人工智能技术及应用

Artificial Intelligence and Application

Classification

Classification

x → Function → Class n

- Credit Scoring
  - Input: income, savings, profession, age, past financial history ……
  - Output: accept or refuse
- Medical Diagnosis
  - Input: current symptoms, age, gender, past medical history ……
  - Output: which kind of diseases
- Handwritten character recognition

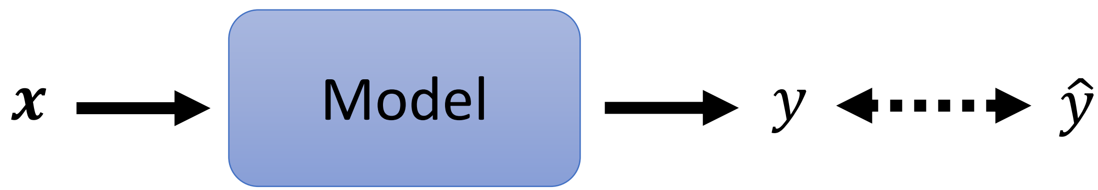
Input: 

output: 金

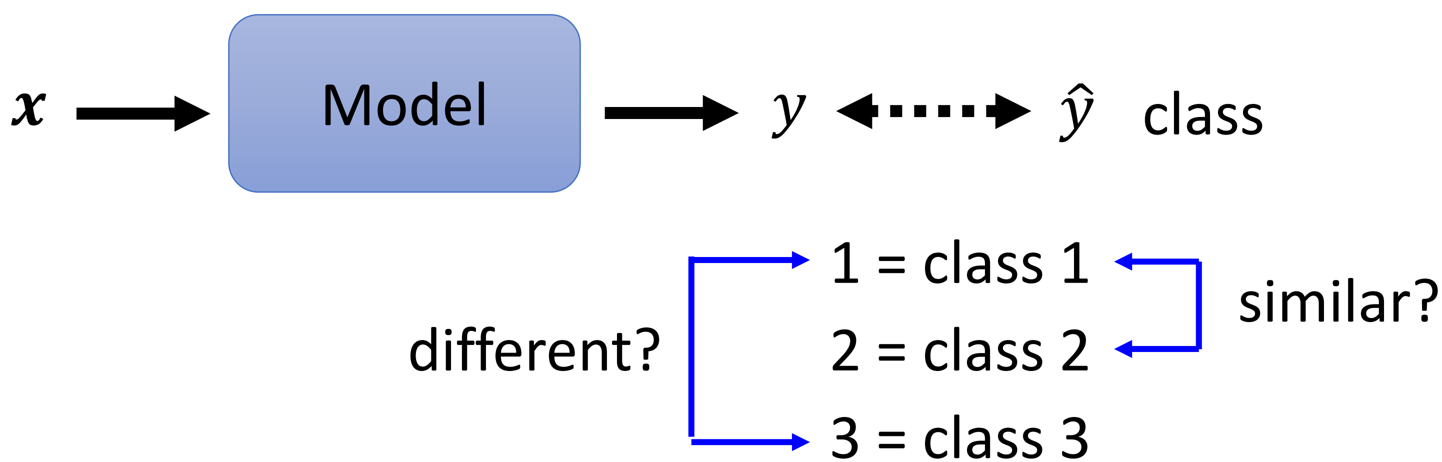
- Face recognition
  - Input: image of a face, output: person

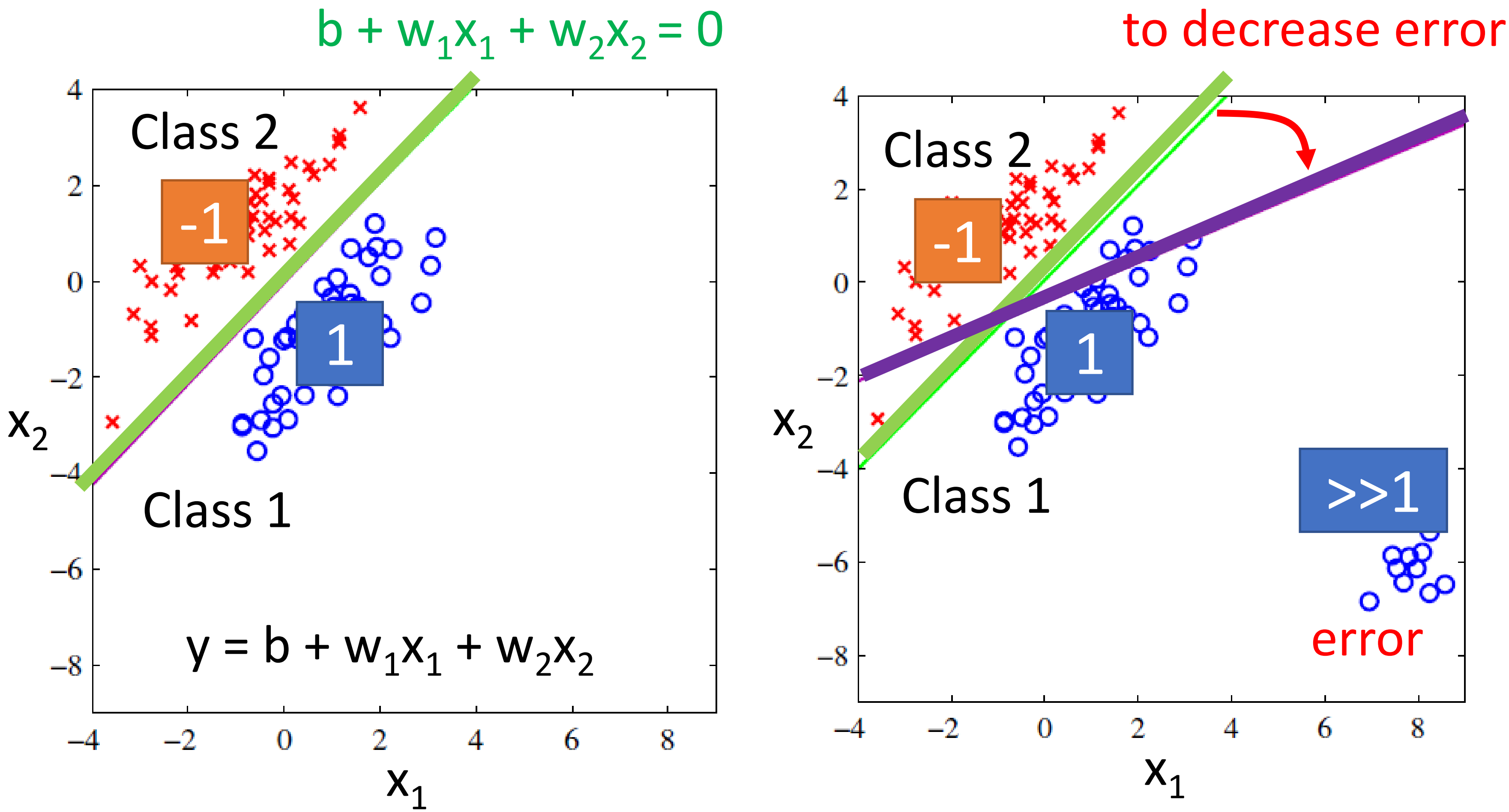
Classification as Regression?

- Regression



- Classification as regression?





Penalize to the examples that are “too correct” …

(Bishop, P186)

- Multiple class: Class 1 means the target is 1; Class 2 means the target is 2; Class 3 means the target is 3 …… problematic

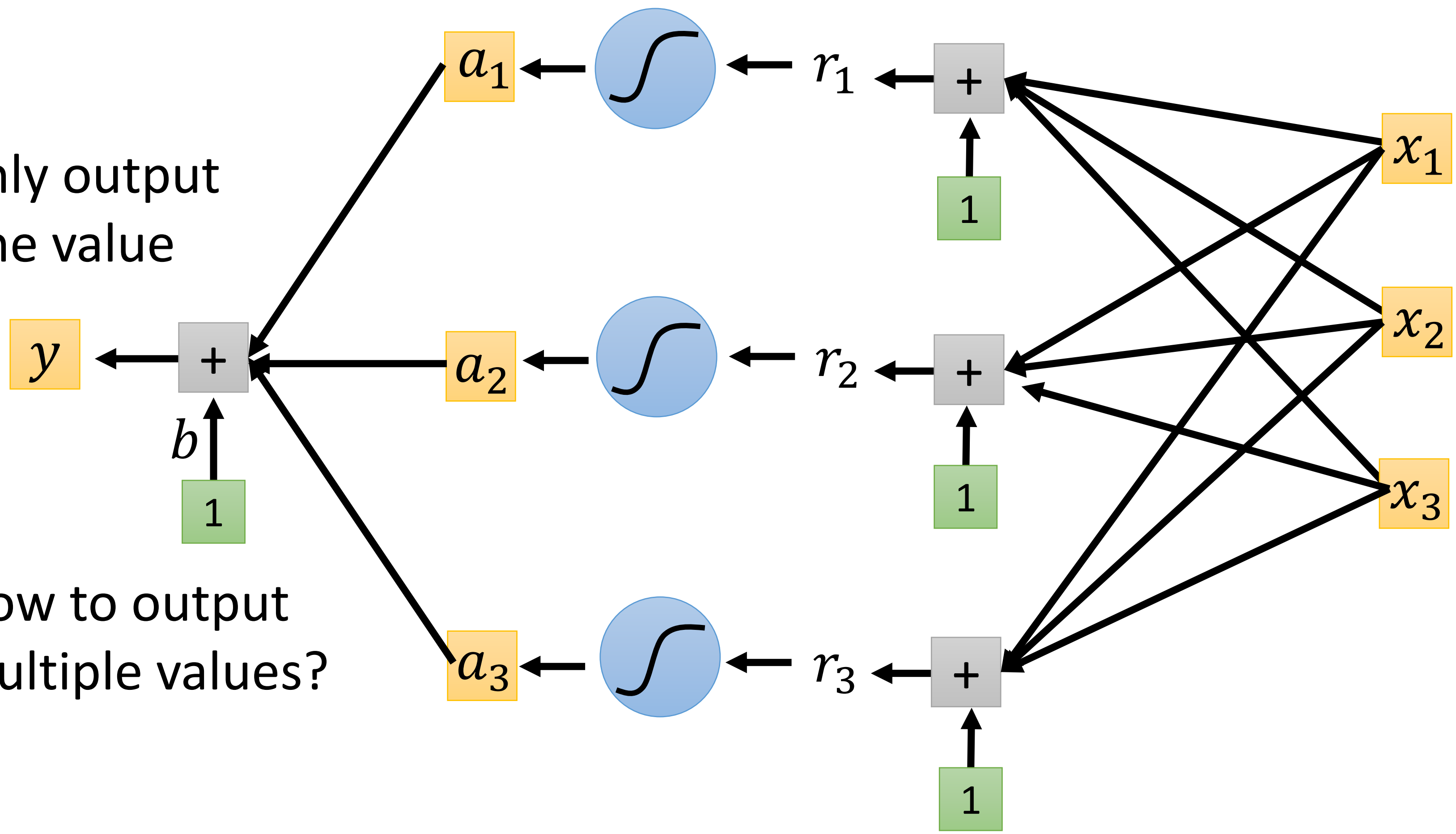
Class as one-hot vector

Class 1 Class 2 Class 3

$$\hat{\boldsymbol{y}} = \begin{bmatrix}1\\0\\0\end{bmatrix} \text{ or } \begin{bmatrix}0\\1\\0\end{bmatrix} \text{ or } \begin{bmatrix}0\\0\\1\end{bmatrix}$$

only output one value

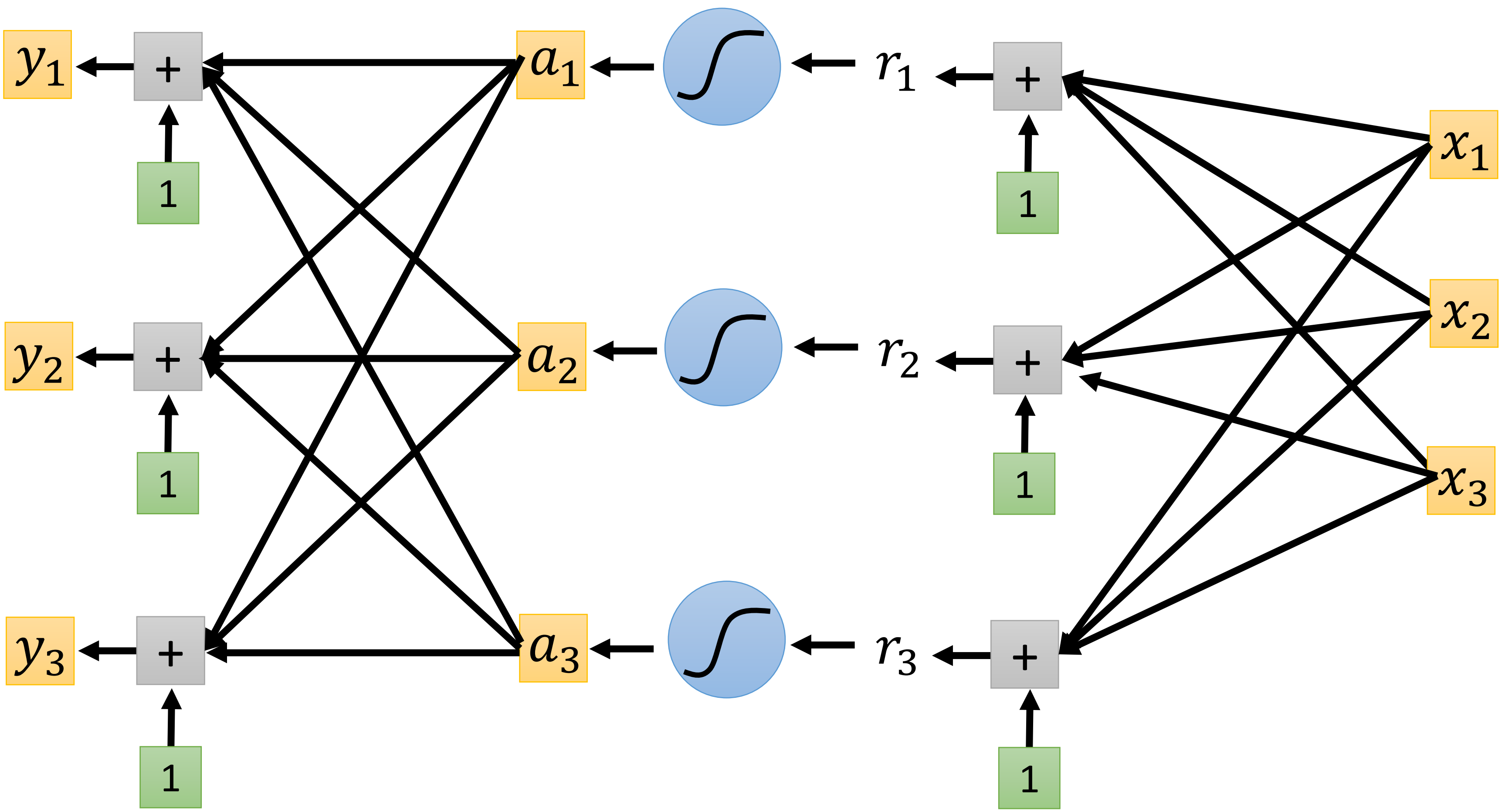
How to output multiple values?



