Introduction to TensorFlow
Shah Bhatti

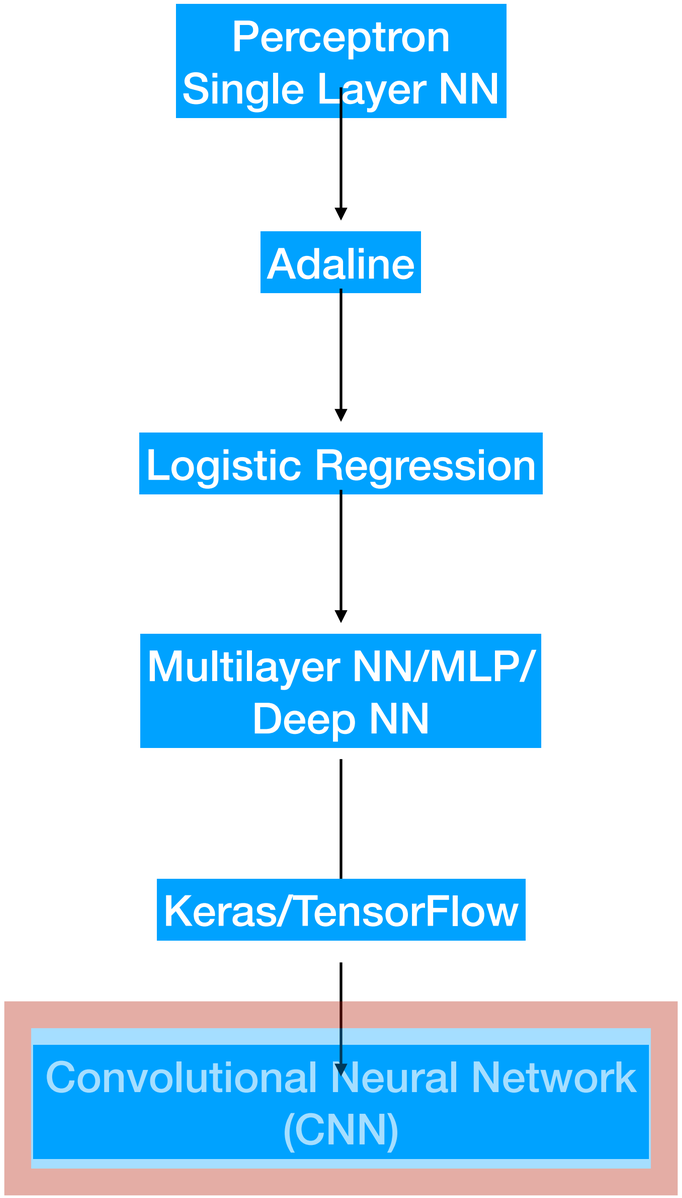
Credit: Slides adapted from Cheriton School of computer Science

What will we learn in Lecture 12?

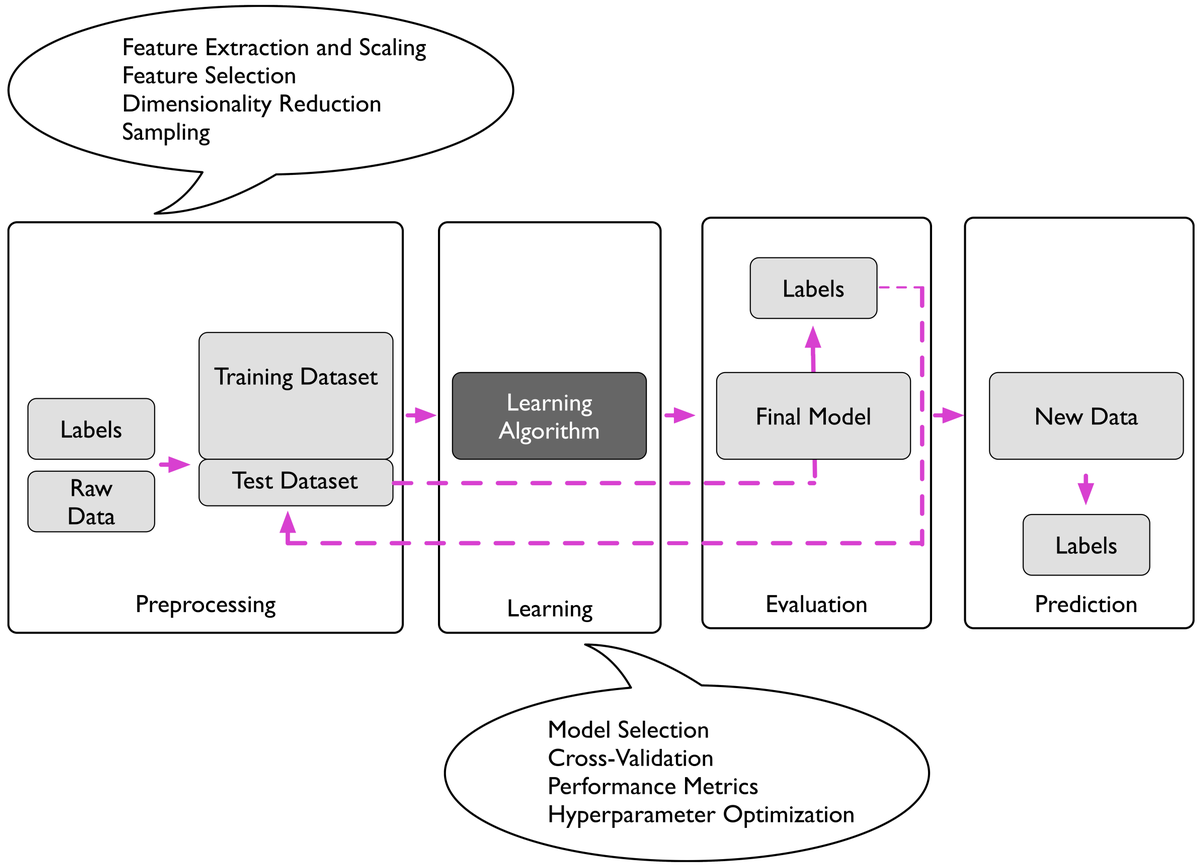
In Lecture 13, we will learn about:

1. The big ideas and concepts behind Convolutional Neural Networks (CNN) using “Brain Learning” (BL = Intuition)
2. How to build a CNN with TensorFlow Keras API (tf.keras) p. 517

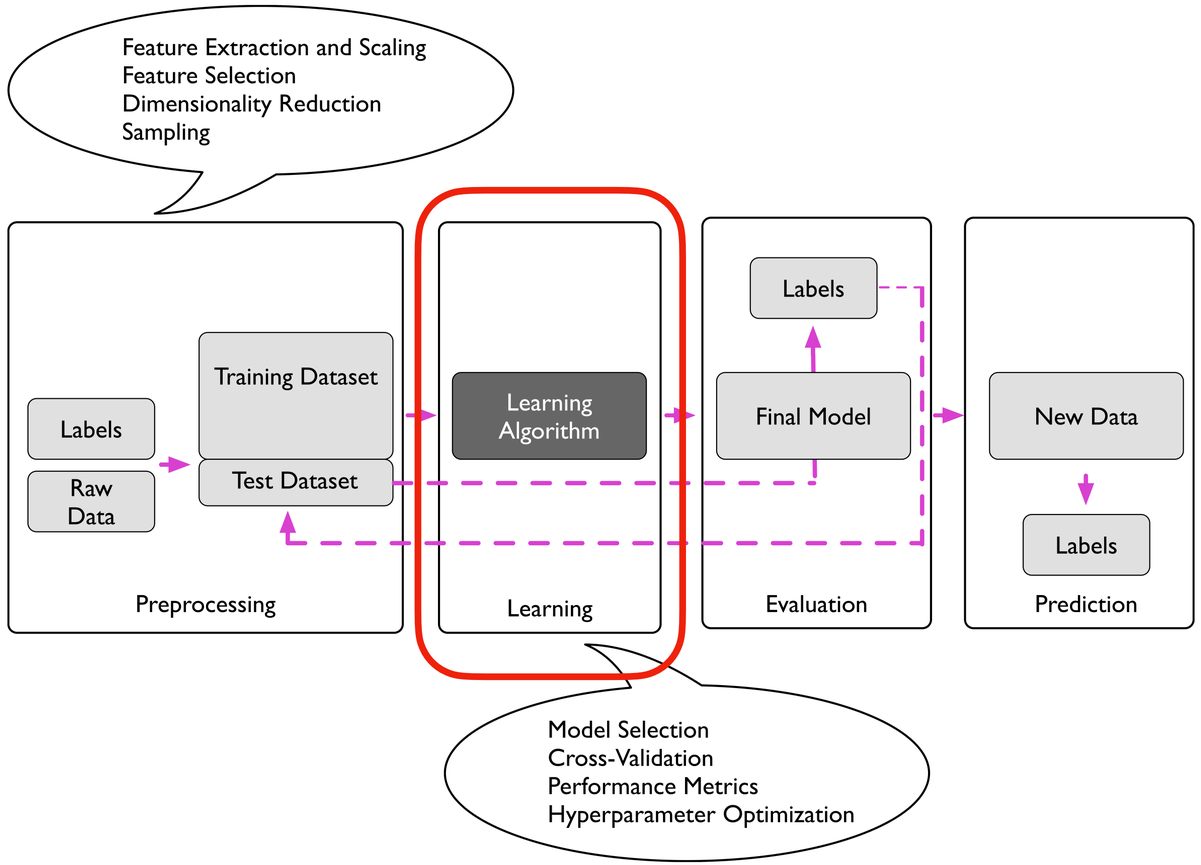
Where are we headed?



