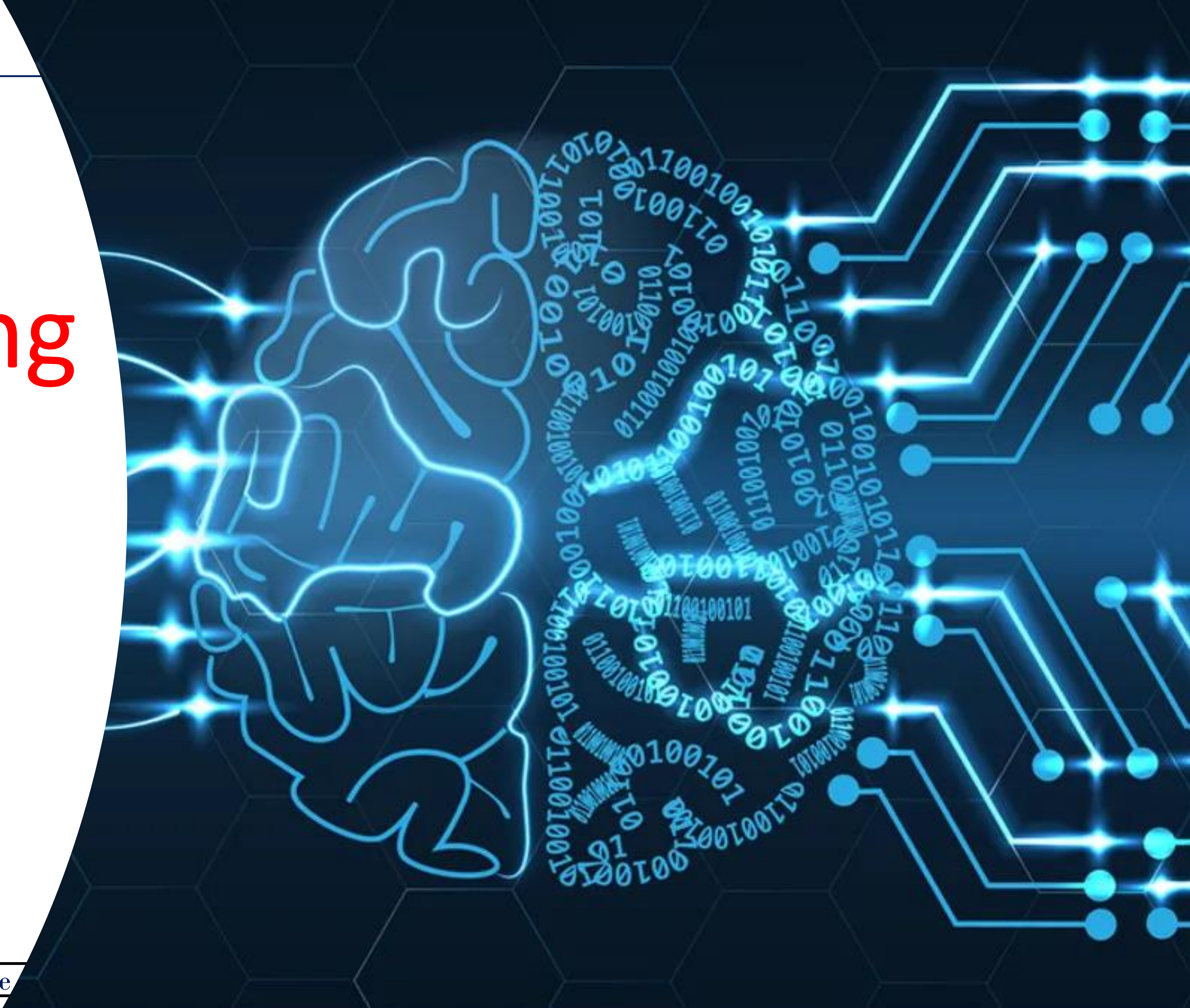
Deep learning

Convolutional Neural Networks

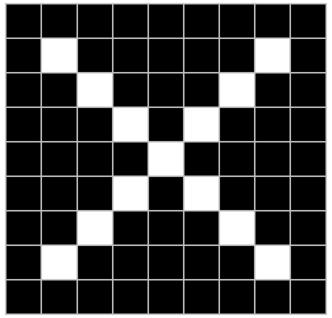
- Lecture

Basics of convolution operation

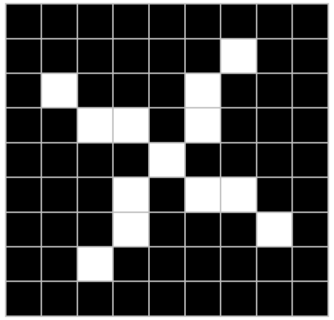


Computer and Human Interpretation

Human Interpretation



=



Computer Interpretation

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

=

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

Computer Interpretation

Pixel wise Matching →

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1
-1	X	X	-1	-1	X	X	-1	-1
-1	-1	X	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	X	-1	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Decision →

