONVOLUTION
CROSS-CORRELATION (what PyTorch/TF call 'conv')
TRUE CONVOLUTION (kernel flipped 180°)
INPUT
KERNEL
OUTPUT (=-1)
-1
KERNEL (flipped)
OUTPUT (=1)
1

KERNEL

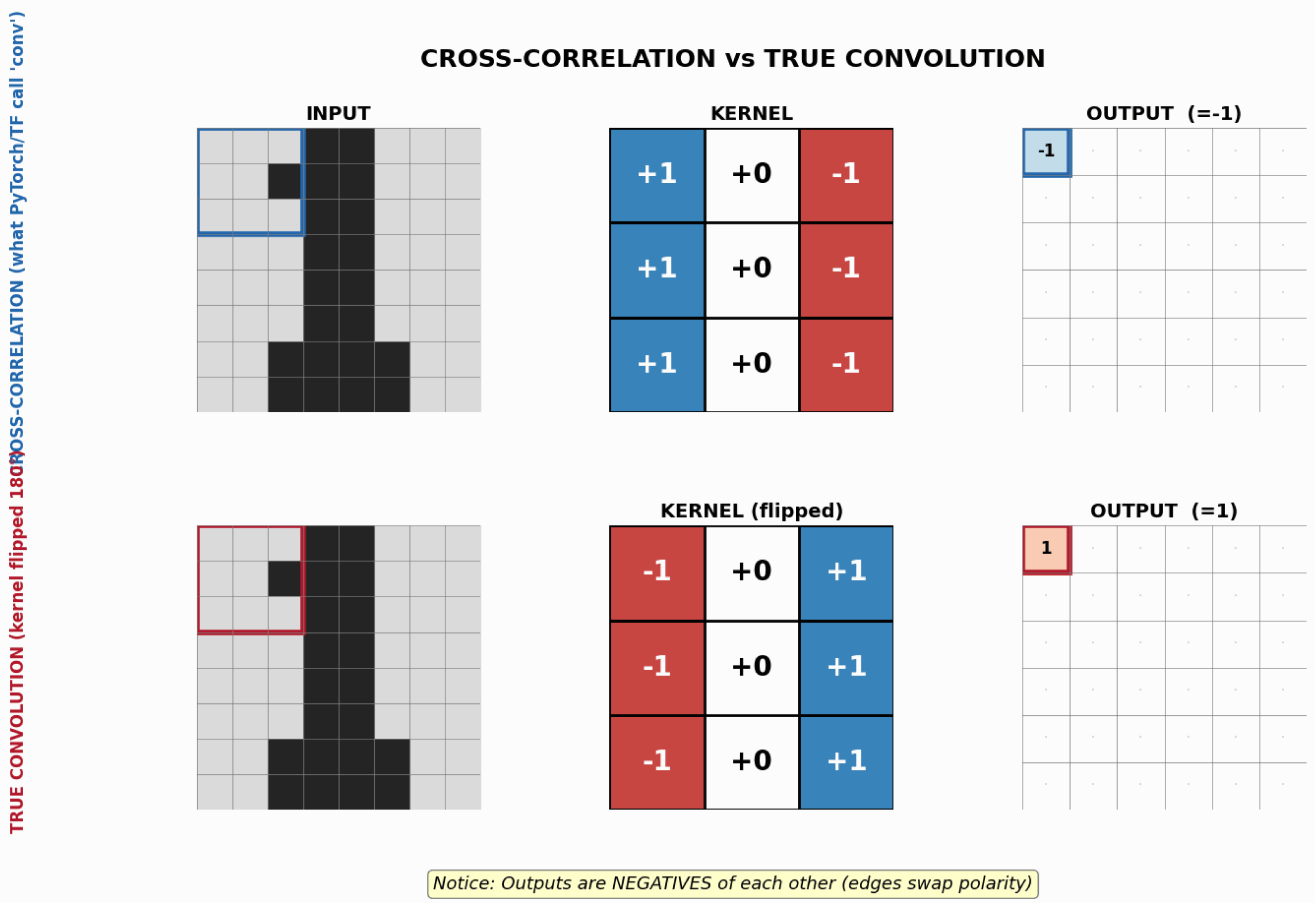
| | | |
|---|---|---|
| +1 | +0 | -1 |
| +1 | +0 | -1 |
| +1 | +0 | -1 |

KERNEL (flipped)

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

*Notice: Outputs are NEGATIVES of each other (edges swap polarity)*

# Example



CROSS-CORRELATION vs TRUE CONVOLUTION

CROSS-CORRELATION (what PyTorch/TF call 'conv')

INPUT

KERNEL

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

OUTPUT (=-1)

TRUE CONVOLUTION (kernel flipped 180°)

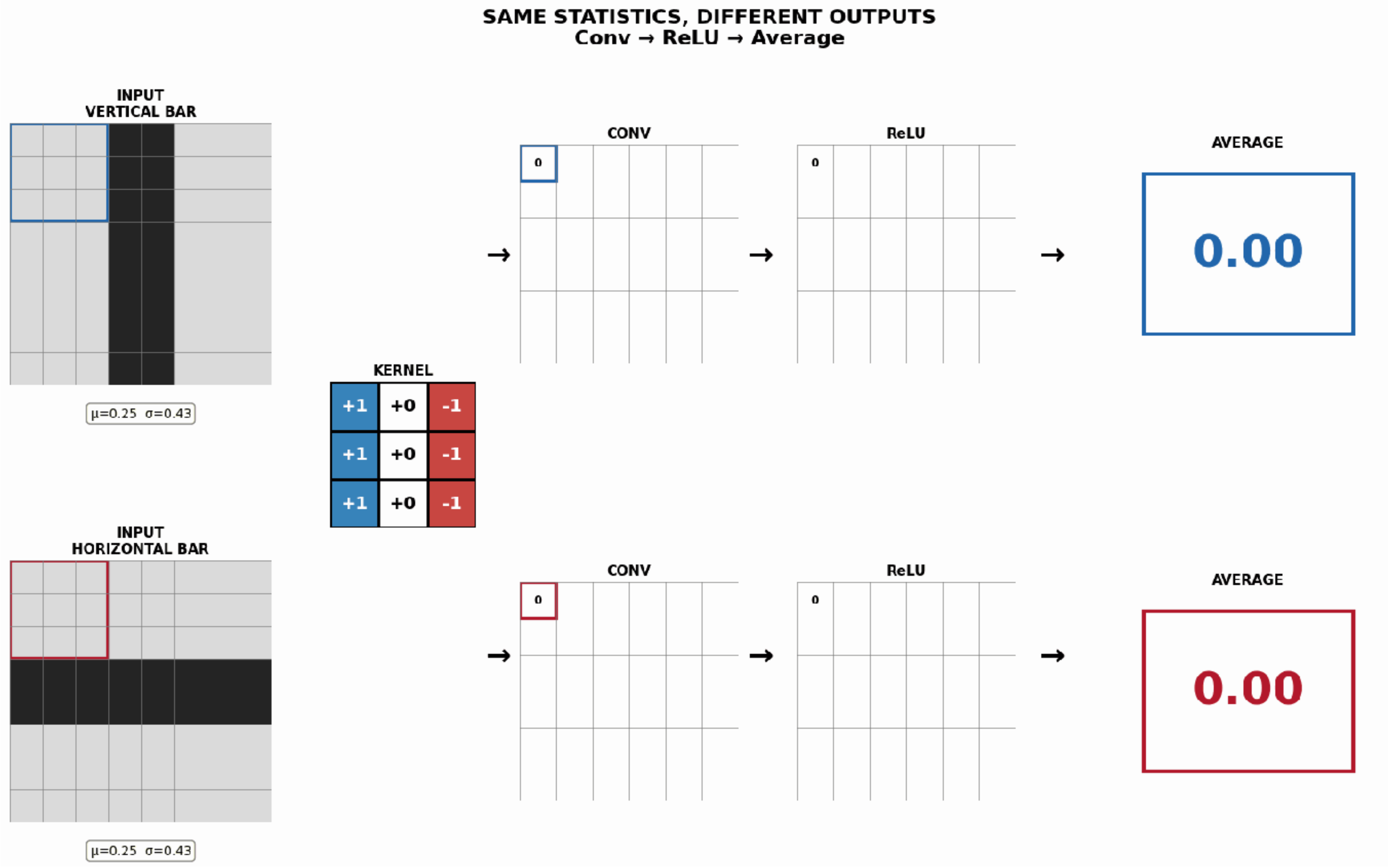
KERNEL (flipped)

