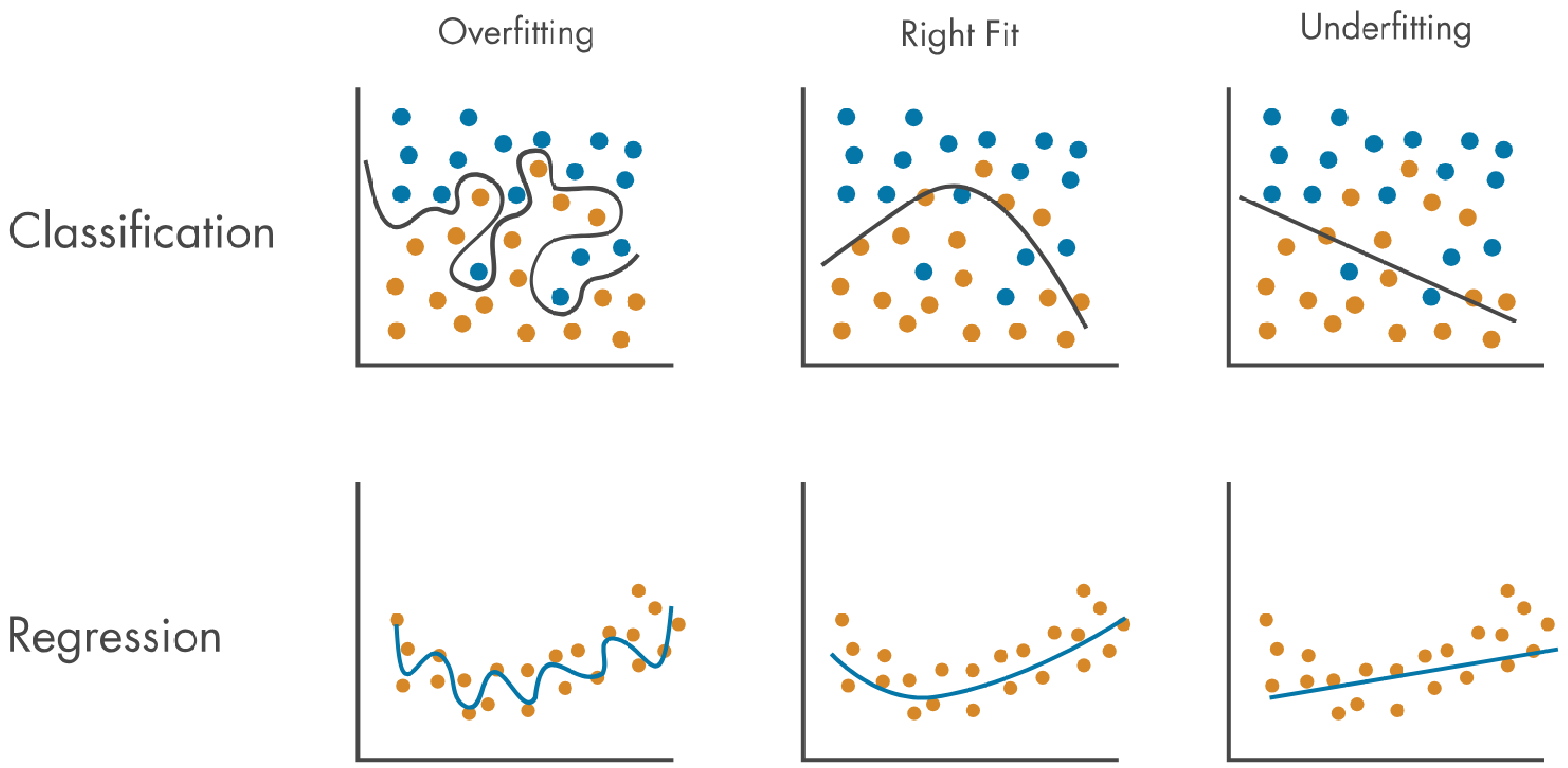
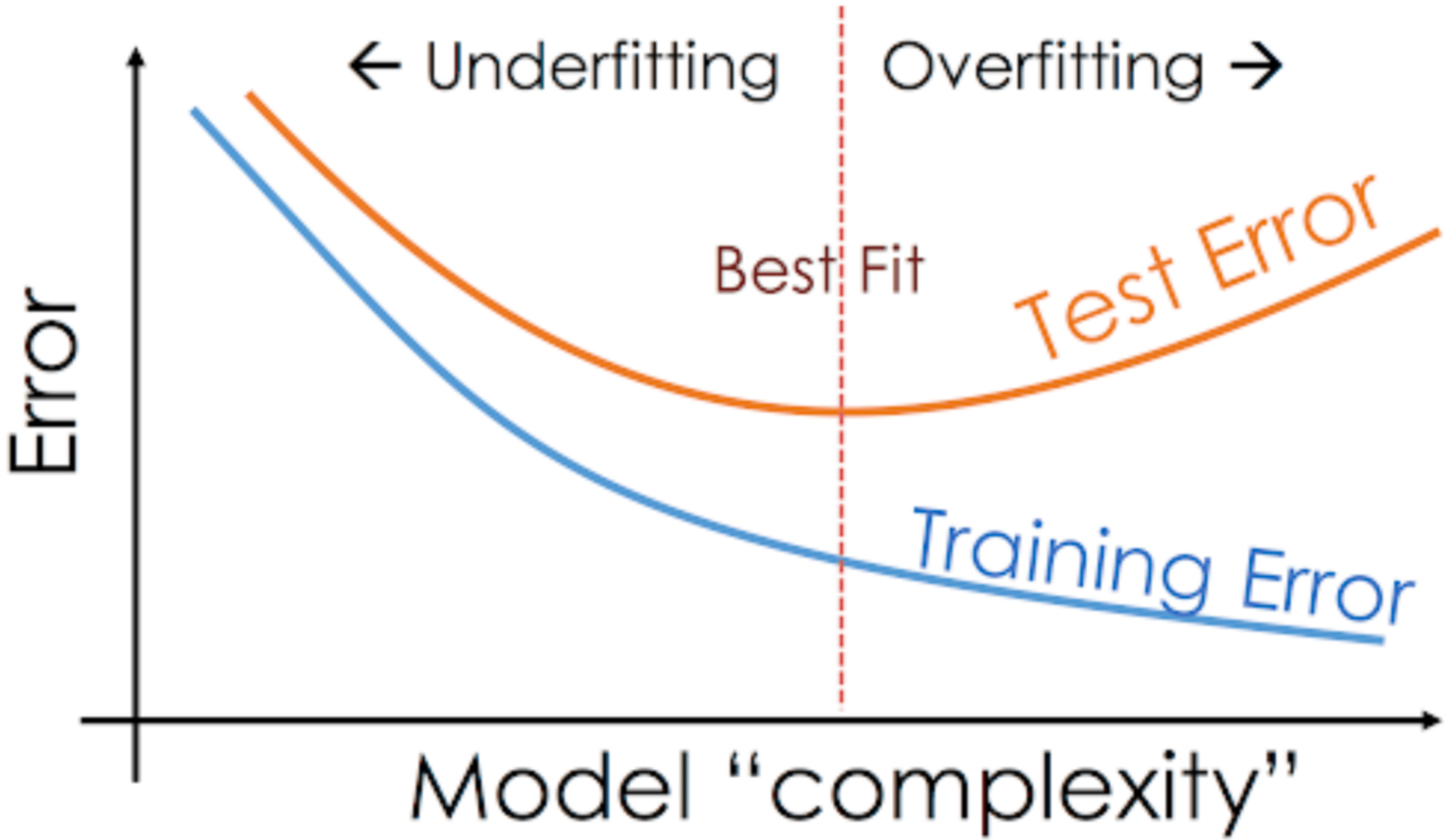
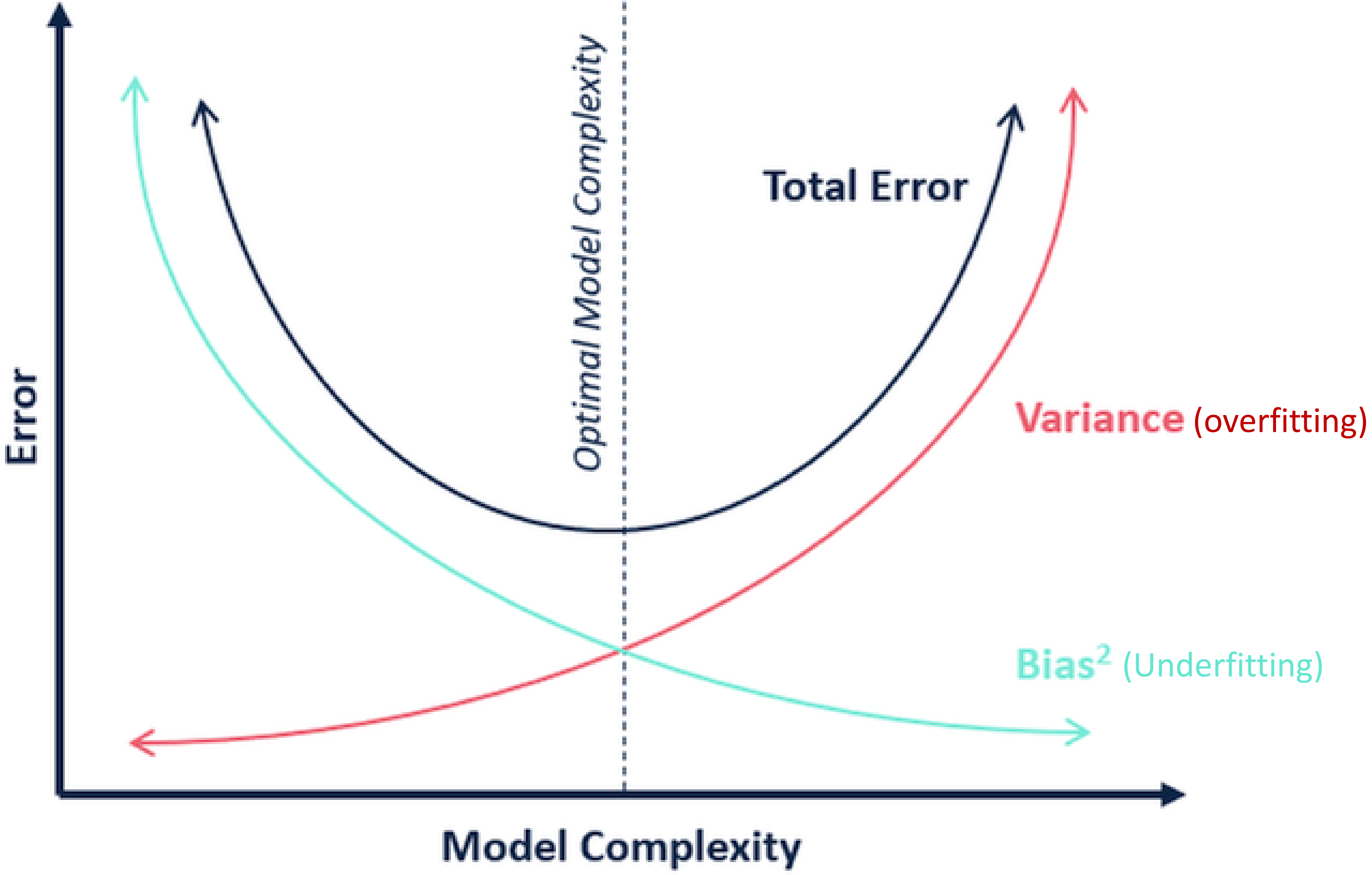
# Computational Intelligence