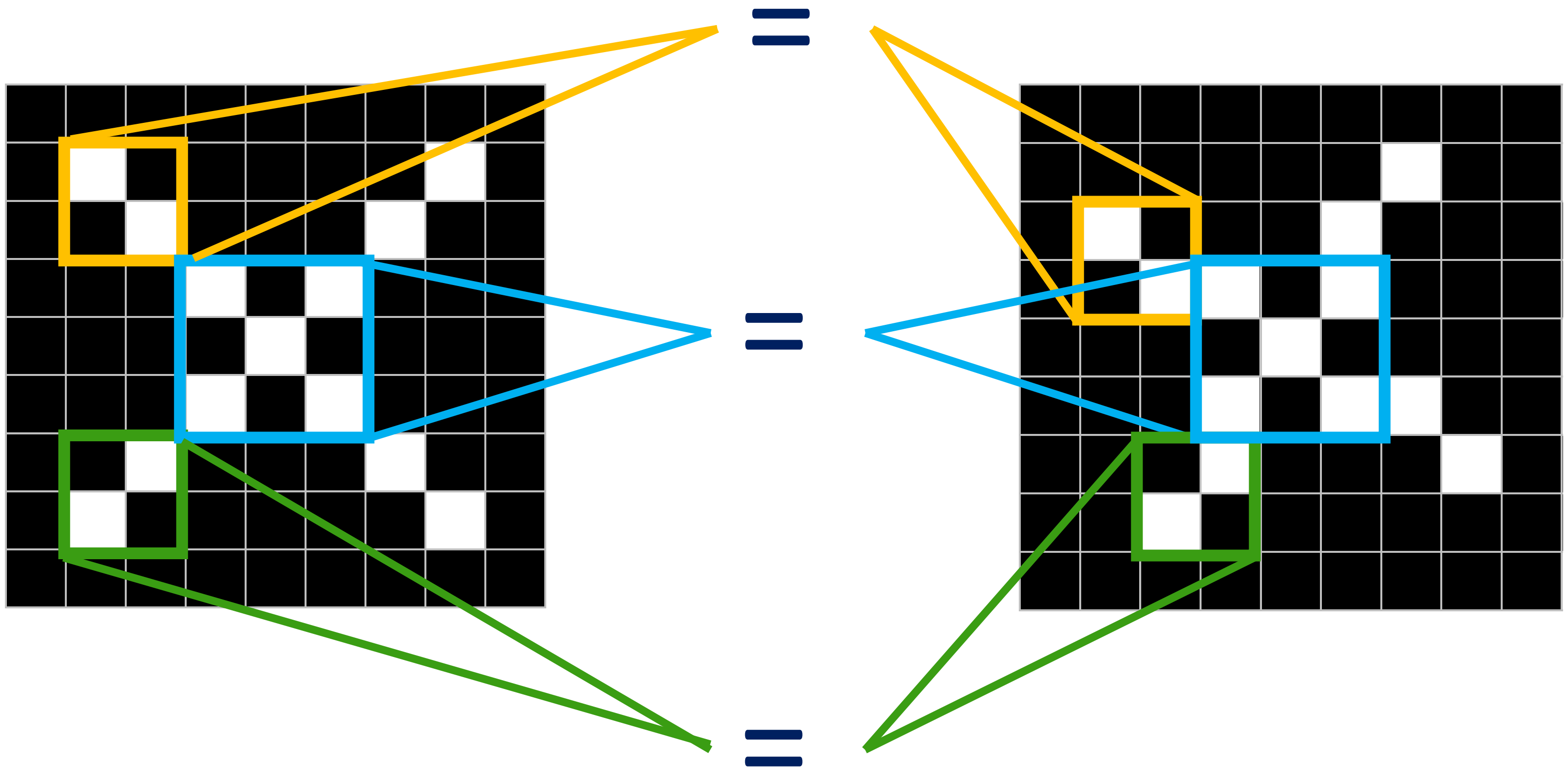
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

$\neq$

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

Computers are Literal

Feature matching for symbol 'X'



