Convolutional Neural Networks

Introduction

- A **CNN (Convolutional Neural Network)** is a special type of **neural network** mainly used to **analyze images and visual data**.
- A **Convolutional Neural Network (CNN)** is a **deep learning model** designed to automatically learn and detect important features (like edges, colors, shapes, and textures) from images — without needing manual feature extraction.
- Convolutional Neural Network (CNN) is an advanced version of artificial neural networks (ANNs), primarily designed to extract features from grid-like matrix datasets.

Example

- If you give a CNN a picture of a **cat**, it will:
  - Detect **edges** (like the outline of the cat).
  - Combine them to detect **parts** (eyes, ears, tail).
  - Finally recognize the **whole object** (a cat).

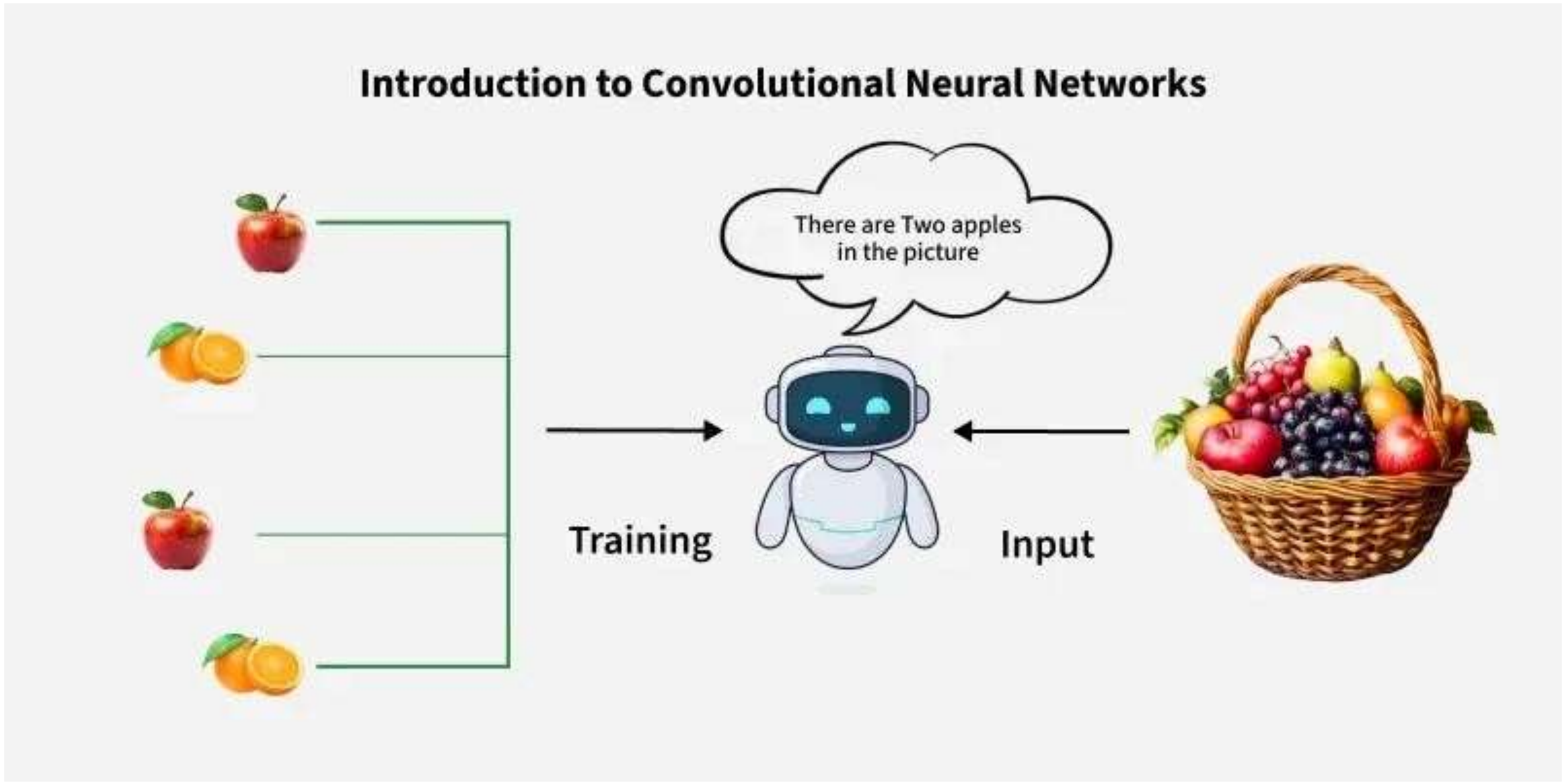
In CNNs, a filter (or kernel) means the small matrix of learnable weights used to detect patterns or features in the input image.

Filters

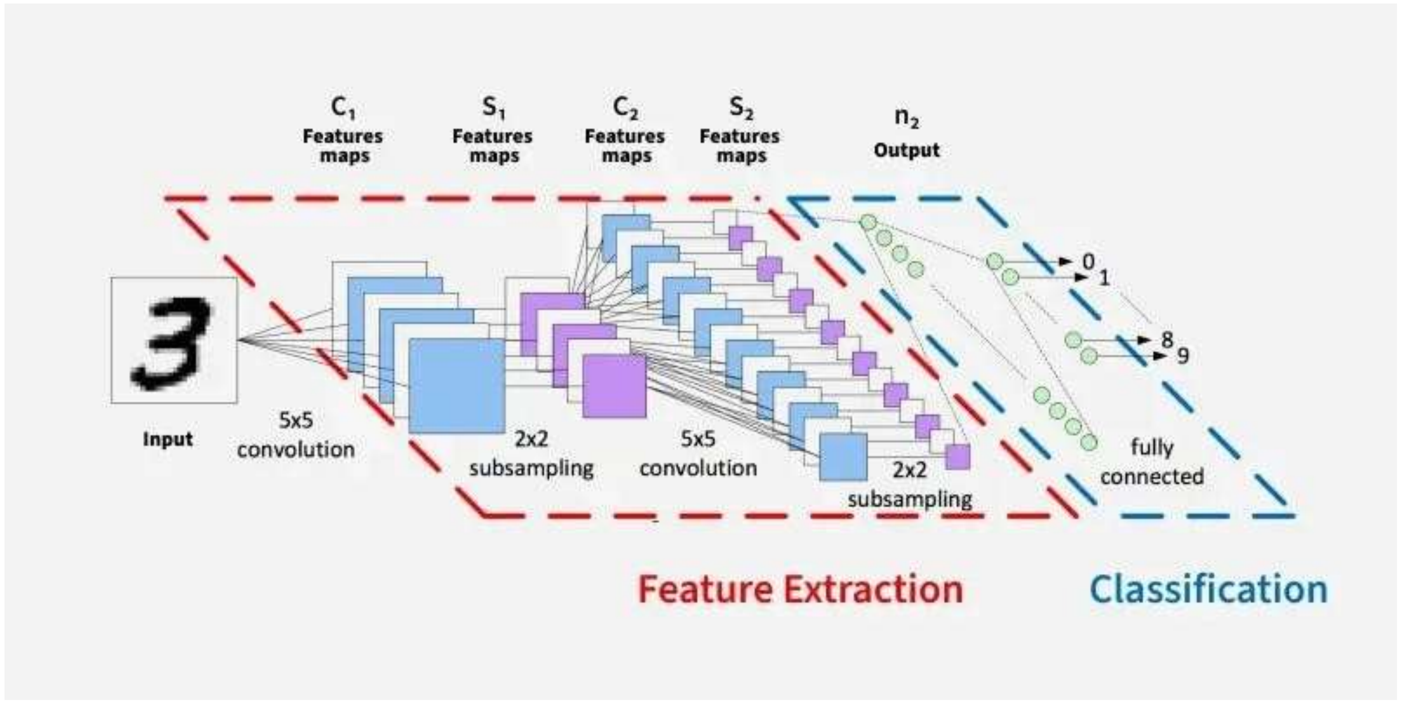
- In CNNs, a filter (or kernel) means the small matrix of learnable weights used to detect patterns or features in the input image.

- When the CNN trains:Every filter in a convolutional layer learns to detect a certain kind of feature.
  - Early filters → simple edges
  - Middle filters → patterns or textures
  - Deep filters → complex parts or full objects

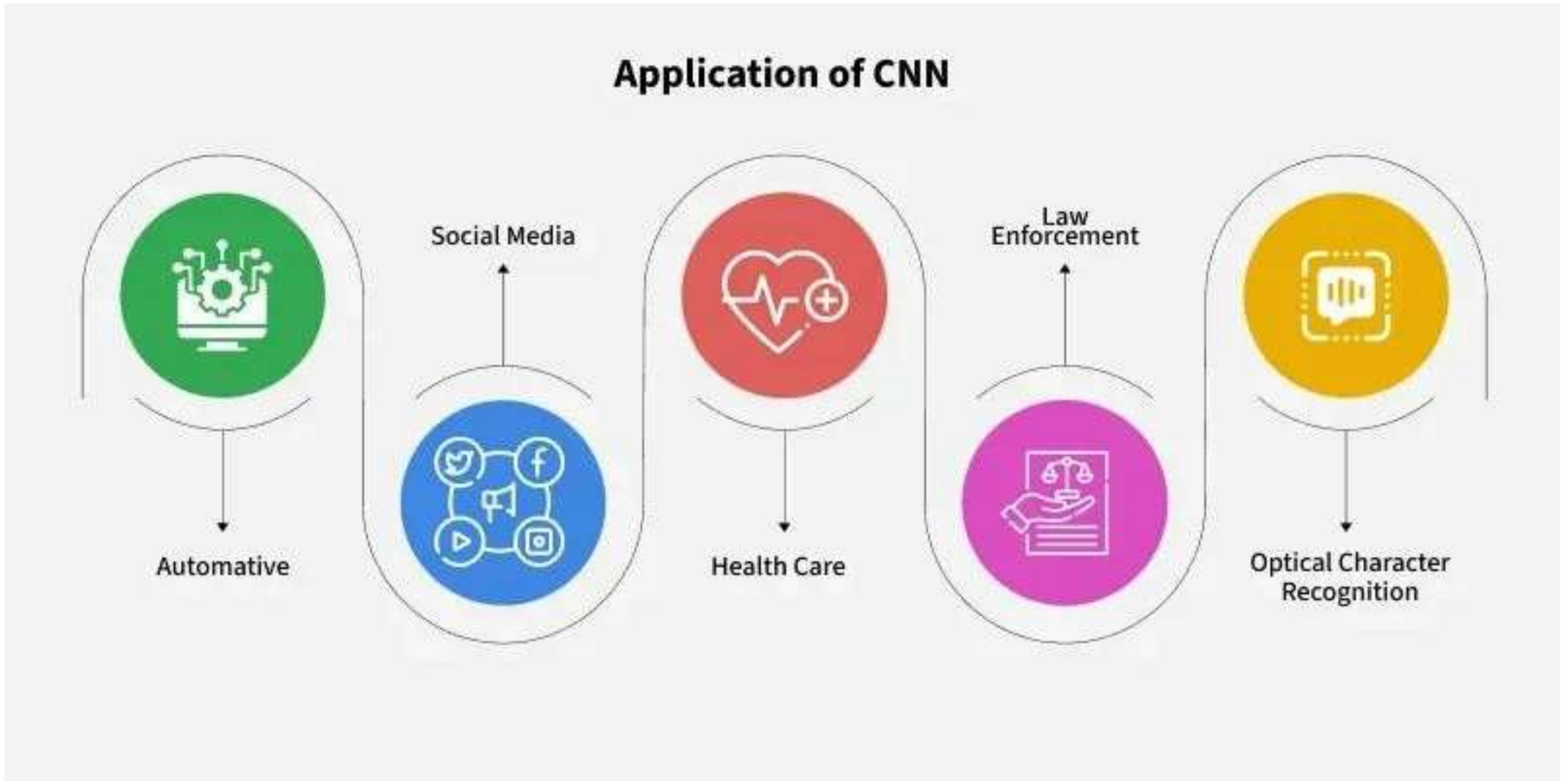
- But one filter alone can’t detect an entire object —many filters work together to build up that understanding.


