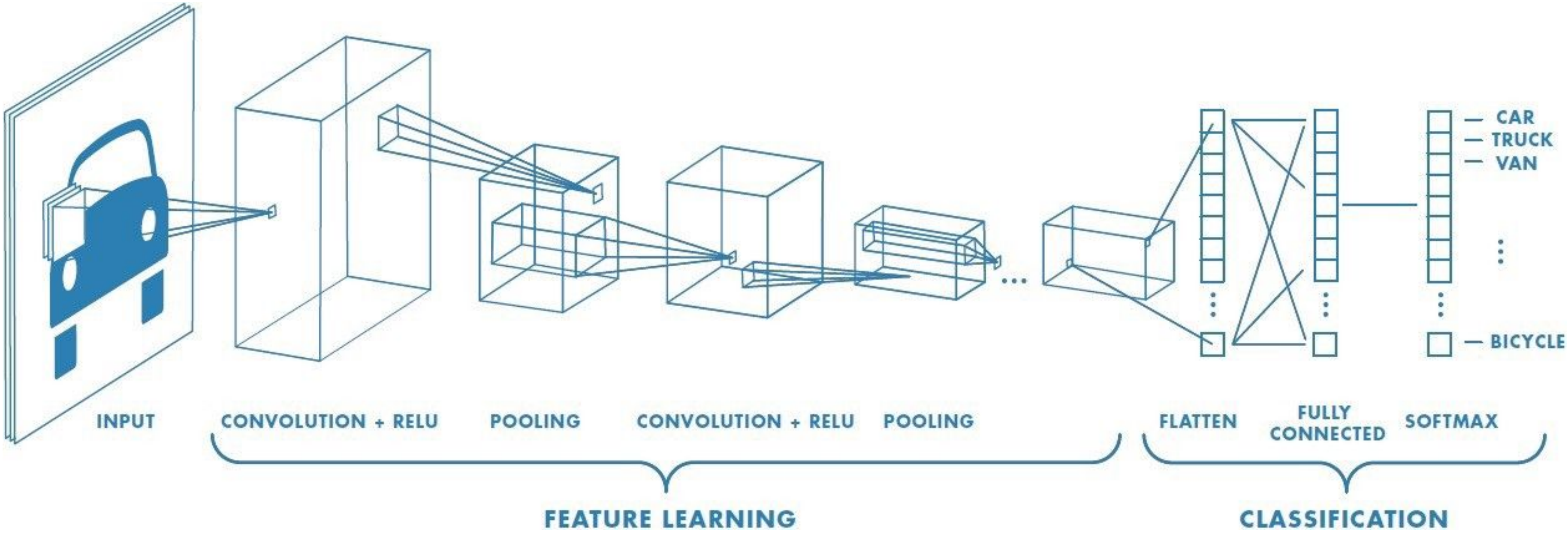
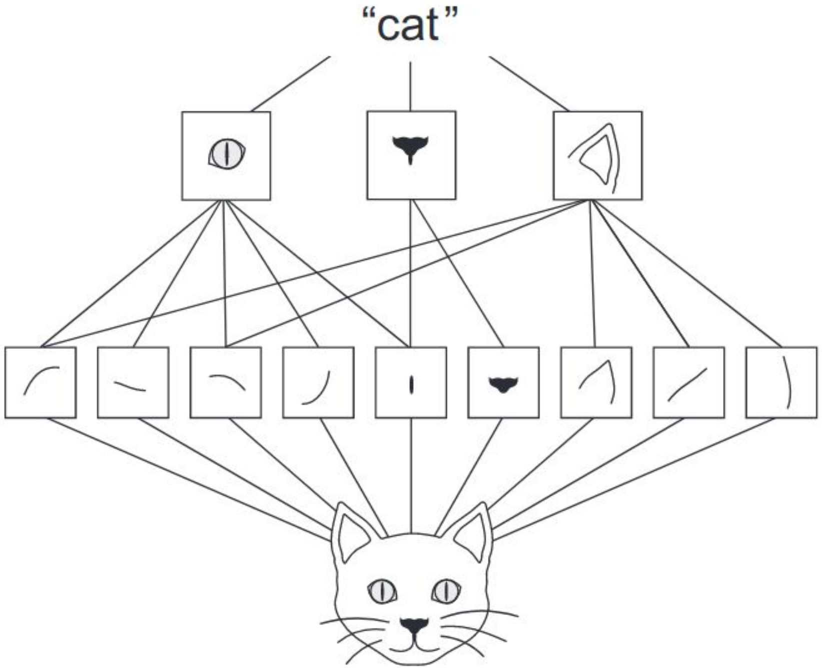
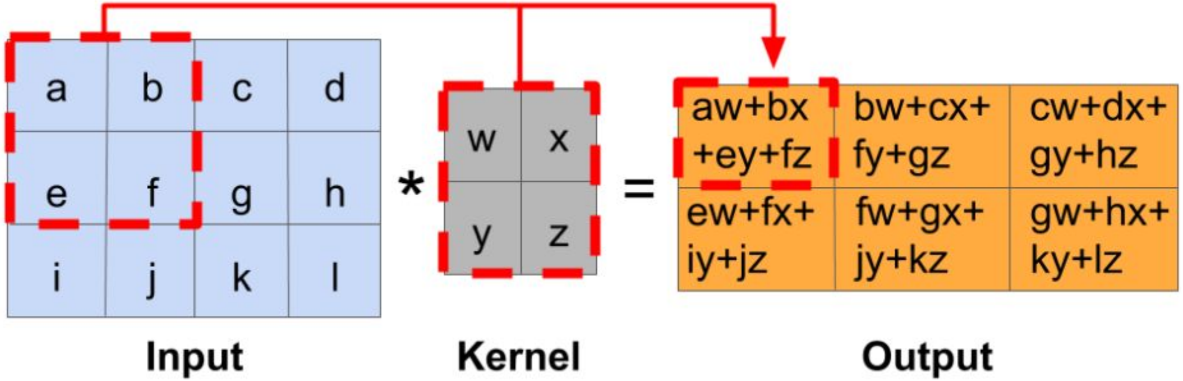
# Convolutional Neural Networks (CNNs)

# Convolutional Neural Networks (CNNs)




"cat"

# Convolution



|   |   |   |   |
|---|---|---|---|
| a | b | c | d |
| e | f | g | h |
| i | j | k | l |

Input

*

|   |   |
|---|---|
| w | x |
| y | z |

Kernel

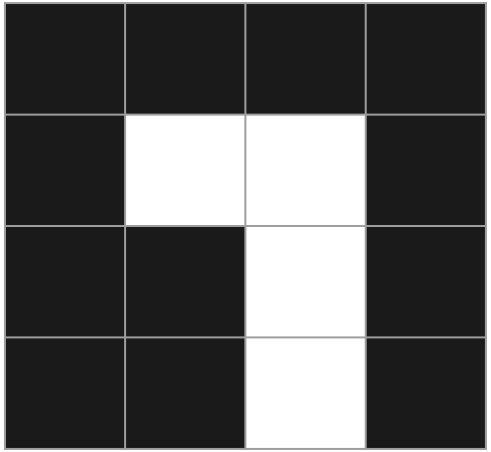
=

|   |   |   |
|---|---|---|
| aw+bx<br>+ey+fz | bw+cx+<br>fy+gz | cw+dx+<br>gy+hz |
| ew+fx+<br>iy+jz | fw+gx+<br>jy+kz | gw+hx+<br>ky+lz |

Output

Fuente: Bengio, Goodfellow y Courville (2016)

# Convolution



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

# Convolution

Input (4x4)

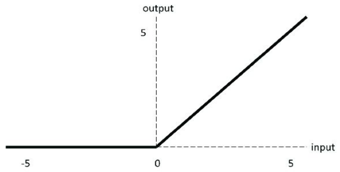
|  |  |  |  |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 |

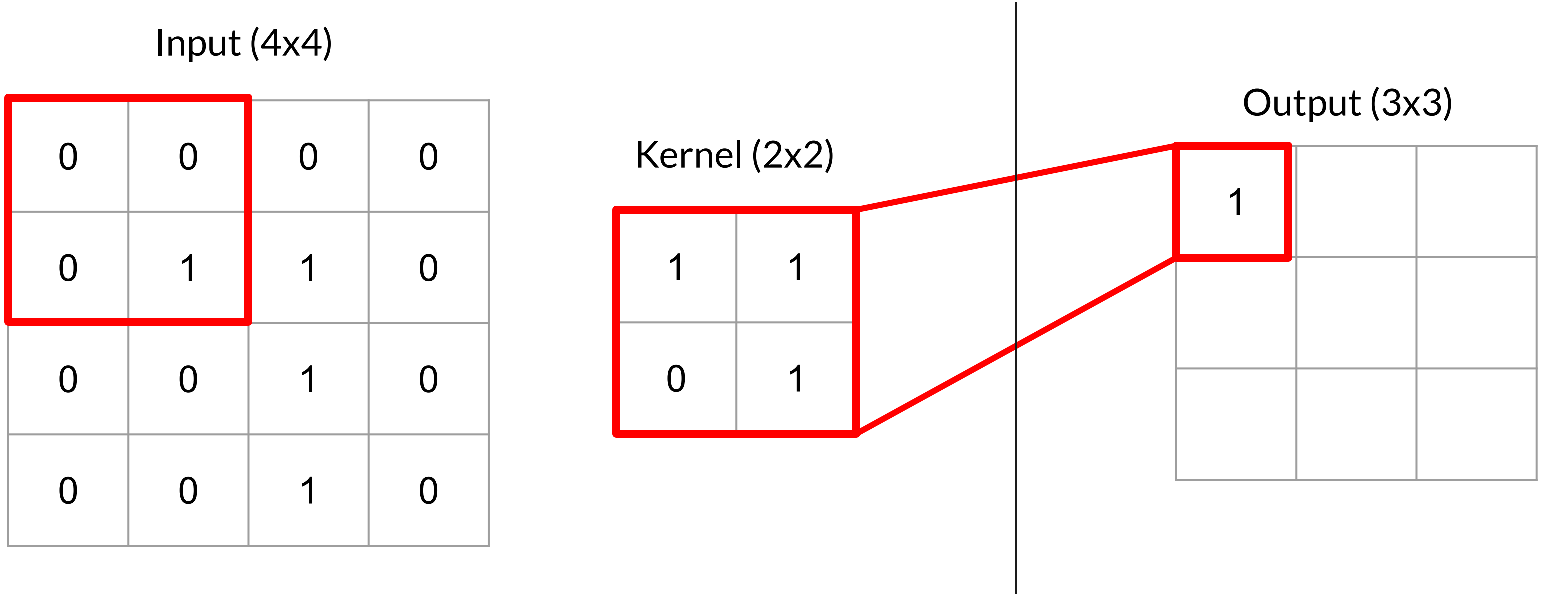
Kernel (2x2)

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

# Convolution

ReLU





Input (4x4)

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

Kernel (2x2)

