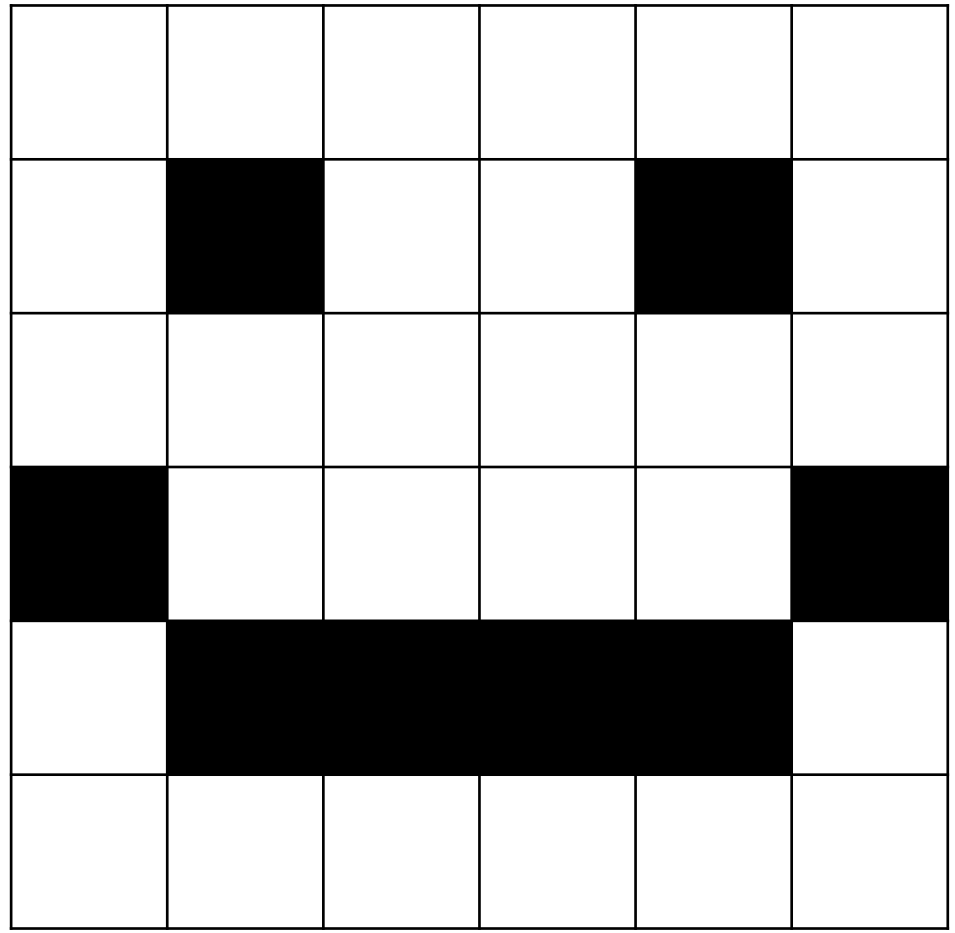
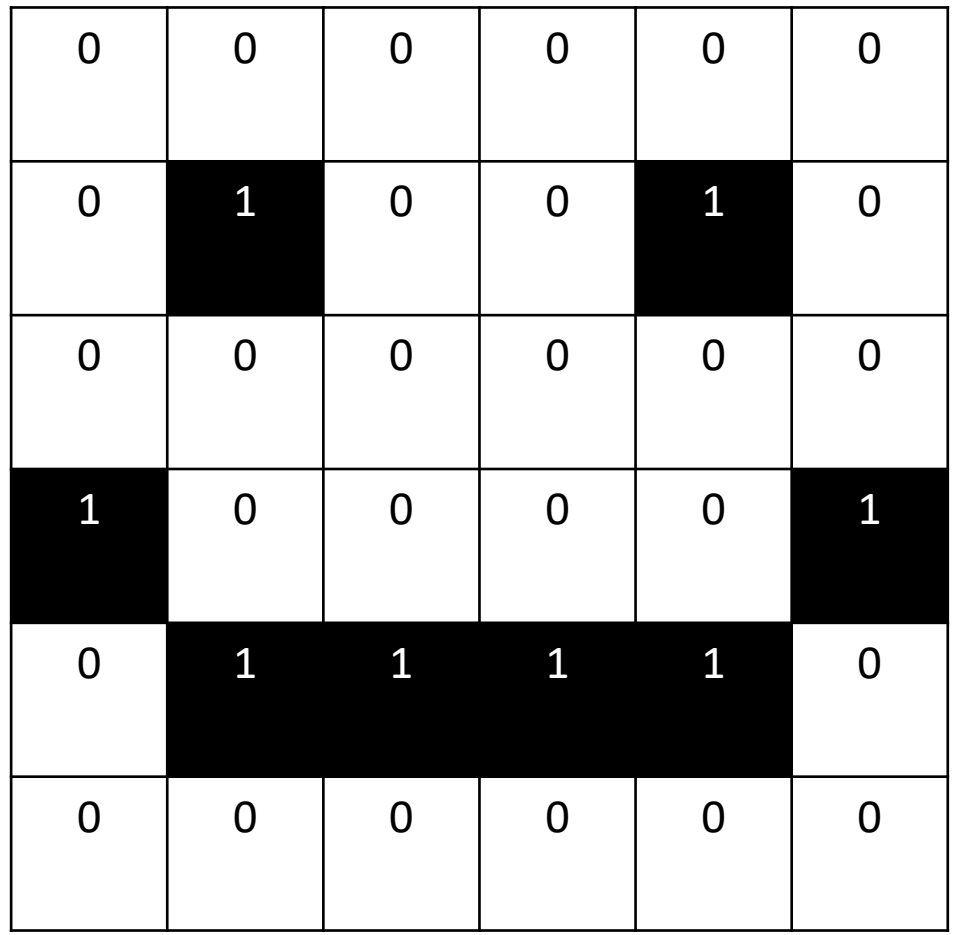
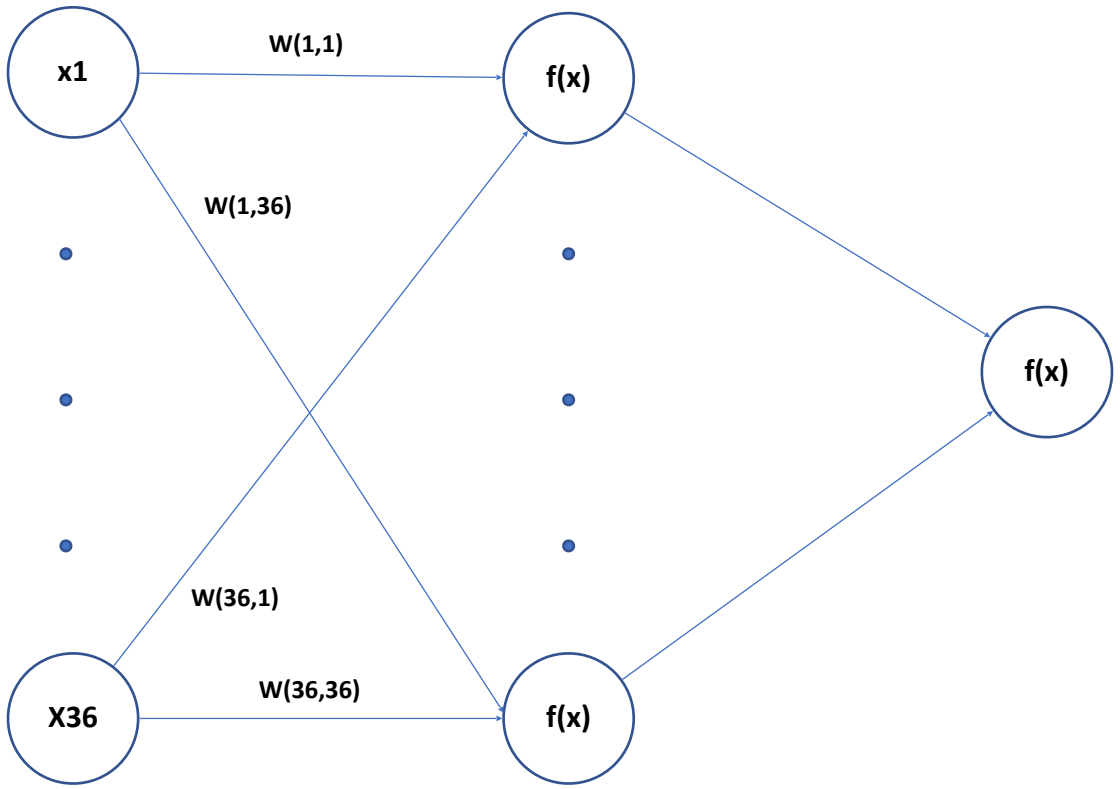



0 0 0 0 0 0
0 1 0 0 1 0
0 0 0 0 0 0
1 0 0 0 0 1
0 1 1 1 1 0
0 0 0 0 0 0

Can we use traditional feed forward nets?



How many weights and biases do we have (if we have one hidden layer with 36 neurons)?

- Hidden layer:
- Output layer:
- Total:

- **What if the image was of a different size?**
- **Let‘s take a 224*224*3 image**
- Input values: 224*224*3 = 150_528
- Hidden layer:
- Output layer:
- Total:

- This will lead to two problems:
  -

Convolutional Neural Networks

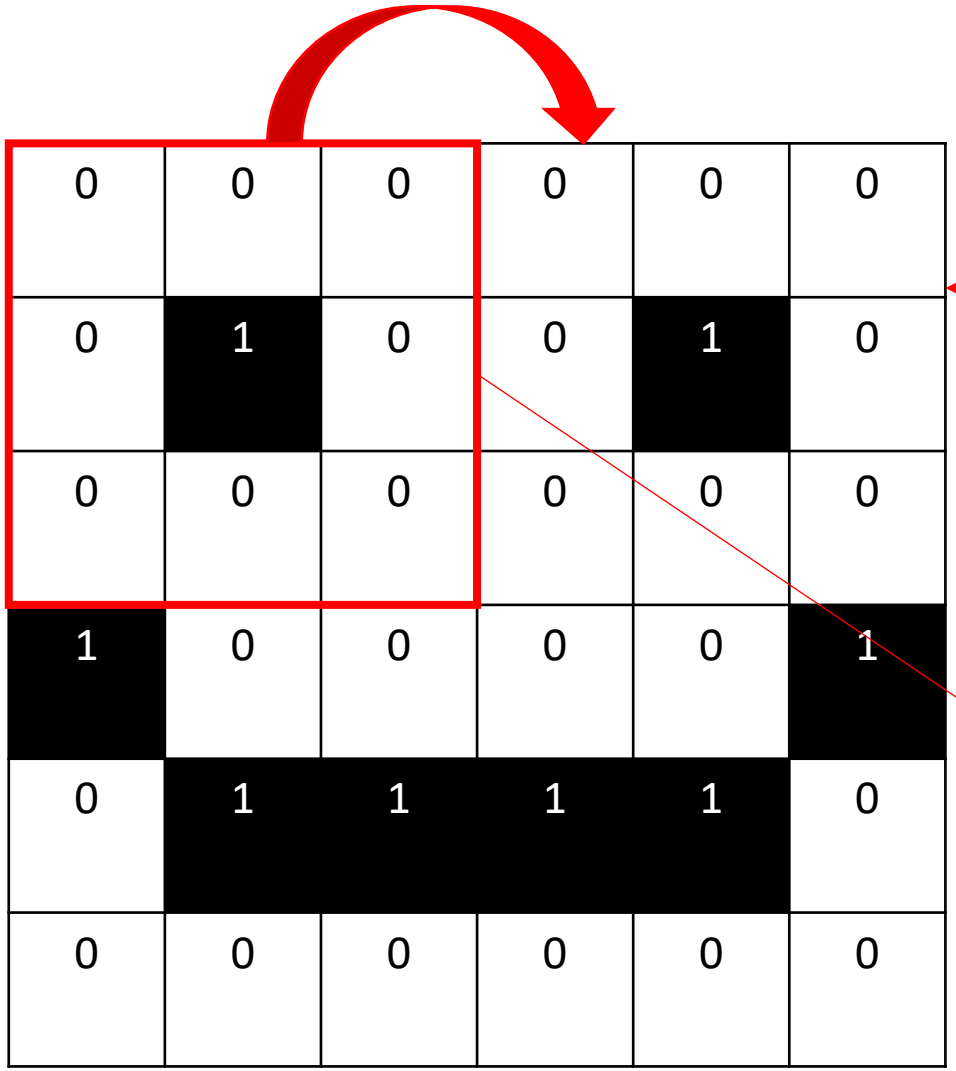
- Don‘t flatten the image
- Apply convolutional kernels/filters to the image

Input Image

Kernel/Filter

Feature Maps

Step size is defined by the „stride“



0	0	0	0	0	0
0	1	0	0	1	0
0	0	0	0	0	0
1	0	0	0	0	1
0	1	1	1	1	0
0	0	0	0	0	0

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

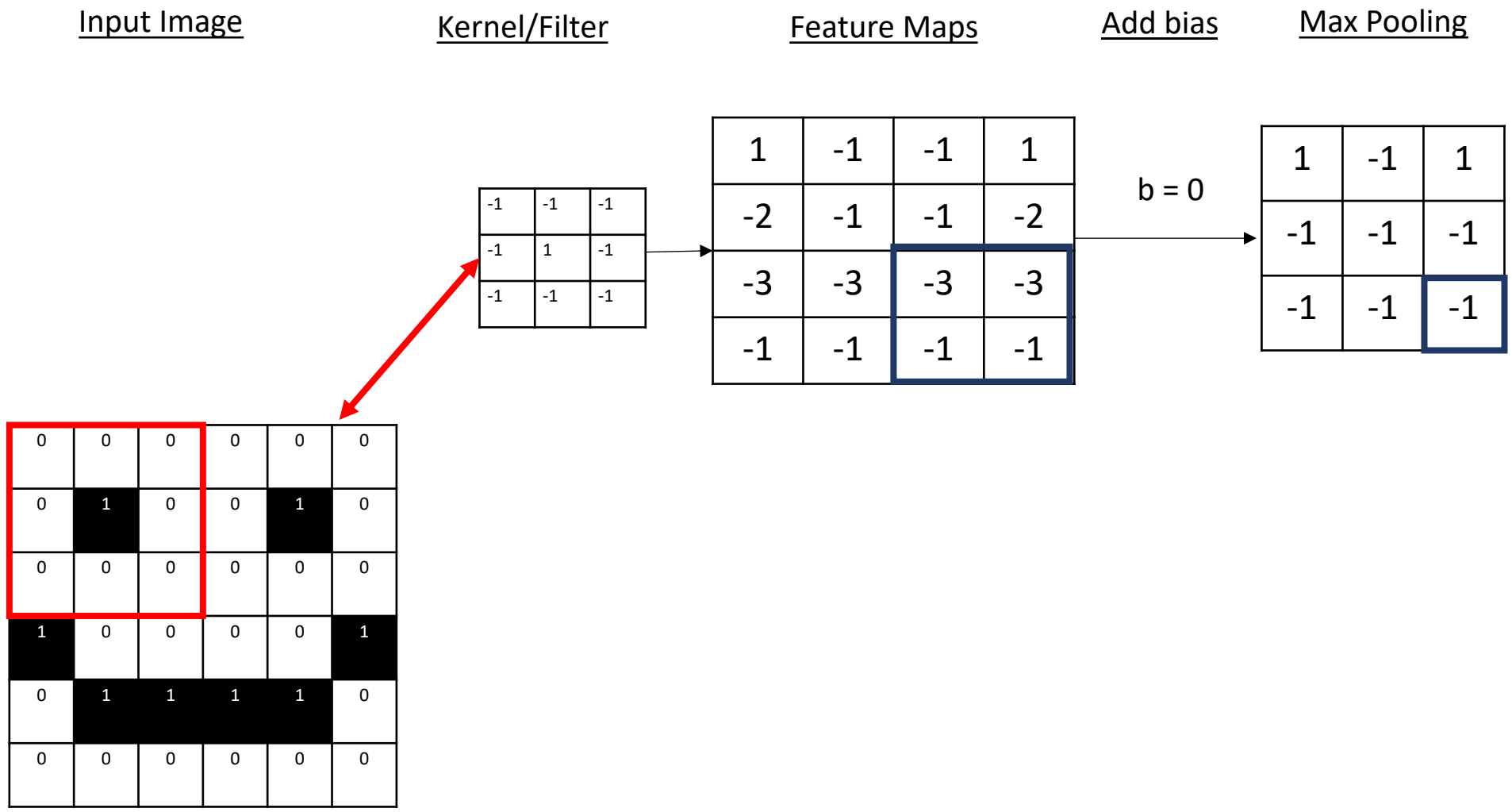
1	-1	-1	1
-2	-1	-1	-2
-3	-3	-3	-3
-1	-1	-1	-1

Take the dot product between the kernel and the section

0 * -1 + 0 * -1 + 0 * -1 +
0 * -1 + 1 * 1 + 0 * -1 +
0 * -1 + 0 * -1 + 0 * -1 = 1

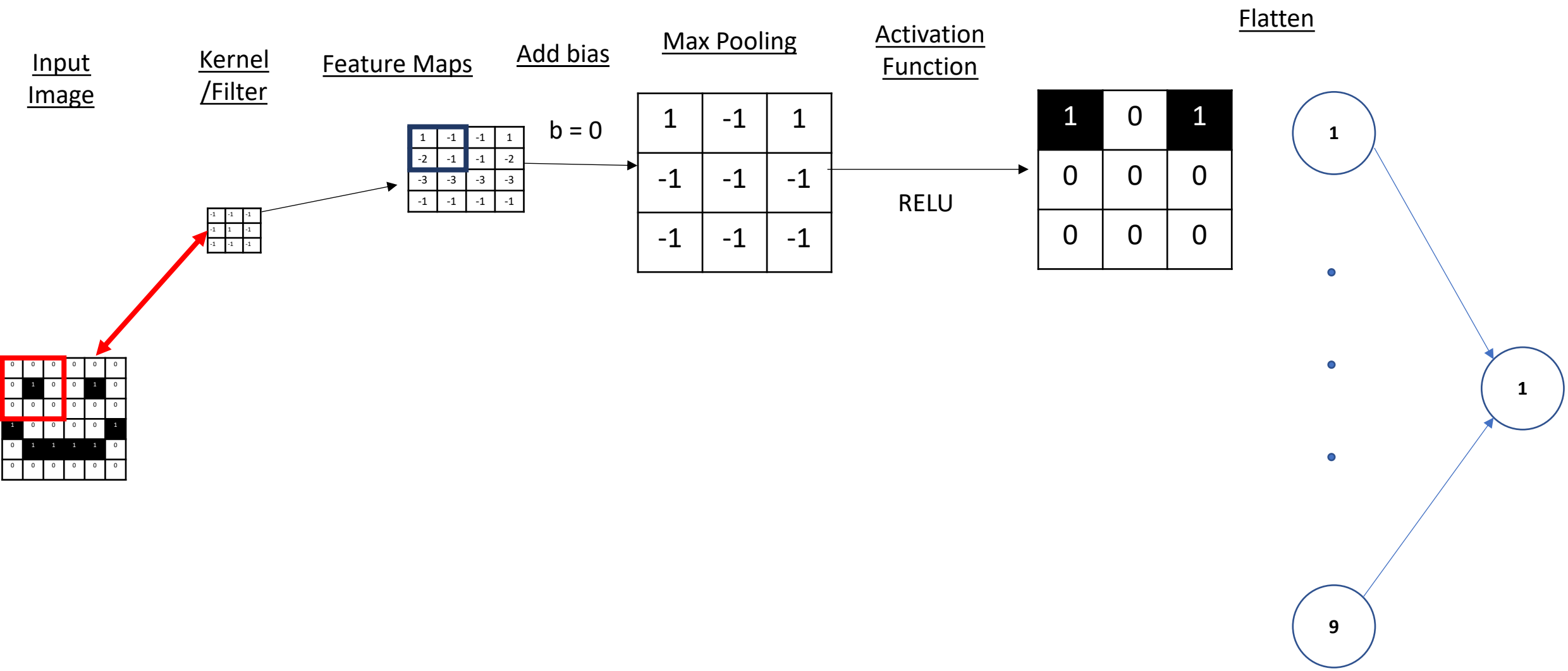
Convolutional Neural Networks

- Don‘t flatten the image
- Apply convolutional kernels/filters (randomly initialized in the beginning) to the image -> Feature Map
- Add a bias (also randomly initialized in the beginning) to every value of the feature map
- Max Pooling: Slide a window over your image


Input Image
Kernel/Filter
Feature Maps
Add bias
Max Pooling
b = 0

Convolutional Neural Networks

- Don‘t flatten the image
- Apply convolutional kernels/filters (randomly initialized in the beginning) to the image -> Feature Map
- Add a bias (also randomly initialized in the beginning) to every value of the feature map
- Max Pooling: Slide a window over your image → End up with a reduced feature map
- Activation Function: In our example we will use ReLu


Input Image
Kernel /Filter
Feature Maps
Add bias
b = 0
Max Pooling
Activation Function
RELU
Flatten
1
9
1

Input Image

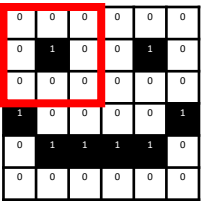
Convolutional Kernel

Add bias

Feature Map

Max Pooling

ReLu/Activation



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

b = 0

1	-1	-1	1
-2	-1	-1	-2
-3	-3	-3	-3
-1	-1	-1	-1

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

1	0	1
0	0	0
0	0	0

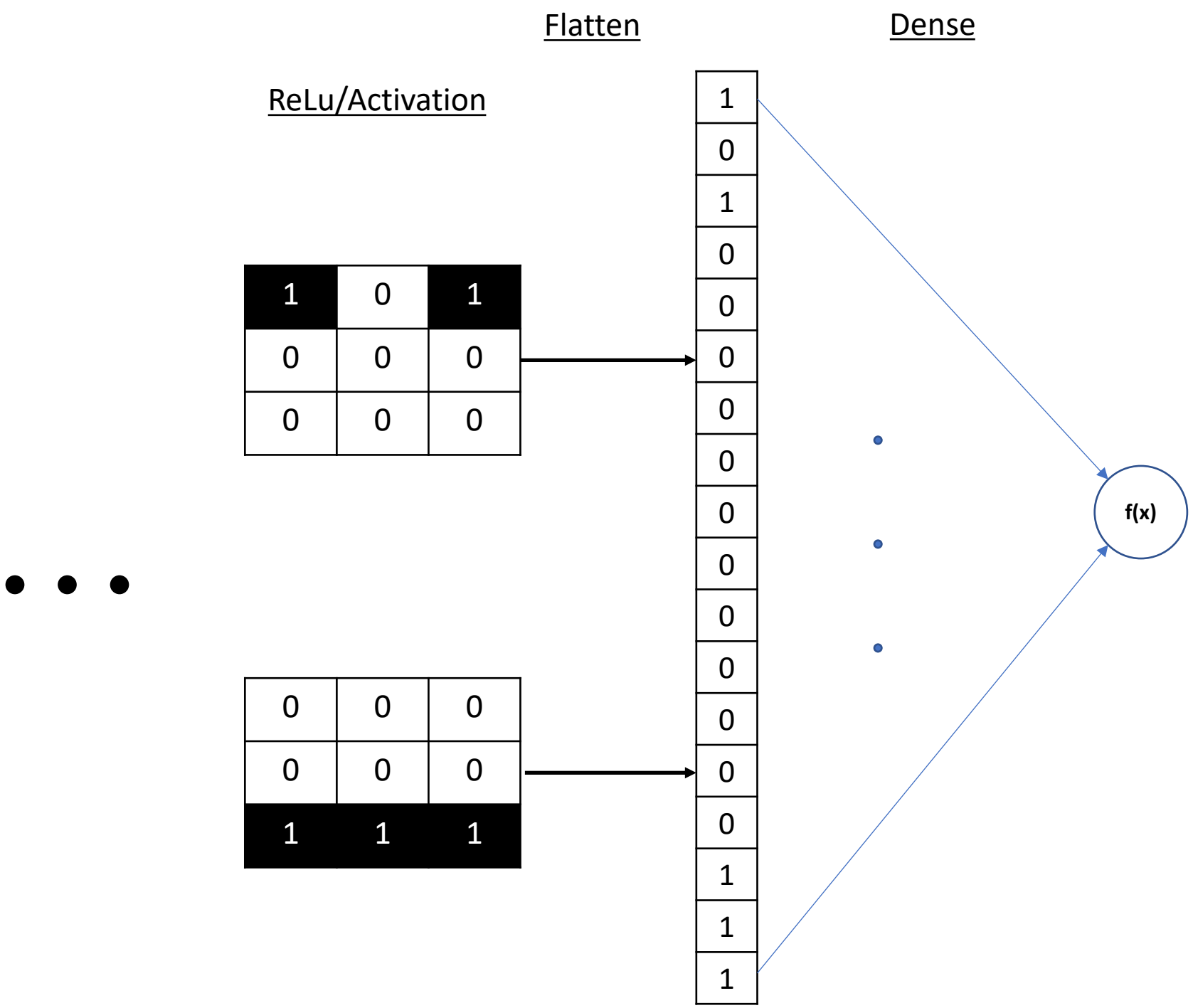
1	-1	1
1	1	1
-1	-1	-1

b = -2

-1	-1	-1	-1
-4	-1	-1	-4
-3	-5	-5	-3
1	1	1	1

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

0	0	0
0	0	0
1	1	1


Flatten
Dense
ReLu/Activation
1 0 1
0 0 0
0 0 0
0 0 0
0 0 0
1 1 1
1
0
1
0
0
0
0
0
0
0
0
0
0
0
0
0
1
1
1
f(x)

Convolutional Layers

- They also have weights and biases. The number just does not increase with the image size.
- Also, convolutional layers exploit the fact that pixels are meaningful in their close environment, not by themselves.
- Do Convolutional Layers suffer from problems?
- Yes:
  - Filter reduces the size of the feature space → loss of information at the edges
  - Solution: padding

Convolutional Layers - Hyperparameters

- What are hyperparameters of convolutional layers?
  - Padding? – yes or no
  - Nr. of kernels
  - Size of kernels, 3x3, 5x5
  - Stride of kernels, (1x1), (2x2)
  - Size of Max-Pooling layer, 2x2, 3x3
  - Stride of Max-Pooling layer, 2x2
  - Activation Function, ReLu
  - Nr. of convolutional layers
  - Sequence of layer parts

Convolutional Layers – Max-Pooling

- There are two main ideas behind Max-Pooling:

1) Reduce the number of features
2) Small variations (shifts by a few pixels) get cancelled out

Convolutional Layers – Padding

- Padding ensures that you don‘t lose information at the edge of the image

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0