Professor: Dr. Mohammad Zare
Teaching Assistant: Ali Kohan

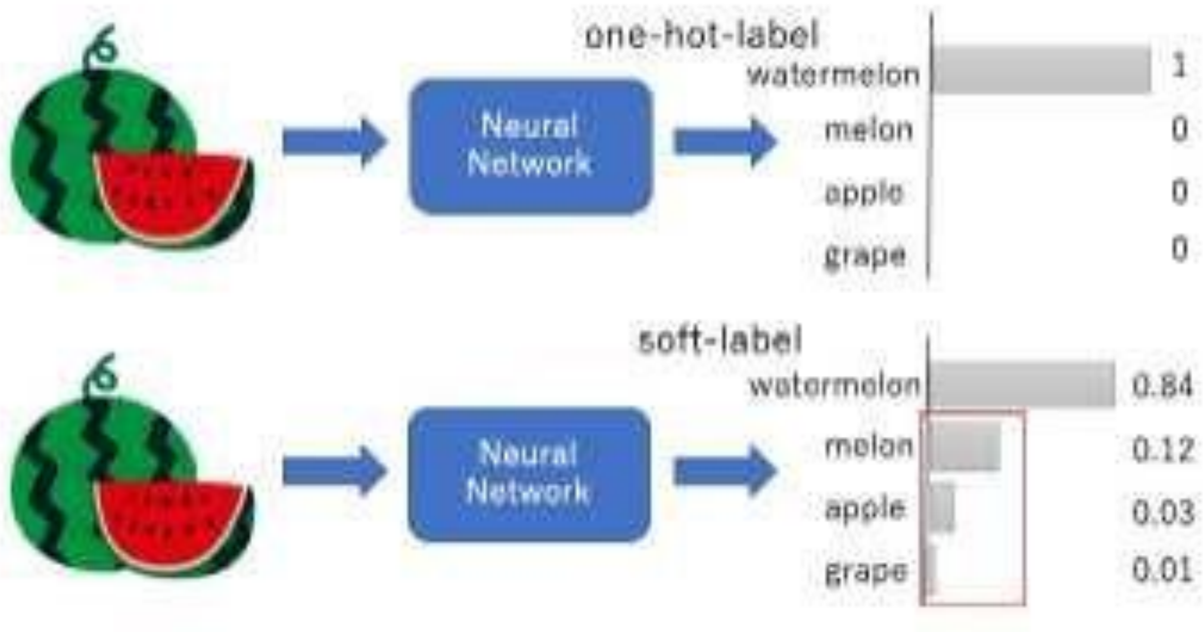

Overfitting
Right Fit
Underfitting
Classification
Regression

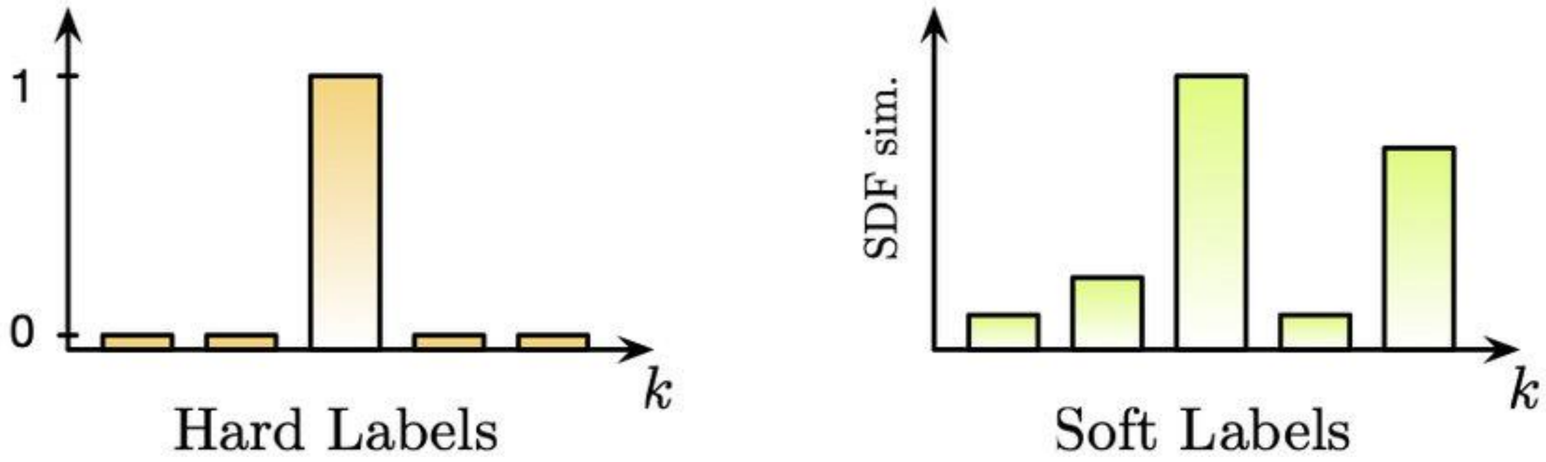

← Underfitting
Overfitting →
Best Fit
Test Error
Training Error
Error
Model "complexity"



Error
Optimal Model Complexity
Total Error
Variance (overfitting)
Bias² (Underfitting)
Model Complexity

# Soft label vs hard label



# Other types of data

The activation function of a node in an artificial neural network is a function that calculates the output of the node based on its individual inputs and their weights.

image

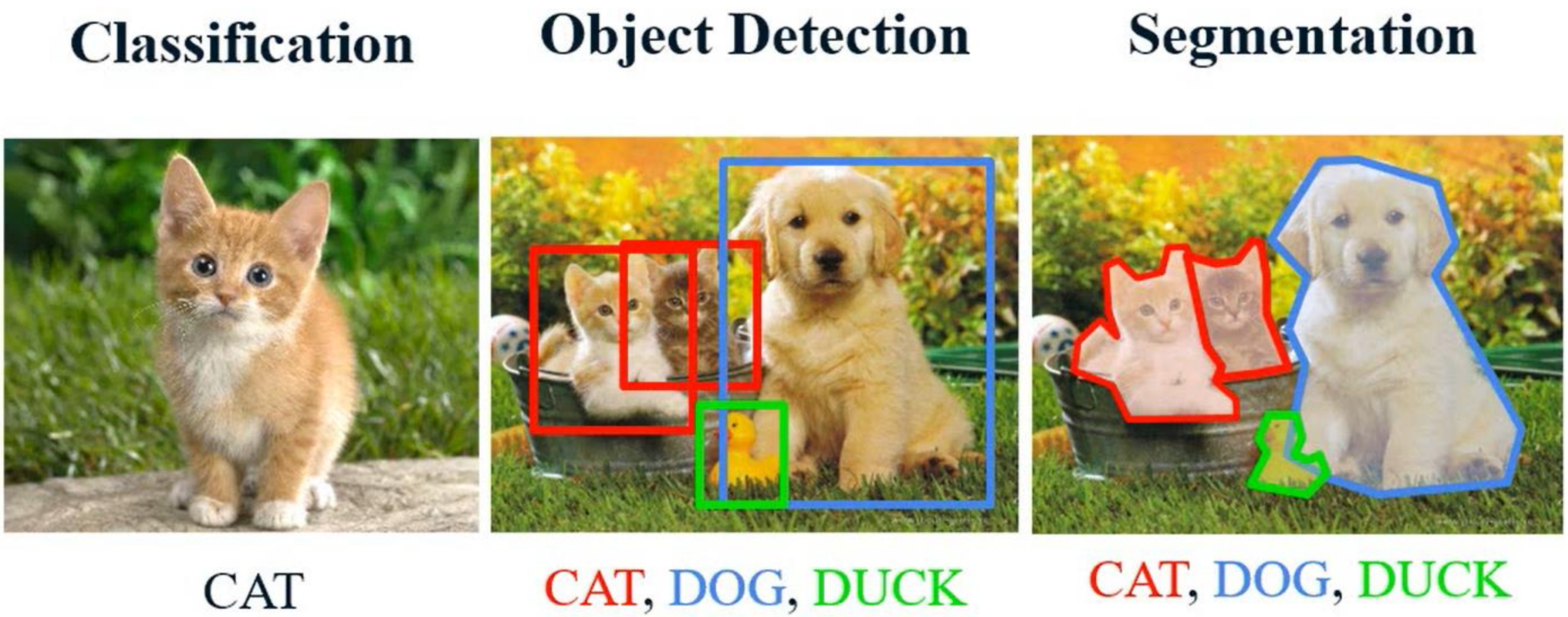
sound

natural language

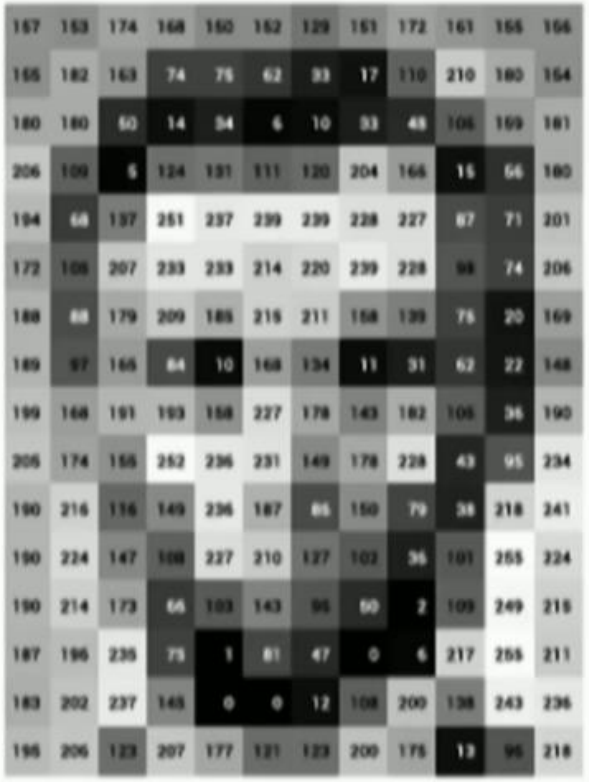
→ CNN

→ RNN

# What can we do with images?



# Images are Numbers

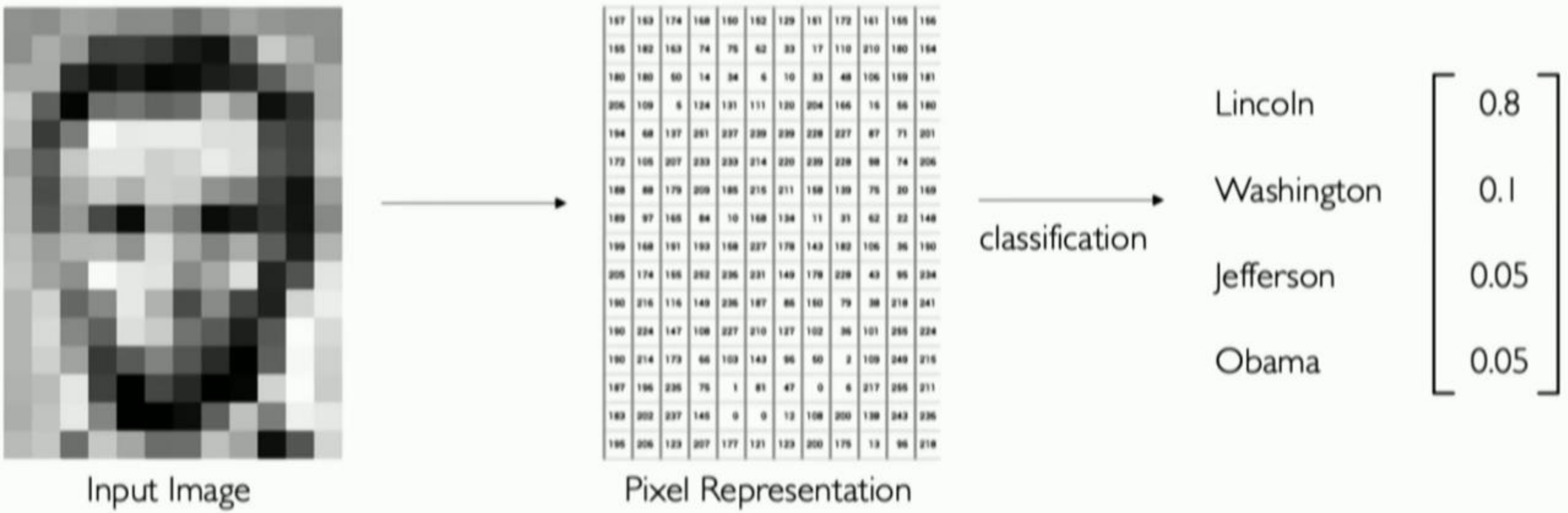


What the computer sees

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 195 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

# Tasks in Computer Vision




- **Regression**: output variable takes continuous value
- **Classification**: output variable takes class label. Can produce probability of belonging to a particular class

# The problem with using fully connected neural networks for computer vision

| | | | | | |
|---|---|---|---|---|---|
| 126<br>251<br>12 | 123<br>210<br>86 | … | … | | |
| … | … | … | … | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

**6 * 6 * 3 = 108**

**But in real images (e.g. 1MP):**
**1000 * 1000 * 3 * 10000 = 30b**
**Just in first layer!**

# The problem with using fully connected neural networks for computer vision

1. Computational Cost: fully connected networks (FCNs) have a large number of parameters and require a significant amount of computation, making them inefficient for large images and computationally expensive.