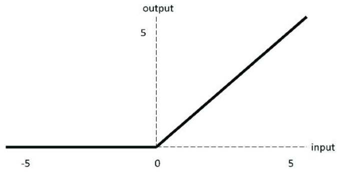
| | |
|---|---|
| 1 | 1 |
| 0 | 1 |

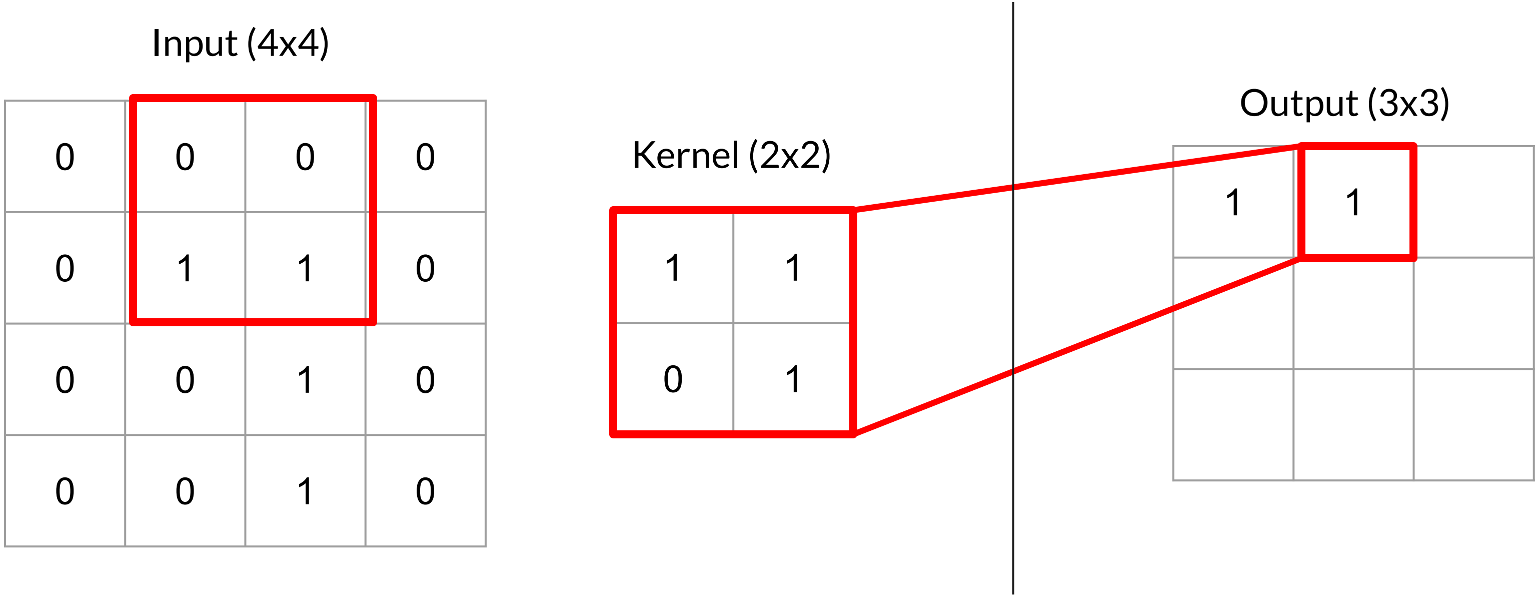
Output (3x3)

| | | |
|---|---|---|
| 1 | | |
| | | |
| | | |

# Convolution

ReLU





Input (4x4)

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

Kernel (2x2)

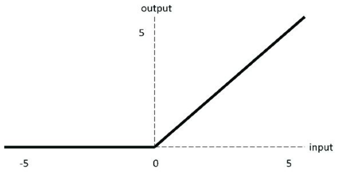
| | |
|---|---|
| 1 | 1 |
| 0 | 1 |

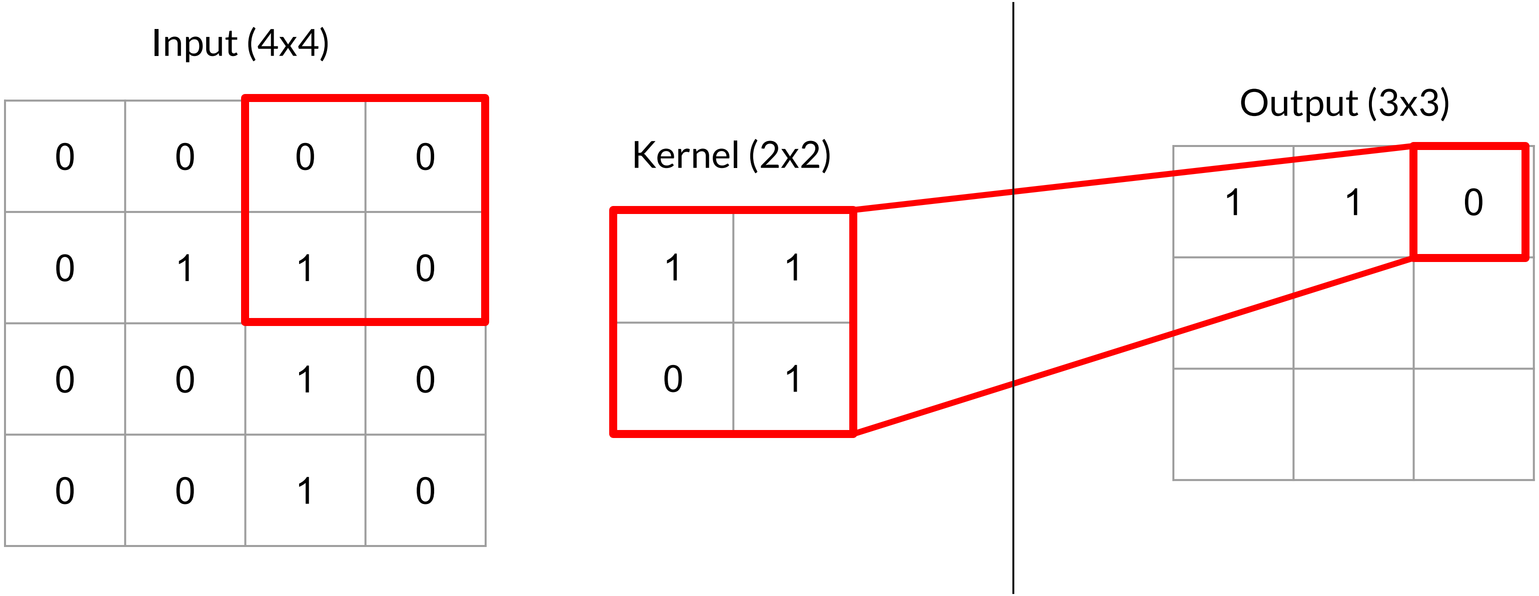
Output (3x3)

| | | |
|---|---|---|
| 1 | 1 | |
| | | |
| | | |

# Convolution

ReLU





Input (4x4)

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

Kernel (2x2)

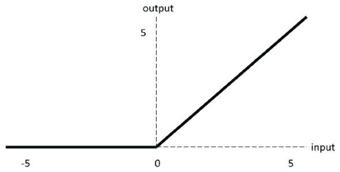
| | |
|---|---|
| 1 | 1 |
| 0 | 1 |

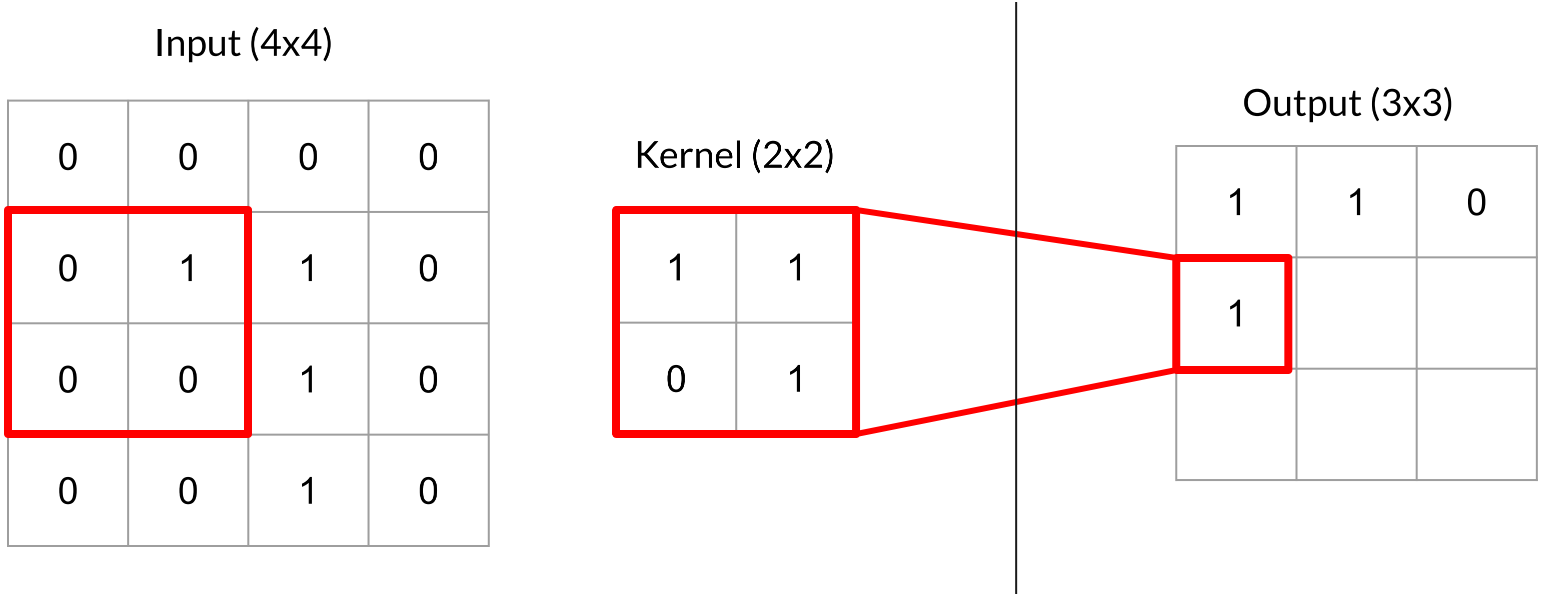
Output (3x3)

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

# Convolution

ReLU





Input (4x4)

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

Kernel (2x2)

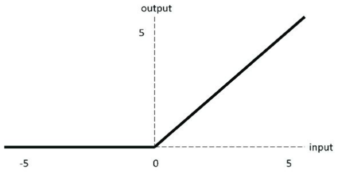
| | |
|---|---|
| 1 | 1 |
| 0 | 1 |

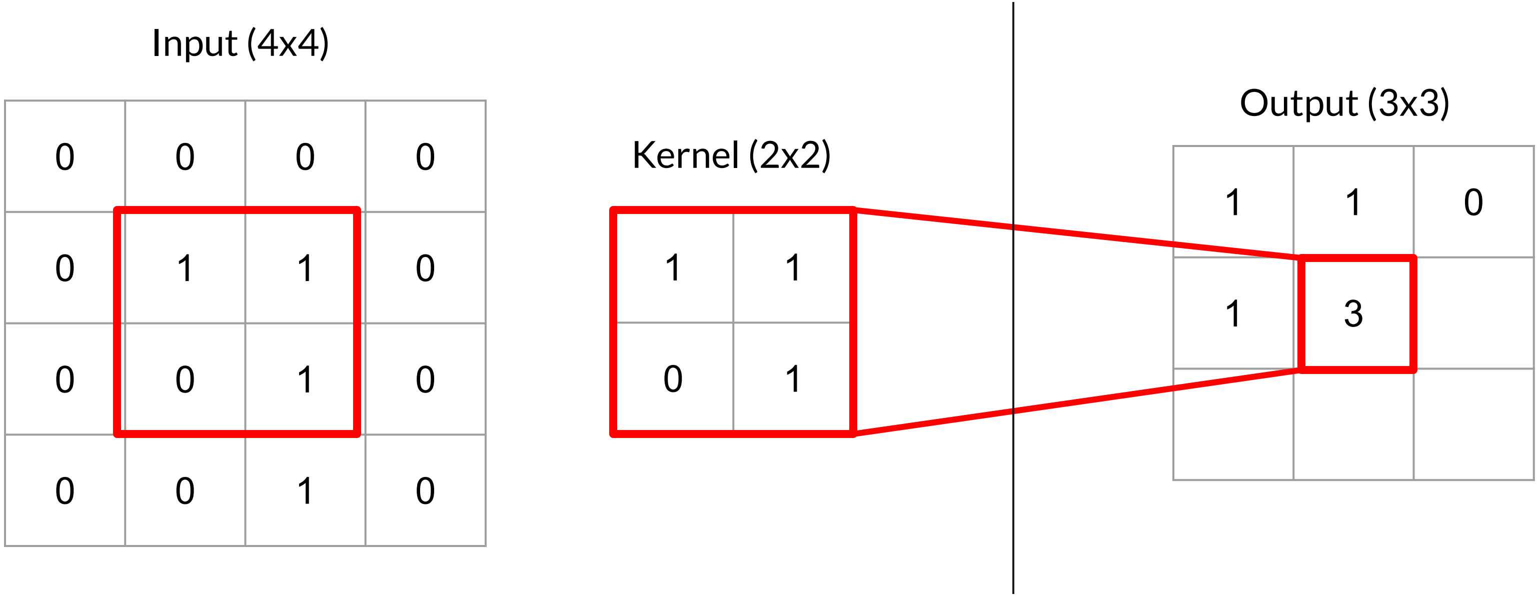
Output (3x3)

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

# Convolution

ReLU





Input (4x4)

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

Kernel (2x2)