Class as one-hot vector

Class 1 Class 2 Class 3

$$\hat{\boldsymbol{y}} = \begin{bmatrix}1\\0\\0\end{bmatrix} \text{ or } \begin{bmatrix}0\\1\\0\end{bmatrix} \text{ or } \begin{bmatrix}0\\0\\1\end{bmatrix}$$



Regression

feature

label

$$\hat{y} \longleftrightarrow y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

Classification

feature

$$\boldsymbol{y} = \boldsymbol{b}' + W' \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

label $\hat{\boldsymbol{y}} \longleftrightarrow \boldsymbol{y}' = \boxed{softmax}(\boldsymbol{y})$

0 or 1

Make all values between 0 and 1

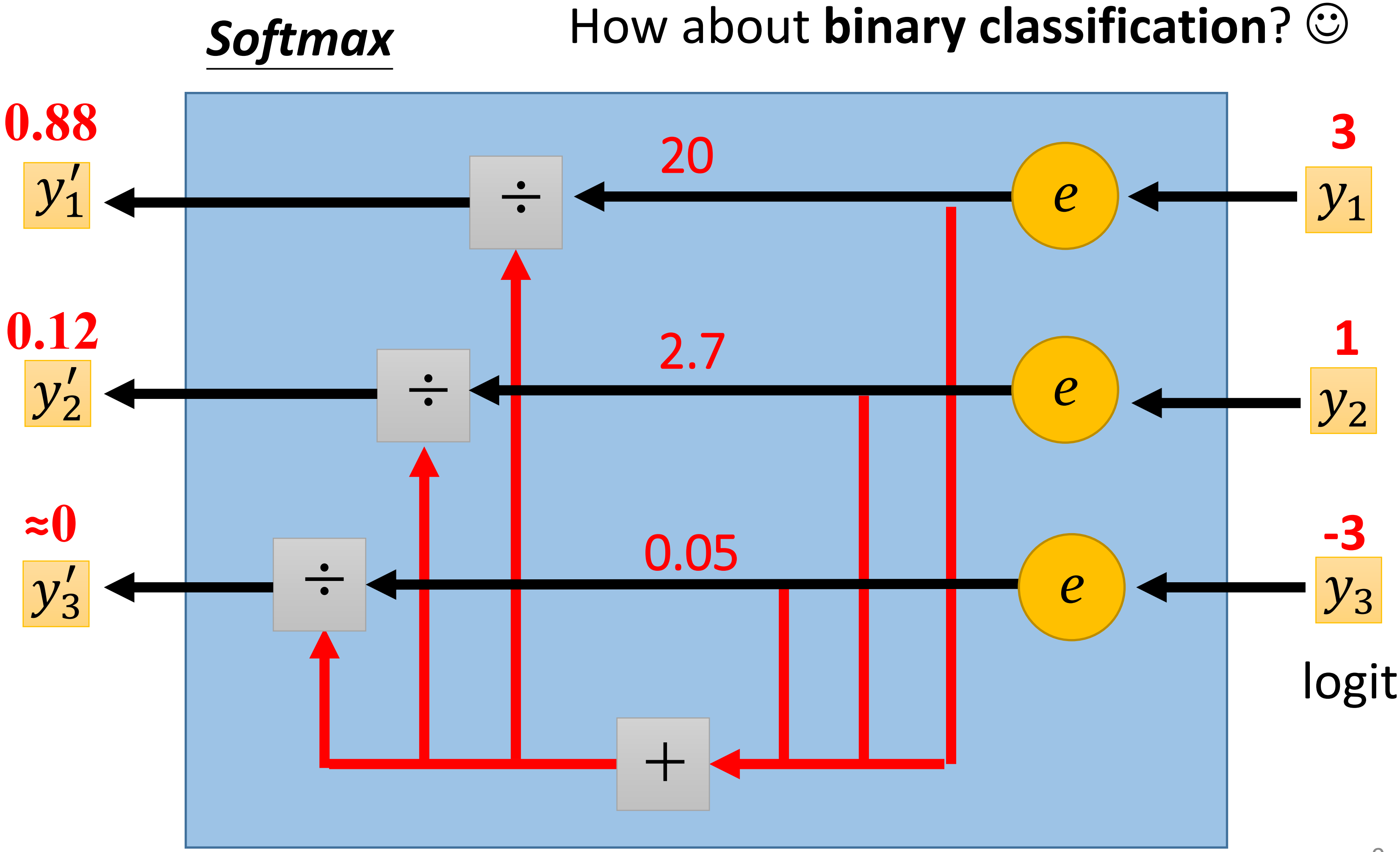
Can have any value

Soft-max

$$y_i' = \frac{exp(y_i)}{\sum_j exp(y_i)}$$

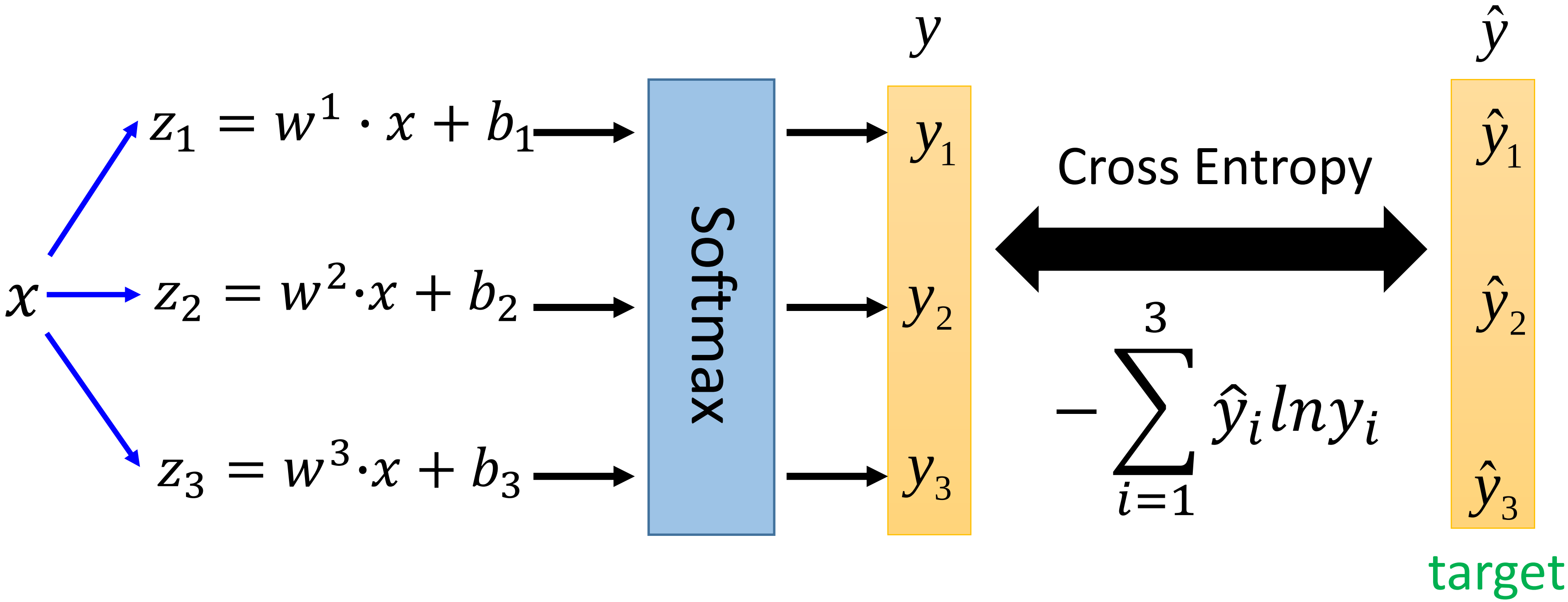
- $1 > y_i' > 0$
- $\sum_i y_i' = 1$

How about **binary classification**? ☺



Multi-class Classification (3 classes as example)

[Bishop, P209-210]



If x ∈ class 1

$$\hat{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$-ln y_1$$

If x ∈ class 2

$$\hat{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$-ln y_2$$

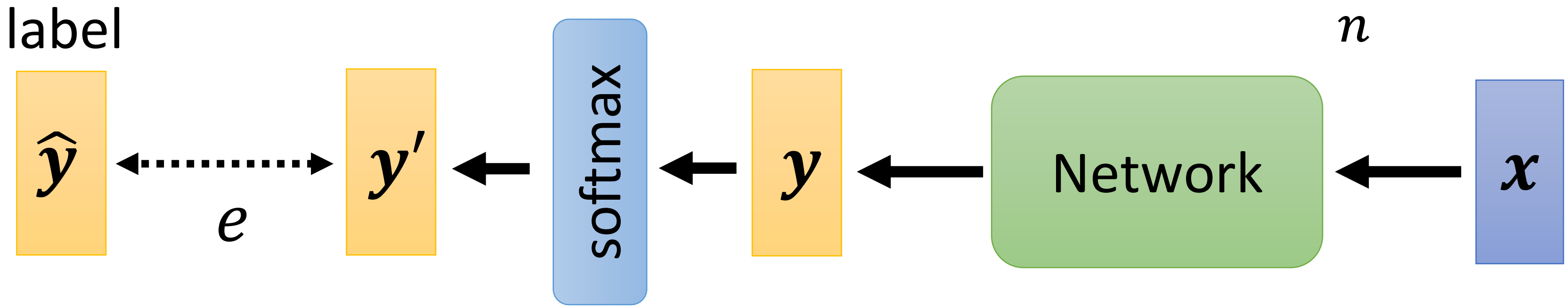
If x ∈ class 3

$$\hat{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$-ln y_3$$

Loss of Classification

$$L = \frac{1}{N}\sum_n e_n$$

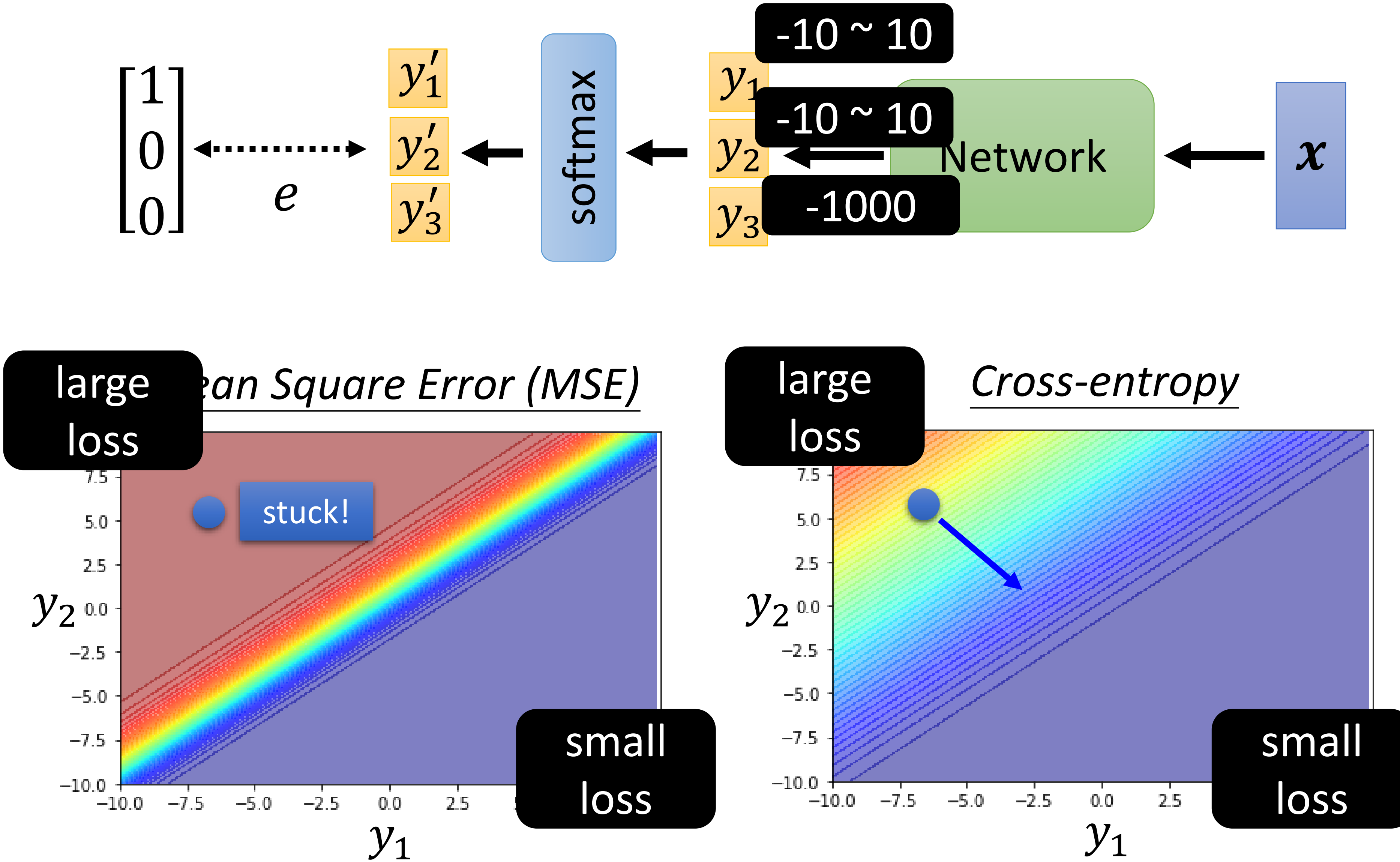


<u>*Mean Square Error (MSE)*</u> $$e = \sum_i (\hat{\boldsymbol{y}}_i - \boldsymbol{y}'_i)^2$$

<u>*Cross-entropy*</u>

$$e = -\sum_i \hat{\boldsymbol{y}}_i ln \boldsymbol{y}'_i$$

Minimizing cross-entropy is equivalent to **maximizing likelihood**.



Changing the loss function can change the difficulty of optimization.

Convolutional Neural Network (CNN)

Image Classification

100 x 100

Model

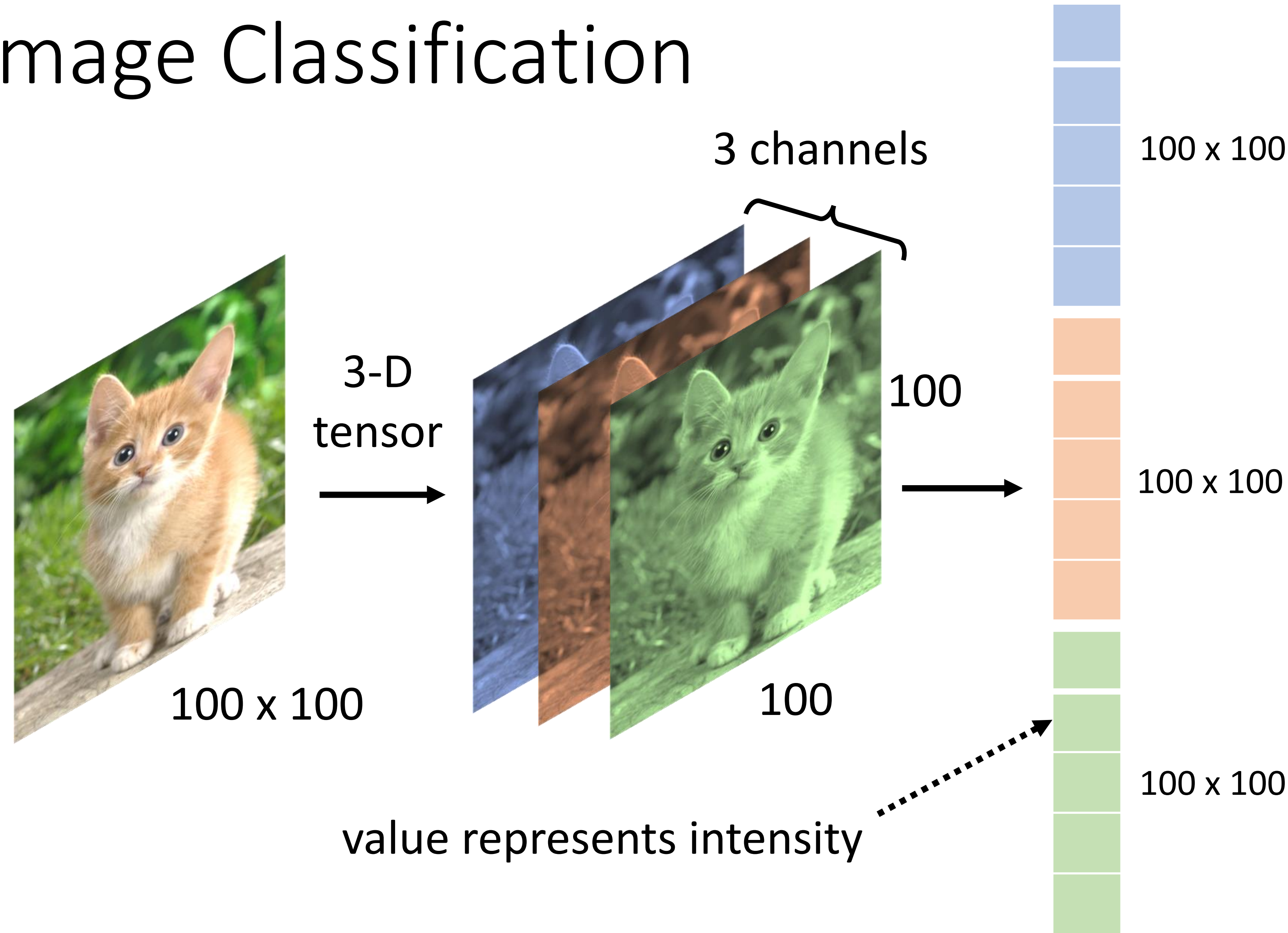
$$\begin{bmatrix} \vdots \\ 0.2 \\ 0.7 \\ 0.1 \\ \vdots \end{bmatrix} \qquad \begin{array}{r} \\ \text{dog} \\ \text{cat} \\ \text{tree} \\ \\ \end{array} \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