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

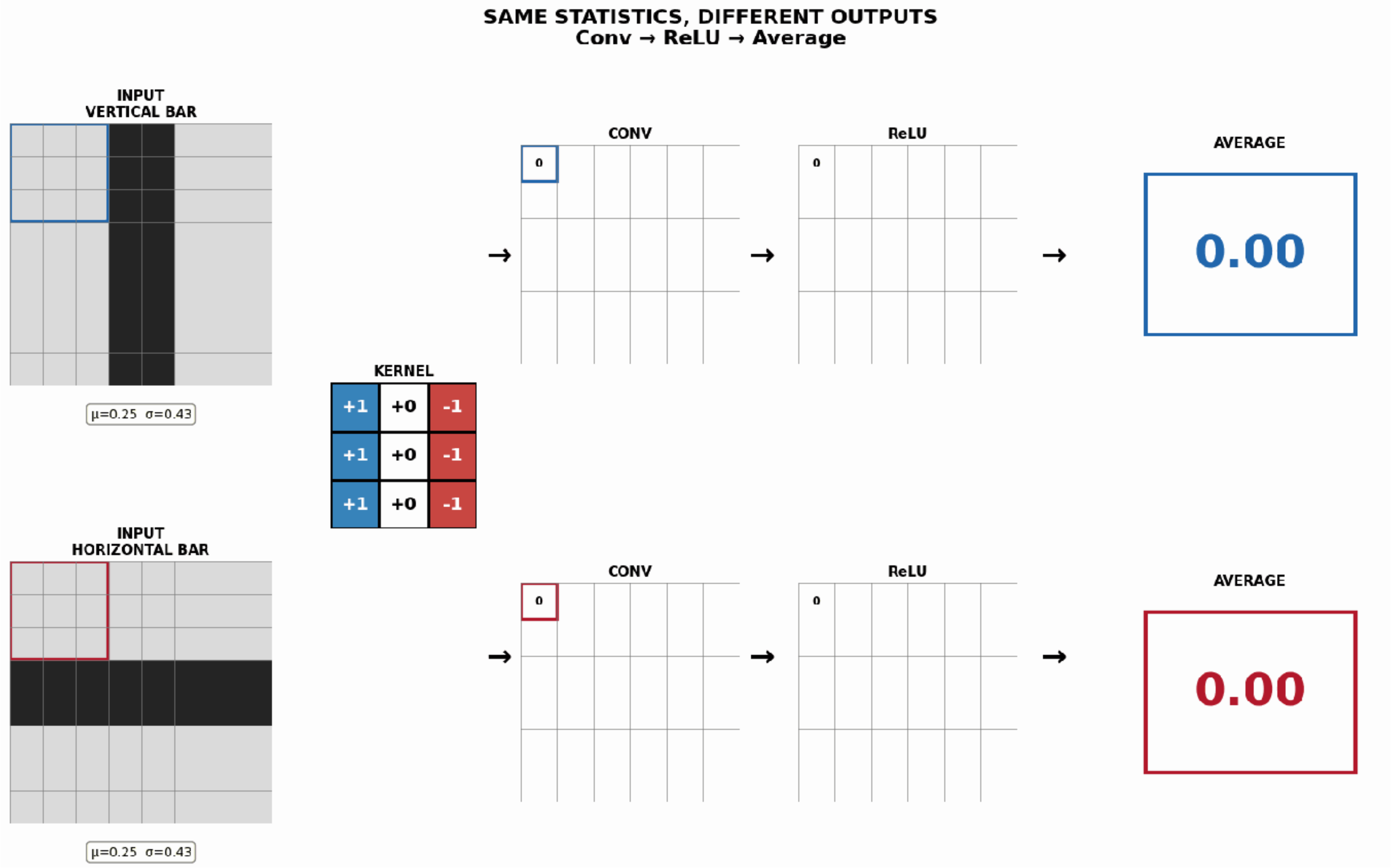
OUTPUT (=1)