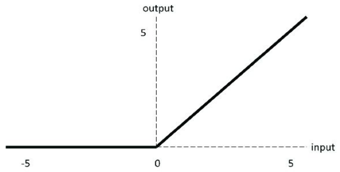
| | |
|---|---|
| 1 | 1 |
| 0 | 1 |

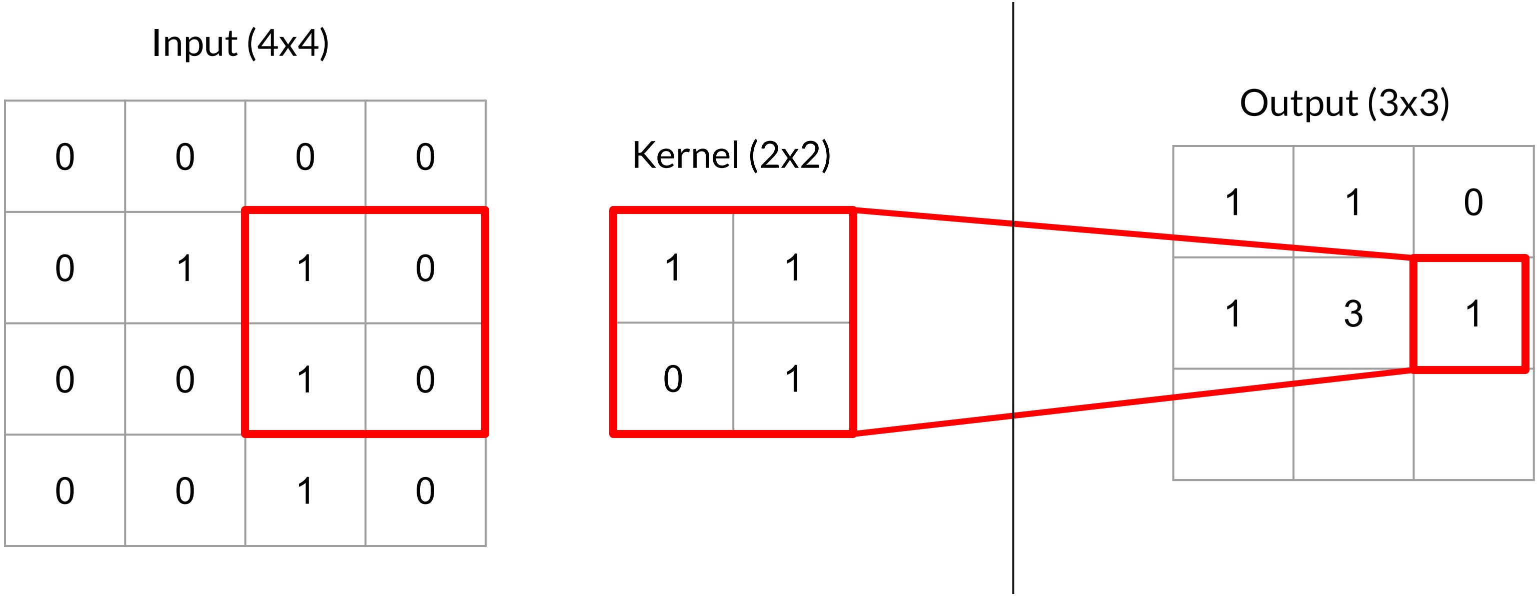
Output (3x3)

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | |
| | | |

# Convolution

ReLU





Input (4x4)

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

Kernel (2x2)

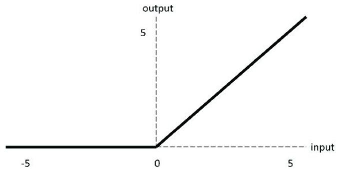
|   |   |
|---|---|
| 1 | 1 |
| 0 | 1 |

Output (3x3)

|   |   |   |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
|   |   |   |

# Convolution

ReLU



Input (4x4)

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

Kernel (2x2)

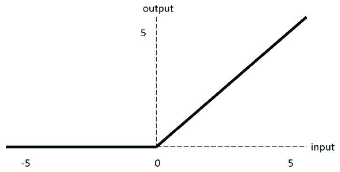
| | |
|---|---|
| 1 | 1 |
| 0 | 1 |

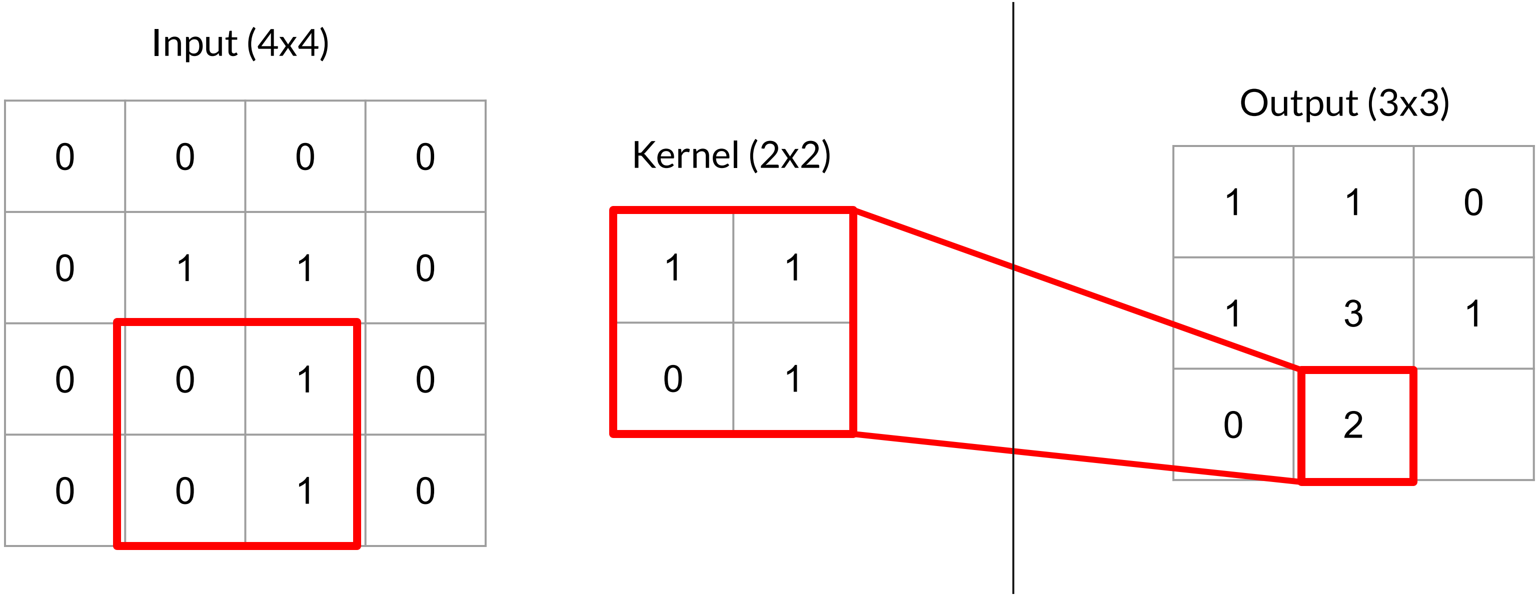
Output (3x3)

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | | |

# Convolution

ReLU





Input (4x4)

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

Kernel (2x2)

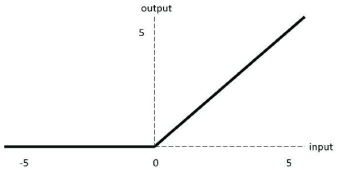
|   |   |
|---|---|
| 1 | 1 |
| 0 | 1 |

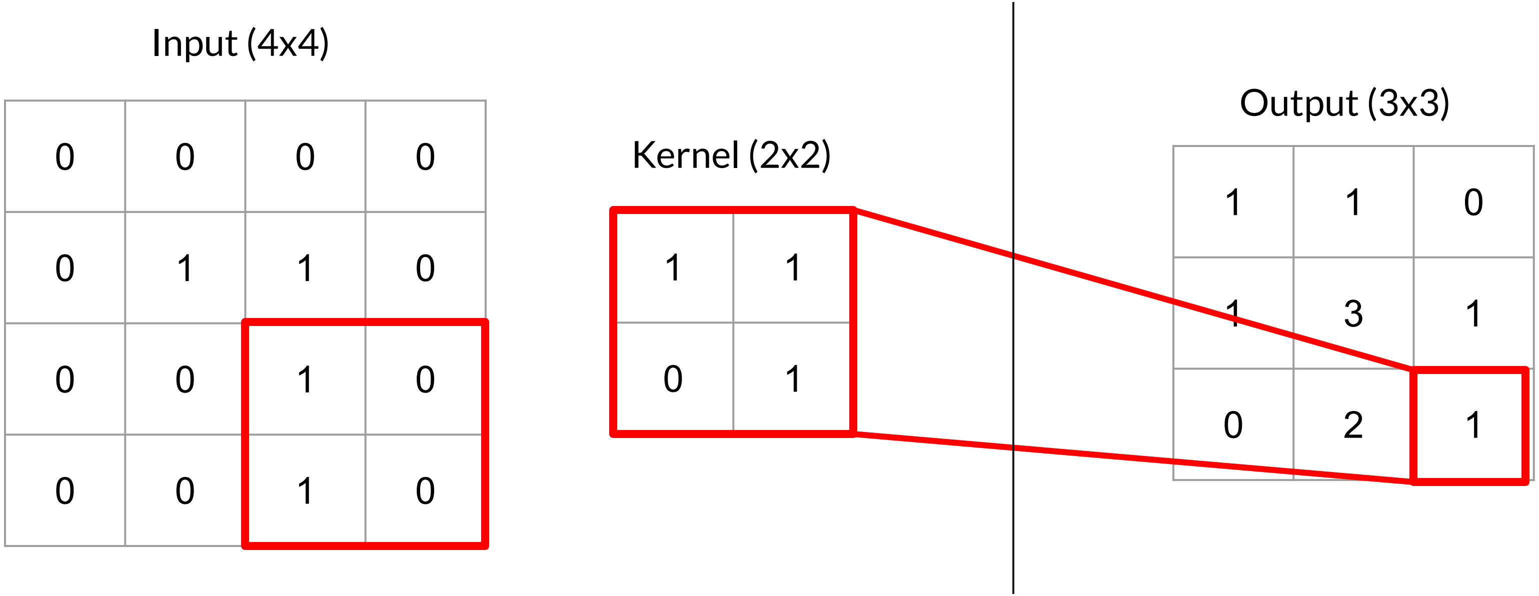
Output (3x3)

|   |   |   |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | 2 |   |

# Convolution

ReLU





Input (4x4)

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

Kernel (2x2)

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

Output (3x3)

|   |   |   |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | 2 | 1 |

# Convolution

Input (4x4)

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

Kernel (2x2)

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

Output (3x3)

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | 2 | 1 |

# Max-Pooling (2x2)

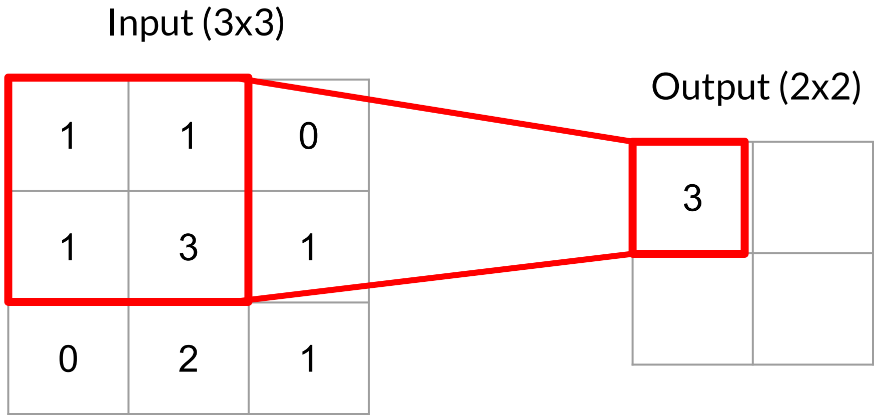
Input (3x3)

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | 2 | 1 |

Output (2x2)

| | |
|---|---|
| | |
| | |

# Max-Pooling (2x2)



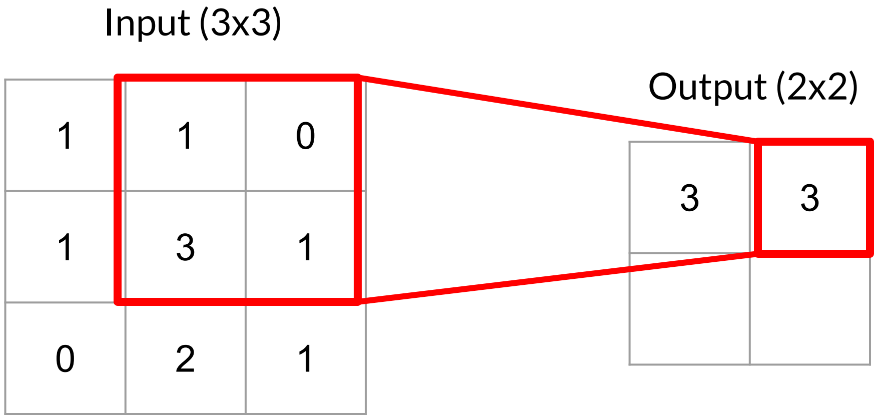
Input (3x3)