Convolution Operation

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

$\boxed{1} \times \boxed{1} = 1$

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

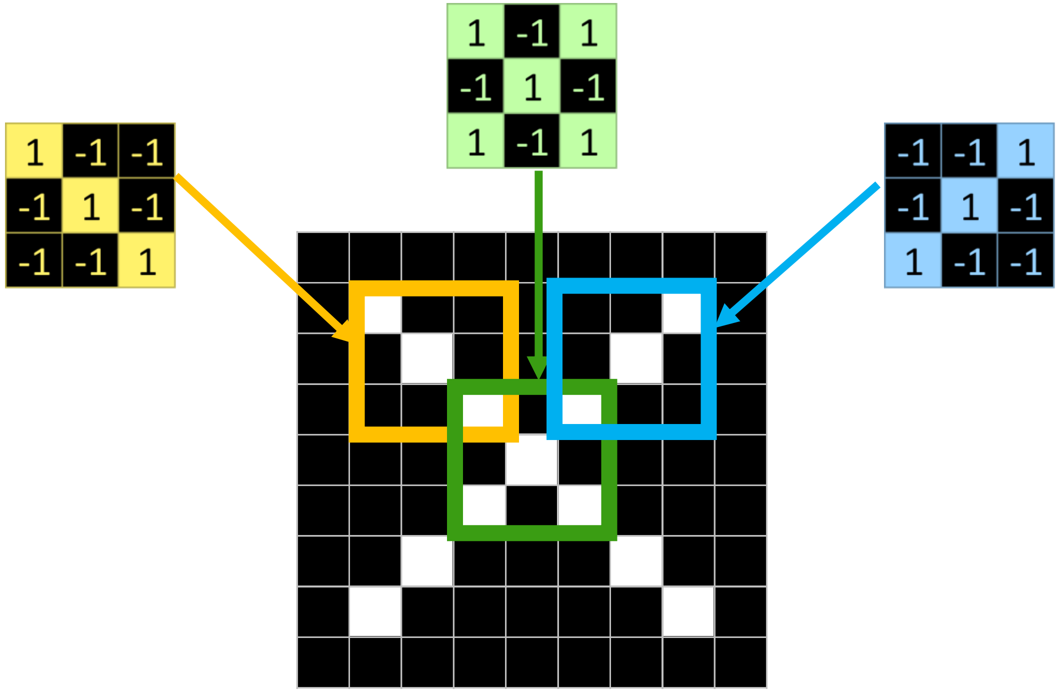
Piece Matching of Features

Features

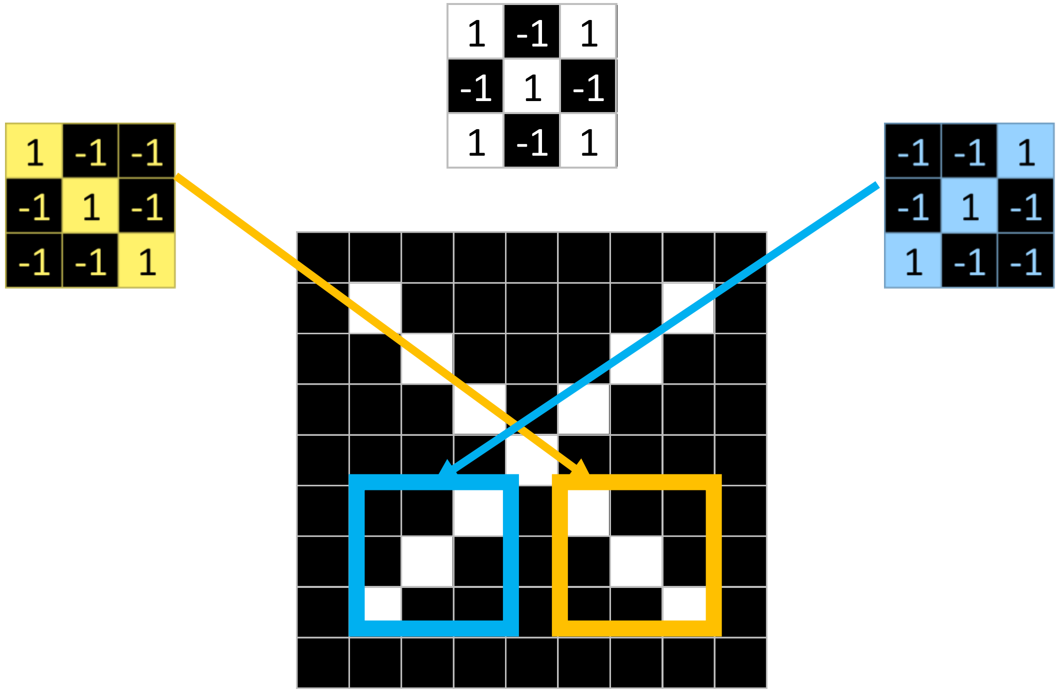
1	-1	-1
-1	1	-1
-1	-1	1

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

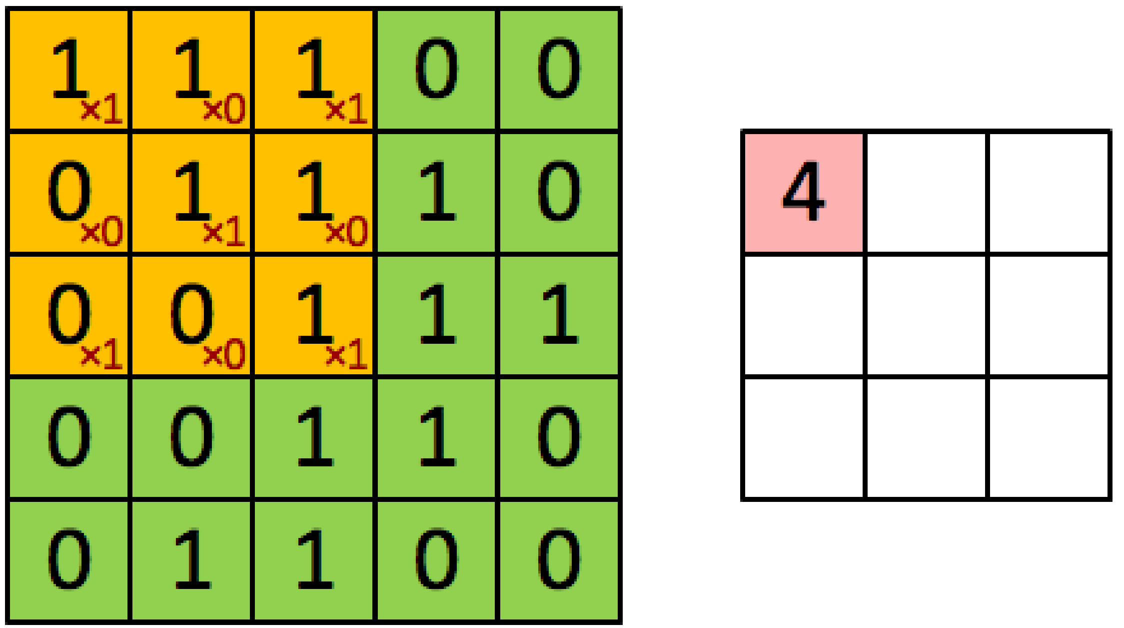
-1	-1	1
-1	1	-1
1	-1	-1



Piece Matching of Features



Convolution Operation



Image

Convolved Feature

Convolution Operation

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

$\boxed{1} \times \boxed{1} = 1$

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

1		

Convolution Operation

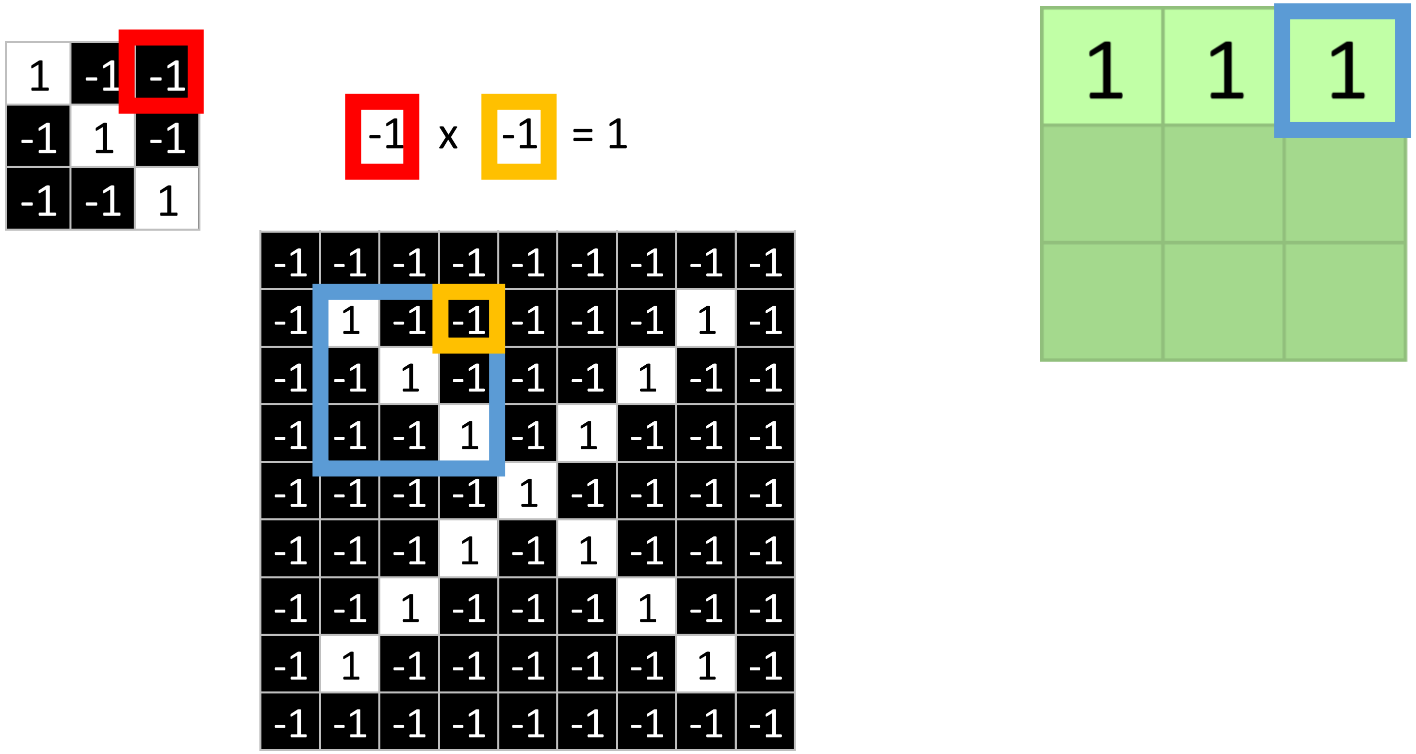
1	-1	-1
-1	1	-1
-1	-1	1

-1 x -1 = 1

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

1	1	

Convolution Operation



Convolution Operation

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

-1 x -1 = 1

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

1	1	1
1		

Convolution Operation

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

1 x 1 = 1

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

1	1	1
1	1	

Convolution Operation

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

$\boxed{-1} \times \boxed{-1} = 1$

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

1	1	1
1	1	1

Convolution Operation

1	-1	-1
-1	1	-1
$\boxed{-1}$	-1	1

$\boxed{-1} \times \boxed{-1} = 1$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	$\boxed{-1}$	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	1
1	1	1
$\boxed{1}$		

Convolution Operation

1	-1	-1
-1	1	-1
-1	$\boxed{-1}$	1

$\boxed{-1}$ x $\boxed{-1}$ = 1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	$\boxed{-1}$	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	1
1	1	1
1	$\boxed{1}$	