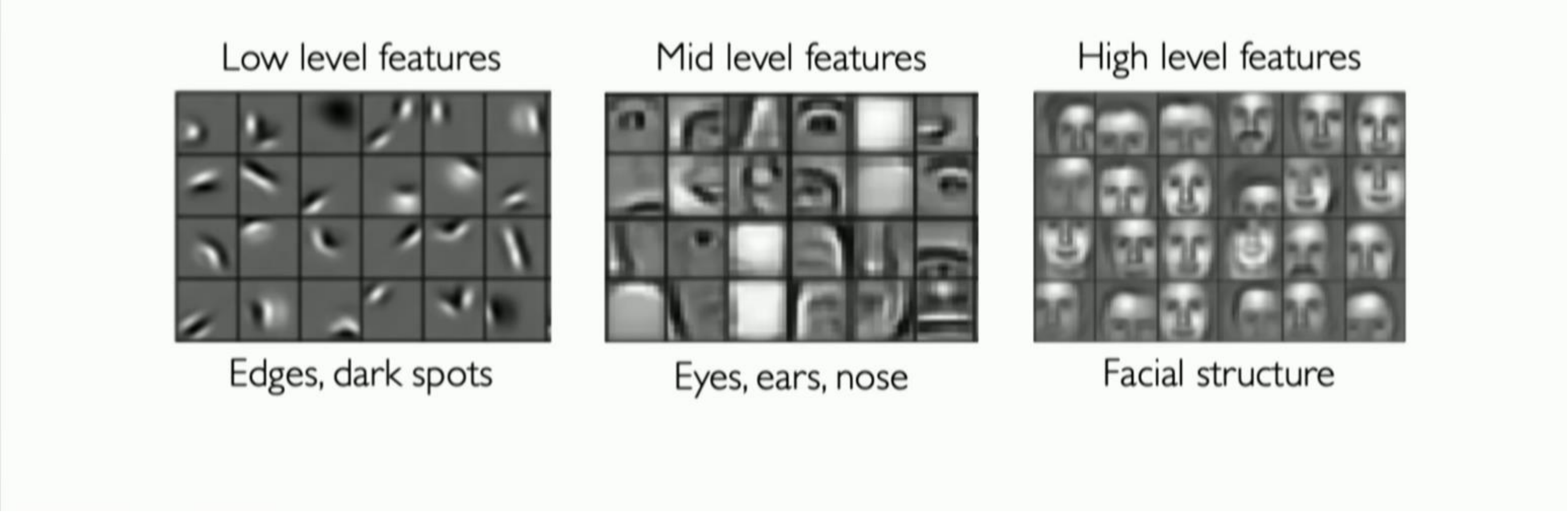
2. Need for Large Amounts of Data: FCNs require a large amount of data to train effectively, which can be a significant limitation in computer vision where data collection and labeling can be time-consuming and expensive.

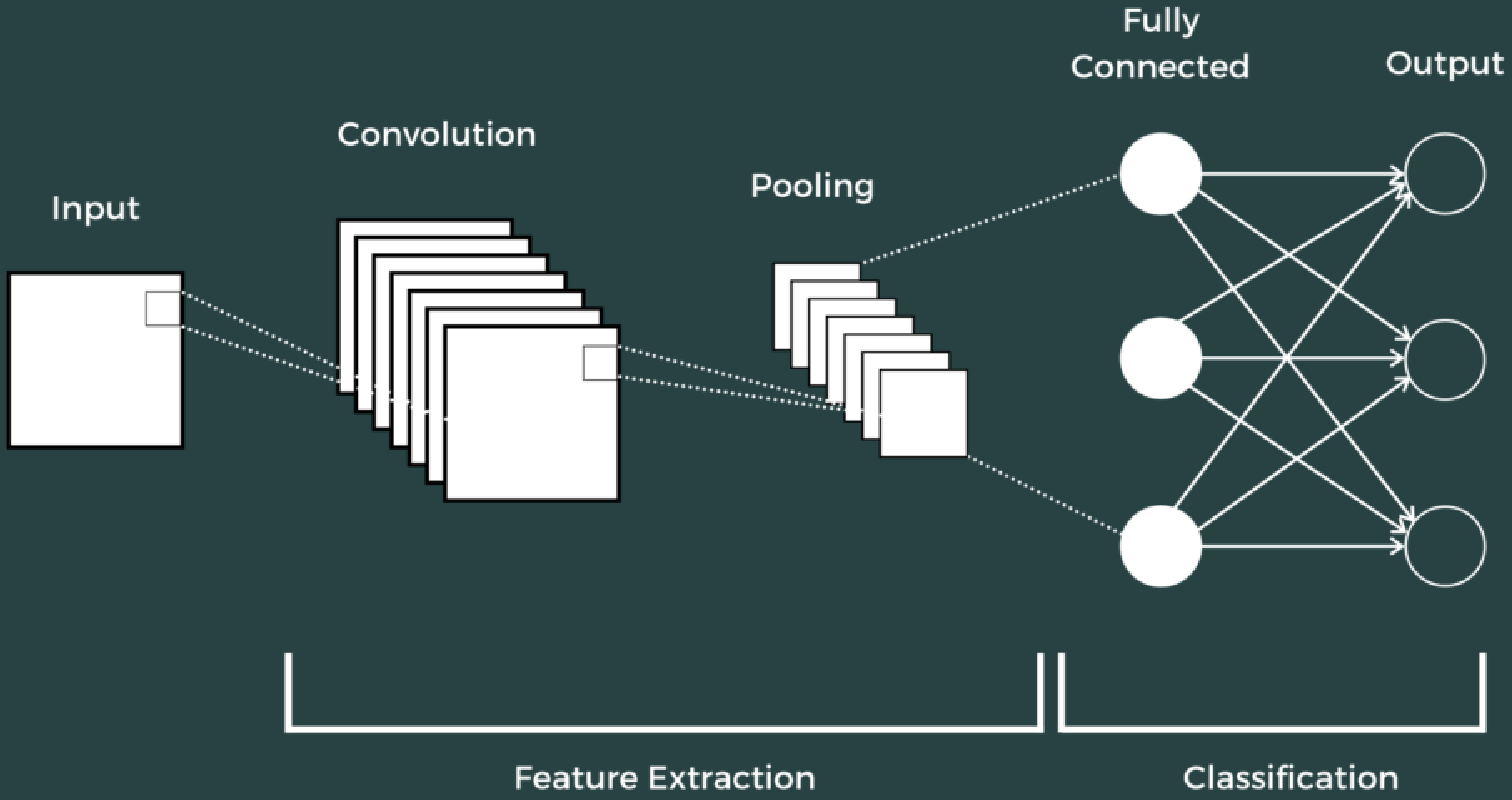
3. Lack of Spatial Hierarchical Representations: FCNs treat images as 1D vectors, ignoring the 2D structure of images. This makes it difficult for them to capture spatial hierarchies and relationships between pixels, which are essential for computer vision tasks.

4. Translation Invariance: FCNs are not translation invariant, meaning that they are sensitive to the location of objects in the image. This can lead to poor performance when the object is moved to a different location.

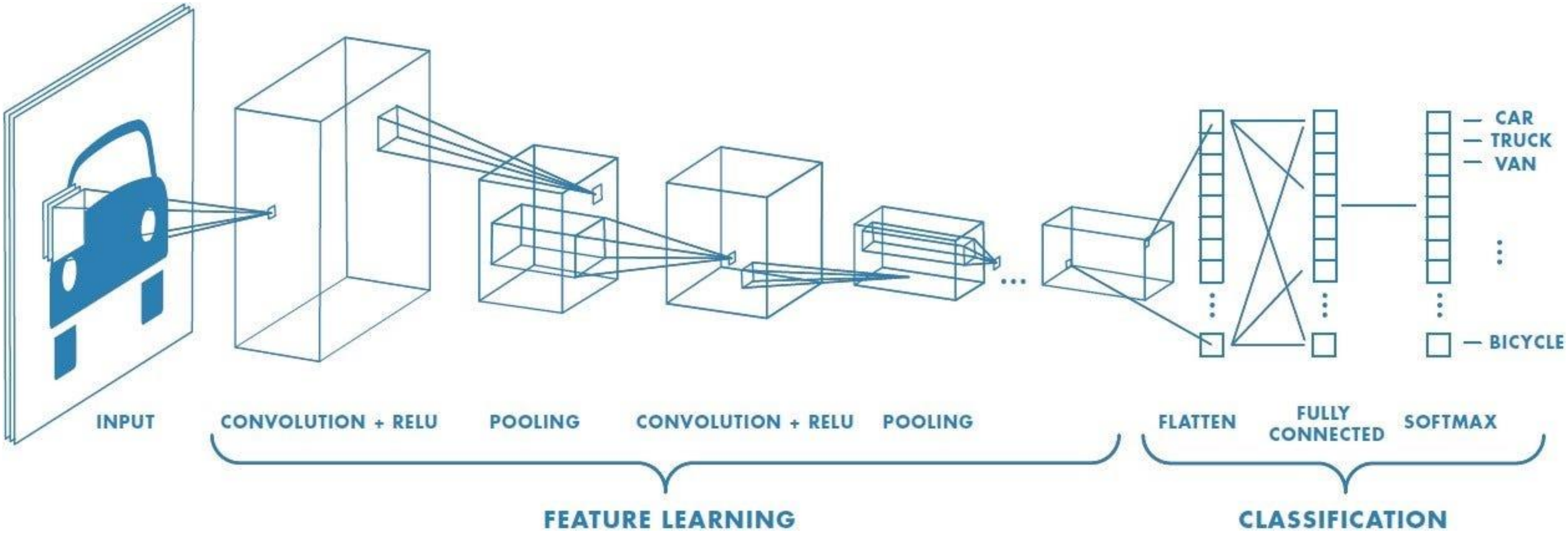
5. Overfitting: FCNs are prone to overfitting, especially when dealing with small datasets. They can memorize the training data rather than learning generalizable features, which can lead to poor performance on new, unseen data.