Introduction to Convolutional Neural Networks
There are Two apples in the picture
Training
Input

Architecture



Applications



- ❑ Image recognition (e.g., Face ID, photo tagging)
- ❑ Object detection (e.g., self-driving cars)
- ❑ Medical imaging (e.g., detecting tumors)
- ❑ Video and gesture recognition

Common Uses of CNN

- Image classification (e.g., recognizing handwritten digits in MNIST)
- Object detection (e.g., detecting faces, cars)
- Image segmentation (e.g., medical image analysis)
- Video analysis (e.g., action recognition)
- Other areas like speech, text, and time-series (with adaptations)

Key Components

- ❑Convolutional Layers[detects features (edges, colors, patterns)]

  These layers apply convolutional operations to input images using filters or kernels to detect features such as edges, textures and more complex patterns. Convolutional operations help preserve the spatial relationships between pixels.

- ❑Pooling Layers[reduces image size while keeping key features]

  They downsample the spatial dimensions of the input, reducing the computational complexity and the number of parameters in the network. Max pooling is a common pooling operation where we select a maximum value from a group of neighboring pixels.

- ❑ Activation Functions: They introduce non-linearity to the model by allowing it to learn more complex relationships in the data.

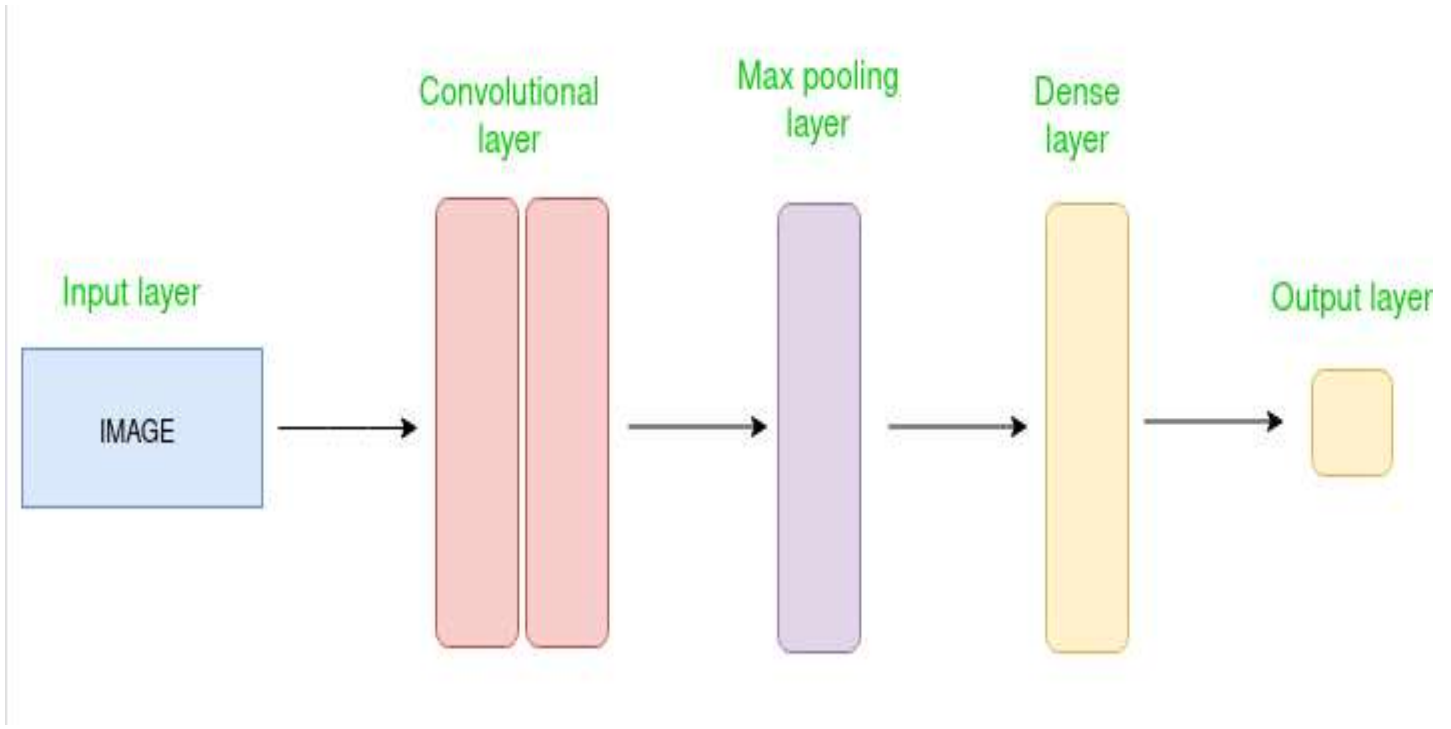
- ❑ Fully Connected Layers[makes the final prediction (like “cat” or “dog”)]
  These layers are responsible for making predictions based on the high-level features learned by the previous layers. They connect every neuron in one layer to every neuron in the next layer.

- ➢ After **every convolution operation**, we apply an **activation function** (usually **ReLU**) to the result (feature map).**So the sequence is: Convolution → Activation → Pooling**

When to Apply Activation?

Step	Apply Activation?	Common Function
After Convolution	✓ Yes	**ReLU**
After Pooling	✗ No	—
After Fully Connected Layer	✓ Yes	**ReLU / Sigmoid / Softmax**
Output Layer	✓ Yes	**Softmax / Sigmoid / Linear**

CNN Architecture



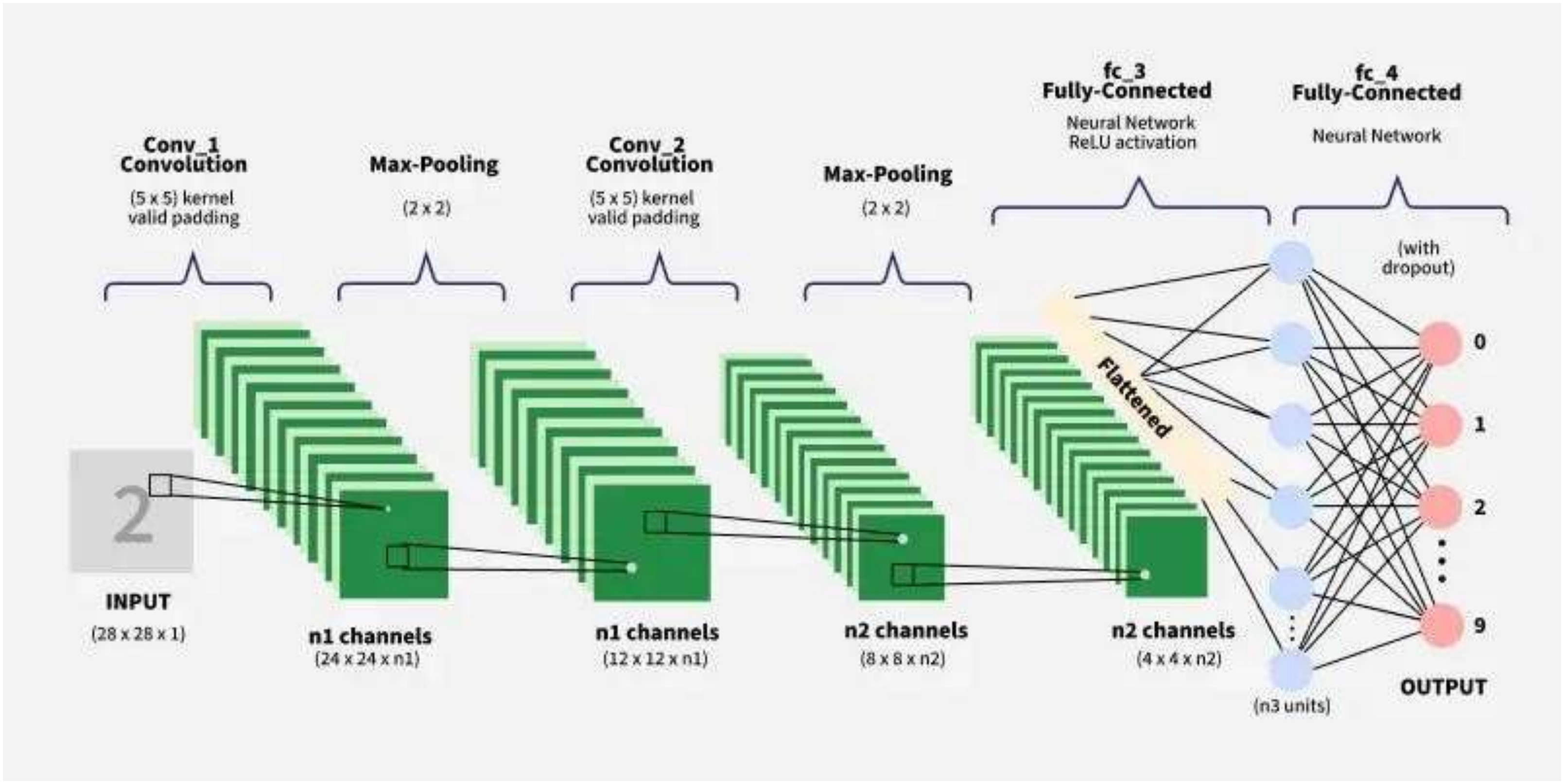
CNN = A deep learning model that helps computers “see” and understand images.

Working of CNN

- ❑ Input Image: CNN receives an input image which is preprocessed to ensure uniformity in size and format.
- ❑ Convolutional Layers: Filters are applied to the input image to extract features like edges, textures and shapes.
- ❑ Pooling Layers: The feature maps generated by the convolutional layers are downsampled to reduce dimensionality.

- Fully Connected Layers: The downsampled feature maps are passed through fully connected layers to produce the final output, such as a classification label.

- Output: The CNN outputs a prediction, such as the class of the image.


Conv_1 Convolution
(5 x 5) kernel valid padding
Max-Pooling
(2 x 2)
Conv_2 Convolution
(5 x 5) kernel valid padding
Max-Pooling
(2 x 2)
fc_3 Fully-Connected
Neural Network ReLU activation
fc_4 Fully-Connected
Neural Network
(with dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
(n3 units)
0
1
2
9
OUTPUT

Training a Convolutional Neural Network

- ❑ CNNs are trained using a supervised learning approach. This means that the CNN is given a set of labeled training images. The CNN learns to map the input images to their correct labels.
- ❑ The training process for a CNN involves the following steps:

- ➢ Data Preparation: The training images are preprocessed to ensure that they are all in the same format and size.
- ➢ Loss Function: A loss function is used to measure how well the CNN is performing on the training data. The loss function is typically calculated by taking the difference between the predicted labels and the actual labels of the training images.
- ➢ Optimizer: An optimizer is used to update the weights of the CNN in order to minimize the loss function.
- ➢ Backpropagation: Backpropagation is a technique used to calculate the gradients of the loss function with respect to the weights of the CNN. The gradients are then used to update the weights of the CNN using the optimizer.