Convolution Operation

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

1 x 1 = 1

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

1	1	1
1	1	1
1	1	1

Convolution Operation

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

$\boxed{1} \times \boxed{1} = 1$

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

1	1	1
1	1	1
1	1	1

Feature Mapping

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

Filter 1

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

Filter 2

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

⋮

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

Feature Mapping

Filter 1

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

Feature Mapping

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

Feature Mapping

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

Feature Mapping

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

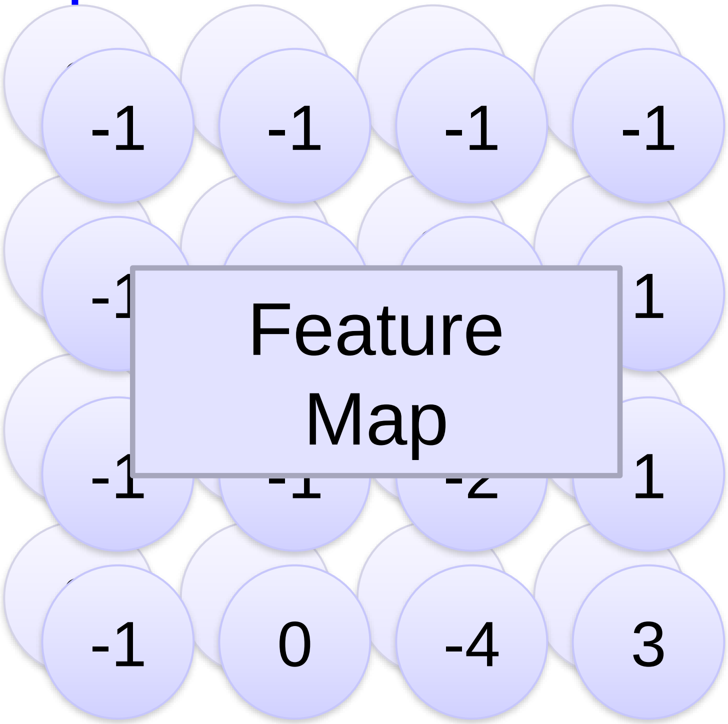
Filter 2

stride=1

Repeat this for each filter

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



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

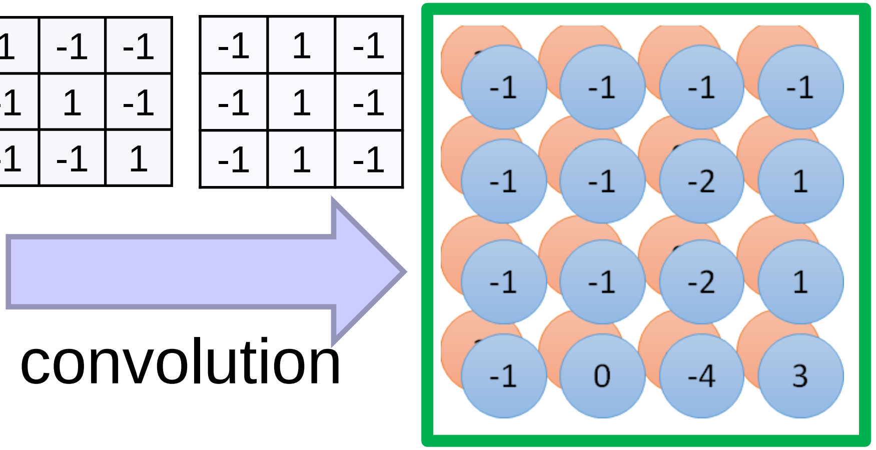
Convolution v.s. Fully Connected

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

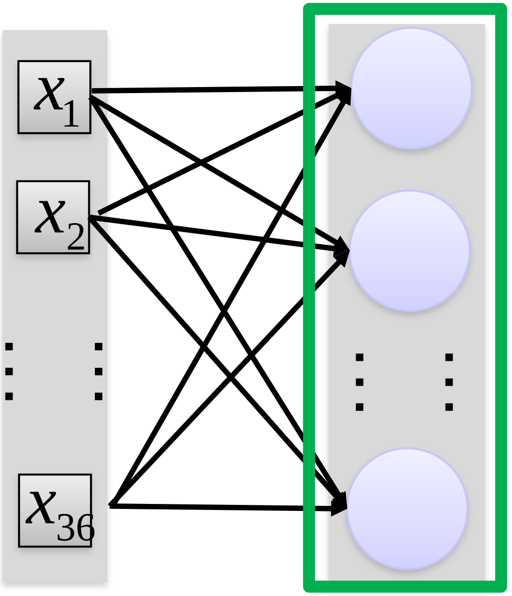
1	-1	-1
-1	1	-1
-1	-1	1

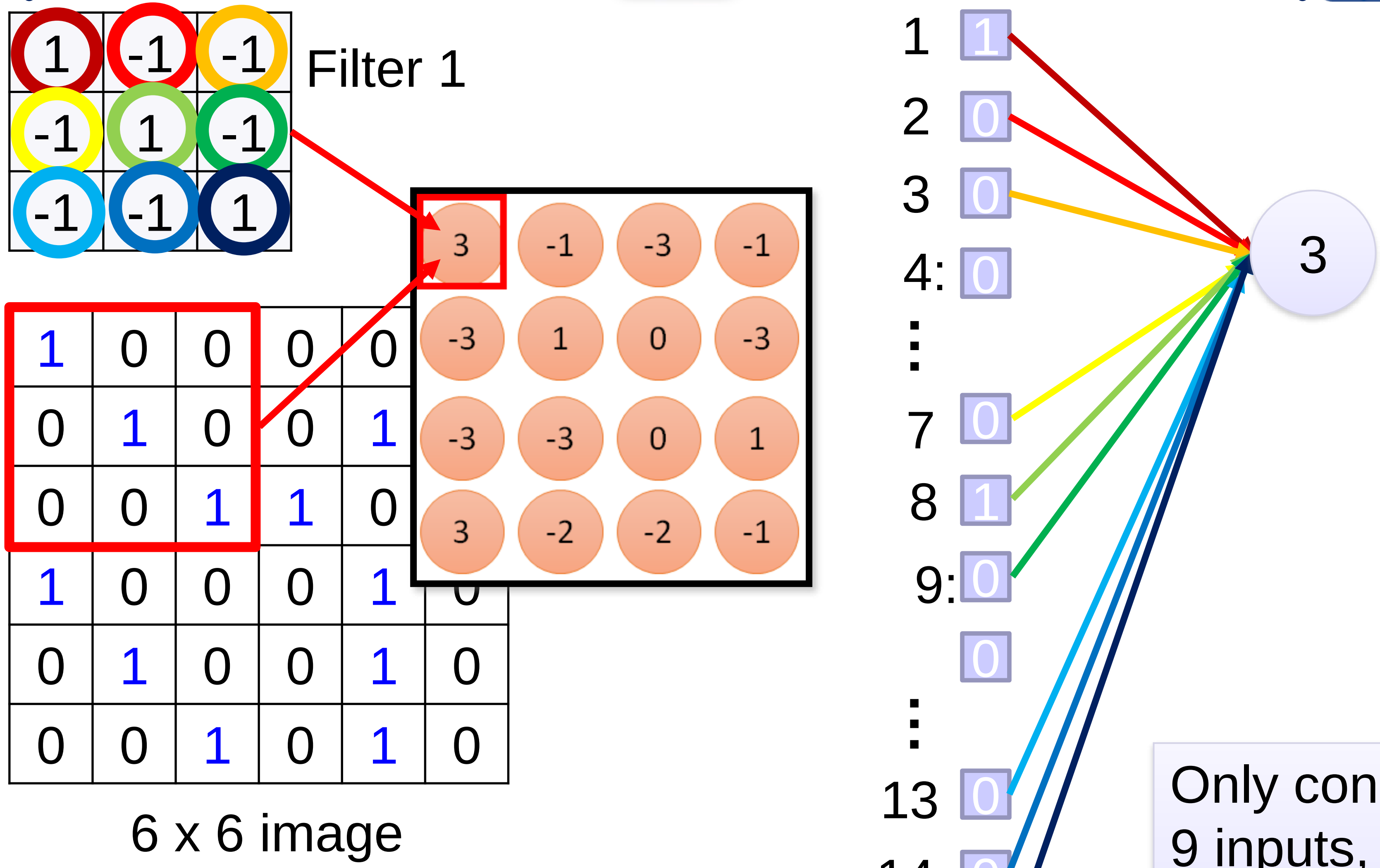
-1	1	-1
-1	1	-1
-1	1	-1



Fully-connected

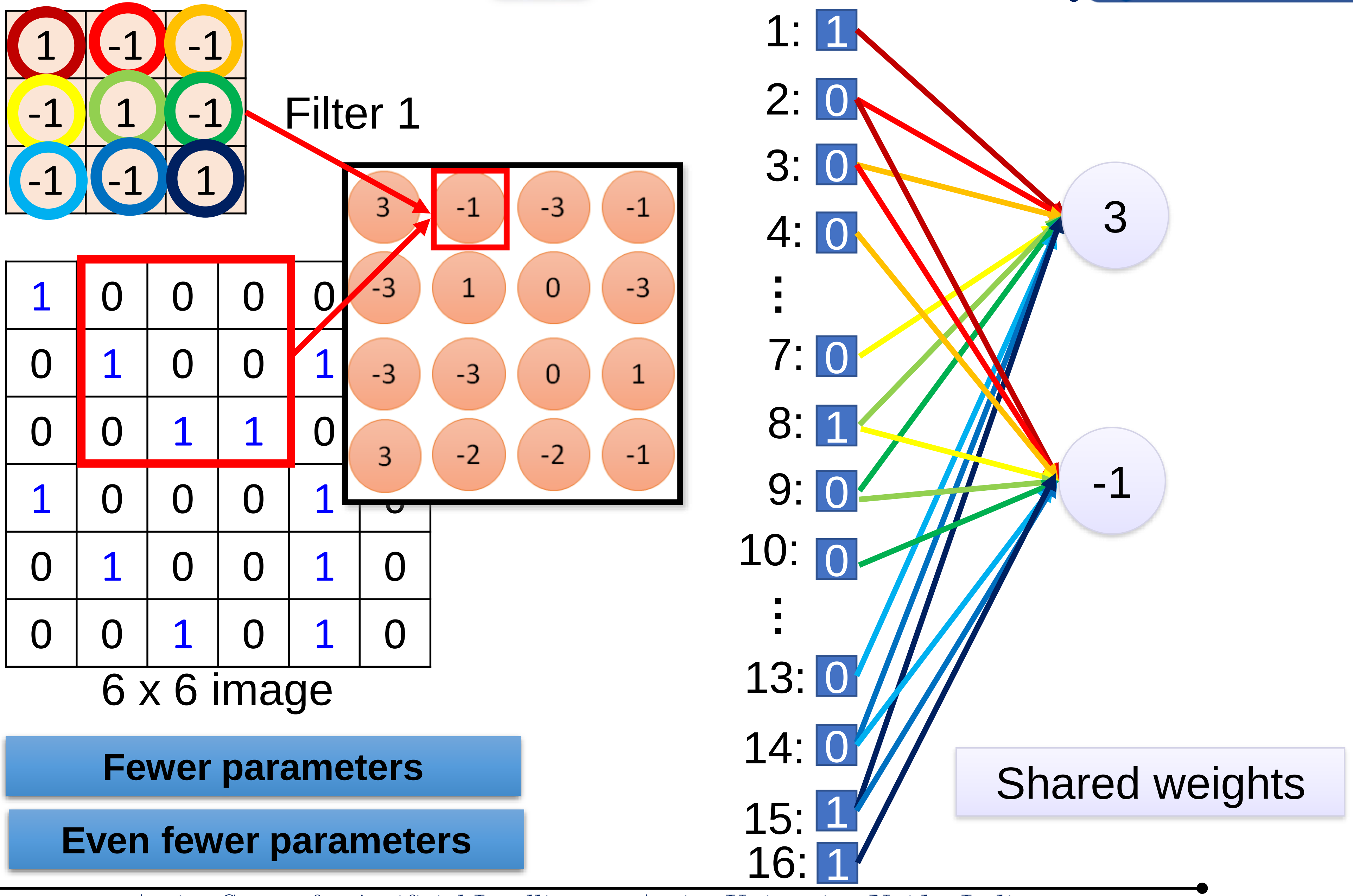
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0