# Feature hierarchy

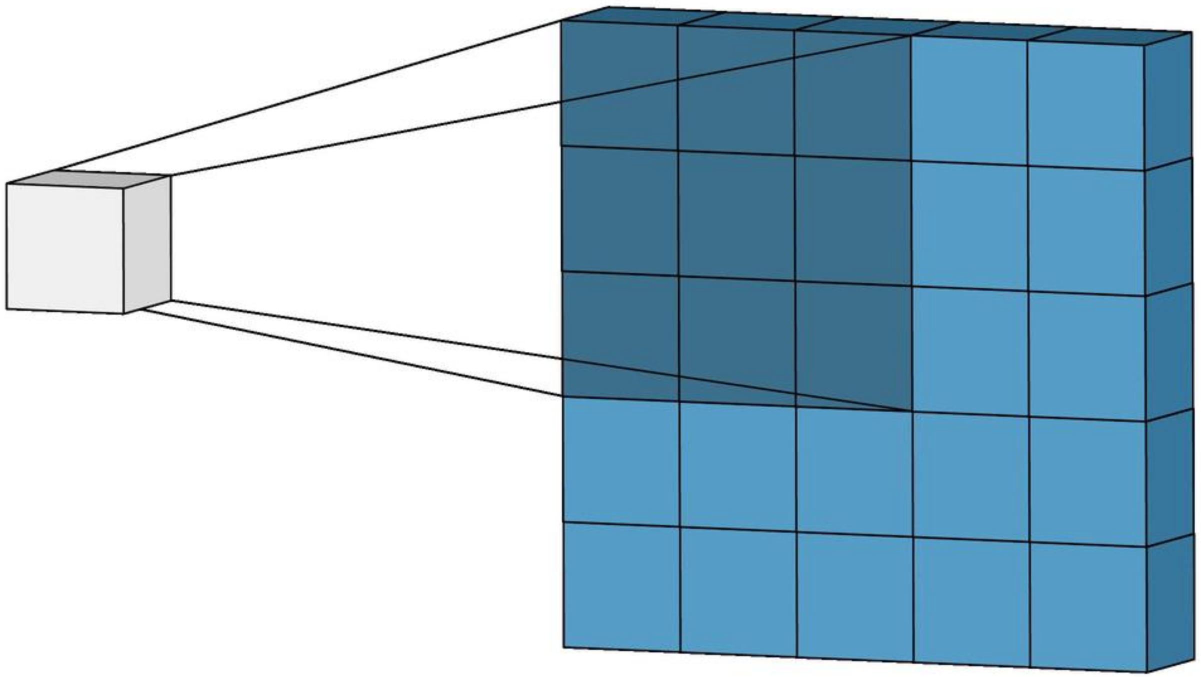



Input
Convolution
Pooling
Fully
Connected
Output
Feature Extraction
Classification

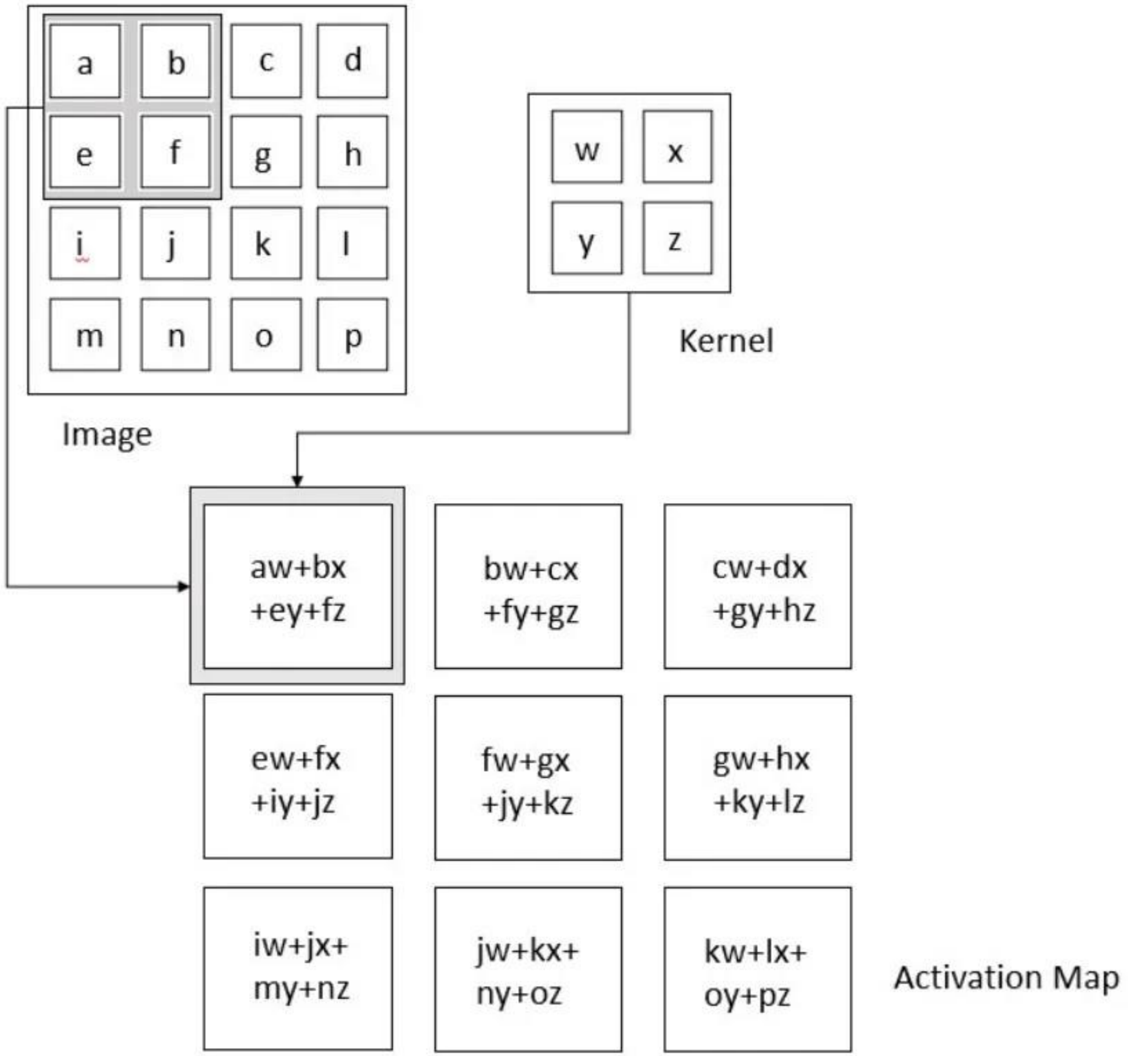
# CNN (overview)