$$\boldsymbol{y}' \longleftrightarrow \hat{\boldsymbol{y}}$$

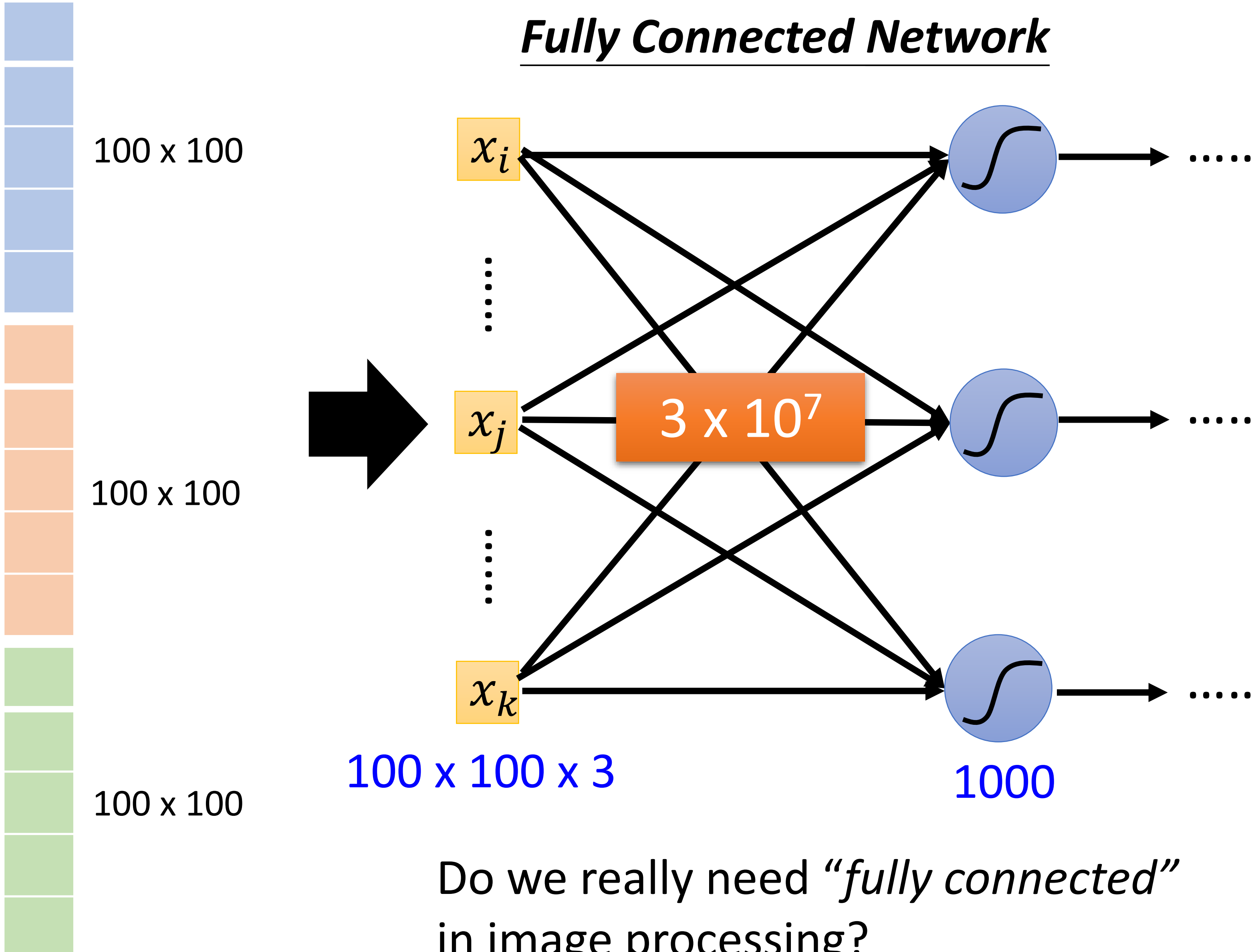
Cross entropy

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

Image Classification



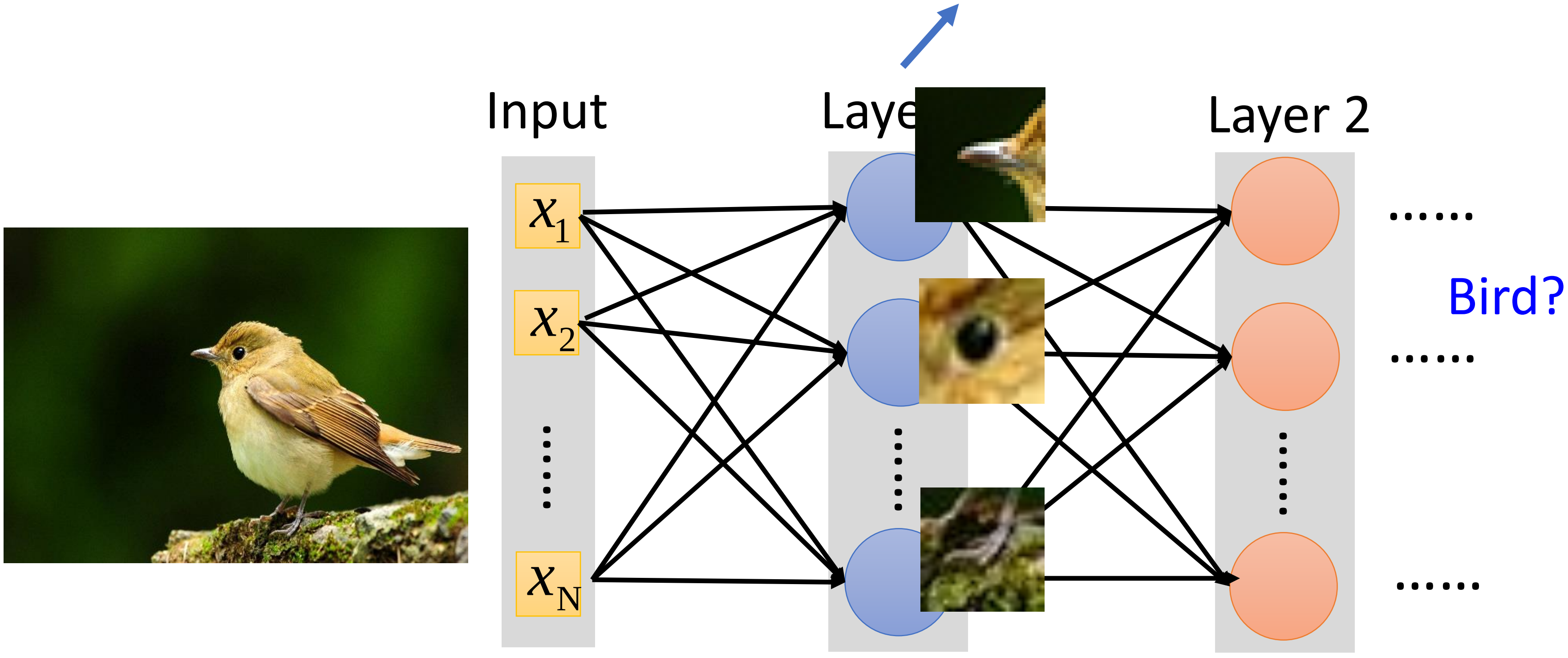
Fully Connected Network



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

Observation 1

Identifying some critical patterns

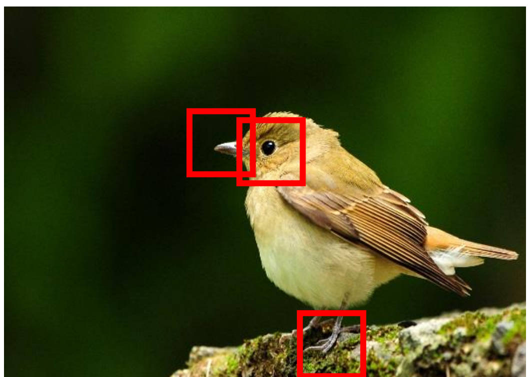


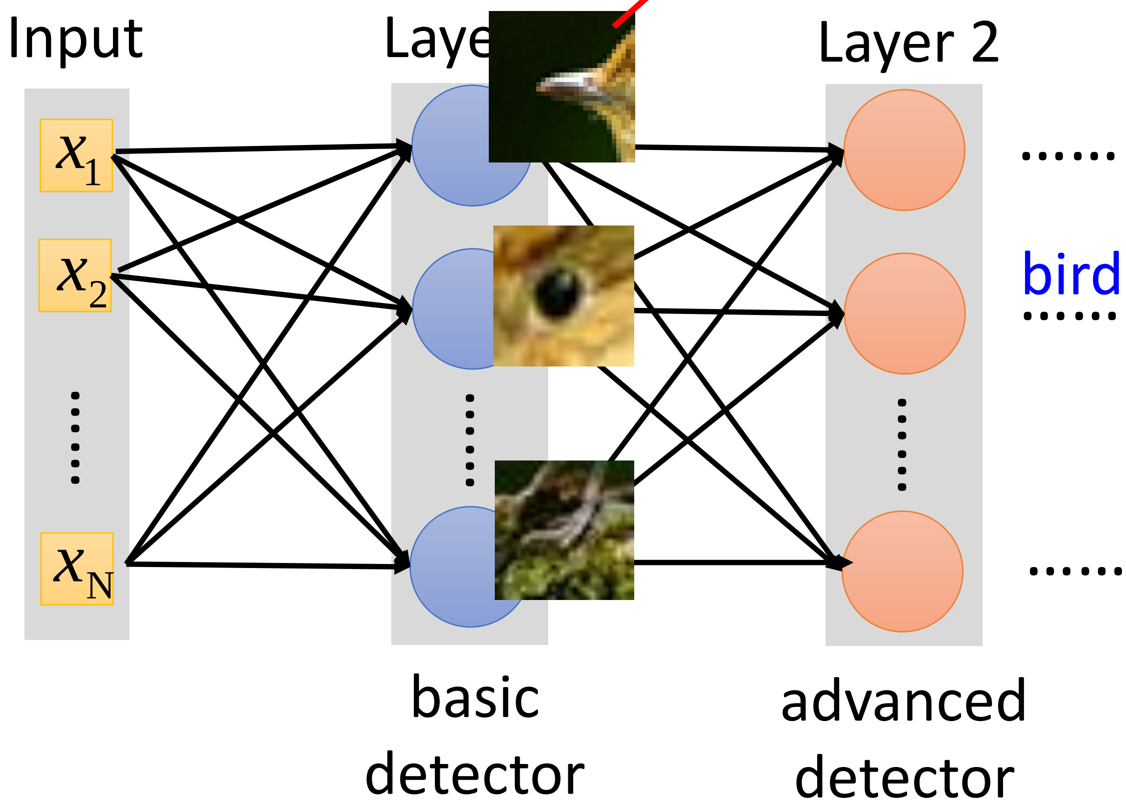
Perhaps human also identify birds in a similar way … ☺

Observation 1

A neuron does not have to see the whole image.

Need to see the whole image?

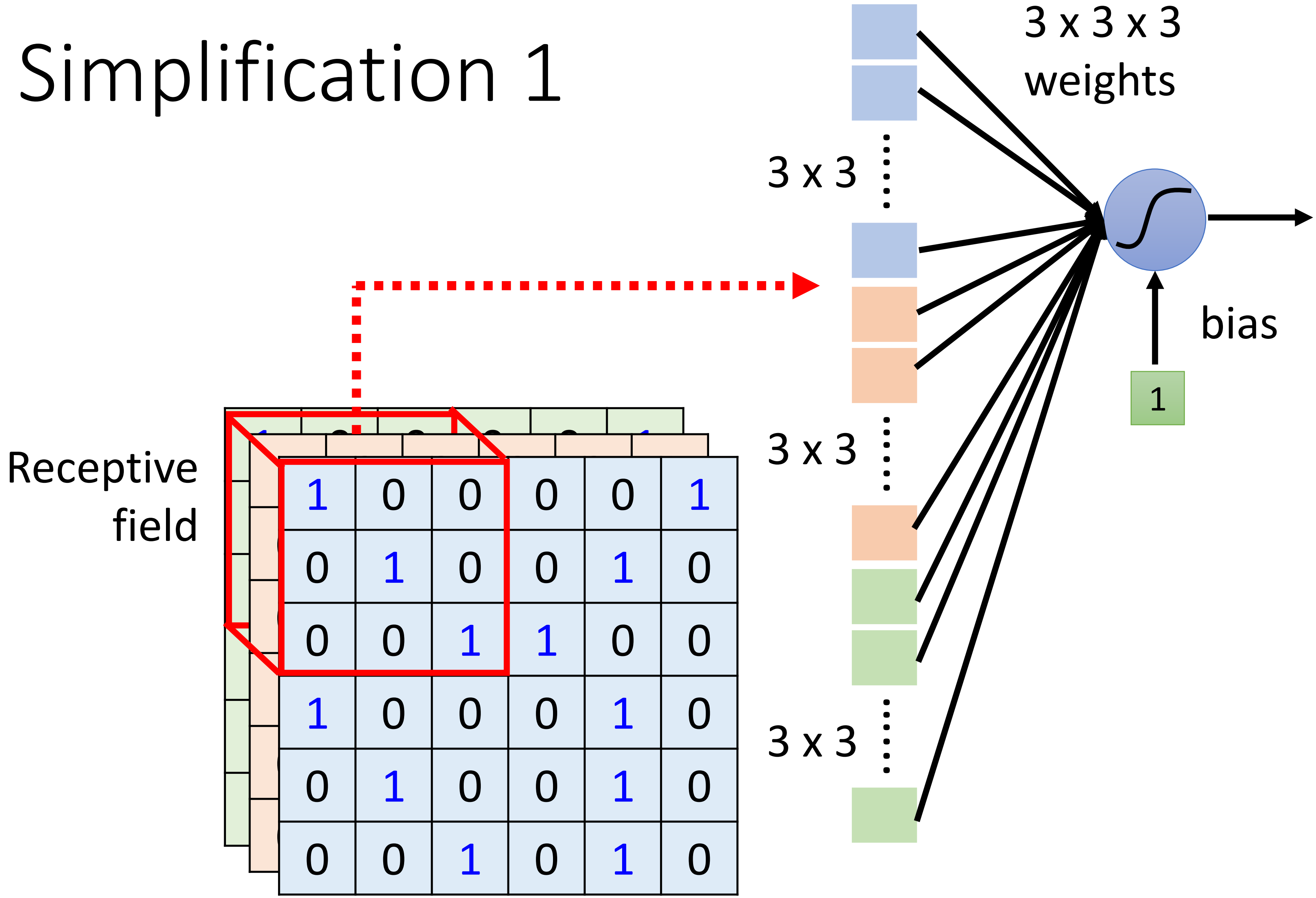




Some patterns are much smaller than the whole image.

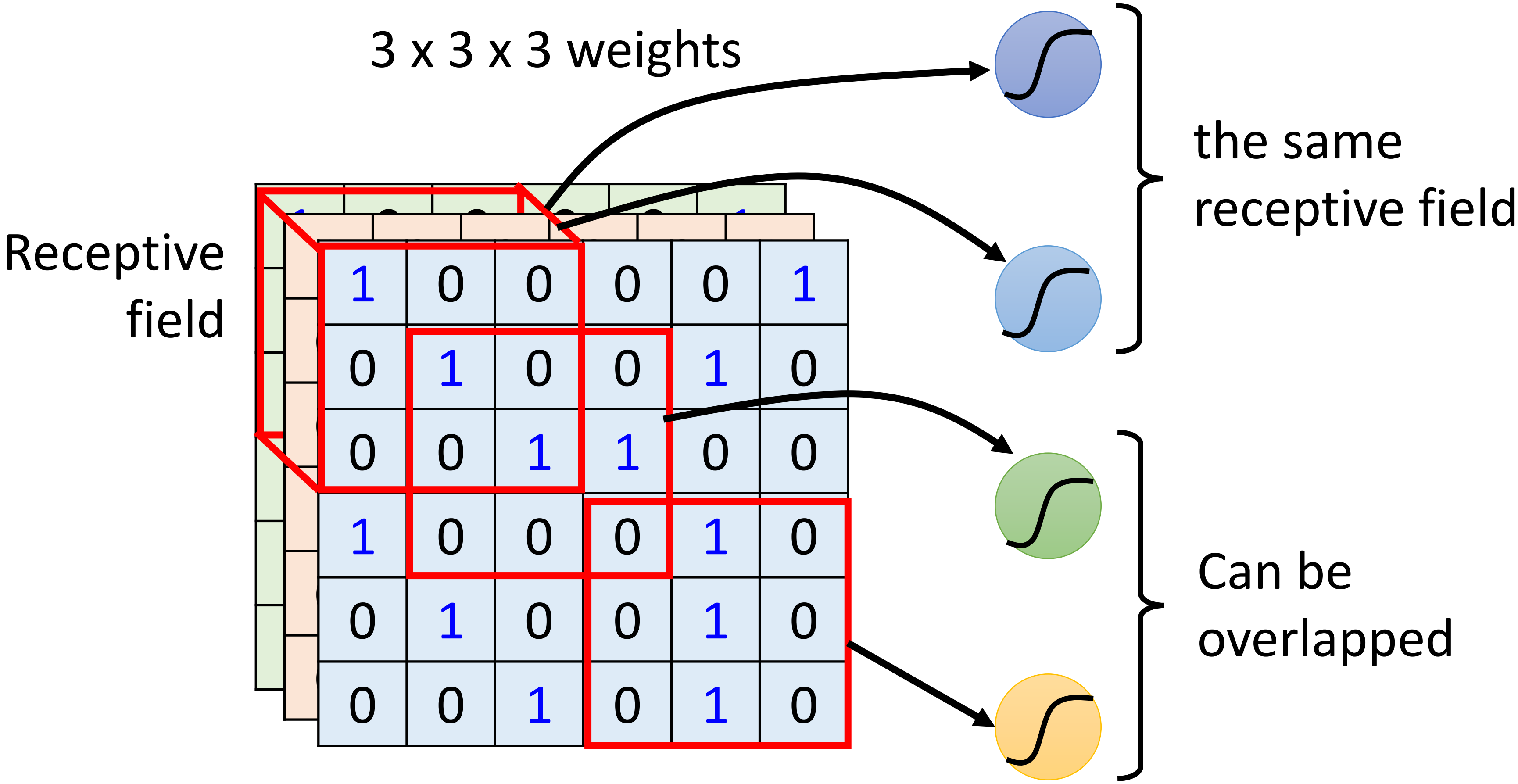
Connecting to small region with less parameters