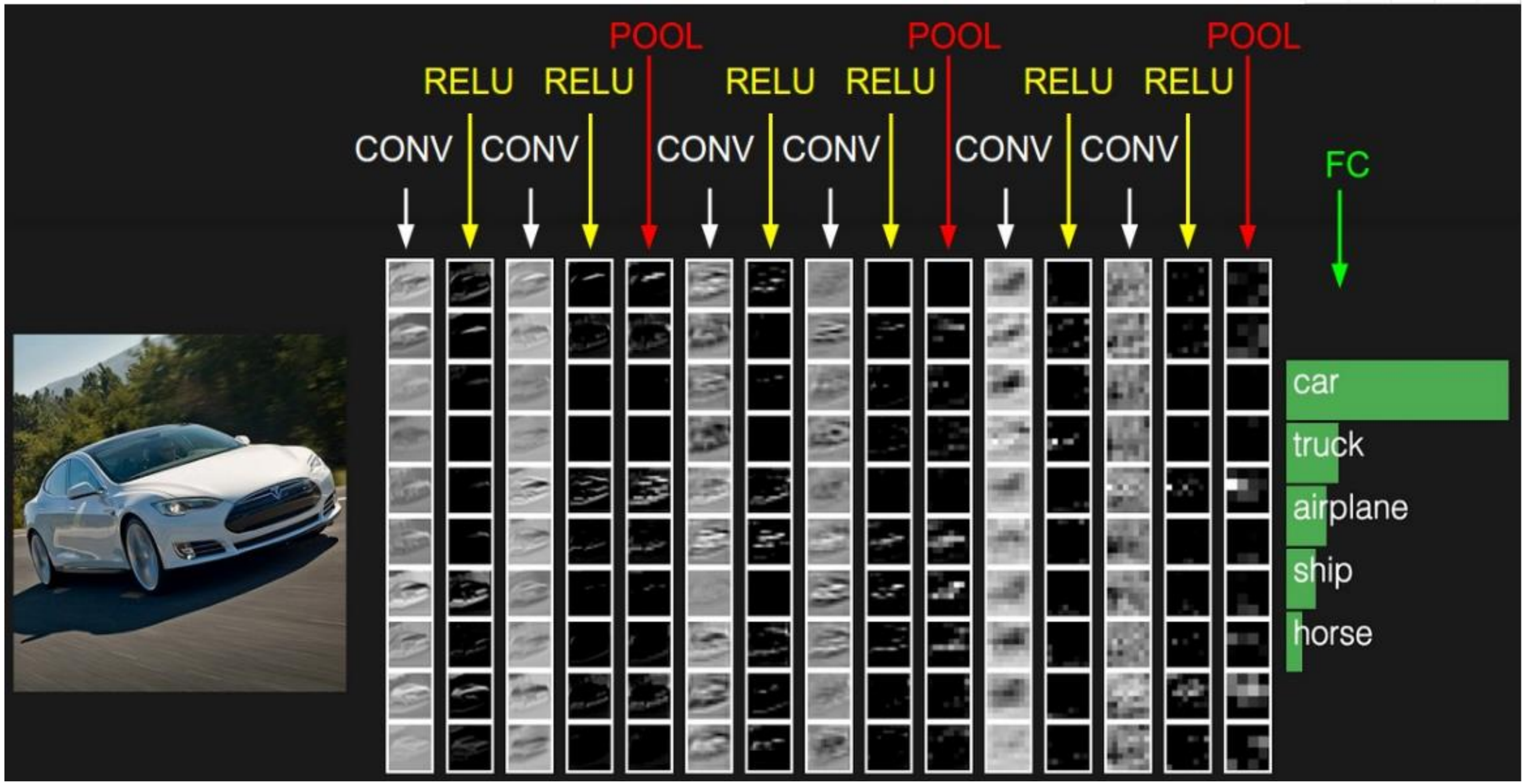


fewer parameters!

Only connect to 9 inputs, not fully connected

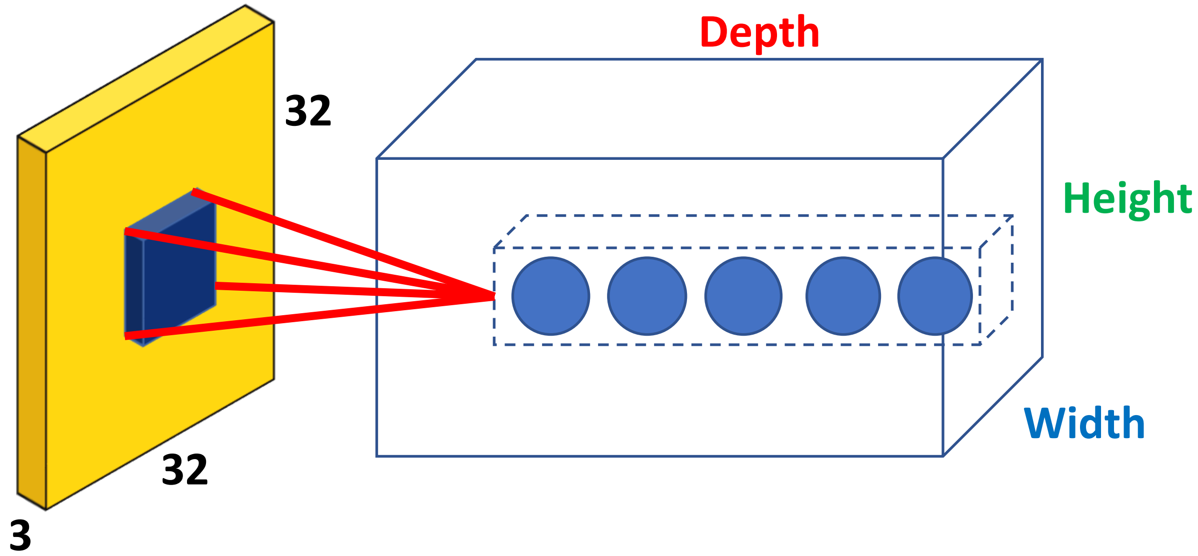

Filter 1
6 x 6 image
Fewer parameters
Even fewer parameters
Shared weights
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
-1