*Notice: Outputs are NEGATIVES of each other (edges swap polarity)*

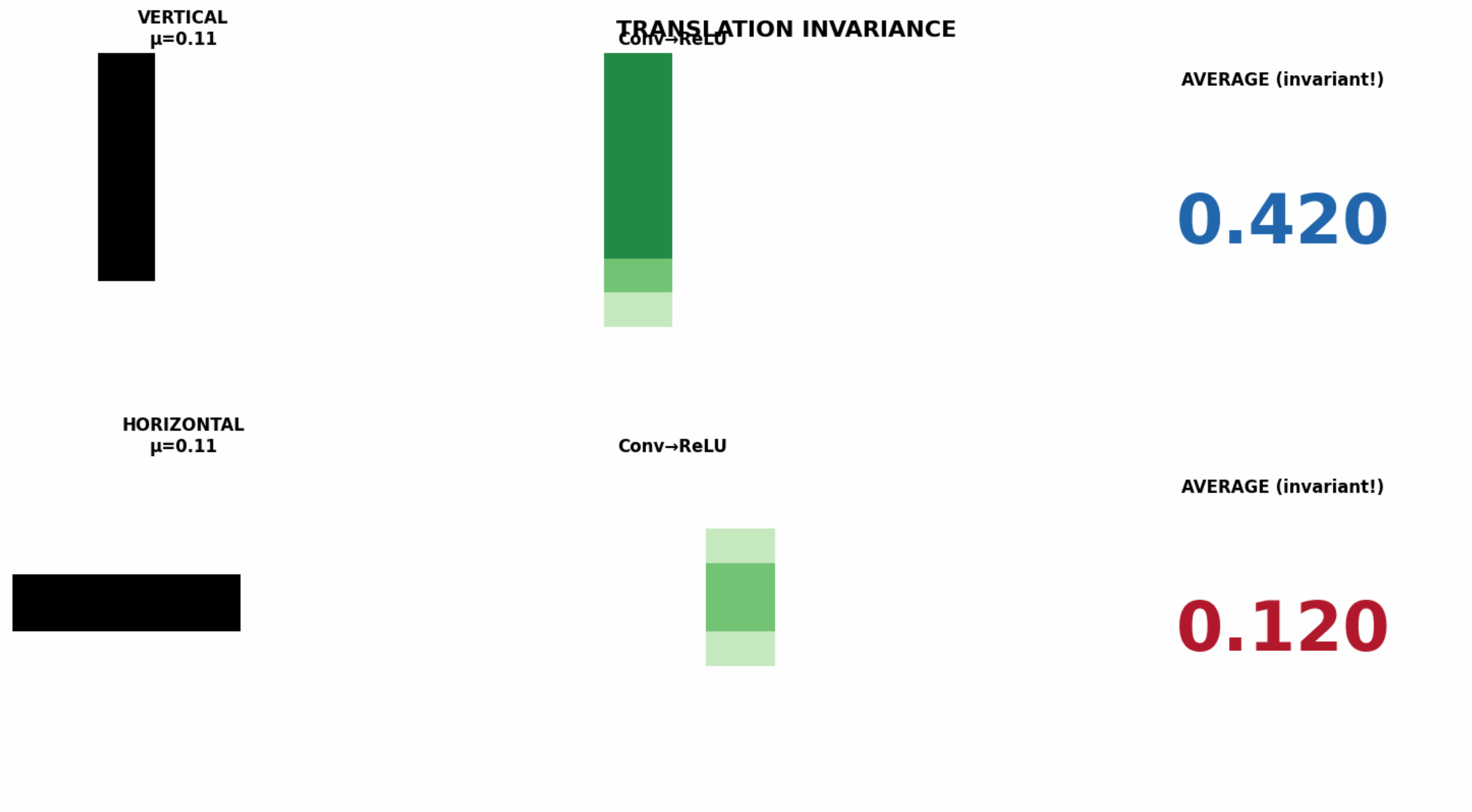
# Your First Convolutional Network



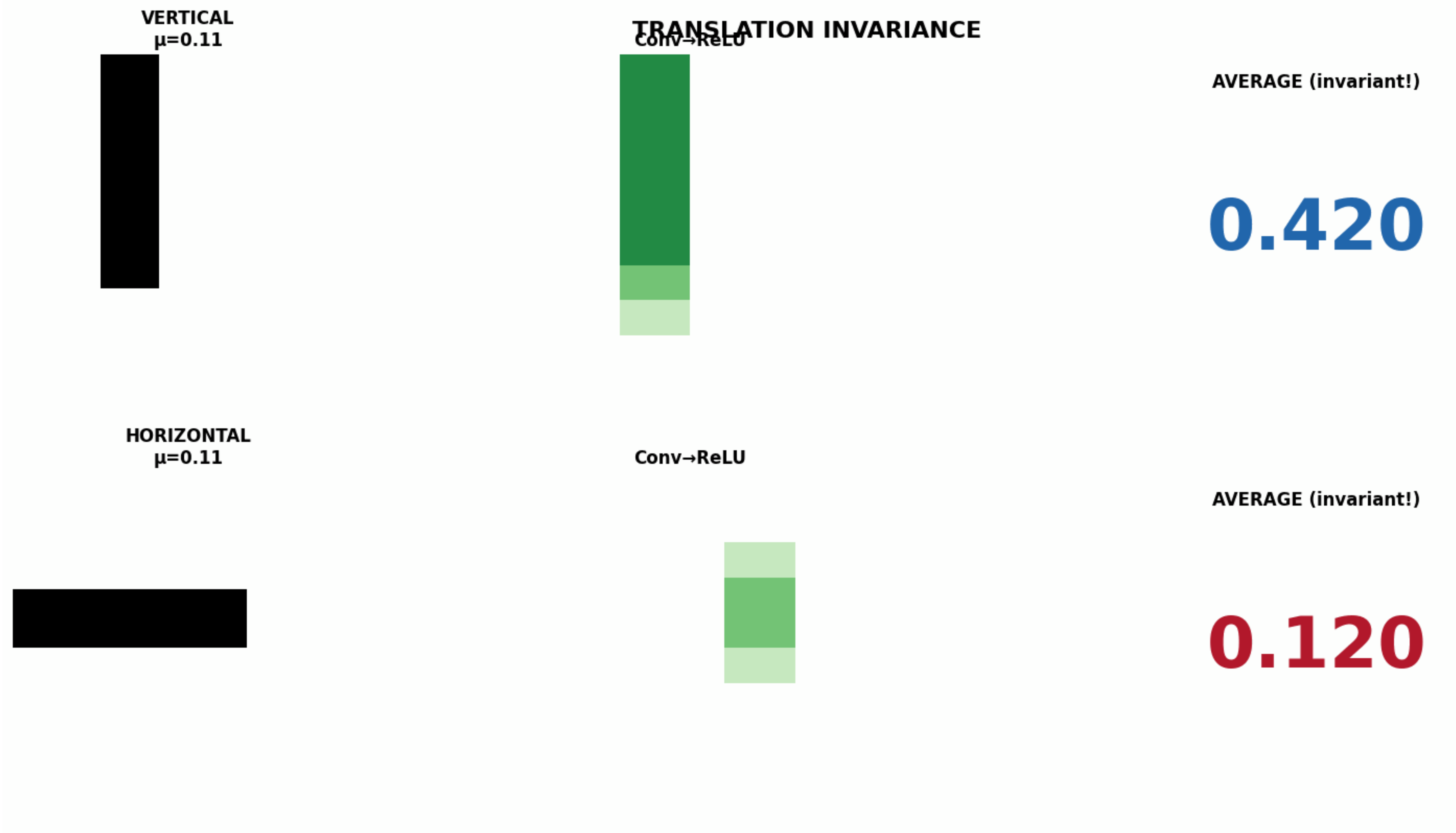
# Your First Convolutional Network



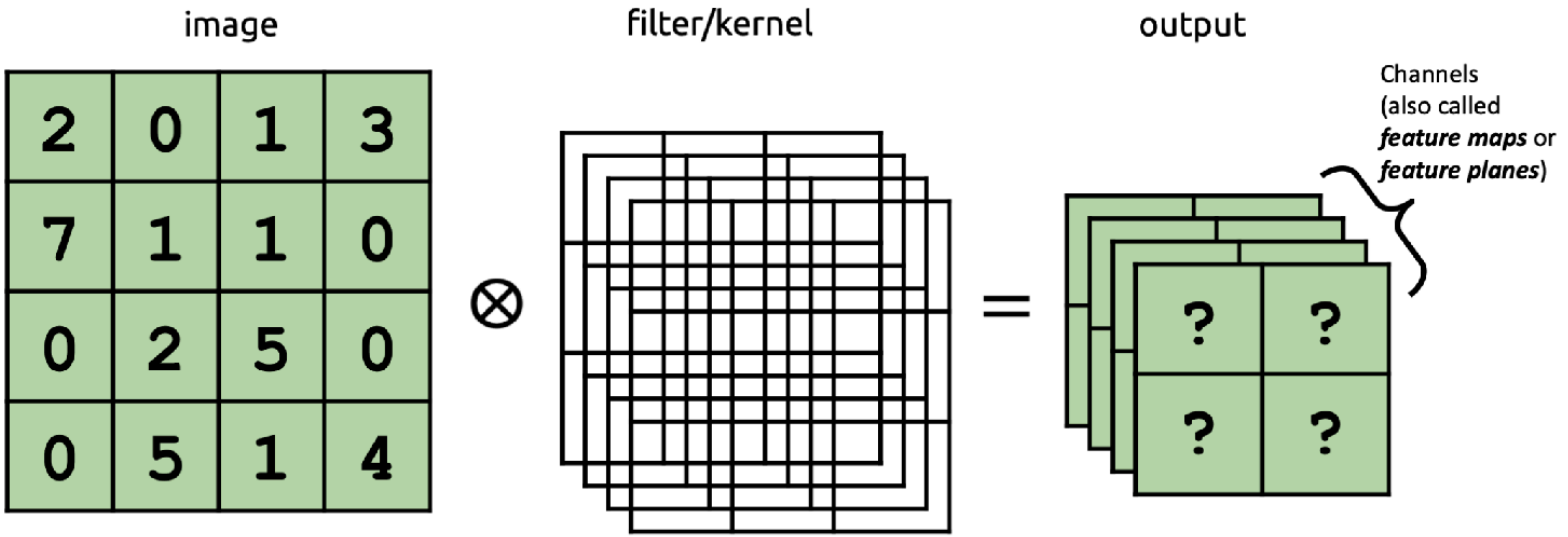
# Your First Convolutional Network



# Your First Convolutional