How to Evaluate CNN Models

- Efficiency of CNN can be evaluated using a variety of criteria. Among the most popular metrics are:

- Accuracy: Accuracy is the percentage of test images that the CNN correctly classifies.
- Precision: Precision is the percentage of test images that the CNN predicts as a particular class and that are actually of that class.
- Recall: Recall is the percentage of test images that are of a particular class and that the CNN predicts as that class.
- F1 Score: The F1 Score is a harmonic mean of precision and recall. It is a good metric for evaluating the performance of a CNN on classes that are imbalanced.

Different Types of CNN Models

- ❑ LeNet: LeNet developed by Yann LeCun and his colleagues in the late 1990s was one of the first successful CNNs designed for handwritten digit recognition. It laid the foundation for modern CNNs and achieved high accuracy on the MNIST dataset which contains 70,000 images of handwritten digits (0-9).

- ❑ AlexNet: AlexNet is a CNN architecture that was developed by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton in 2012. It was the first CNN to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) a major image recognition competition. It consists of several layers of convolutional and pooling layers followed by fully connected layers. The architecture includes five convolutional layers, three pooling layers and three fully connected layers.

- Resnet: ResNets (Residual Networks) are designed for image recognition and processing tasks. They are renowned for their ability to train very deep networks without overfitting making them highly effective for complex tasks. It introduces skip connections that allow the network to learn residual functions making it easier to train deep architecture.

- GoogleNet: GoogleNet also known as InceptionNet is renowned for achieving high accuracy in image classification while using fewer parameters and computational resources compared to other state-of-the-art CNNs. The core component of GoogleNet allows the network to learn features at different scales simultaneously to enhance performance.

- VGG: VGGs are developed by the Visual Geometry Group at Oxford, it uses small 3x3 convolutional filters stacked in multiple layers, creating a deep and uniform structure. Popular variants like VGG-16 and VGG-19 achieved state-of-the-art performance on the ImageNet dataset demonstrating the power of depth in CNNs.

Example: Build ConvNets

- Let's take an example by running a covnets on of image of dimension 32 x 32 x 3.

- **Input Layers:** It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.

- **Convolutional Layer:** This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2x2, 3x3, or 5x5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension 32 x 32 x 12.

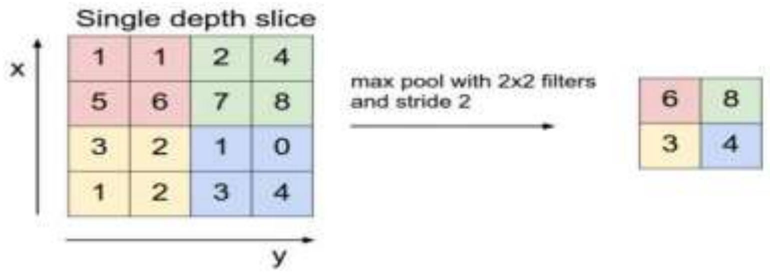
- Activation Layer: By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.

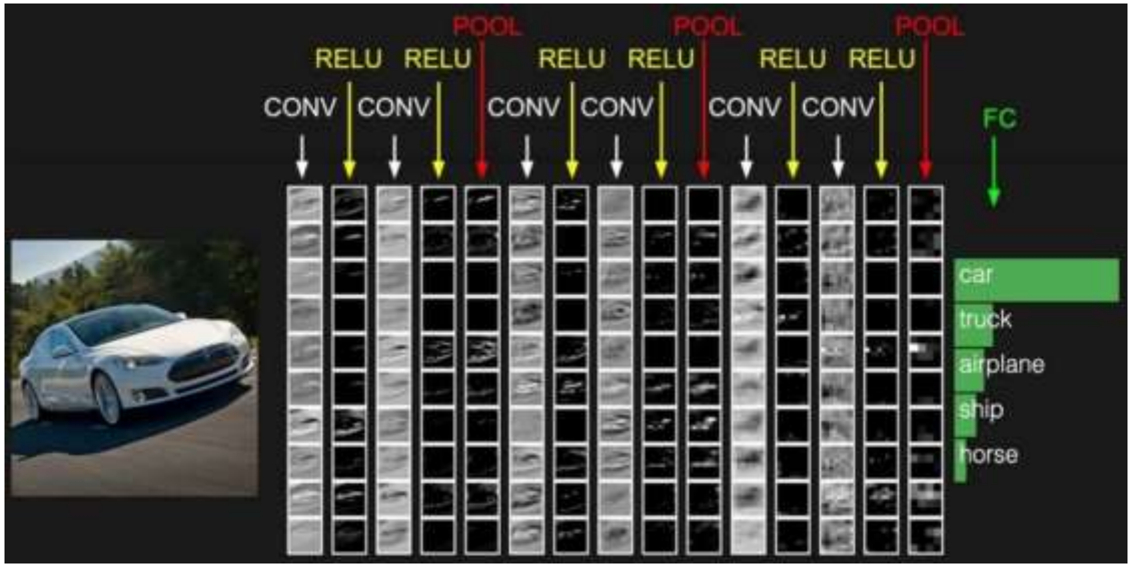
- Pooling layer: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

- **Flattening:** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.
- **Fully Connected Layers:** It takes the input from the previous layer and computes the final classification or regression task.

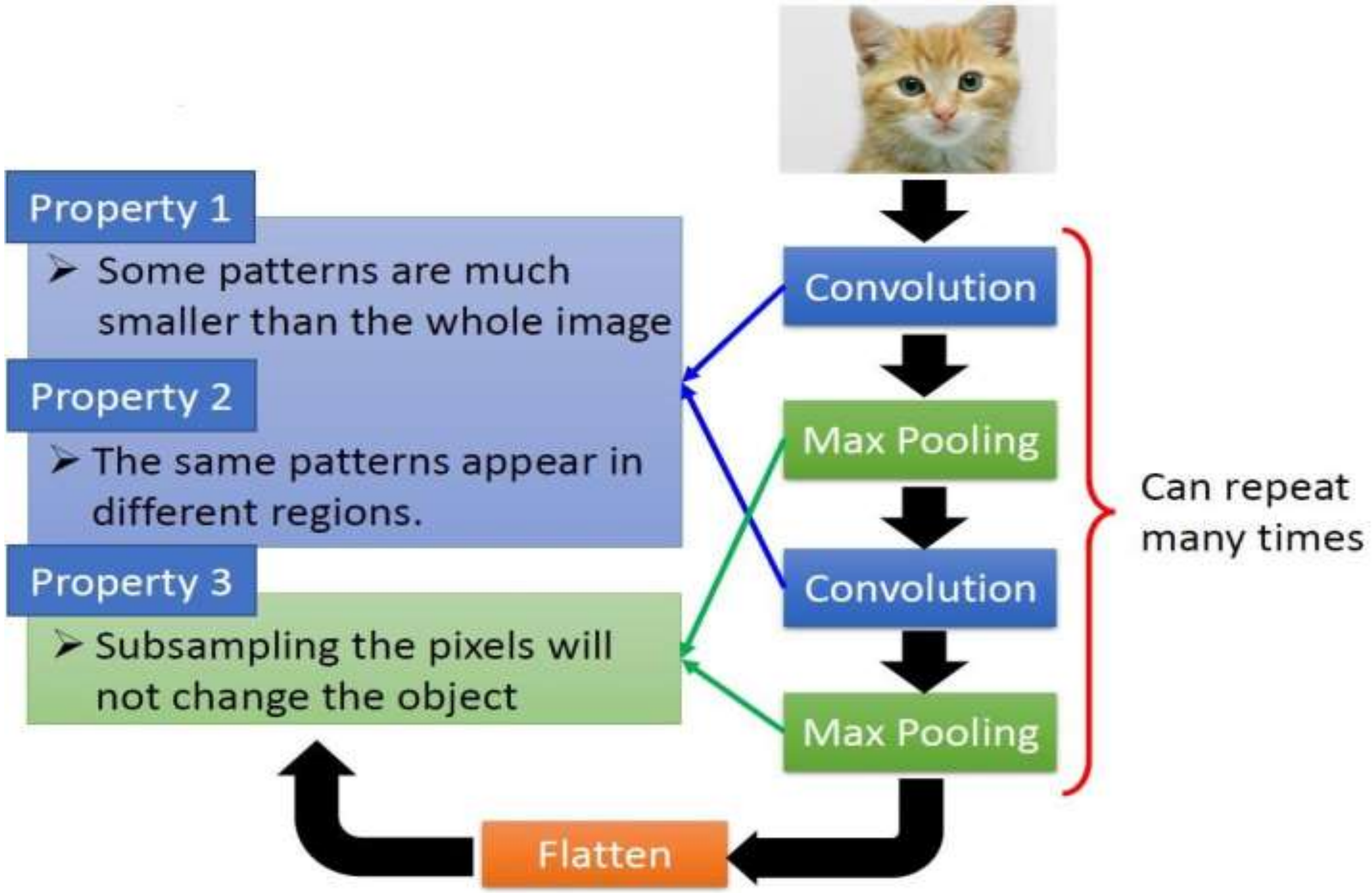
- **Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