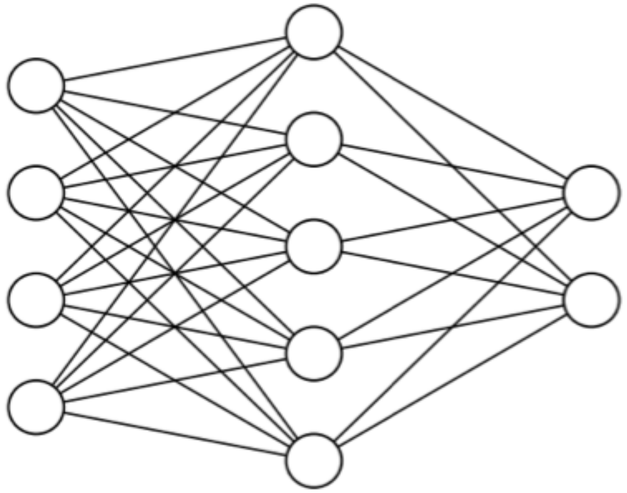
A roadmap for building machine learning systems



A roadmap for building machine learning systems

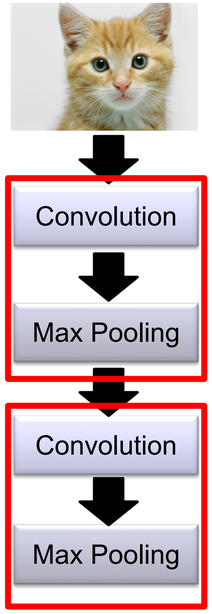


Context:
Multilayer NN to Convolutional NN



cat

dog



The whole CNN

Convolution

Max Pooling

A new image

Convolution

Max Pooling

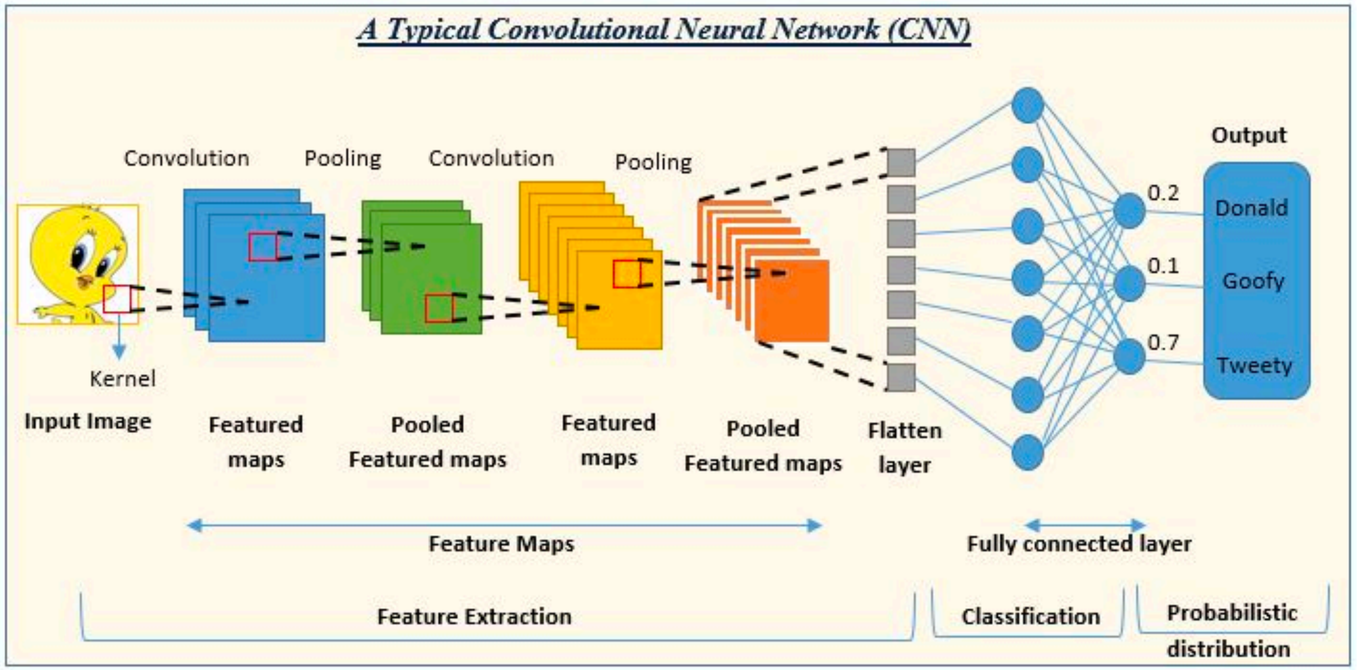
A new image

Flattened

Fully Connected Feedforward network

cat dog ……

Anohter view of the whole CNN



What does a CNN give us?

Three BIG ideas:

1. Reduces the number of connections
2. Shared weights on the edges
3. Max pooling further reduces the complexity

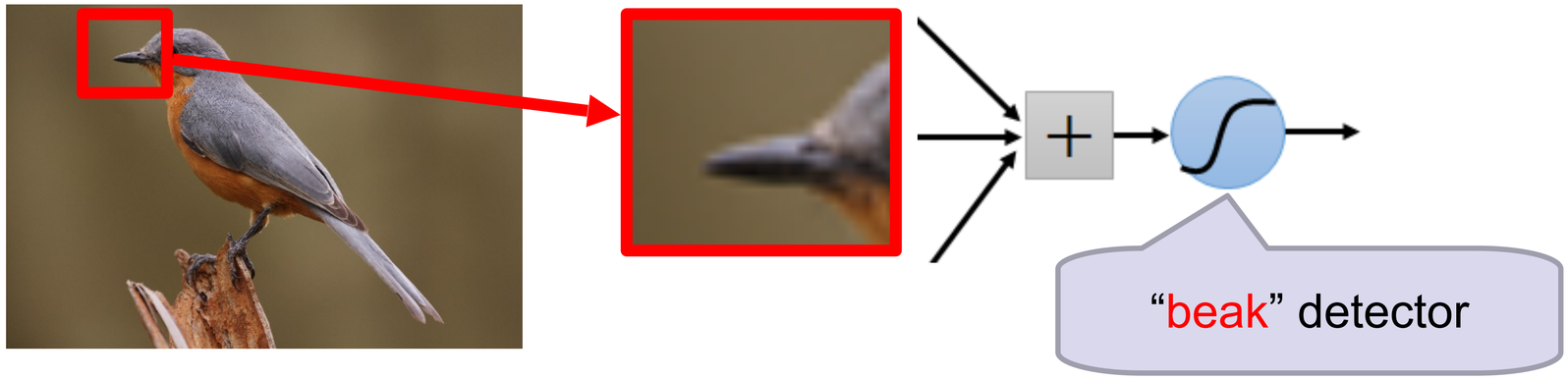
Because we know more about images:

- Spatial locality
- Translation invariance

Intuition: Consider learning an image:

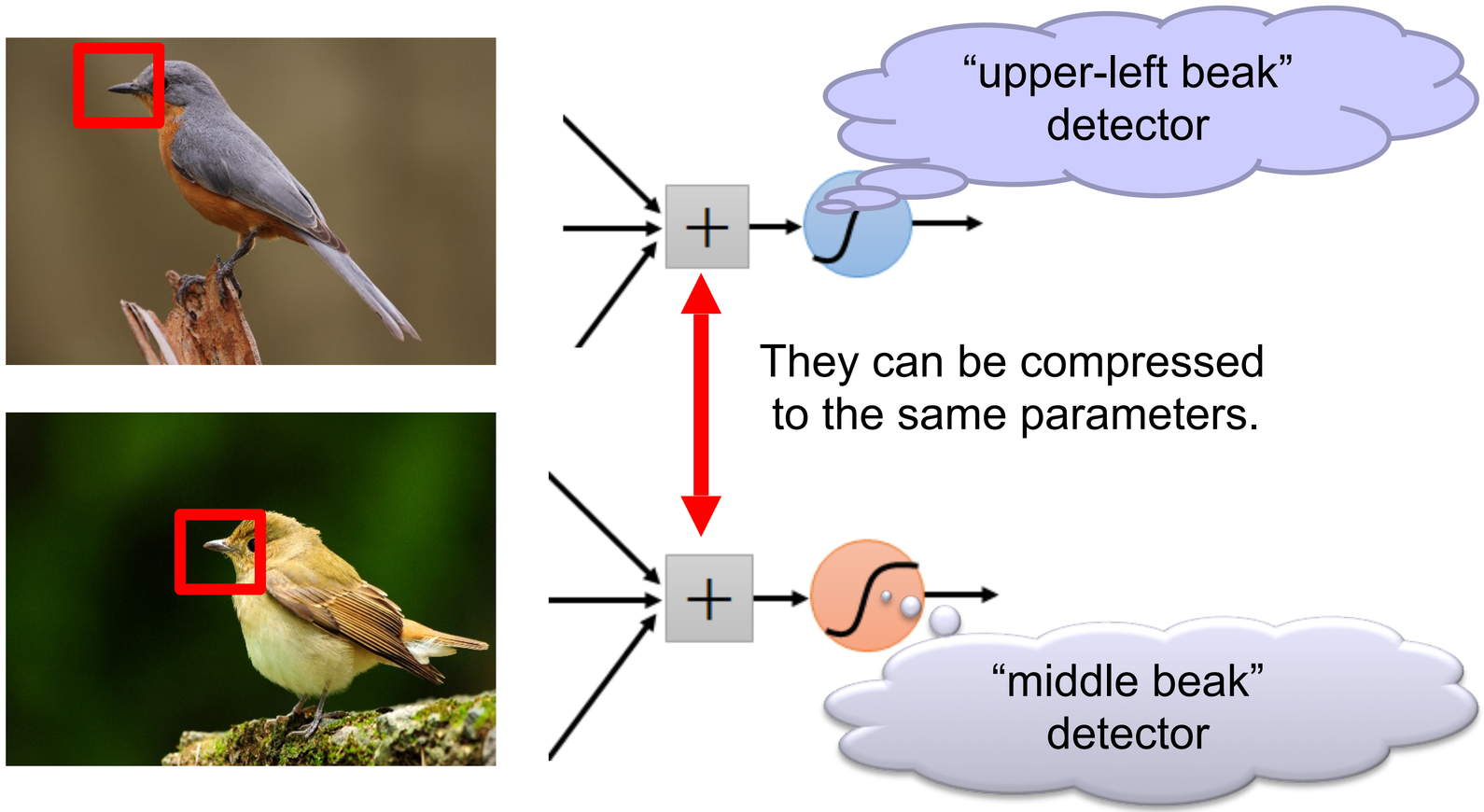
Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters



Intuition: Consider learning an image:

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”.