Network



# Multiple Filters



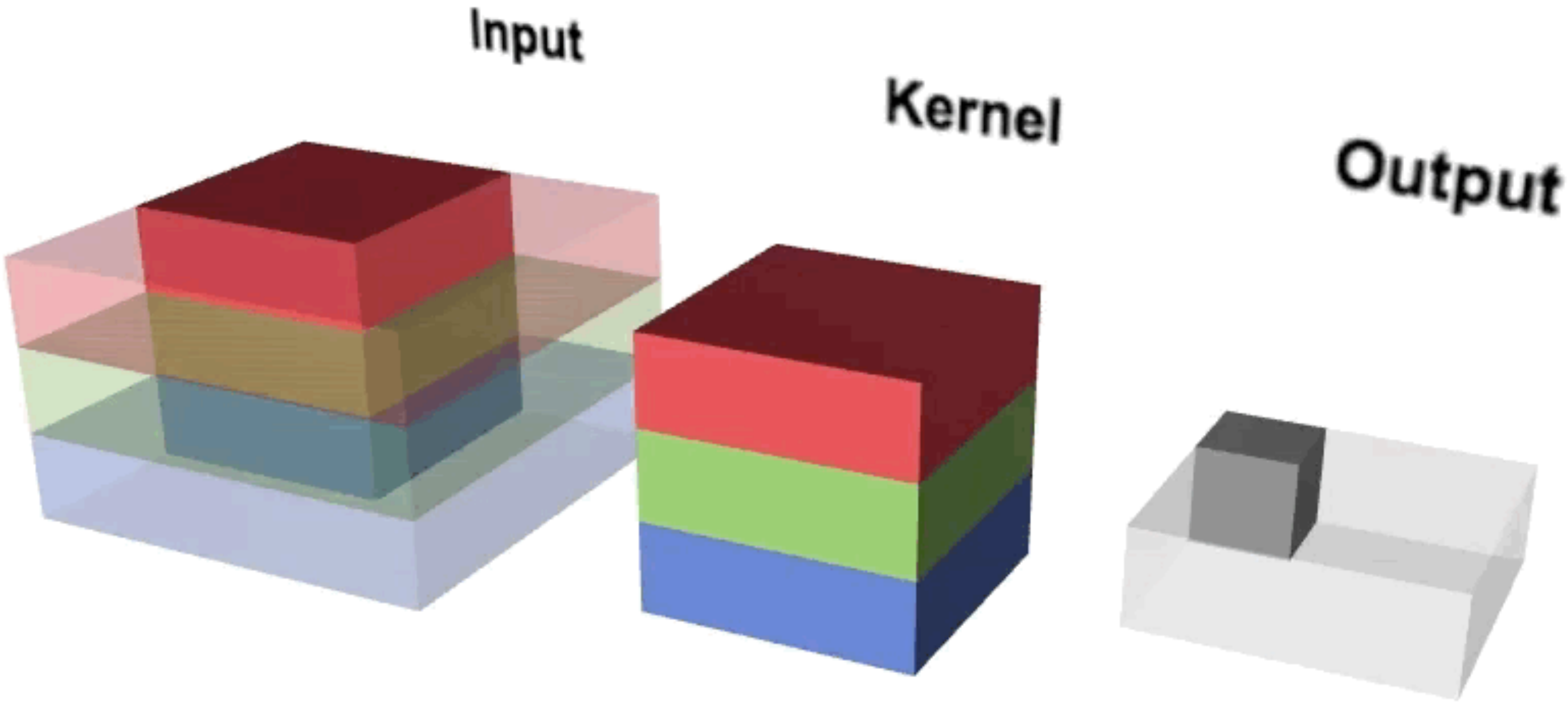
|   |   |   |   |
|---|---|---|---|
| 2 | 0 | 1 | 3 |
| 7 | 1 | 1 | 0 |
| 0 | 2 | 5 | 0 |
| 0 | 5 | 1 | 4 |

|   |   |
|---|---|
| ? | ? |
| ? | ? |

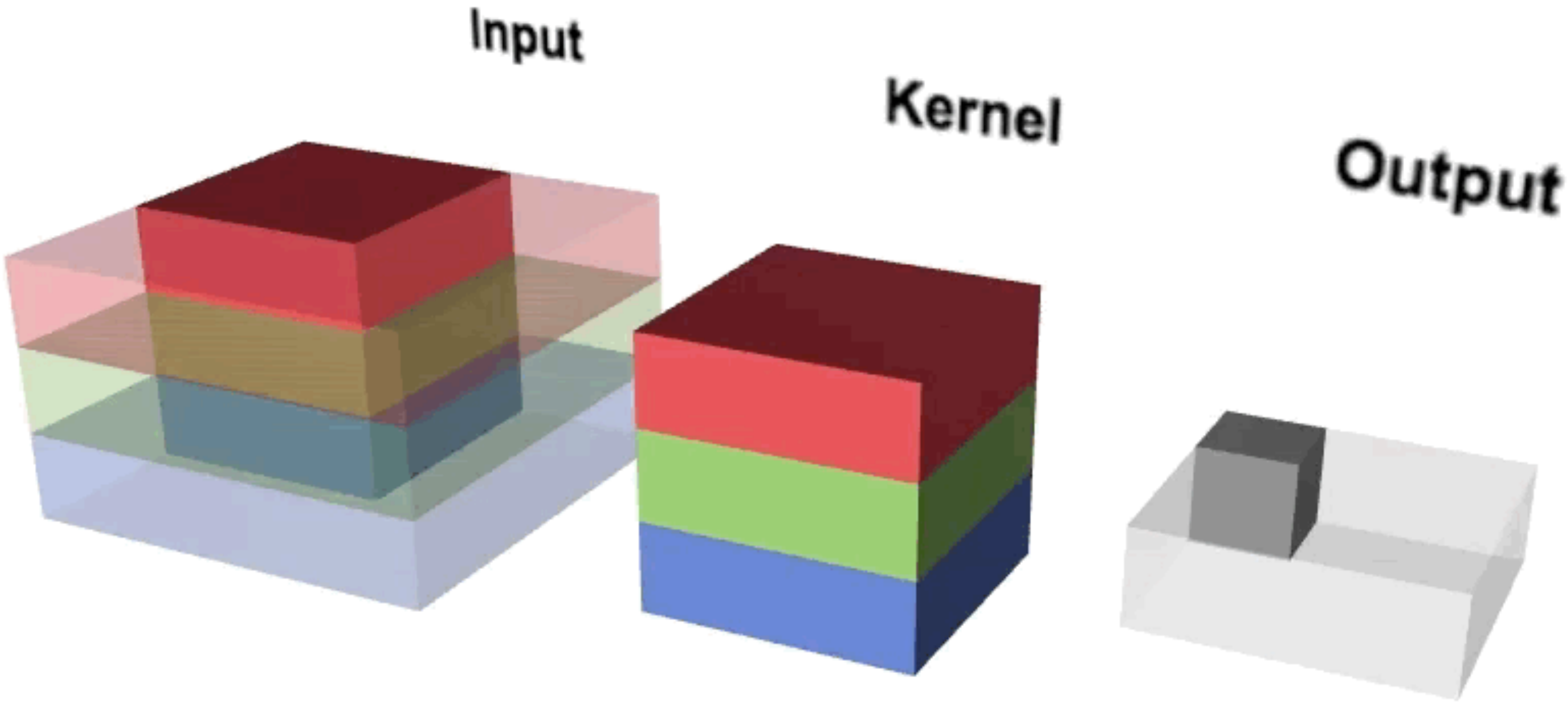
Channels
(also called
***feature maps*** or
***feature planes***)

