# Convolution Neural Network (CNN)

# Image Classification

100 x 100

Model

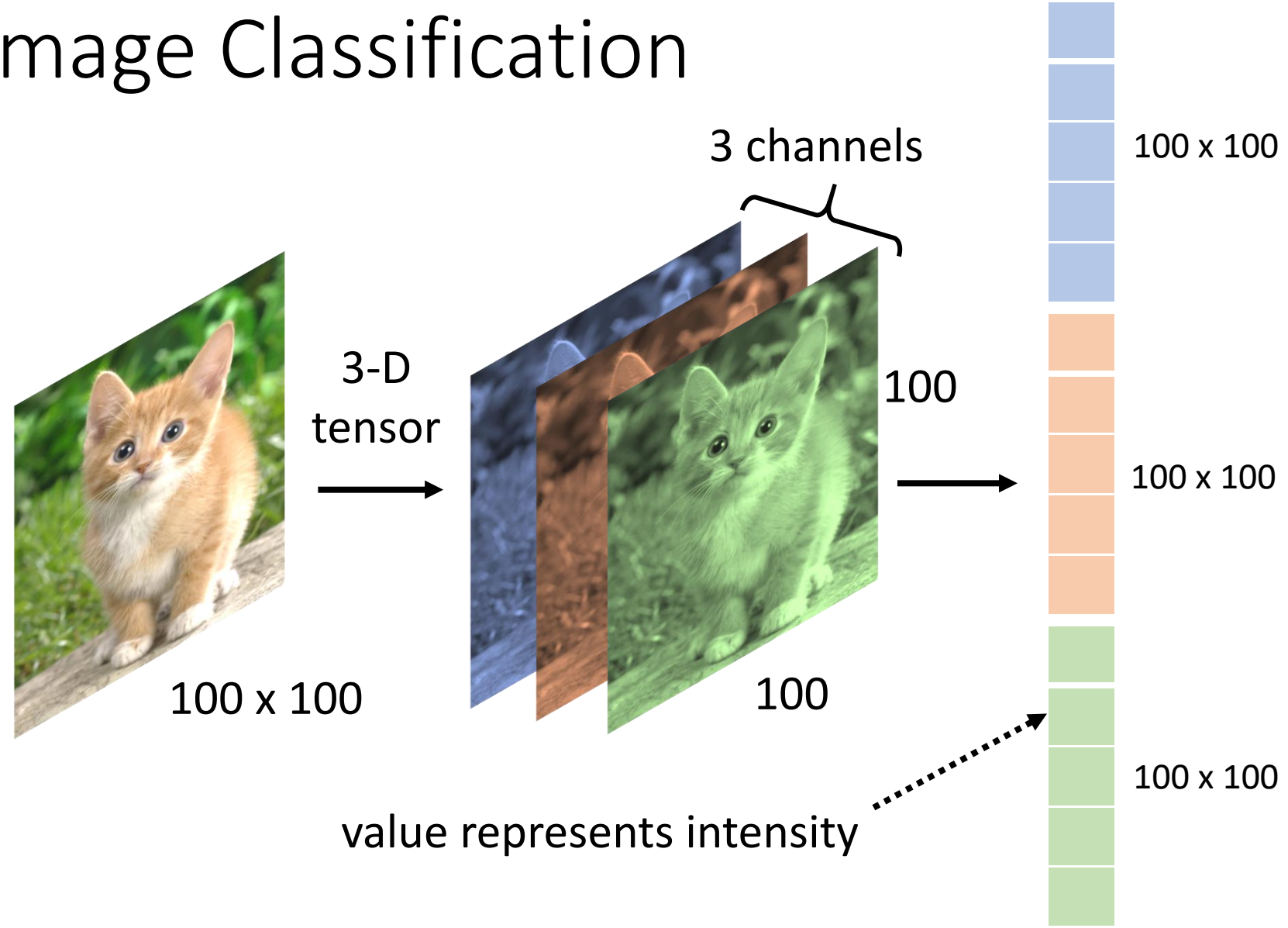
$$\begin{bmatrix} \vdots \\ 0.2 \\ 0.7 \\ 0.1 \\ \vdots \end{bmatrix} \quad y'$$

$$\begin{matrix} \\ \text{dog} \\ \text{cat} \\ \text{tree} \\ \\ \end{matrix} \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \quad \hat{y}$$

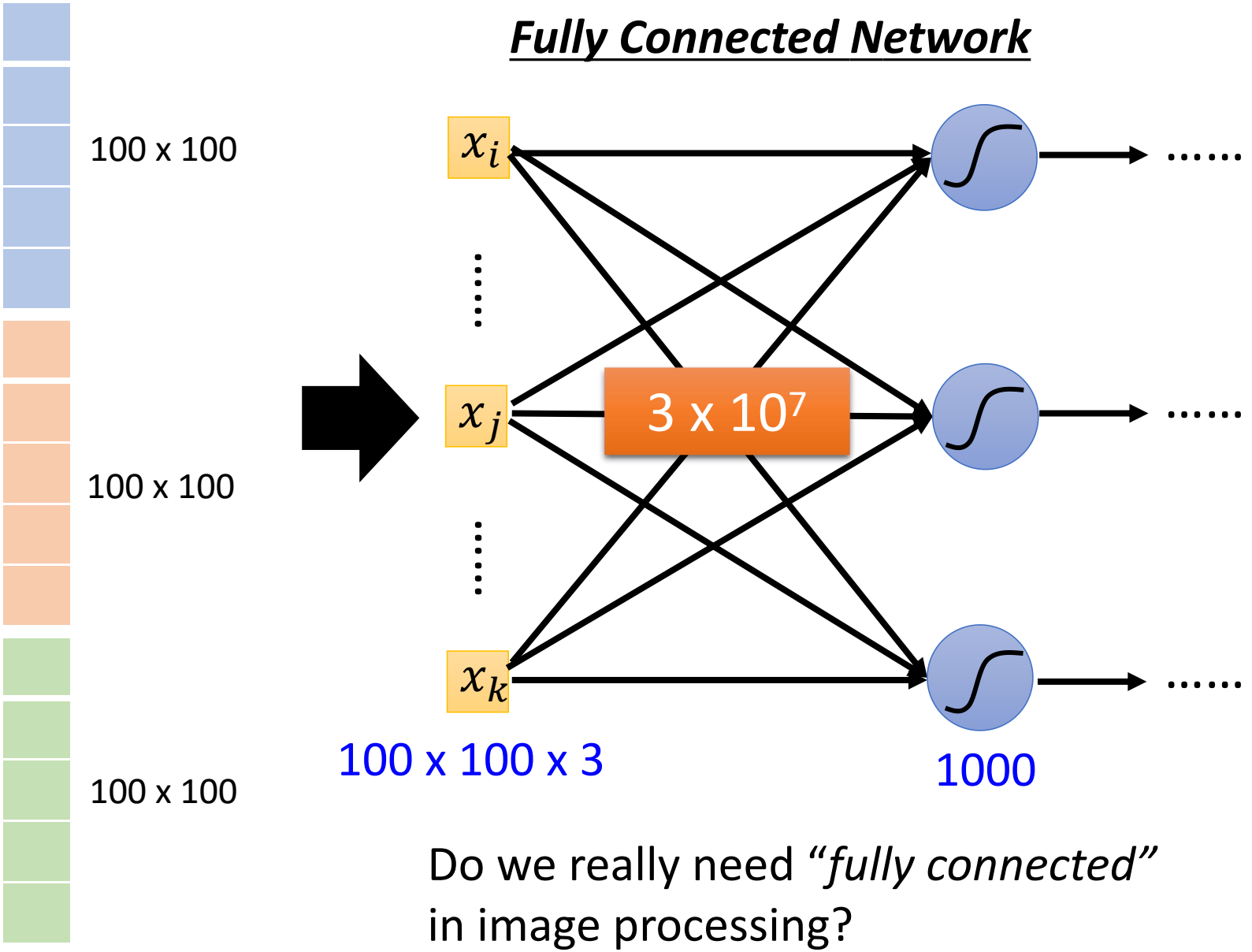
Cross entropy

(All the images to be classified have the same size.)

# Image Classification

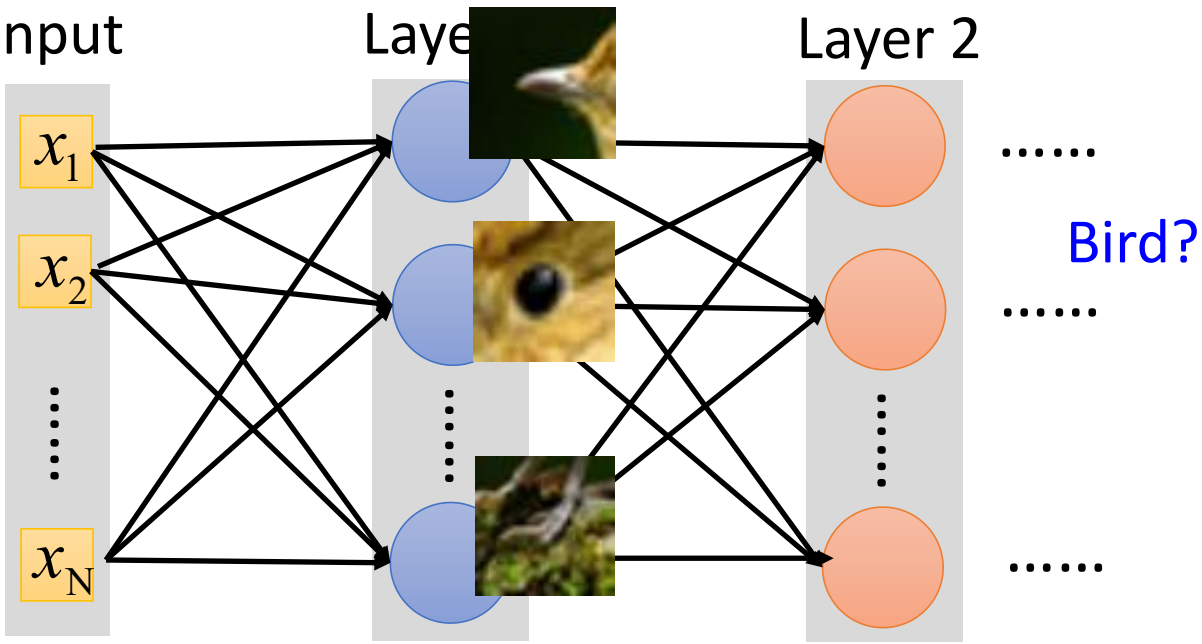


# ***Fully Connected Network***




Do we really need *"fully connected"* in image processing?