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | 2 | 1 |

Output (2x2)

| | |
|---|---|
| 3 | |
| | |

# Max-Pooling (2x2)



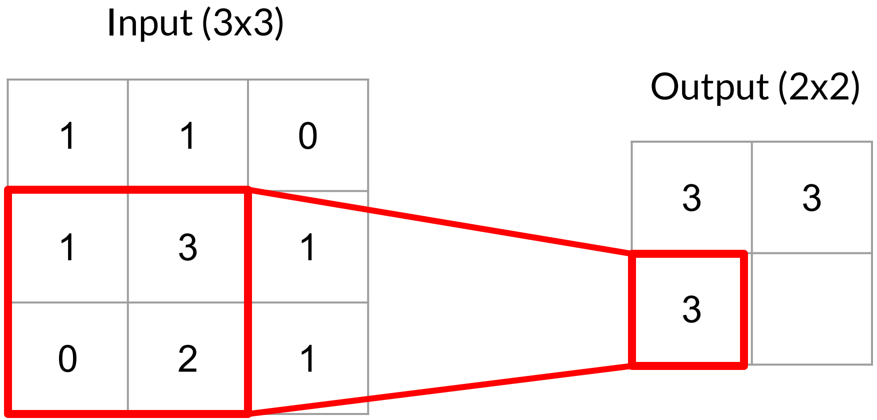
Input (3x3)

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | 2 | 1 |

Output (2x2)

| | |
|---|---|
| 3 | 3 |
| | |

# Max-Pooling (2x2)



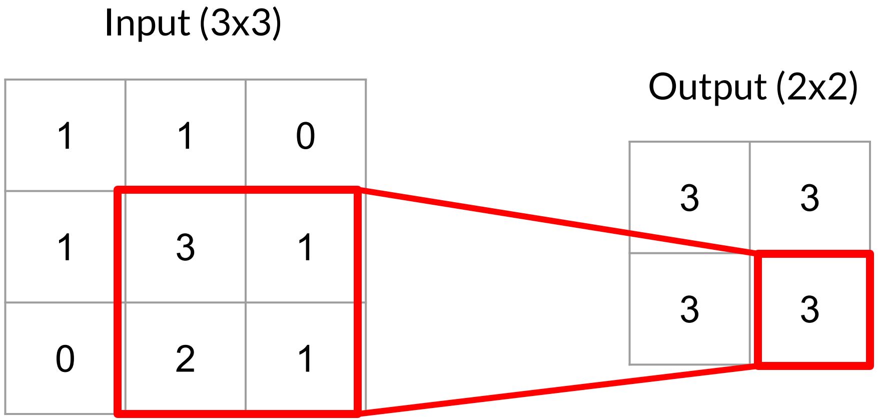
Input (3x3)

|   |   |   |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | 2 | 1 |

Output (2x2)

|   |   |
|---|---|
| 3 | 3 |
| 3 |   |

# Max-Pooling (2x2)



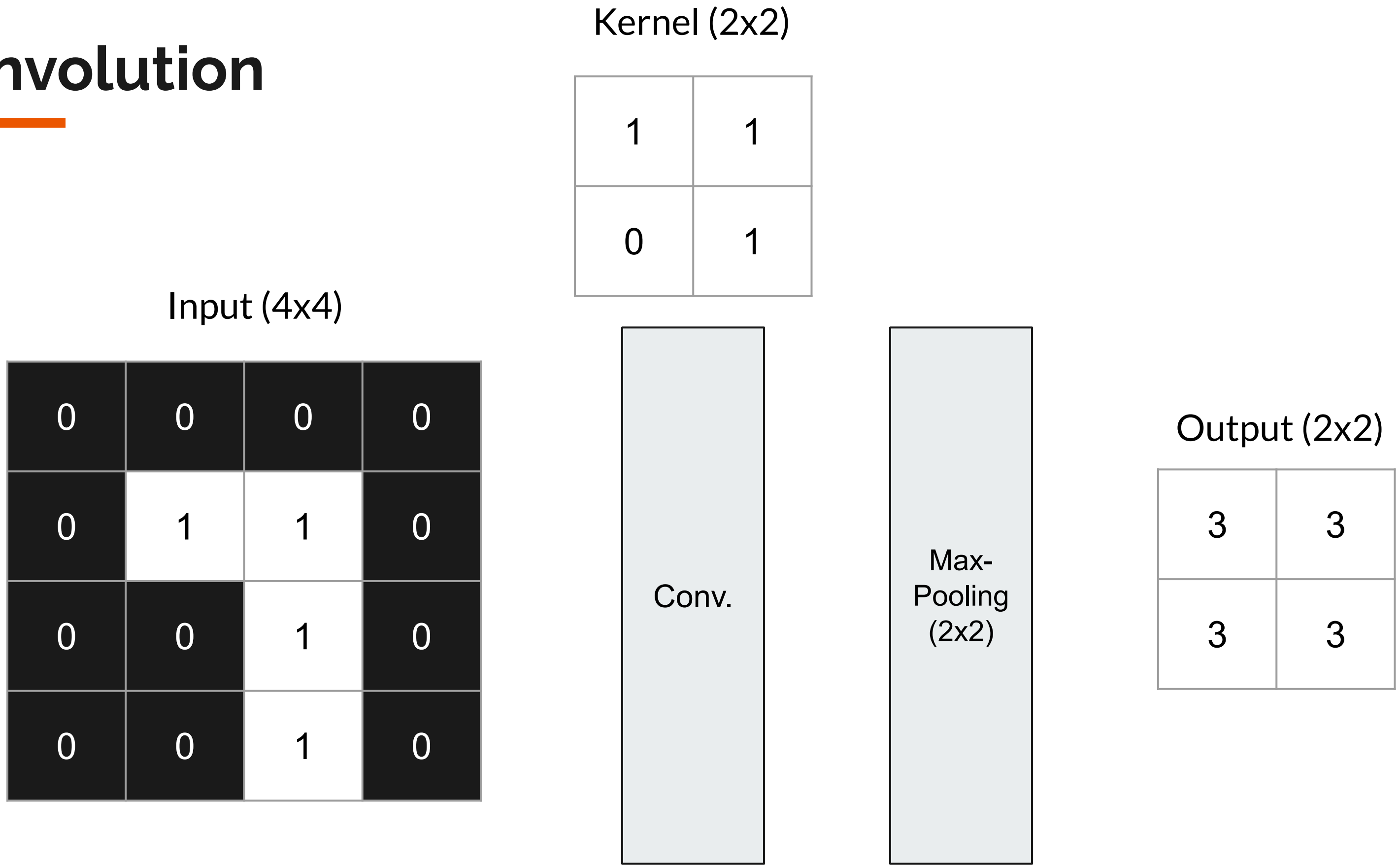
Input (3x3)

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 3 | 1 |
| 0 | 2 | 1 |

Output (2x2)

| | |
|---|---|
| 3 | 3 |
| 3 | 3 |

# Convolution



Kernel (2x2)

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

Input (4x4)

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

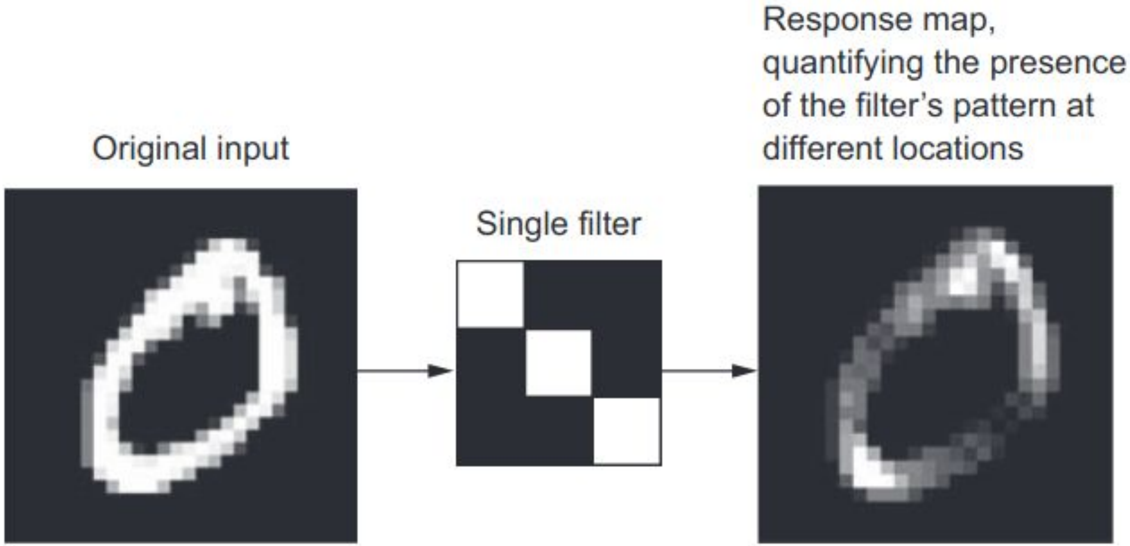
Conv.

Max-Pooling (2x2)

Output (2x2)

