



Kissipo Learning for Deep Learning

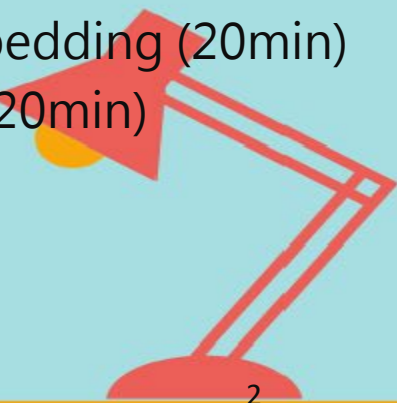
Topic 12: CNN with TensorFlow (20min)

Hsueh-Ting Chu

KLDL-W4-T11

Topics

- Topic 01: Introduction to Deep Learning (20min)
- Topic 02: KISSIPO Learning for Deep Learning (20min)
- Topic 03: Python quick tutorial (20min)
- Topic 04: Numpy quick tutorial (15min)
- Topic 05: Pandas quick tutorial (15min)
- Topic 06: Scikit-learn quick tutorial (15min)
- Topic 07: OpenCV quick tutorial (15min)
- Topic 08: Image Processing basics (20min)
- Topic 09: Machine Learning basics (20min)
- Topic 10: Deep Learning basics (20min)
- Topic 11: TensorFlow overview (20min)
- **Topic 12: CNN with TensorFlow (20min)**
- Topic 13: RNN with TensorFlow (20min)
- Topic 14: PyTorch overview (20min)
- Topic 15: CNN with PyTorch (20min)
- Topic 16: RNN with Pytorch (20min)
- Topic 17: Introduction to AOI (20min)
- Topic 18: AOI simple Pipeline (A) (20min)
- Topic 19: AOI simple Pipeline (B) (20min)
- Topic 20: Introduction to Object detection (20min)
- Topic 21: YoloV5 Quick Tutorial (20min)
- Topic 22: Using YoloV5 for RSD (20min)
- Topic 23: Introduction to NLP (20min)
- Topic 24: Introduction to Word Embedding (20min)
- Topic 25: Name prediction project (20min)



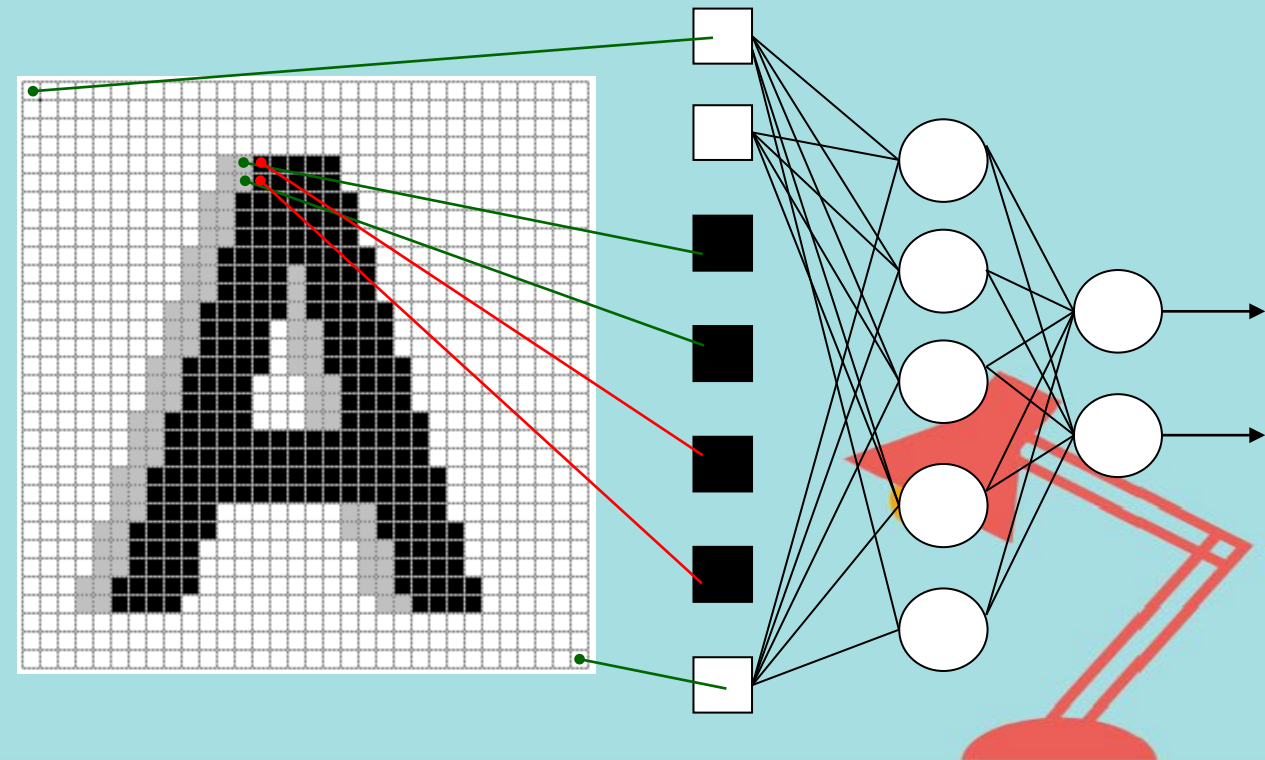
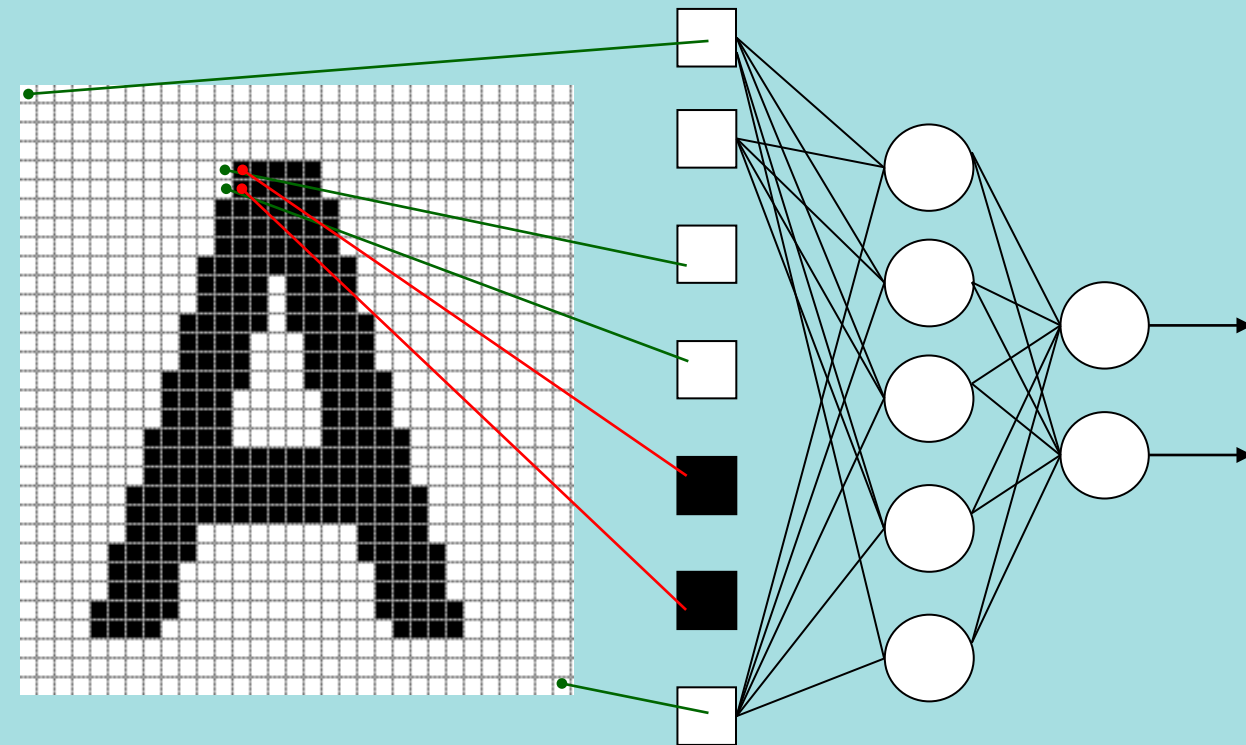
Content

