

Convolutional Neural Networks

Dr. Dongchul Kim

CNN

CNN, or Convolutional Neural Network, primarily utilized in image and video recognition, comprises five essential layers:

Input Layer

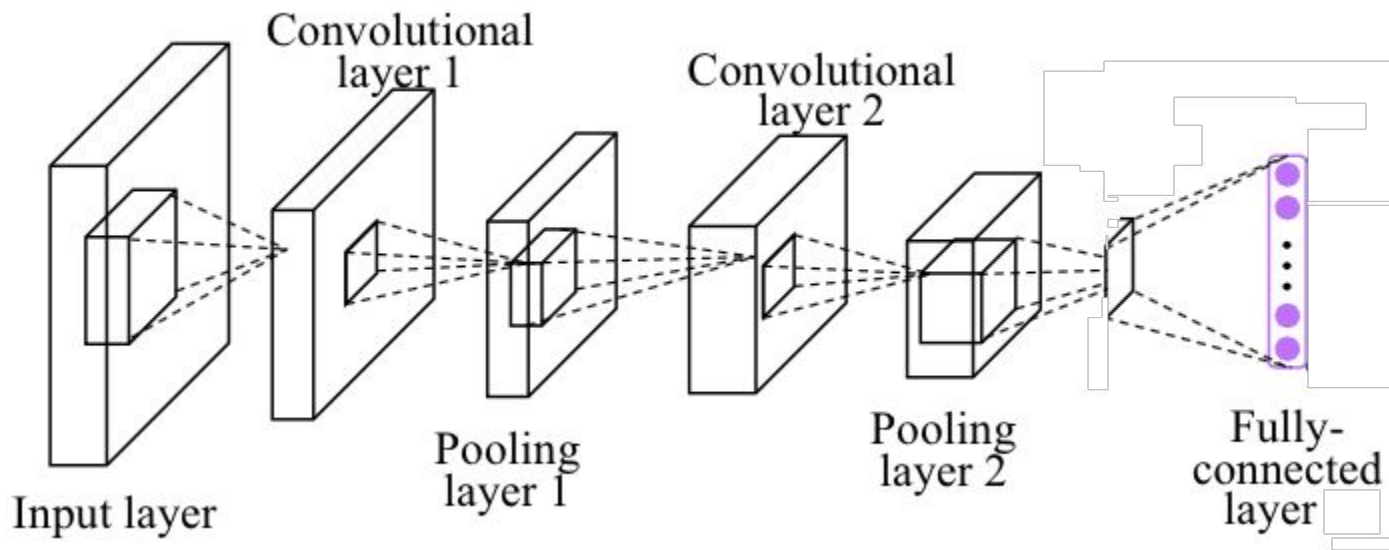
Convolution Layer

Pooling Layer

Fully Connected Layer

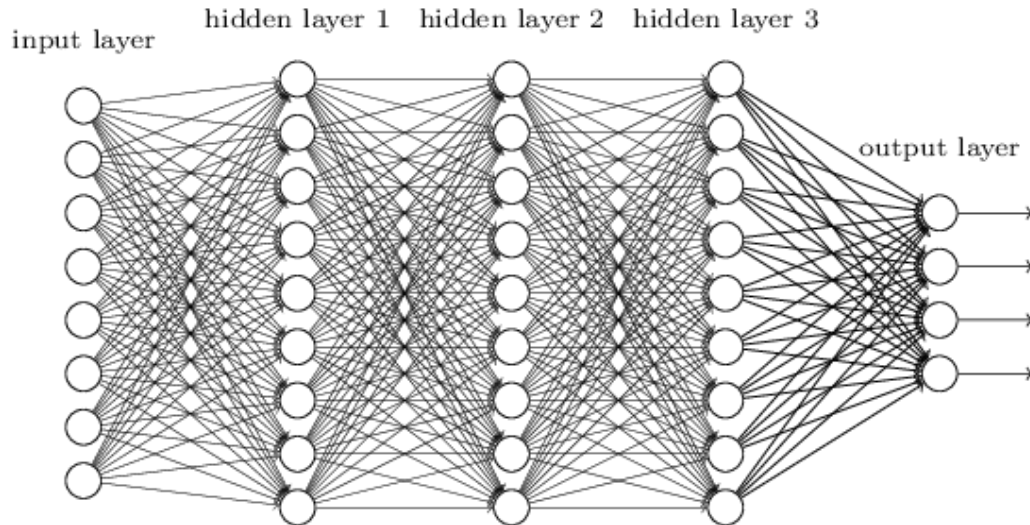
Output Layer

CNN Implementation



Why Convolutional Layer?

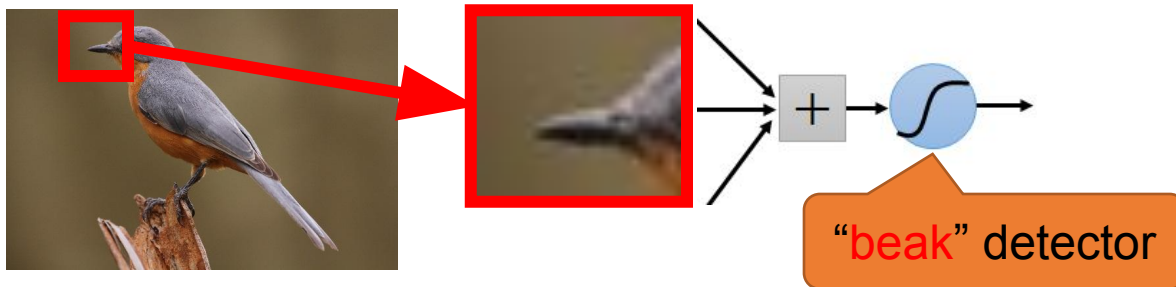
- We know the performance is sometimes good with a larger (deeper) model.
- From this fully connected model (Dense layers), do we really need all the edges?



Consider learning an image:

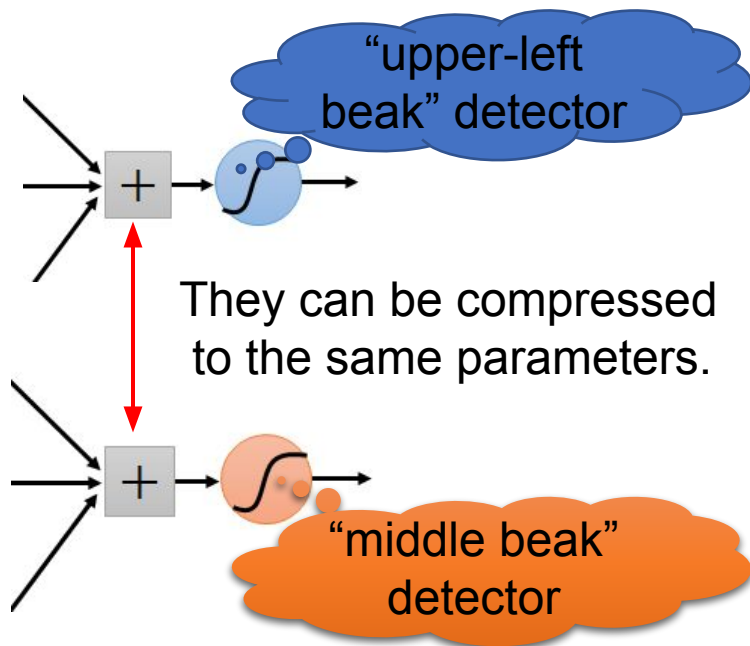
- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters



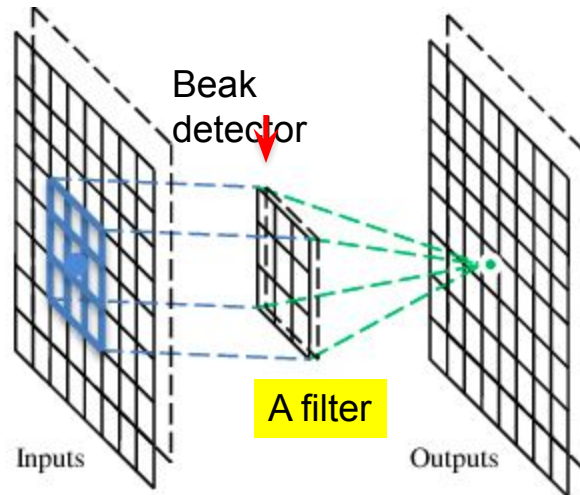
Same pattern appears in different places:
They can be compressed!

What about training a lot of such “small” detectors
and each detector must “move around”.



A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



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

6 x 6 image

These are the network parameters to be learned.

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

Convolution

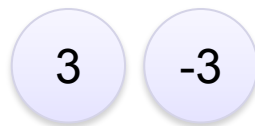
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

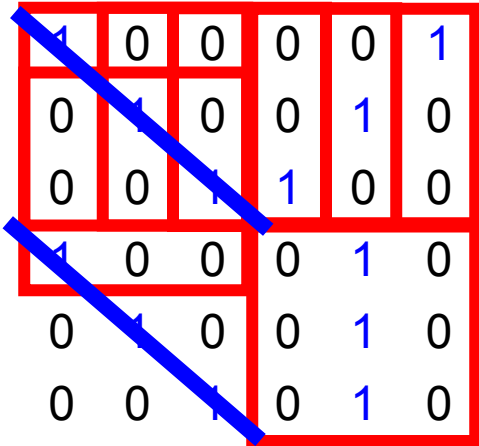
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

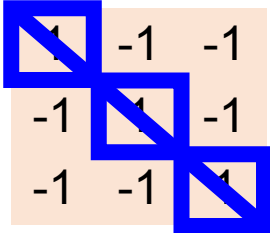


Convolution

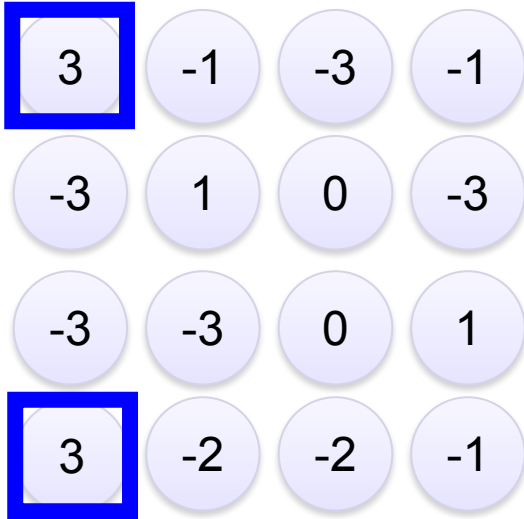
stride=1



6 x 6 image



Filter 1



Convolution

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

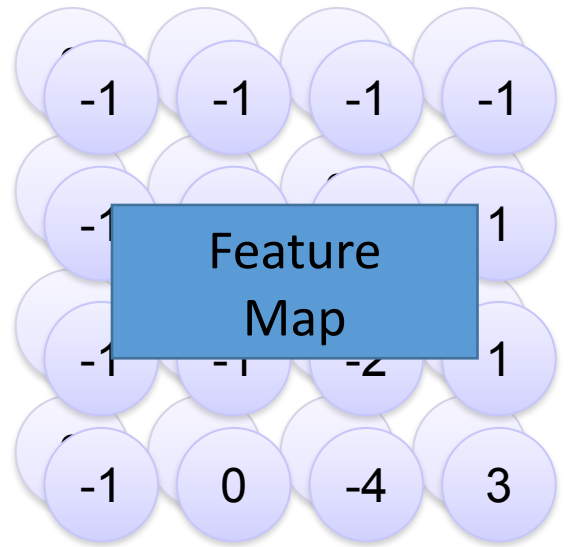
Filter 2

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

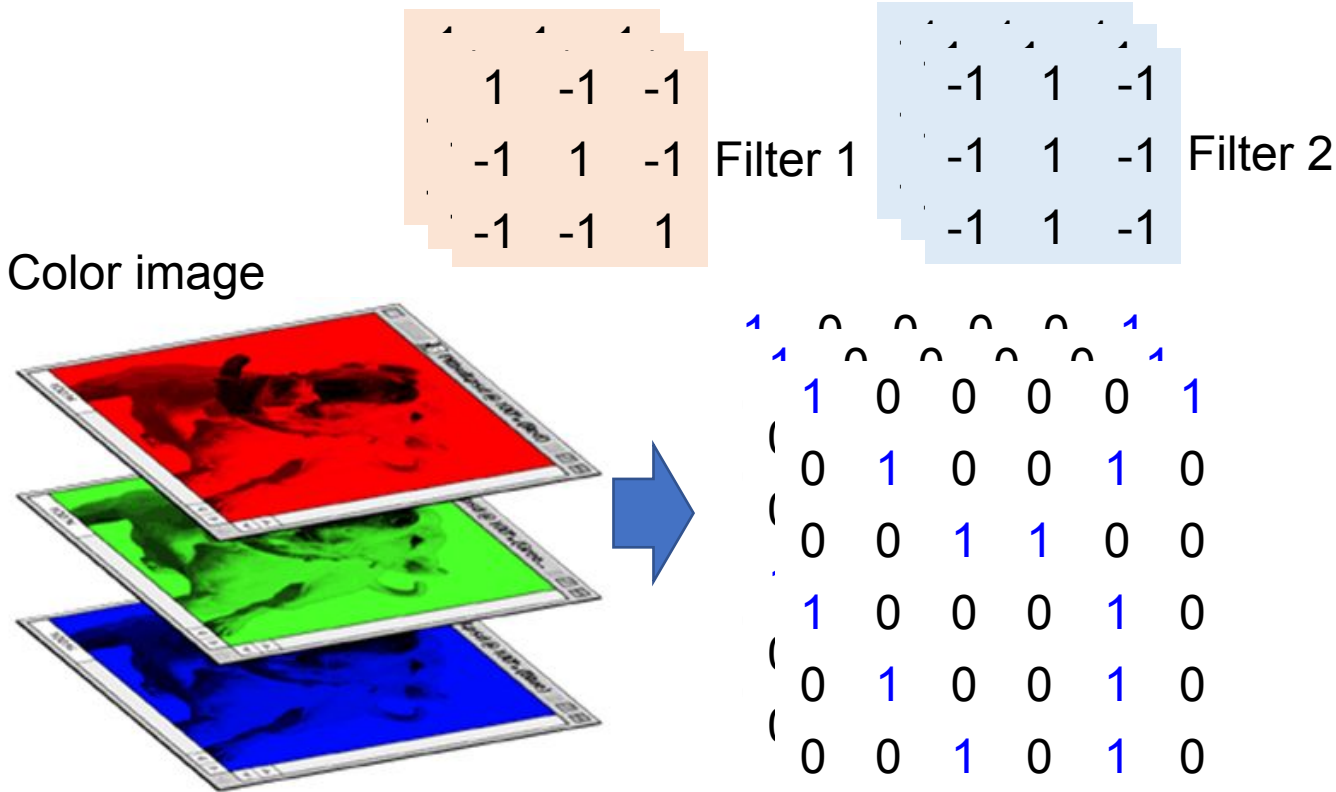
6 x 6 image

Repeat this for each filter

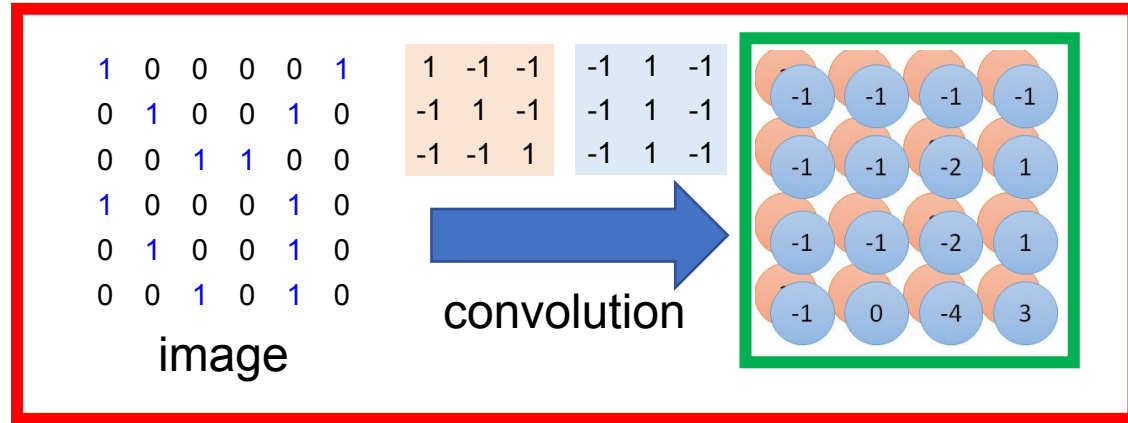


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 channels

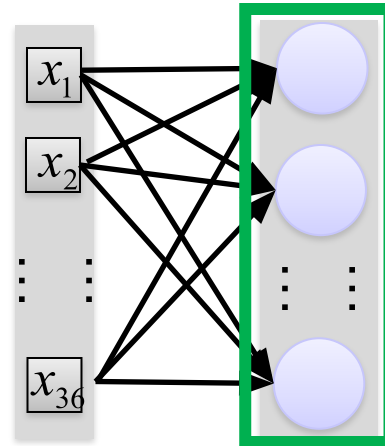


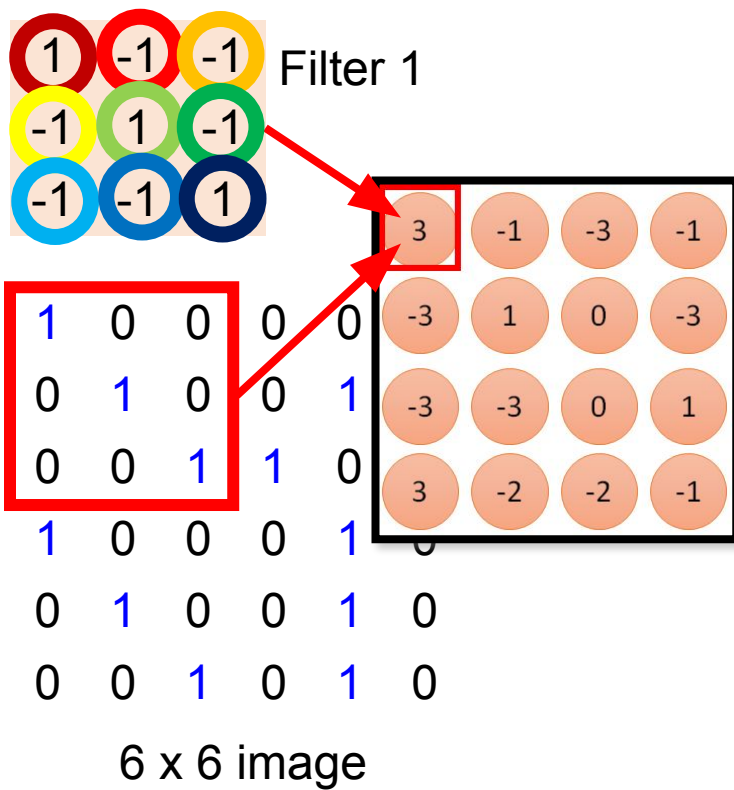
Convolution v.s. Fully Connected



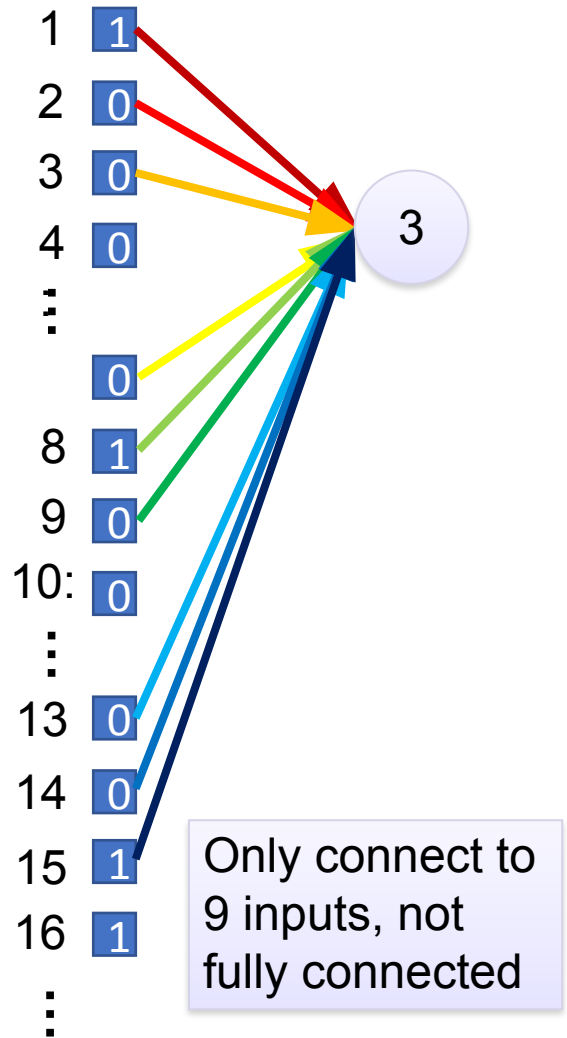
Fully-connected

$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$

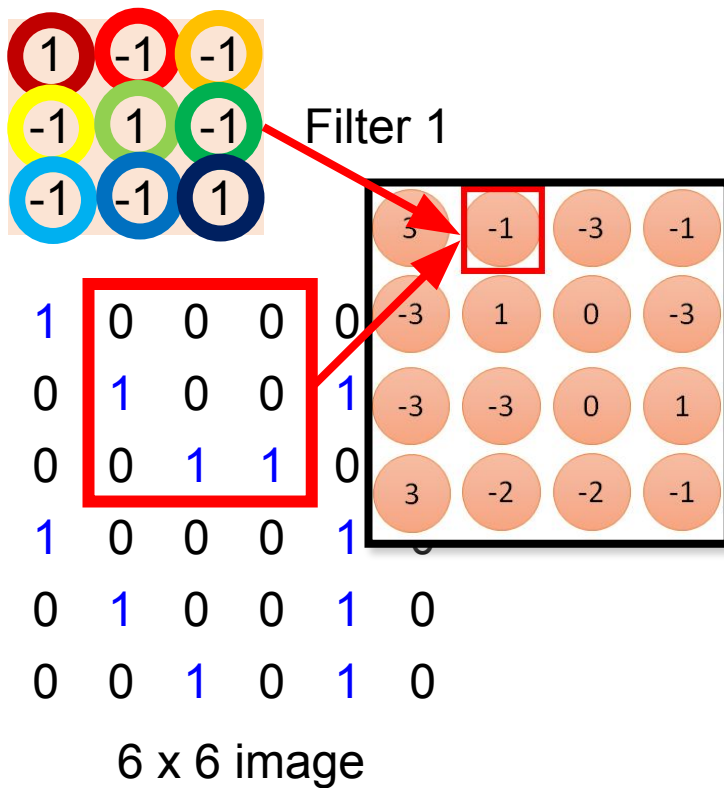




fewer parameters!

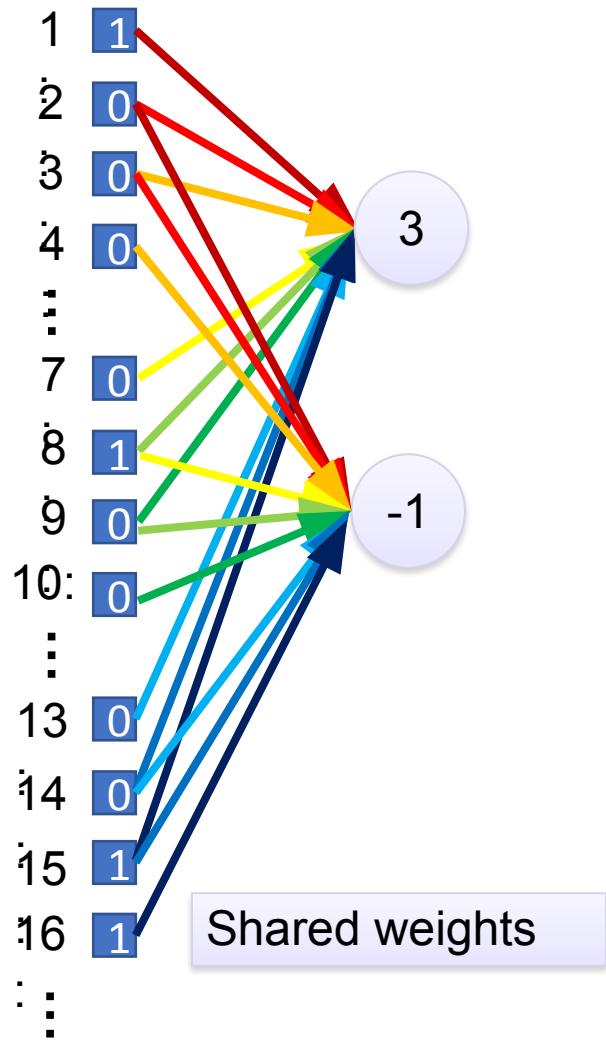


Only connect to 9 inputs, not fully connected

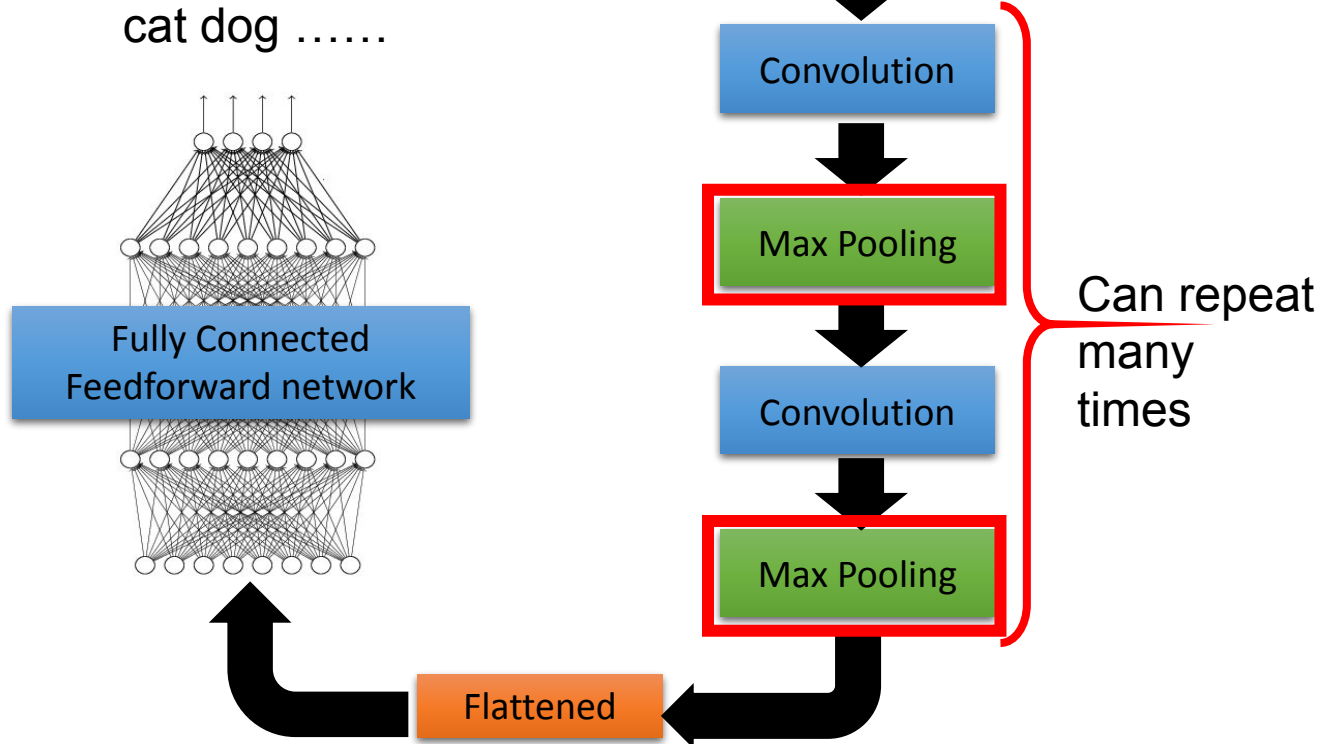


Fewer parameters

Even fewer parameters



The whole CNN



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

Why Pooling

- Subsampling pixels will not change the object

bird



Subsampling

bird



We can subsample the pixels to make image smaller

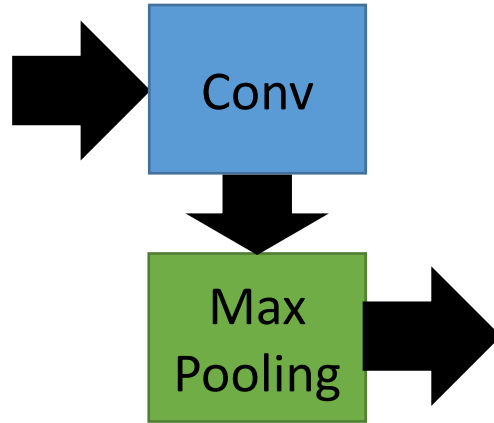


fewer parameters to characterize the image

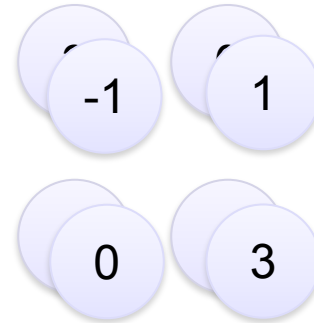
Max Pooling

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

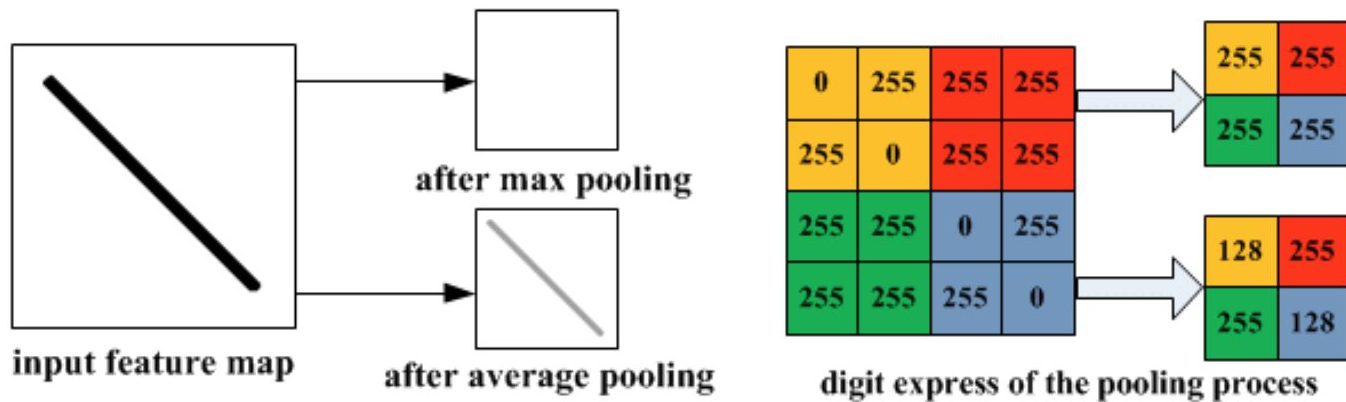


New image
but smaller

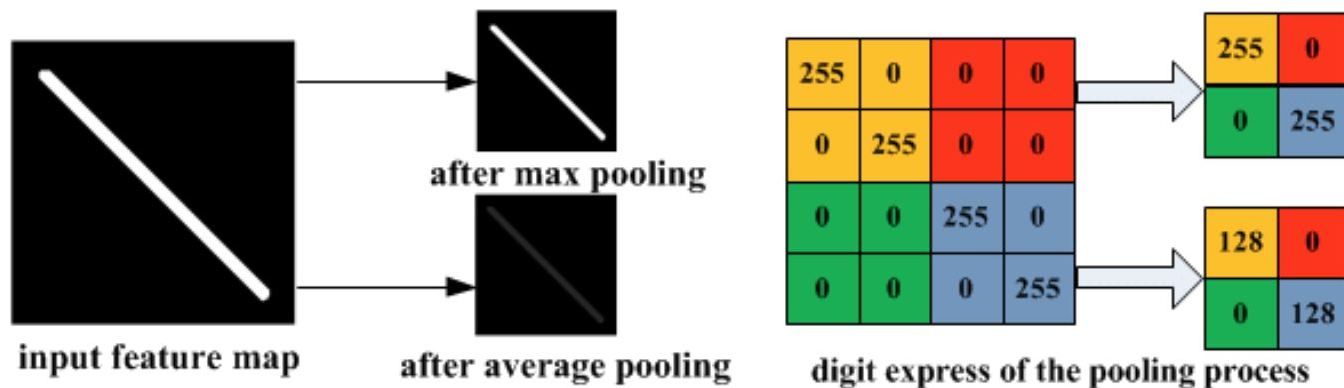


2 x 2 image

Each filter
is a channel

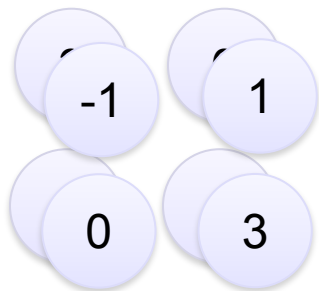


(a) Illustration of max pooling drawback



(b) Illustration of average pooling drawback

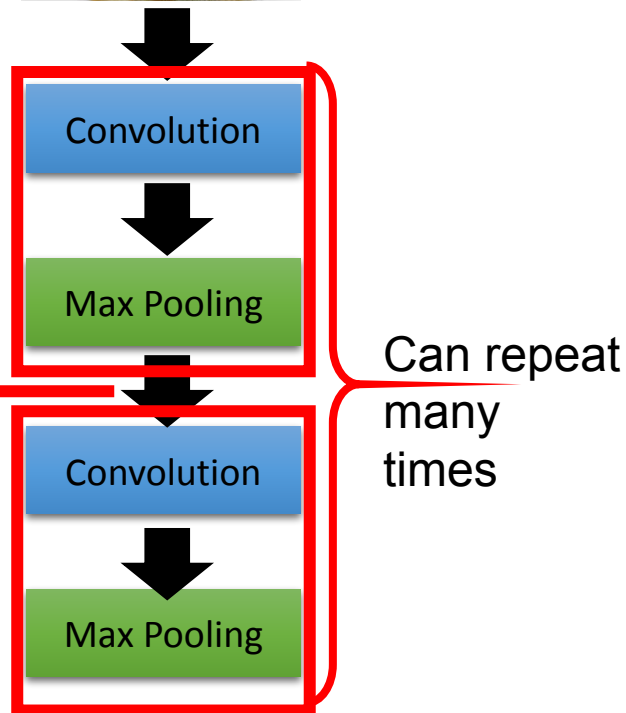
The whole CNN



A new image

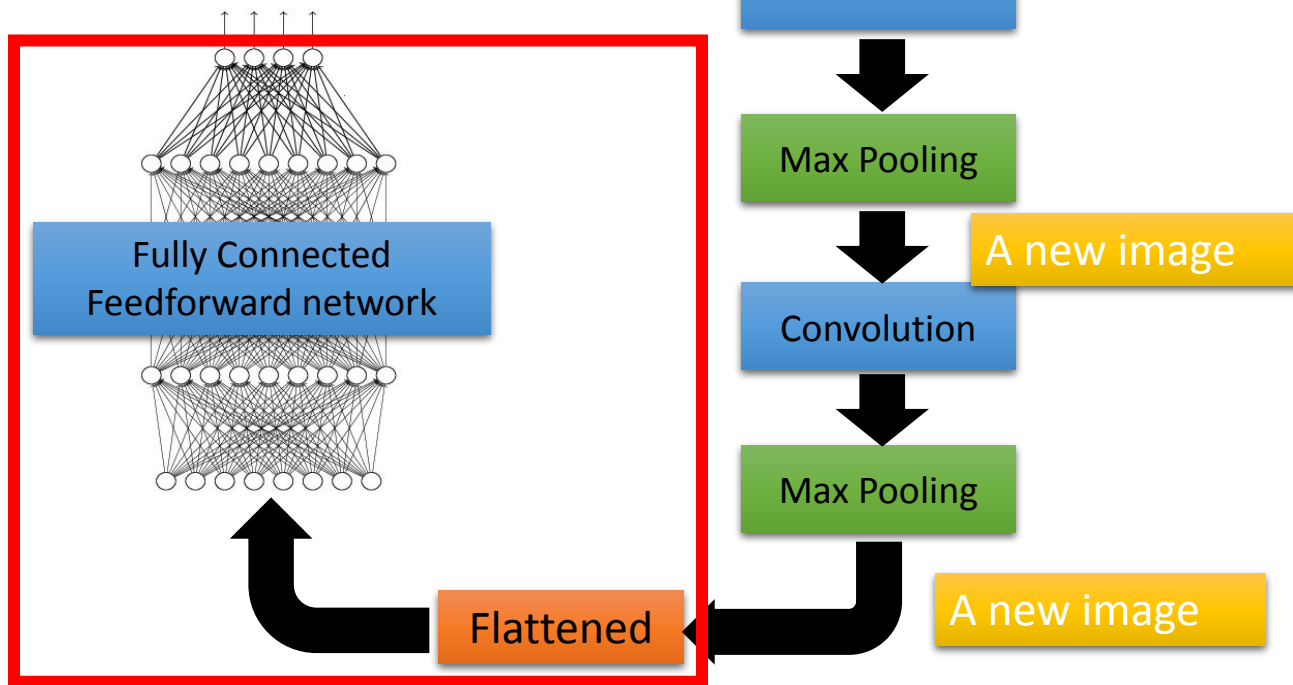
Smaller than the original image

The number of channels is the number of filters

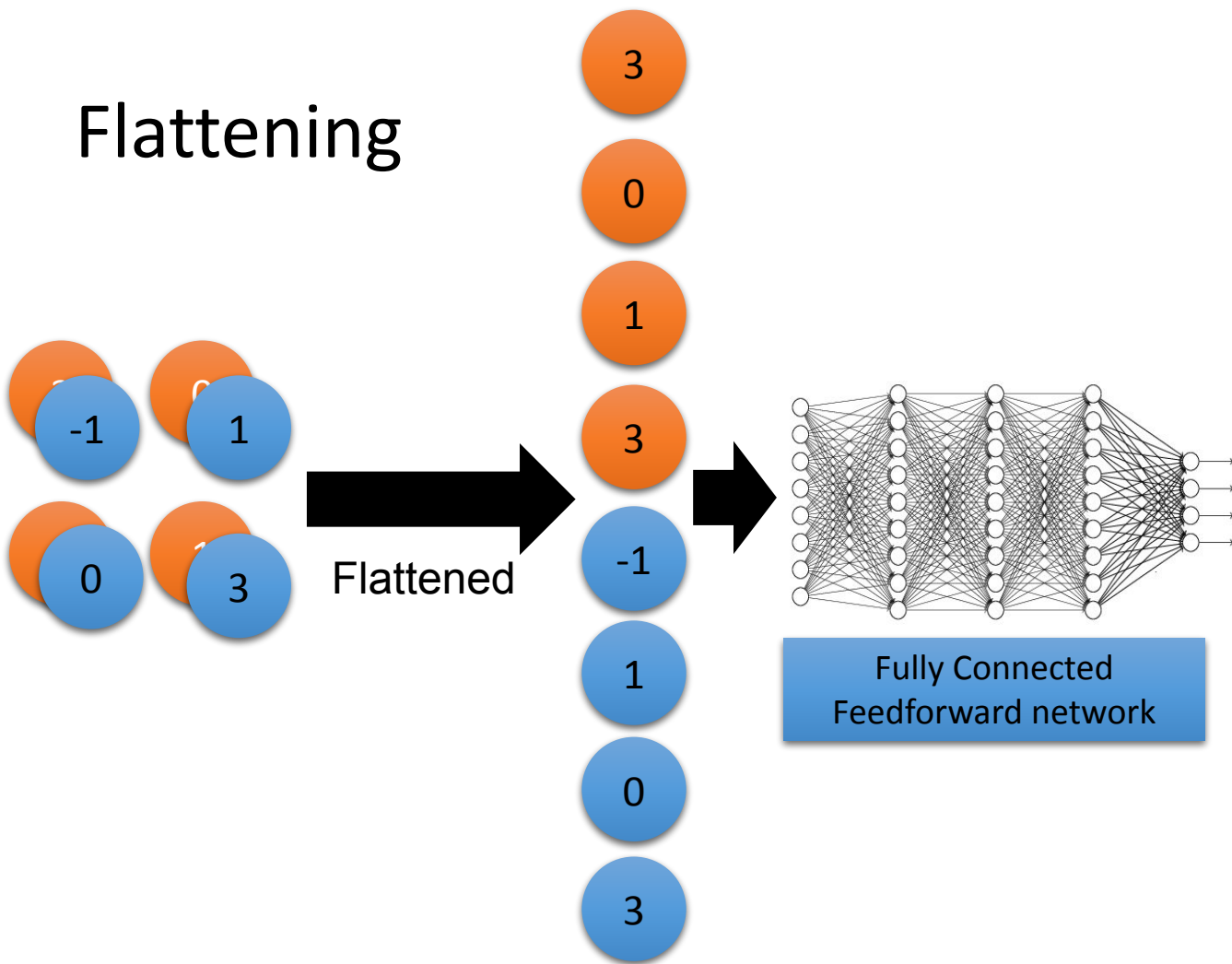


The whole CNN

cat dog



Flattening



Lab 17

CNN

Lab 17

Part 1

Before you start this lab, review the example of CNNs with **FashionMNIST** data

17_Example_FashionMNIST.ipynb

Part 2

Implement a CNN and train with **MNIST** (train data) in Colab.

1 Measure an accuracy with test data.

Submit your code (.ipynb or .py) and captured accuracy in blackboard.

Requirement: greater than 99% **test** accuracy