How do filters learn?

In a convolutional neural network (CNN), each filter (or kernel) learns to detect different features in the input data by:

1. Focusing on a Local Receptive Fields
2. Parameter Sharing
3. Hierarchical Feature Learning
4. Backpropagation and Gradient Descent
5. The Diversity of Filters

1. Focusing on a Local Receptive Fields

Each filter in a CNN has a limited spatial extent, meaning it only looks at a small region of the input image at a time. This **spatially local** receptive field allows the filter to focus on specific patterns or features, such as edges, textures, or shapes, present in that region.

- **Receptive Field** - the region of the input image that a particular neuron in a convolutional layer is “looking at” or taking into account when making its predictions or feature extraction

2. Parameter Sharing

Filters are applied across the entire input image, which means the same filter is used multiple times at different spatial locations. This parameter sharing enables the network to learn **translation-invarian**t features; that is, the same feature can be recognized regardless of where it appears in the image.

3. Hierarchical Feature Learning

As the CNN layers progress, each subsequent layer combines features learned by previous layers. Early filters may learn simple features like edges or corners, while deeper layers can combine these features to detect more complex structures, such as shapes or objects. This hierarchy allows the network to learn a rich representation of the input data.

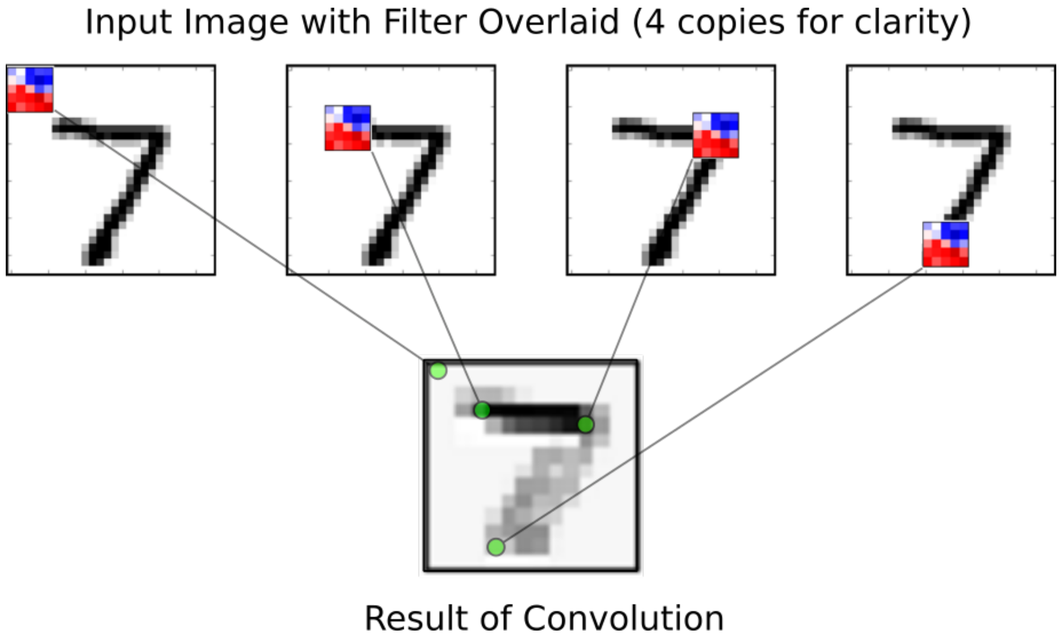
4. Backpropagation and Gradient Descent

Filters are initialized with random weights and adjusted through training using backpropagation and gradient descent. During this process, filters that respond strongly to certain features will have their weights reinforced, while those that do not contribute significantly to the loss will be adjusted. This selective learning process leads to each filter becoming specialized in detecting specific features.

5. The Diversity of Filters

Different filters are initialized and trained independently, allowing them to capture various aspects of the input data. The diversity in learning arises from the randomness in weight initialization and the distinct gradients during training, leading to a specialization in recognizing different features.

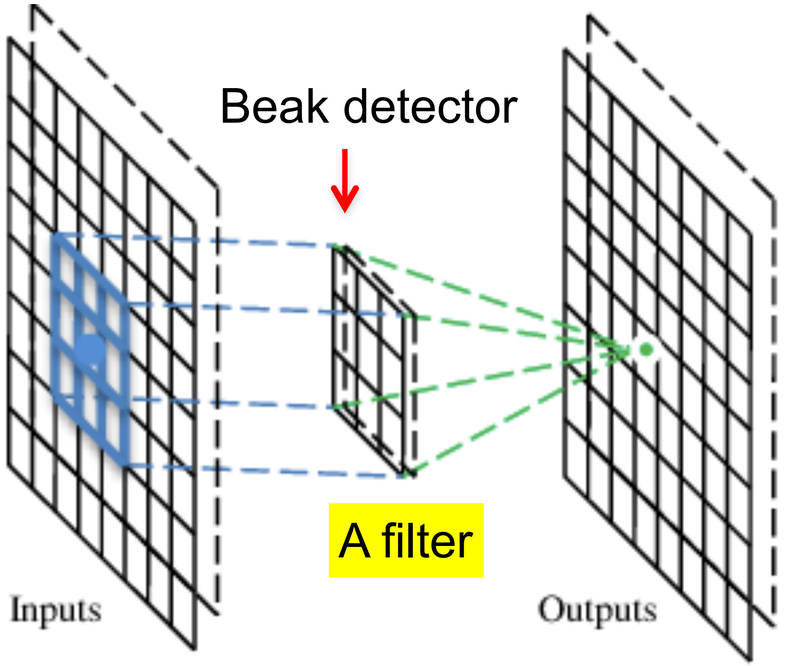
How do filters learn?



The red filter-weights means that the filter has a positive reaction to black pixels in the input image, while blue pixels means the filter has a negative reaction to black pixels. In this case it appears that the filter recognizes the horizontal line of the 7-digit, as can be seen from its stronger reaction to that line in the output image.