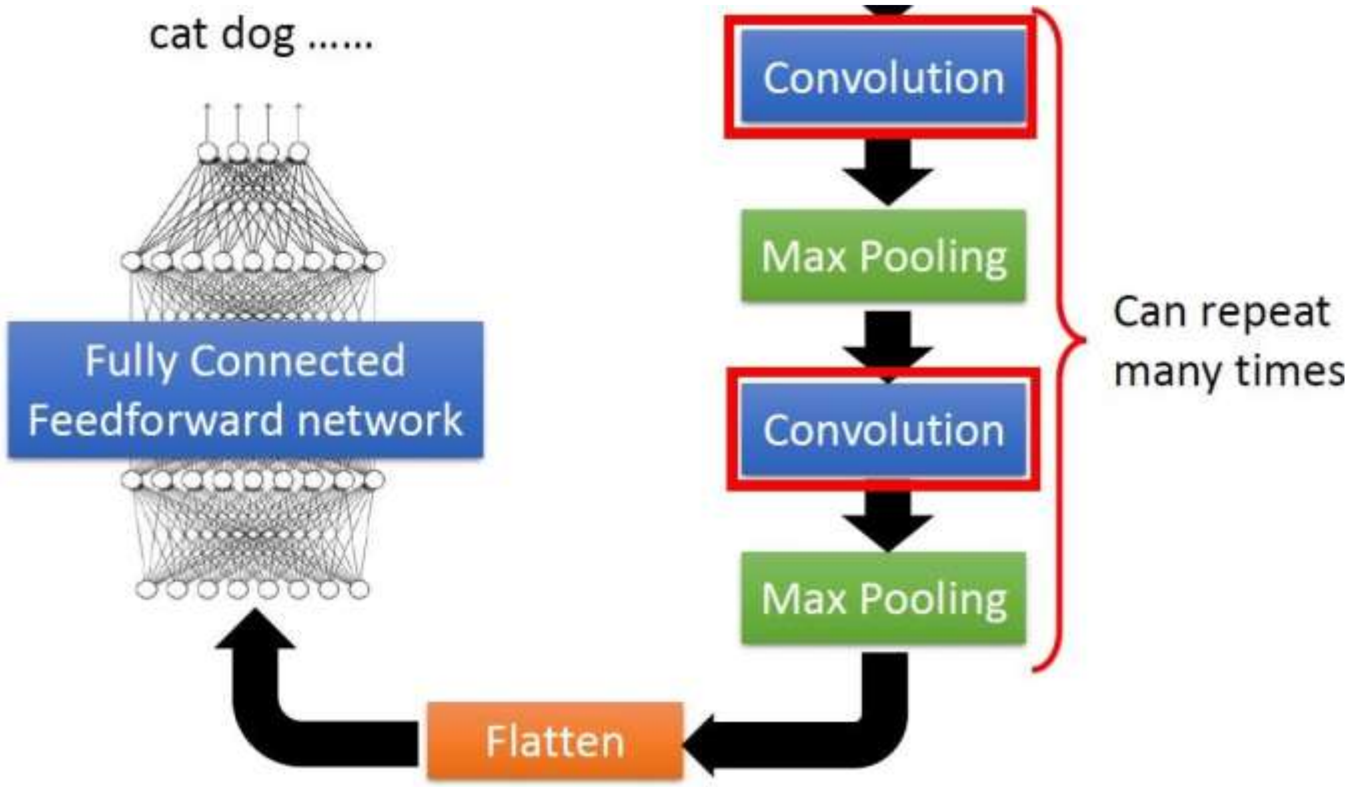
Pooling Illustration





Whole CNN Elaborations


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


cat dog ……
Fully Connected
Feedforward network
Convolution
Max Pooling
Convolution
Max Pooling
Can repeat
many times
Flatten

Convolution

- In CNNs, it means sliding a small matrix called a filter or kernel over the image and performing element-wise multiplication and summation.

  - Basic Idea:
  - Imagine your image as a grid of numbers — each number represents the intensity (brightness) of a pixel.
  - A filter (kernel) is also a small grid of numbers (like 3×3 or 5×5) that focuses on a small region of the image at a time.
  - You slide (or convolve) the filter across the image, multiply and sum the values, and store the result in a new grid called a feature map.

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

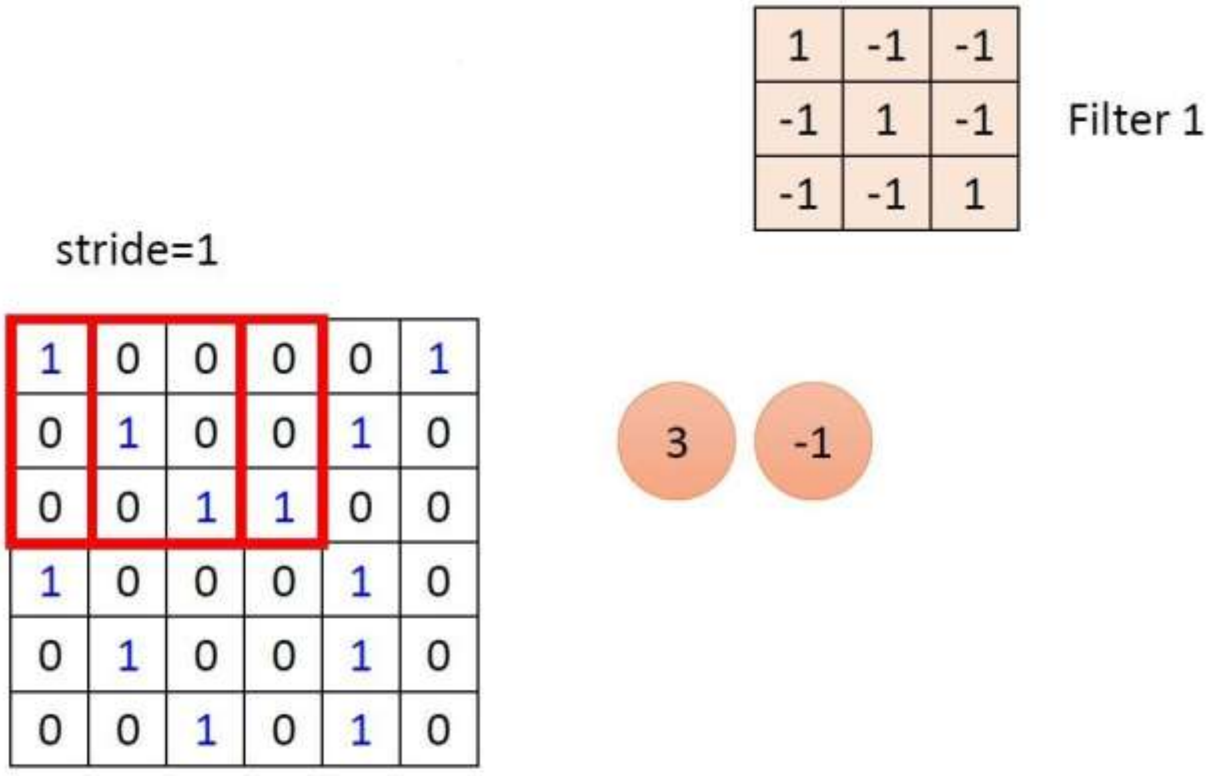
Filter 1
Matrix

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

Filter 2
Matrix

⋮

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



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

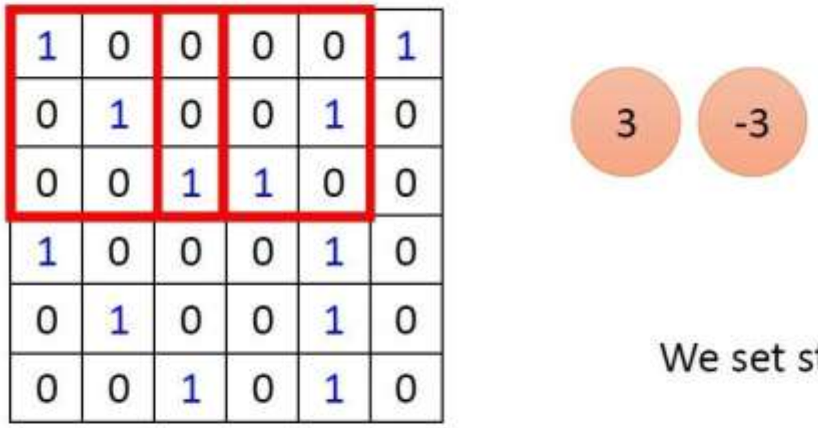
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