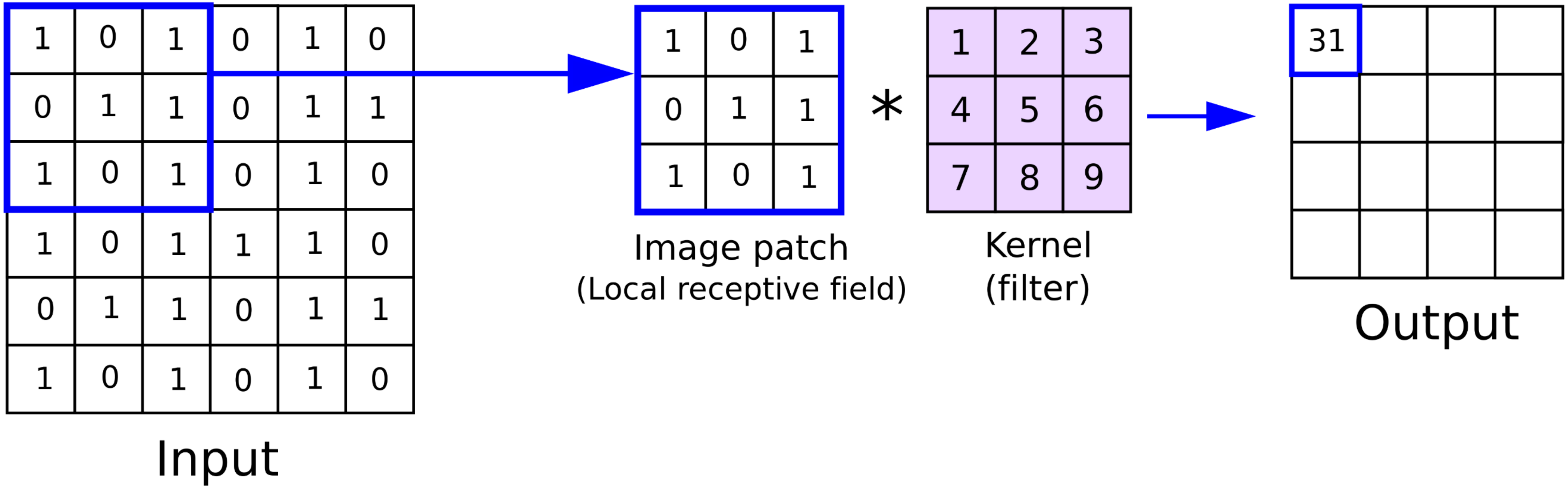


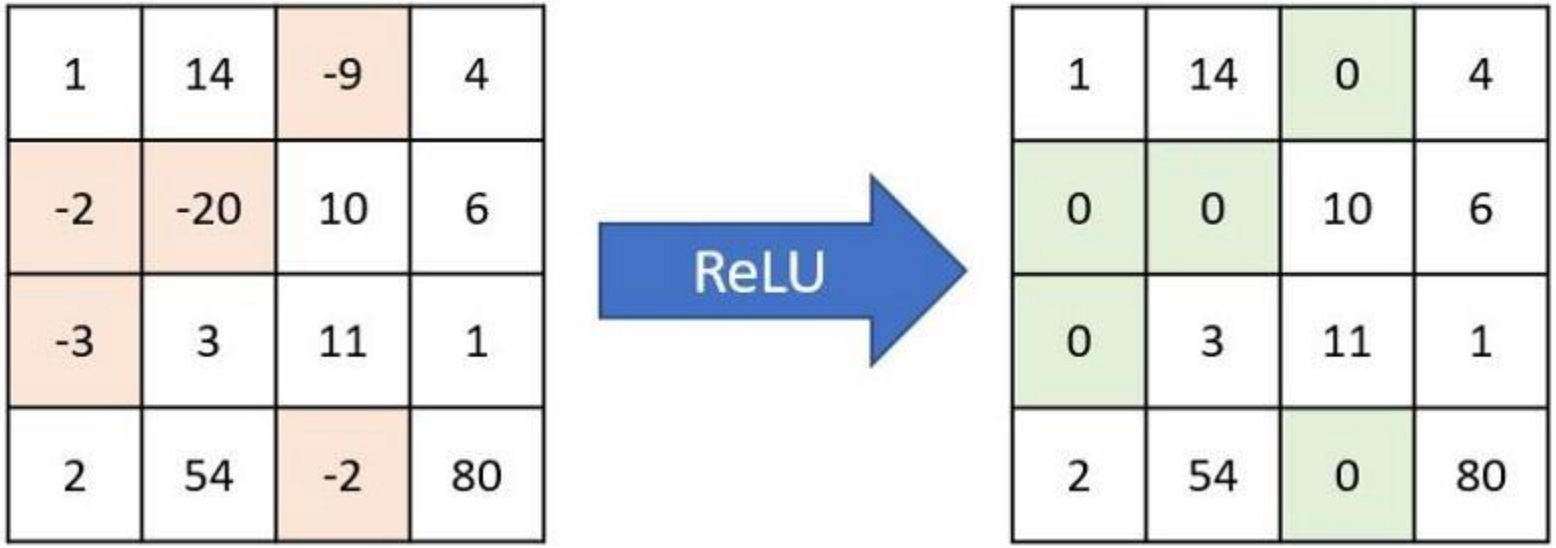
# Illustration of Convolution Operation

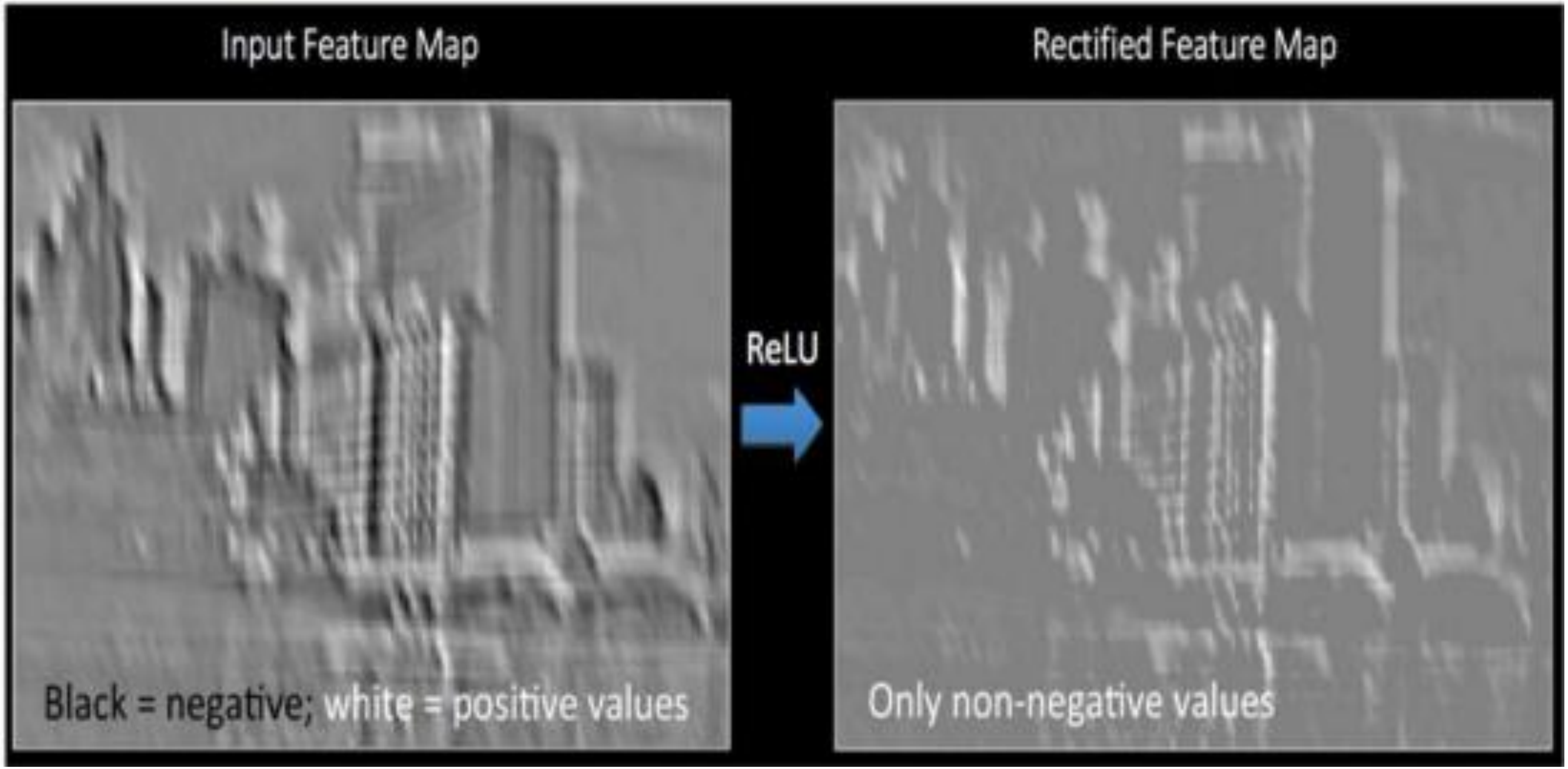


# Convolution Operation

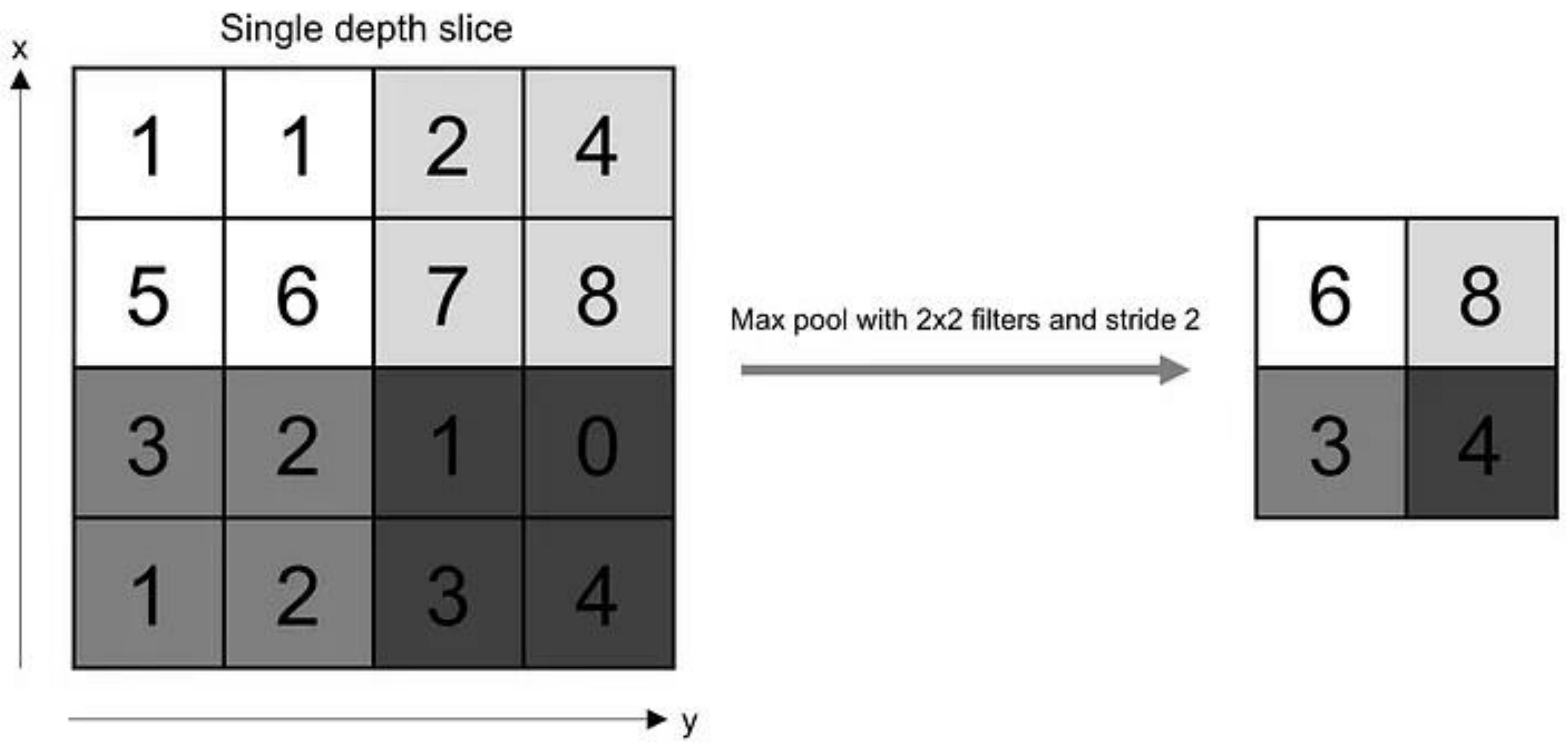


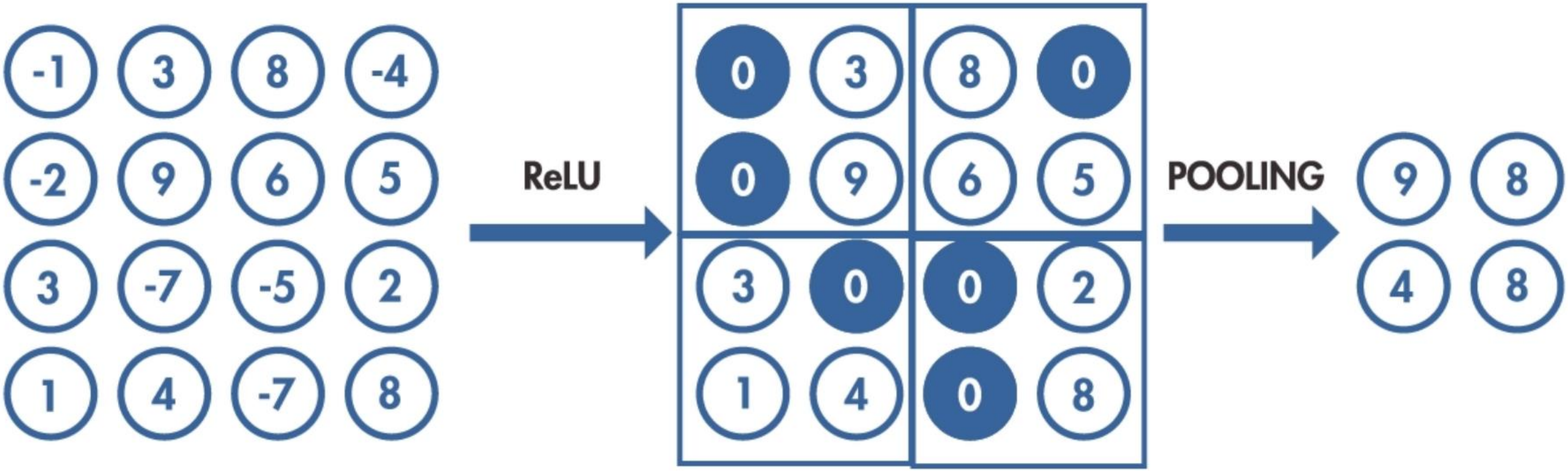

1 0 1 0 1 0
0 1 1 0 1 1
1 0 1 0 1 0
1 0 1 1 1 0
0 1 1 0 1 1
1 0 1 0 1 0
Input
1 0 1
0 1 1
1 0 1
Image patch
(Local receptive field)
*
1 2 3
4 5 6
7 8 9
Kernel
(filter)
31
Output

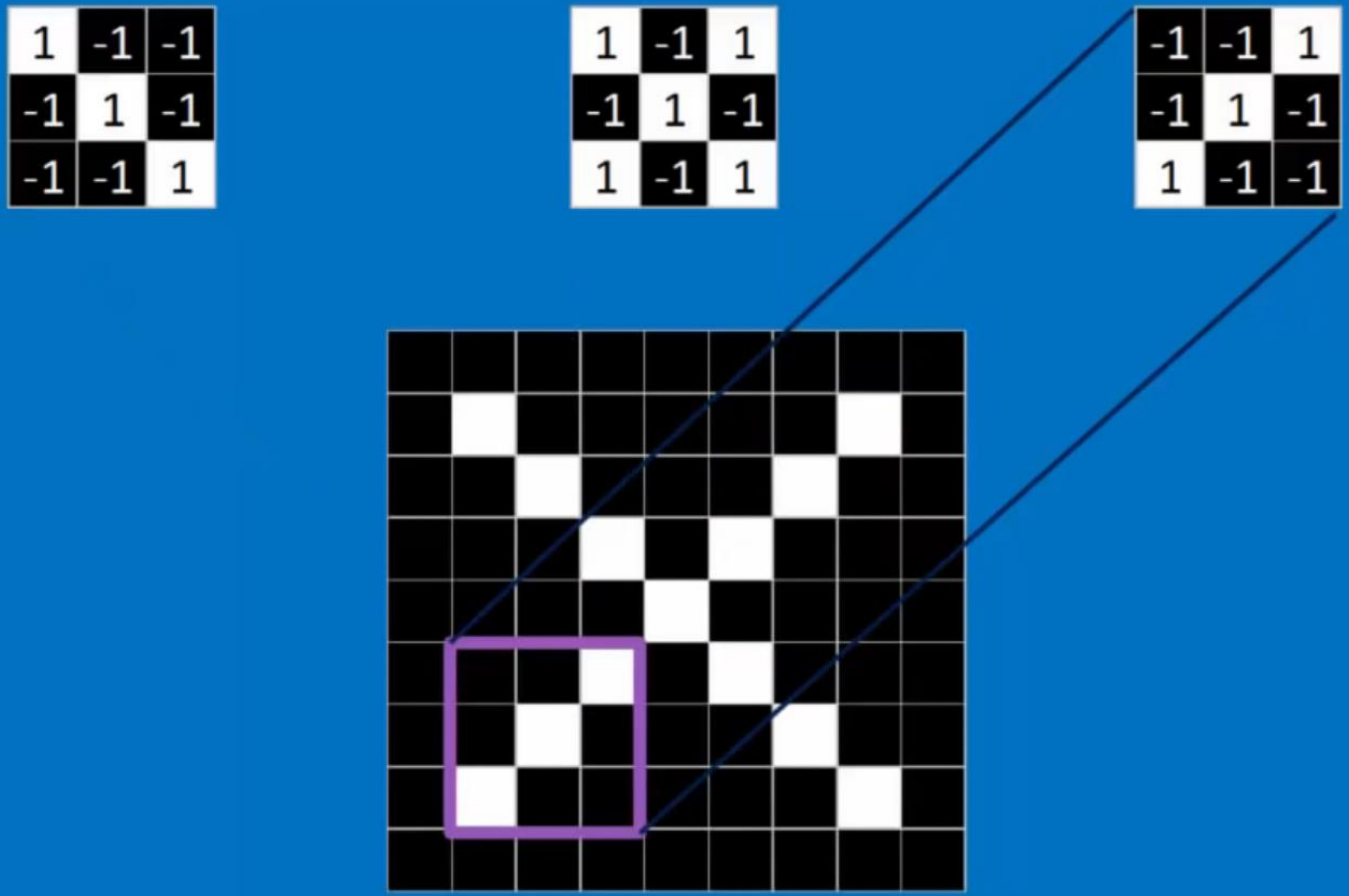

1 14 -9 4
-2 -20 10 6
-3 3 11 1
2 54 -2 80
ReLU
1 14 0 4
0 0 10 6
0 3 11 1
2 54 0 80