The output is now a <u>multi-channel</u> image

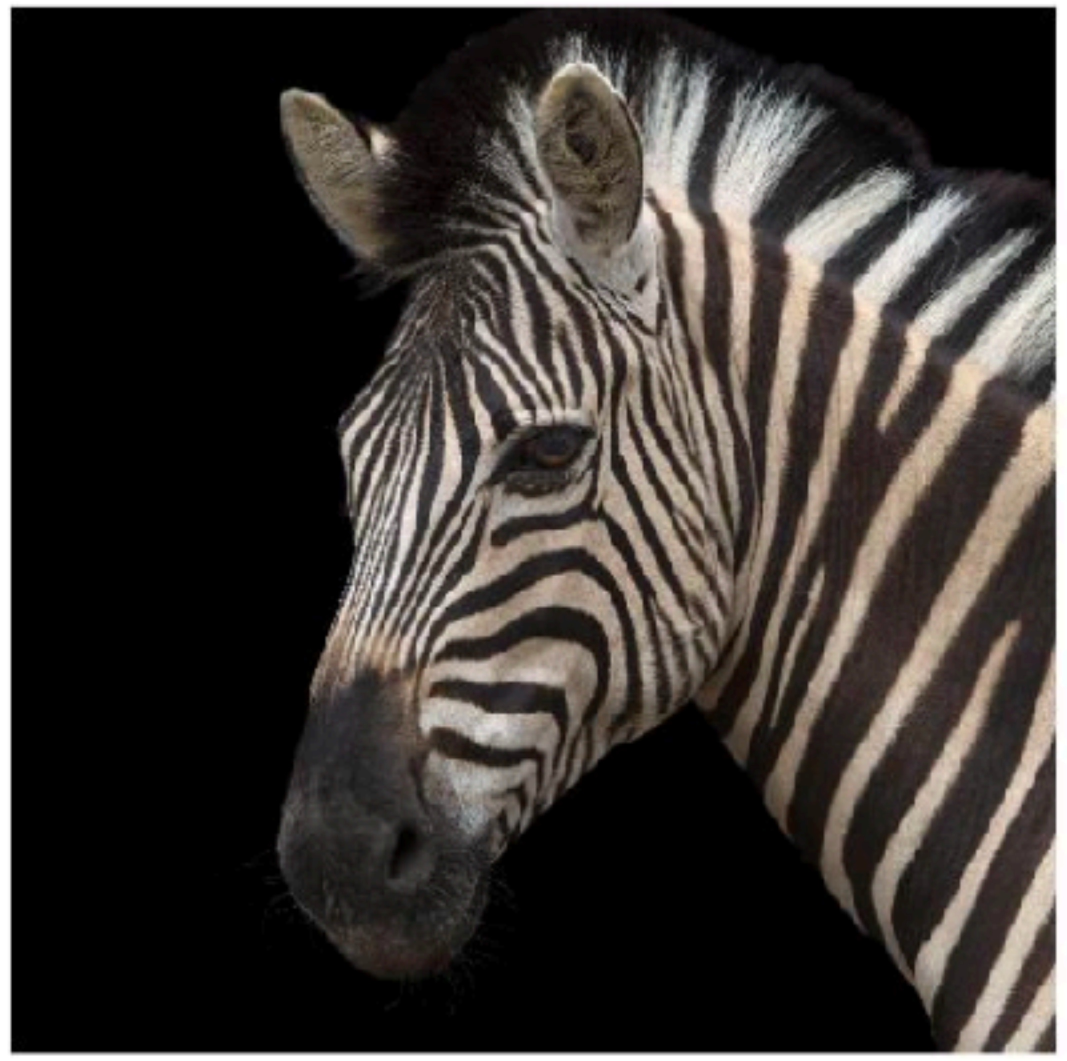
# Multiple Multi-Channel Filters



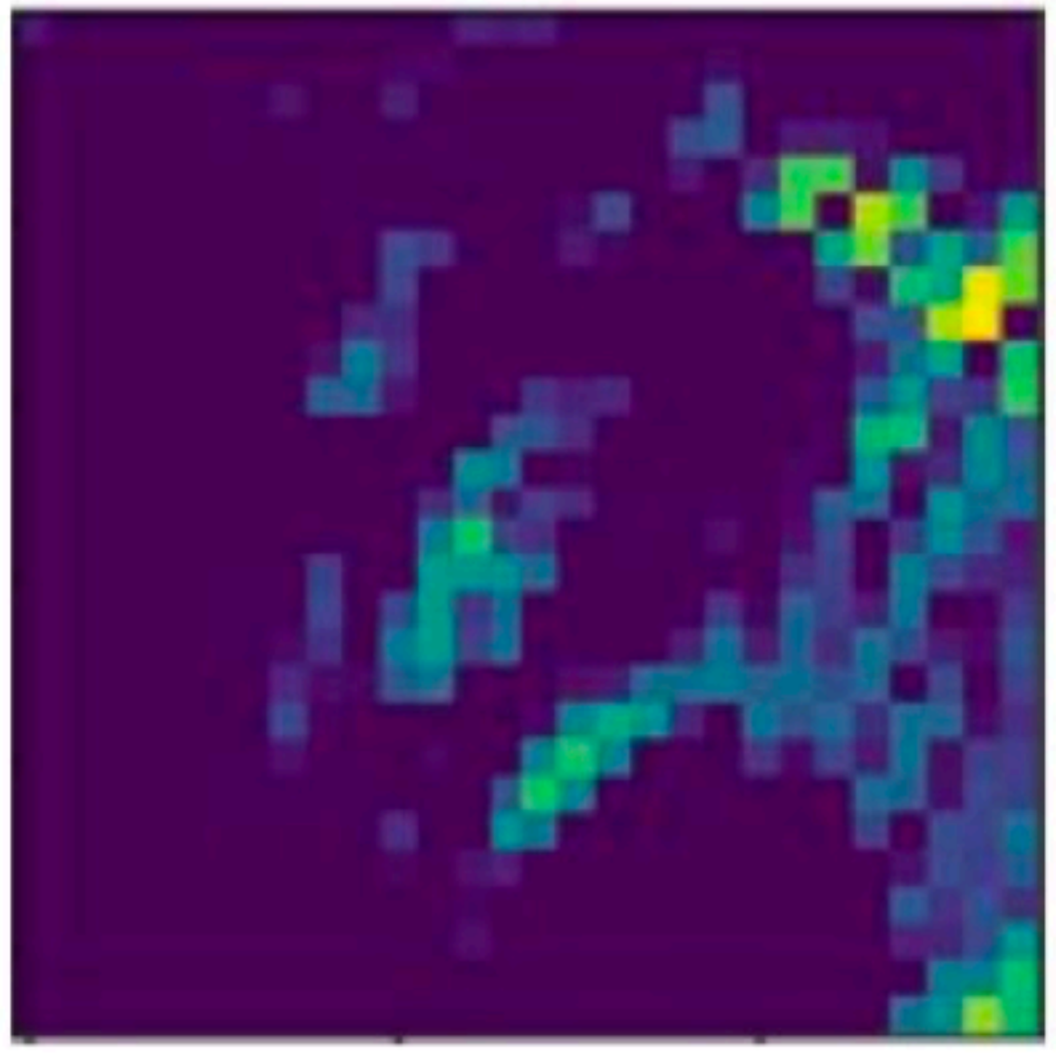
# Multiple Multi-Channel Filters



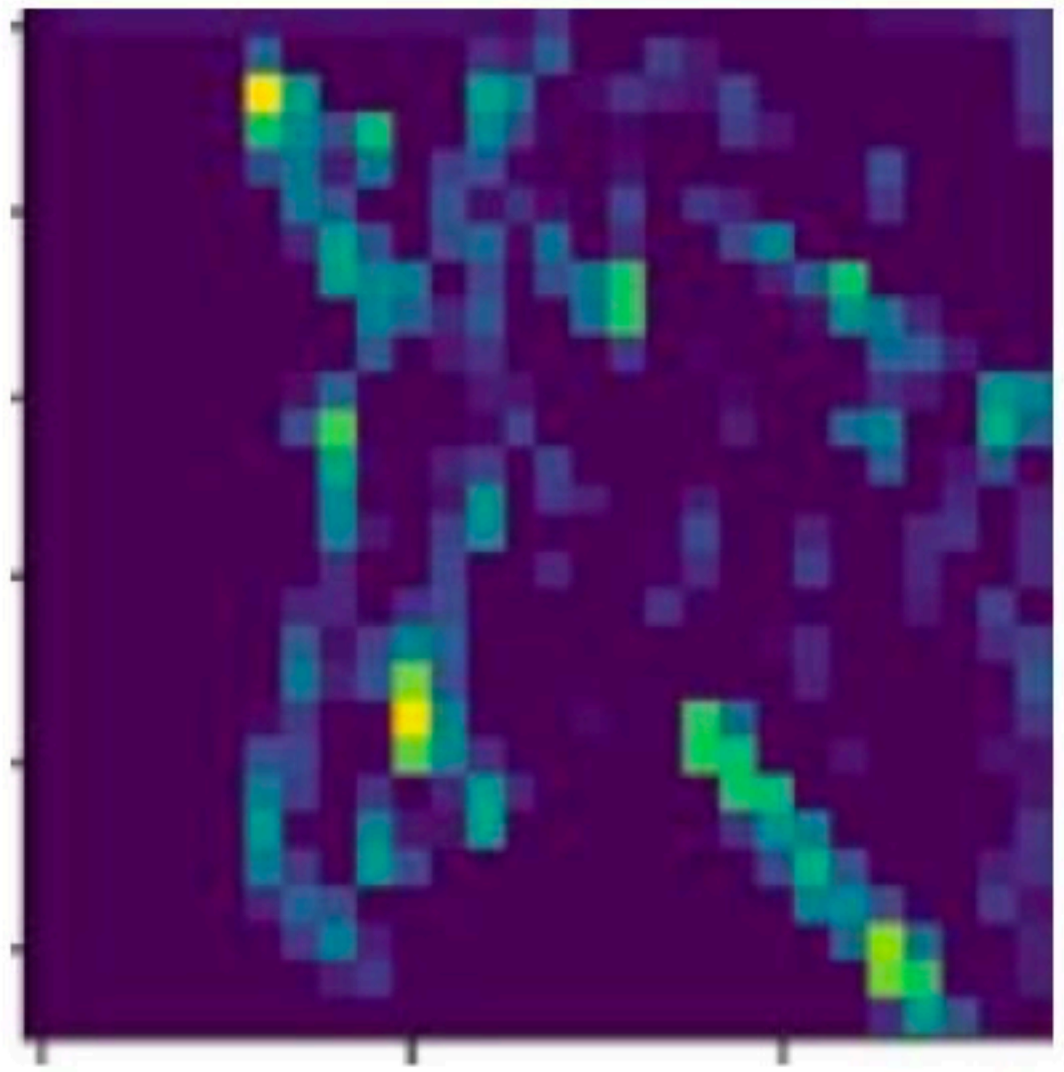
# Multiple Multi-Channel Filters



Input image

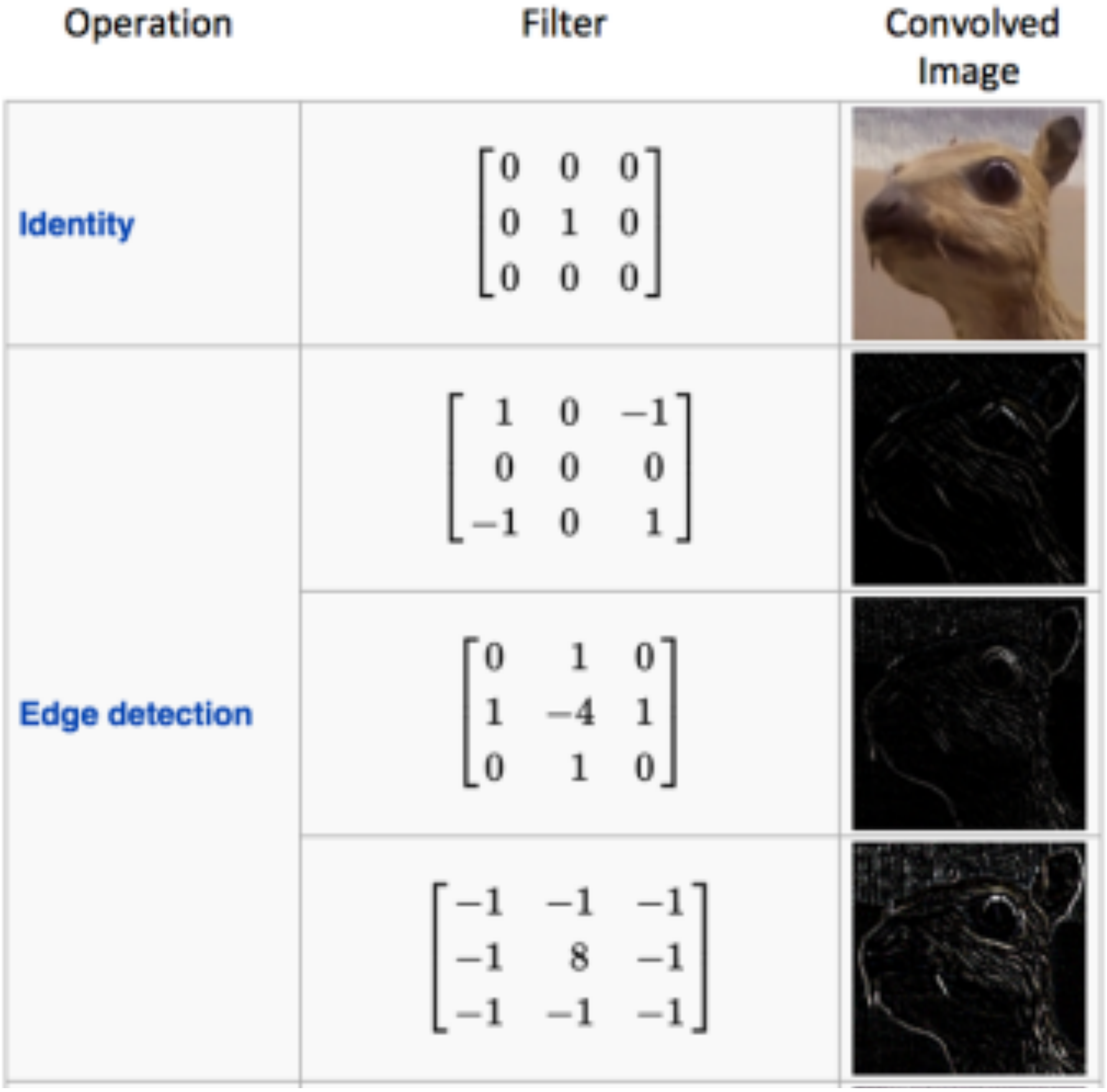


Output of filter 1