Simplification 1



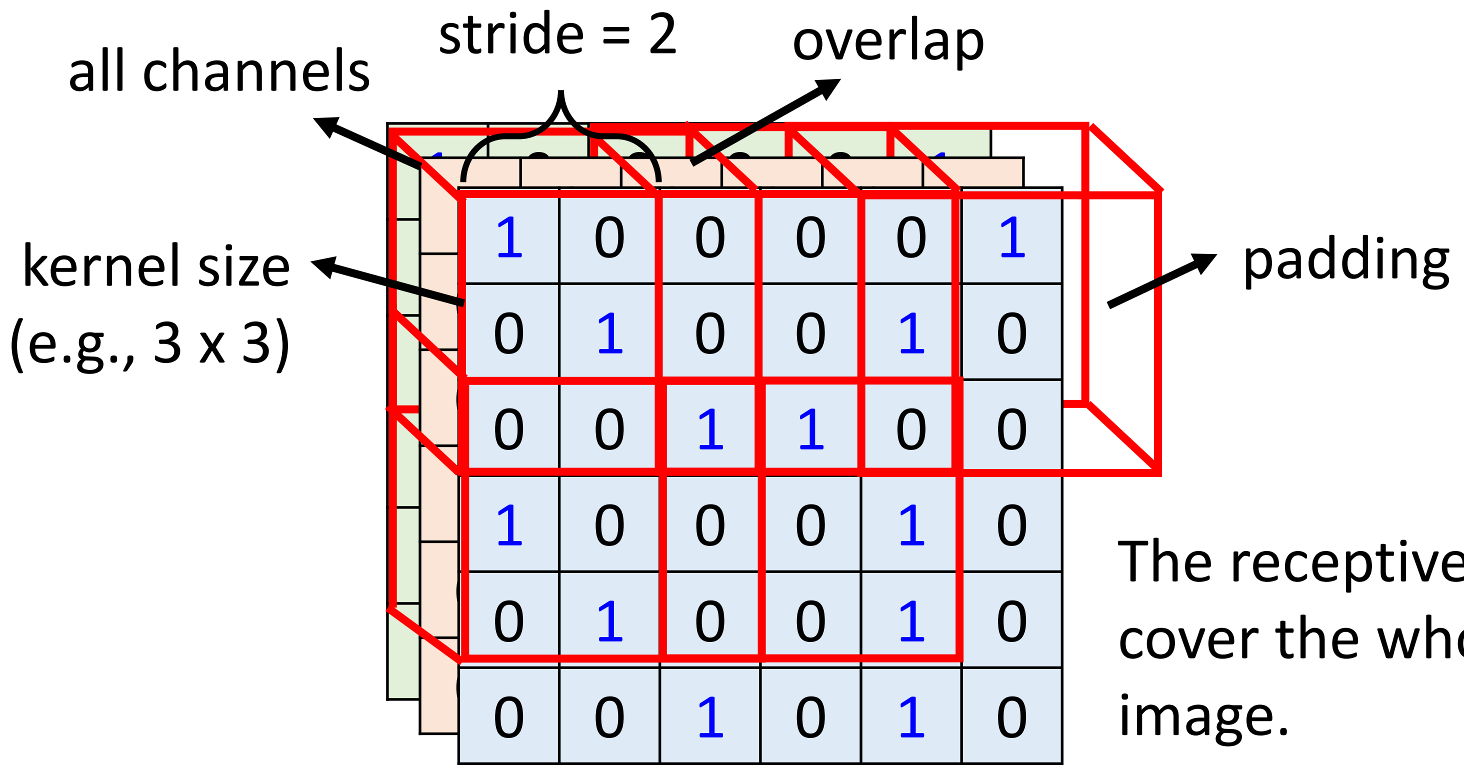
Simplification 1

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



Simplification 1 – Typical Setting

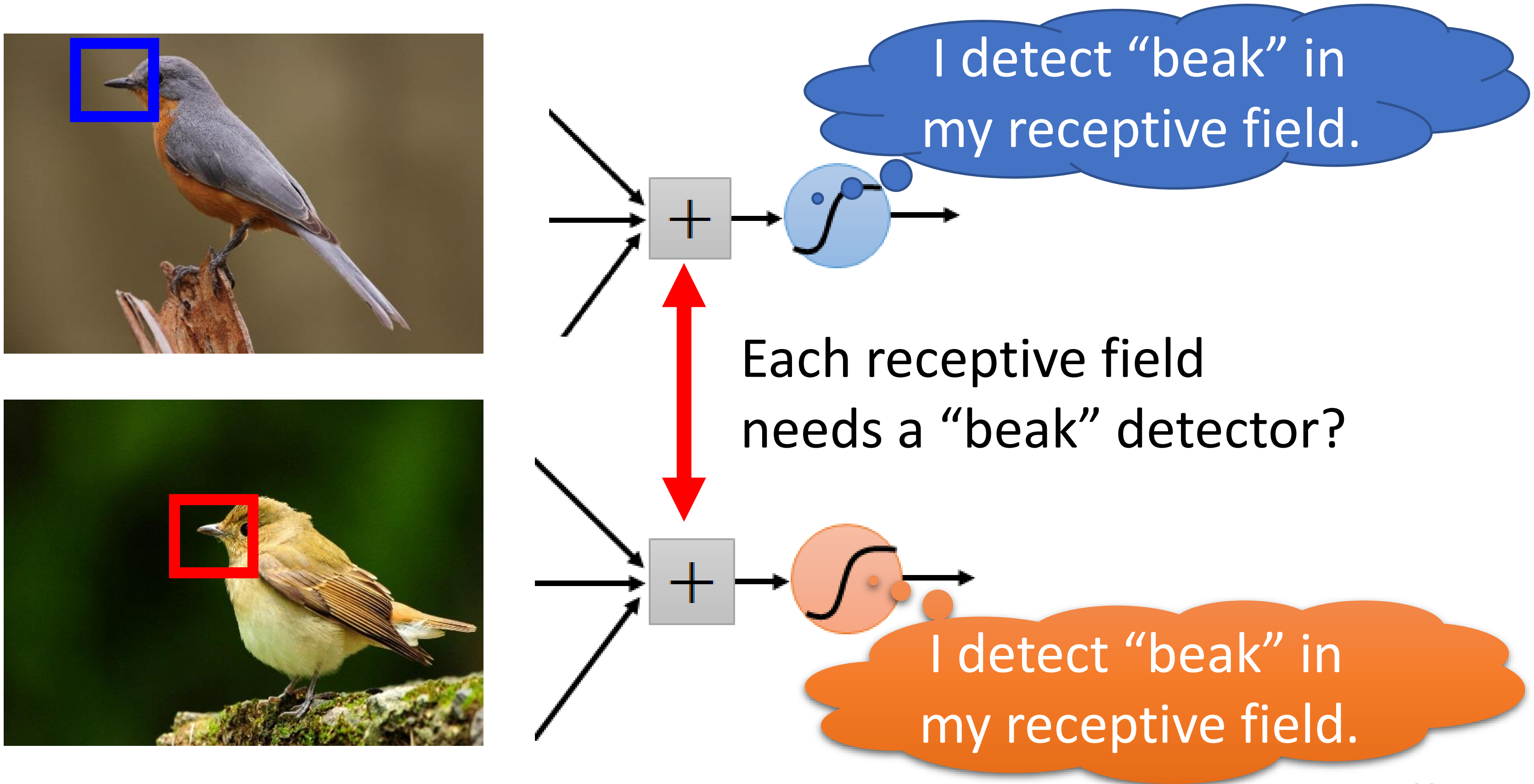
Each receptive field has a set of neurons (e.g., 64 neurons).



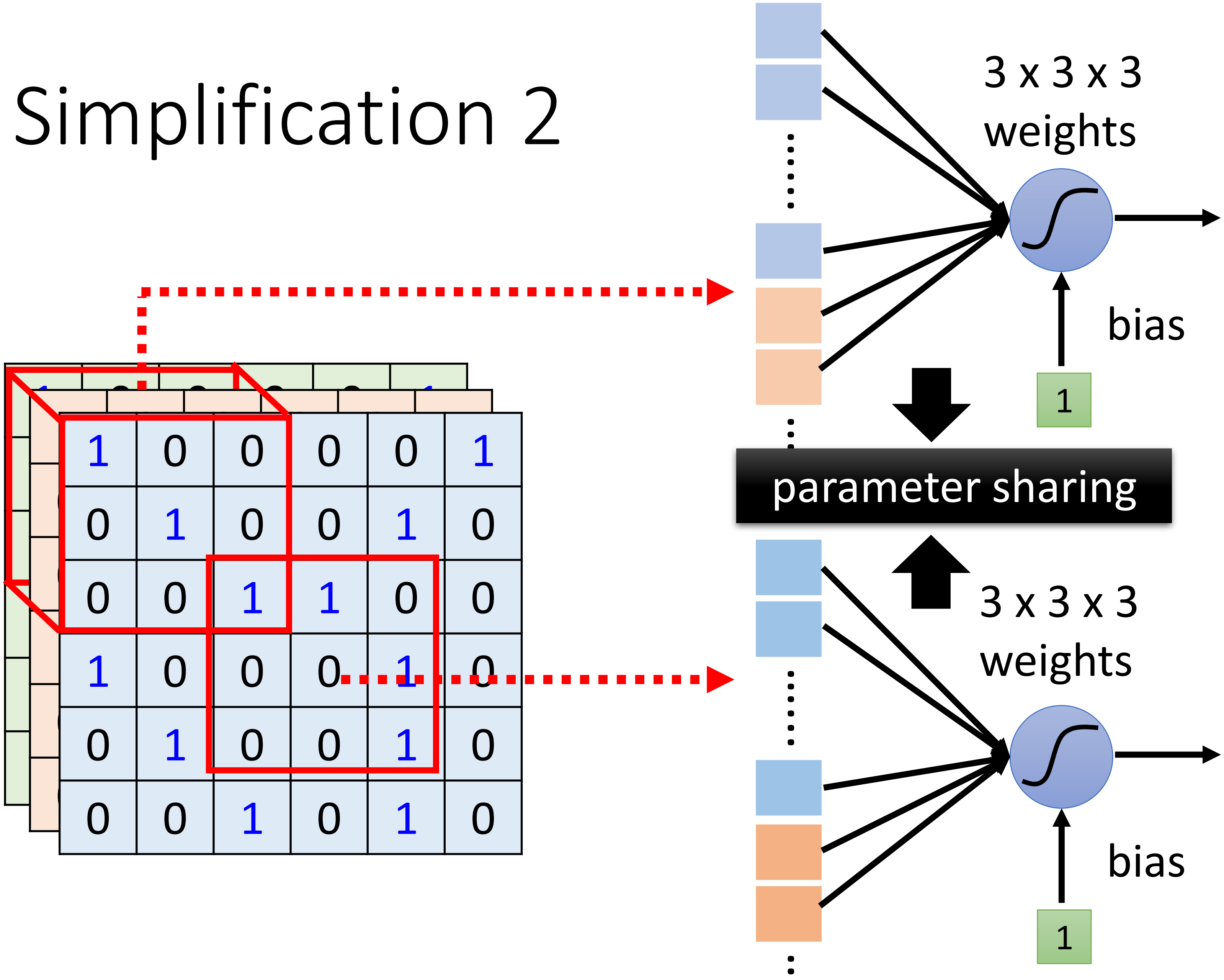
The receptive fields cover the whole image.

Observation 2

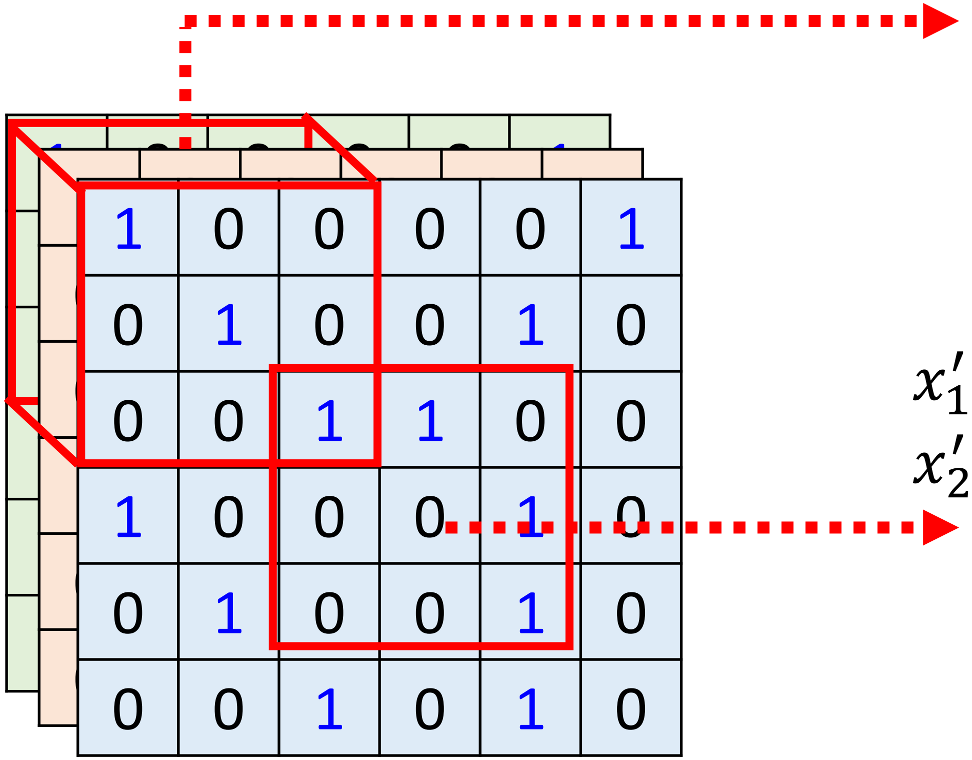
- The same patterns appear in different regions.

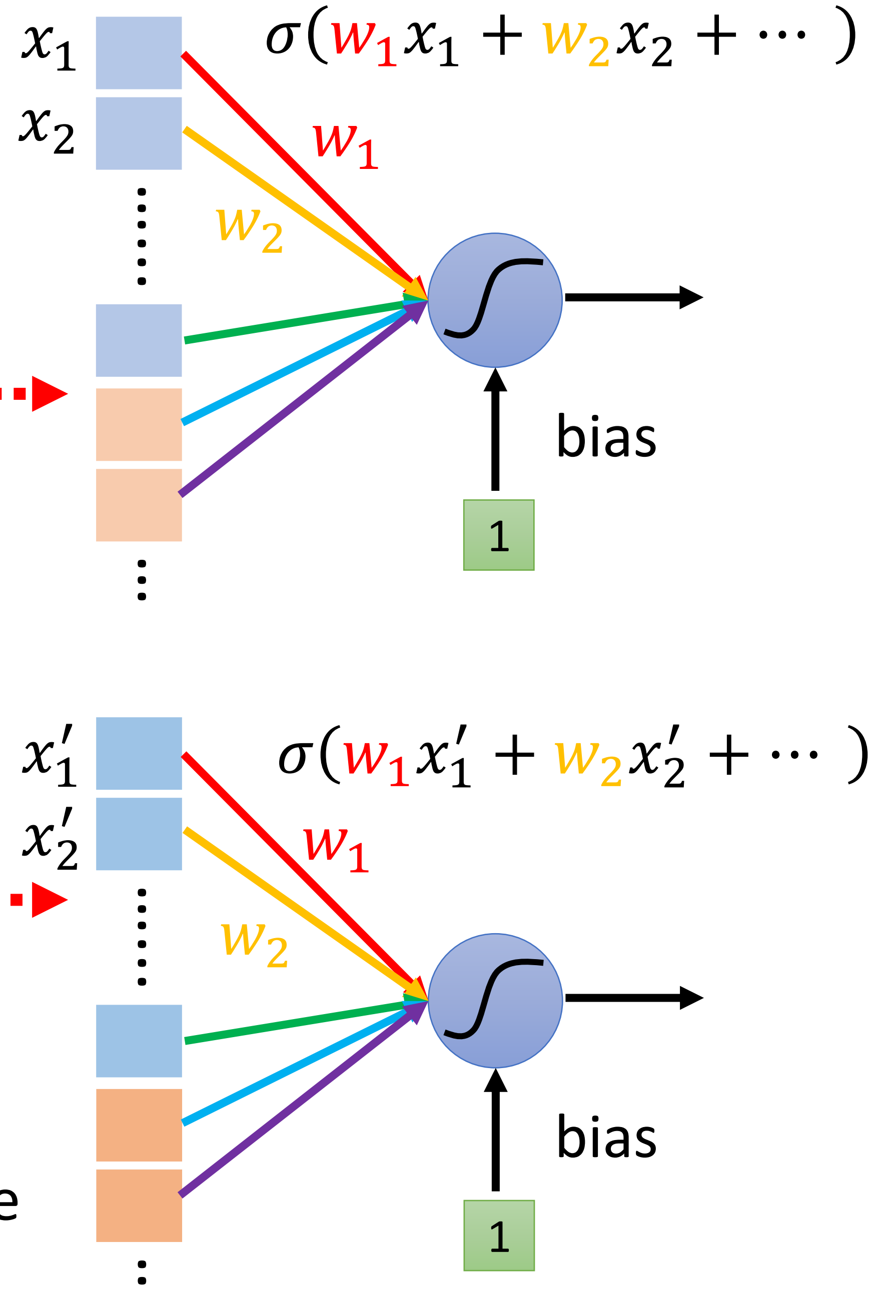


Simplification 2



Simplification 2

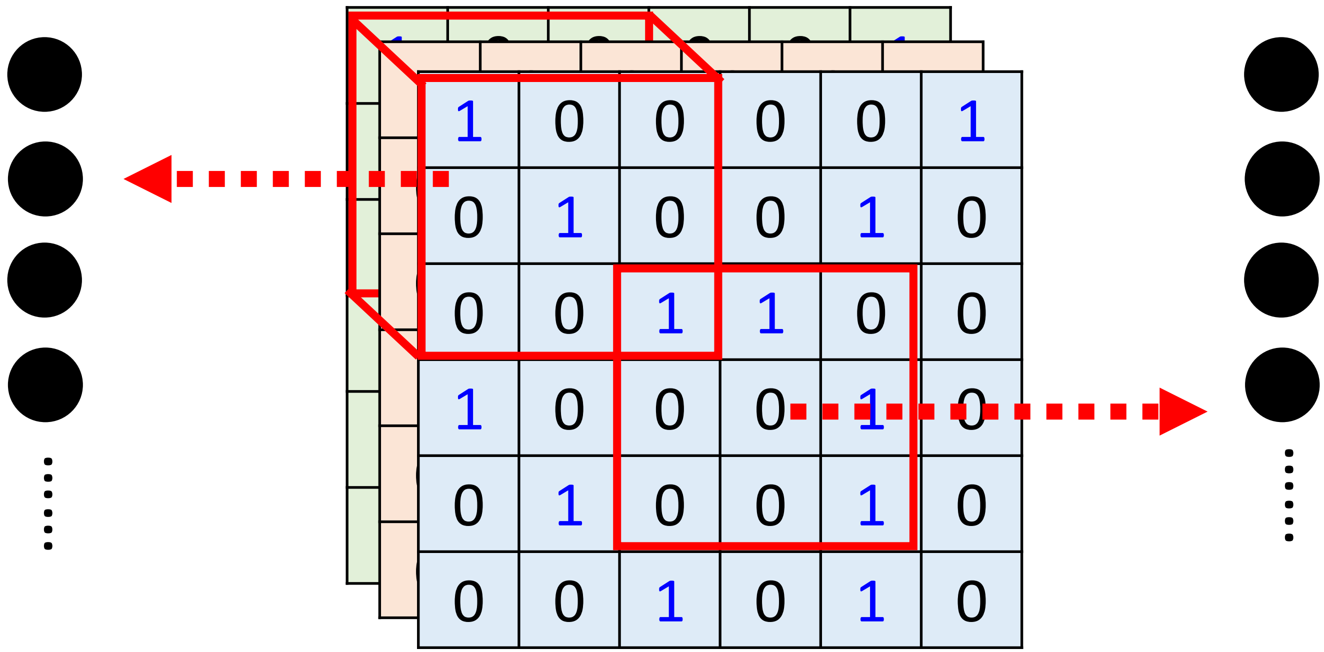




Two neurons with the same receptive field would not share parameters.