Filter 1

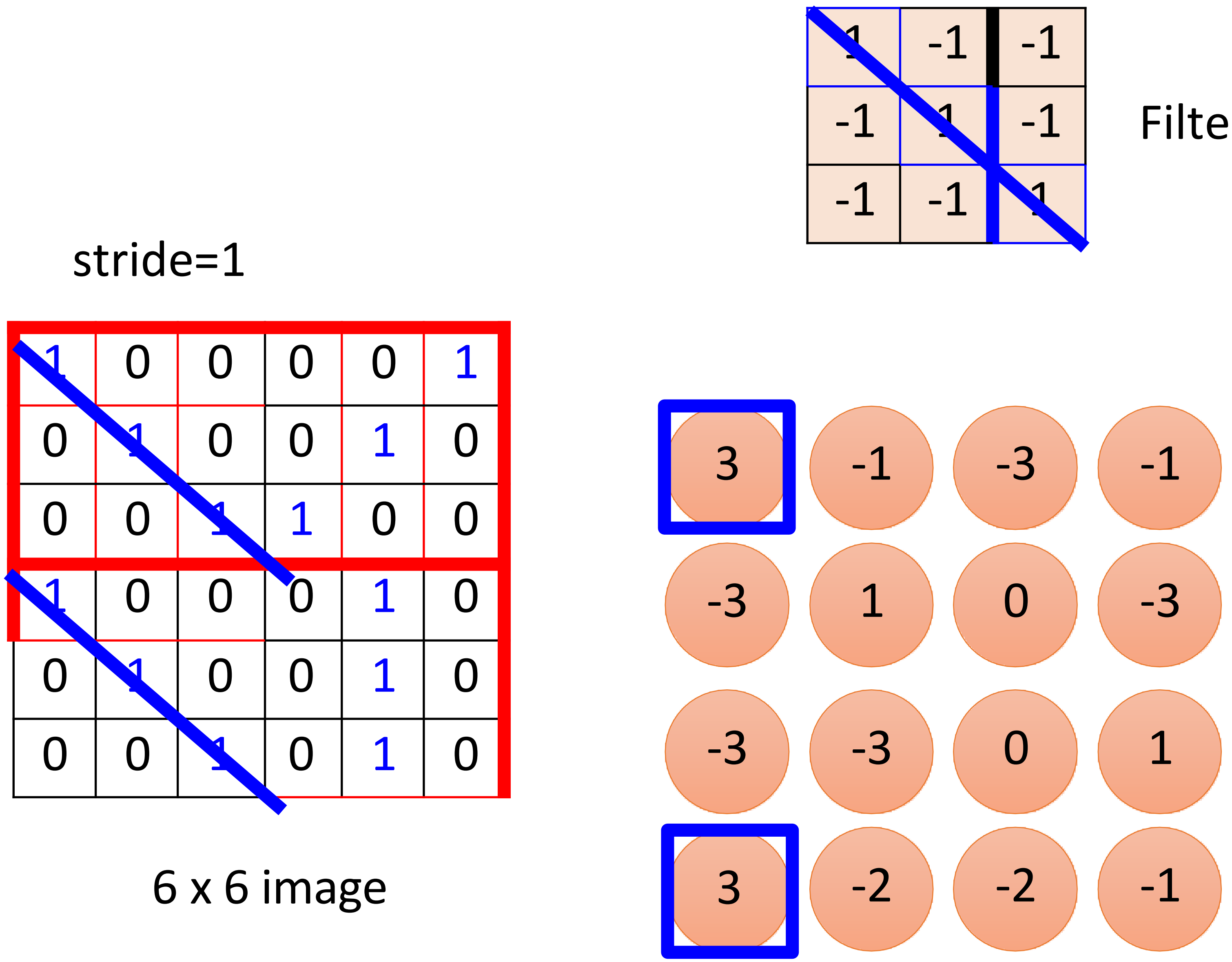
If stride=2



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

We set stride=1 below

6 x 6 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

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

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

Property 2

CNN – Convolution

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

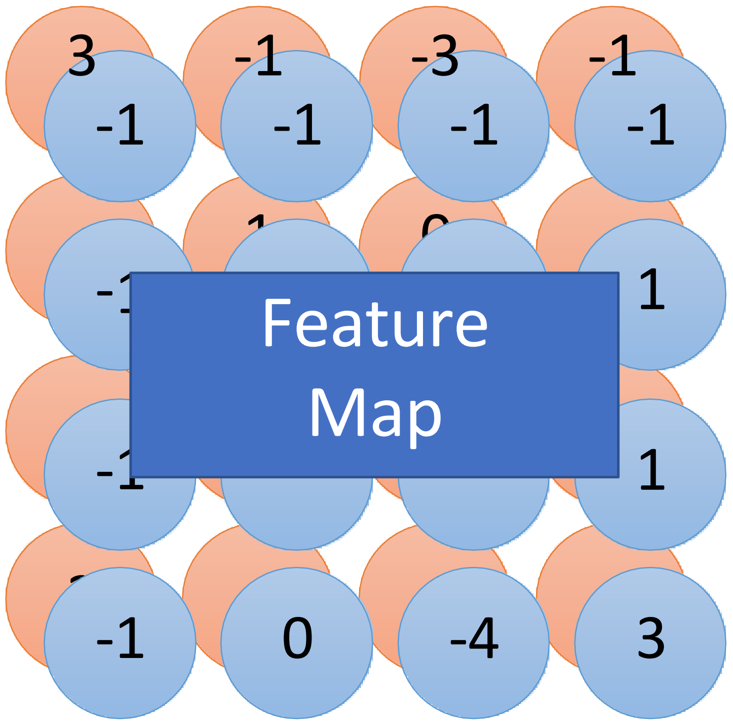
Filter 2

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

Do the same process for every filter



4 x 4 image

CNN – Colorful image

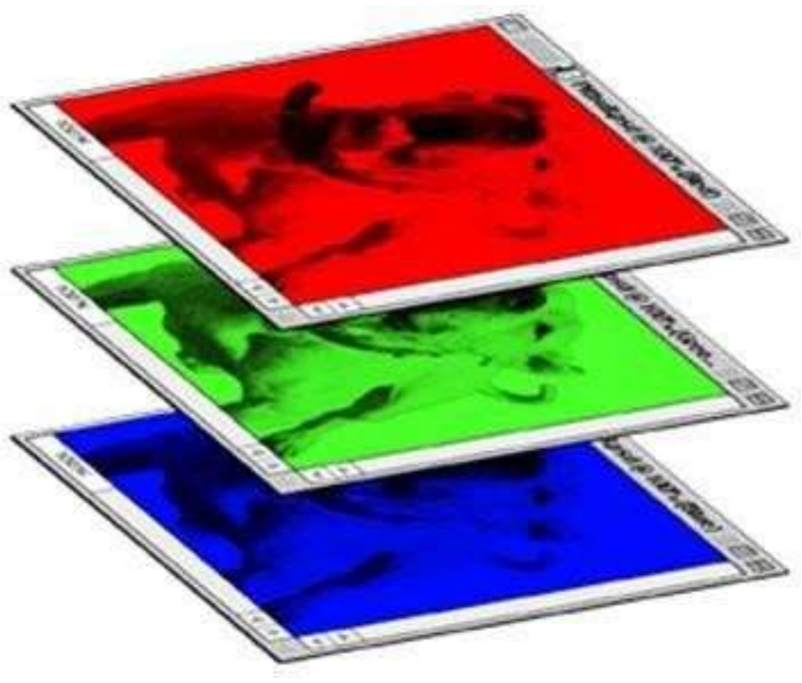
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

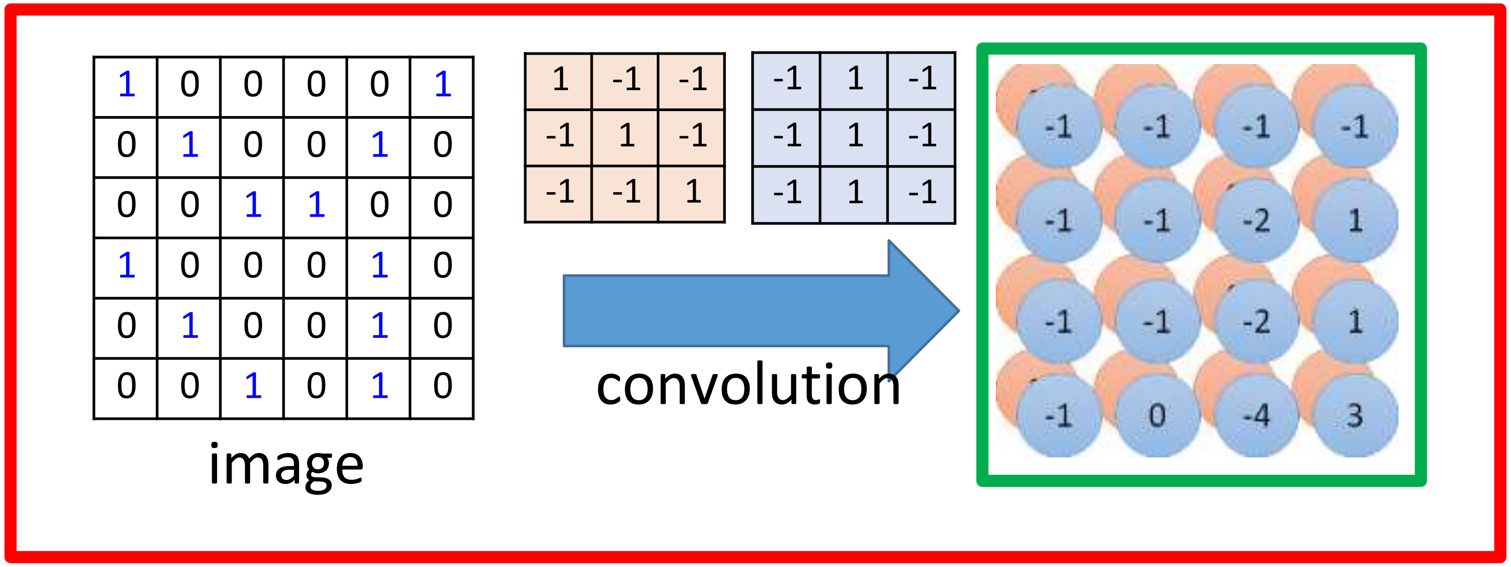
Filter 2

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

Convolution v.s. Fully Connected



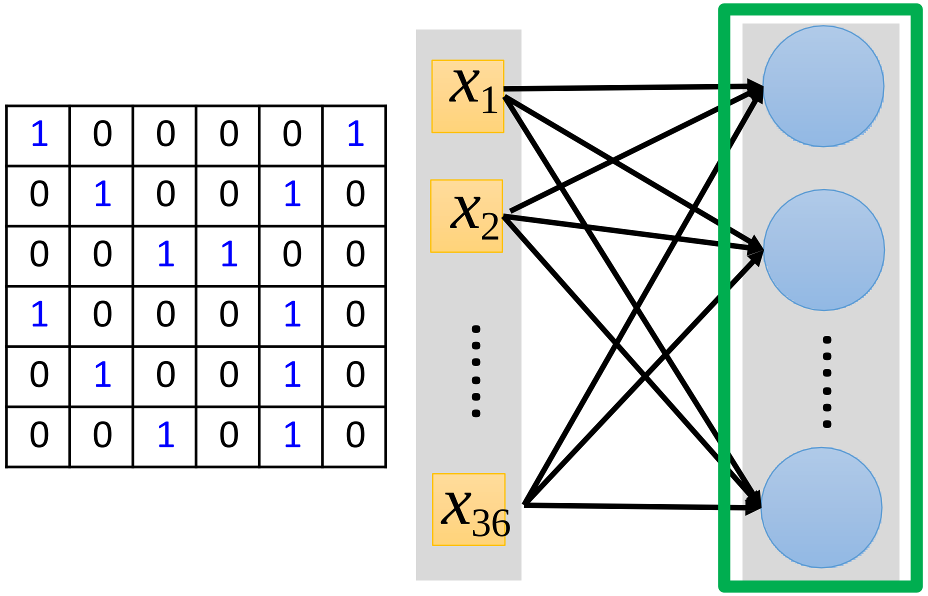
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

image

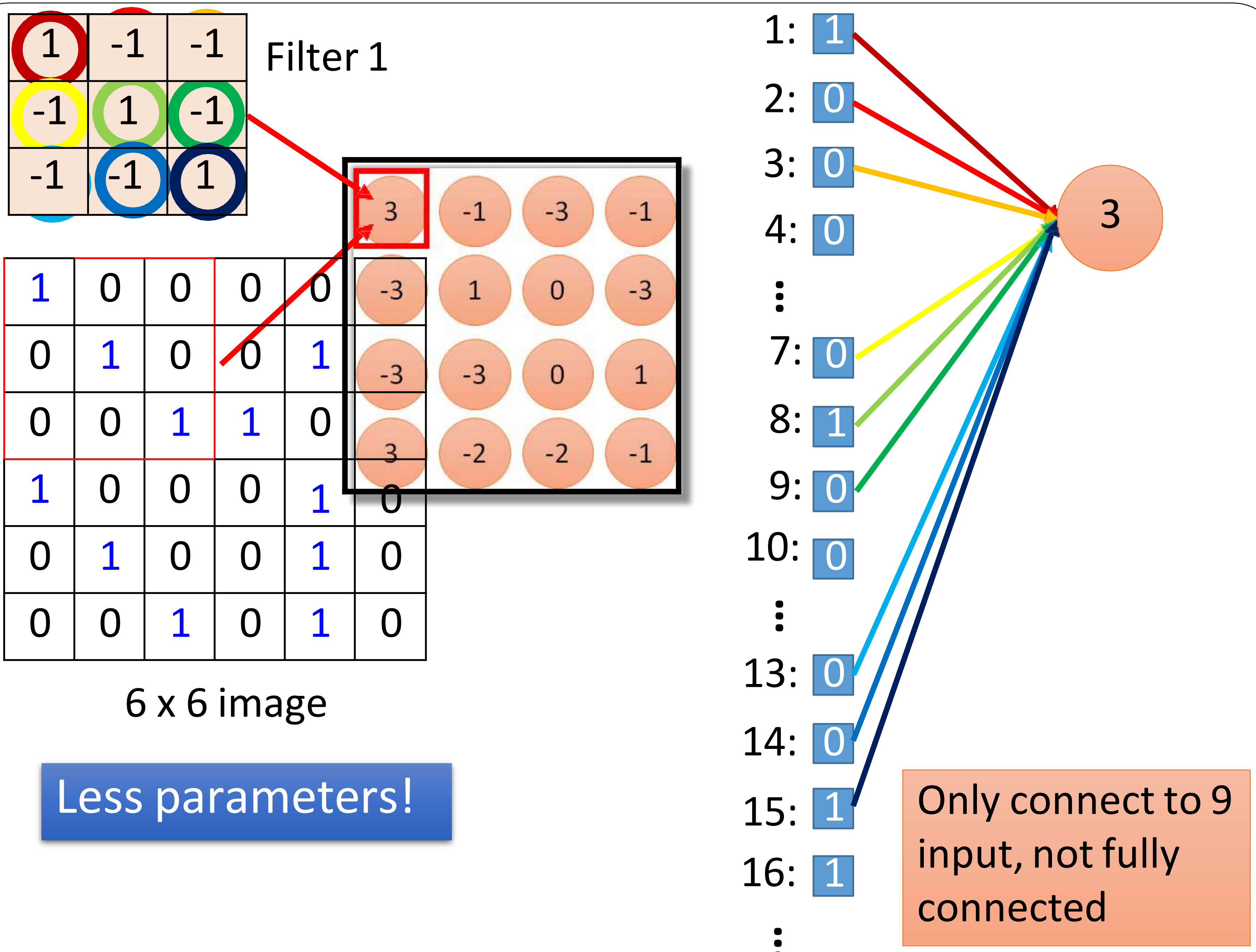
1	-1	-1
-1	1	-1
-1	-1	1

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

Fully-connected

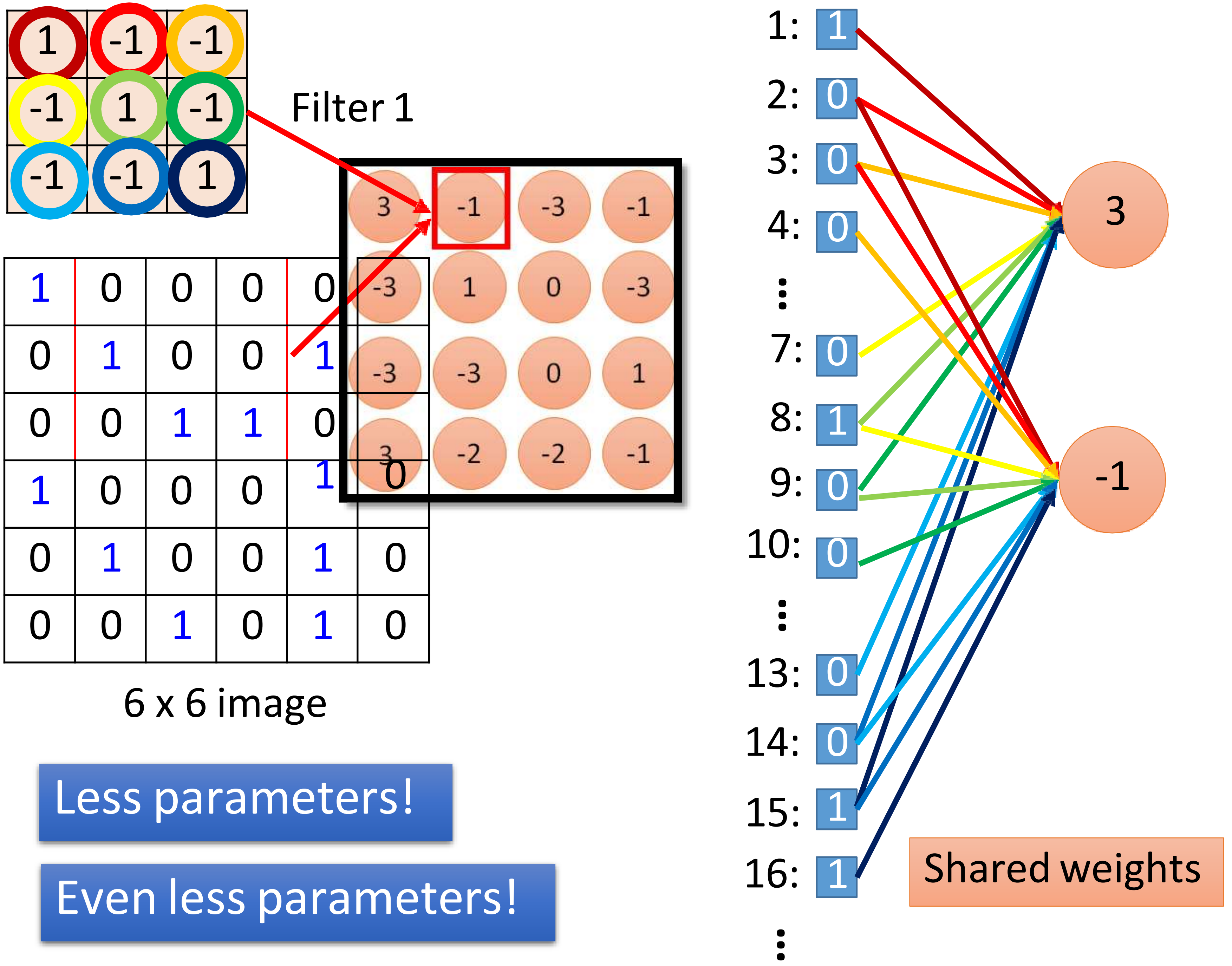


1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0



Less parameters!

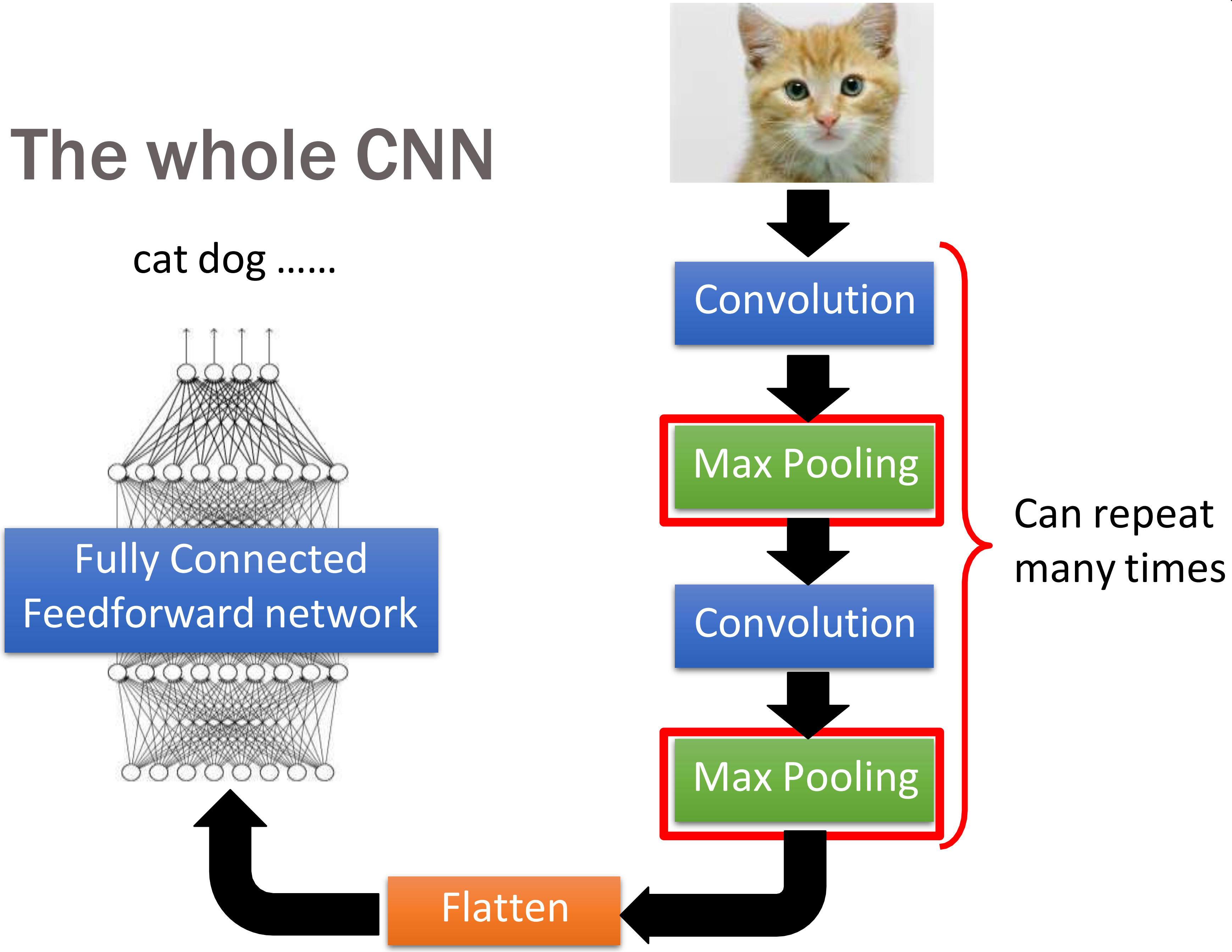
Only connect to 9 input, not fully connected



Less parameters!

Even less parameters!

The whole CNN



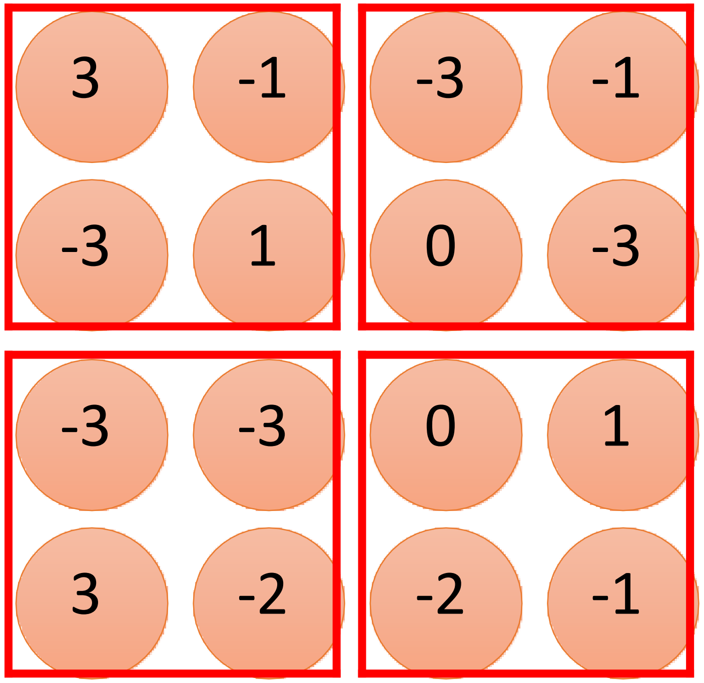
CNN – Max Pooling

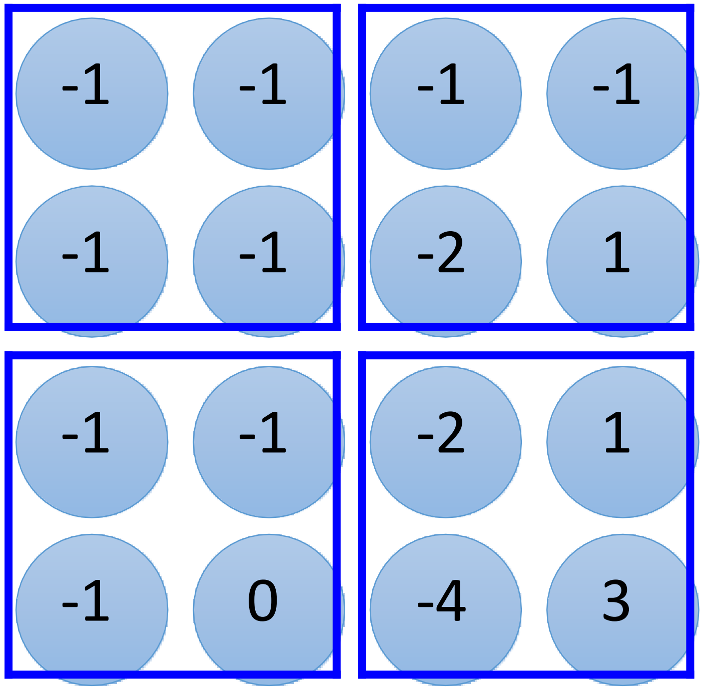
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2



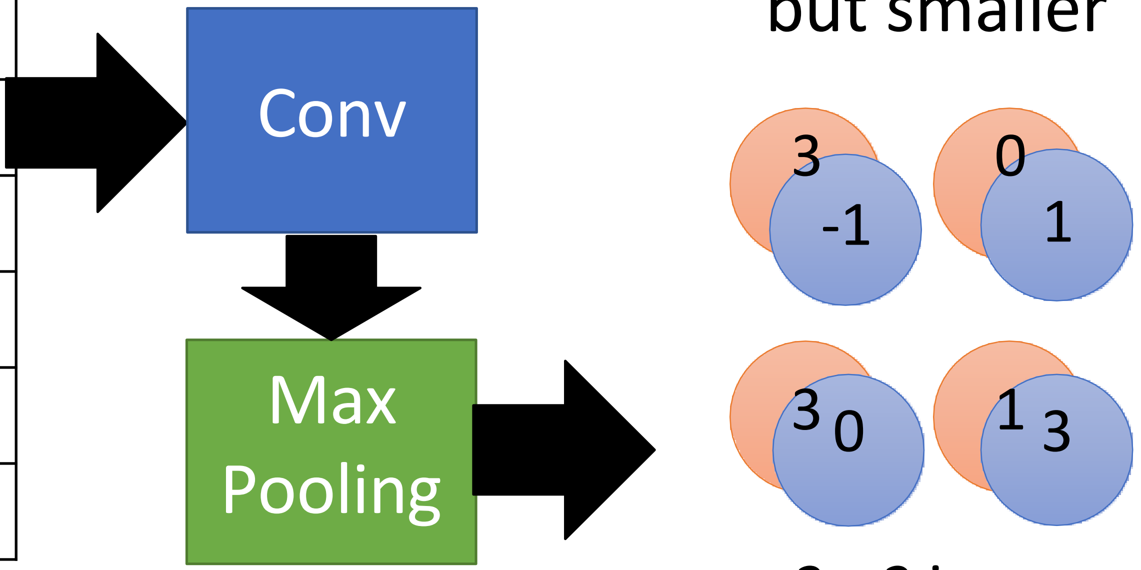


CNN – Max Pooling

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

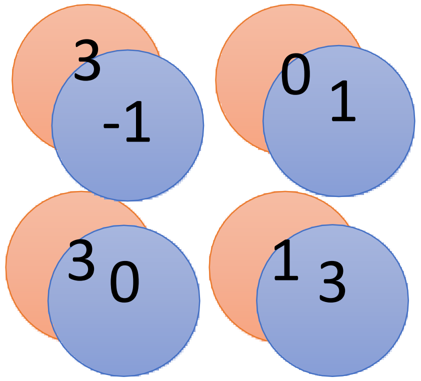
New image but smaller

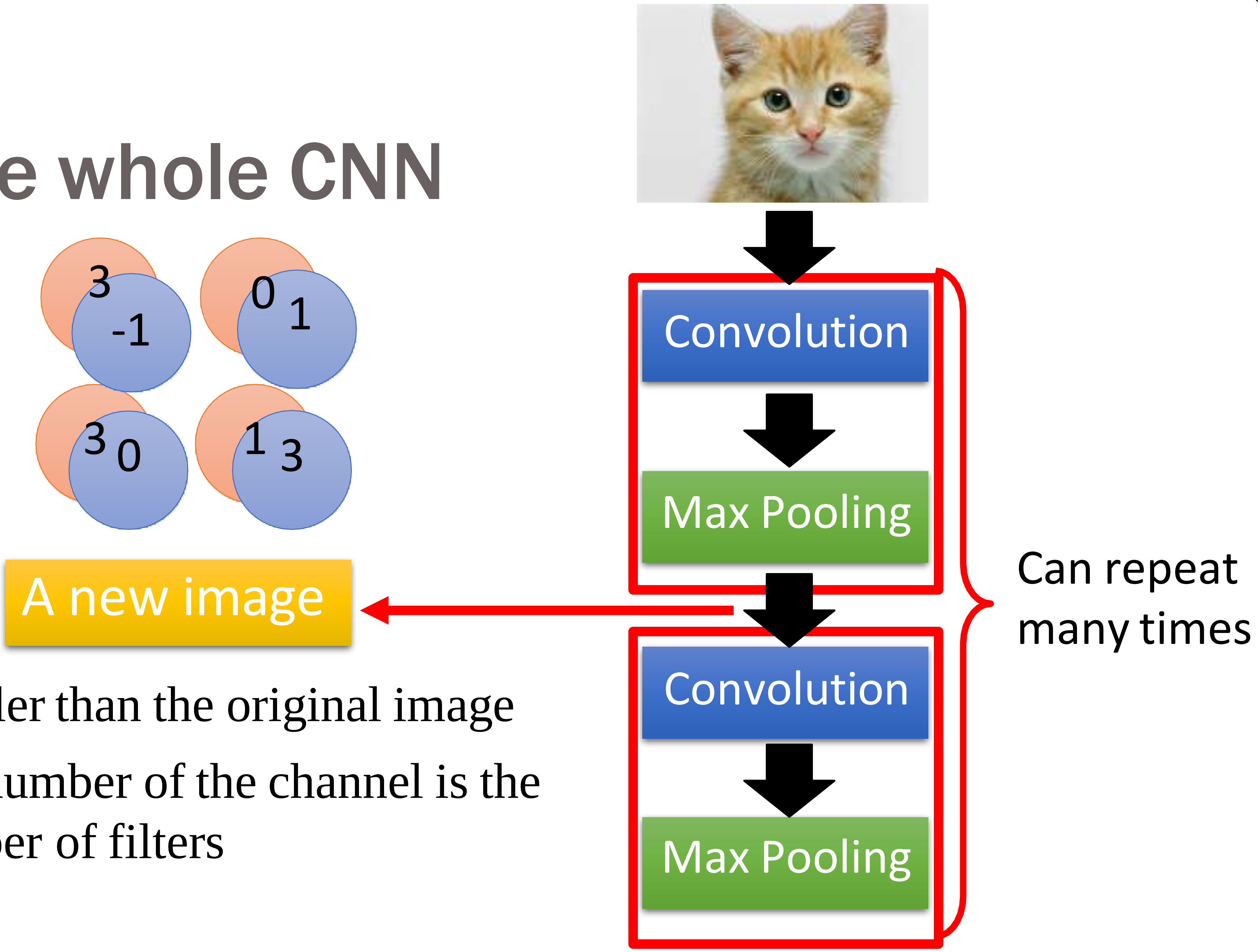


2 x 2 image

Each filter is a channel

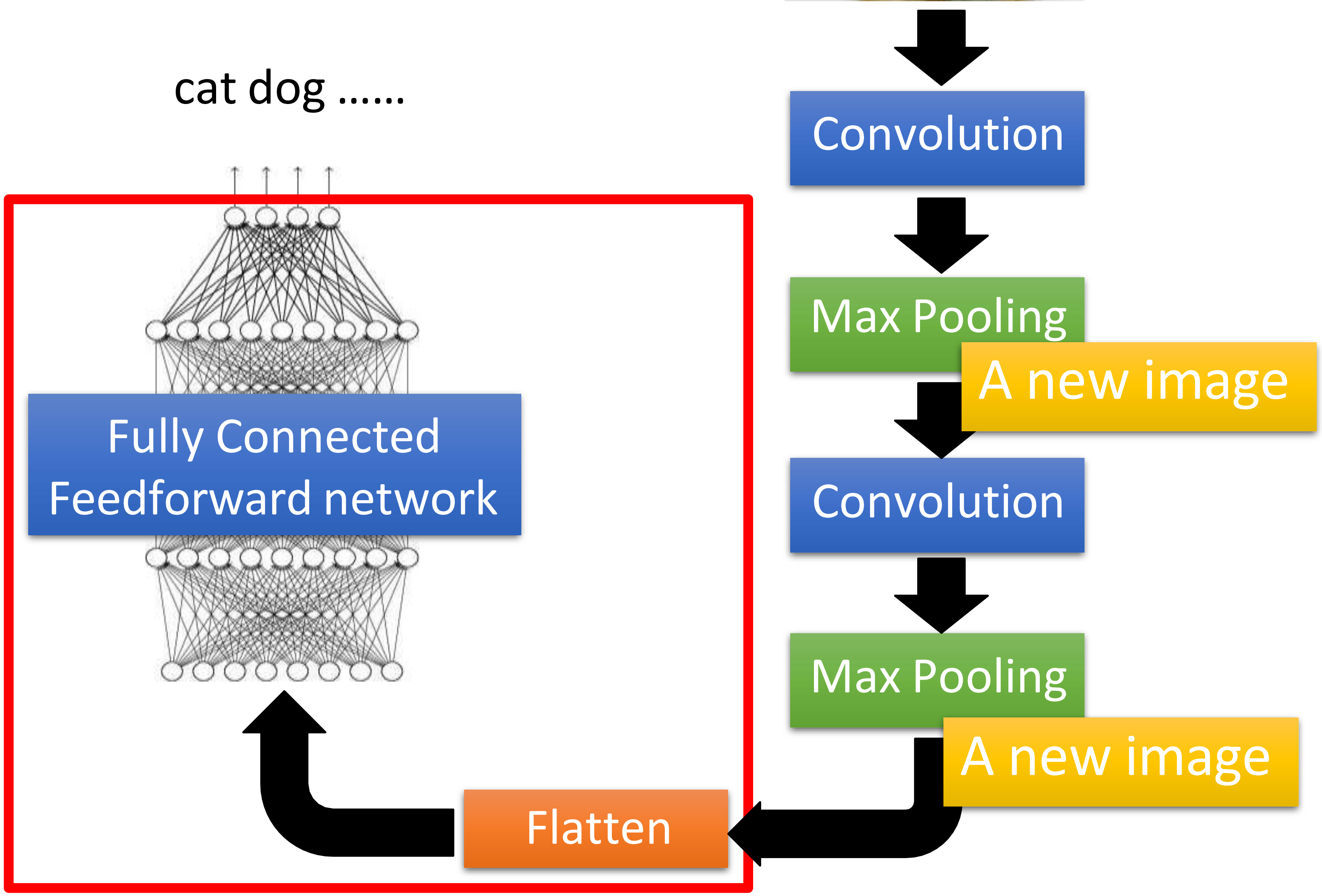
The whole CNN





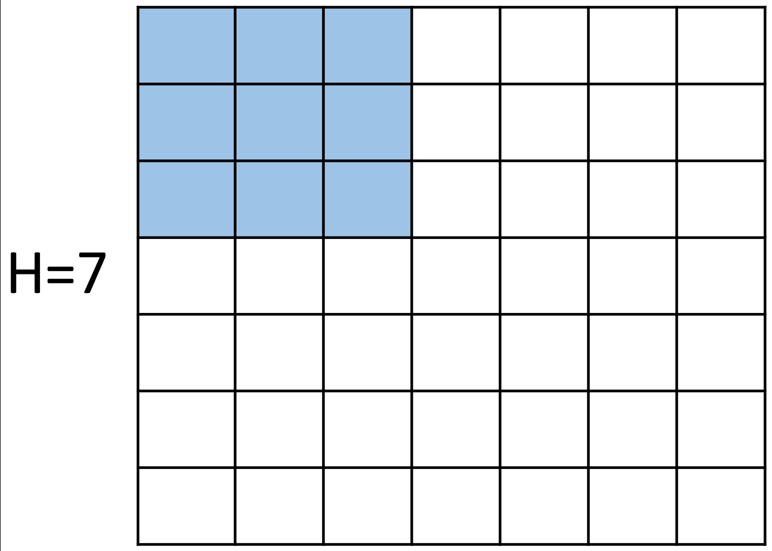
- ❑ Smaller than the original image
- ❑ The number of the channel is the number of filters

The whole CNN



Activation Map Size

- What is the size of the image after application of a filter with a given size and stride? W=7

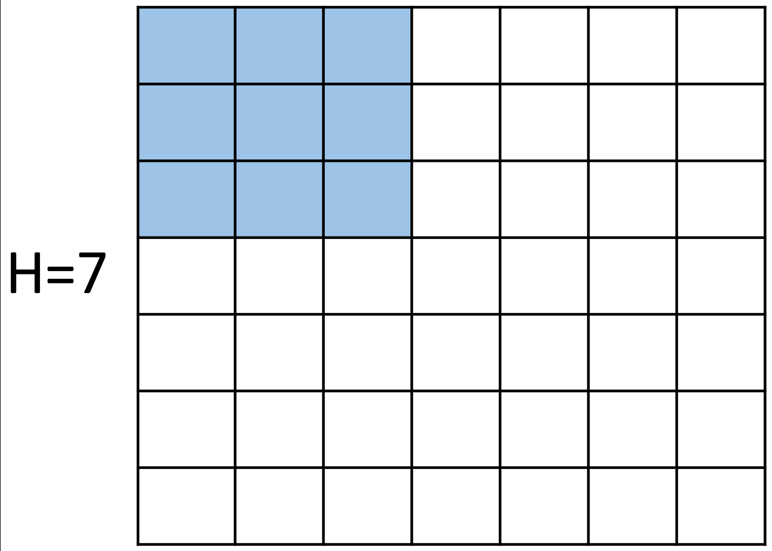


Take a 3x3 filter with stride 1 K=3, S=1

⇩

Output image is: 5x5

- What is the size of the image after application of a filter with a given size and stride? W=7



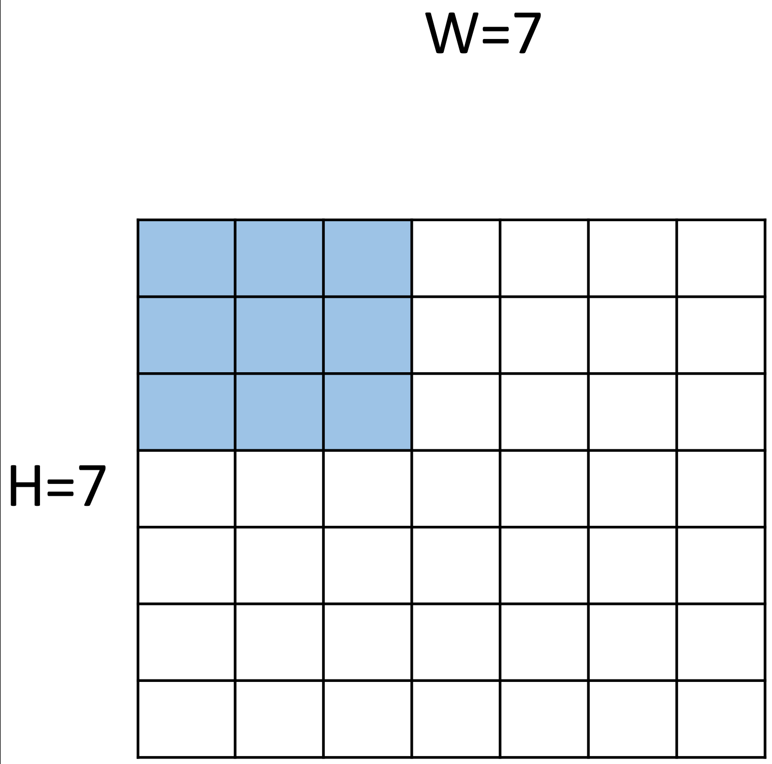
Take a 3x3 filter with stride 2

K=3, S=2

⇩

Output image is: 3x3

- What is the size of the image after application of a filter with a given size and stride?

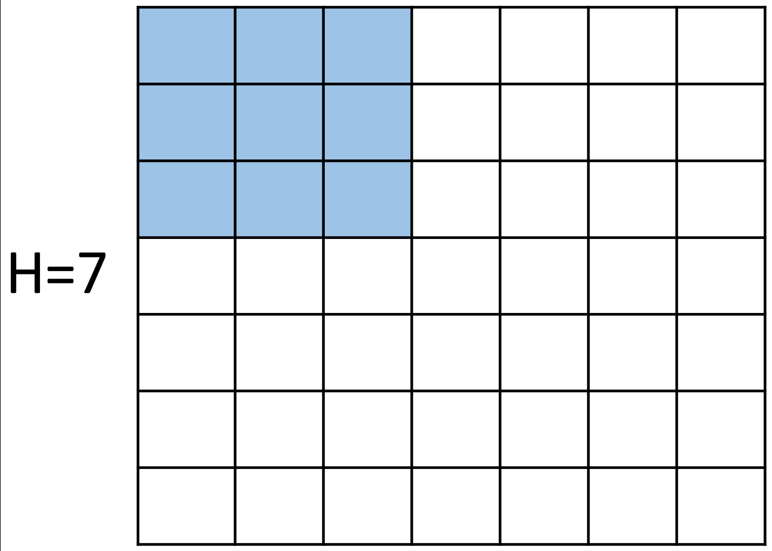


General rule

$$W' = \frac{W-K}{S} + 1$$

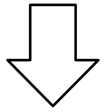
$$H' = \frac{H-K}{S} + 1$$

- What is the size of the image after application of a filter with a given size and stride? W=7



Take a 3x3 filter with stride 3

K=3, S=3



Output image is:

Doesn't fit! Cannot scan the whole image

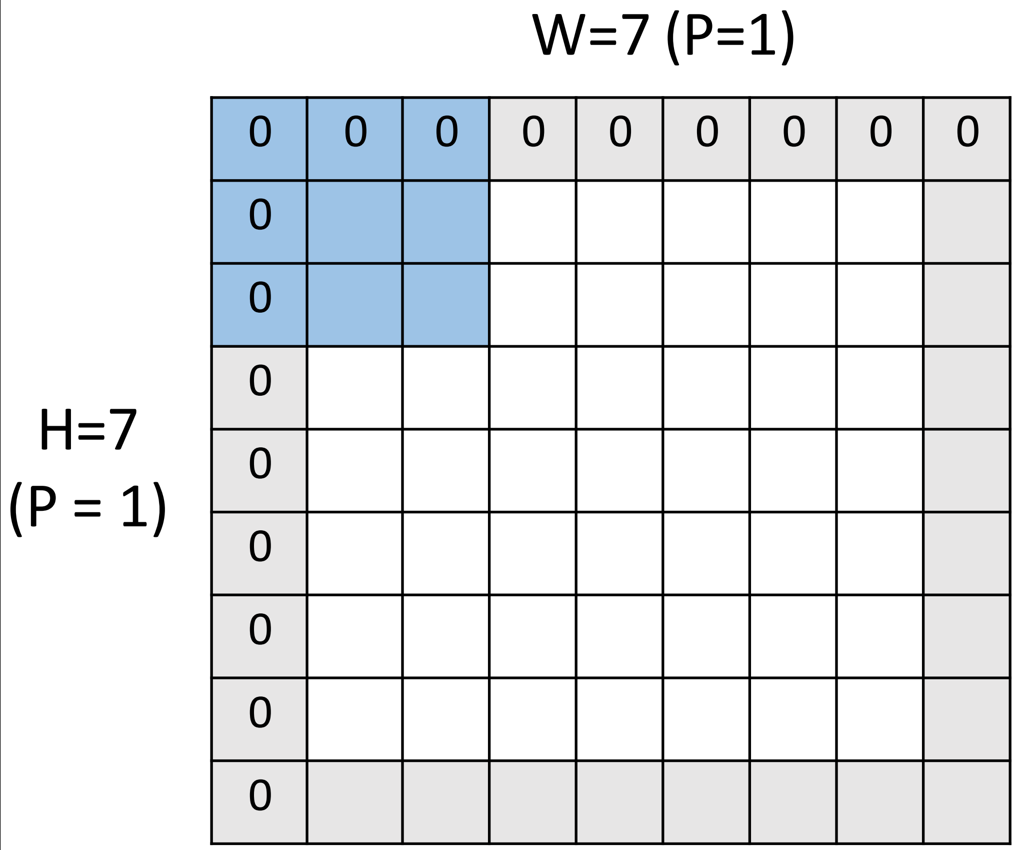
- Add columns and rows of zeros to the border of the image

W=7

H=7

0	0	0	0	0	0	0	0	0
0								
0								
0								
0								
0								
0								
0								
0								

- Add columns and rows of zeros to the border of the image



K=3, S=1

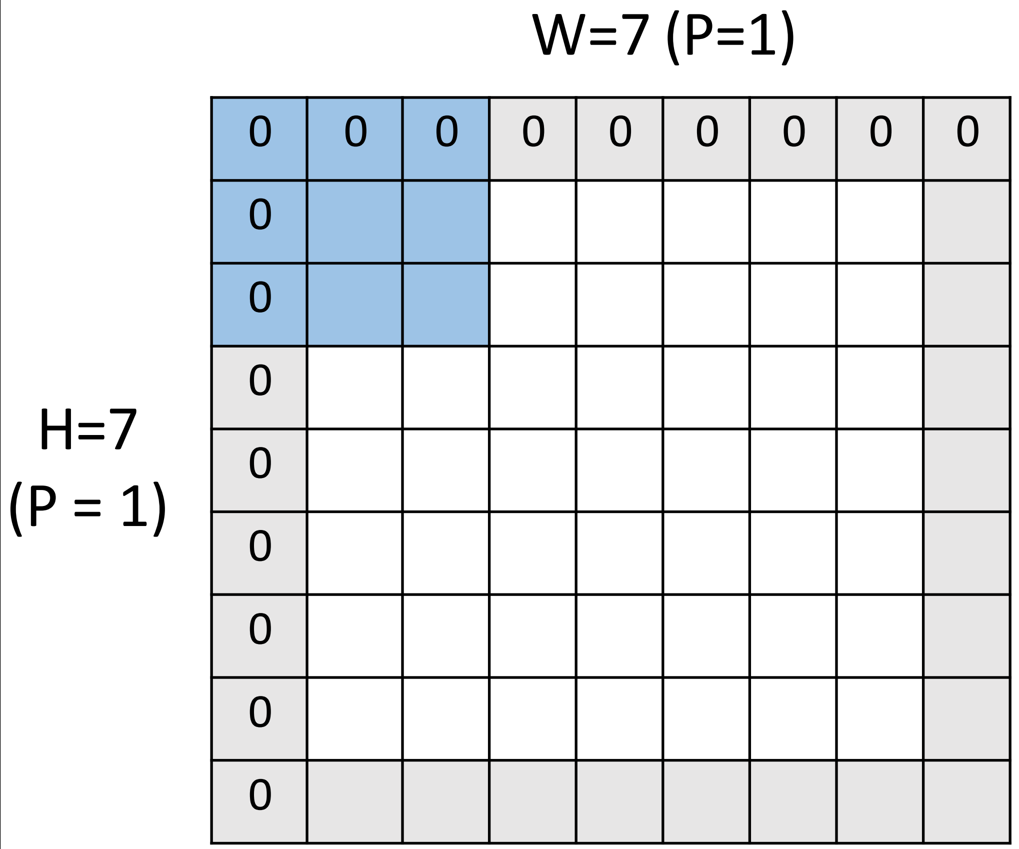
⇩

Output image is?

$$W' = \frac{W - K + 2P}{S} + 1$$

7x7

- ❑ Add columns and rows of zeros to the border of the image



- ❑ Zero padding serves to retain the original size of image

$$P = \frac{K-1}{2}$$

- ❑ Pad as necessary to perform convolutions with a given stride S