# Observation 1

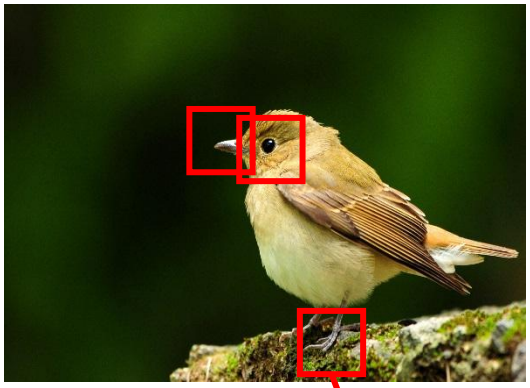


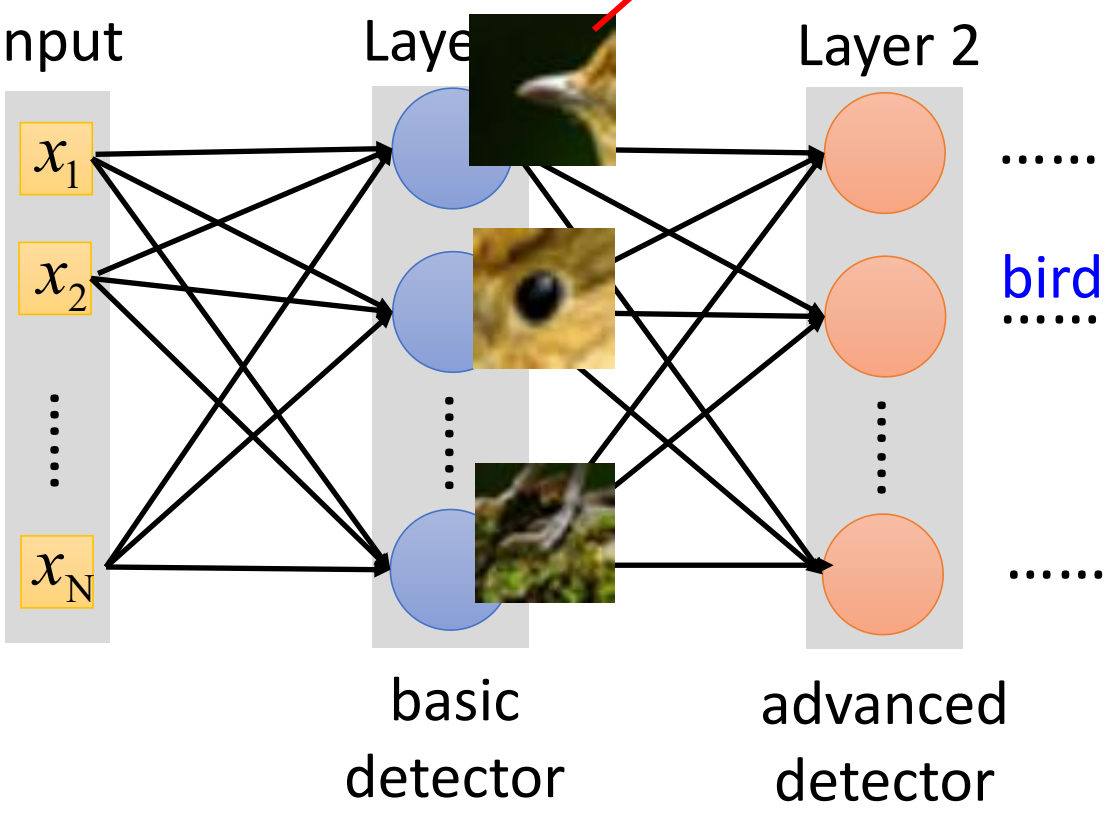

Perhaps human also identify birds in a similar way … ☺

# Observation 1

A neuron does not have to see the whole image.

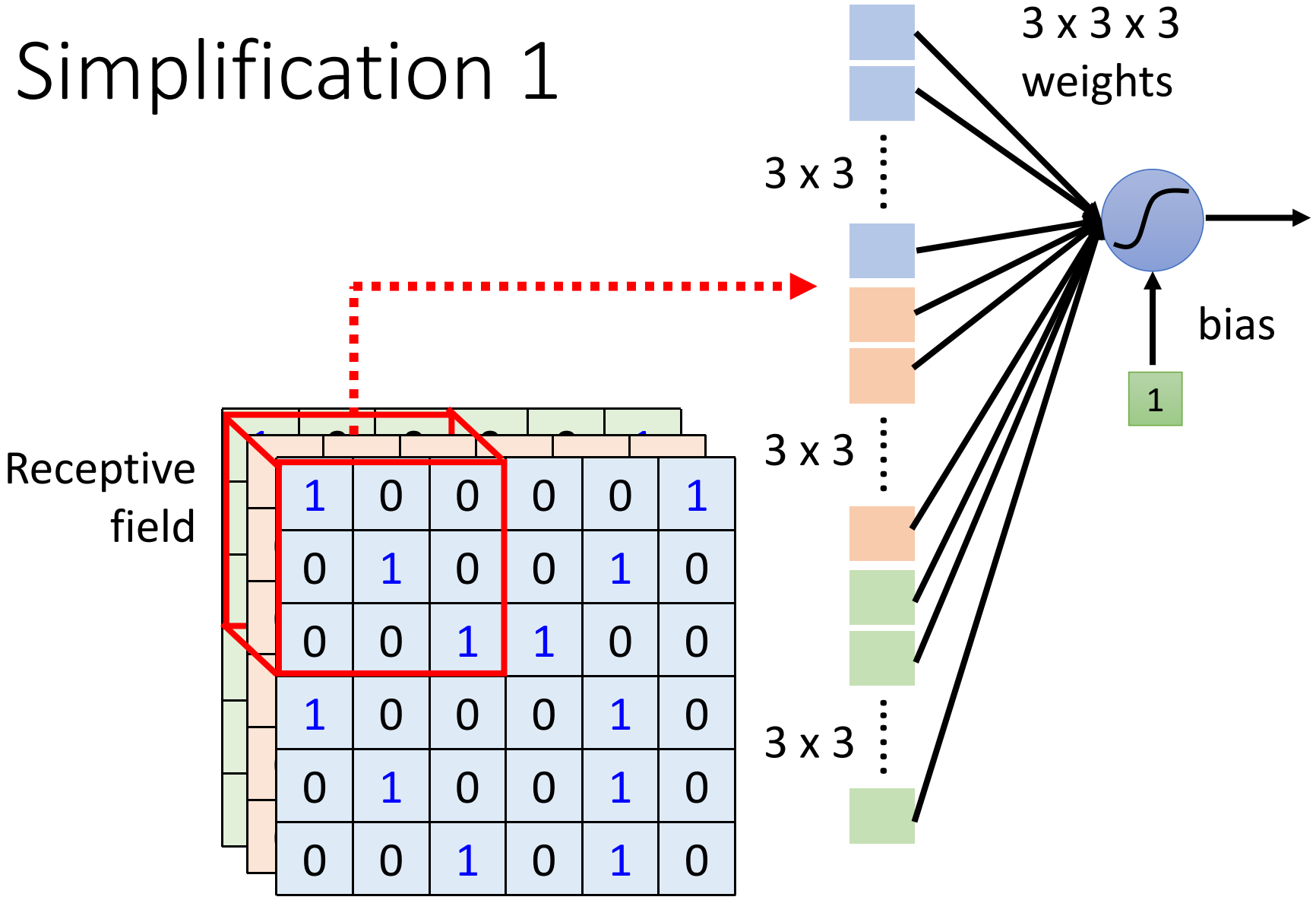
Need to see the whole image?





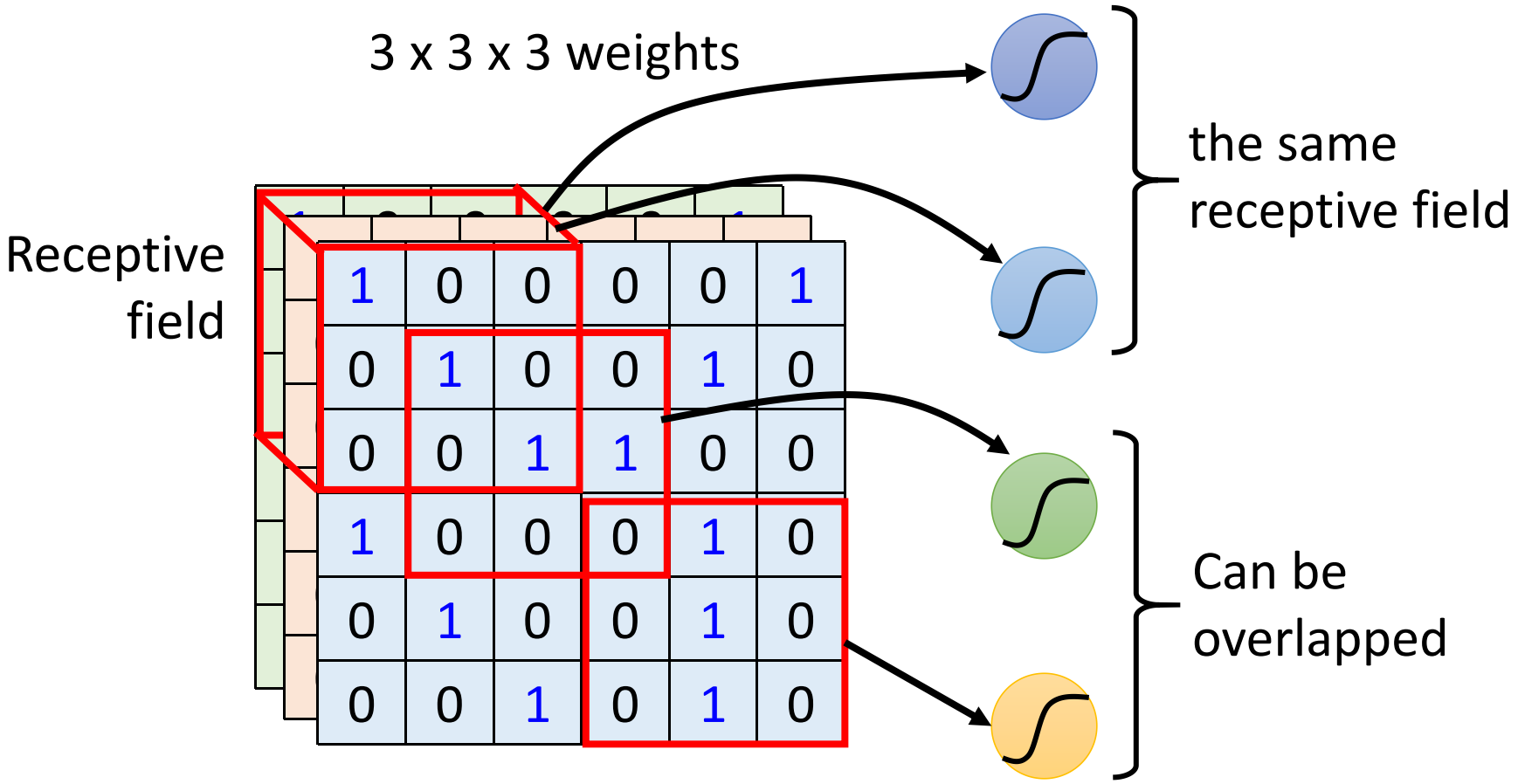

Some patterns are much smaller than the whole image.