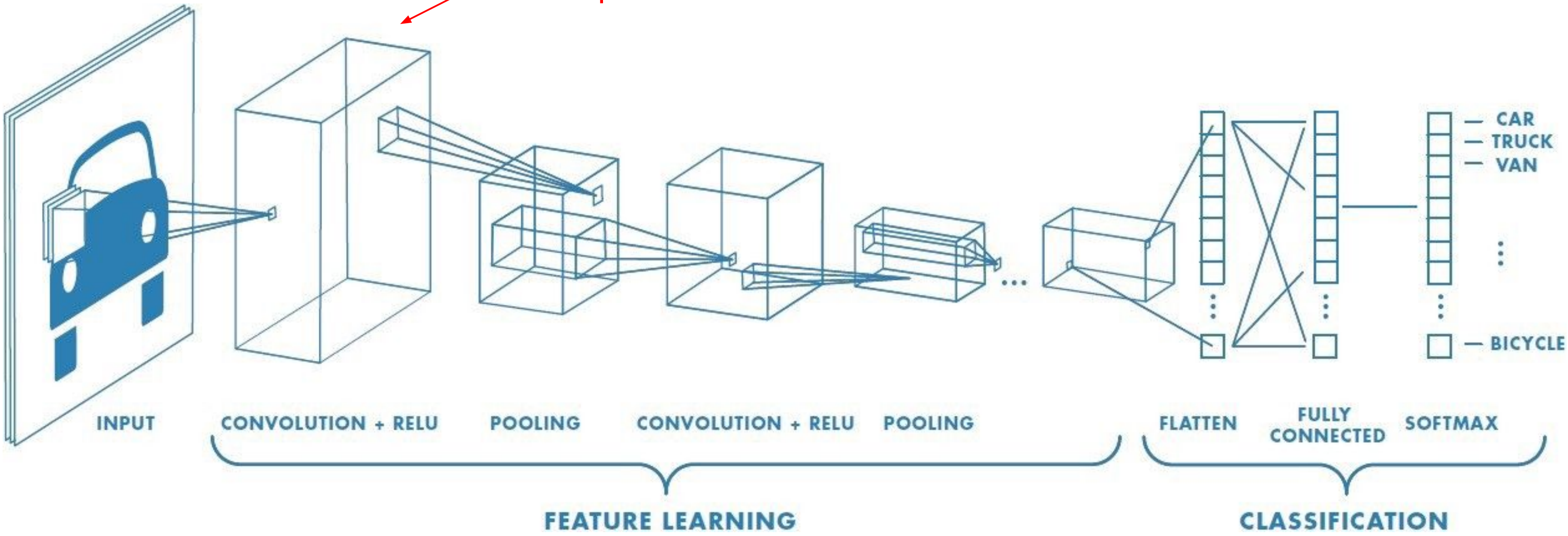
| | |
|---|---|
| 3 | 3 |
| 3 | 3 |

# Convolution

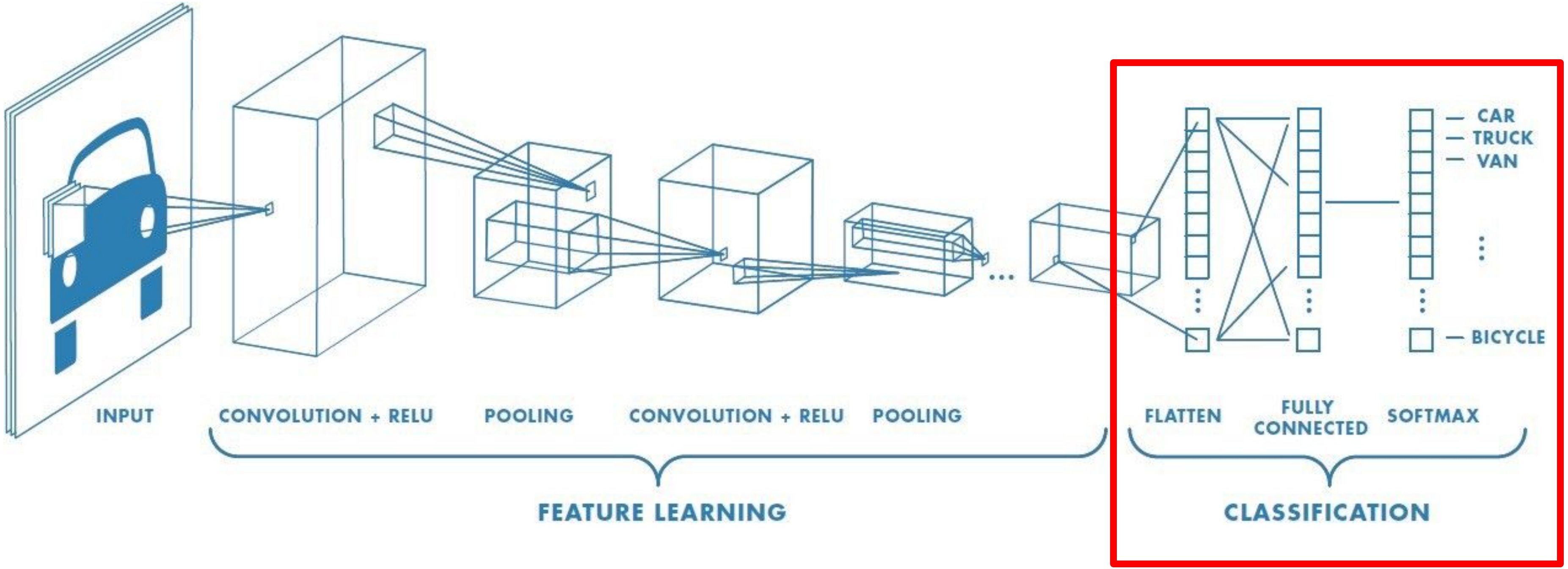

Original input
Single filter
Response map, quantifying the presence of the filter's pattern at different locations

# Convolutional Neural Networks (CNNs)

There can be multiple filters!



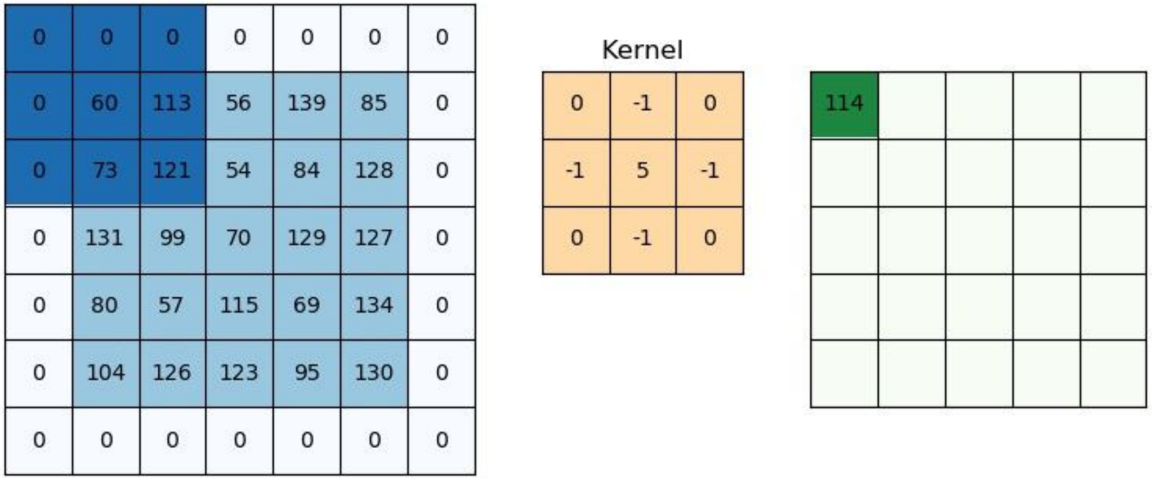
# Convolutional Neural Networks (CNNs)



# Convolution

https://adamharley.com/nn_vis/cnn/2d.html

# Padding



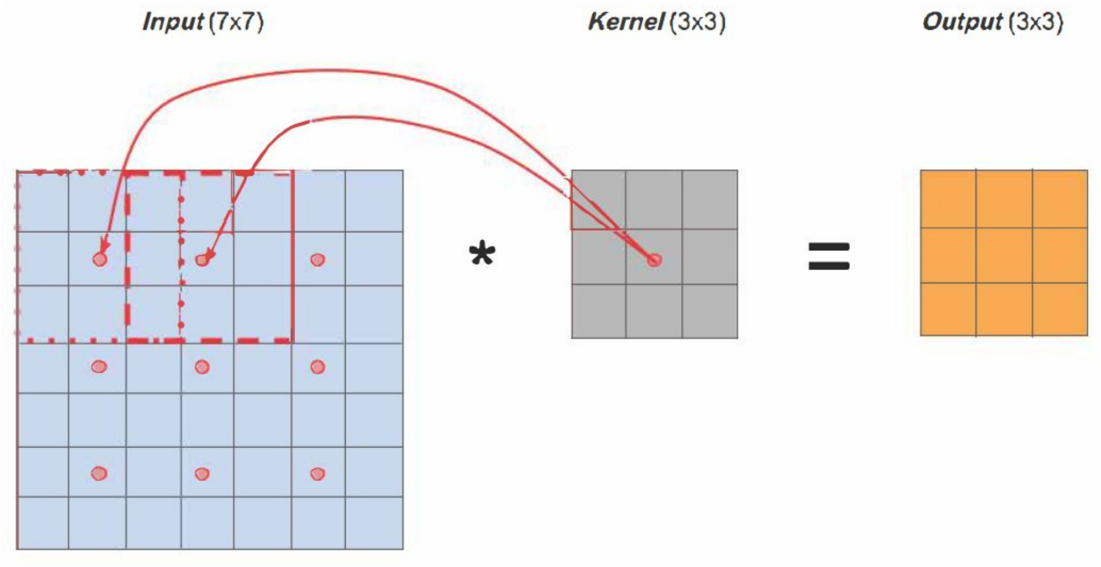
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 60 | 113 | 56 | 139 | 85 | 0 |
| 0 | 73 | 121 | 54 | 84 | 128 | 0 |
| 0 | 131 | 99 | 70 | 129 | 127 | 0 |
| 0 | 80 | 57 | 115 | 69 | 134 | 0 |
| 0 | 104 | 126 | 123 | 95 | 130 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Kernel

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

| | | | | |
|---|---|---|---|---|
| 114 | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