Input Feature Map
Rectified Feature Map
ReLU
Black = negative; white = positive values
Only non-negative values

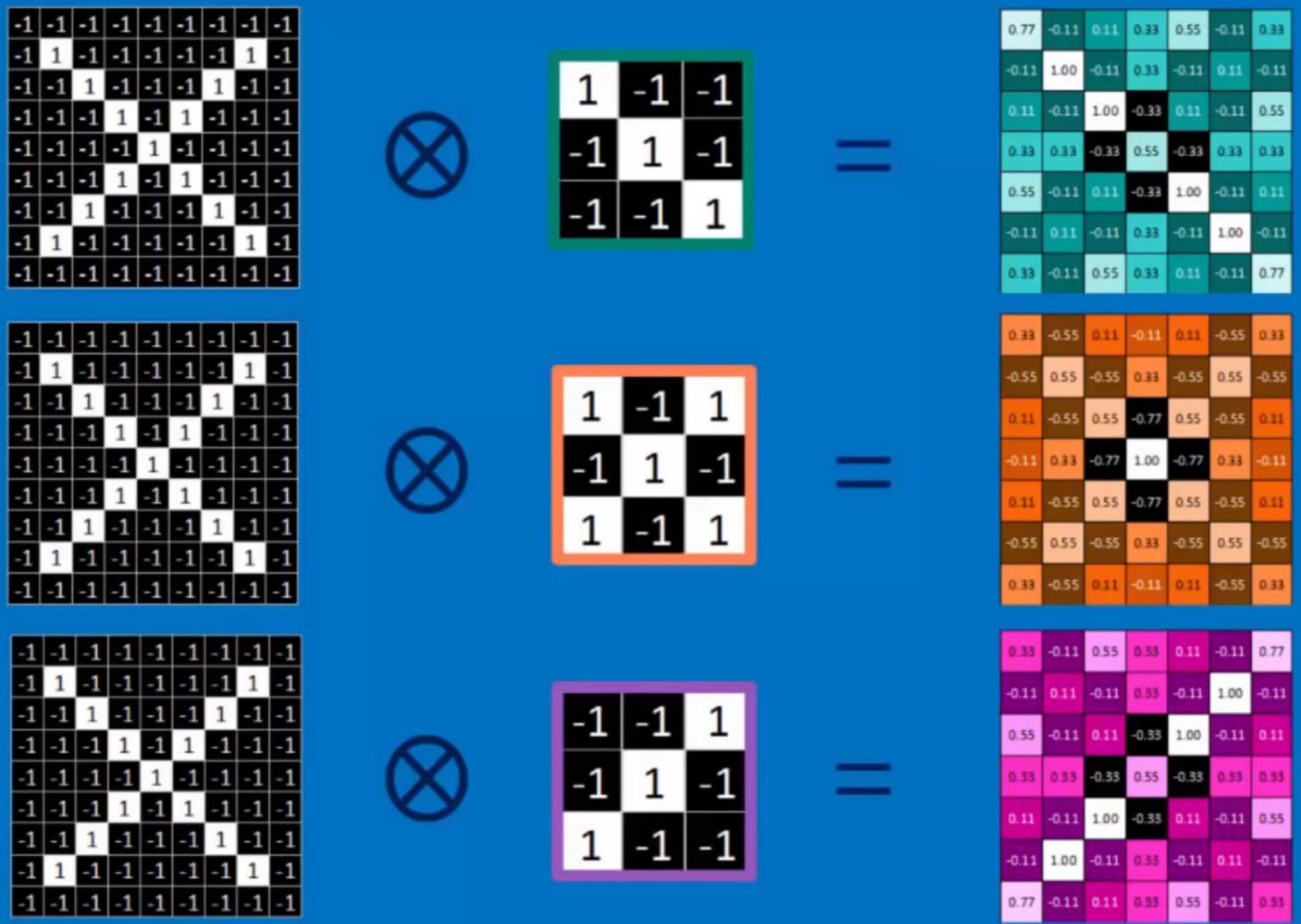
# Pooling Operation



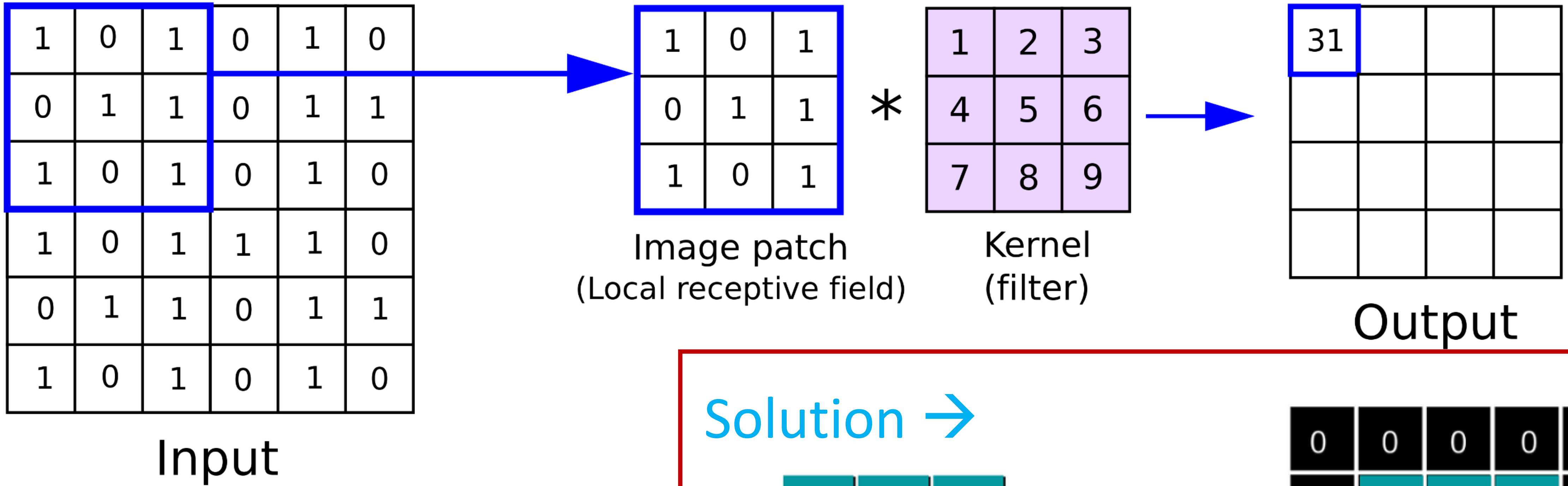

-1 3 8 -4
-2 9 6 5
3 -7 -5 2
1 4 -7 8
ReLU
0 3 8 0
0 9 6 5
3 0 0 2
1 4 0 8
POOLING
9 8
4 8



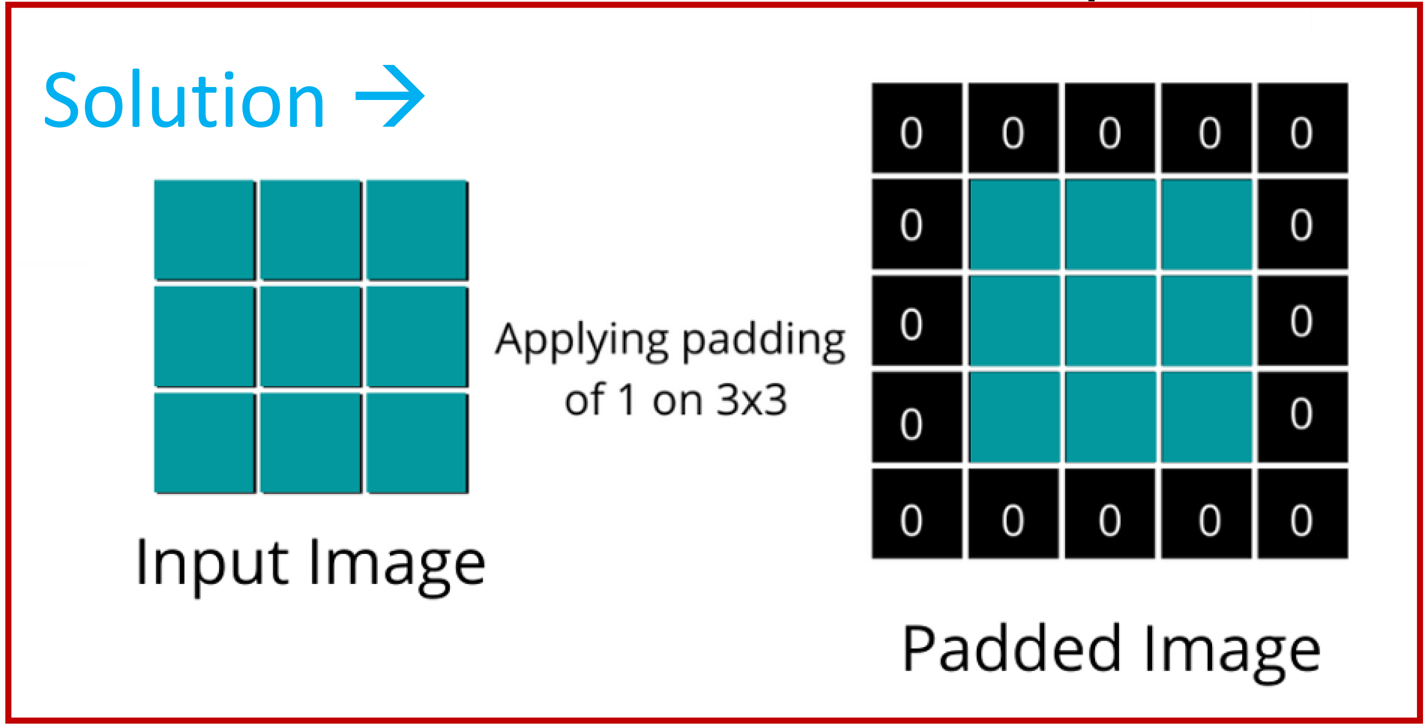
1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1


-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 -1
-1 1 -1
-1 -1 1
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 1
-1 1 -1
1 -1 1
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.11 0.33 -0.77 1.00 -0.77 0.33 -0.11
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 1
-1 1 -1
1 -1 -1
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33

# padding




In Convolutional Neural Networks (CNNs), padding is used to add extra pixels around the borders of the input image or feature maps. This is done to maintain the spatial dimensions of the feature maps throughout the network, ensuring that the convolutional filters can operate on the entire input image.