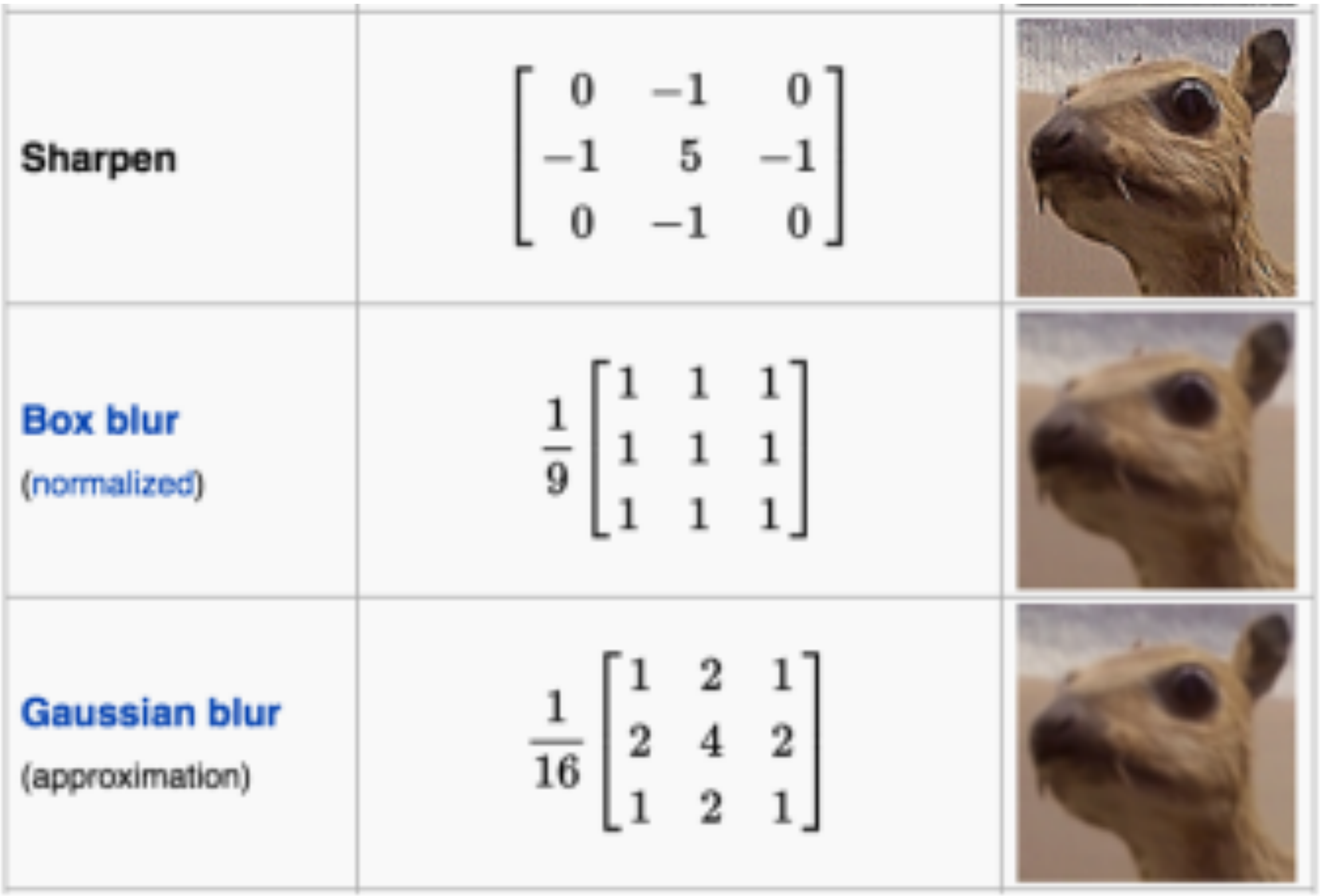


Output of filter 2

# More Filter Examples



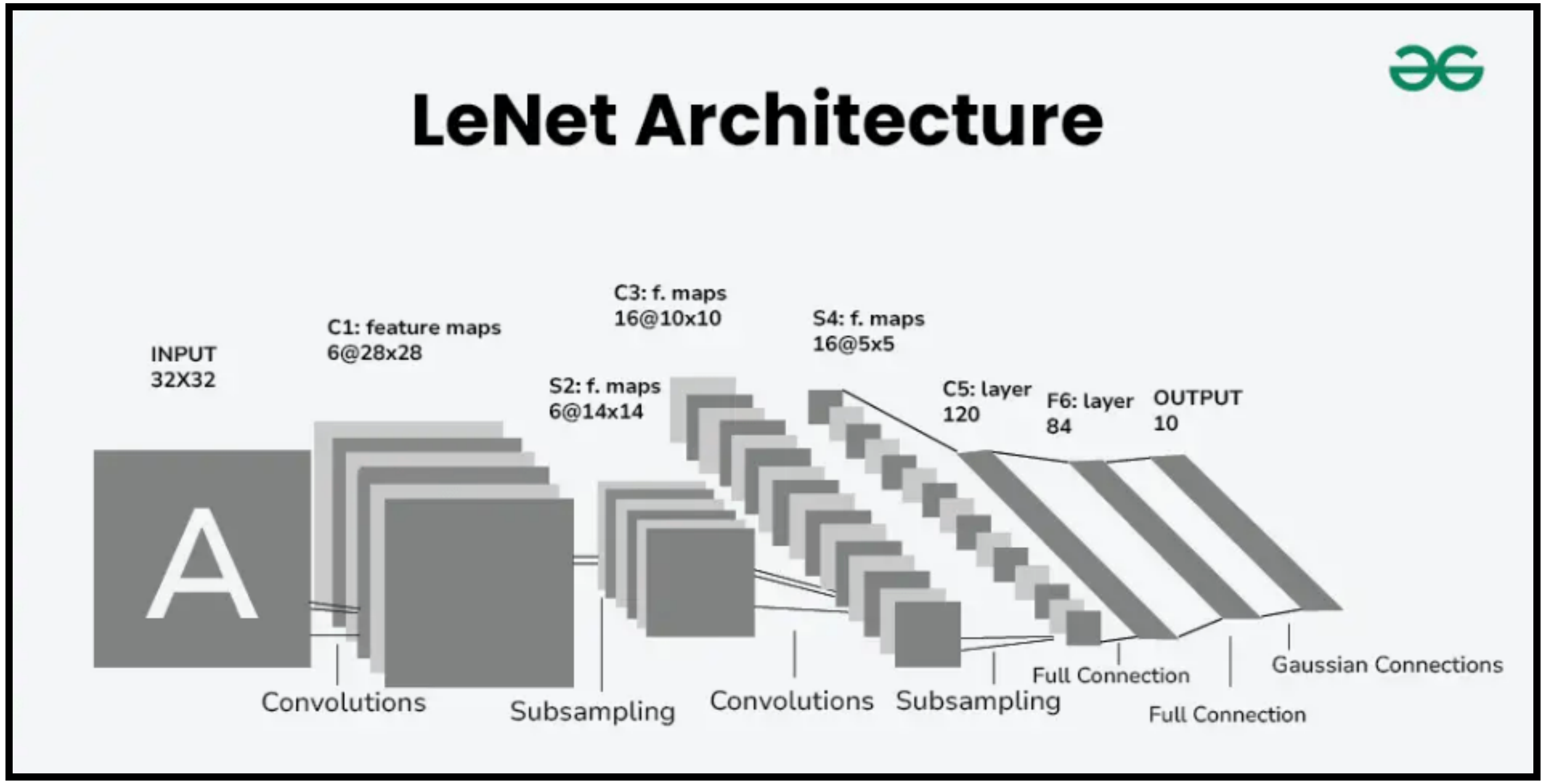
| Operation | Filter | Convolved Image |
|---|---|---|
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ | |
| Edge detection | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ | |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ | |
| | $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ | |



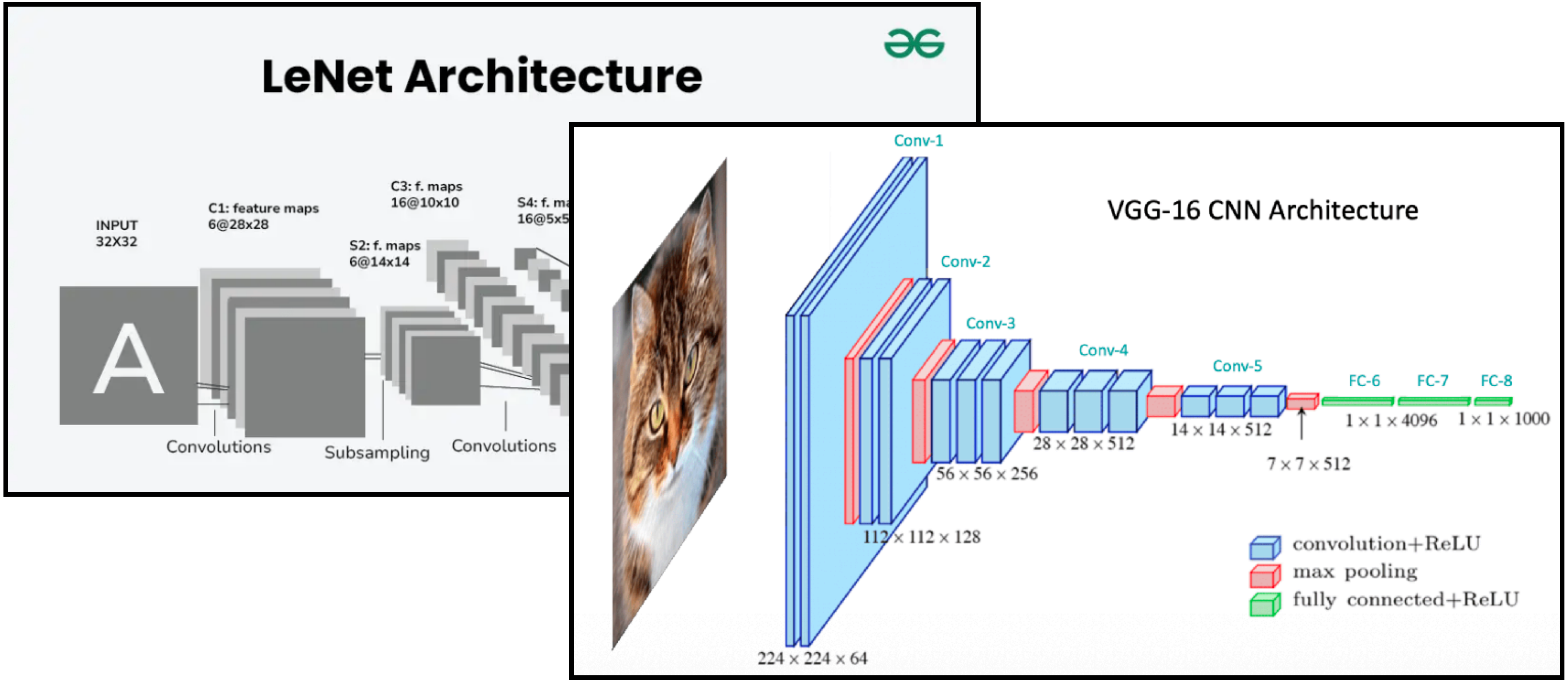
| | | |
|---|---|---|
| Sharpen | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ | |
| Box blur<br>(normalized) | $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ | |
| Gaussian blur<br>(approximation) | $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ | |

# Deep Convolutional Networks

# Deep Convolutional Networks

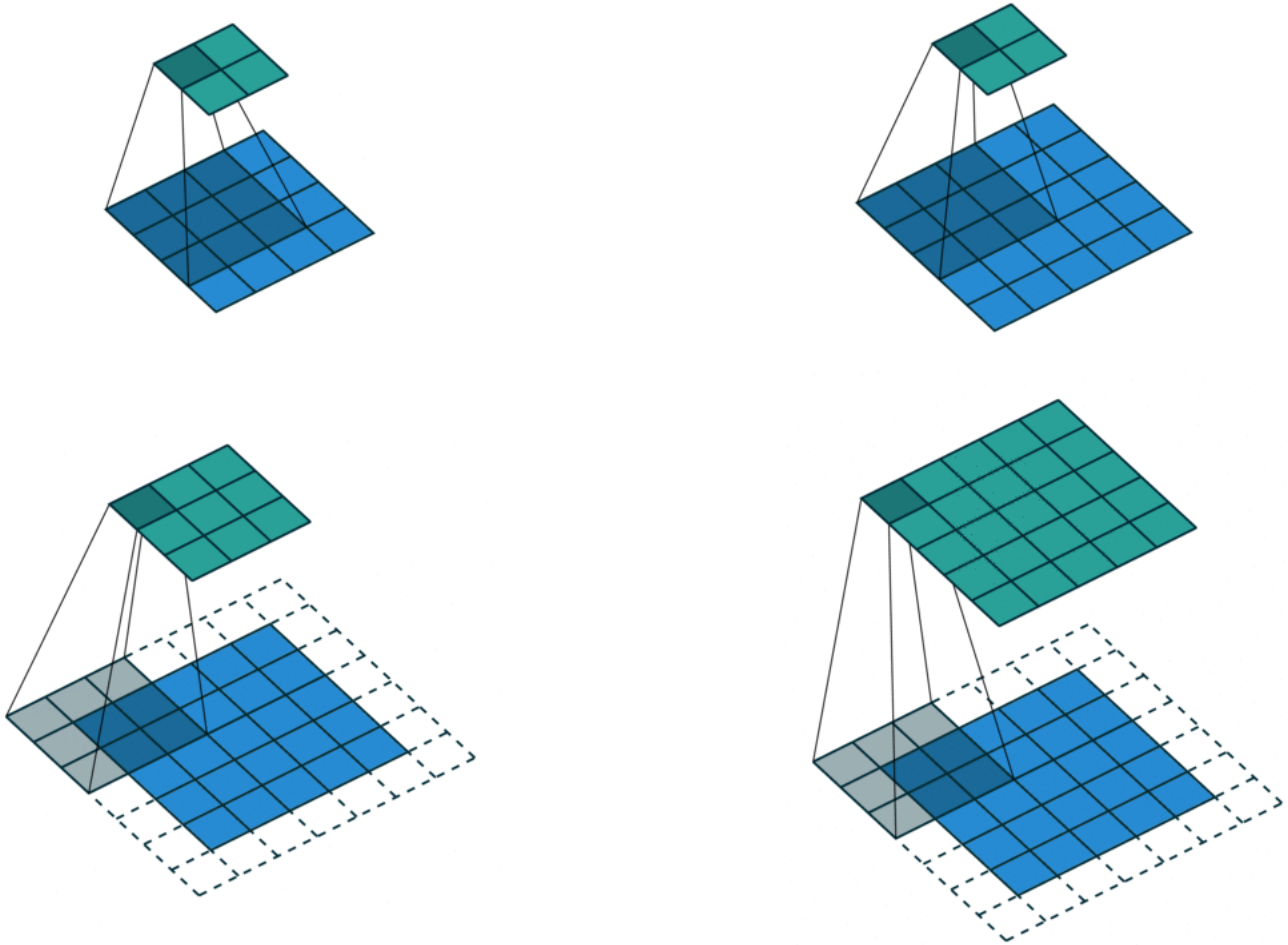


# Deep Convolutional Networks



# Other Convolution Parameters

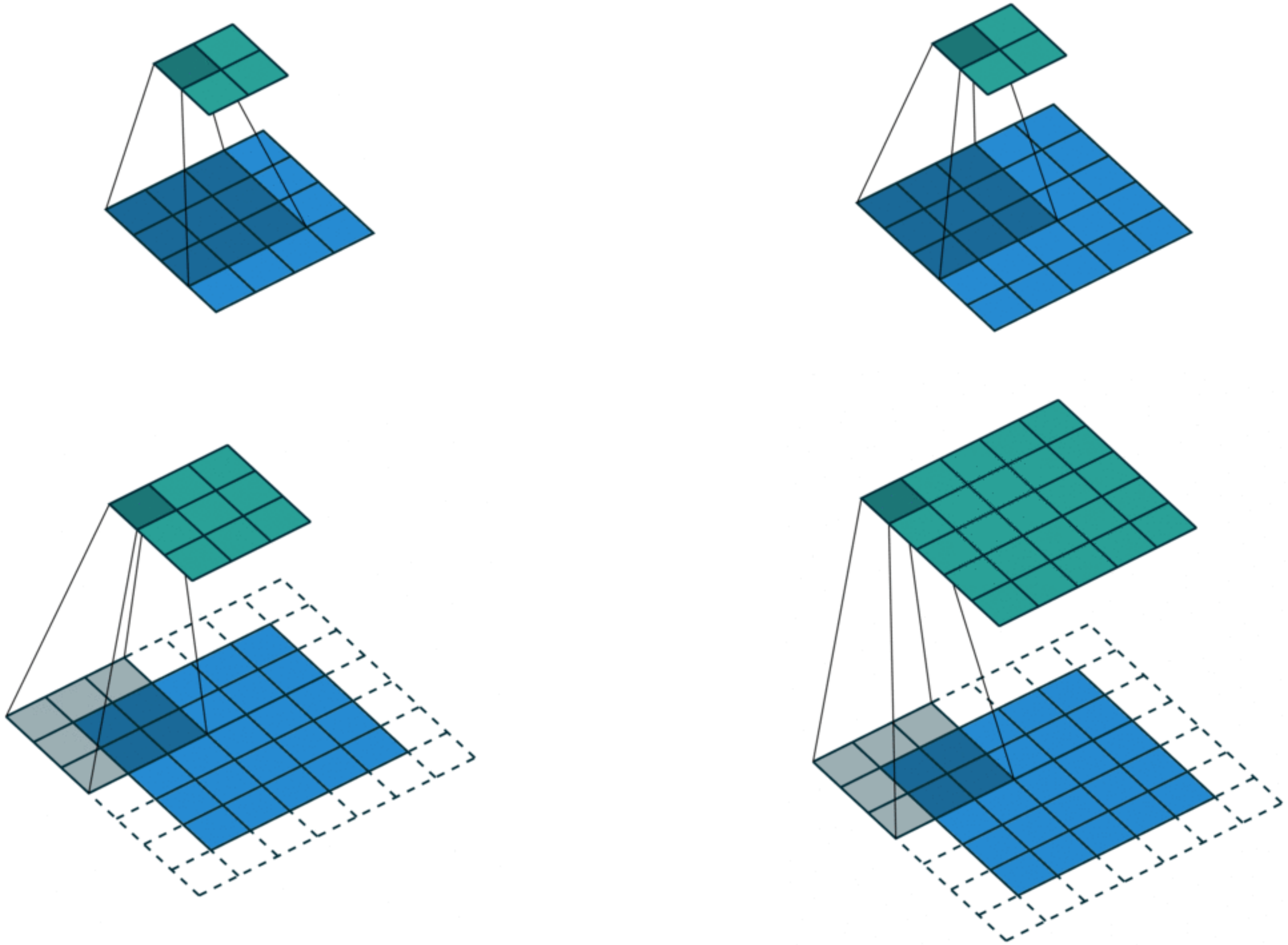
## Stride and padding!



https://github.com/vdumoulin/conv_arithmetic

# Other Convolution Parameters

## Stride and padding!



https://github.com/vdumoulin/conv_arithmetic

# Provable Benefit of Convolution

- You will encounter the same “patterns” at different translations (position)
- Holds for 1d (audio, music), 2d (images), 3d (videos)
- This is what the brain does!

# Provable Benefit of Convolution

- You will encounter the same “patterns” at different translations (position)
- Holds for 1d (audio, music), 2d (images), 3d (videos)
- This is what the brain does!

Exercise: can you think of inputs for which we DO NOT want convolutions?