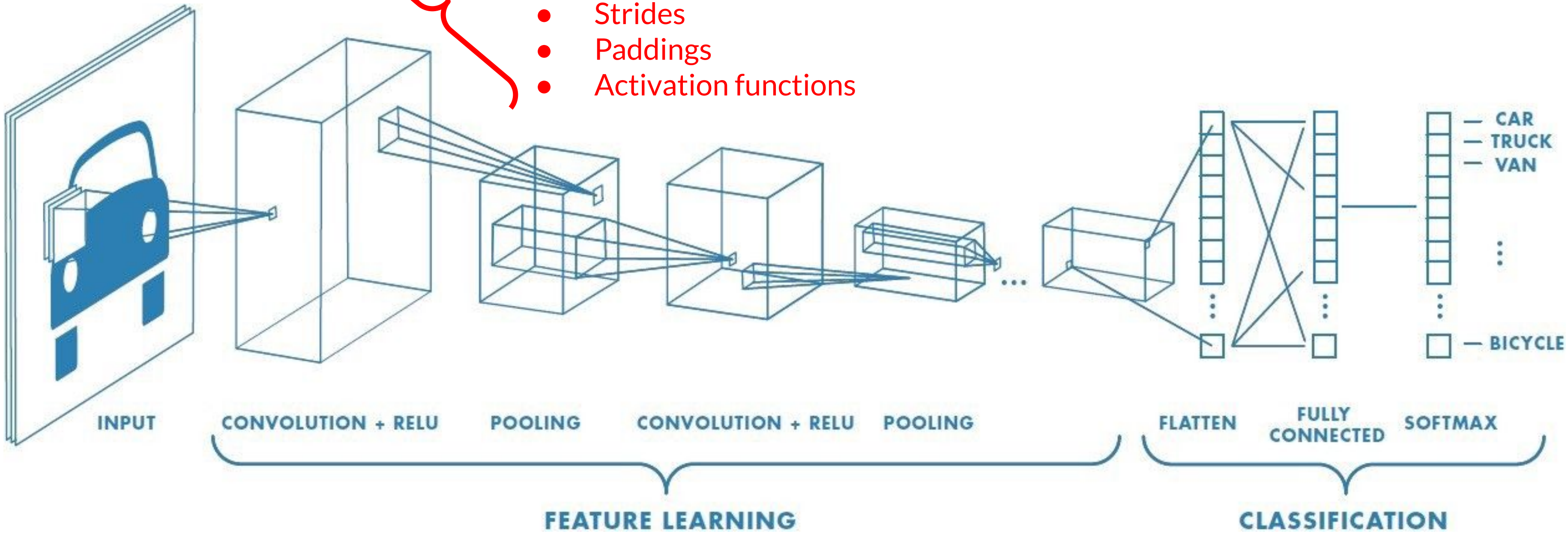
# Stride = 2



# Convolutional Neural Networks (CNNs)

**Convolution**

- Num. Kernels
- Kernel sizes
- Strides
- Paddings
- Activation functions

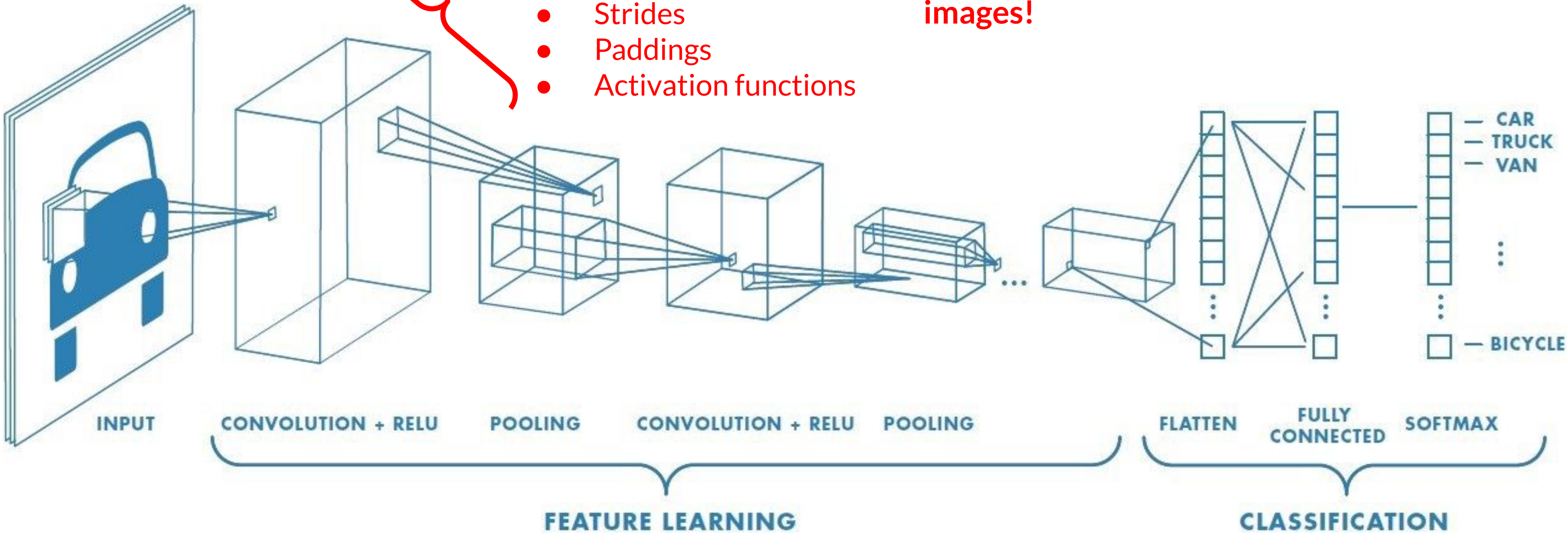


# Convolutional Neural Networks (CNNs)

**Convolution**

- Num. Kernels
- Kernel sizes
- Strides
- Paddings
- Activation functions

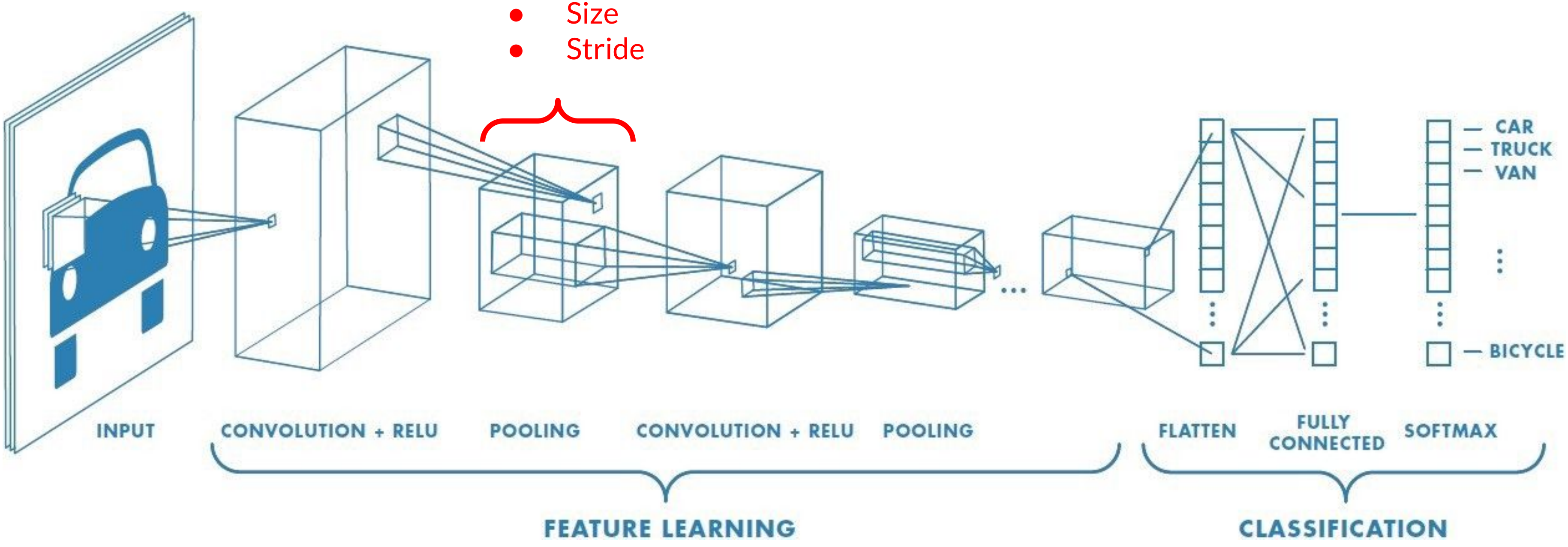
**NN learns best kernel values for predicting supervised images!**



# Convolutional Neural Networks (CNNs)

**Pooling**

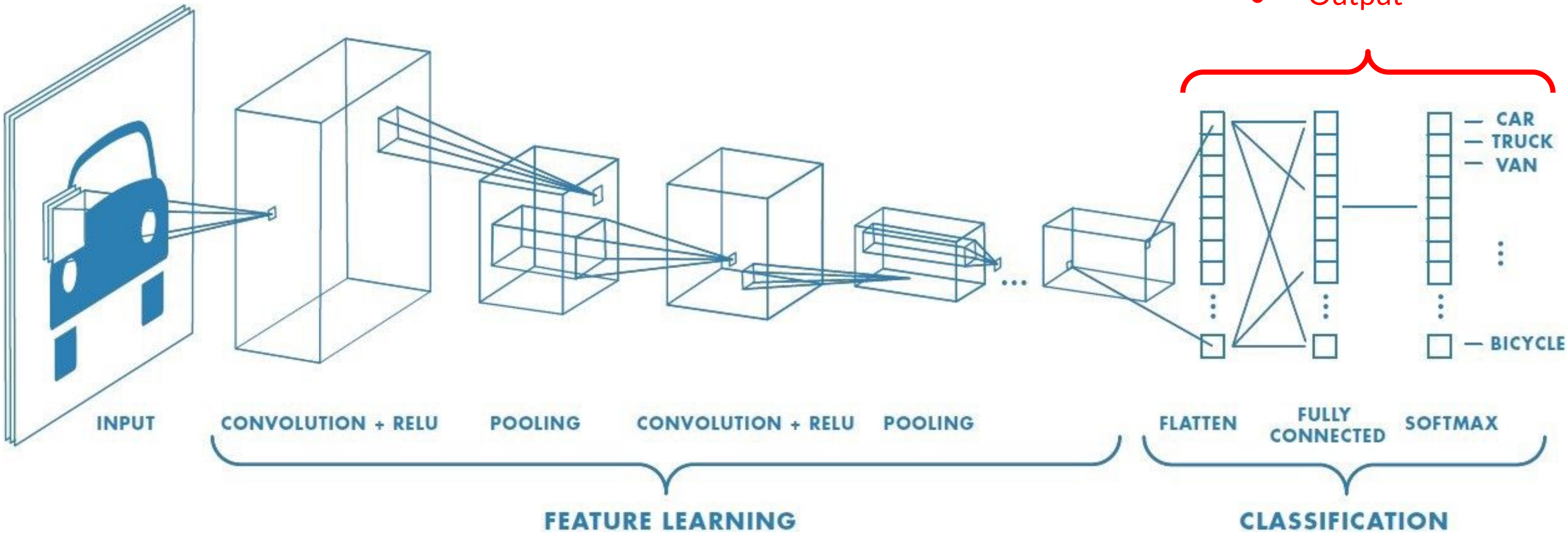
- Type
- Size
- Stride



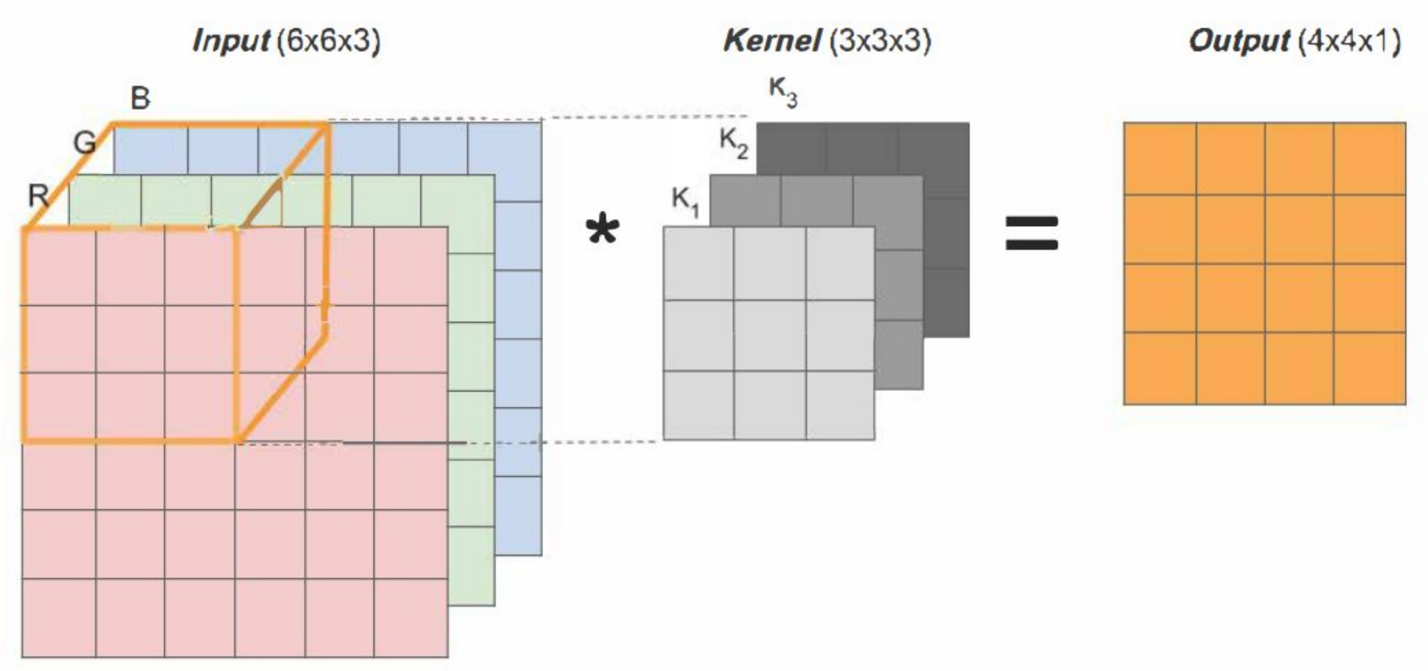
# Convolutional Neural Networks (CNNs)

**FC**

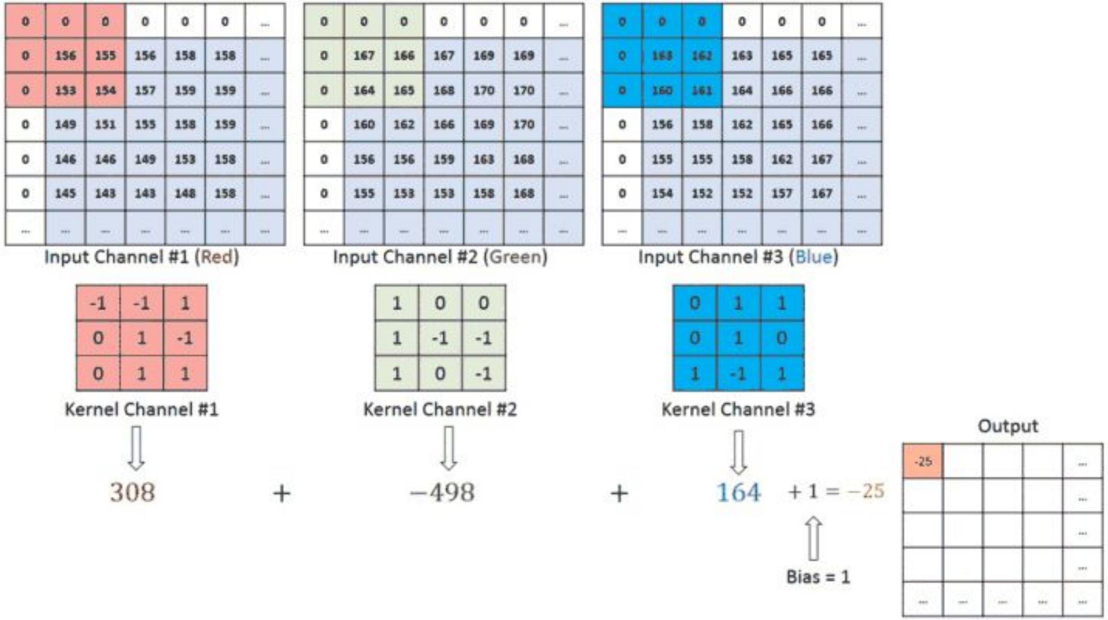
- Num. layers
- Num. neurons
- Activations
- Output



# RGB Convolution



# RGB Convolution



Input Channel #1 (Red)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 156 | 155 | 156 | 158 | 158 | ... |
| 0 | 153 | 154 | 157 | 159 | 159 | ... |
| 0 | 149 | 151 | 155 | 158 | 159 | ... |
| 0 | 146 | 146 | 149 | 153 | 158 | ... |
| 0 | 145 | 143 | 143 | 148 | 158 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Input Channel #2 (Green)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 167 | 166 | 167 | 169 | 169 | ... |
| 0 | 164 | 165 | 168 | 170 | 170 | ... |
| 0 | 160 | 162 | 166 | 169 | 170 | ... |
| 0 | 156 | 156 | 159 | 163 | 168 | ... |
| 0 | 155 | 153 | 153 | 158 | 168 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Input Channel #3 (Blue)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 163 | 162 | 163 | 165 | 165 | ... |
| 0 | 160 | 161 | 164 | 166 | 166 | ... |
| 0 | 156 | 158 | 162 | 165 | 166 | ... |
| 0 | 155 | 155 | 158 | 162 | 167 | ... |
| 0 | 154 | 152 | 152 | 157 | 167 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Kernel Channel #1