# Simplification 1



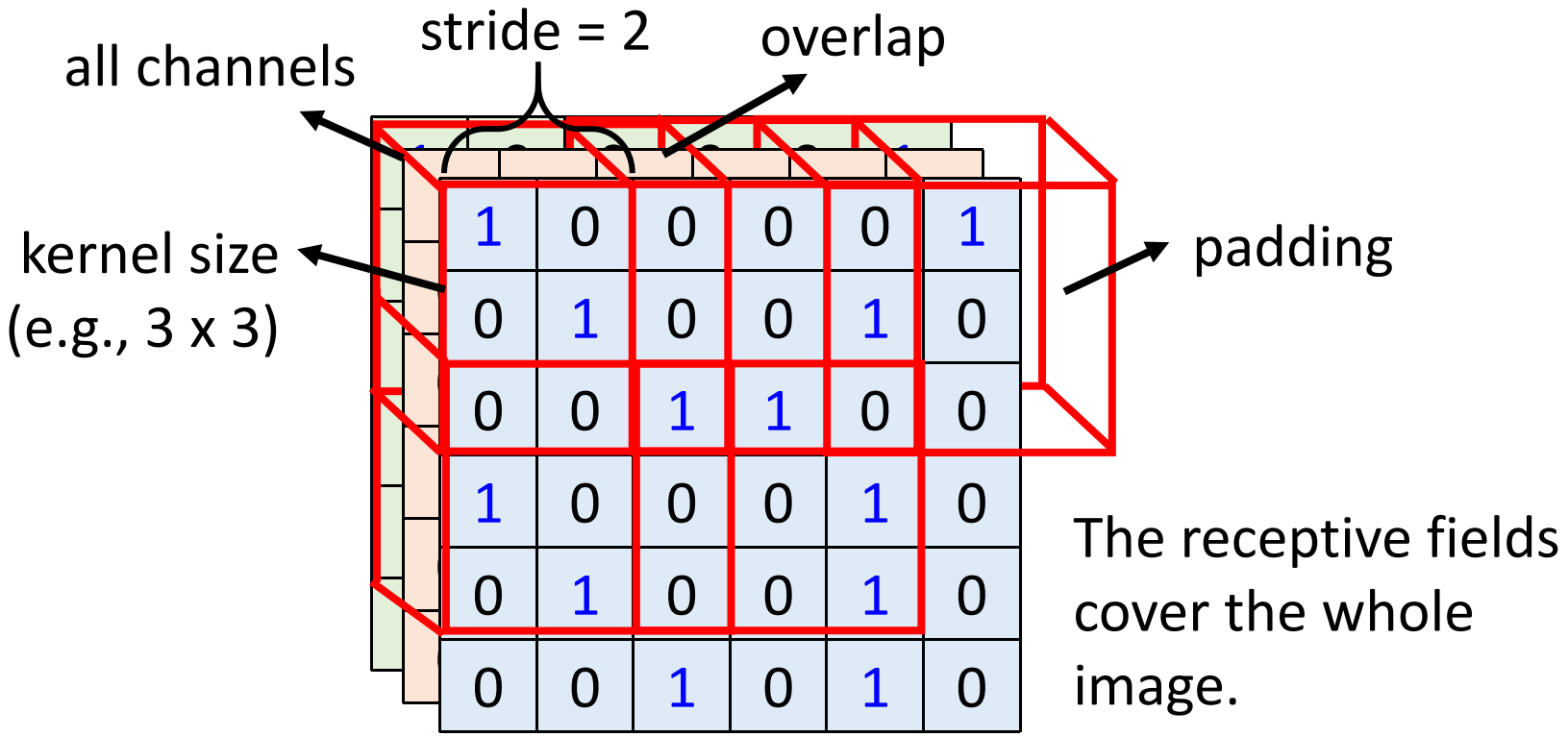
# Simplification 1

- Can different neurons have different sizes of receptive field?
- Cover only some channels?
- Not square receptive field?



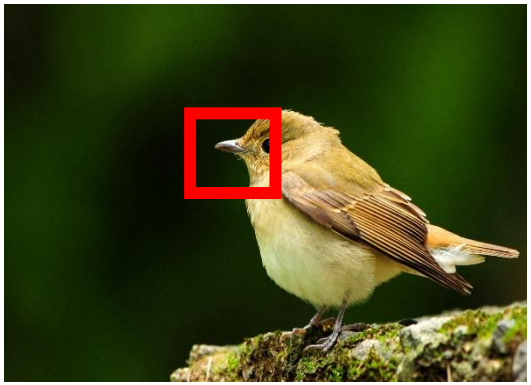
# Simplification 1 – Typical Setting

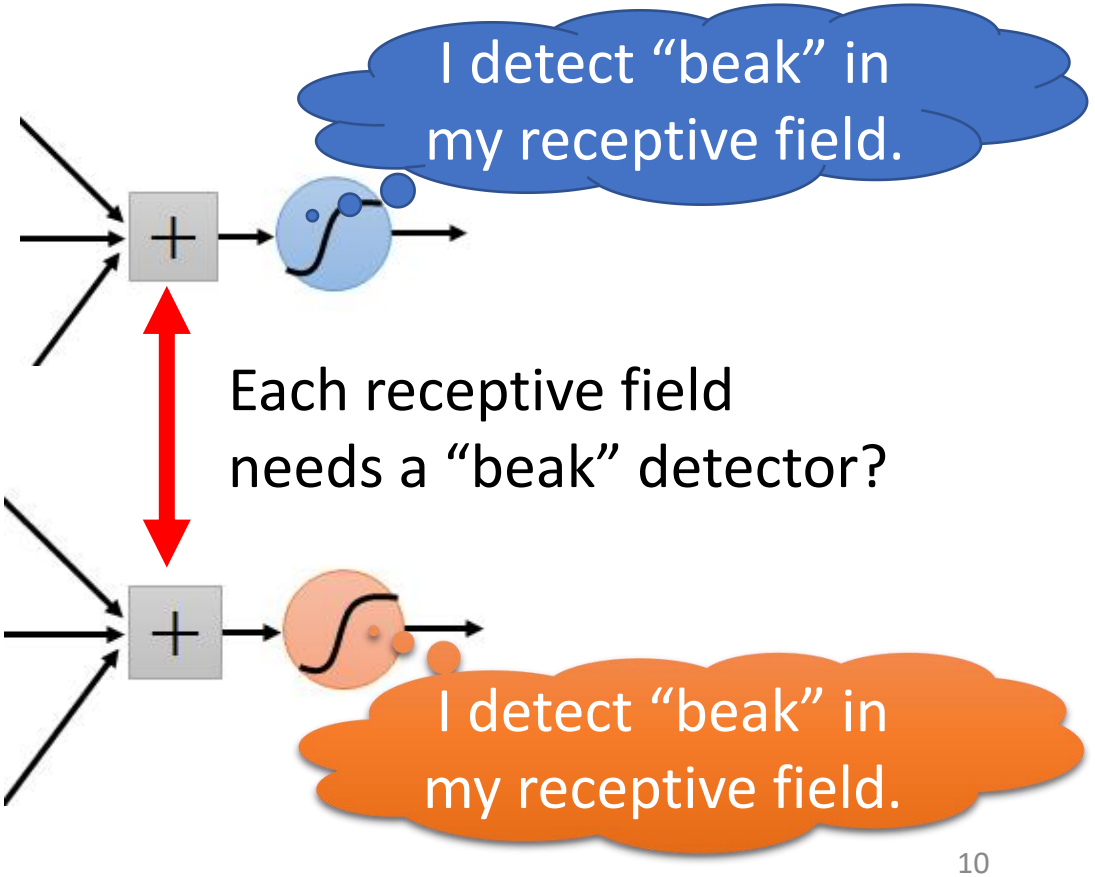
Each receptive field has a set of neurons (e.g., 64 neurons).



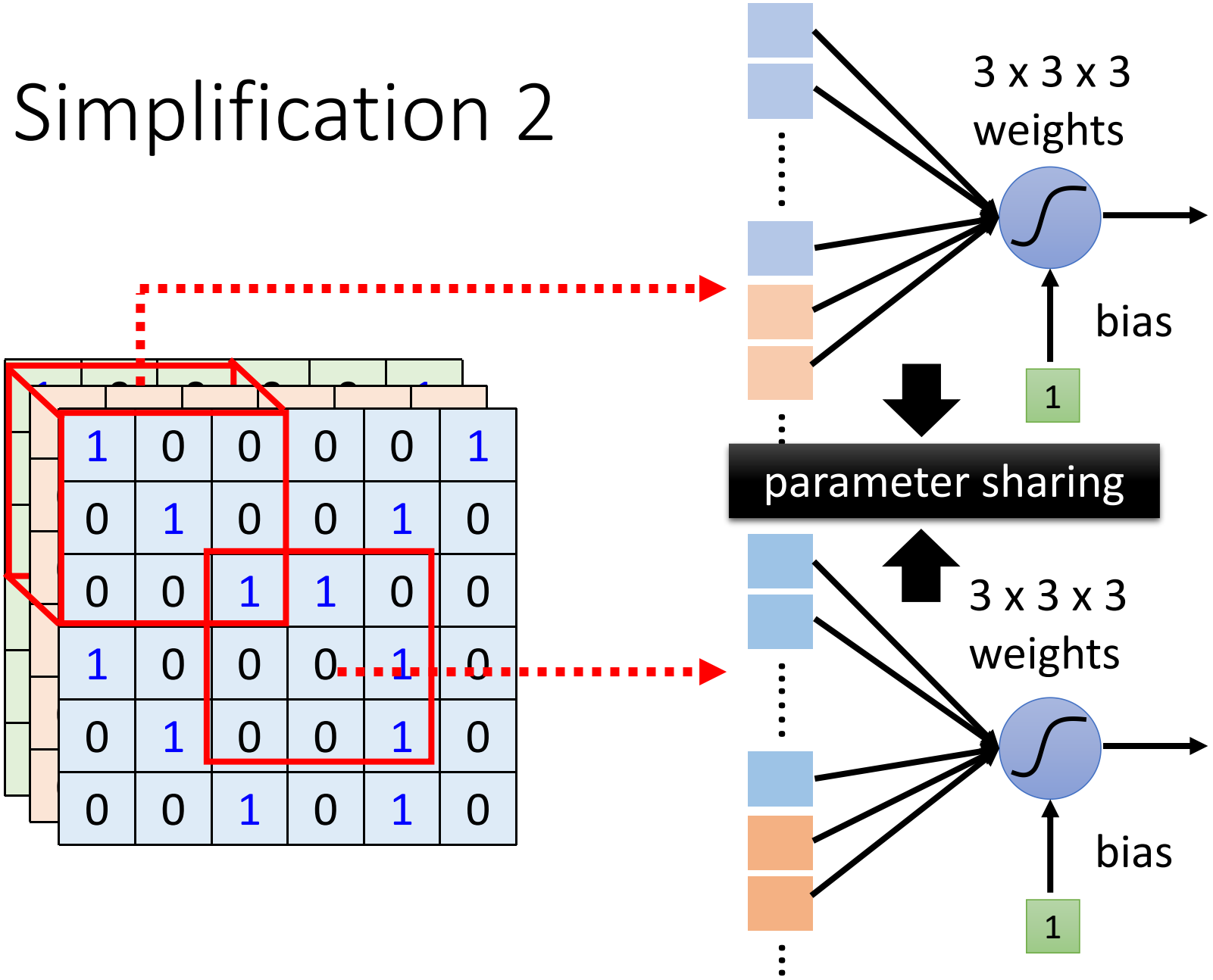
# Observation 2

- The same patterns appear in different regions.

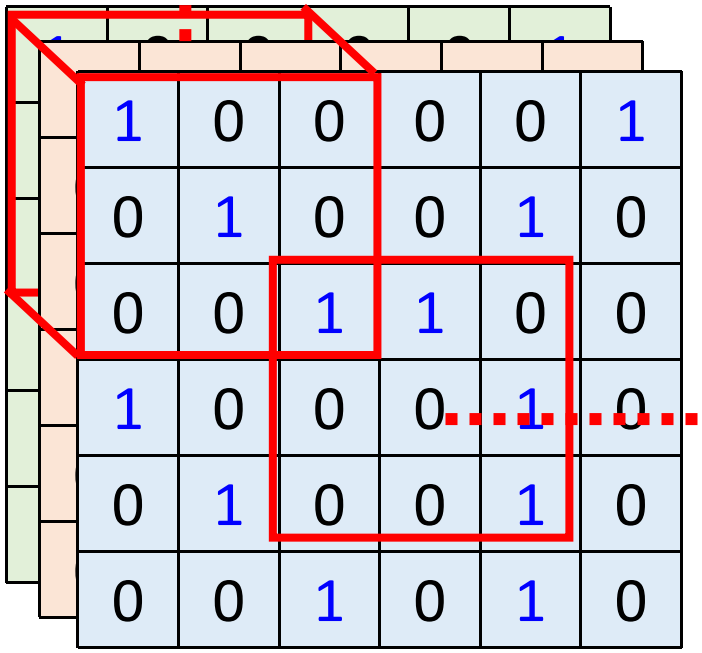


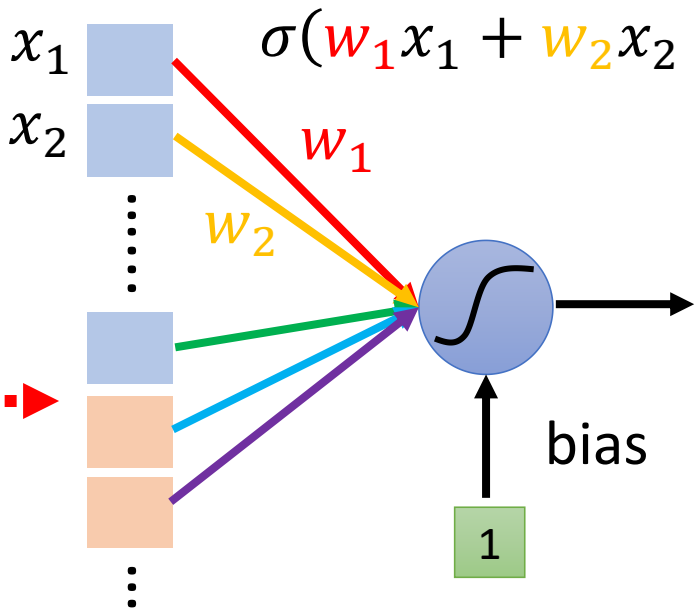


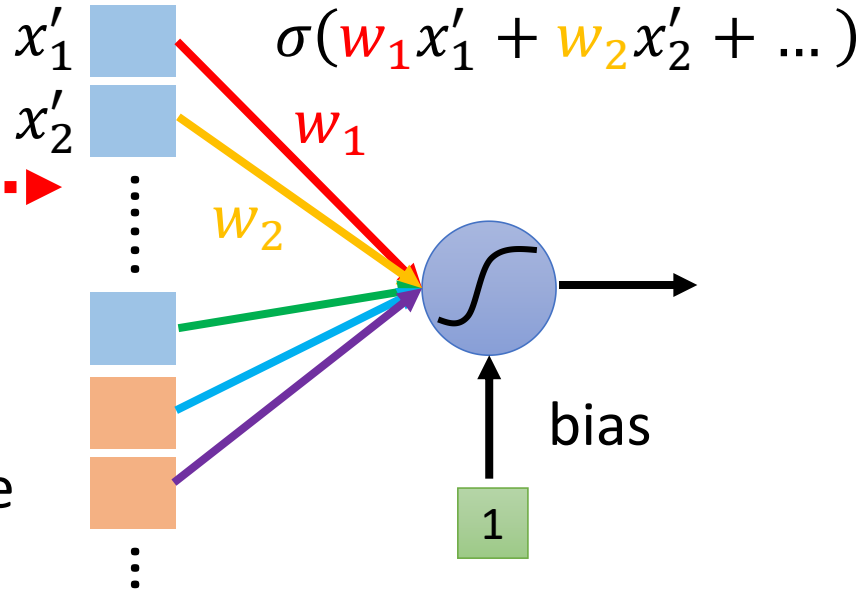
# Simplification 2



# Simplification 2



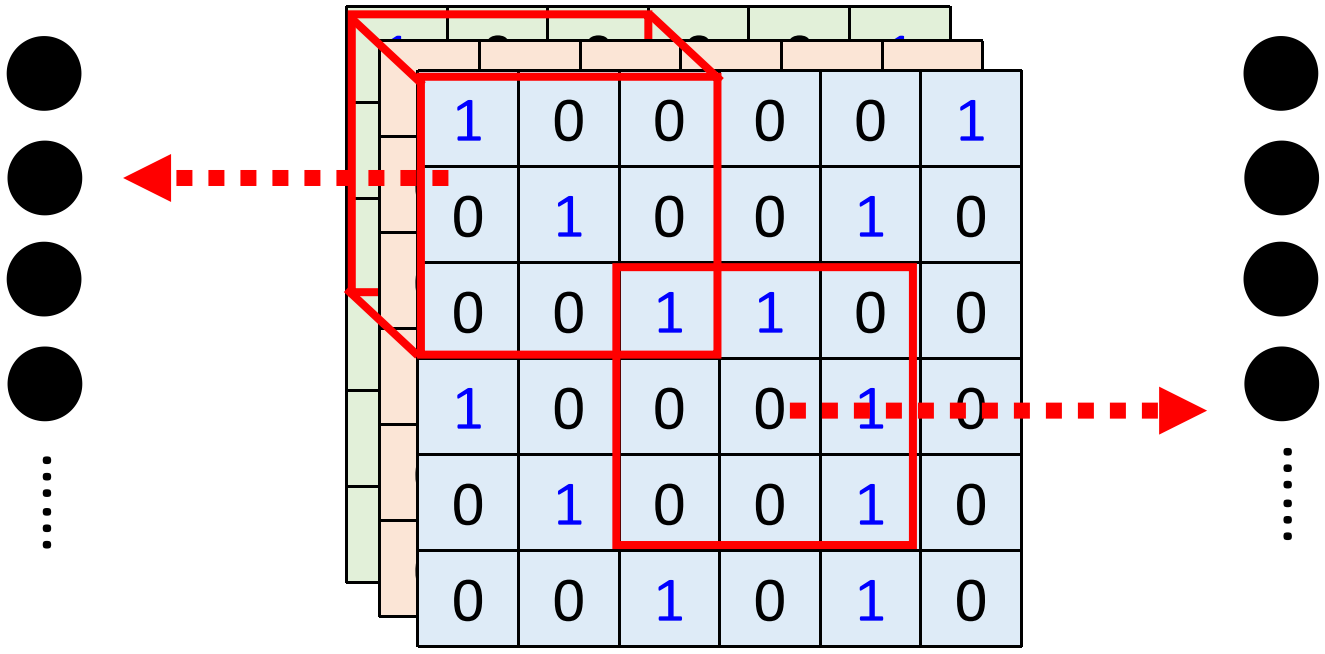




Two neurons with the same receptive field would not share parameters.

# Simplification 2 – Typical Setting

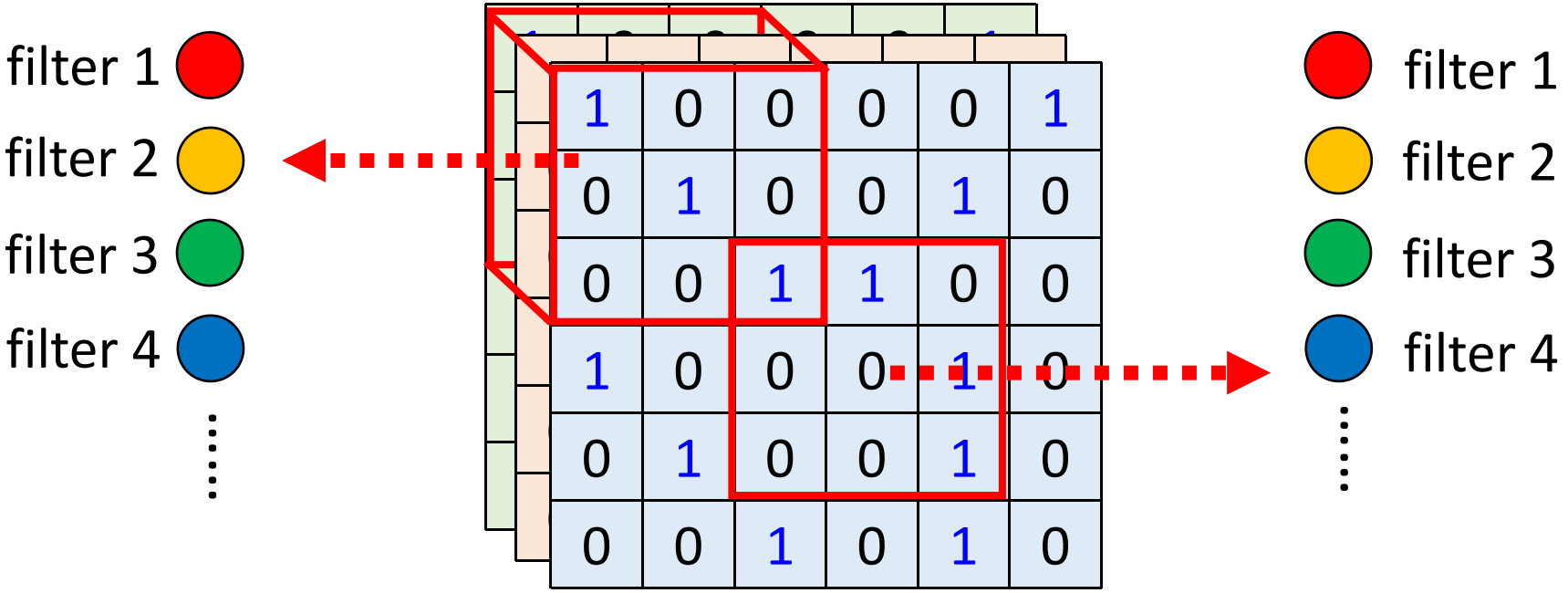
Each receptive field has a set of neurons (e.g., 64 neurons).



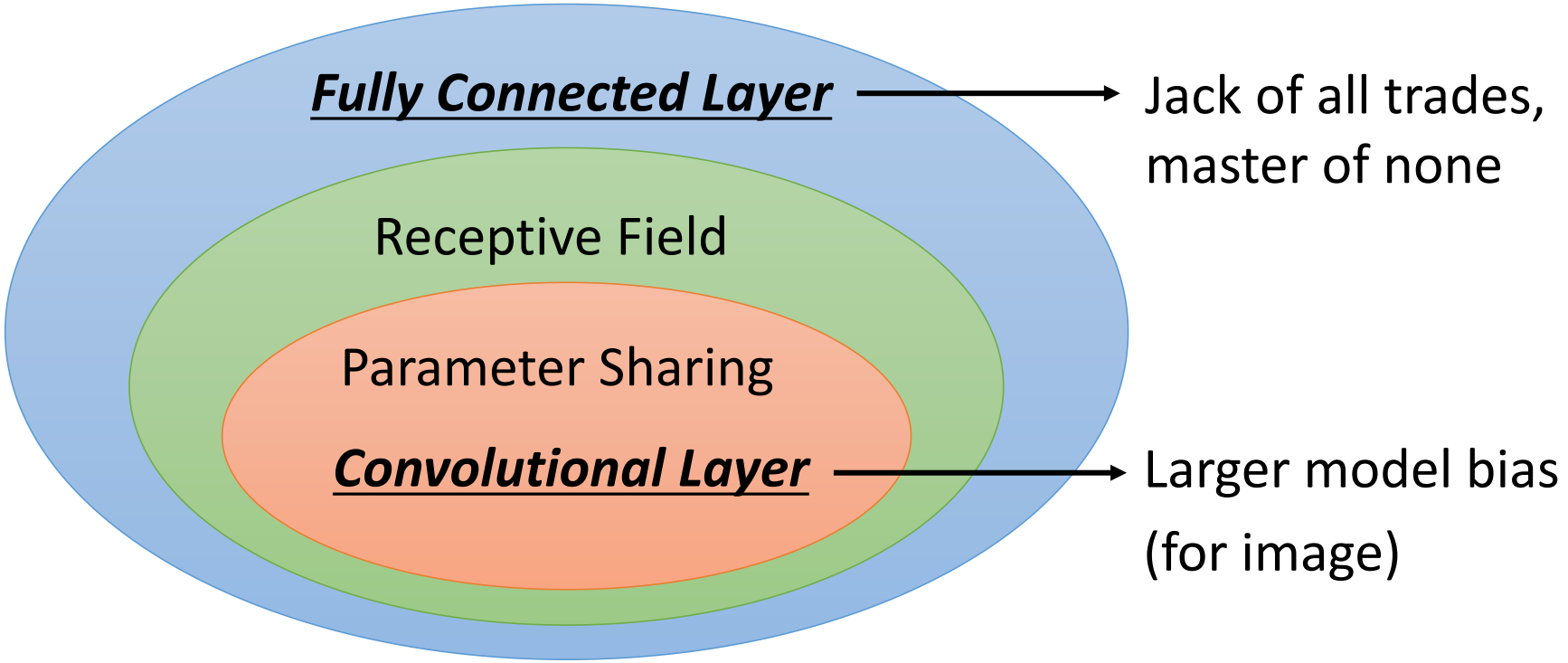
# Simplification 2 – Typical Setting

Each receptive field has a set of neurons (e.g., 64 neurons).

Each receptive field has the neurons with the same set of parameters.



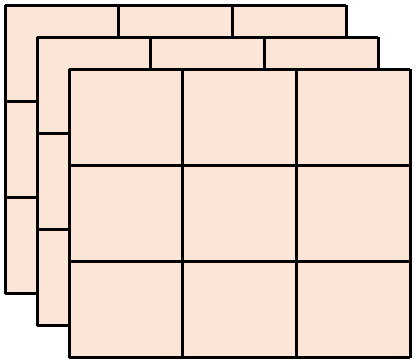
# Benefit of Convolutional Layer



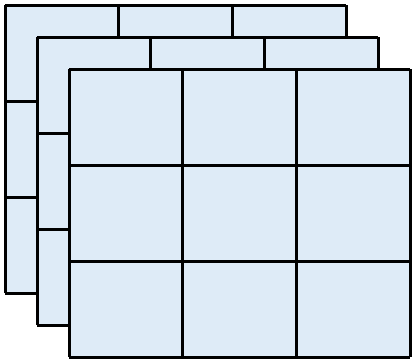

- Some patterns are much smaller than the whole image.
- The same patterns appear in different regions.

# Convolutional Layer

Convolution



Filter 1
3 x 3 x channel
tensor



Filter 2
3 x 3 x channel
tensor

channel = 3 (colorful)

channel = 1 (black and white)

Each filter detects a small pattern (3 x 3 x channel).

# Convolutional Layer

Consider channel = 1
(black and white image)

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

6 x 6 image

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

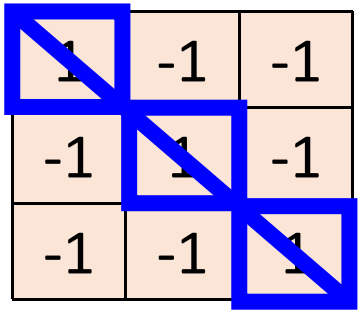
Filter 1

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

Filter 2

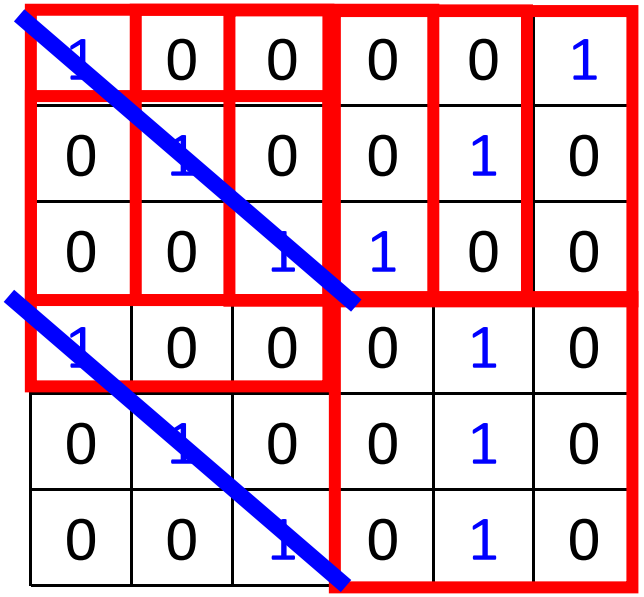
(The values in the filters are unknown parameters.)

# Convolutional Layer

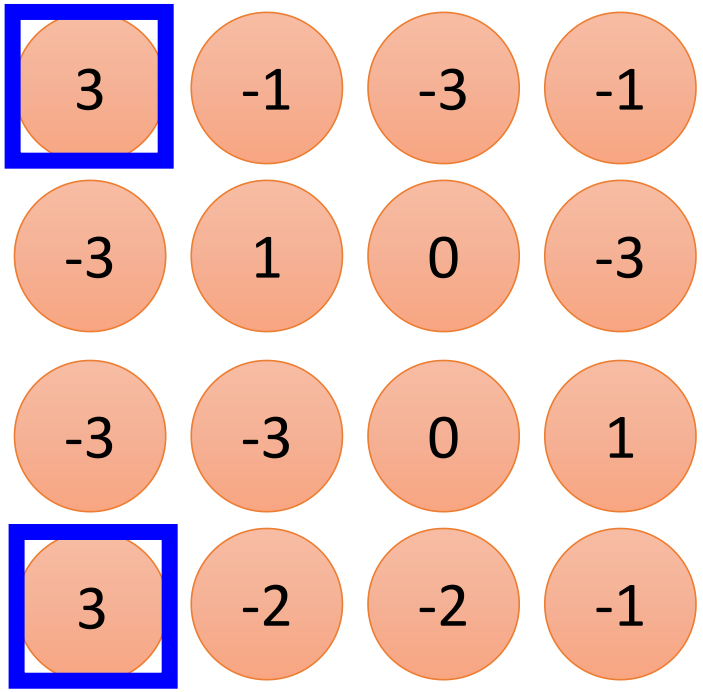



Filter 1

stride=1




6 x 6 image



# Convolutional Layer

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

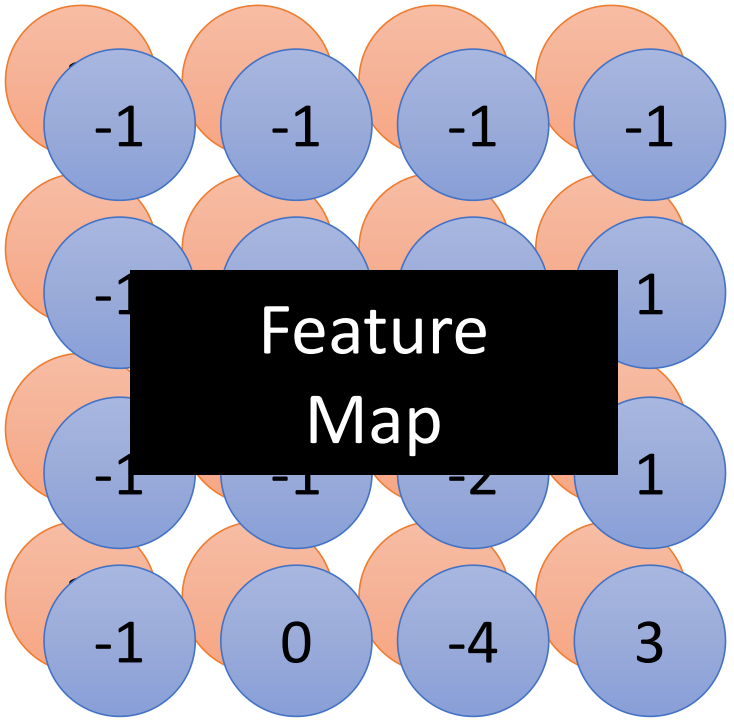
Filter 2

stride=1

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

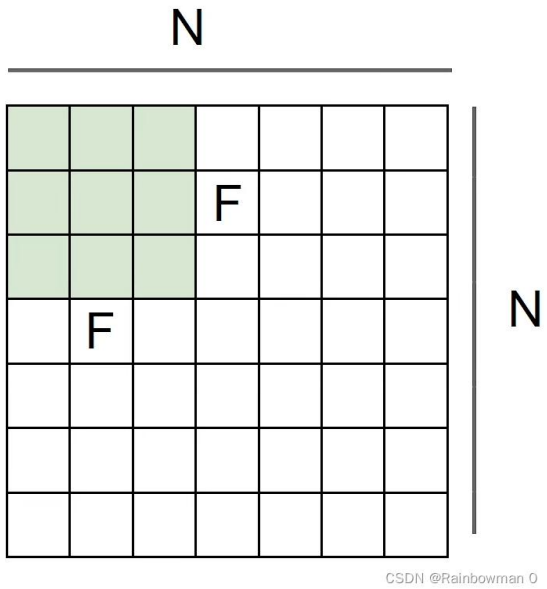
6 x 6 image

Do the same process for every filter



# Quiz

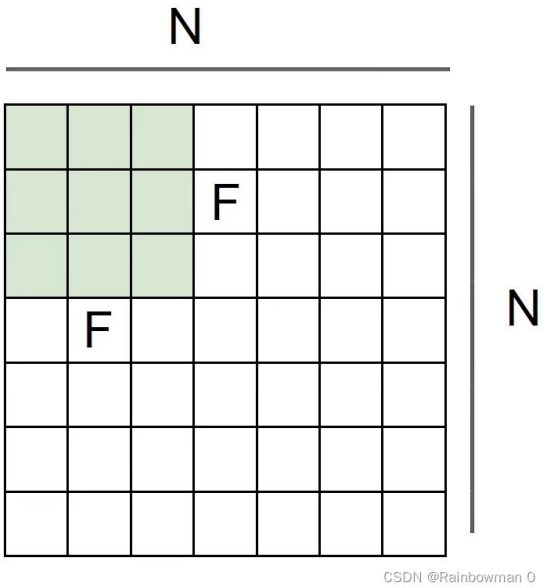
- Given a N*N size picture, filter size F*F,  stride S, what is the size of output feature map? (No padding)



# Quiz

- Given a N*N size picture, filter size F*F,  stride S, what is the size of output feature map?

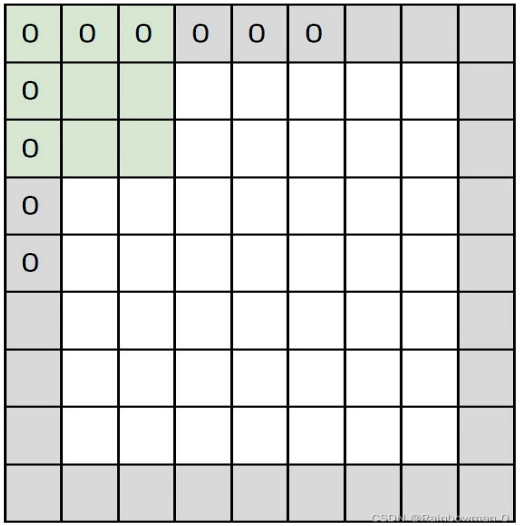
$$outputsize = \frac{N - F}{S} + 1$$



# Quiz

- Given a N*N size picture, filter size F*F,  stride S, padding size P, what is the size of output feature map?

(N-F+2P)/S+1

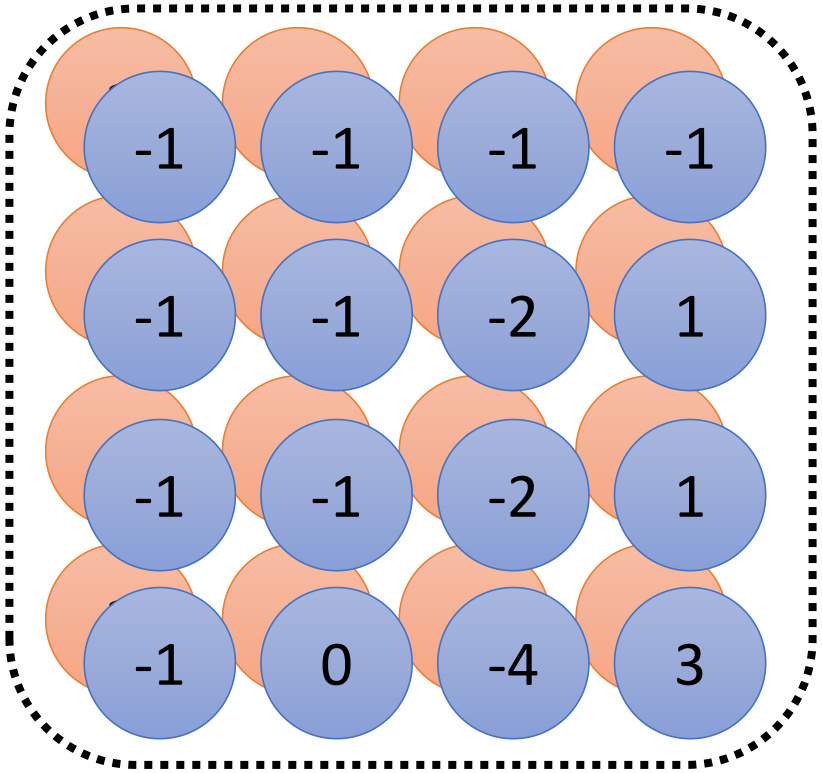


# ***Convolutional Layer***

64 filters

Convolution

Convolution



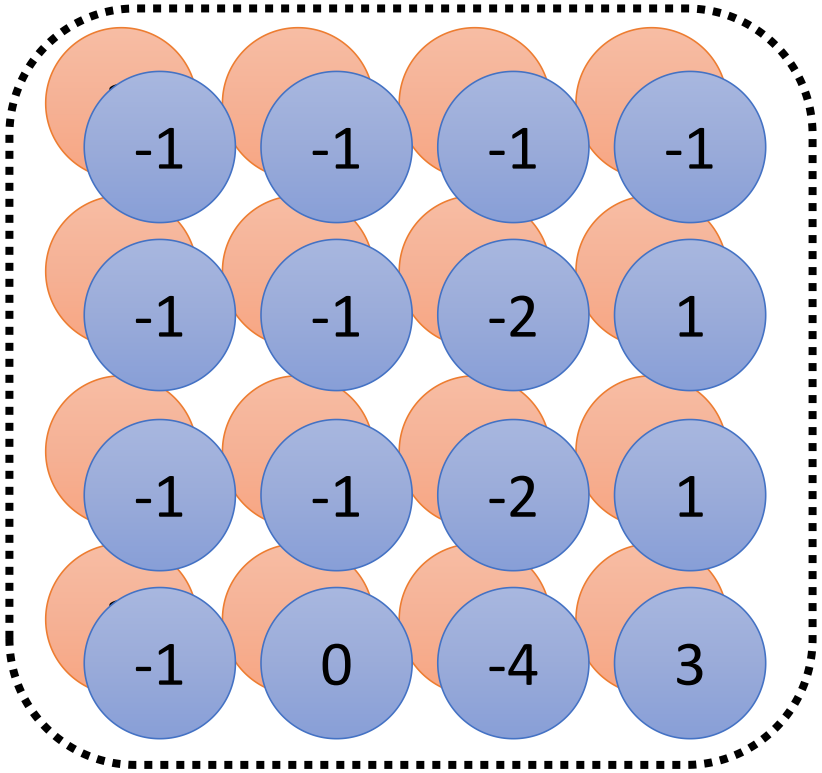

"Image" with 64 channels

# ***Multiple Convolutional Layers***

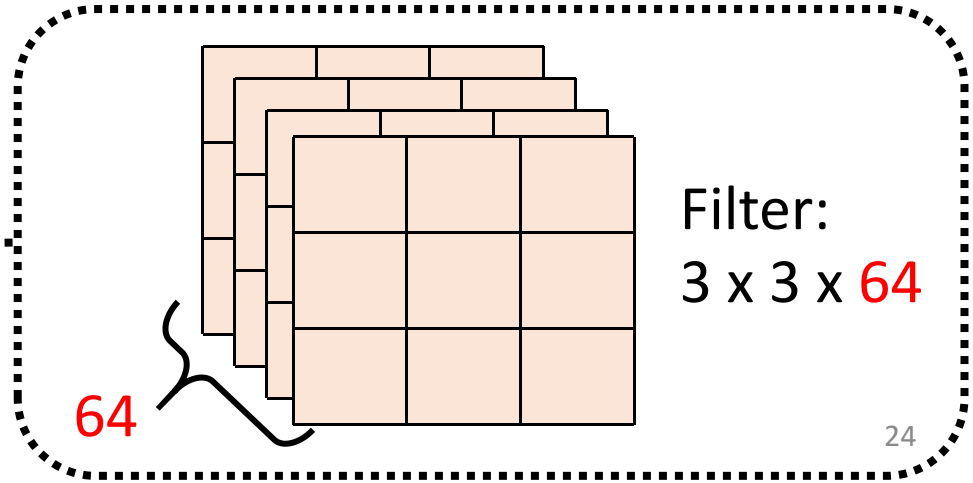
64 filters

Convolution

Convolution




"Image" with 64 channels



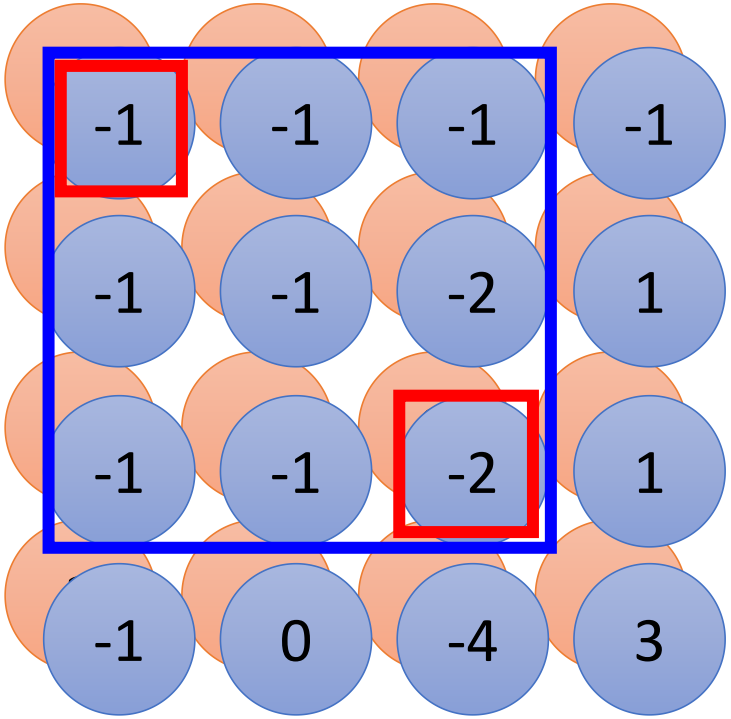
# ***Multiple Convolutional Layers***

64 filters

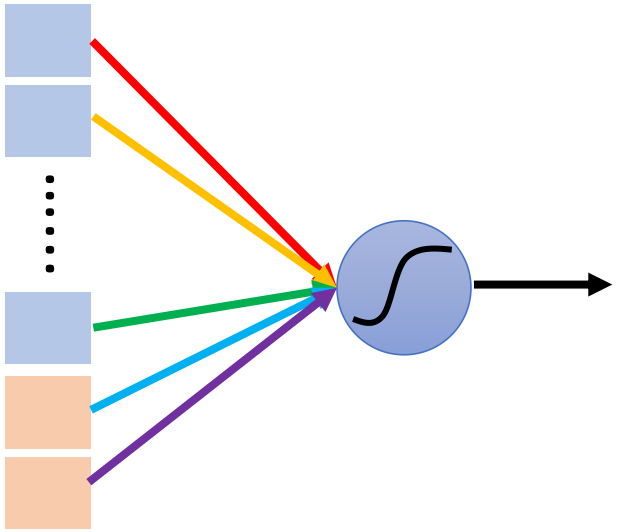
Convolution

Convolution

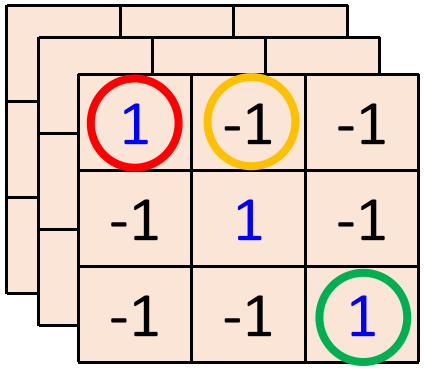
| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |



# Comparison of Two Stories



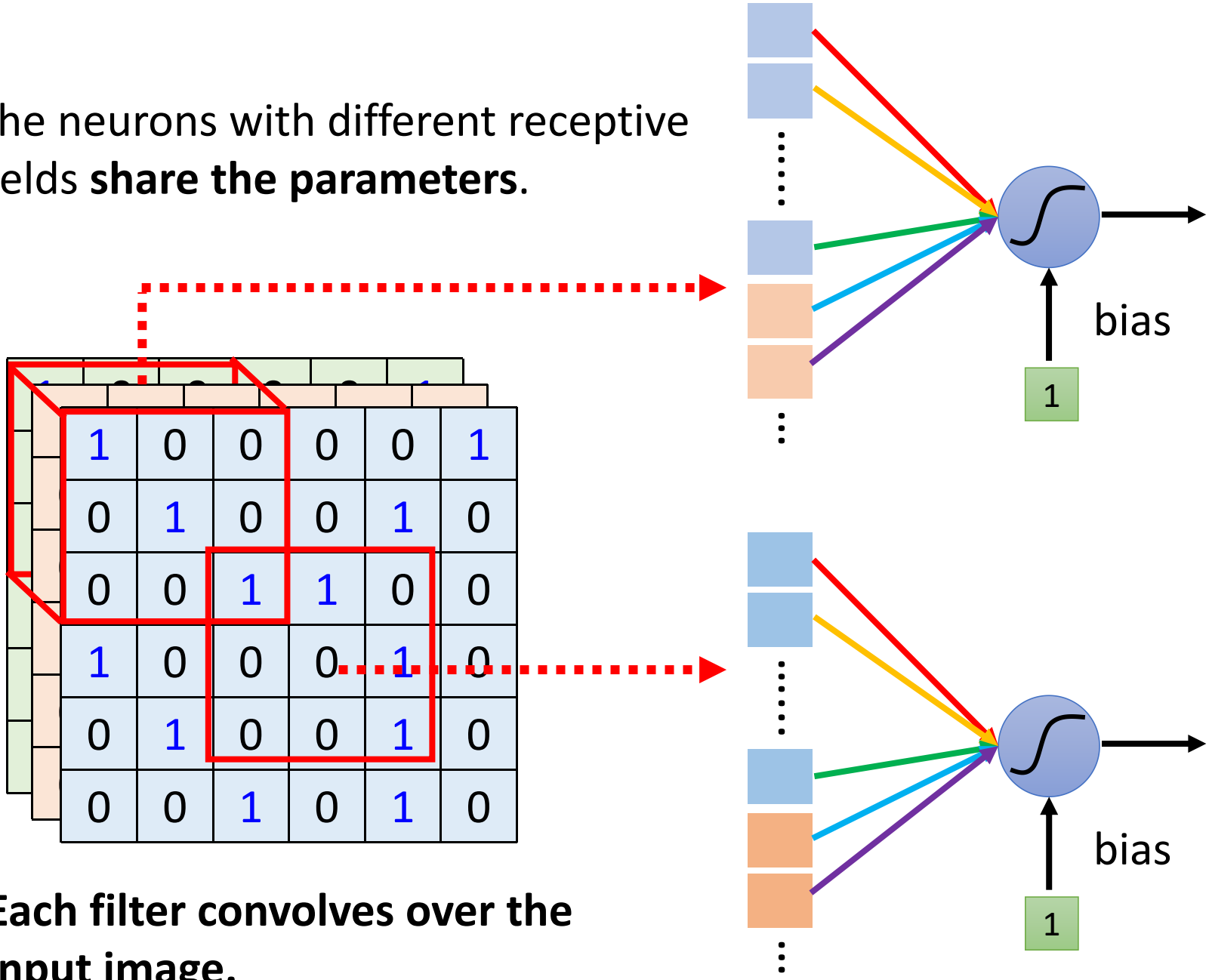
Receptive field




Filter
3 x 3 x channel tensor

(ignore bias in this slide)

The neurons with different receptive fields **share the parameters**.




**Each filter convolves over the input image.**

# Convolutional Layer

| ***Neuron Version Story*** | ***Filter Version Story*** |
| --- | --- |
| Each neuron only considers a receptive field. | There are a set of filters detecting small patterns. |
| The neurons with different receptive fields share the parameters. | Each filter convolves over the input image. |

They are the same story.

# Observation 3

- Subsampling the pixels will not change the object

bird

subsampling

bird

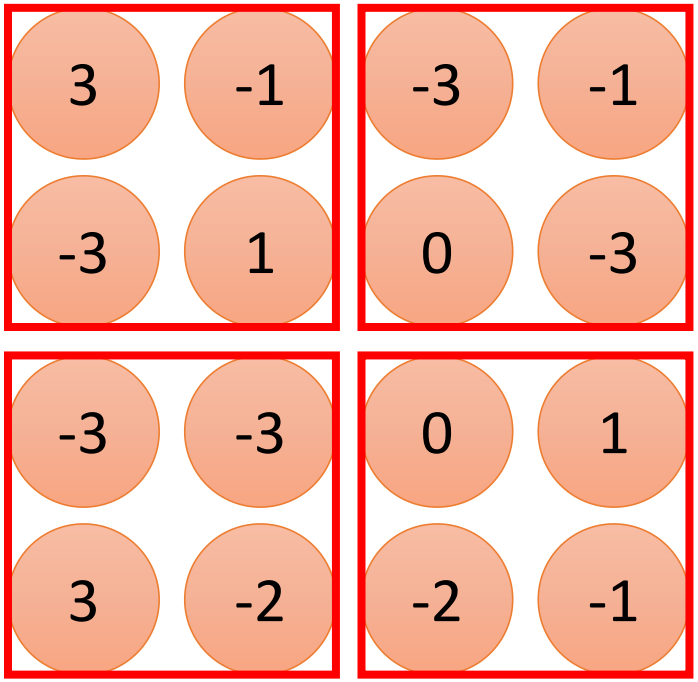
# Pooling – Max Pooling

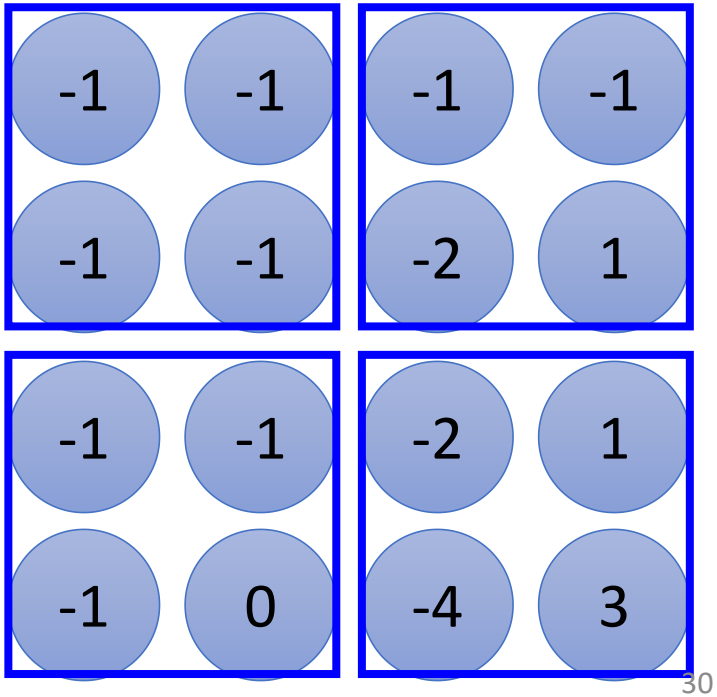
| | | |
|---|---|---|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

Filter 1

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

Filter 2



# ***Convolutional Layers + Pooling***

Repeat

Convolution

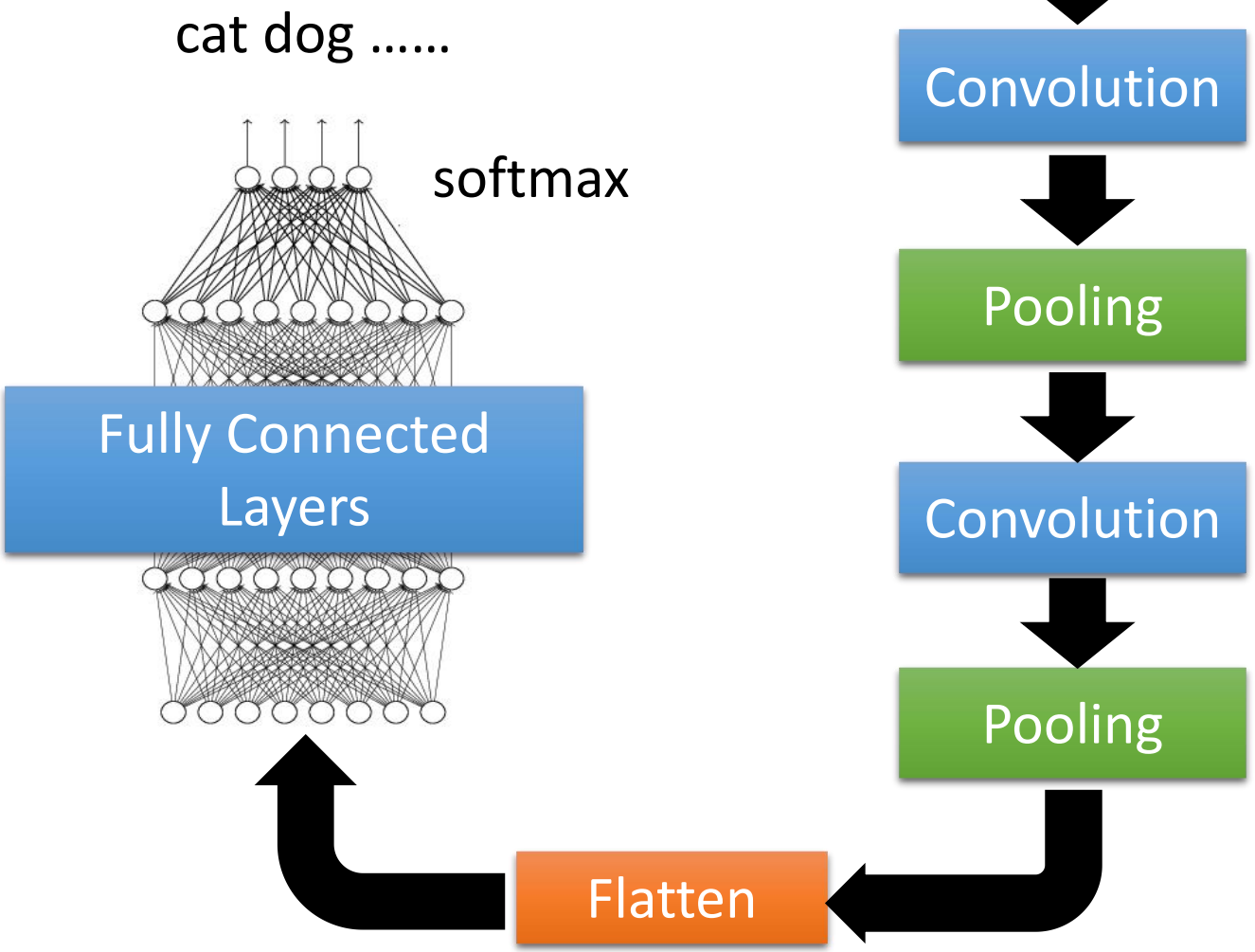
Pooling

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

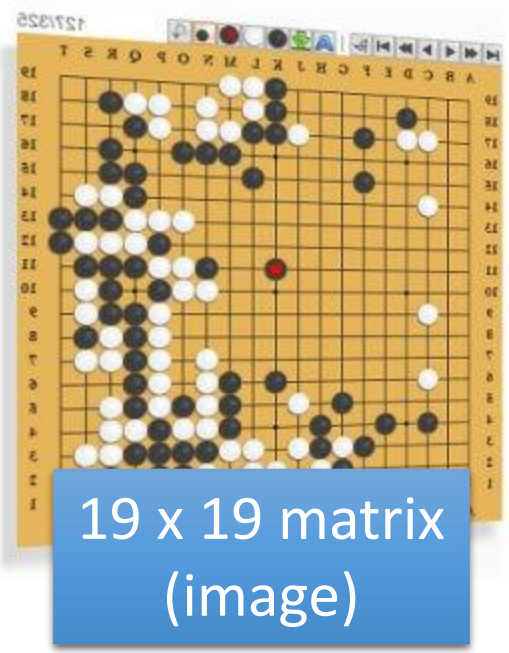
"Image" with 64 channels

-1 1
0 3

# The whole CNN



# Application: Playing Go



19 x 19 matrix (image)

Network

Next move (19 x 19 positions)

19 x 19 classes

48 channels in Alpha Go

Black: 1

white: -1

none: 0

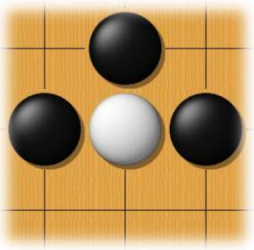
Fully-connected network can be used

But CNN performs much better.

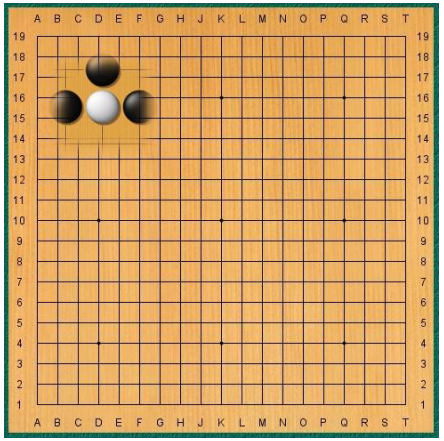
# Why CNN for Go playing?

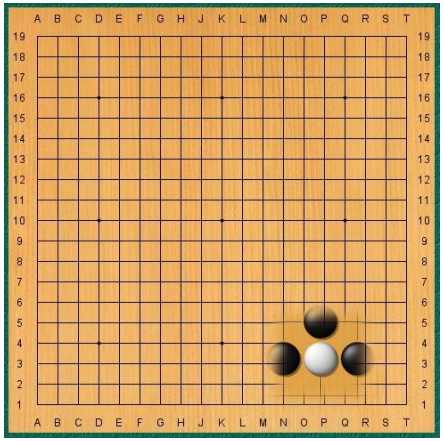
- Some patterns are much smaller than the whole image

  Alpha Go uses 5 x 5 for first layer



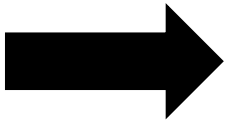
- The same patterns appear in different regions.





# Why CNN for Go playing?

- Subsampling the pixels will not change the object



Pooling

How to explain this???

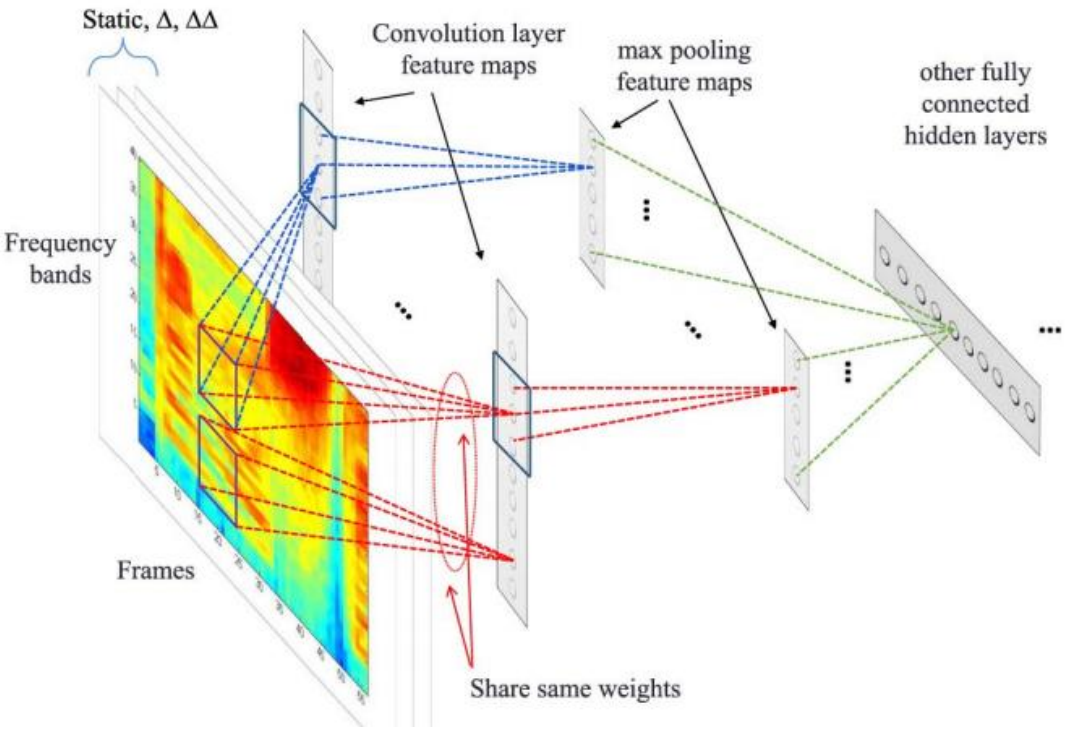
**Neural network architecture.** The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a $23 \times 23$ image, then convolves $k$ filters of kernel size $5 \times 5$ with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a $21 \times 21$ image, then convolves $k$ filters of kernel size $3 \times 3$ with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size $1 \times 1$ with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Tabl 256 and 384 filters

Alpha Go does not use Pooling ……

# More Applications

## Speech

https://dl.acm.org/doi/10.1109/TASLP.2014.2339736




## Natural Language Processing

https://www.aclweb.org/anthology/S15-2079/