Simplification 2 – Typical Setting

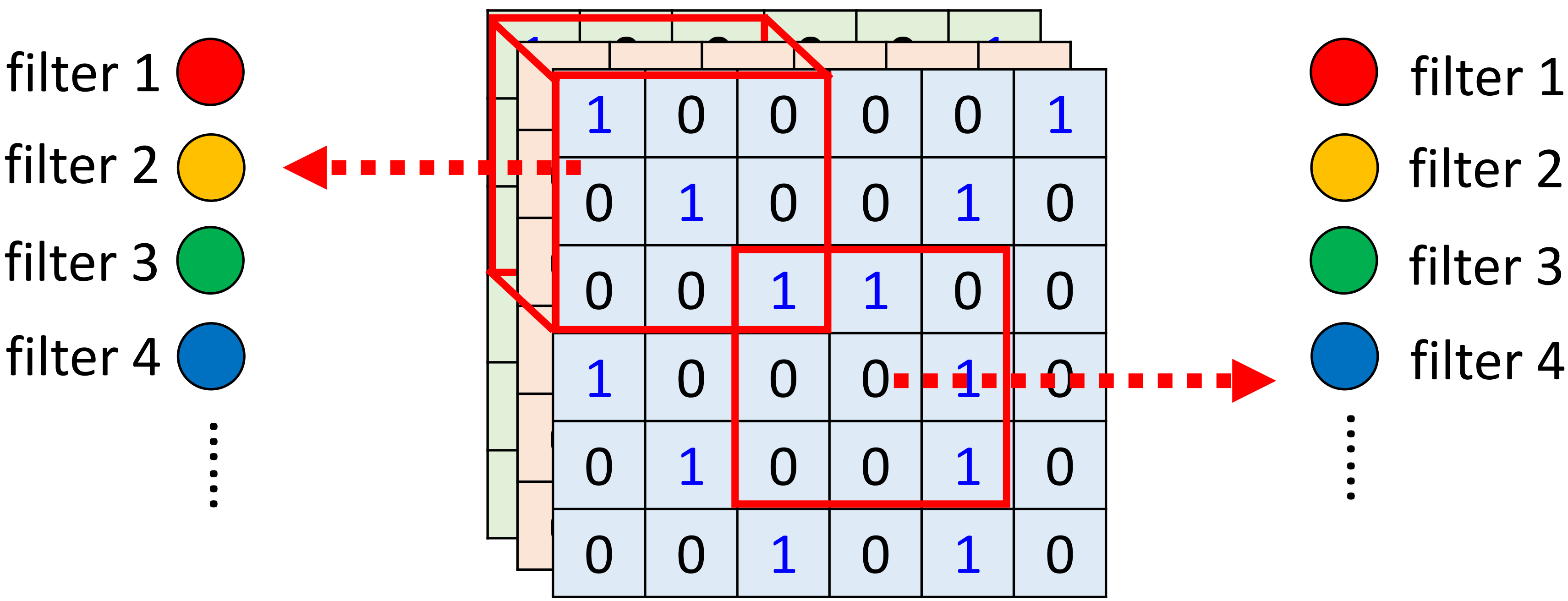
Each receptive field has a set of neurons (e.g., 64 neurons).



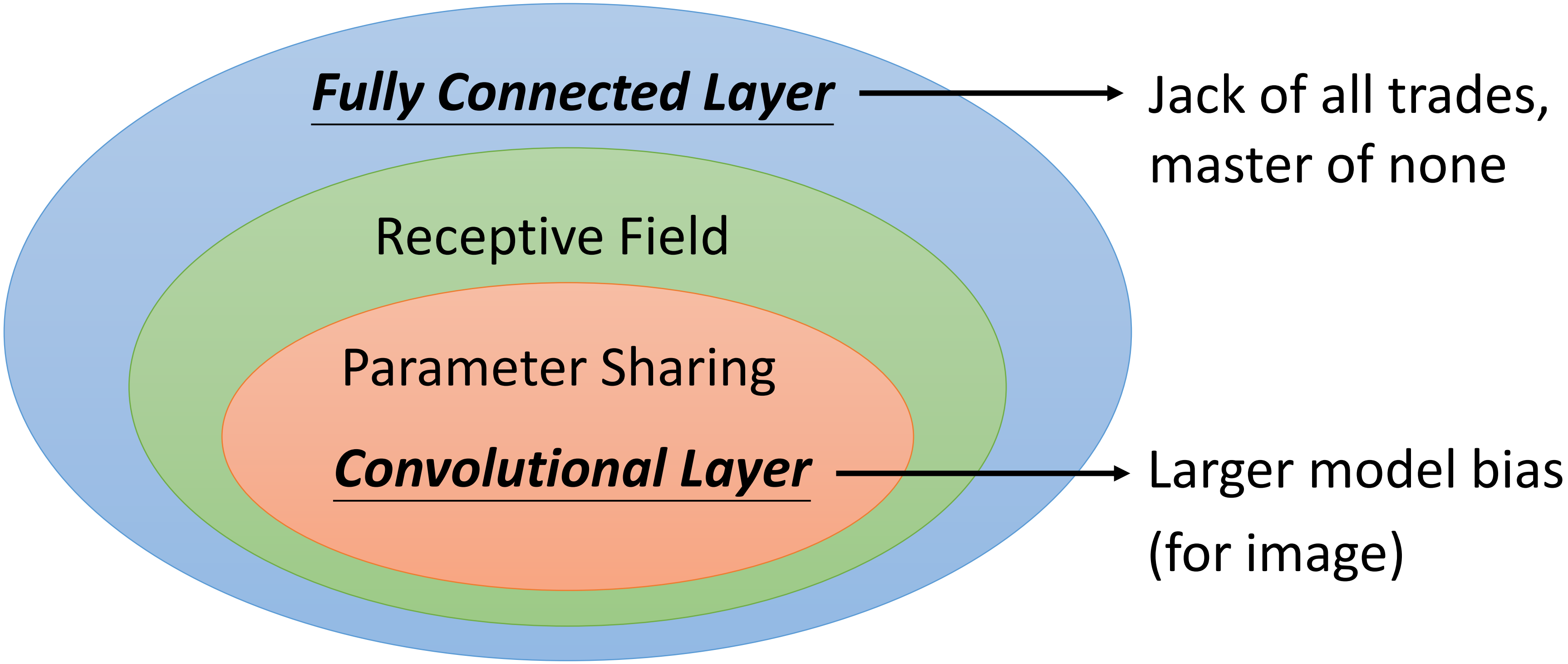
Simplification 2 – Typical Setting

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



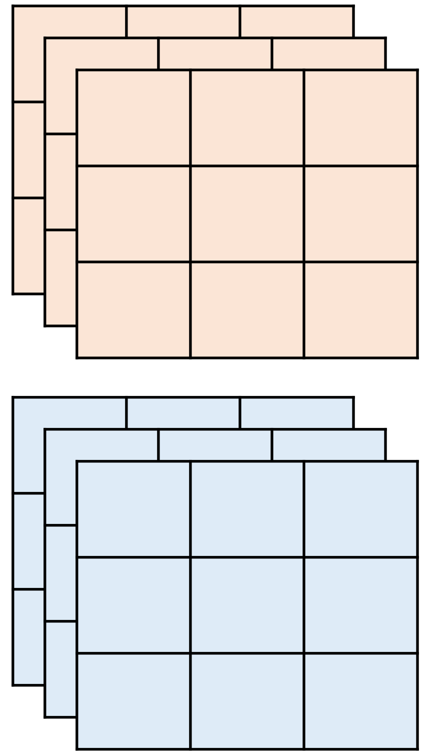
- Some patterns are much smaller than the whole image.
- The same patterns appear in different regions.

Another story based on ***filter*** ☺

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).

CNN – Convolution

Those are the network parameters to be learned.

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

Matrix

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

Filter 2

Matrix

⋮

Property 1 Each filter detects a small pattern (3 x 3).

CNN – Convolution

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

Filter 1

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

3 -1

CNN – Convolution

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

Filter 1

If stride=2

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

3 -3

We set stride=1 below

CNN – Convolution

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

Filter 1

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

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

Property 2

Convolutional Layer

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

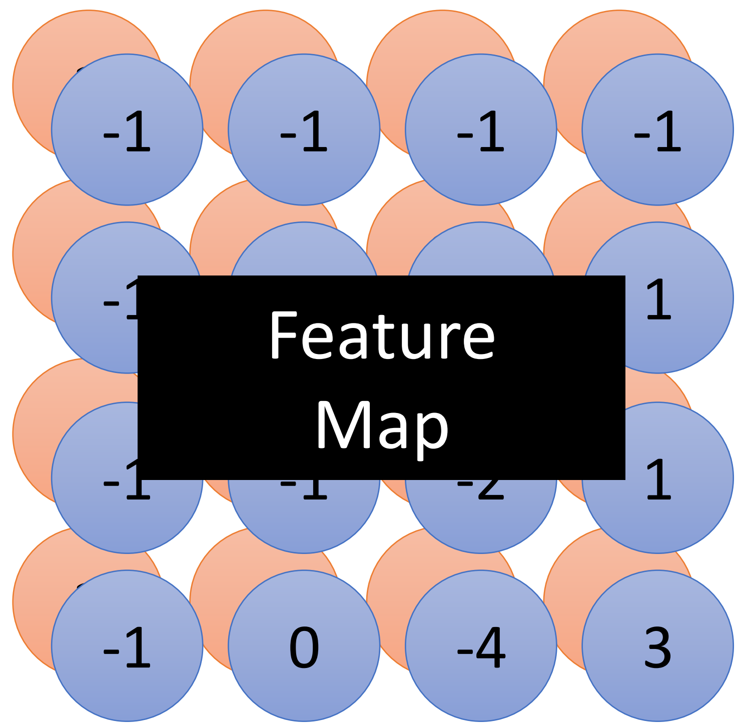
Filter 2

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



CNN – Colorful image

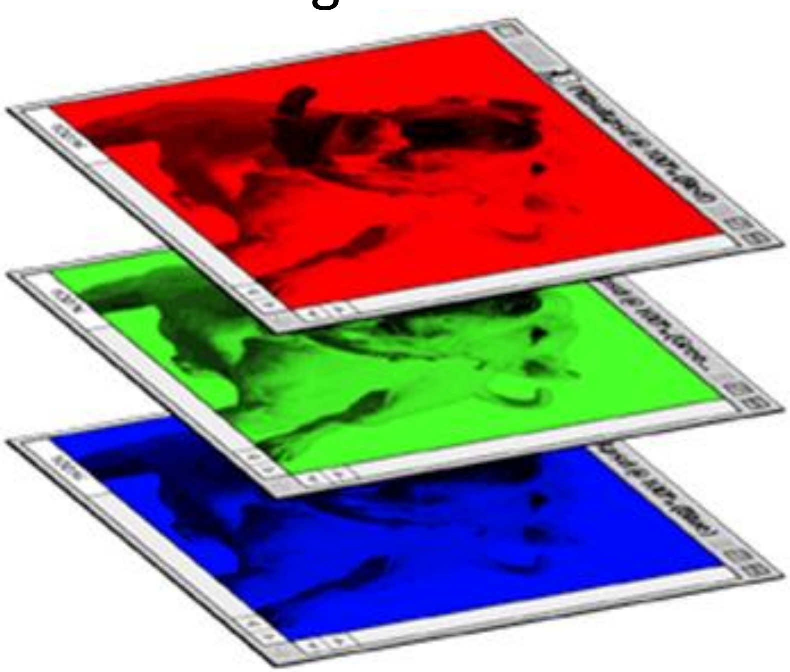
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

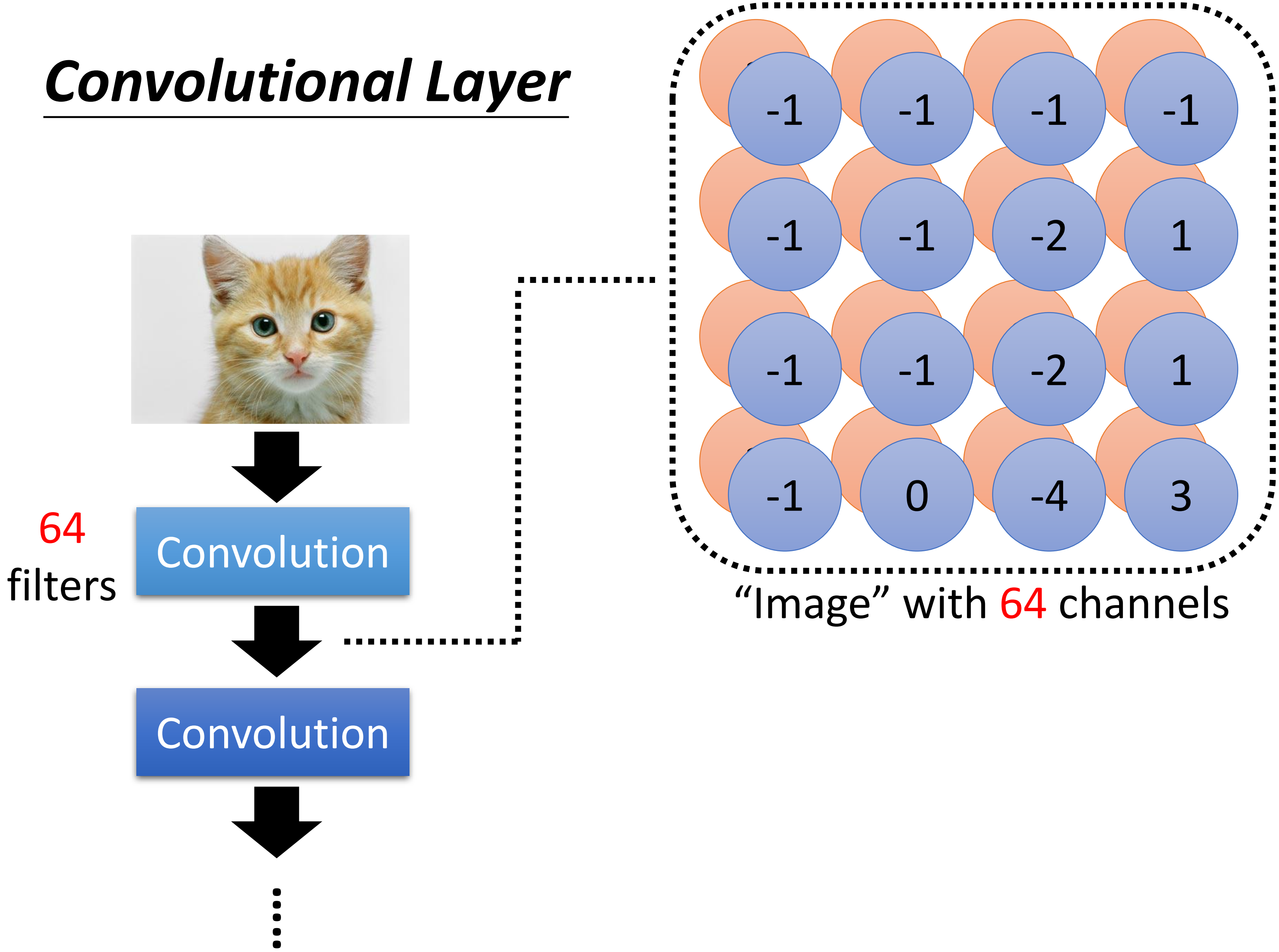
Filter 2

Colorful image

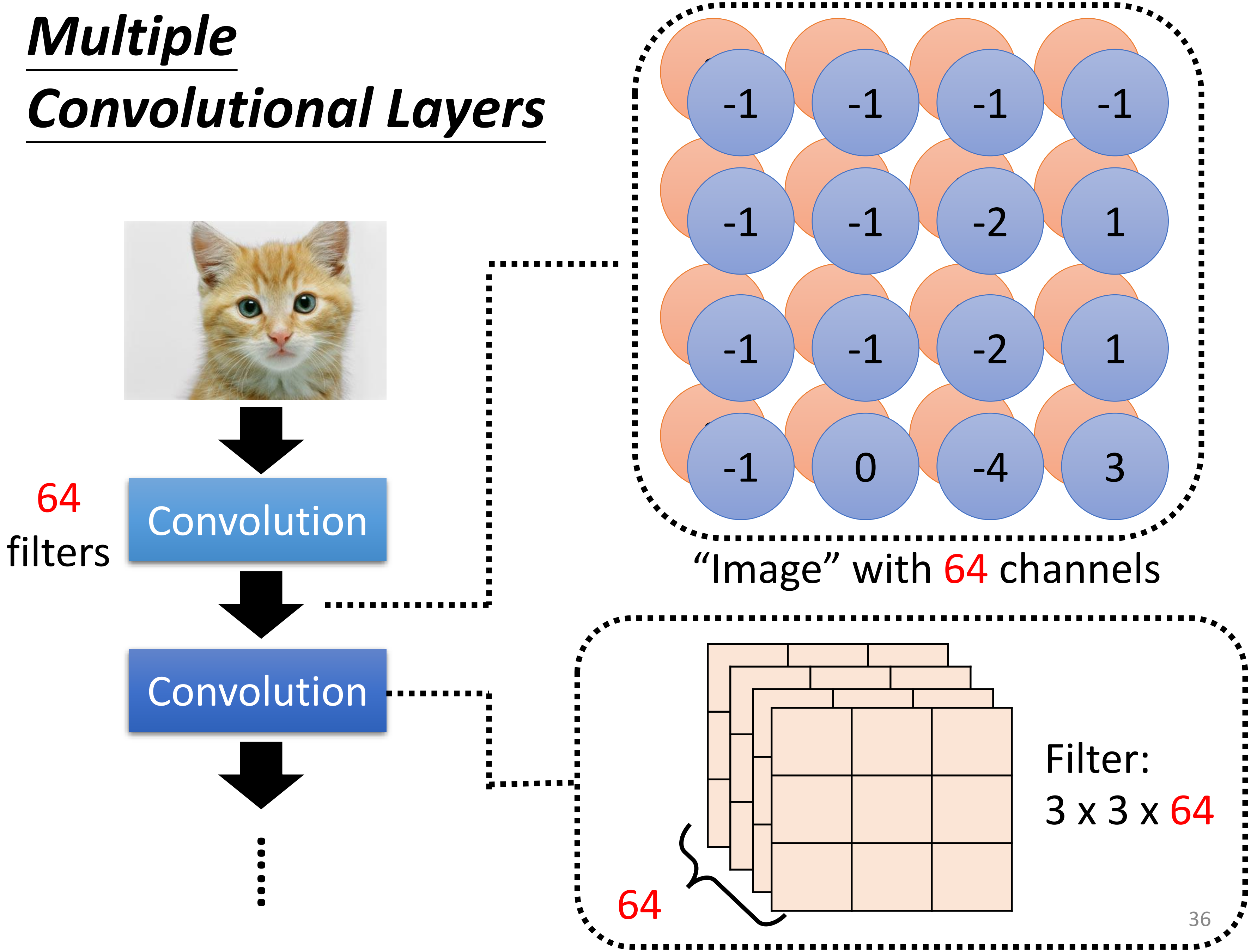


1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

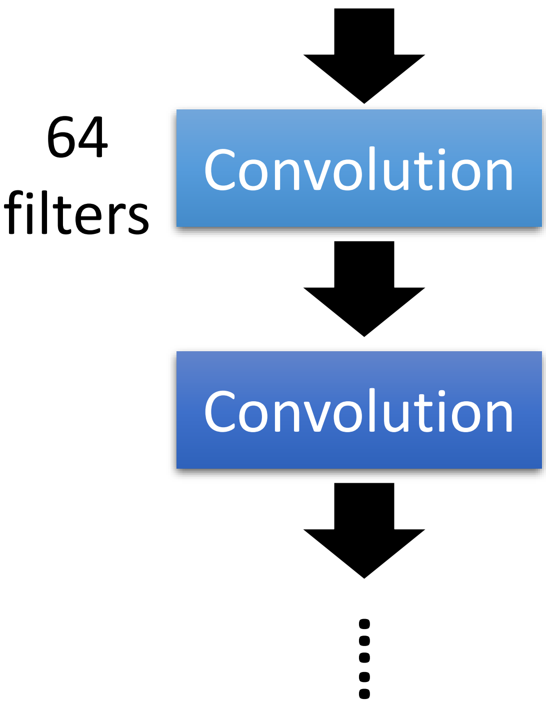
Convolutional Layer



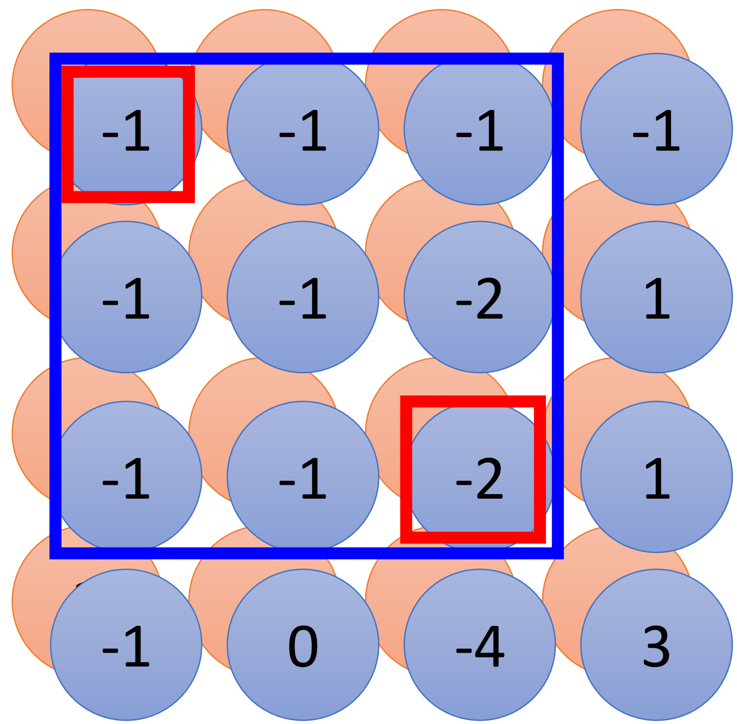
Multiple Convolutional Layers



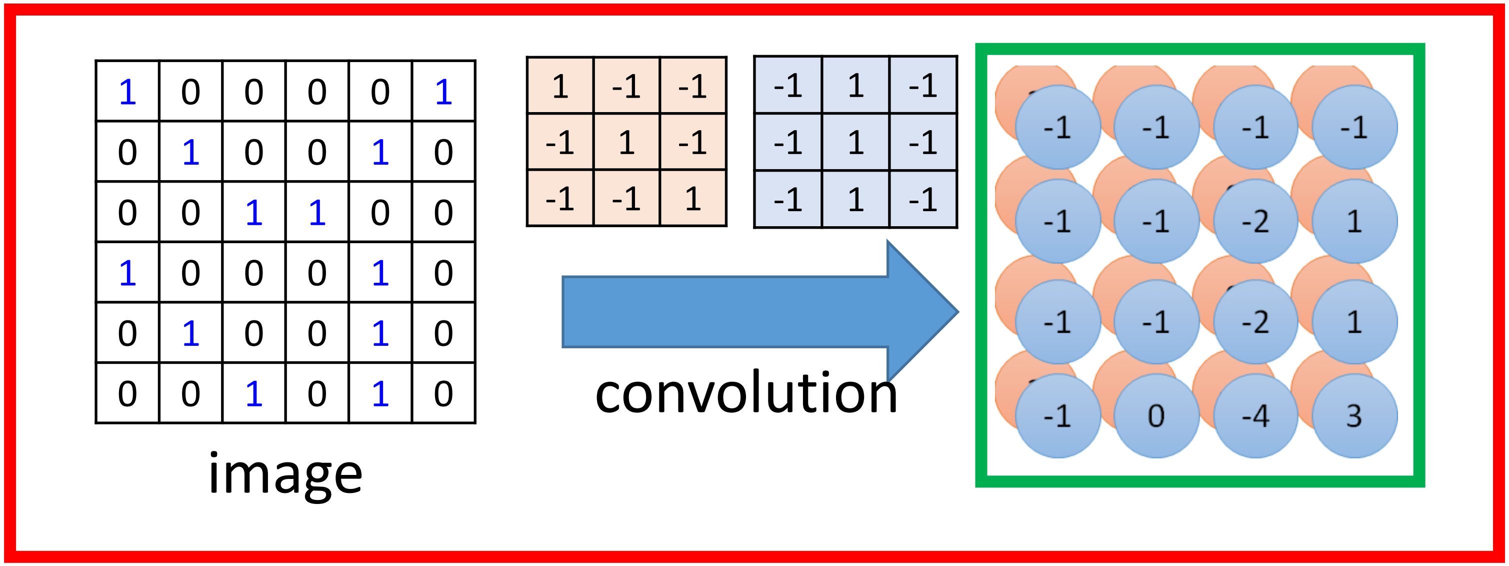
Multiple Convolutional Layers

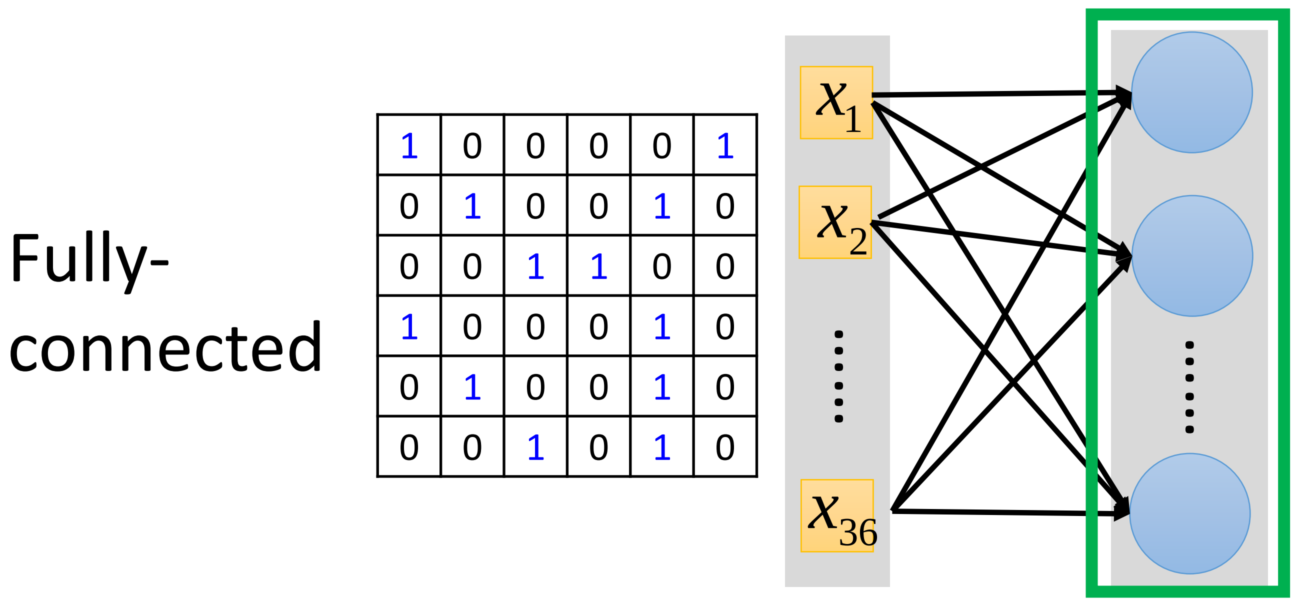


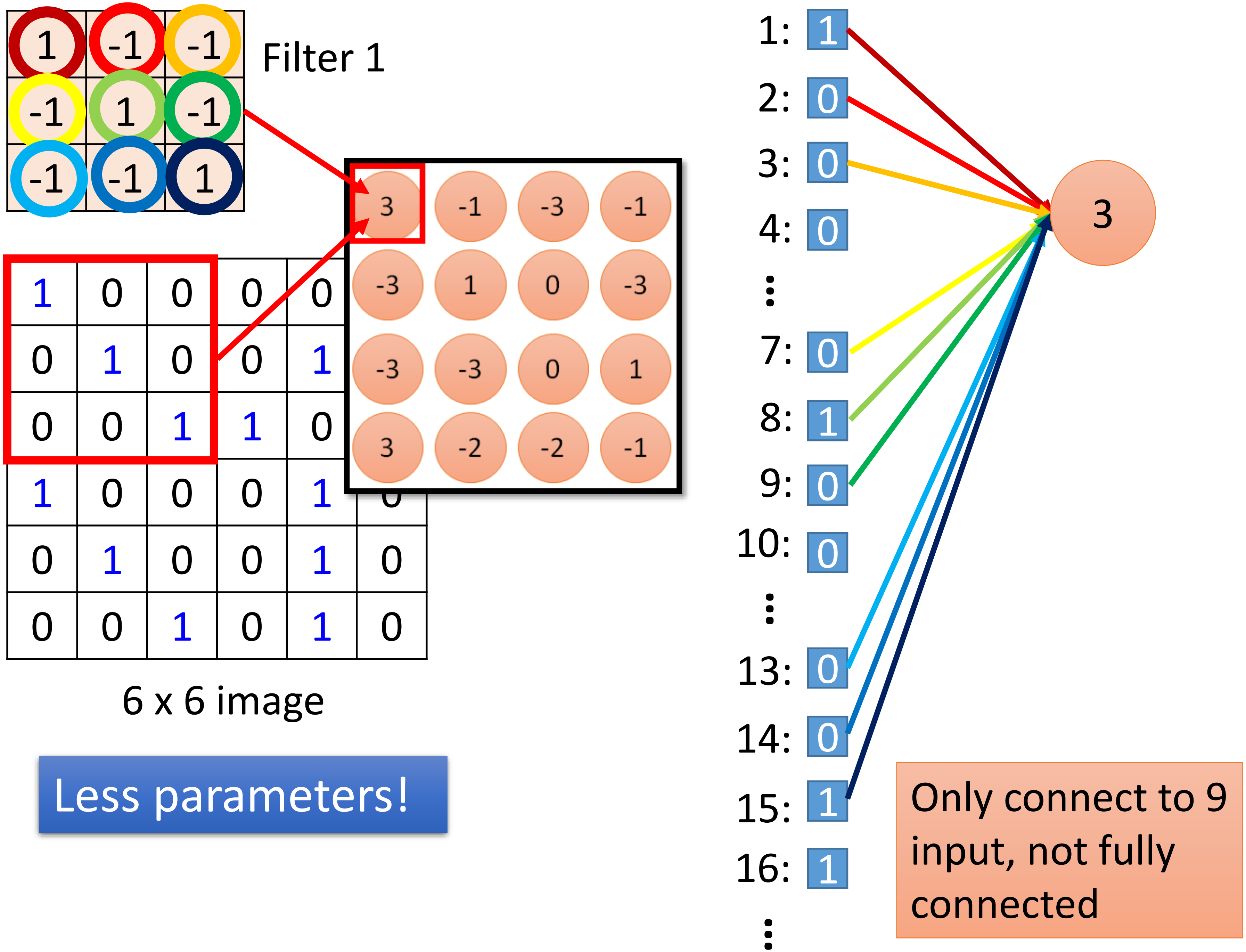
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

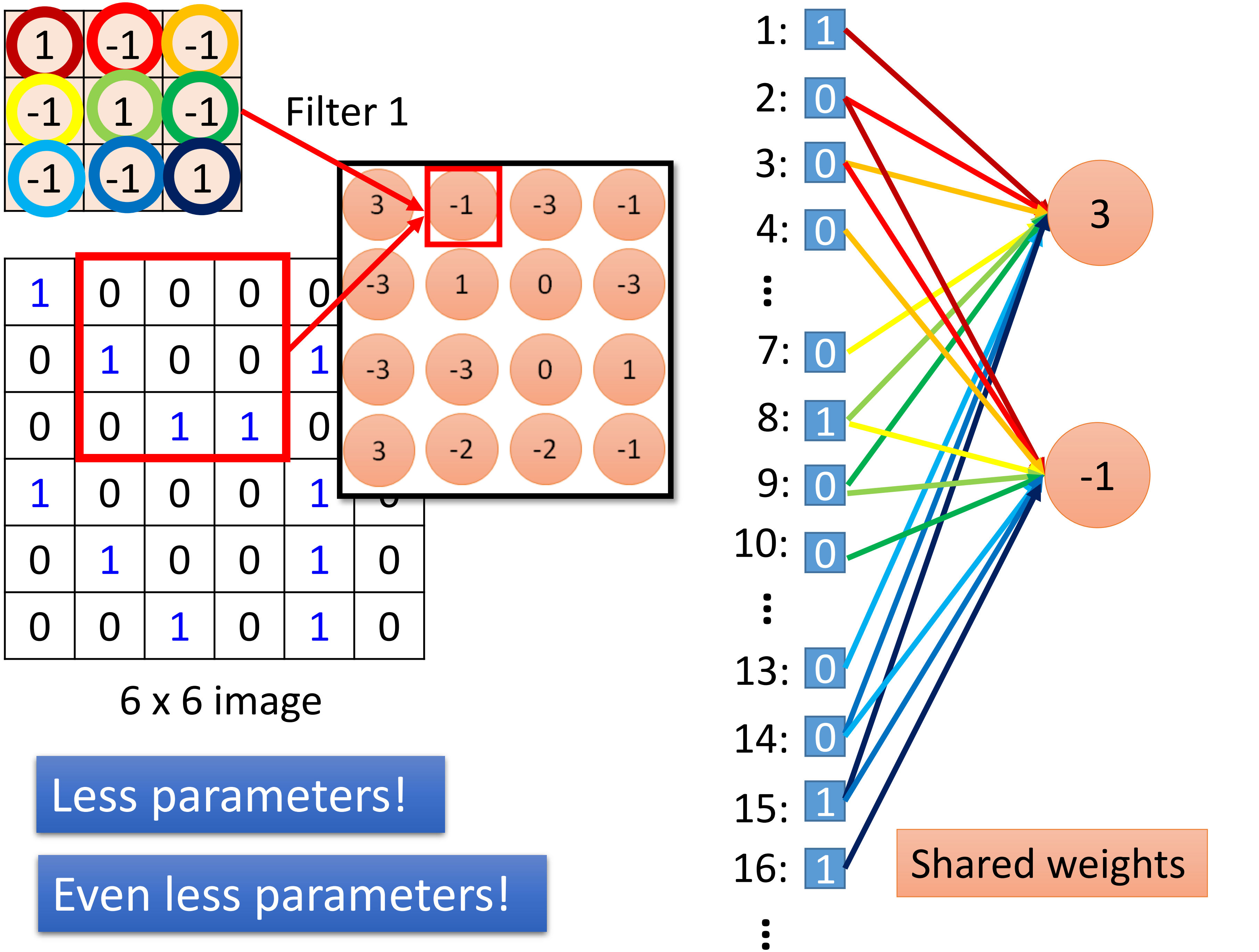


Convolution v.s. Fully Connected

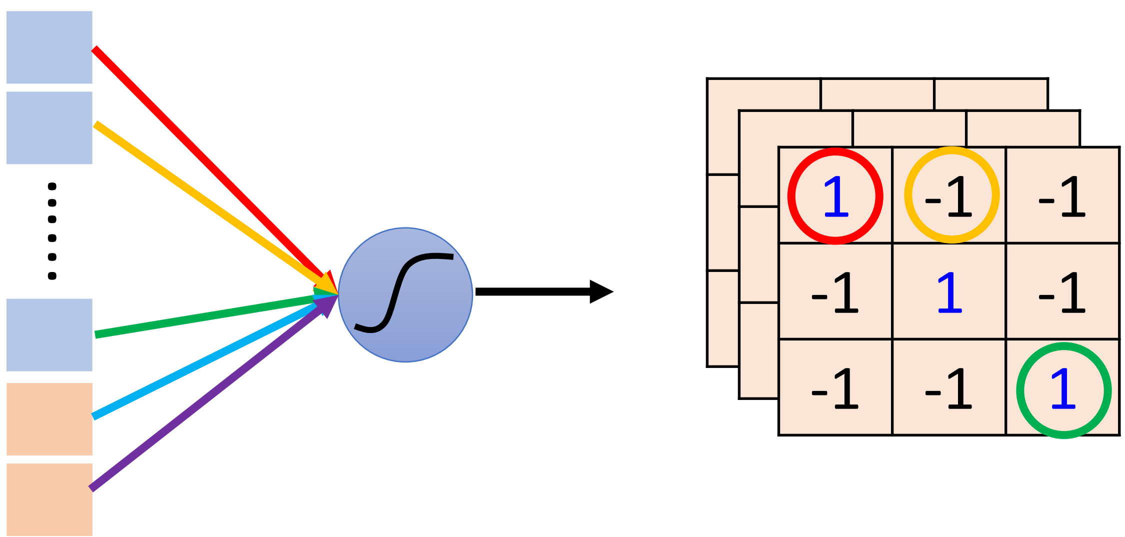





Filter 1
6 x 6 image
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected


Filter 1
6 x 6 image
Less parameters!
Even less parameters!
1:
2:
3:
4:
7:
8:
9:
10:
13:
14:
15:
16:
3
-1
Shared weights

Comparison of Two Stories

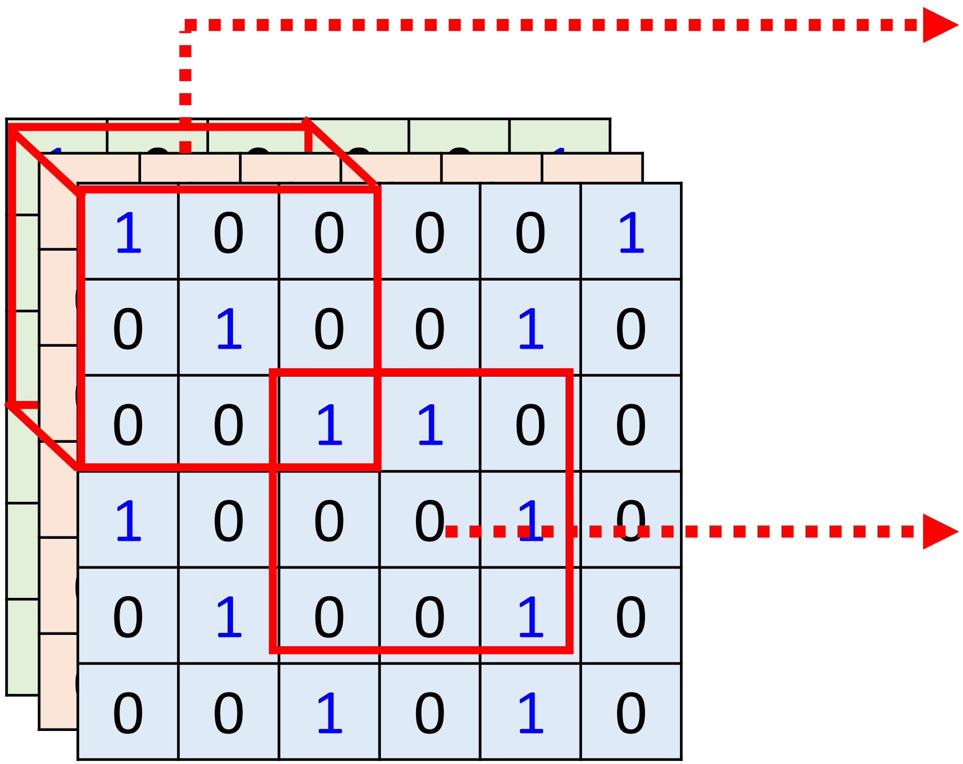


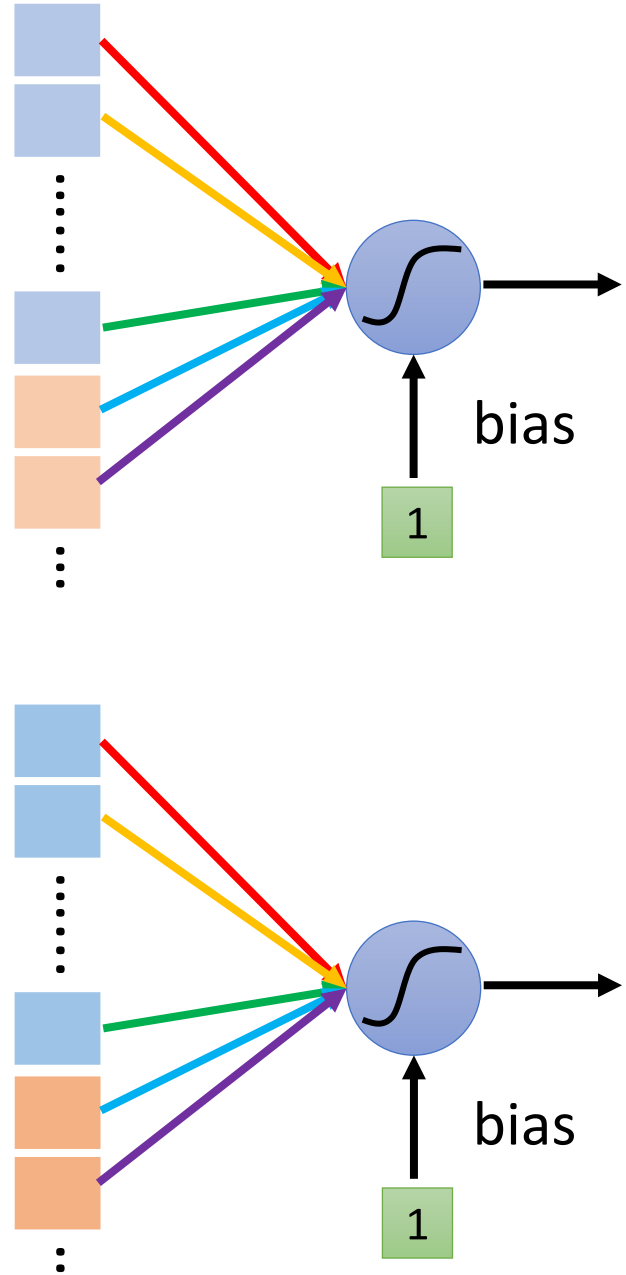
Receptive field

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

We can subsample the pixels to make image smaller

Less parameters for the network to process the image

Pooling – Max Pooling

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

Filter 1

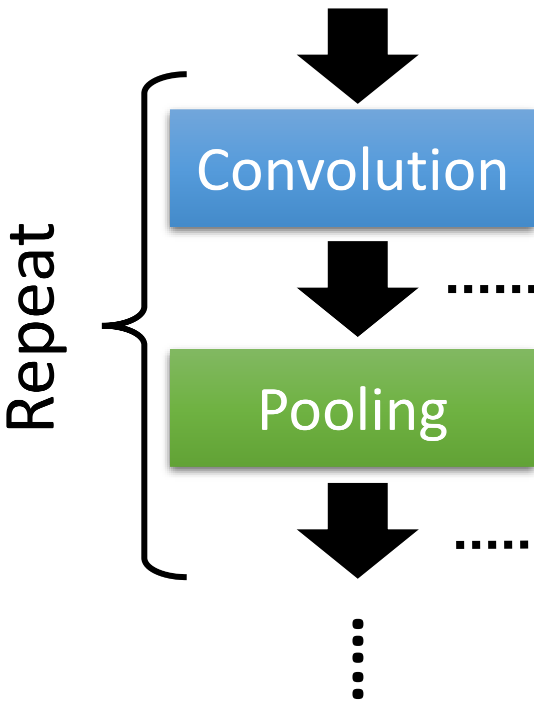
-1	1	-1
-1	1	-1
-1	1	-1

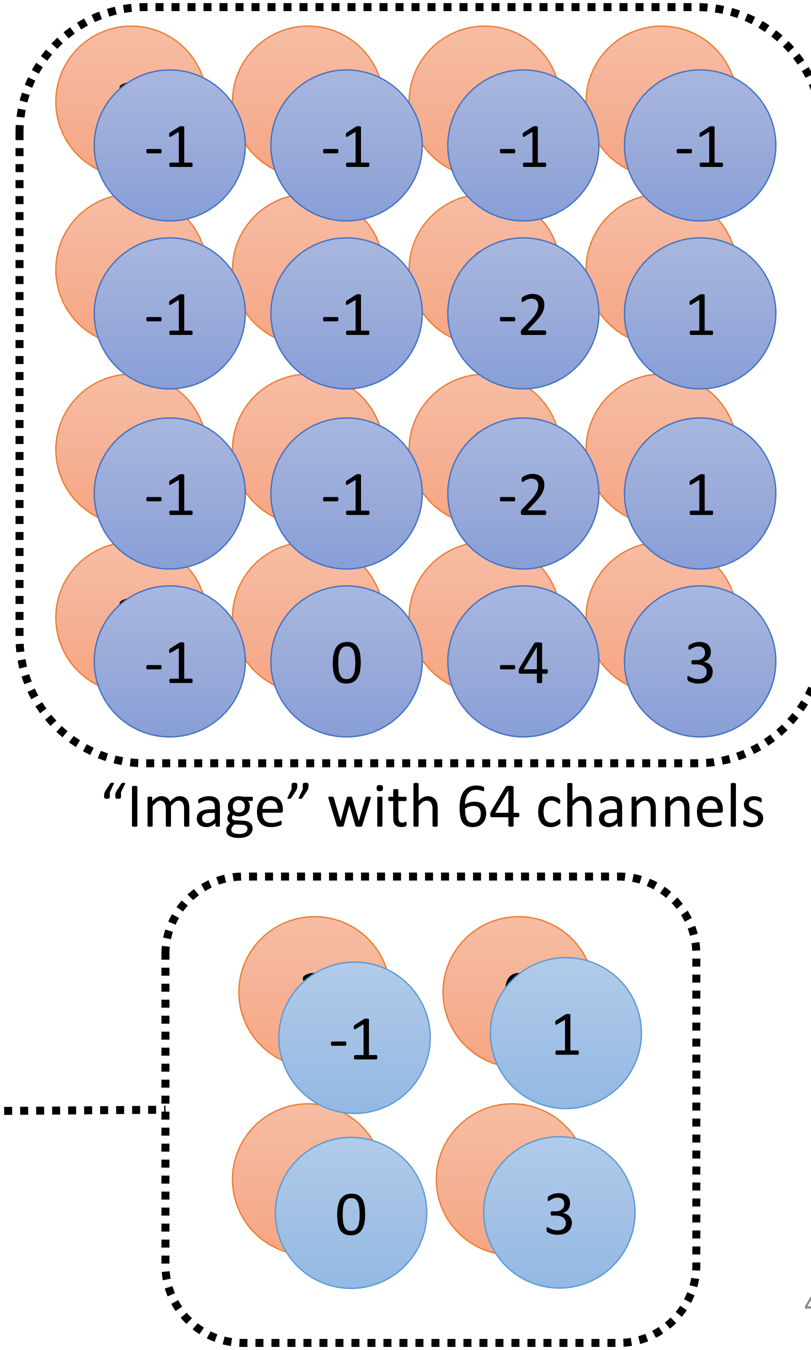
Filter 2

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

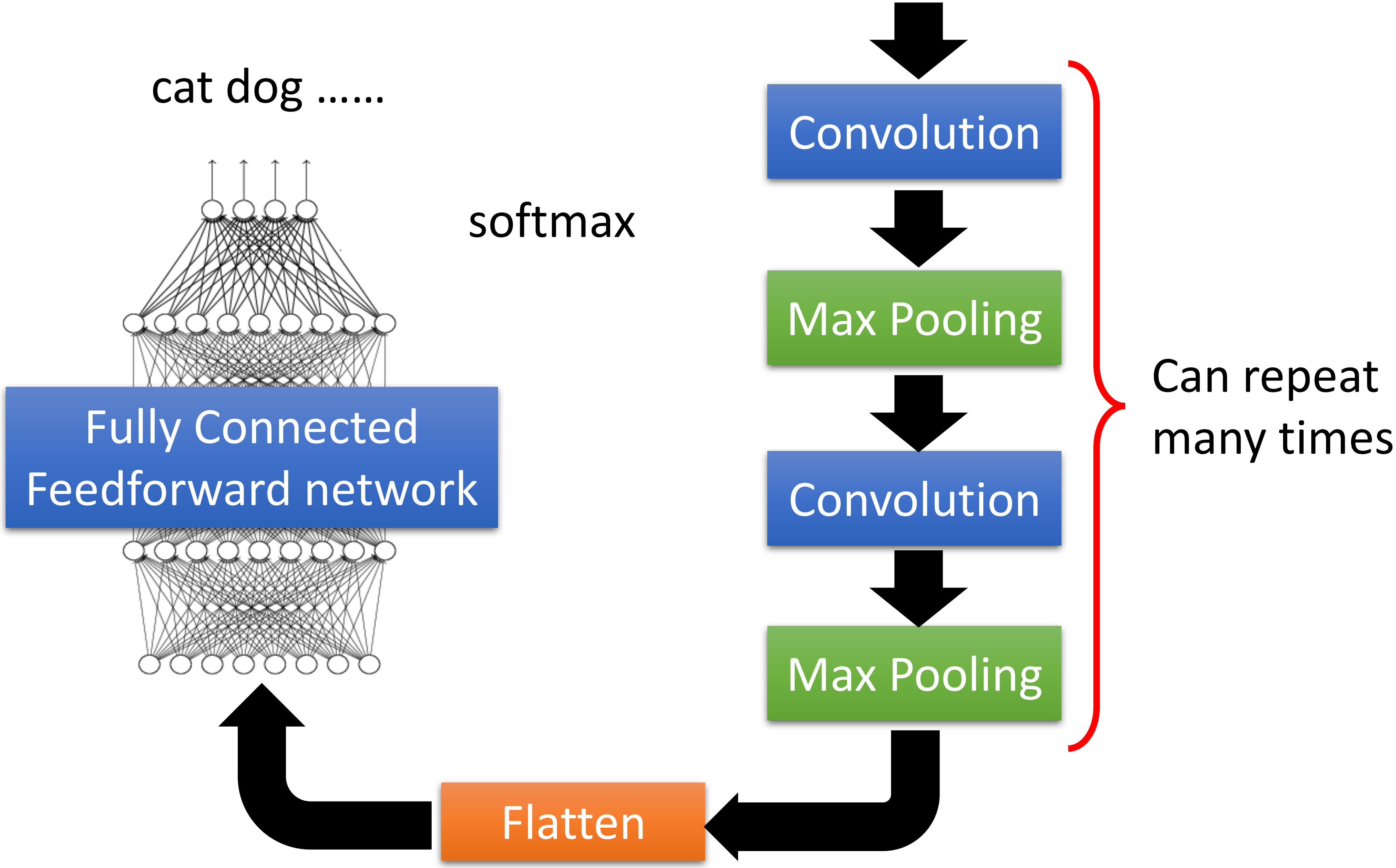
-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