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

Kernel Channel #2

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

Kernel Channel #3

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

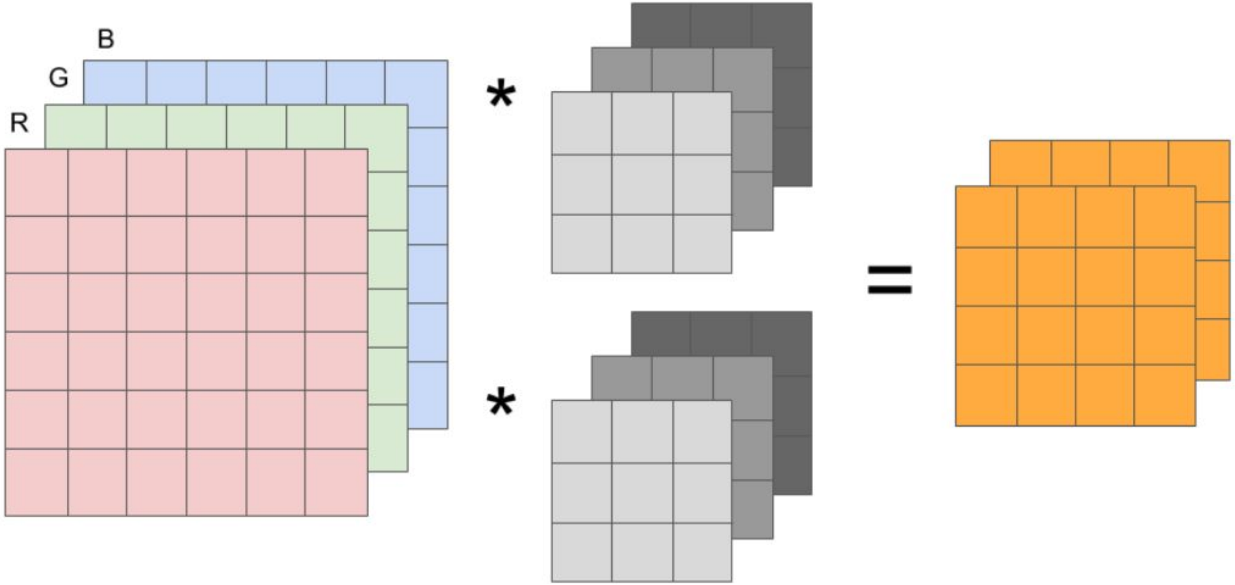
$$308 \quad + \quad -498 \quad + \quad 164 \quad + 1 = -25$$

Bias = 1

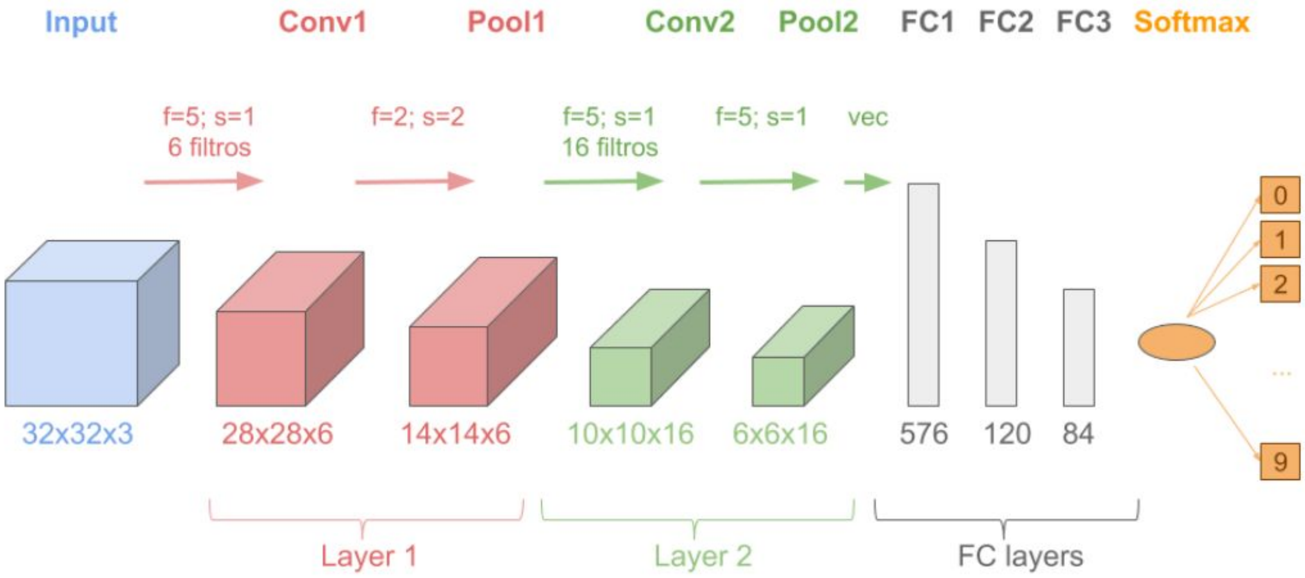
Output

| | | | | |
|---|---|---|---|---|
| -25 | | | | ... |
| | | | | ... |
| | | | | ... |
| | | | | ... |
| ... | ... | ... | ... | ... |

# RGB Convolution

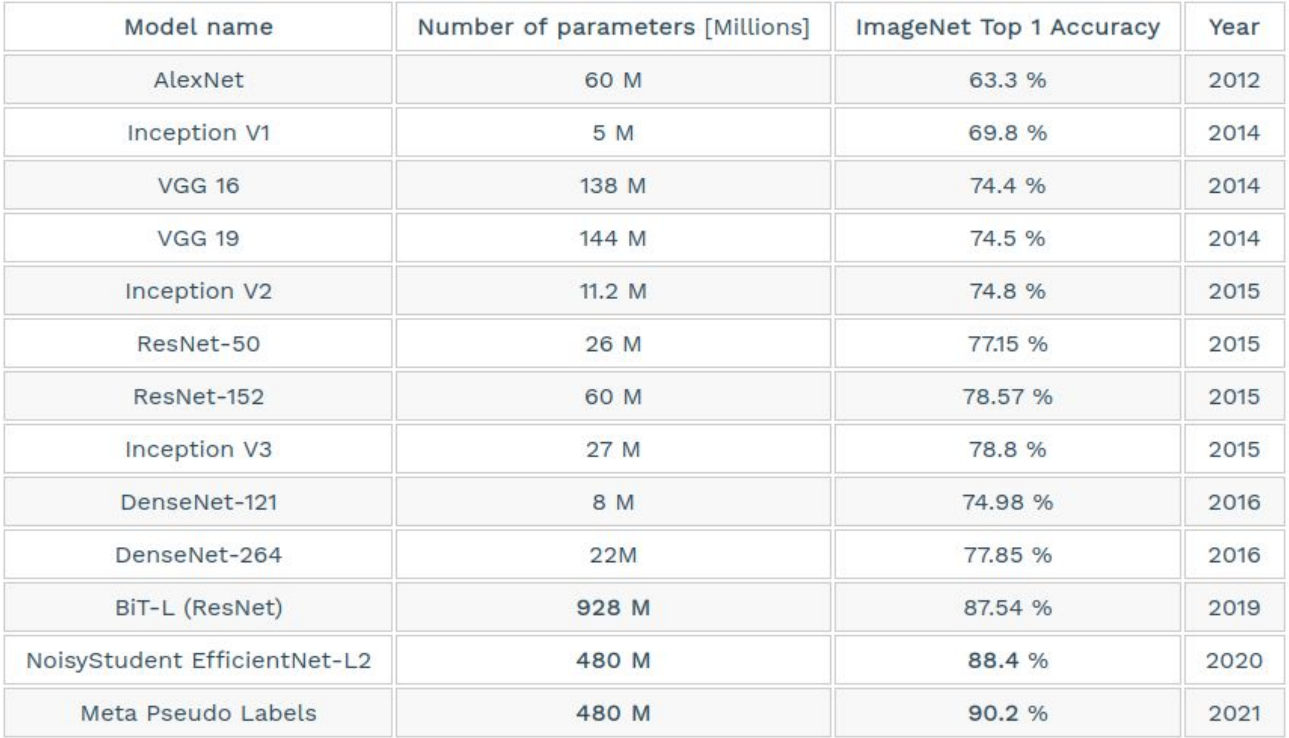


# Inspired in LeNet-5



Fuente: LeCun y col. (1989)

# Famous CNN architectures



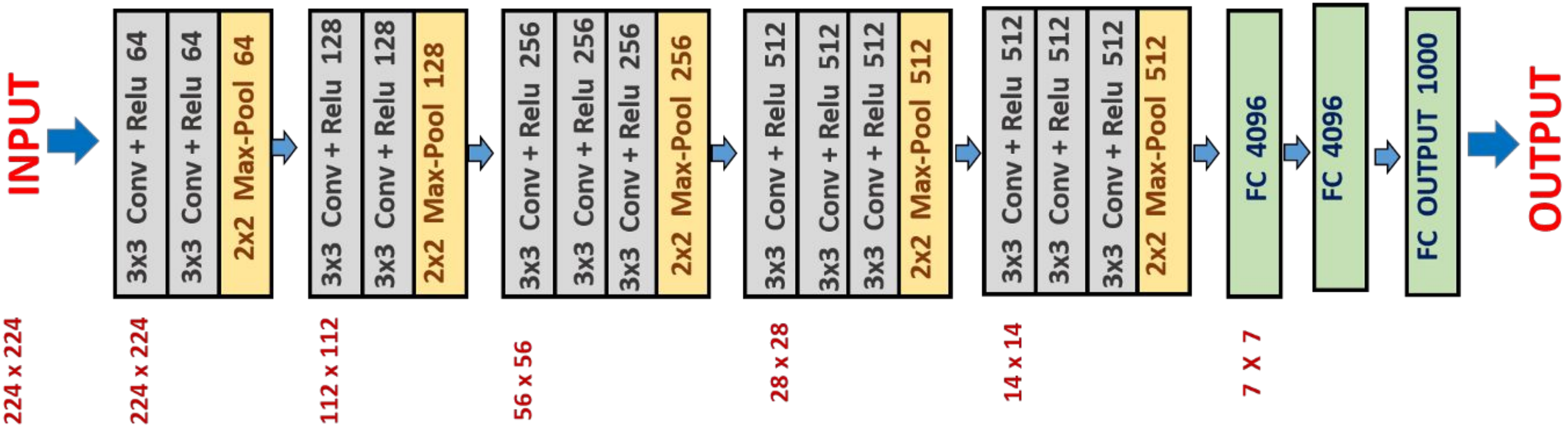
| Model name | Number of parameters [Millions] | ImageNet Top 1 Accuracy | Year |
|---|---|---|---|
| AlexNet | 60 M | 63.3 % | 2012 |
| Inception V1 | 5 M | 69.8 % | 2014 |
| VGG 16 | 138 M | 74.4 % | 2014 |
| VGG 19 | 144 M | 74.5 % | 2014 |
| Inception V2 | 11.2 M | 74.8 % | 2015 |
| ResNet-50 | 26 M | 77.15 % | 2015 |
| ResNet-152 | 60 M | 78.57 % | 2015 |
| Inception V3 | 27 M | 78.8 % | 2015 |
| DenseNet-121 | 8 M | 74.98 % | 2016 |
| DenseNet-264 | 22M | 77.85 % | 2016 |
| BiT-L (ResNet) | 928 M | 87.54 % | 2019 |
| NoisyStudent EfficientNet-L2 | 480 M | 88.4 % | 2020 |
| Meta Pseudo Labels | 480 M | 90.2 % | 2021 |