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

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

3 -1

Convolution

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

Filter 1

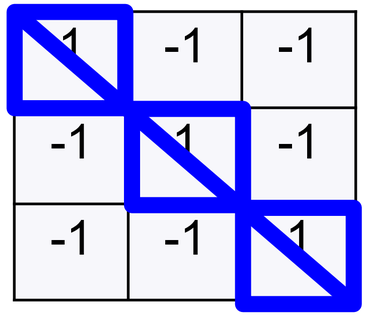
If stride=2

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

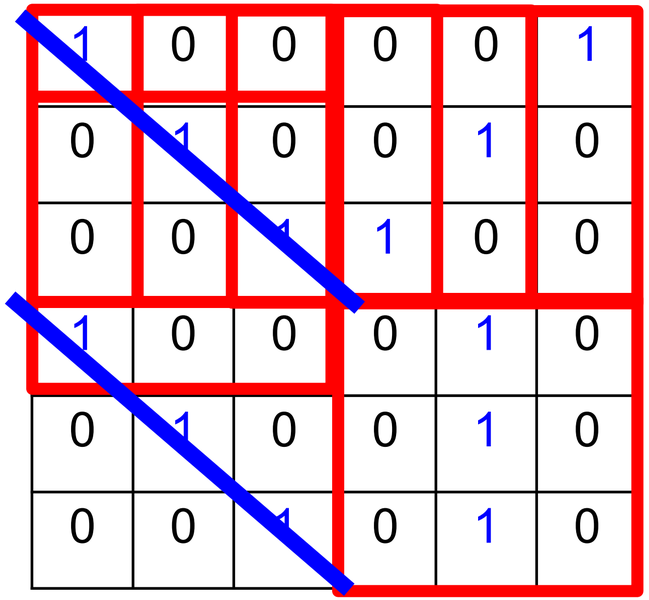
3 -3

Convolution

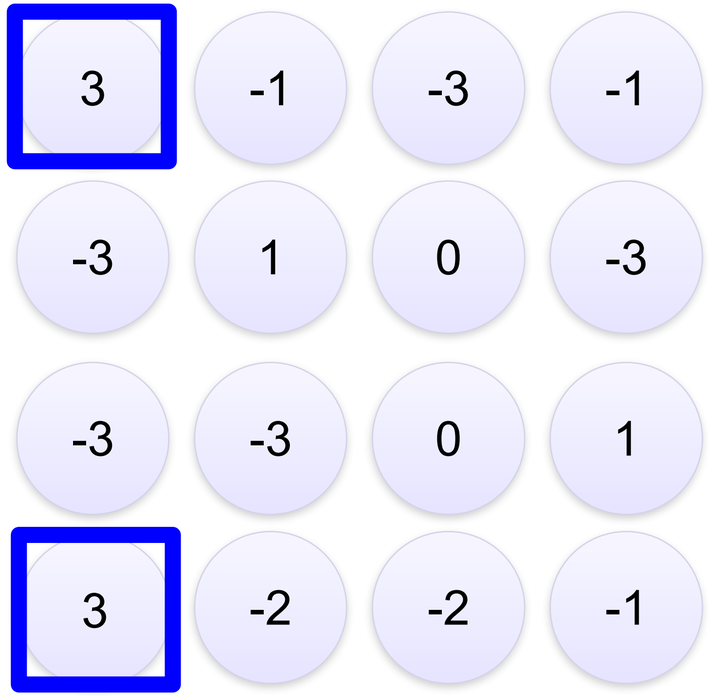


Filter 1

stride=1



6 x 6 image



Convolution

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

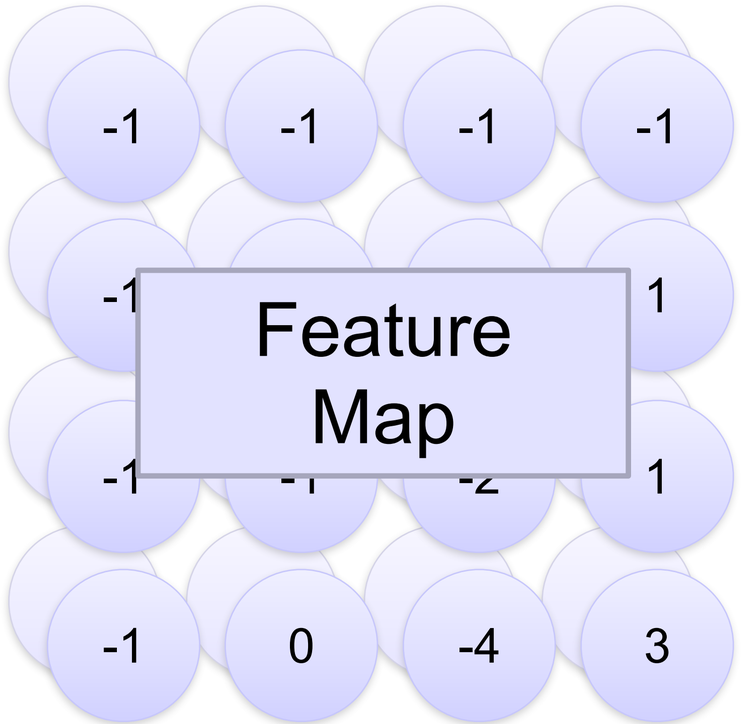
Filter 2

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

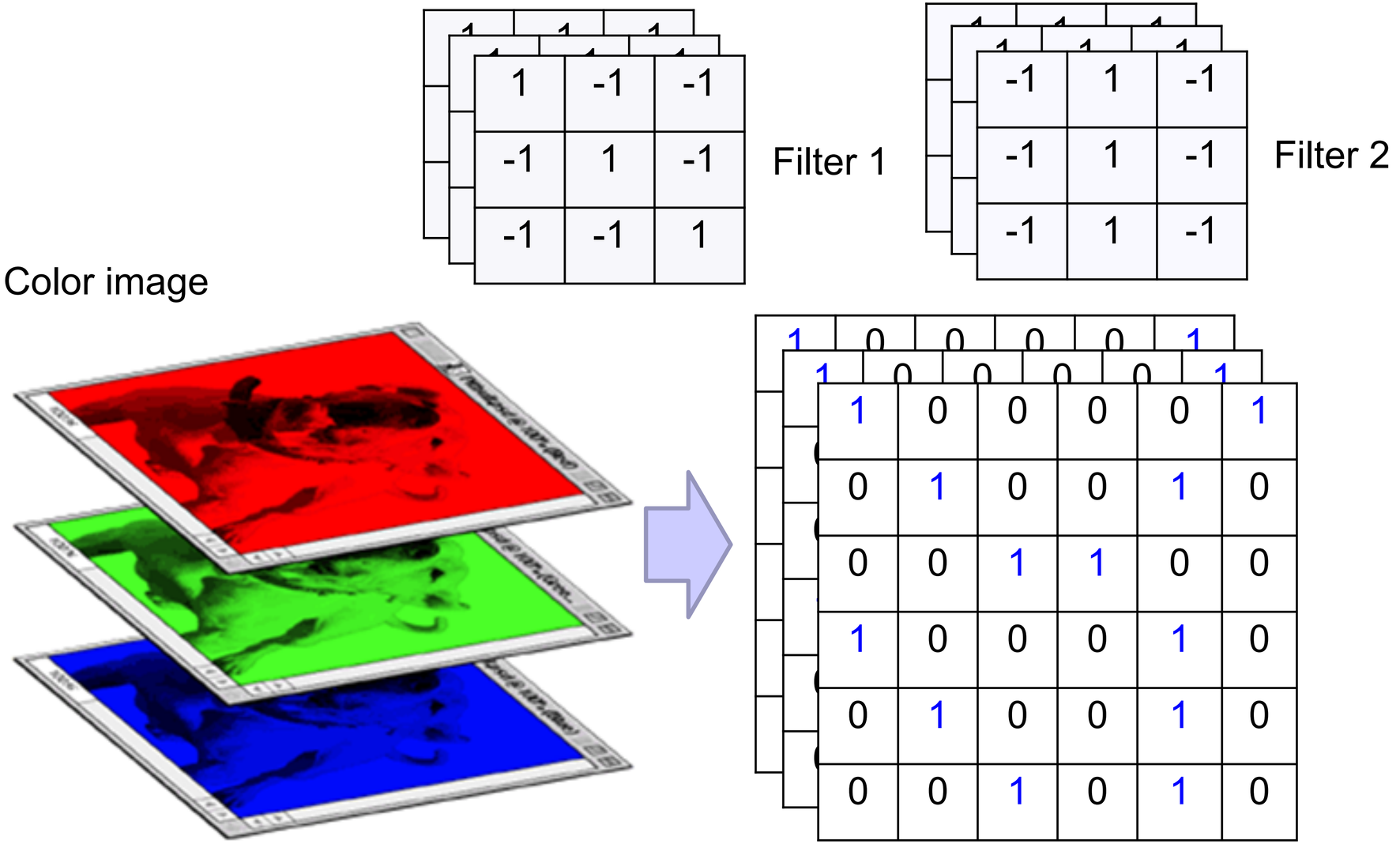
6 x 6 image

Repeat this for each filter

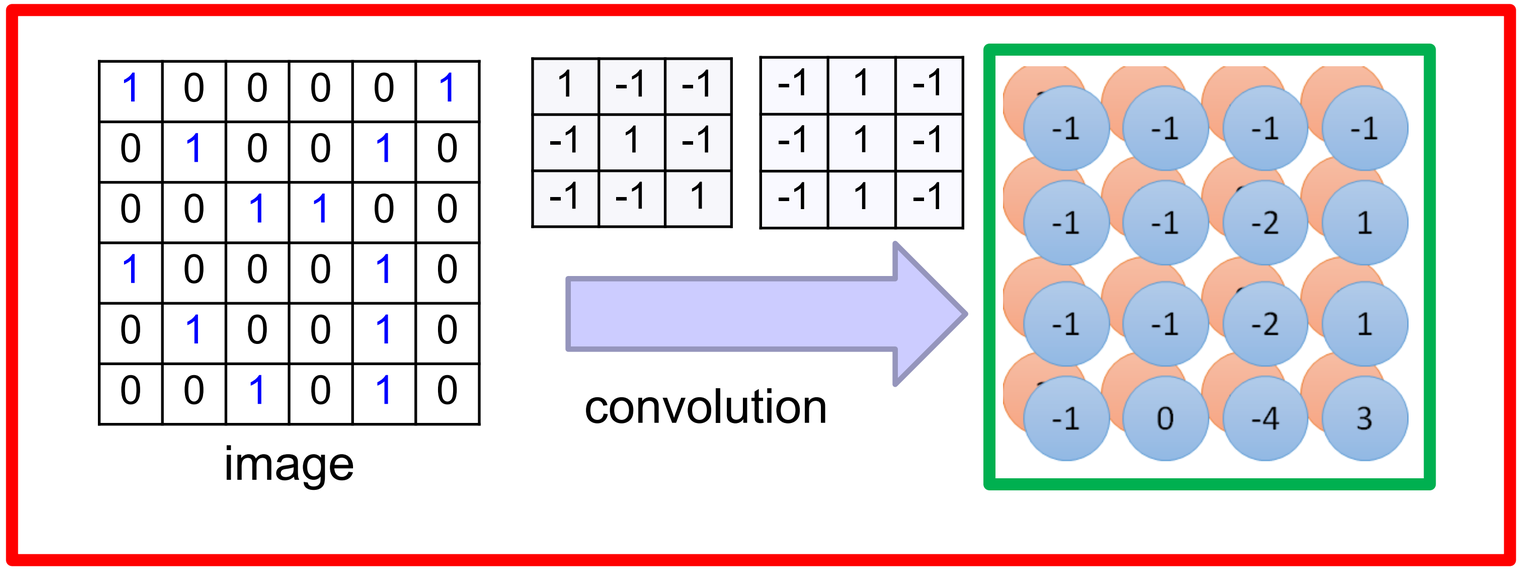


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

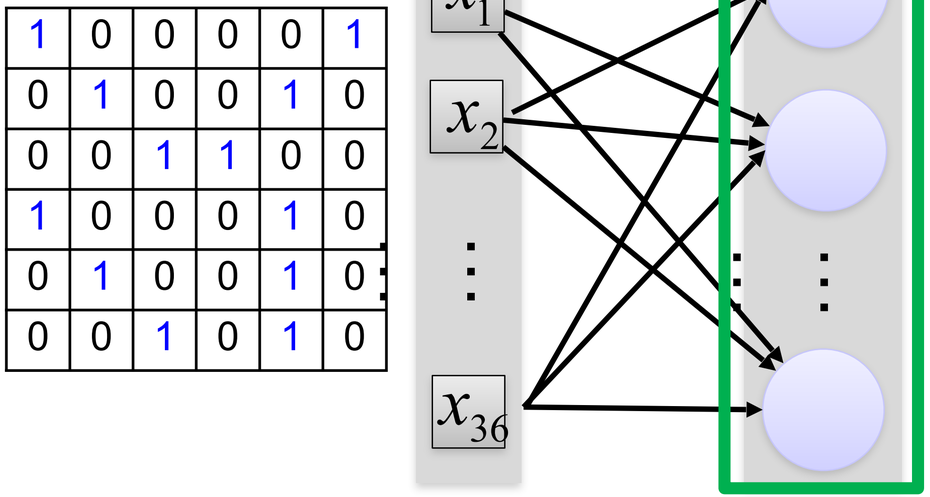
Color image: RGB 3 channels

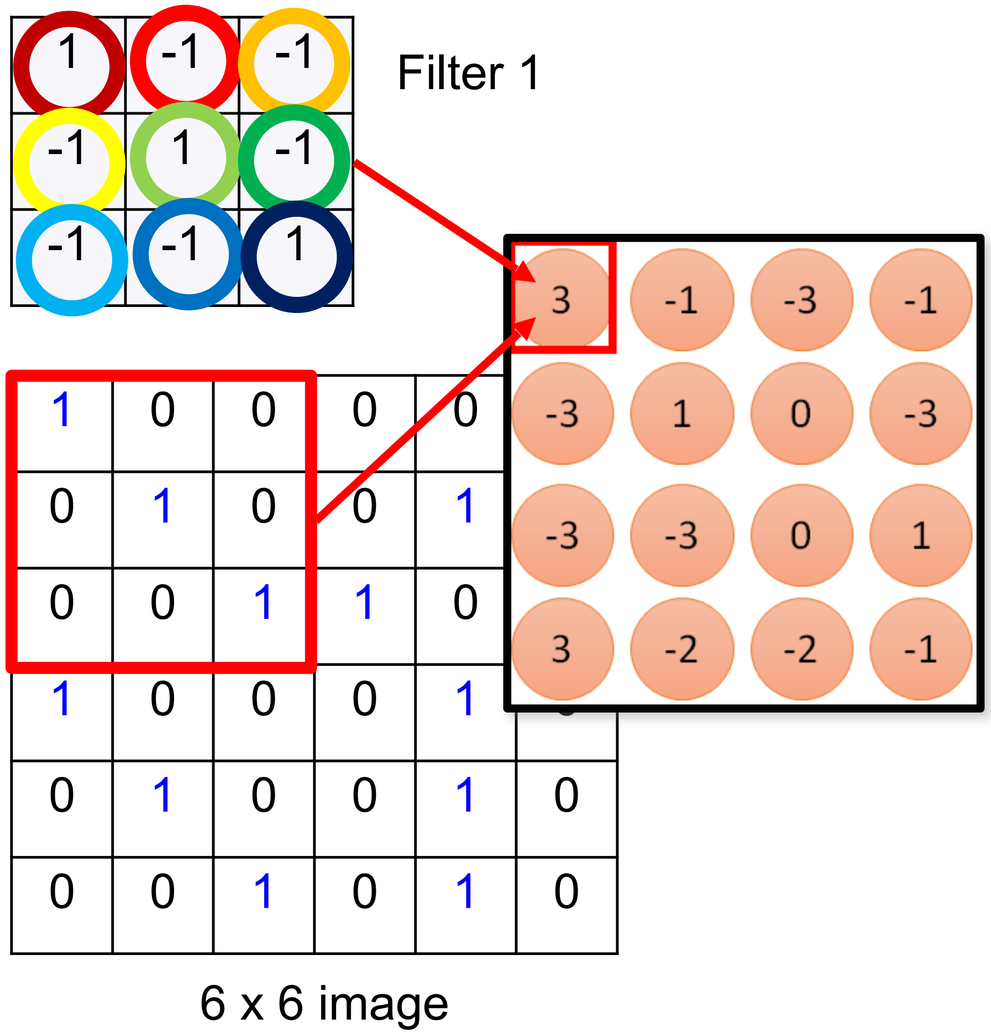


Convolution v.s. Fully Connected

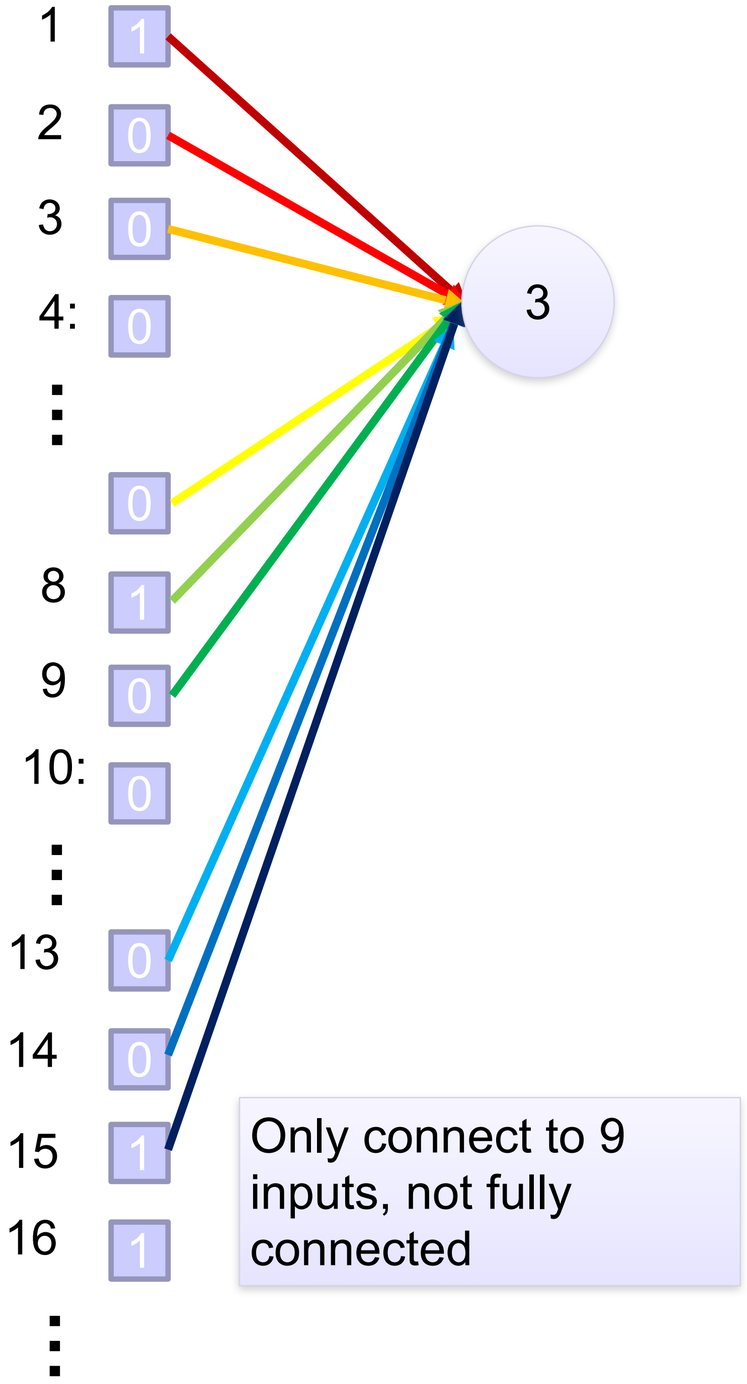


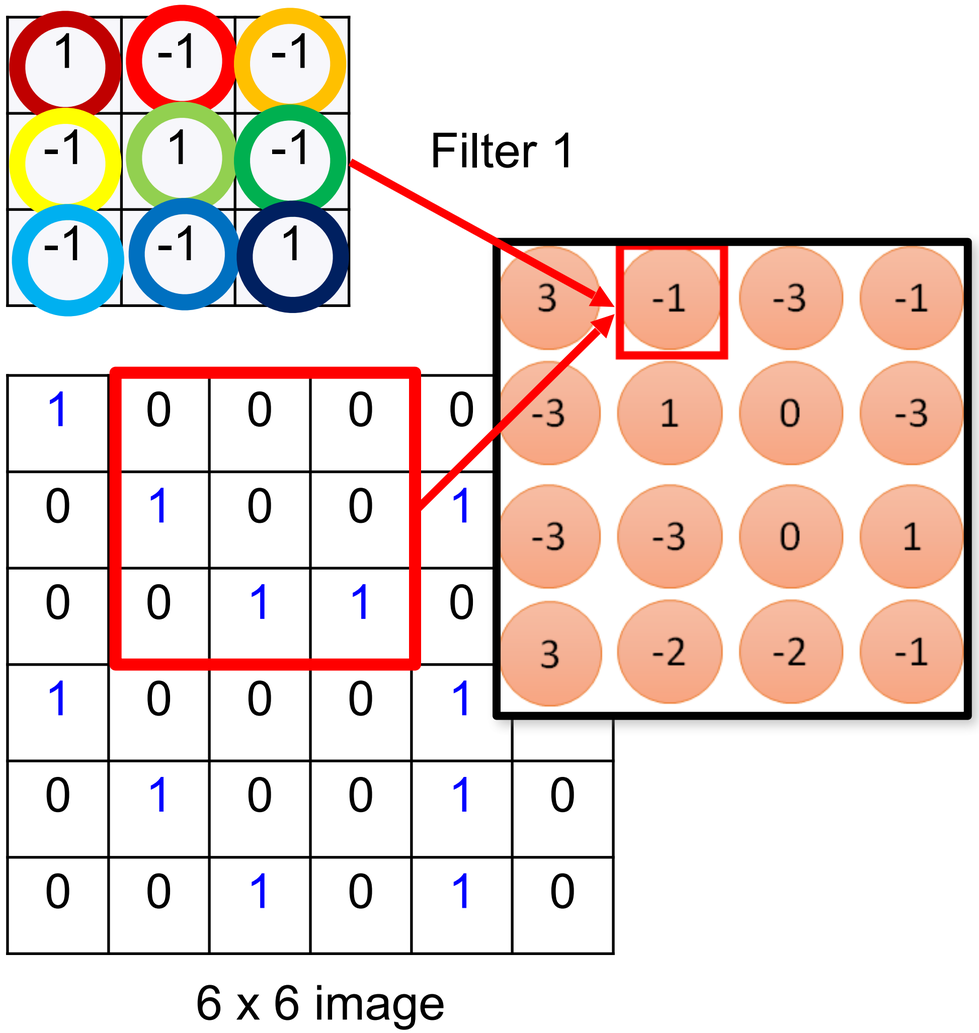
Fully-connected





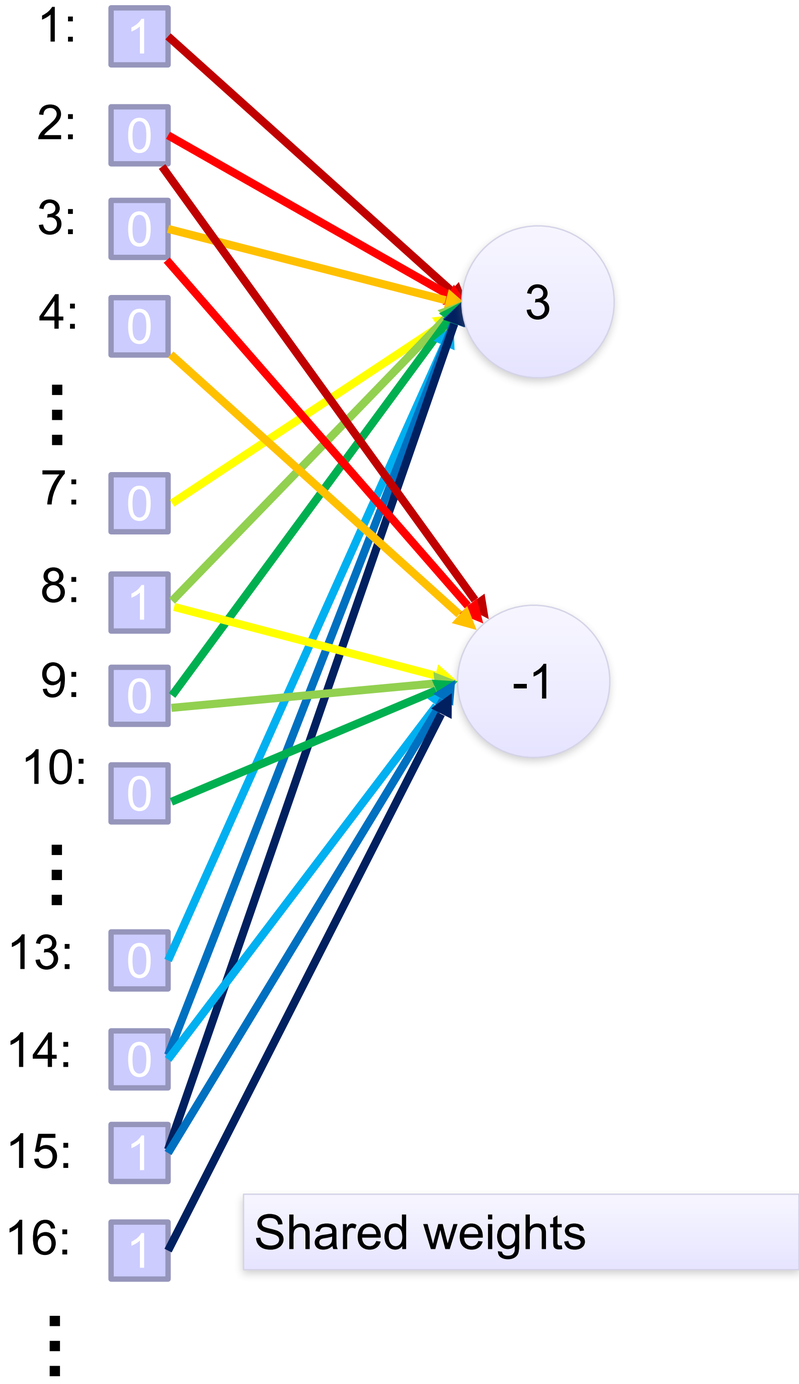
fewer parameters!



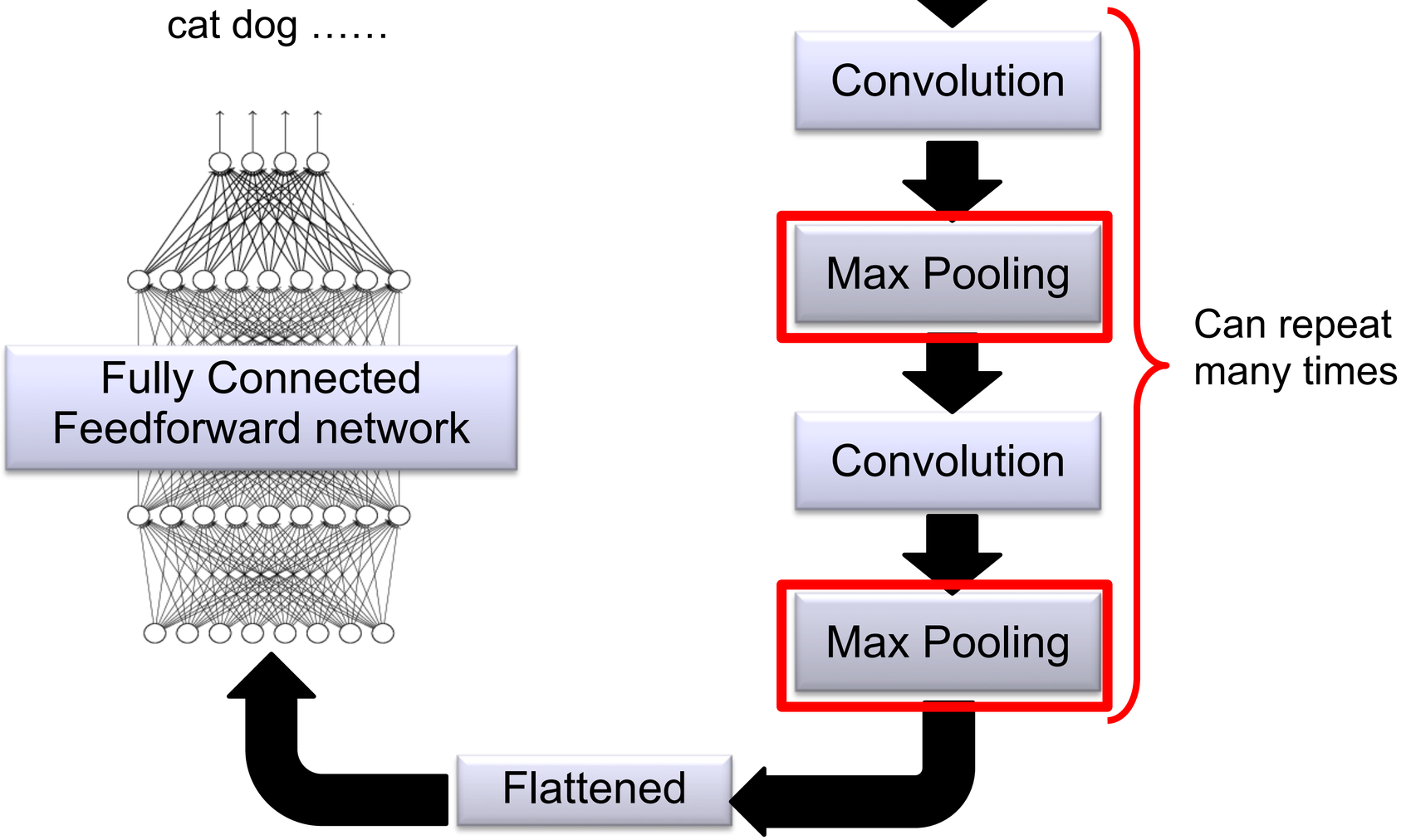


Fewer parameters

Even fewer parameters



The whole CNN



Why Pooling

bird

Subsampling

bird

We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

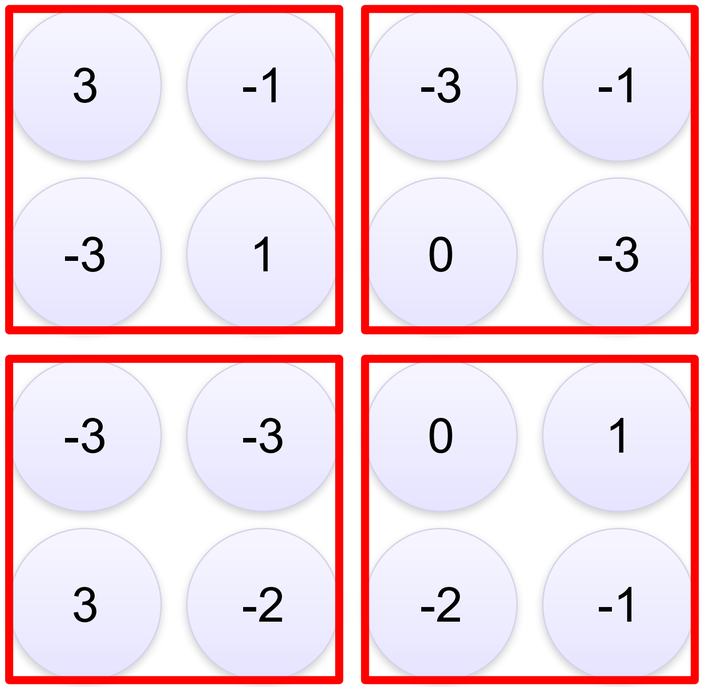
Max Pooling

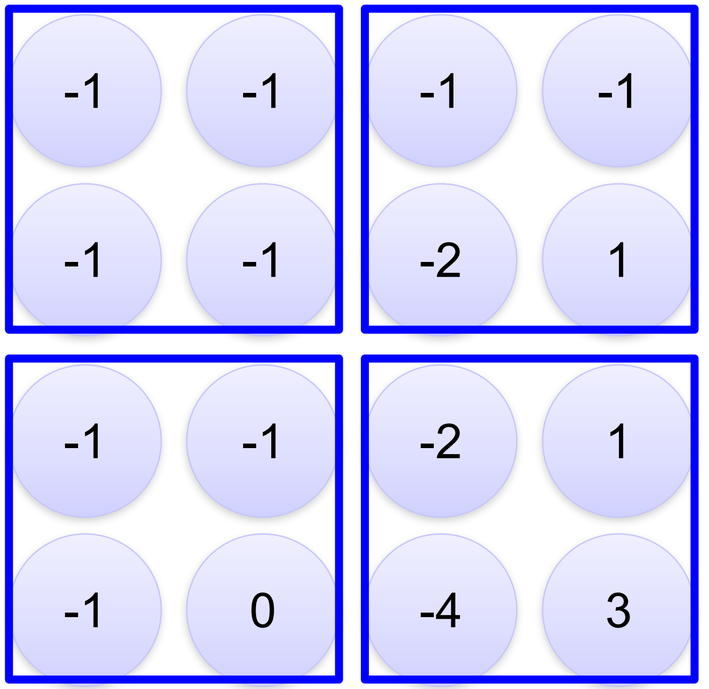
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

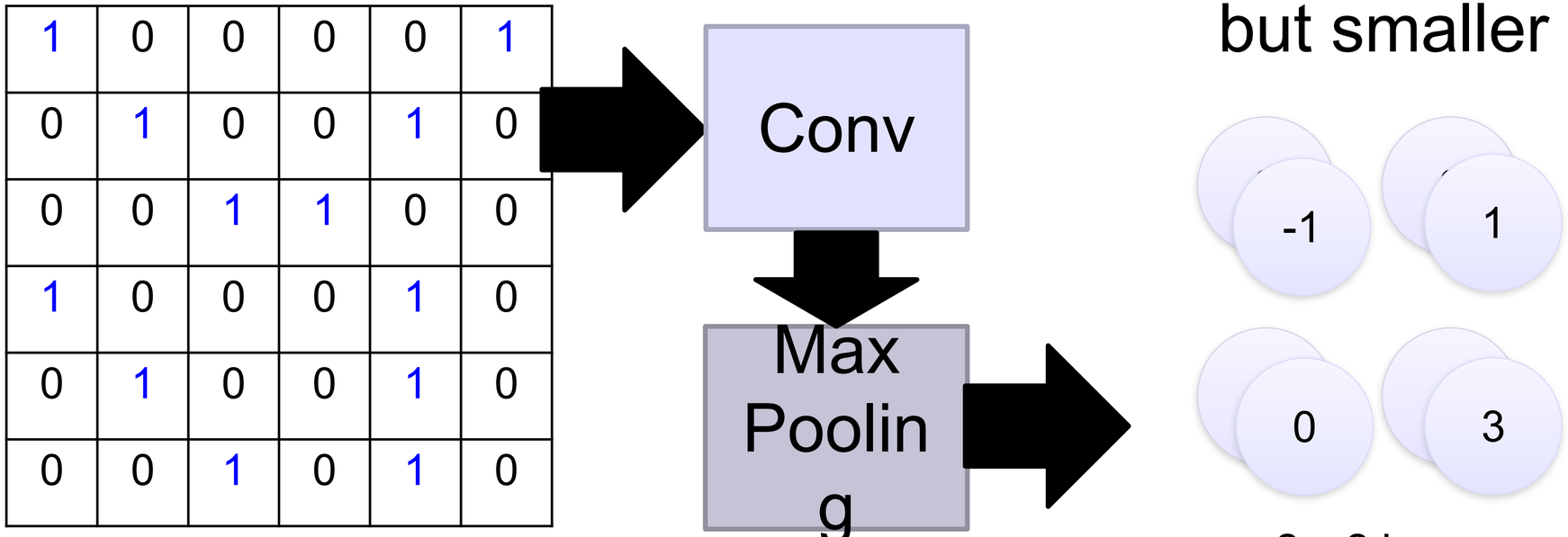
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2



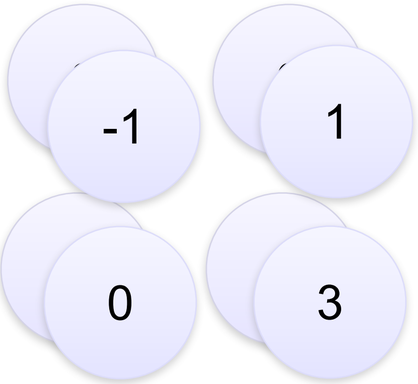


Max Pooling



Each filter is a channel

The whole CNN



A new image

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Smaller than the original image

The number of channels is the number of filters

The whole CNN

Convolution

Max Pooling

A new image

Convolution

Max Pooling

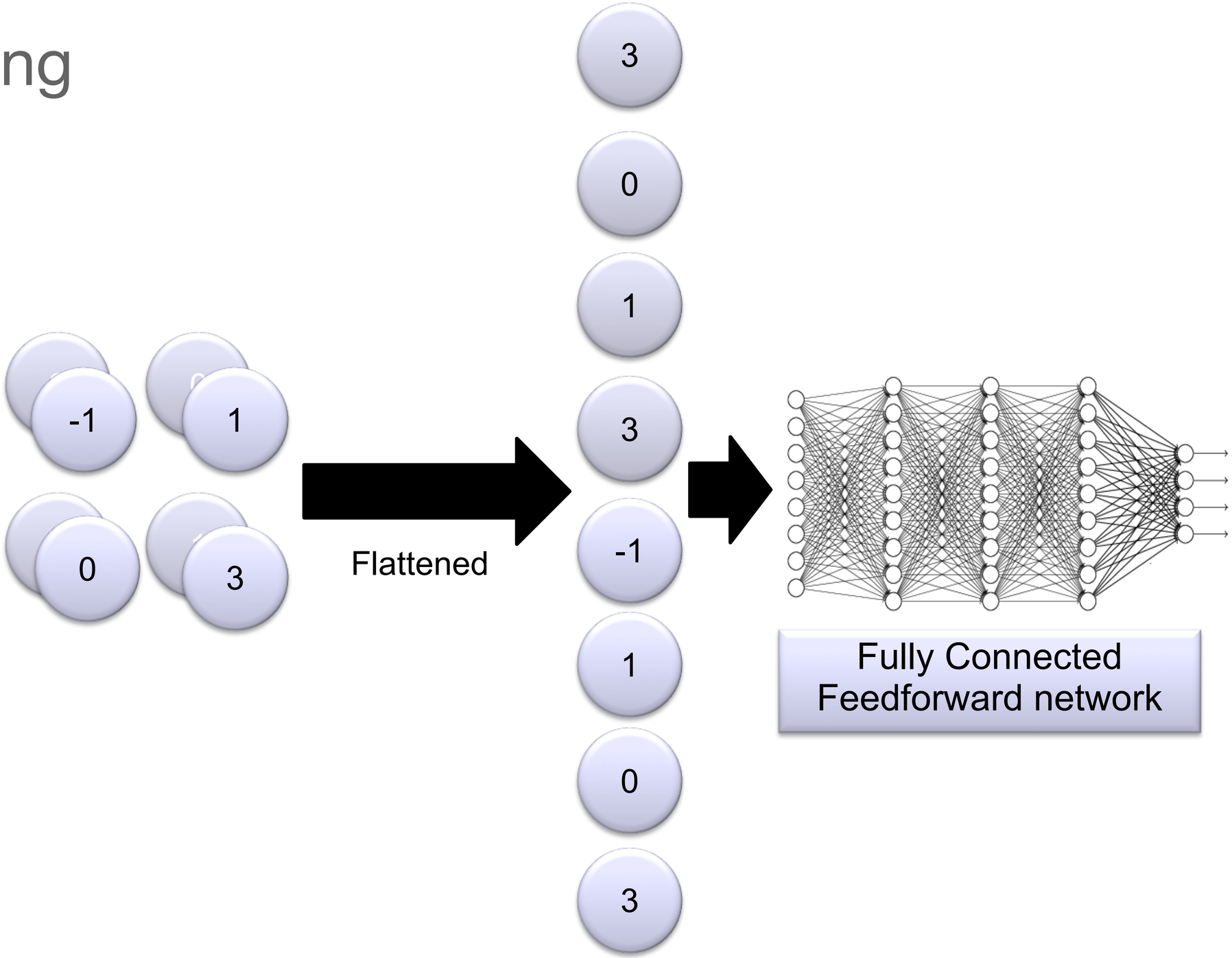
A new image

Flattened

Fully Connected Feedforward network

cat dog ……

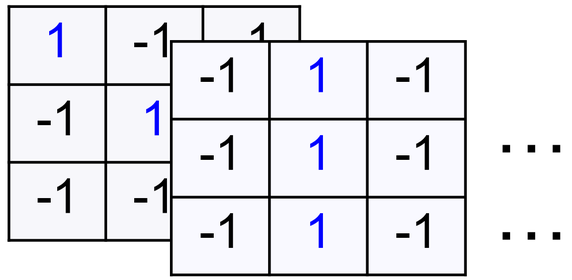
Flattening


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

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
                input_shape=(28,28,1)) )
```



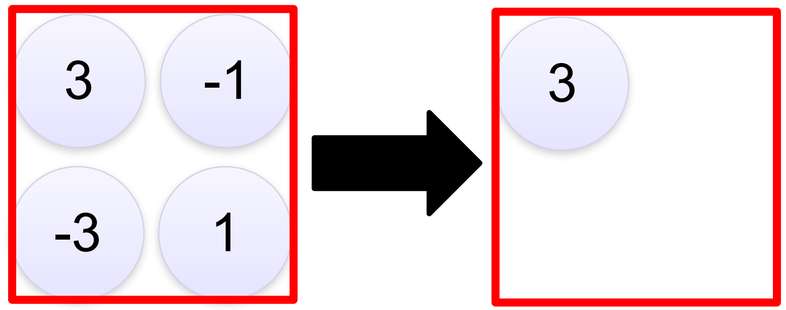
There are 25 3x3 filters.

Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

```
model2.add(MaxPooling2D((2,2)))
```



input

Convolution

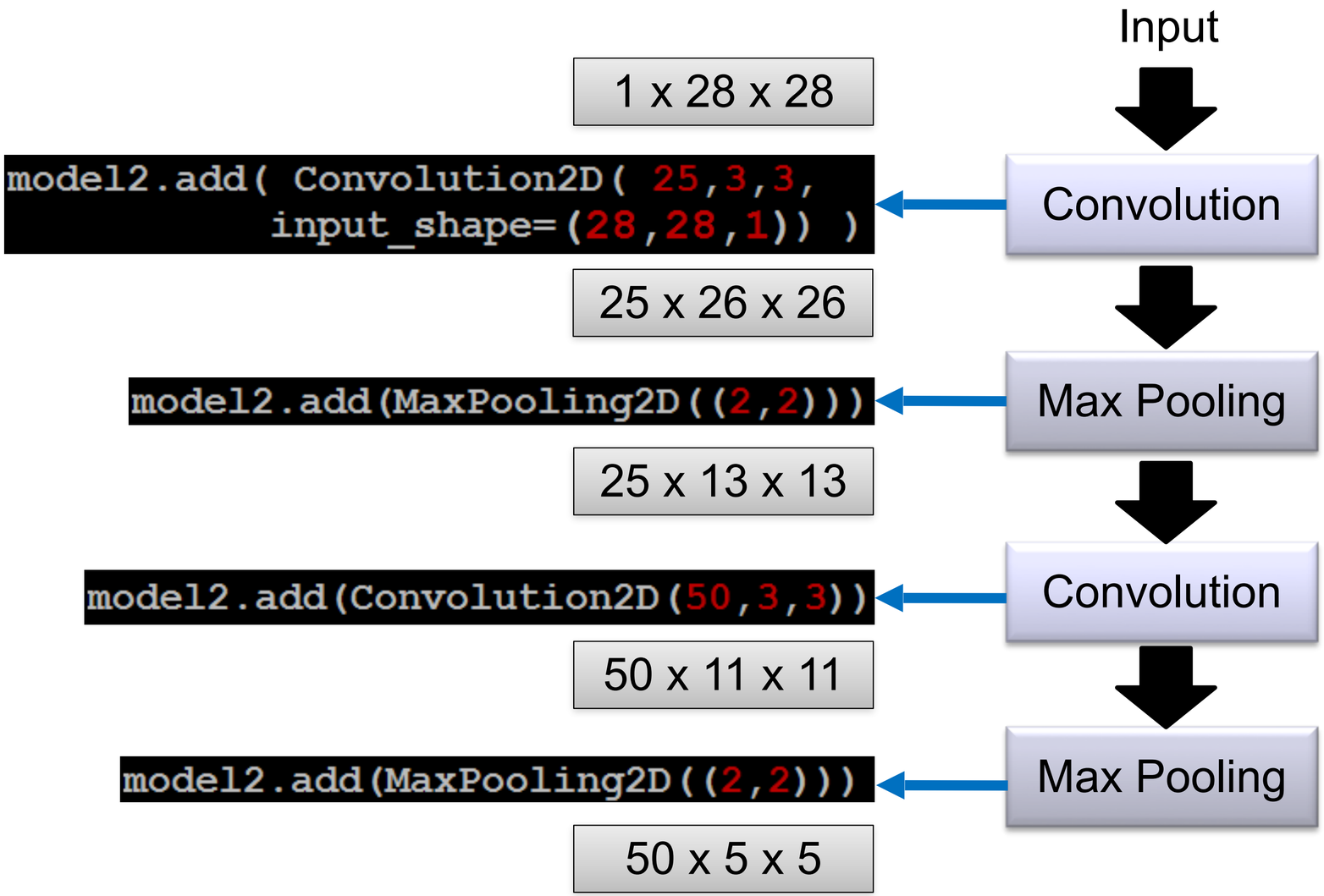
Max Pooling

Convolution

Max Pooling

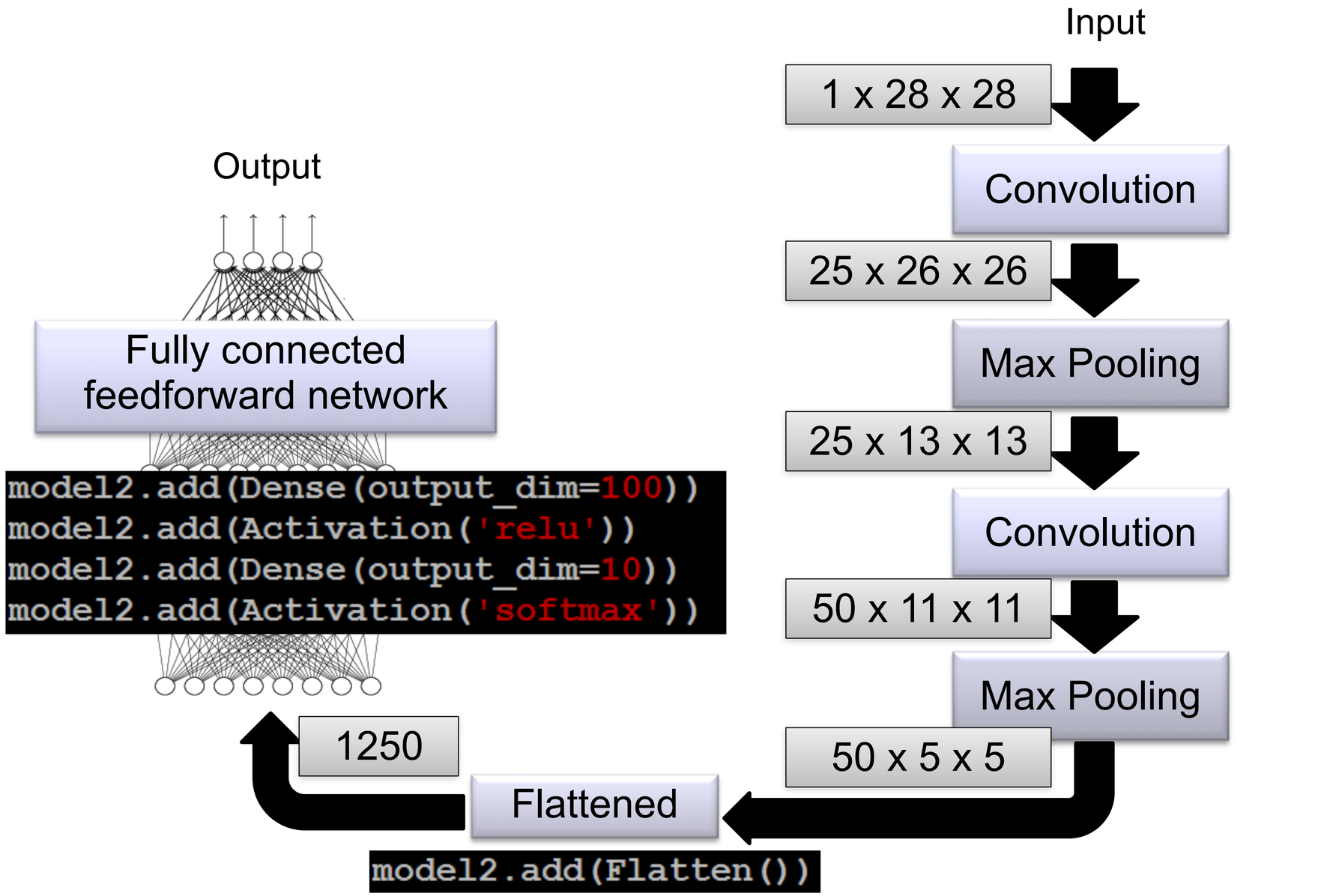
CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



Is there an optimal filter/kernel size?

Short answer: No

Depends on:

1. The Task and the Data
2. The Input size and Complexity
3. The Balance Between Local and Global Knowledge
4. The Information Loss you are willing to accept
5. The Computational Resources available

1. The Task and the Data

- For tasks that require the detection of fine-grained details, smaller kernel sizes are generally preferred. Conversely, larger kernel sizes are appropriate for capturing larger patterns.
- Does the data offer fine detail or larger patterns. For example, a close-up image of a bird with many fine-grained details v. an aerial image taken by a satellite of a farm

2. The Input size and Complexity

- The size of the input data also affects the choice of kernel size. Smaller kernel sizes are generally more suitable for smaller input sizes, while larger input sizes can accommodate larger kernel sizes.
- For simple tasks or datasets with less complexity, smaller kernel sizes may be sufficient, but more complex tasks or datasets may require larger kernel sizes to capture complex features.

3. The Balance Between Local and Global Knowledge

- Kernel size determines the receptive field of each convolutional operation. Smaller kernel sizes capture more local information, while larger kernel sizes capture more global information.
- It’s important to strike a balance between capturing local details and considering the broader context.

4. The Information Loss you are willing to accept

- Choosing an excessively small kernel size can lead to information loss, especially in deeper layers of the network where the receptive field becomes large. In such cases, stacking multiple layers with small kernel sizes can help preserve information.

- Conversely, using excessively large kernel sizes can result in over-smoothing or loss of fine details in the input data.

5. The Computational Resources available

- Larger kernel sizes require more computational resources for processing during training and inference. Therefore, it is important to consider the available resources and computational constraints when choosing the kernel size.

Is there an optimal filter/kernel size?

Short answer: No

Long answer:
Choosing the appropriate kernel size in a CNN requires a careful balance between capturing relevant features, preserving spatial information, and considering computational constraints. By understanding the task requirements, data characteristics, and **experimenting** with different kernel sizes, we can effectively design CNN architectures that achieve optimal performance for the given task.

Is there an optimal filter/kernel size?

If you ask me:

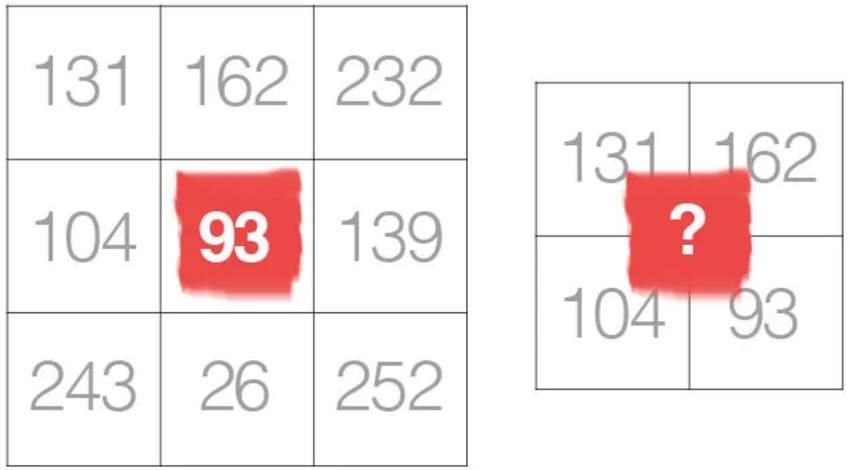
1. Use a 3x3 kernel.
2. Always choose kernels with odd dimensions: 3x3, 5x5, 7x7, etc. BUT not 1x1

Why?

Why?

For an odd-sized filter, all the previous layer pixels would be symmetrically around the output pixel. Without this symmetry, we will have to account for distortions across the layers which happens when using an even sized kernel. Therefore, even sized kernel filters are mostly skipped to promote implementation simplicity. If you think of convolution as an interpolation from the given pixels to a center pixel, we cannot interpolate to a center pixel using an even-sized filter.

Why?



- Also see: https://medium.com/@abhishekjainindore24/all-about-convolutions-kernels-features-in-cnn-c656616390a1

Lets code!

- Let us go to ICHW 13 (in-class homework) in Canvas (we will work on the in-class homework code together in-class).