Convolutional Layers + Pooling





The whole CNN



The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

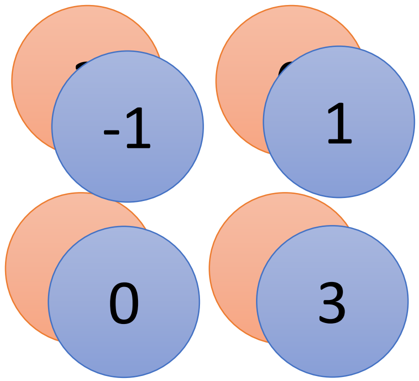
Convolution

Max Pooling

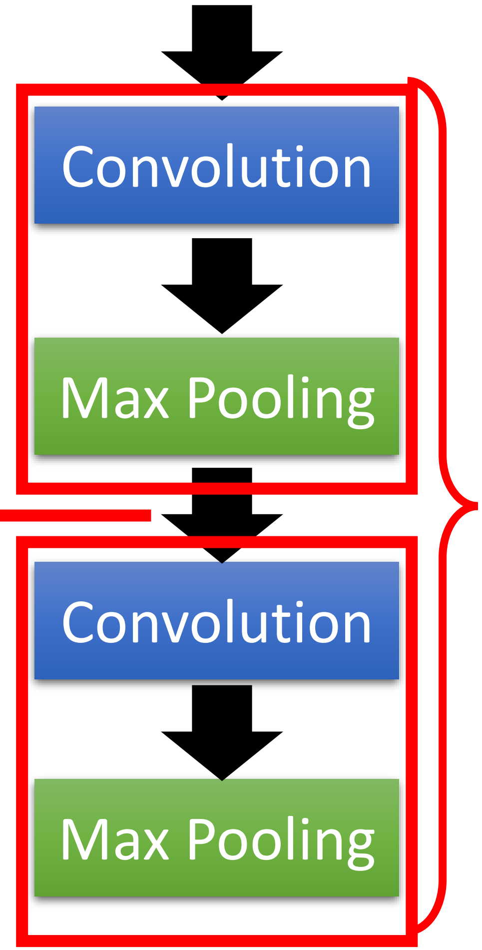
Can repeat many times

Flatten

The whole CNN



A new image

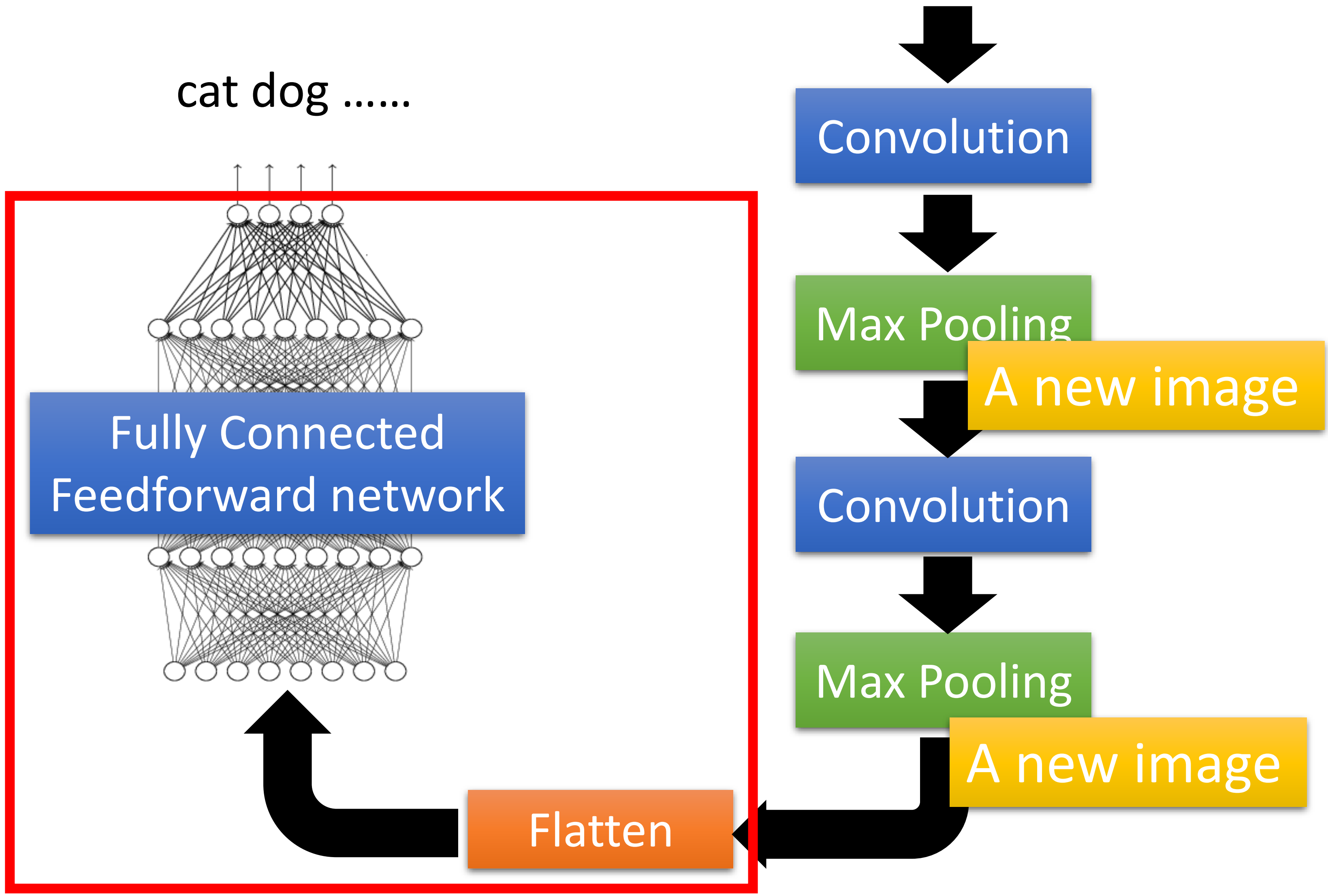


Can repeat many times

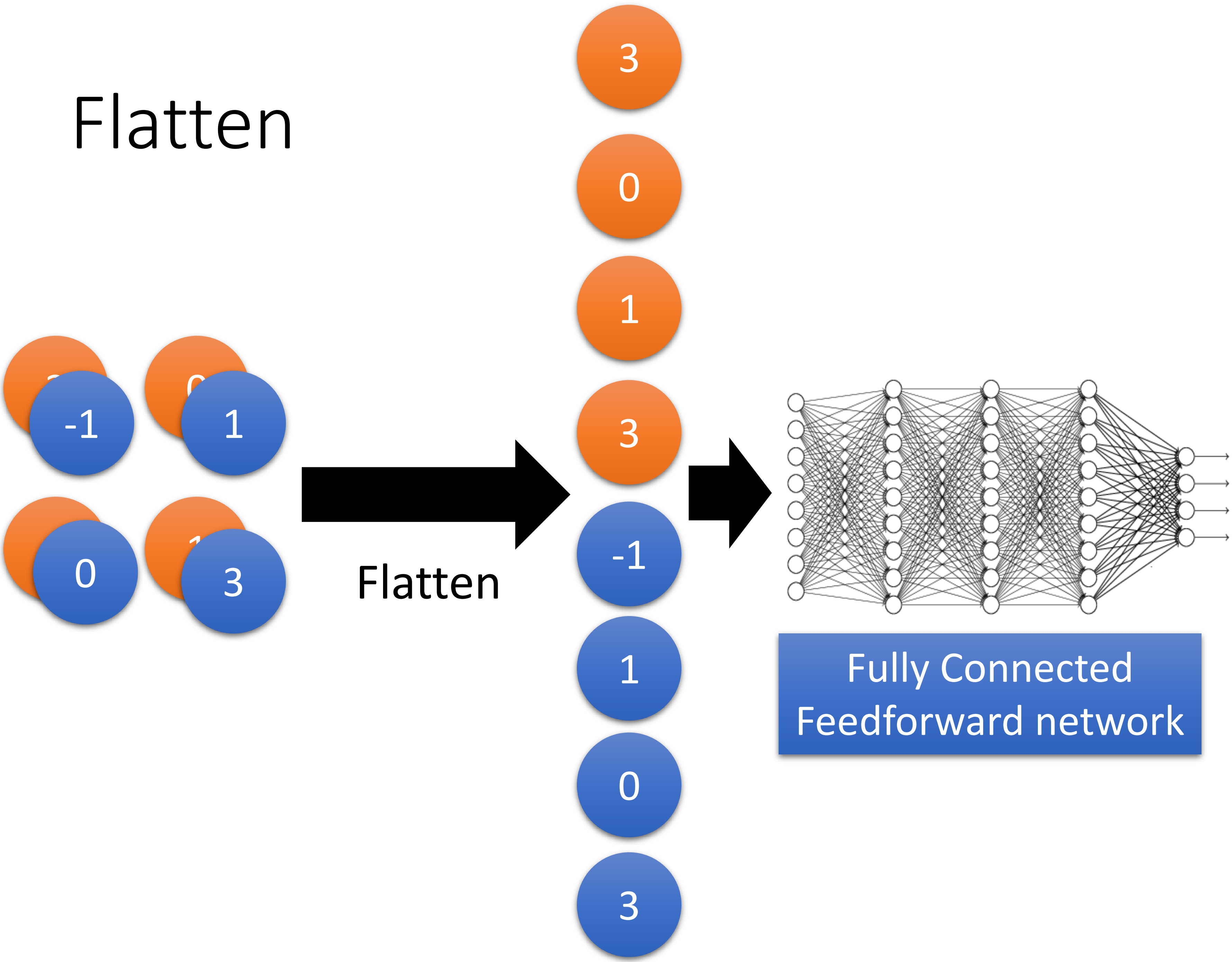
Smaller than the original image

The number of the channel is the number of filters

The whole CNN



Flatten


-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

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

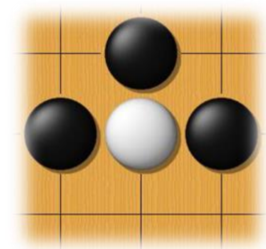
Fully-connected network can be used

But CNN performs much better.

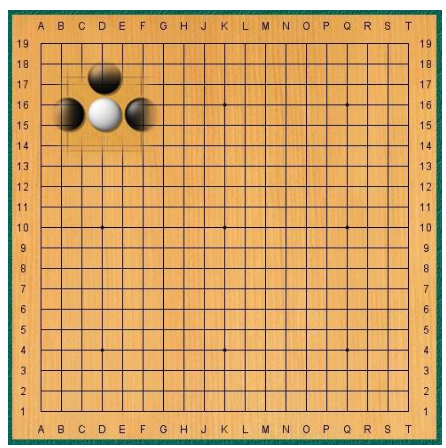
Why CNN for Go playing?

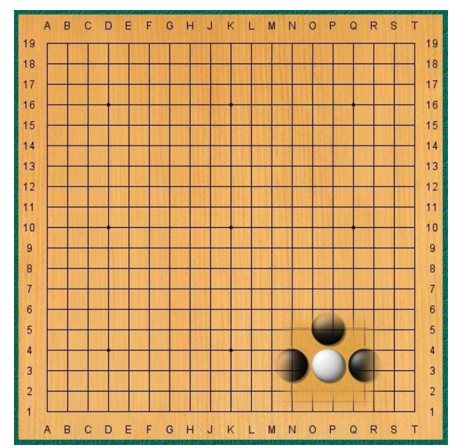
- Some patterns are much smaller than the whole image

  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

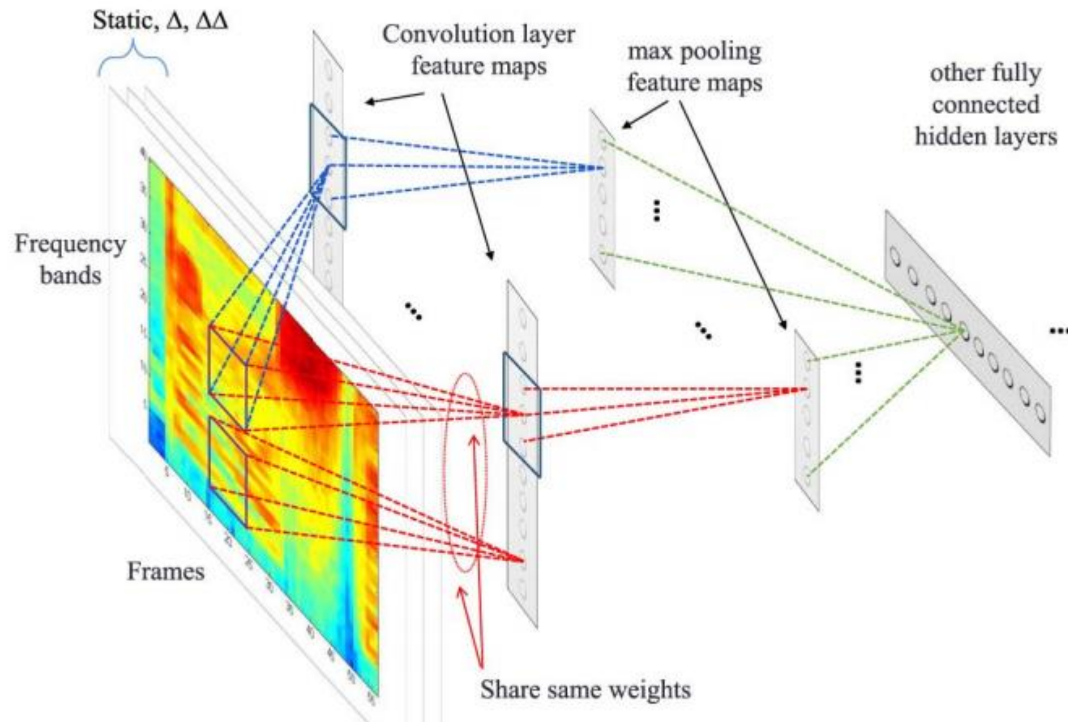
→ Pooling

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 23×23 image, then convolves k filters of kernel size 5×5 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 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 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

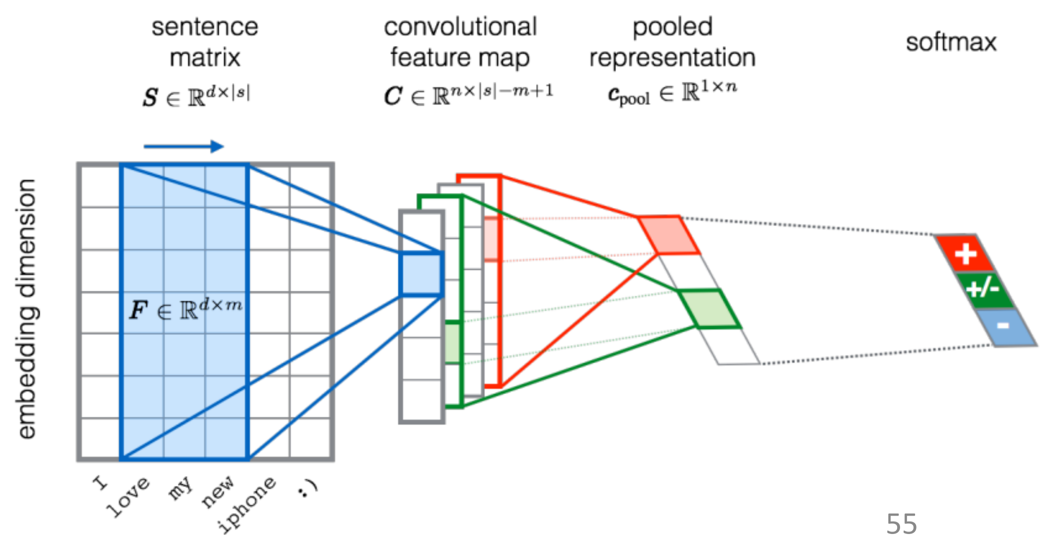
Alpha Go does not use Pooling ……

More Applications



Speech

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



Natural Language Processing

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