# Famous CNN architectures

https://theaisummer.com/cnn-architectures/

https://keras.io/api/applications/

# VGG-16



INPUT

224 x 224

3x3 Conv + Relu 64

3x3 Conv + Relu 64

2x2 Max-Pool 64

224 x 224

3x3 Conv + Relu 128

3x3 Conv + Relu 128

2x2 Max-Pool 128

112 x 112

3x3 Conv + Relu 256

3x3 Conv + Relu 256

3x3 Conv + Relu 256

2x2 Max-Pool 256

56 x 56

3x3 Conv + Relu 512

3x3 Conv + Relu 512

3x3 Conv + Relu 512

2x2 Max-Pool 512

28 x 28

3x3 Conv + Relu 512

3x3 Conv + Relu 512

3x3 Conv + Relu 512

2x2 Max-Pool 512

14 x 14

FC 4096

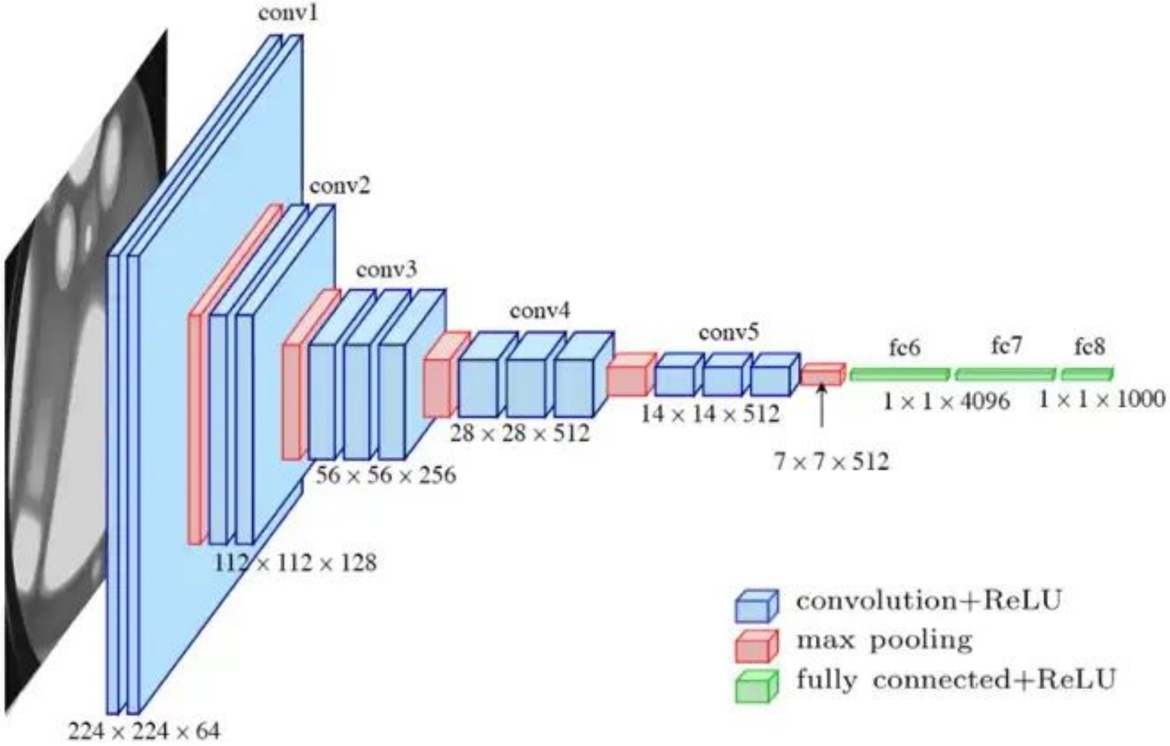
7 X 7

FC 4096

FC OUTPUT 1000

OUTPUT

# VGG-16


conv1
conv2
conv3
conv4
conv5
fc6
fc7
fc8
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

# Building CNNs – Recommendations

- Number of kernels
    - Small number in the first layers and increase it as you go deeper.
        - Early layers capture simple patterns and then you combine these in deeper layers.
    - Starting with 32 or 64 filters is common.

# Building CNNs – Recommendations

- Kernel sizes
  - 3x3 and 5x5 are common options.
  - Smaller are usually preferred for capturing more specific details.
  - Sometimes, large kernels (e.g. 7x7) in the first layers for capturing general patterns.

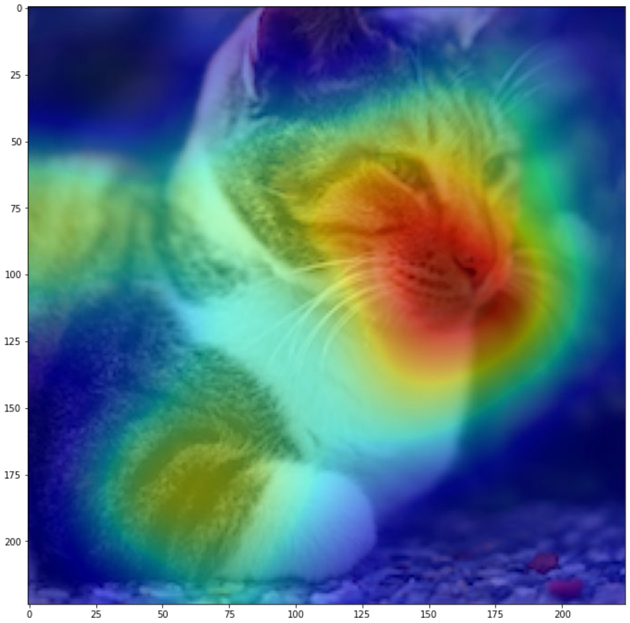
# Building CNNs – Recommendations

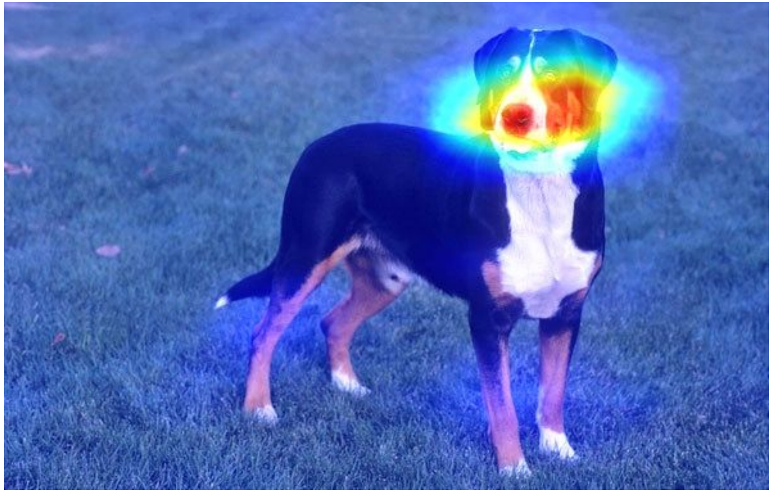
- Pooling layers
  - Generally used for reducing dimensions.
  - Using too many can imply losing valuable information.
  - 2x2 is very common, with greater poolings you risk losing valuable information.

# Building CNNs – Recommendations

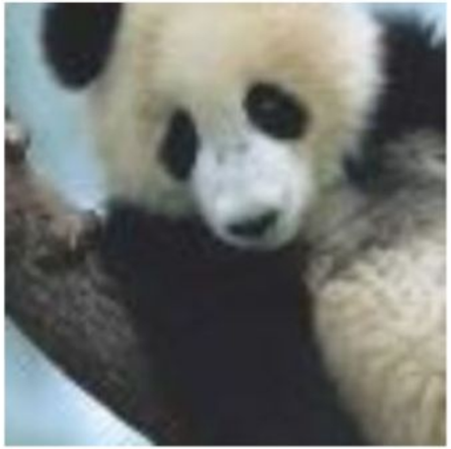
- Stride
  - A stride of 1 is common in convolutional layers.
  - A stride of 2 is common in pooling layers for reducing the dimensions of the image.

# Interpretability



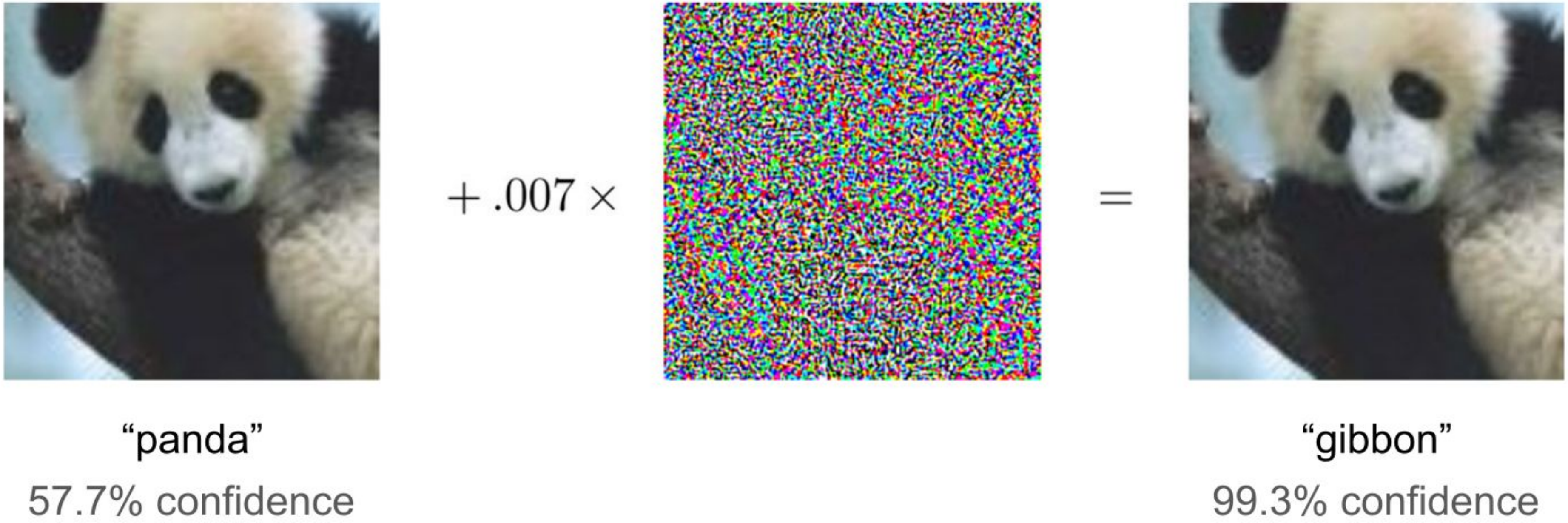


## Misleading classification

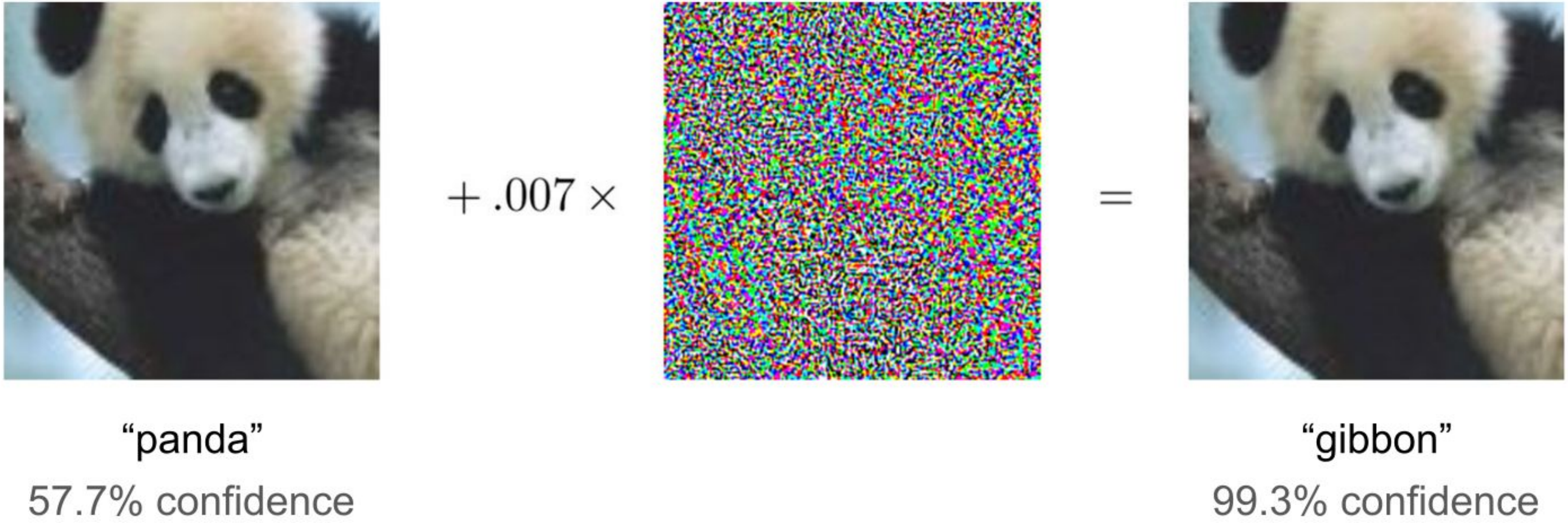


“panda”
57.7% confidence

# Misleading classification



# Misleading classification



# Resources

- https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks