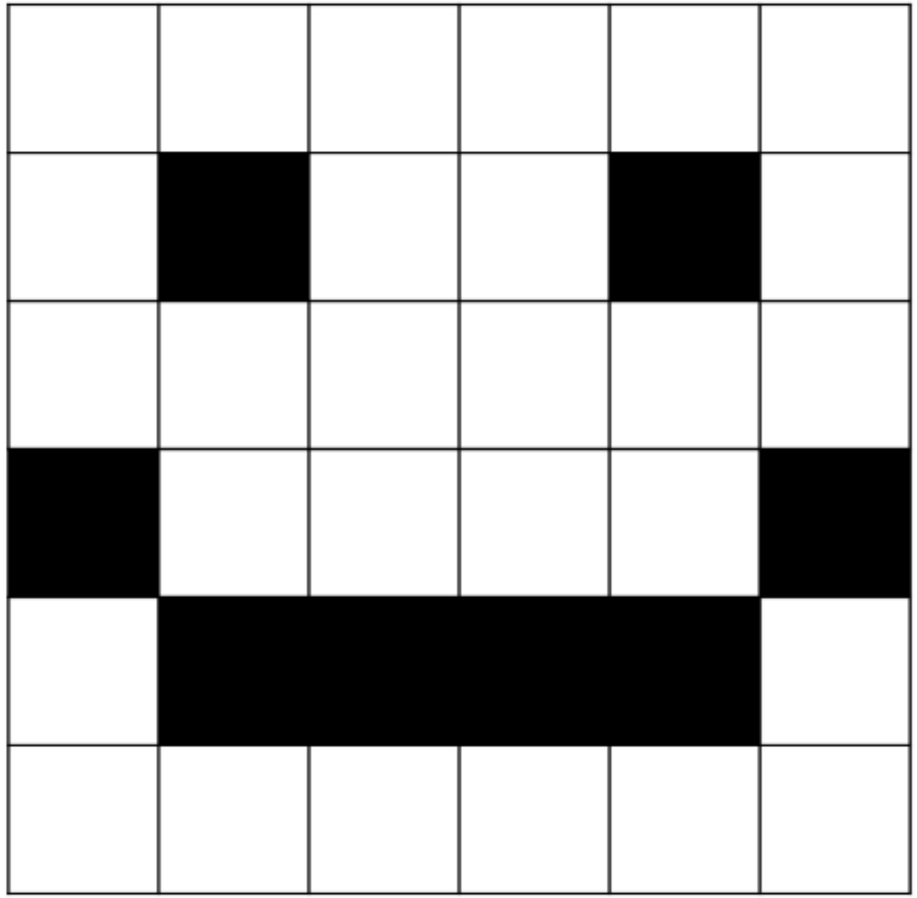
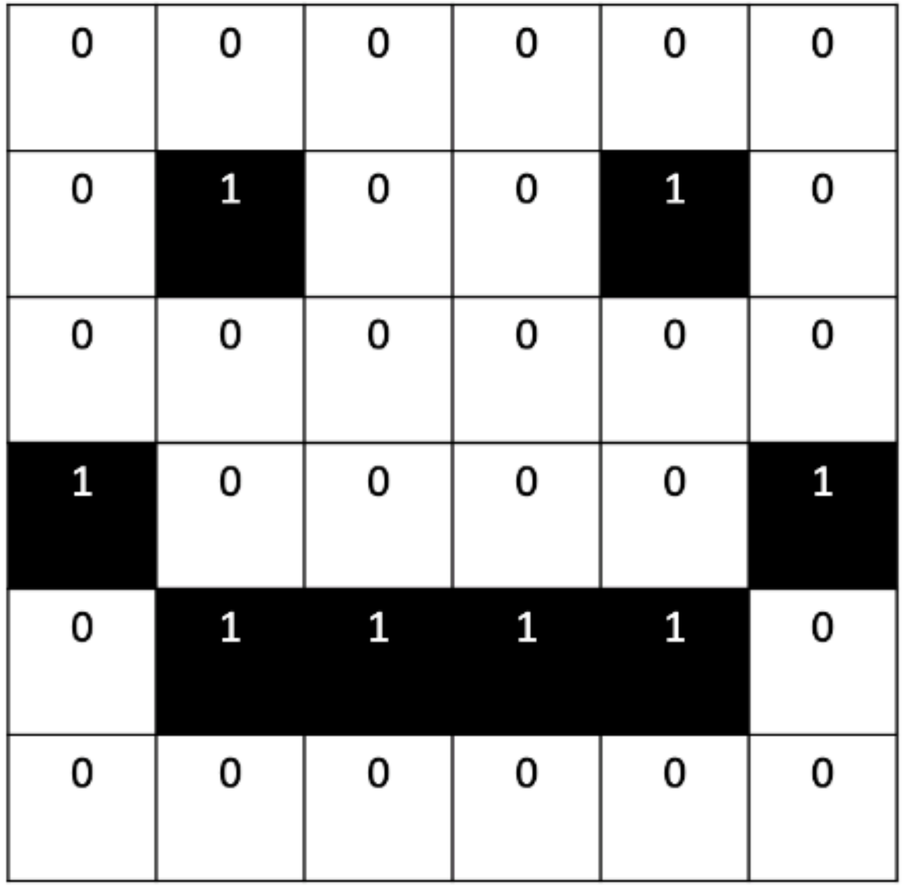
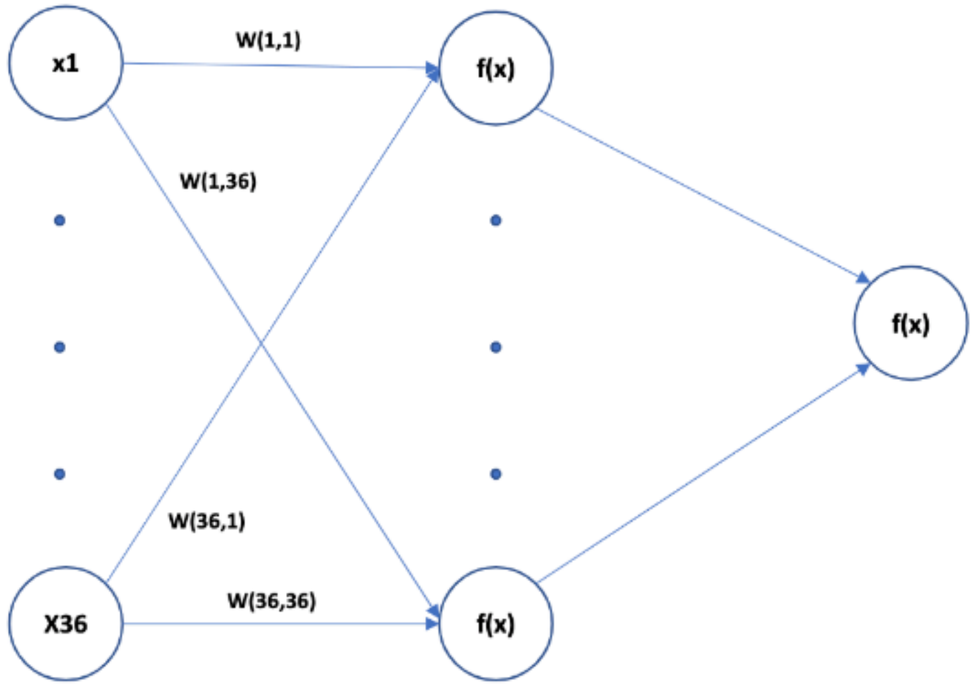




Shape: 6x6

Using a traditional feed forward net with one hidden layer:

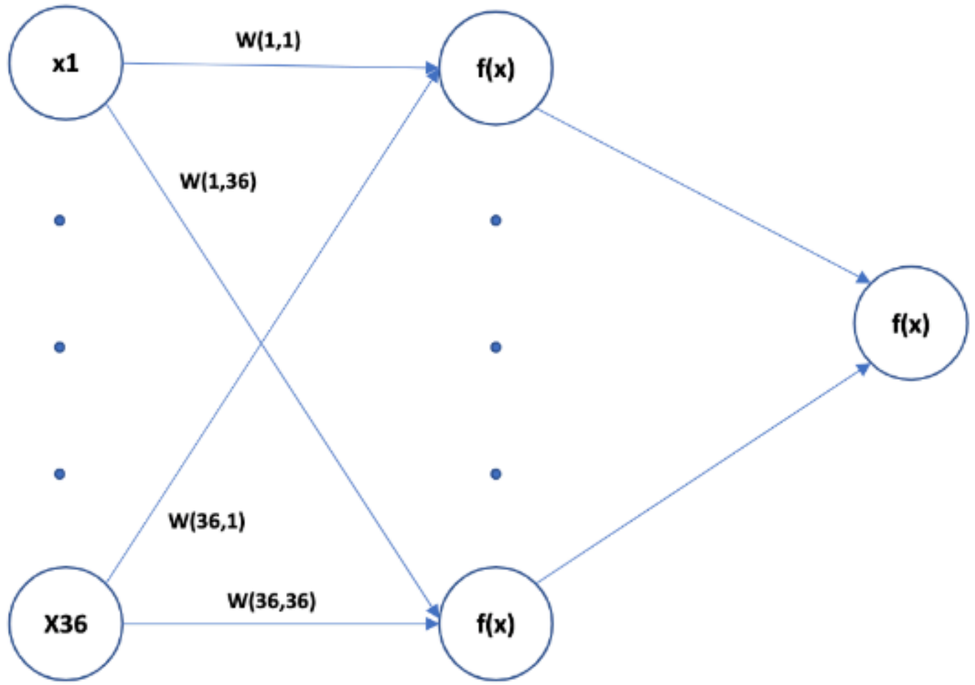
How many weights and biases would we have?



Using a traditional feed forward net with one hidden layer:

How many weights and biases would we have?

36 * 36 + 36 (bias) = 1332



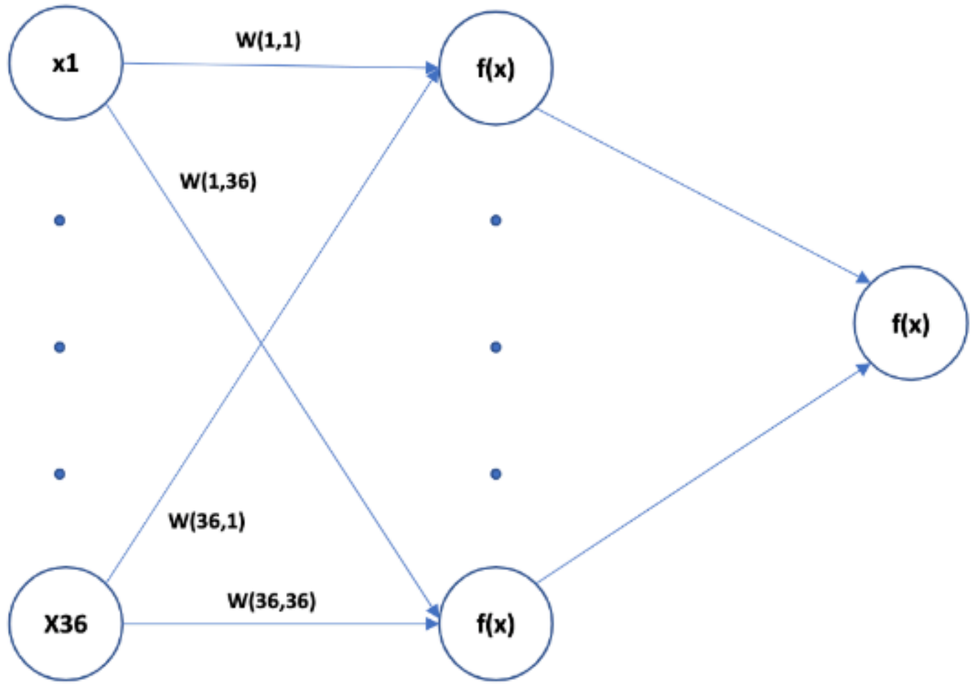
Using a traditional feed forward net with one hidden layer:

How many weights and biases would we have?

36 * 36 + 36 (bias) = 1332

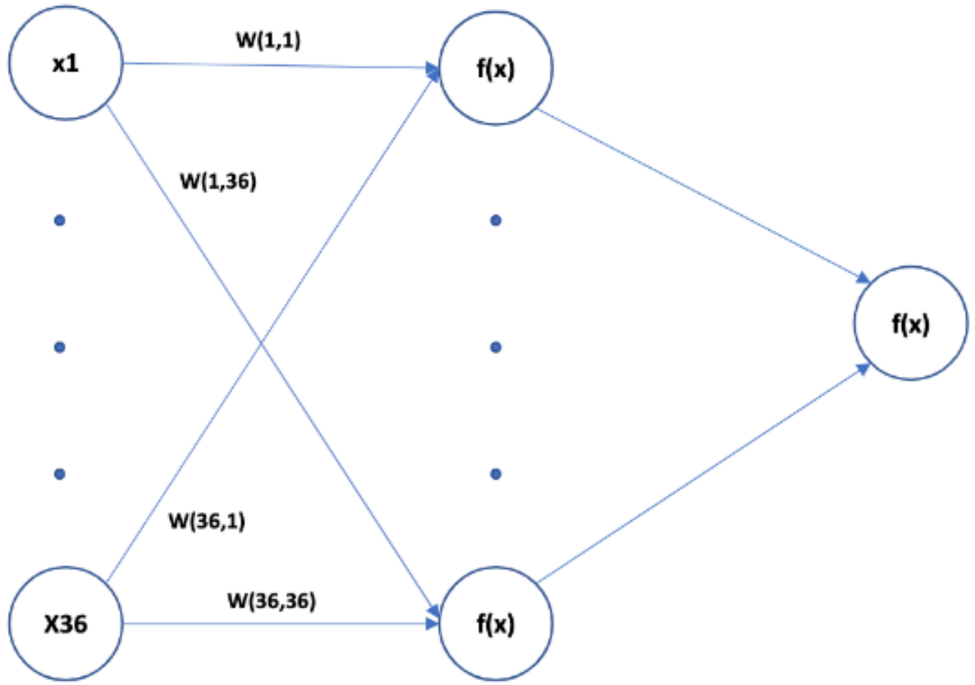
36 * 1 + 1 = 37

In total: 1369



Using a traditional feed forward net with one hidden layer:

What if our image was not this small, but e.g. 400 * 400 * 3?



Using a traditional feed forward net with one hidden layer:

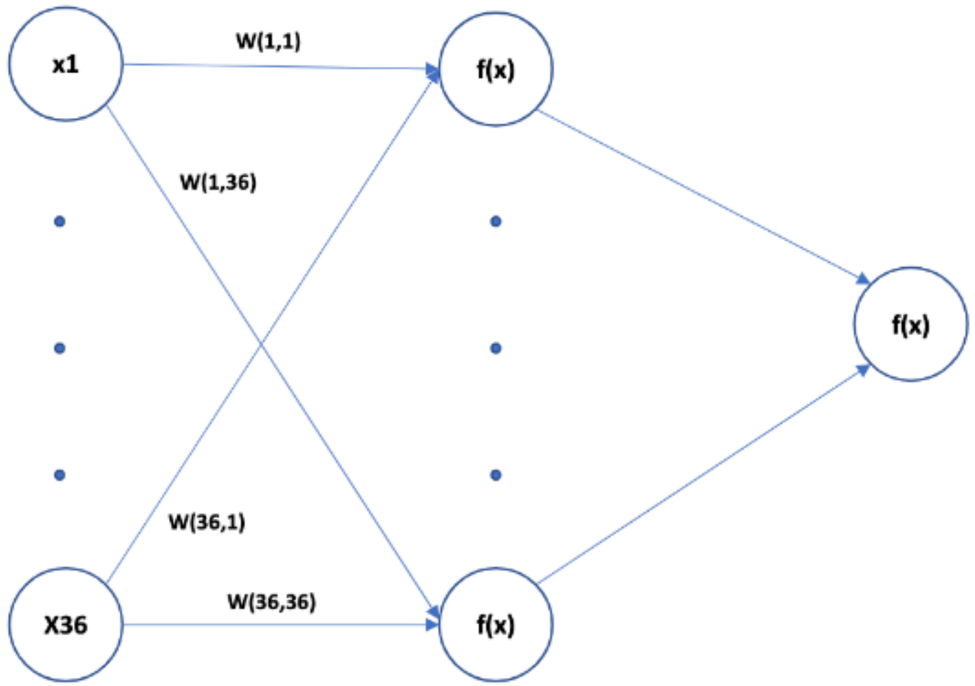
What if our image was not this small, but e.g. 400 * 400 * 3?

400 * 400 * 3 = 480_000

480_000 * 100 + 100 = 48_000_100

100 * 1 + 1 = 101

In total: 48_000_201



Using a traditional feed forward net with one hidden layer:

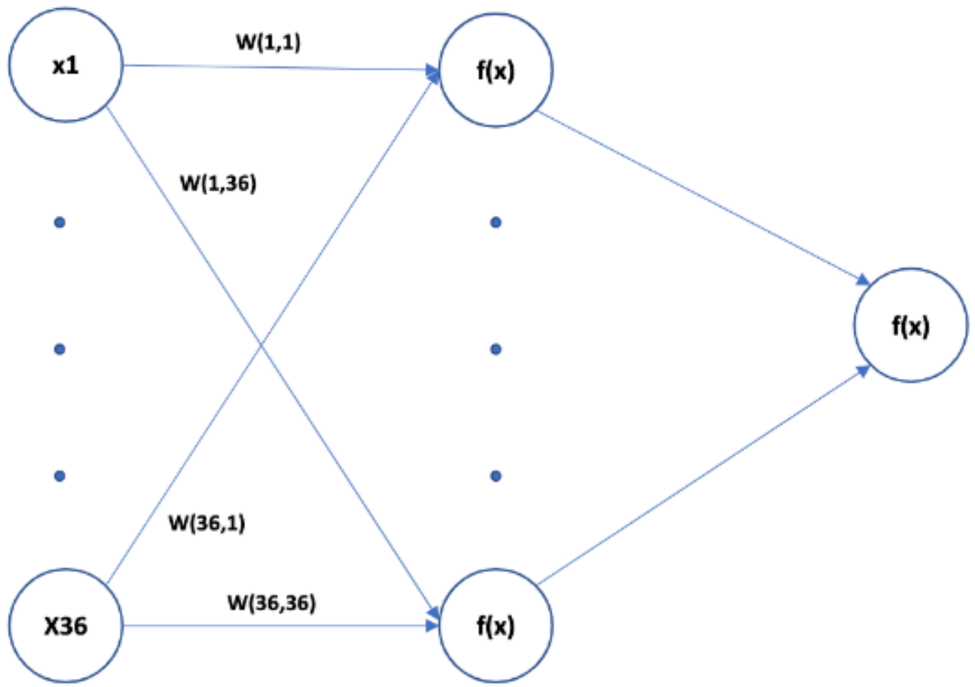
What if our image was not this small, but e.g. 400 * 400 * 3?

400 * 400 * 3 = 480_000

480_000 * 100 + 100 = 48_000_100

100 * 1 + 1 = 101

In total: 48_000_201

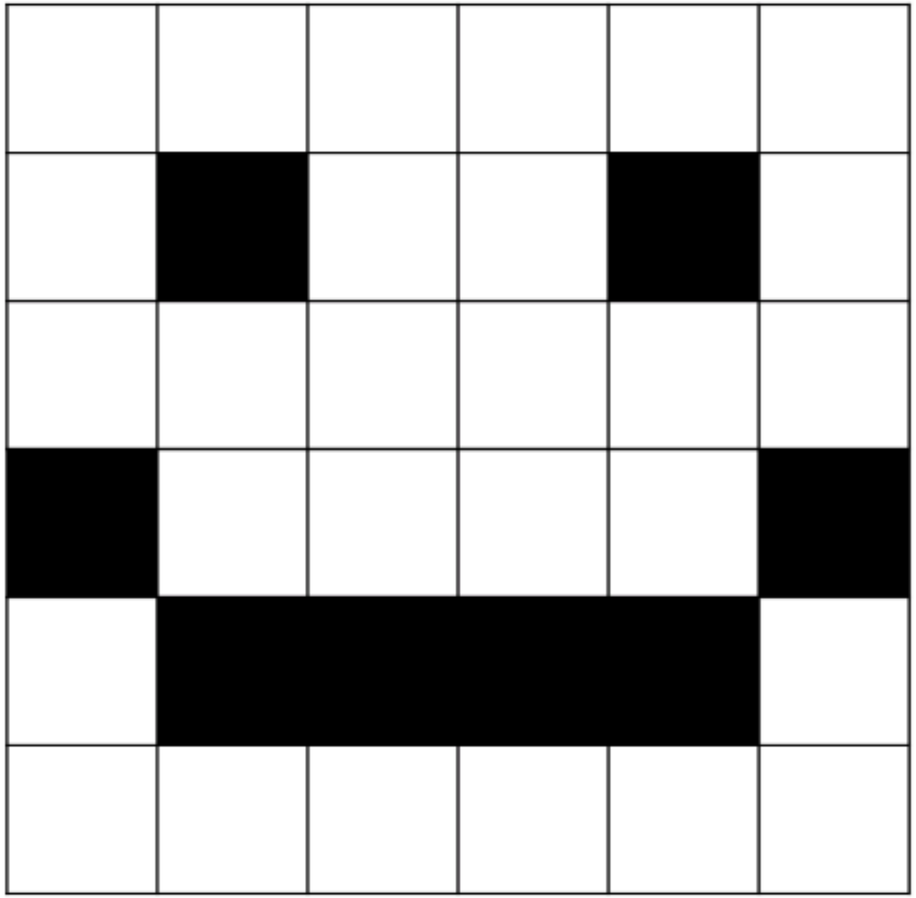


-> Leads to overfitting and slow models

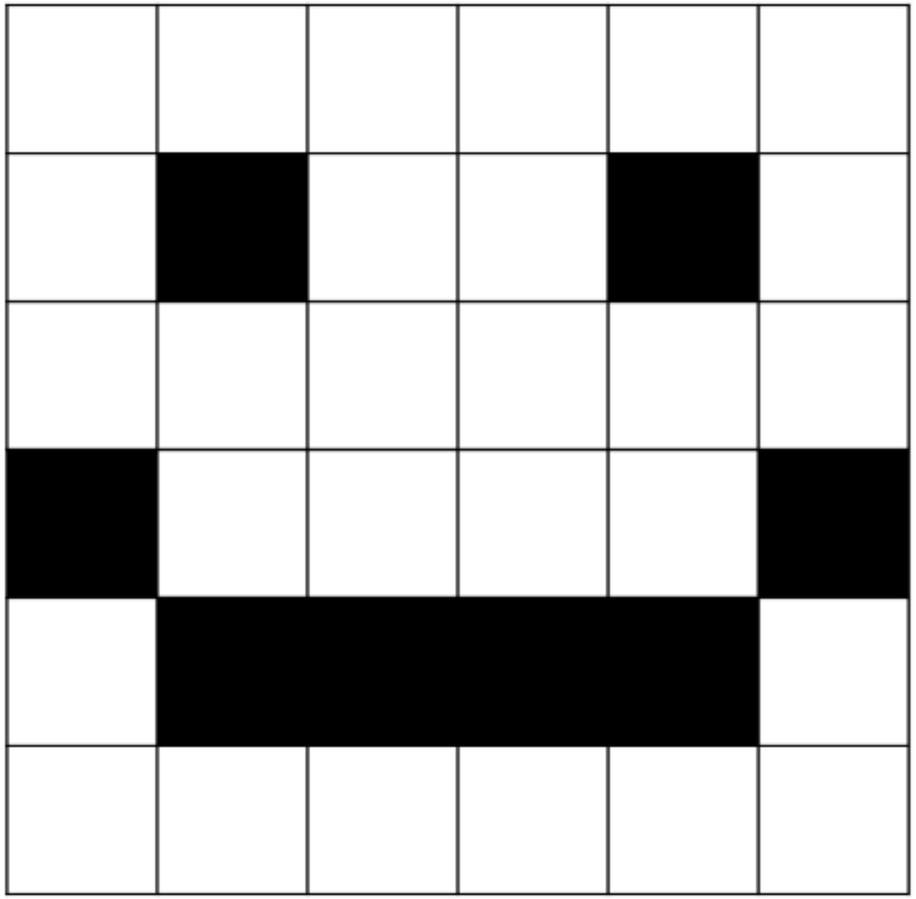
ConvNets to the rescue!

They:

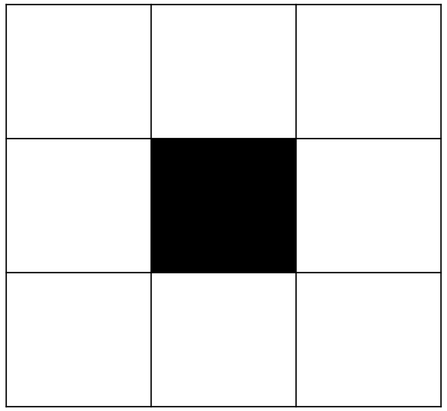
- reduce the number of input nodes
- tolerate shifts
- capture spatial connections between pixels



How do we recognise the eyes in this image?



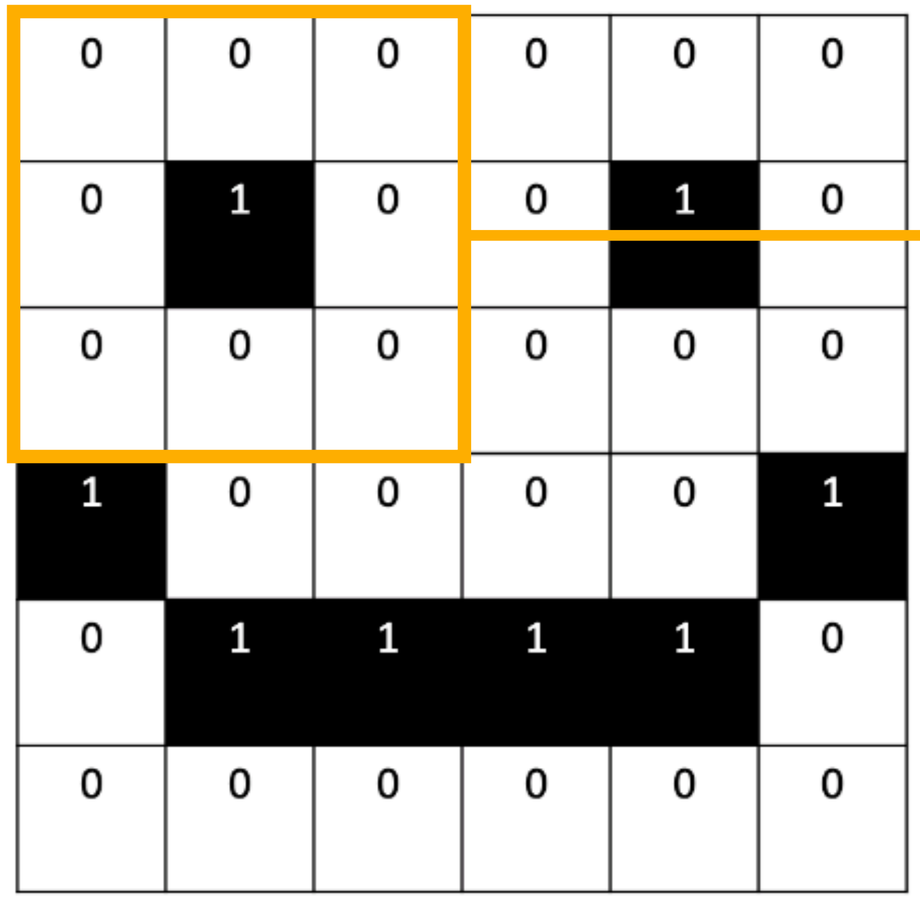
How do we recognise the eyes in this image?



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

Filter

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

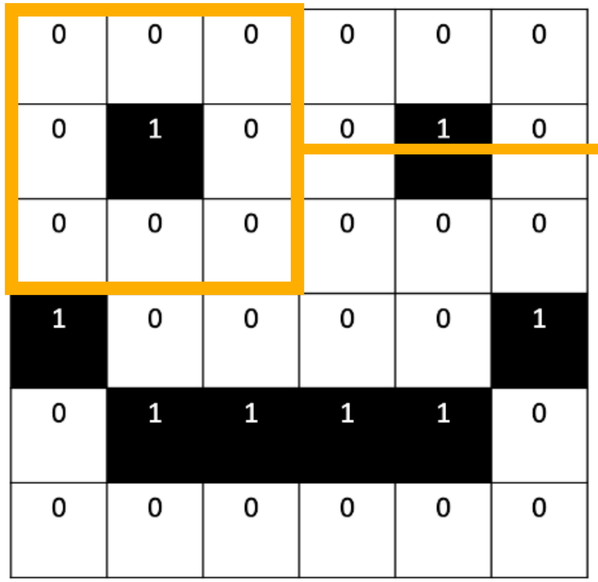


Filter

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

Multiply overlapping values and add them together:

$$(0*-1) + (0*-1) + (0*-1) + (0*-1) + (1*1) + (0*-1) + (0*-1) + (0*-1) + (0*-1) = 1$$



Filter

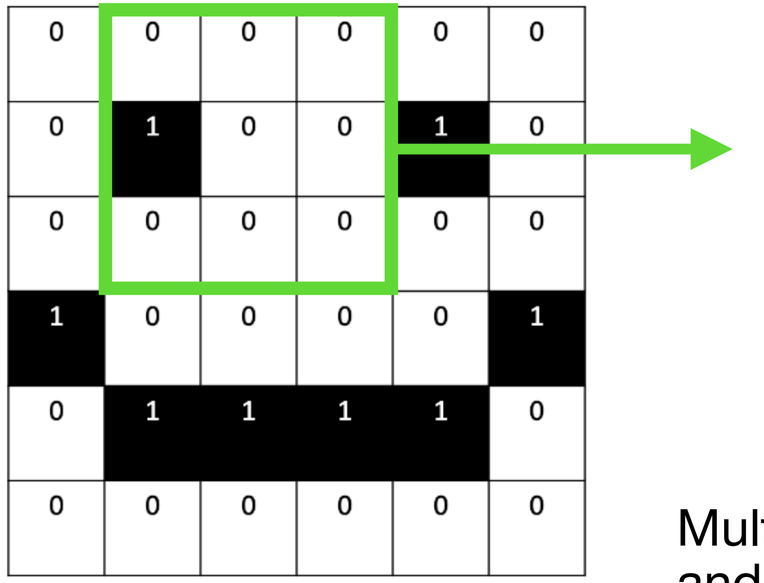
-1	-1	-1
-1	1	-1
-1	-1	-1

Multiply overlapping values and add them together:

(0*-1) + (0*-1) + (0*-1)
+ (0*-1) + (1*1) + (0*-1)
+ (0*-1) + (0*-1) + (0*-1) = 1

Feature Map

1			



Filter

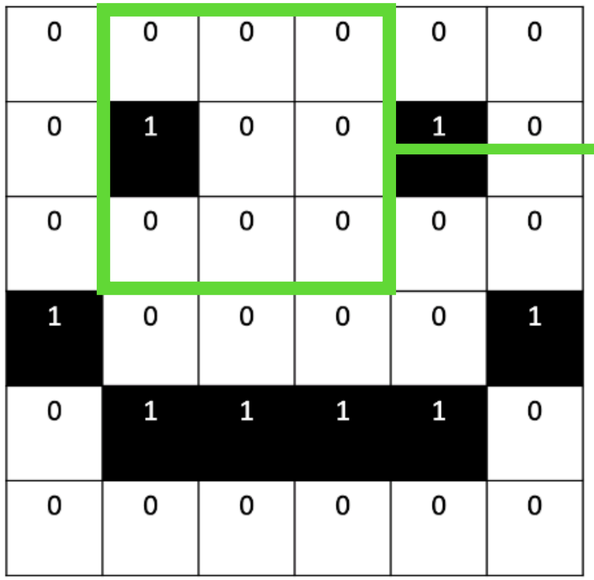
-1	-1	-1
-1	1	-1
-1	-1	-1

Multiply overlapping values and add them together:

(…) + (…) + (…)
+ (…) + (…) + (…)
+ (…) + (…) + (…) = ?

Feature Map

1			



Filter

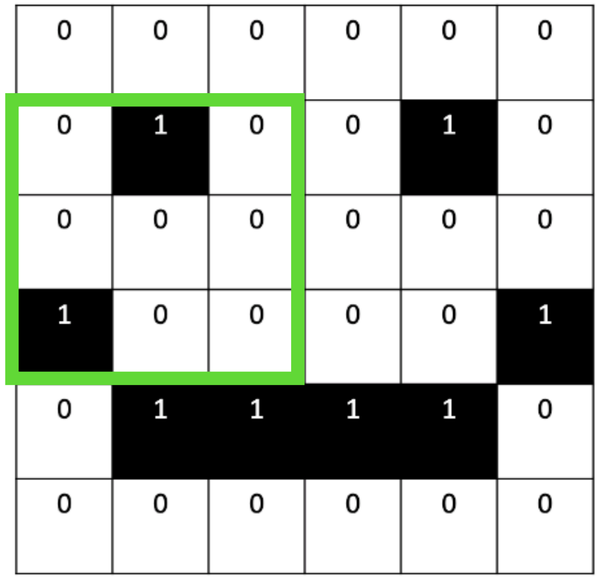
-1	-1	-1
-1	1	-1
-1	-1	-1

Multiply overlapping values and add them together:

(0*-1) + (0*-1) + (0*-1)
+ (1*-1) + (0*1) + (0*-1)
+ (0*-1) + (0*-1) + (0*-1) = -1

Feature Map

1	-1		



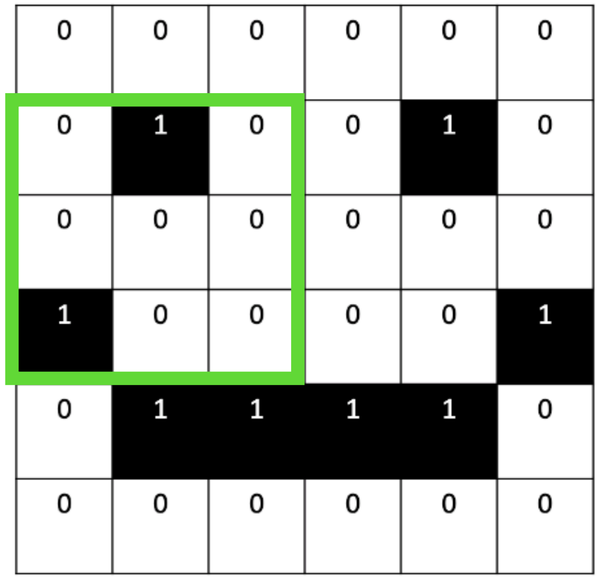
&

Filter

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

Feature Map

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



&

Filter

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

Feature Map

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

We could add another feature map for the mouth here!

Filter

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

...

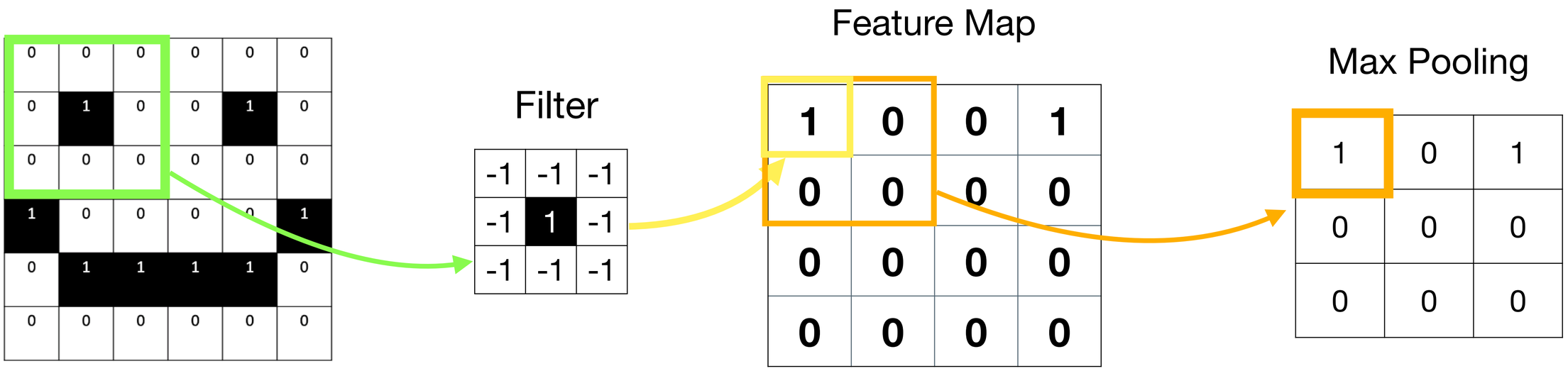
Next steps:

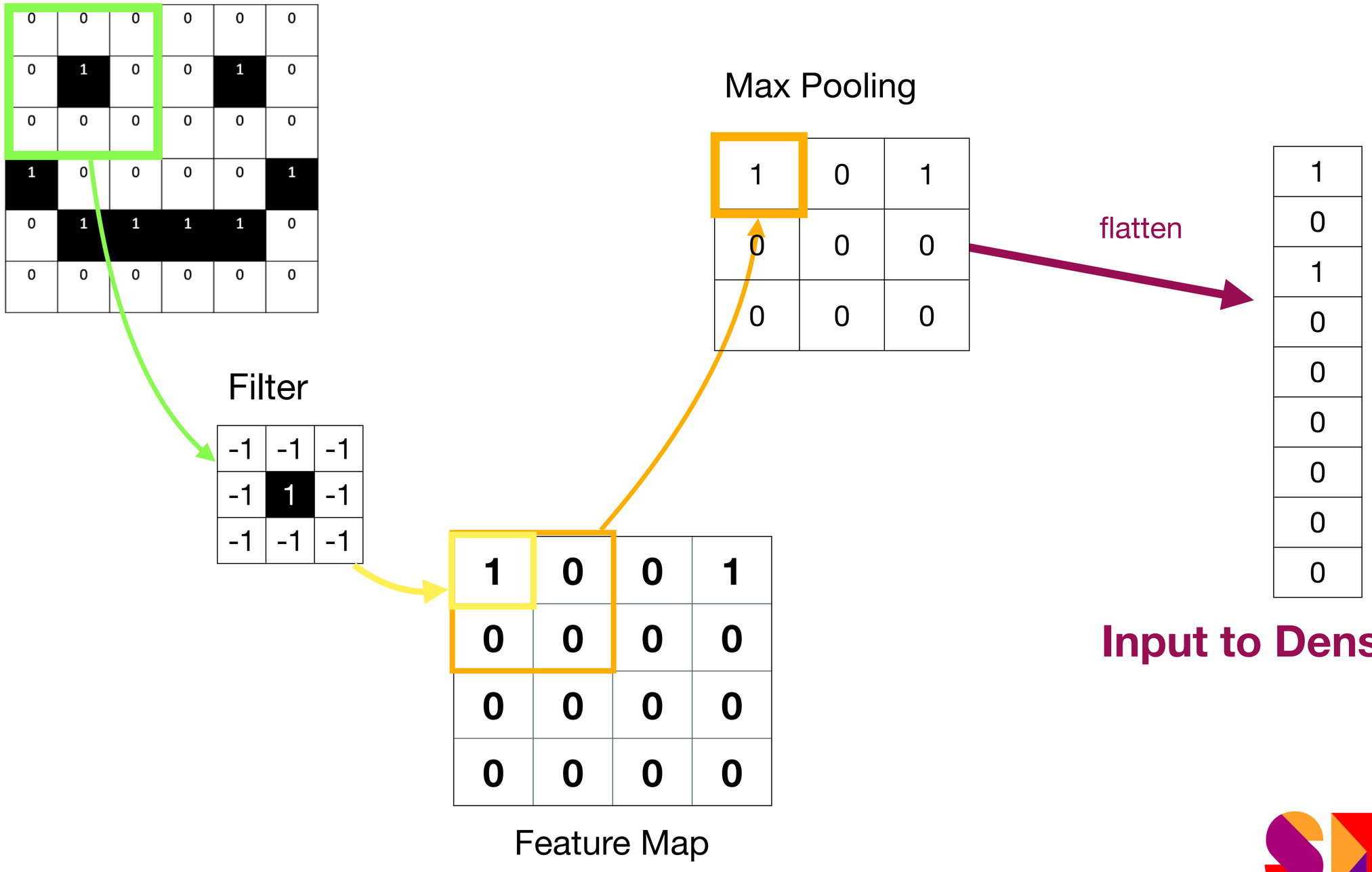
- Feature map gets run through an activation function (ReLu), so all negative values are set to 0.

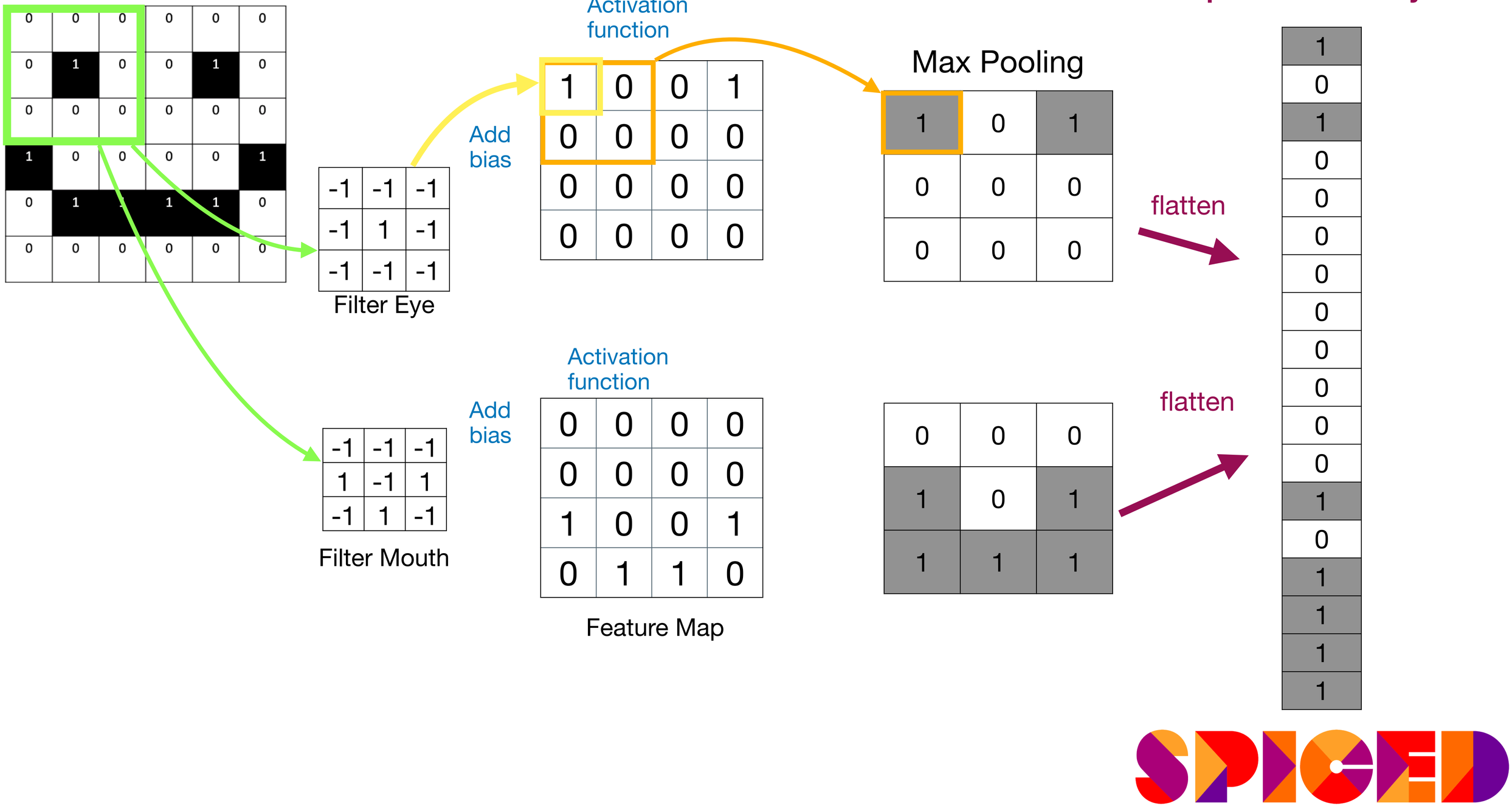
- We apply another filter!

  But here, we simply select the maximum value of the patch, the filter can overlap with former positions or not.

→ **Max Pooling**


Filter
Feature Map
Max Pooling

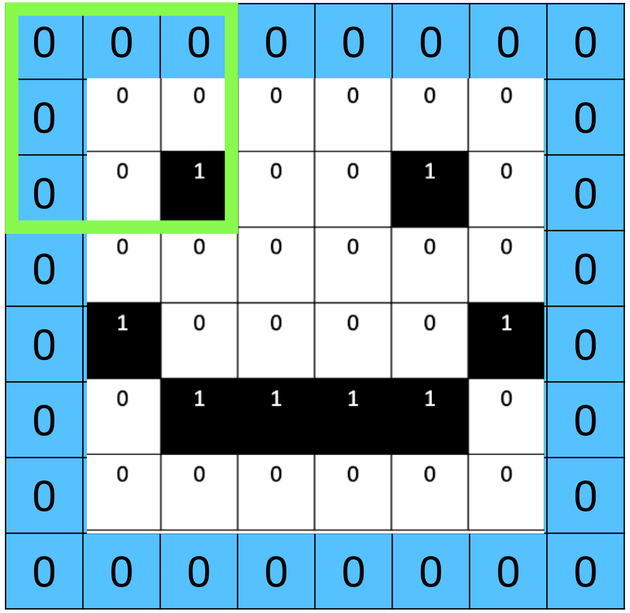

Max Pooling
Filter
flatten
Feature Map
Input to Dense Layer


Input to Dense Layer
Activation function
Add bias
Filter Eye
Max Pooling
flatten
Activation function
Add bias
Filter Mouth
flatten
Feature Map

Most important things to remember:

- We use filters to get feature maps
- Then we use Max-Pooling
- The result can be flattened and fed as input to a Dense Layer

Padding



We add zeros around our image, so that the information of the “border”-pixels is incorporated more.