# padding

**Input with padding**

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 3 | 1 | 4 | 0 |
| 0 | 3 | 1 | 3 | 2 | 0 |
| 0 | 3 | 0 | 1 | 3 | 0 |
| 0 | 0 | 2 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

conv

**kernel**

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

=

Output of original size

| | | | |
|---|---|---|---|
| -4 | 1 | -2 | 4 |
| -4 | 3 | -5 | 5 |
| -3 | 2 | -3 | 4 |
| -2 | 2 | -2 | 1 |

# padding

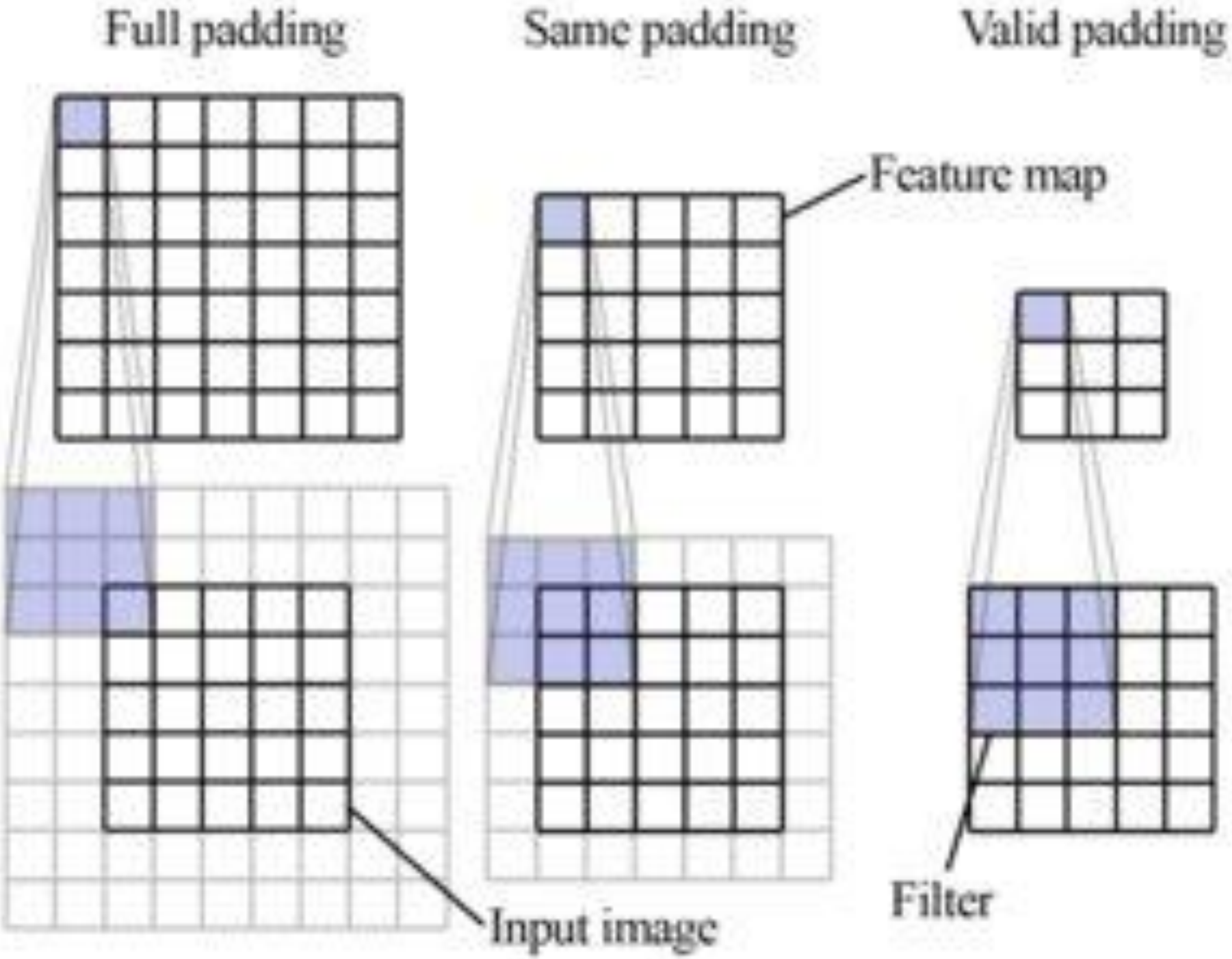
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 60 | 113 | 56 | 139 | 85 | 0 |
| 0 | 73 | 121 | 54 | 84 | 128 | 0 |
| 0 | 131 | 99 | 70 | 129 | 127 | 0 |
| 0 | 80 | 57 | 115 | 69 | 134 | 0 |
| 0 | 104 | 126 | 123 | 95 | 130 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Kernel

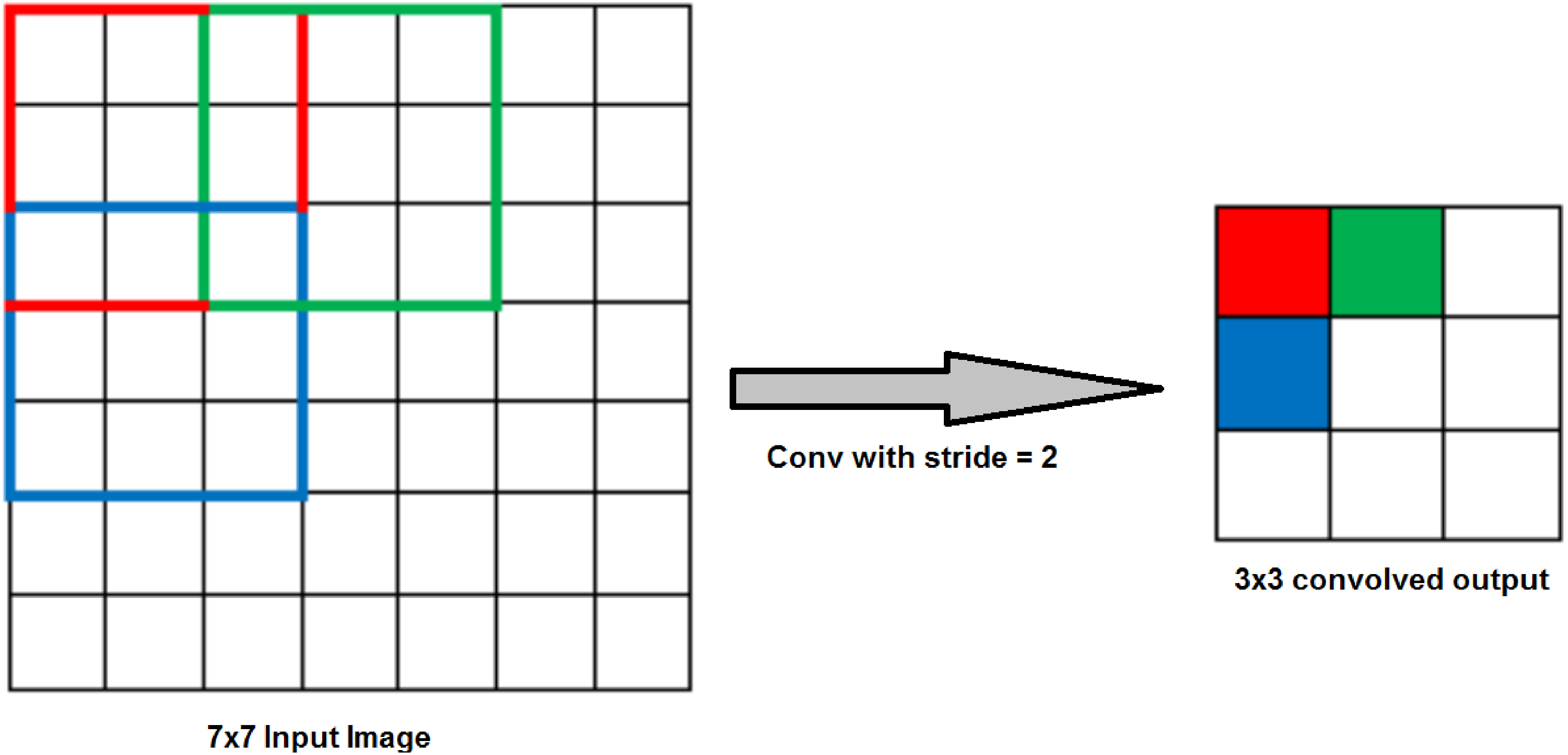
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

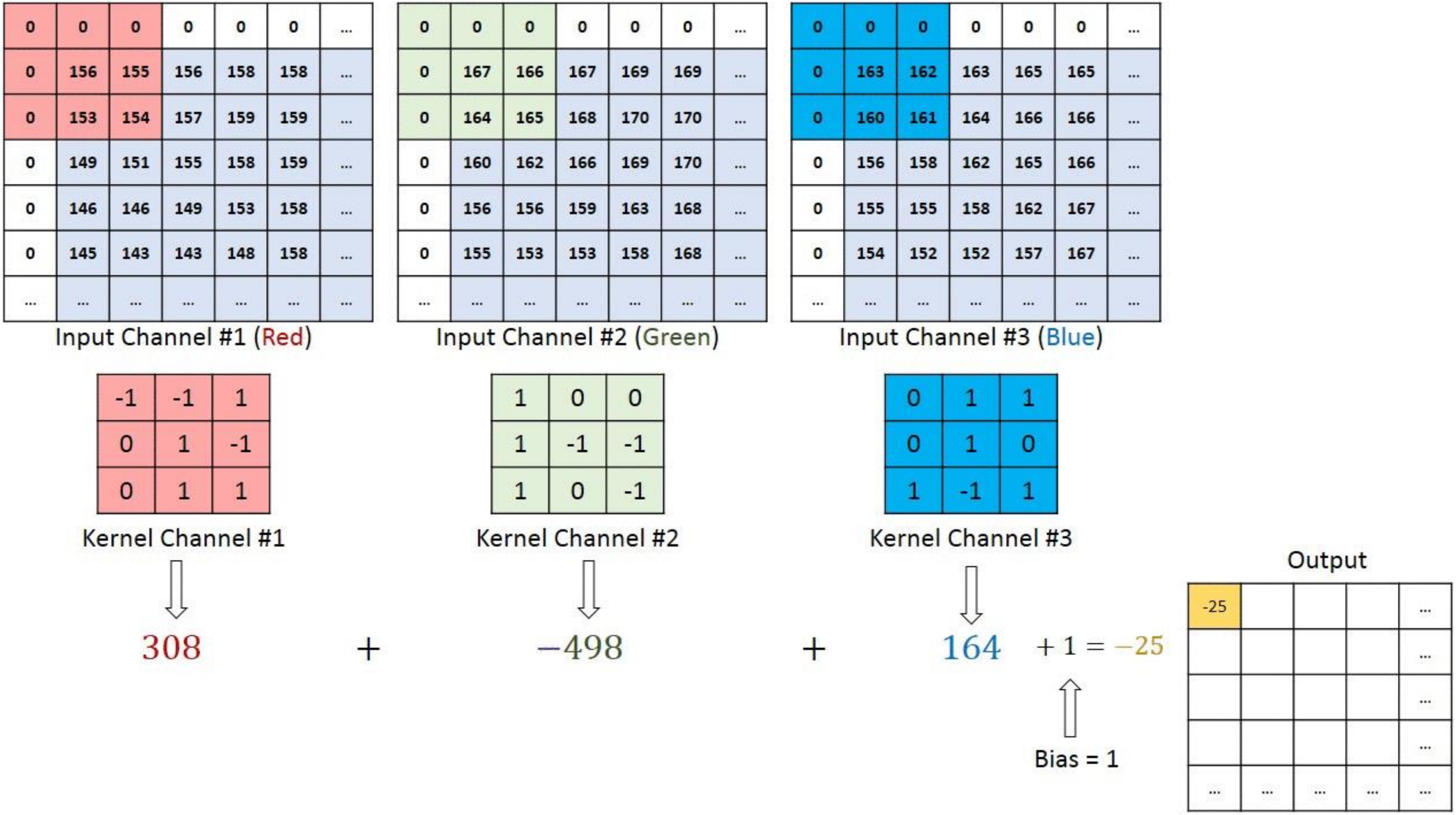
| | | | | |
|---|---|---|---|---|
| 114 | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# padding