- Topic 12: CNN with TensorFlow (20min)
 - Drawbacks of Feed Forward neural networks
 - Convolutional neural network
 - Training a CNN model for MNIST dataset

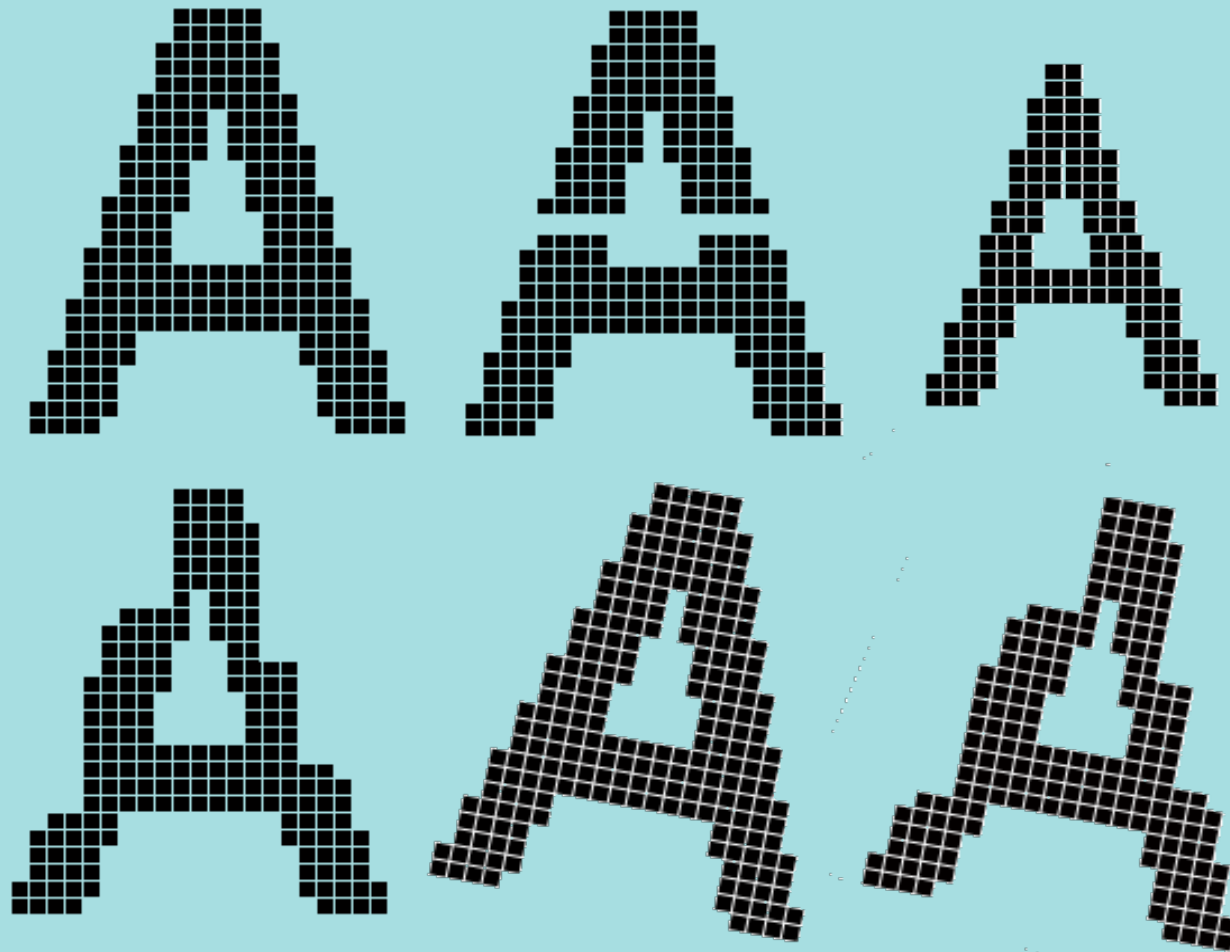


Drawbacks of Feed Forward neural networks

Little or no invariance to shifting, scaling, and other forms of distortion



Scaling, and other forms of distortion



AAA
AAA



CNN History



Yann LeCun, Professor of Computer Science
The Courant Institute of Mathematical Sciences
New York University
Room 1220, 715 Broadway, New York, NY 10003, USA.
(212)998-3283 yann@cs.nyu.edu

In 1995, Yann LeCun and Yoshua Bengio introduced the concept of convolutional neural networks.



Convolutional neural network(CNN)

Add convolution and pooling layers before feedforward neural network

Convolution

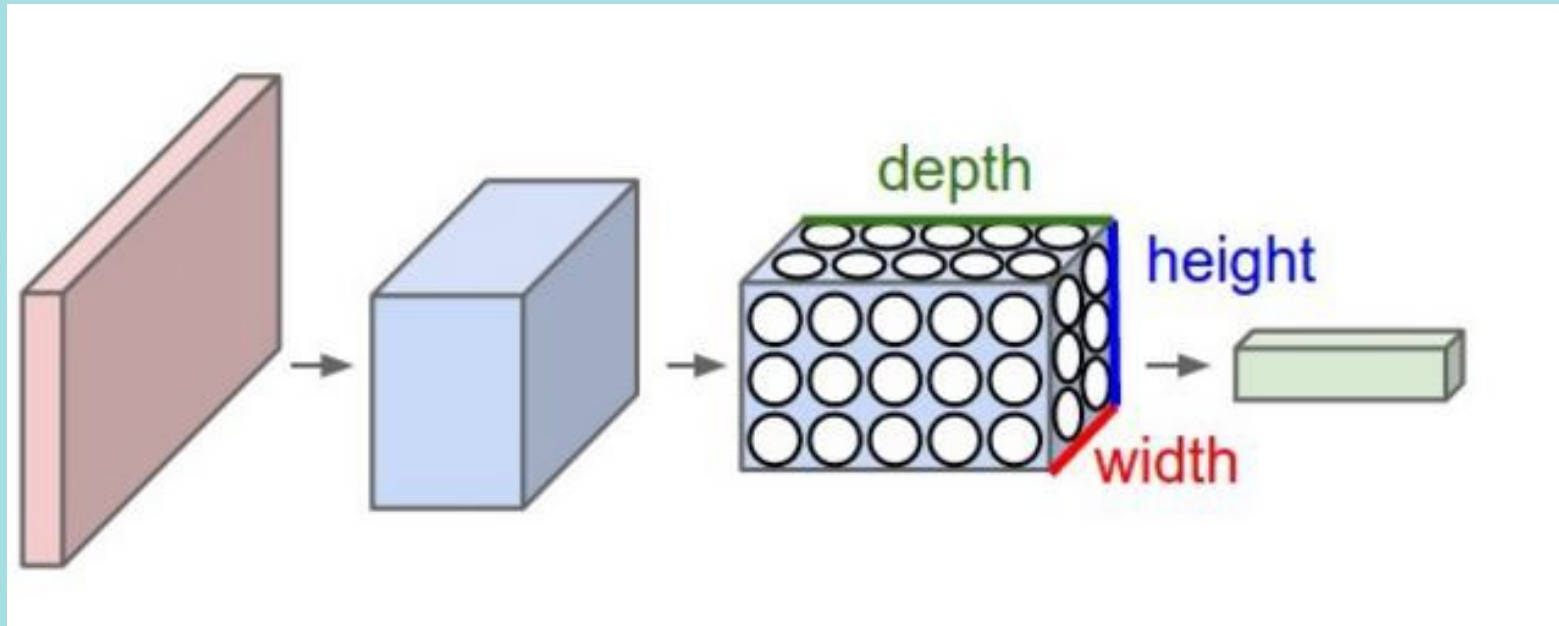
Pooling

```
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1),  
padding='valid', data_format=None, dilation_rate=(1, 1),  
activation=None, use_bias=True,  
kernel_initializer='glorot_uniform', bias_initializer='zeros',  
kernel_regularizer=None, bias_regularizer=None,  
activity_regularizer=None, kernel_constraint=None,  
bias_constraint=None)
```

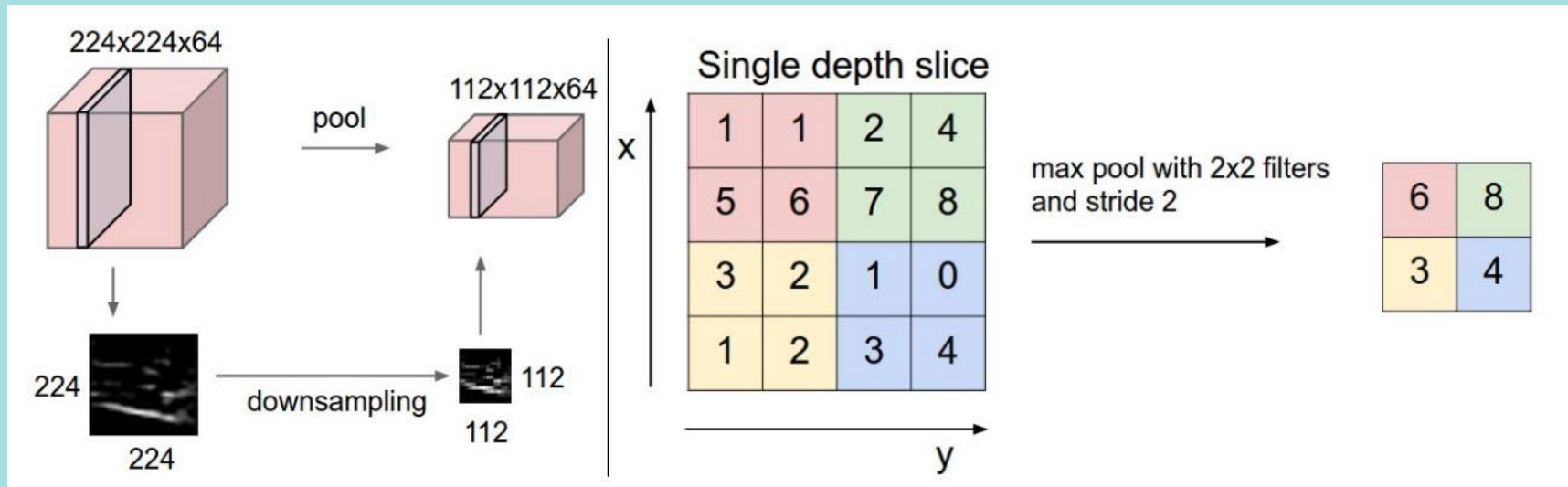
```
keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None,  
padding='valid', data_format=None)
```



Convolutional layer



Pooling layer



Convolution

These 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

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

Filter 2

⋮ ⋮

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



Convolution

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

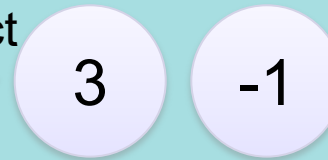
Filter 1

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

Dot
product



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



Convolution

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

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

Filter 1

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



Convolution

-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

Repeat this for each filter

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

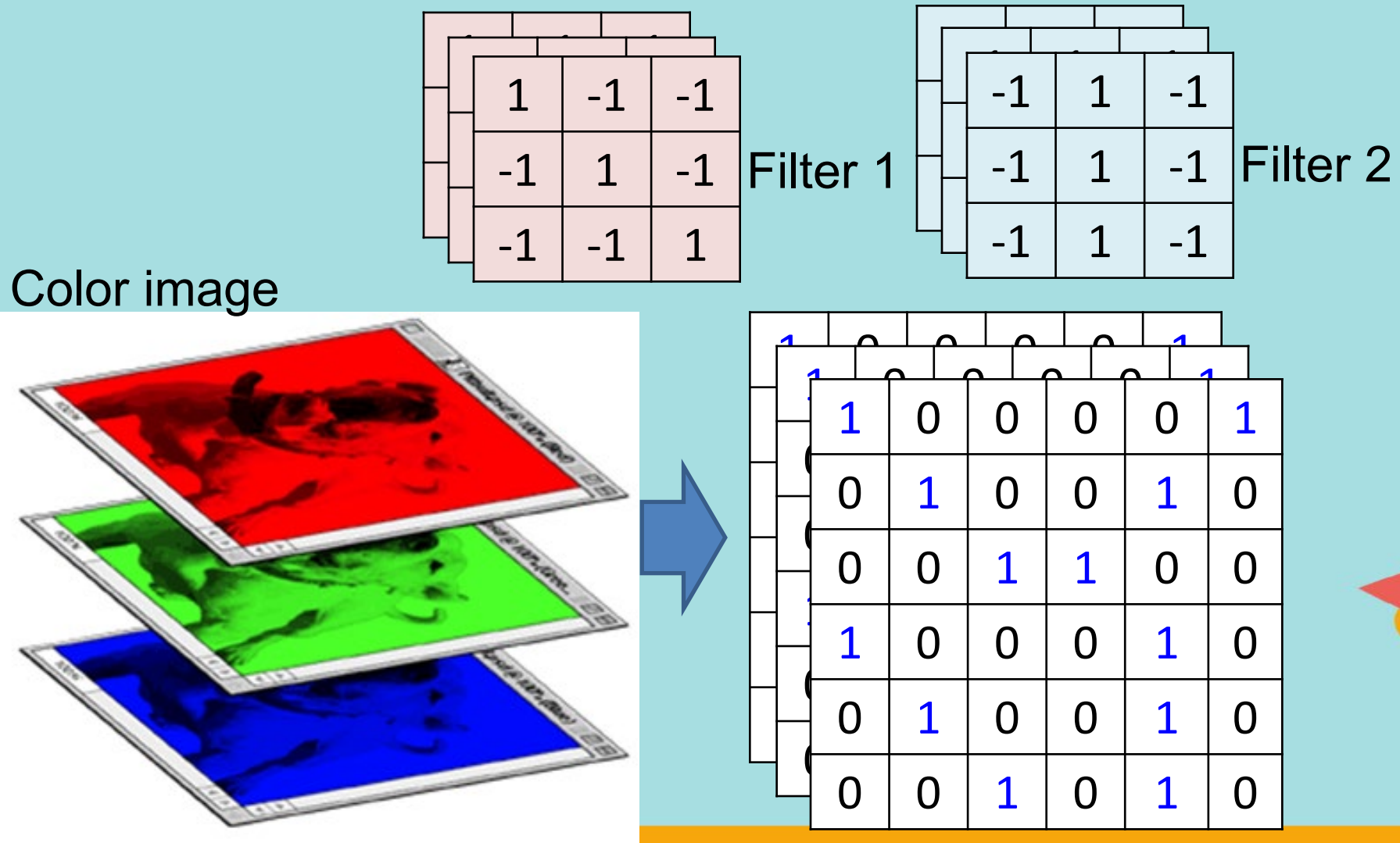
Feature Map

Two 4 x 4 images

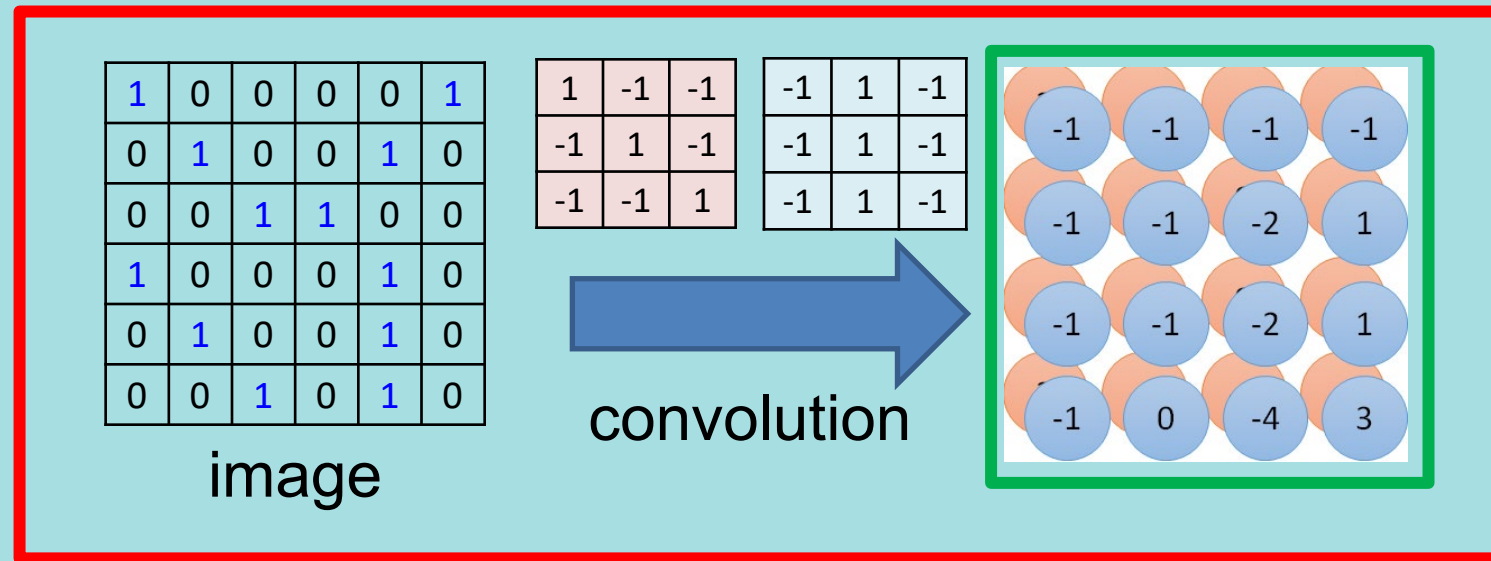
Forming 2 x 4 x 4 matrix



Color image: RGB 3 channels

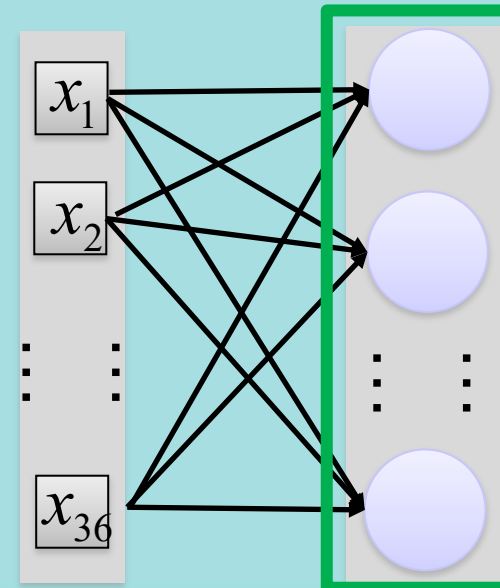


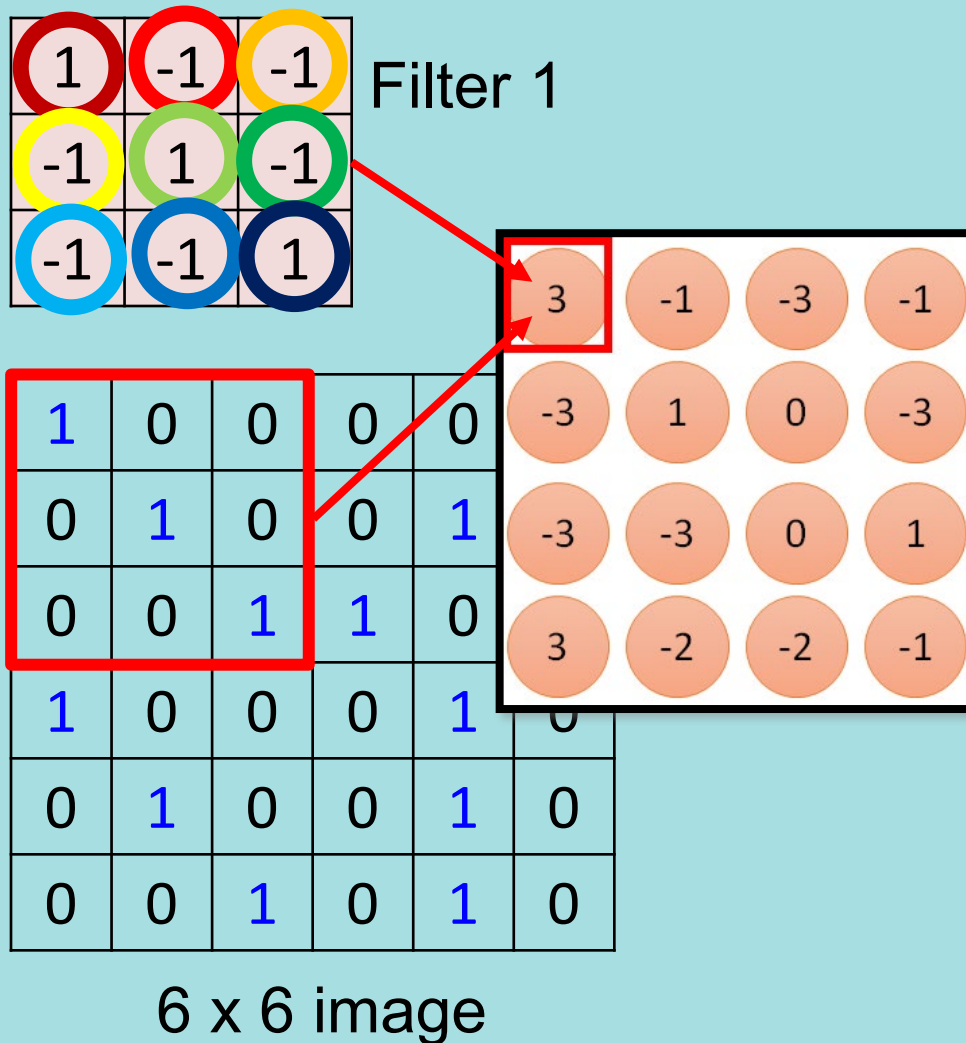
Convolution v.s. Fully Connected



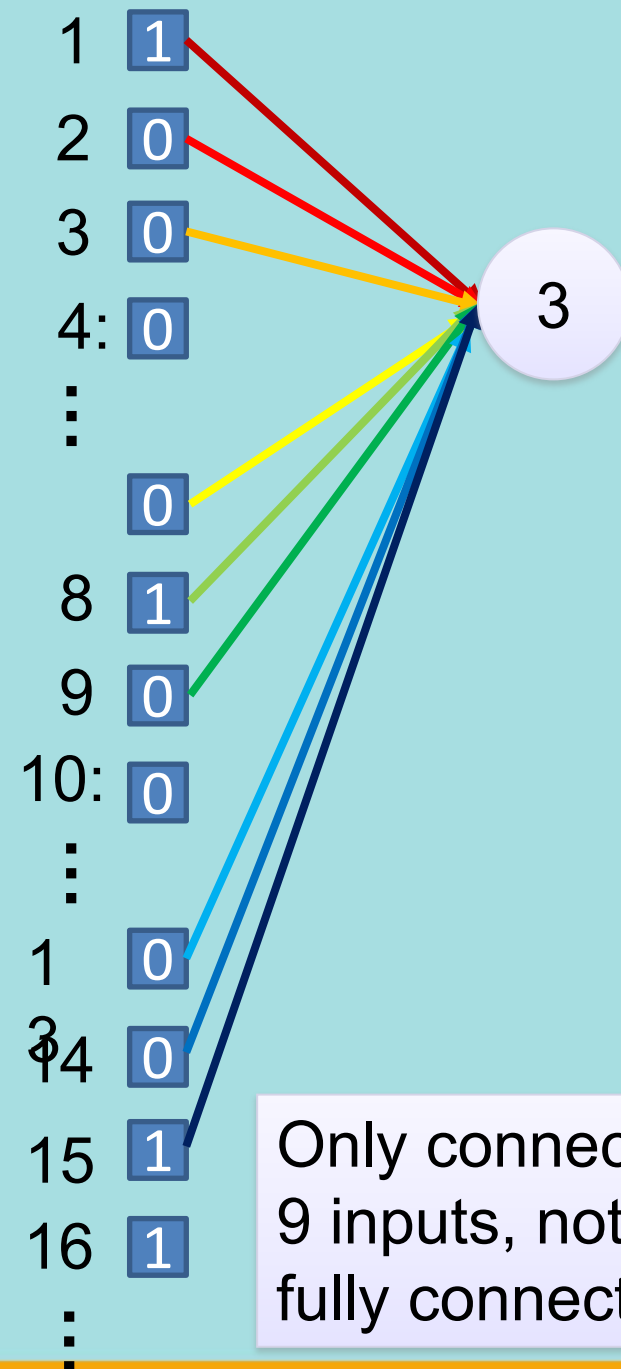
Fully-
connected

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



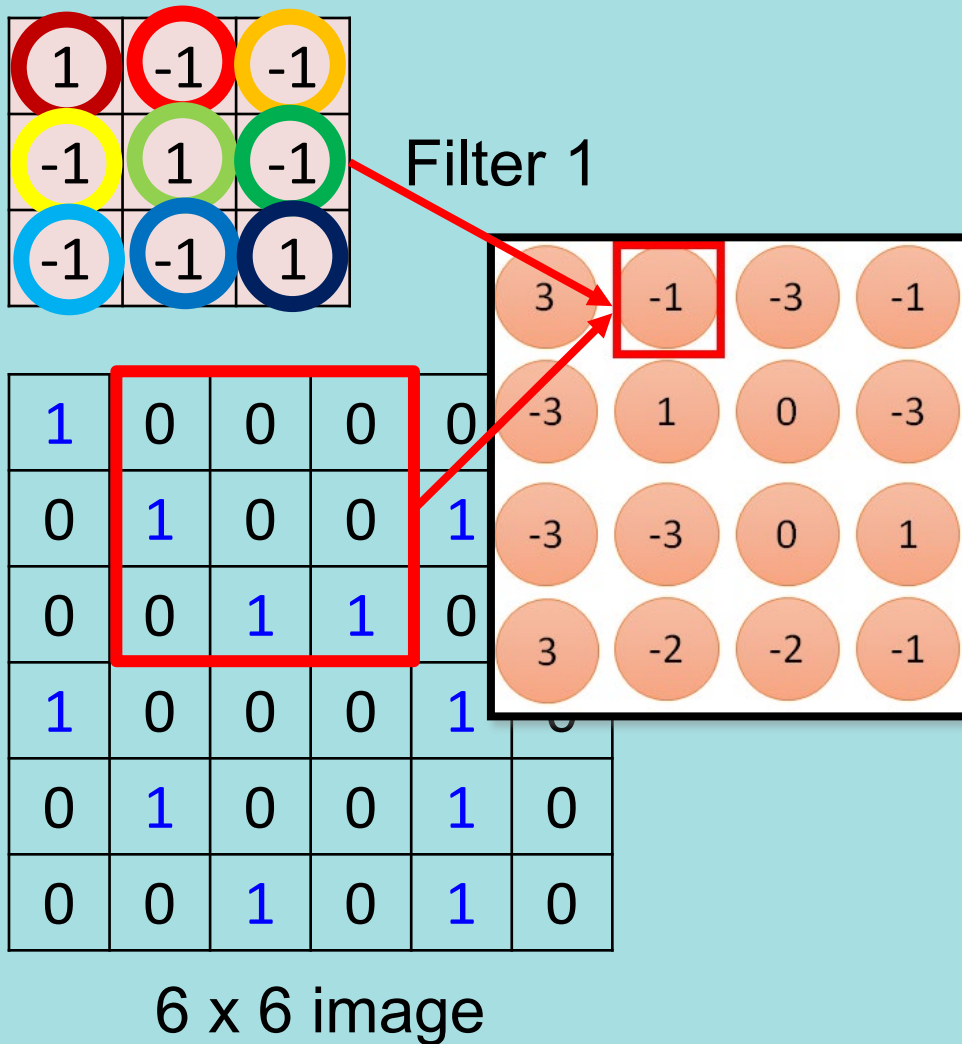


fewer parameters!



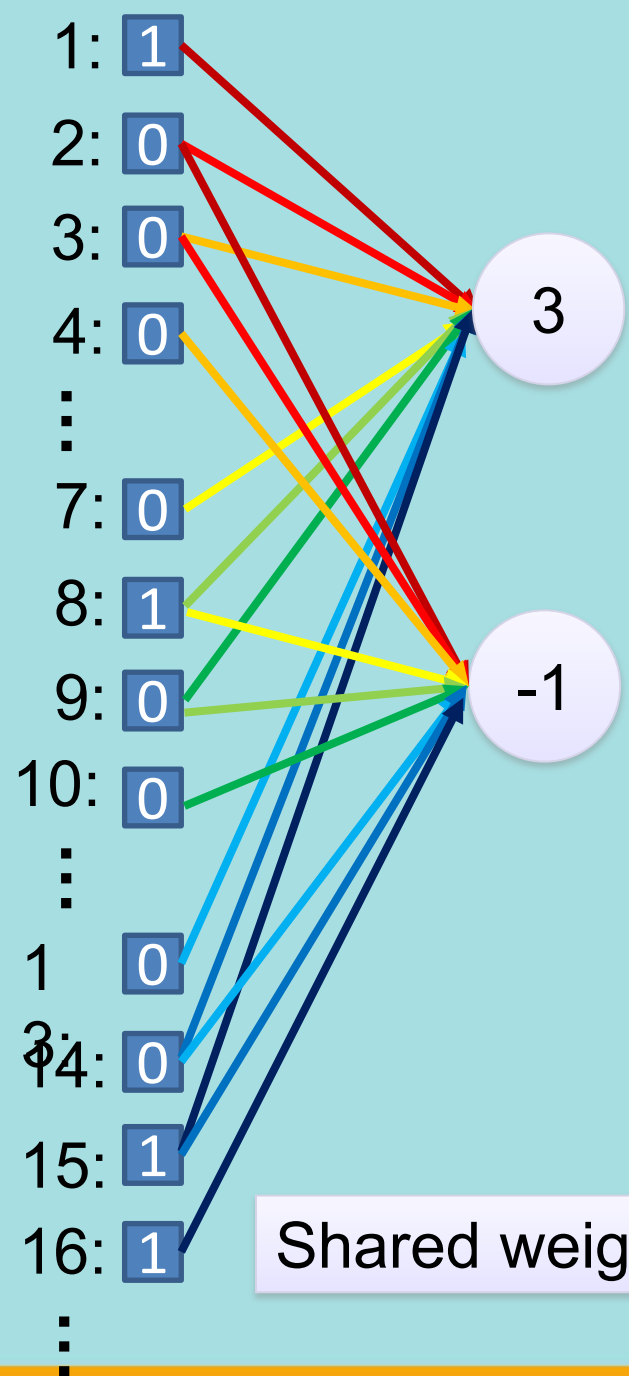
Only connect to 9 inputs, not fully connected





Fewer parameters

Even fewer parameters



Max Pooling

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

Filter 1

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

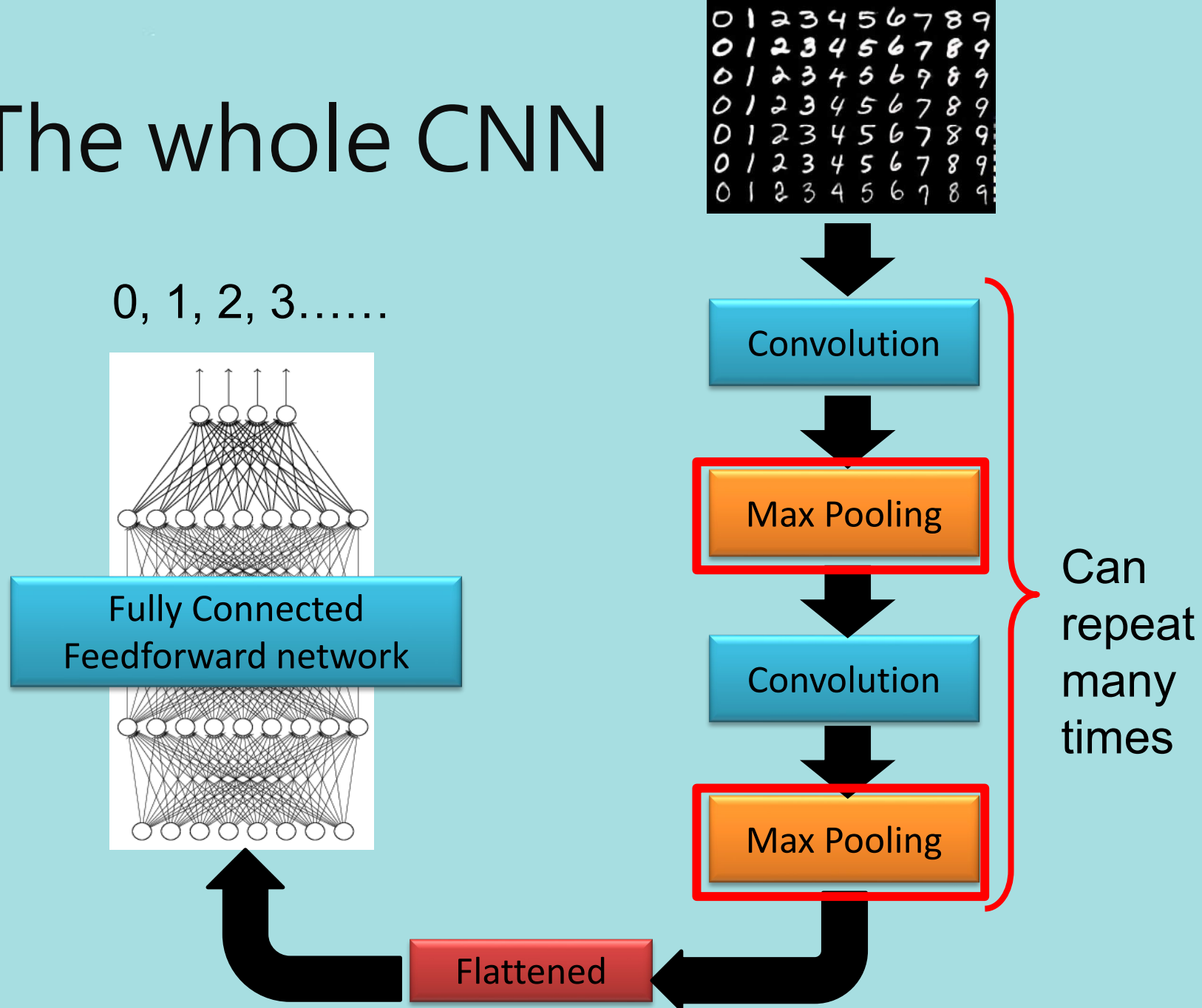
Filter 2

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

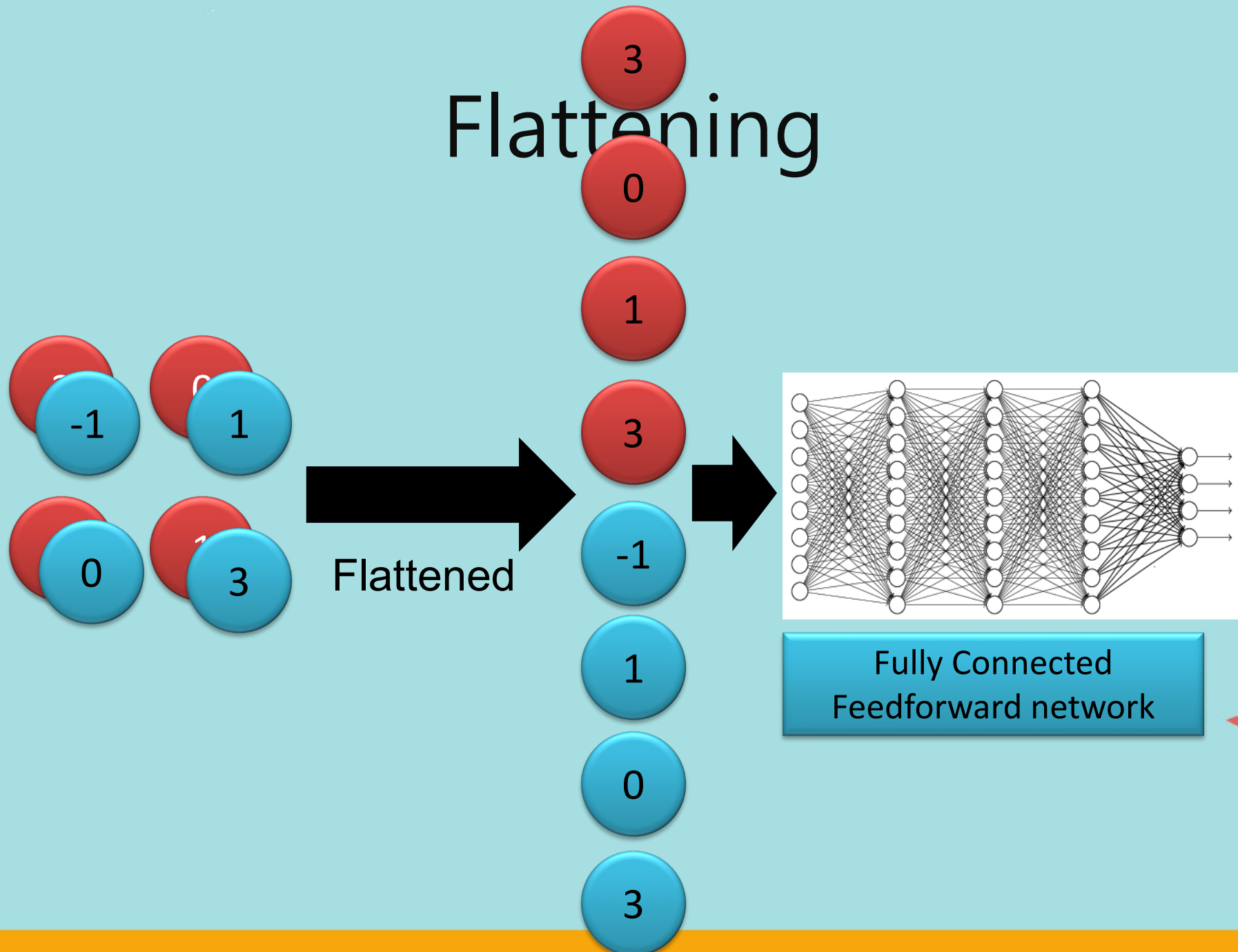
-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3



The whole CNN



Flattening





Thanks!

Q&A