Full padding
Same padding
Valid padding
Feature map
Input image
Filter

# stride





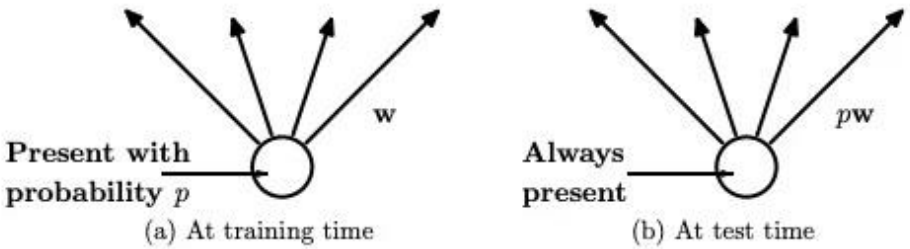
Input Channel #1 (Red)
Input Channel #2 (Green)
Input Channel #3 (Blue)
Kernel Channel #1
Kernel Channel #2
Kernel Channel #3
308
+
−498
+
164
+ 1 = −25
Bias = 1
Output
-25

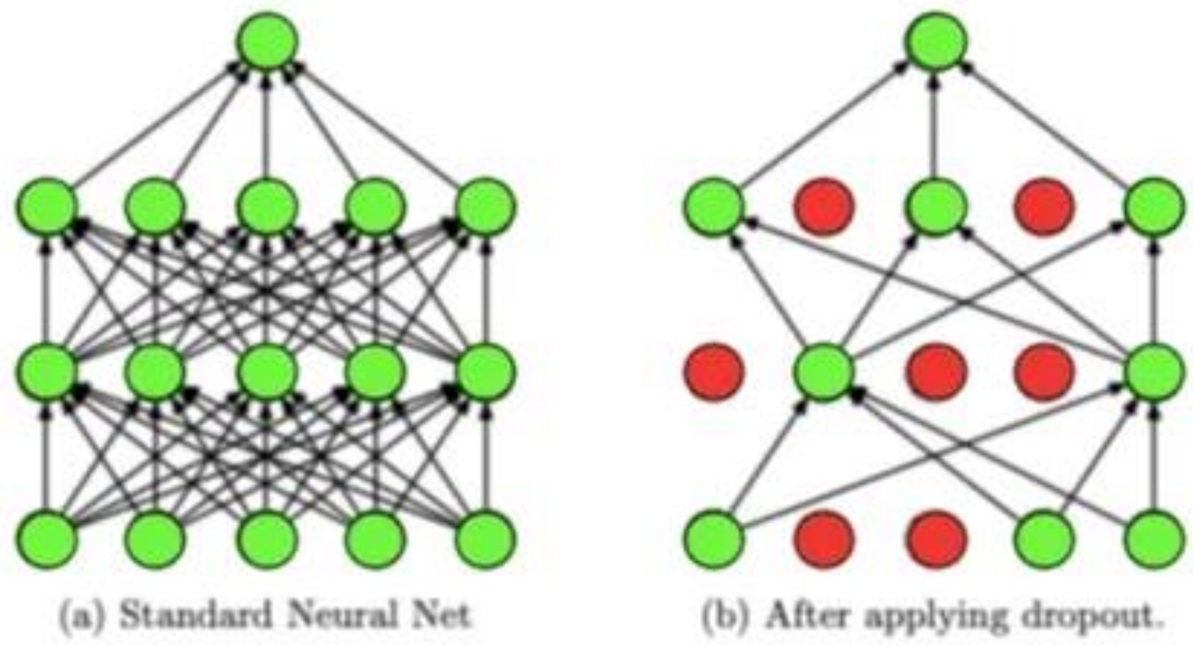
# Dropout layer

During training: At each training step, a certain percentage of neurons in the dropout layer are randomly deactivated, or "dropped out."
During inference: When the network is used for inference (making predictions on new data), all neurons are active, and dropout is not applied.

benefits of using dropout:

- Prevents overfitting
- Improves generalization performance
- Reduces the number of parameters



Given an image of a ball,
can you predict where it will go next?

Given an image of a ball,
can you predict where it will go next?

# Sequences in the Wild

Audio

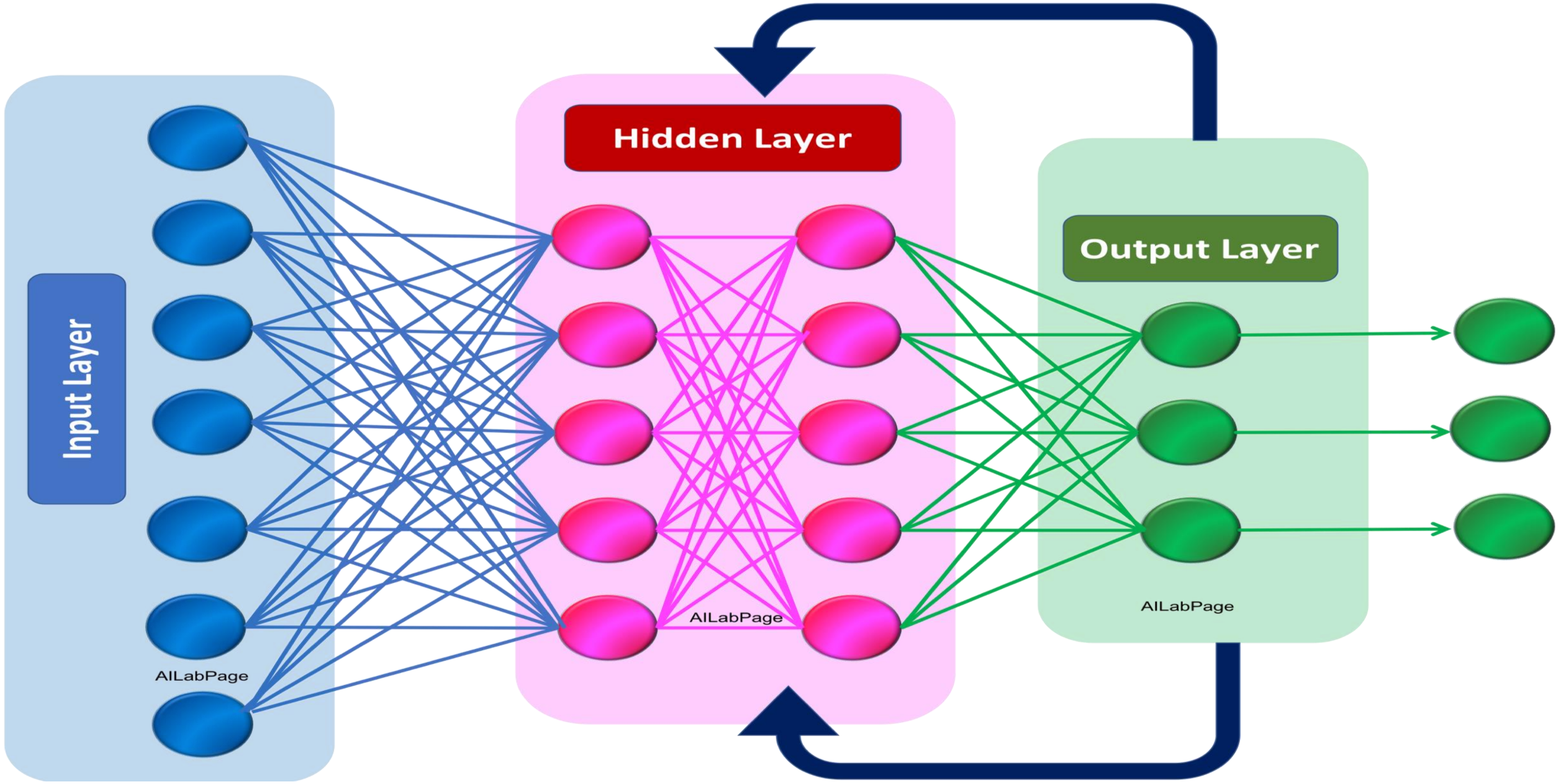
# Sequences in the Wild

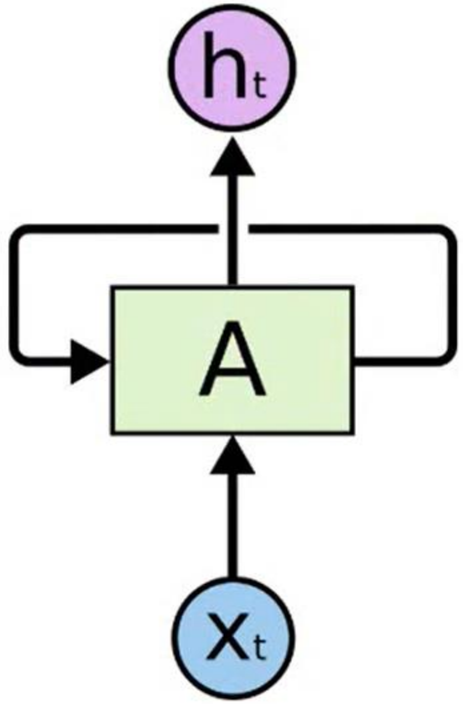
**character:** 6 . S 1 9 1

**word:** Introduction to Deep Learning

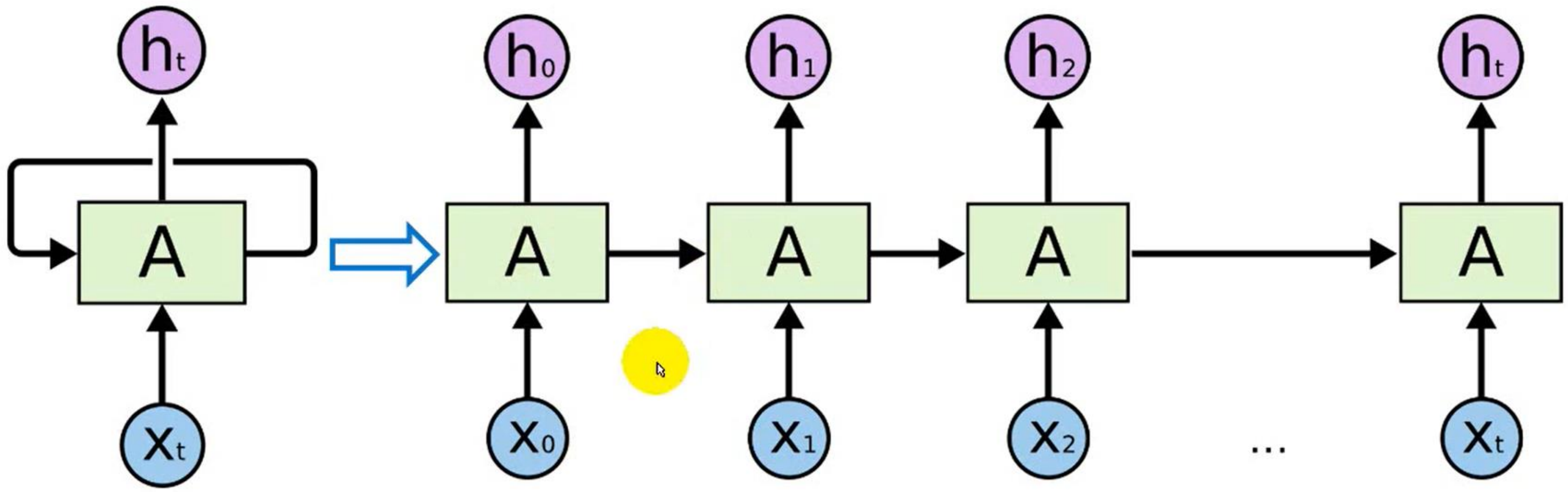
Text

# Recurrent Neural Networks






$$h_t = f_W(h_{t-1}, x_t)$$



h
t
A
X
t
h
0
A
X
0
h
1
A
X
1
h
2
A
X
2
...
h
t
A
X
t