Simple CNN architecture



CONV: Convolutional kernel layer
RELU: Activation function
POOL: Dimension reduction layer
FC: Fully connection layer

Convolutional Neural Network--- Spatial View

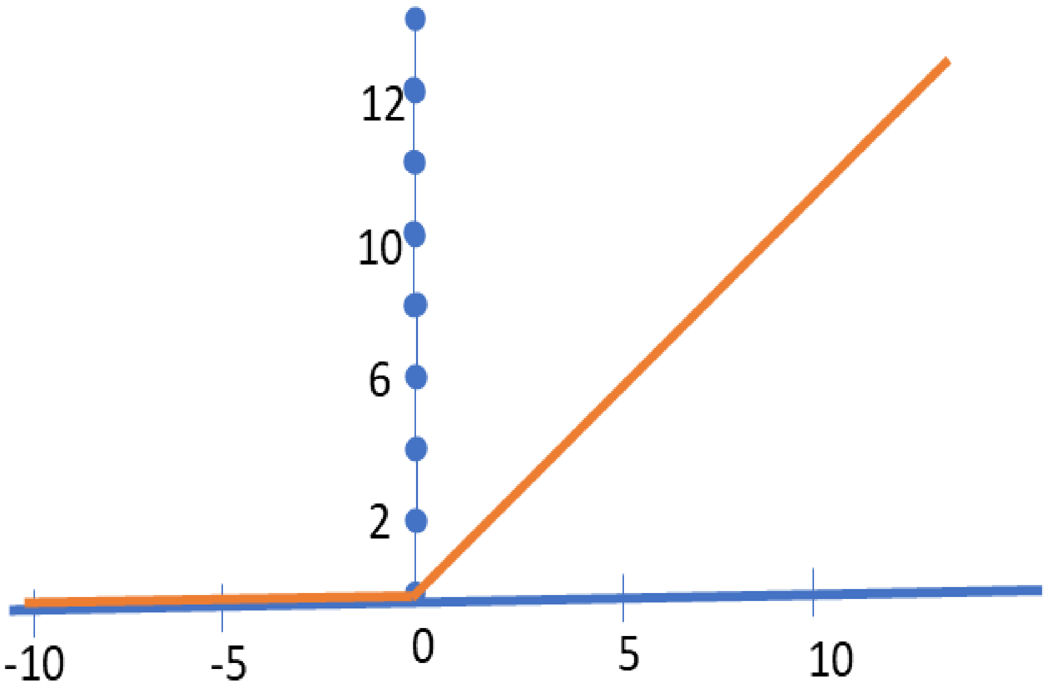


Dimensions of Layer
H * W * D

H (height) and W (width) are spatial dimensions whereas D (depth) is number of filters

Stride = Step size of filter, Receptive Field = Location of connected path in an input image

Non-Linearity in Convolutional Neural Network



Applied after every convolutional layer. The rectified linear activation (ReLU) function is a simple calculation that returns the value provided as input directly, or the value 0.0 if the input is 0.0 or less.

$$\boxed{g(x) = \max(0, x)}$$

Rectified Linear Units (ReLUs)

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0					

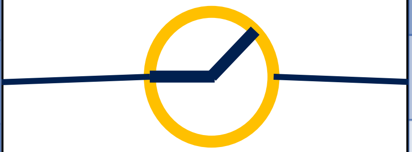
Rectified Linear Units (ReLUs)

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33

Rectified Linear Units (ReLUs)

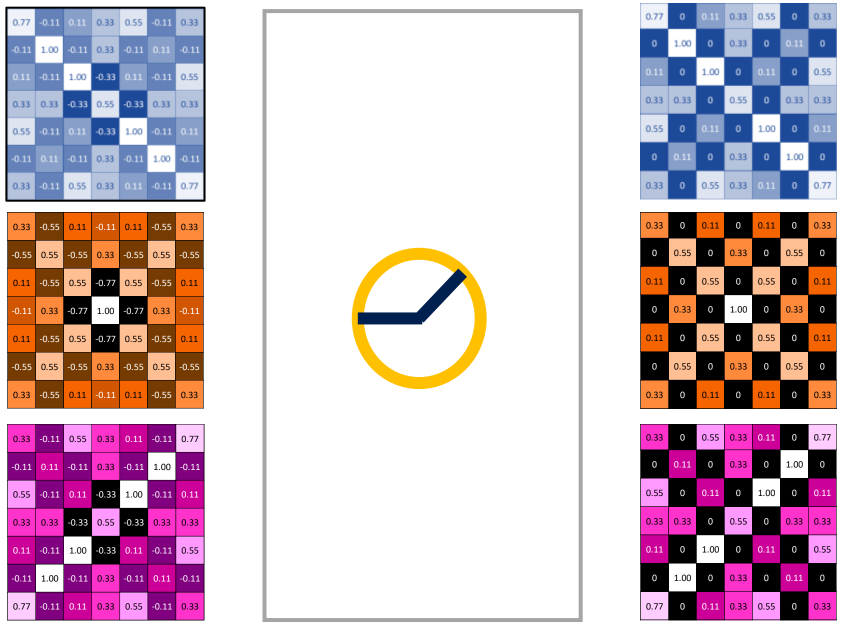
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Rectified Linear Units (ReLUs)

A stack of images becomes a stack of images with no negative values.



Pooling

- *Dimensionality Reduction*
- *Preserve Spatial Invariance*

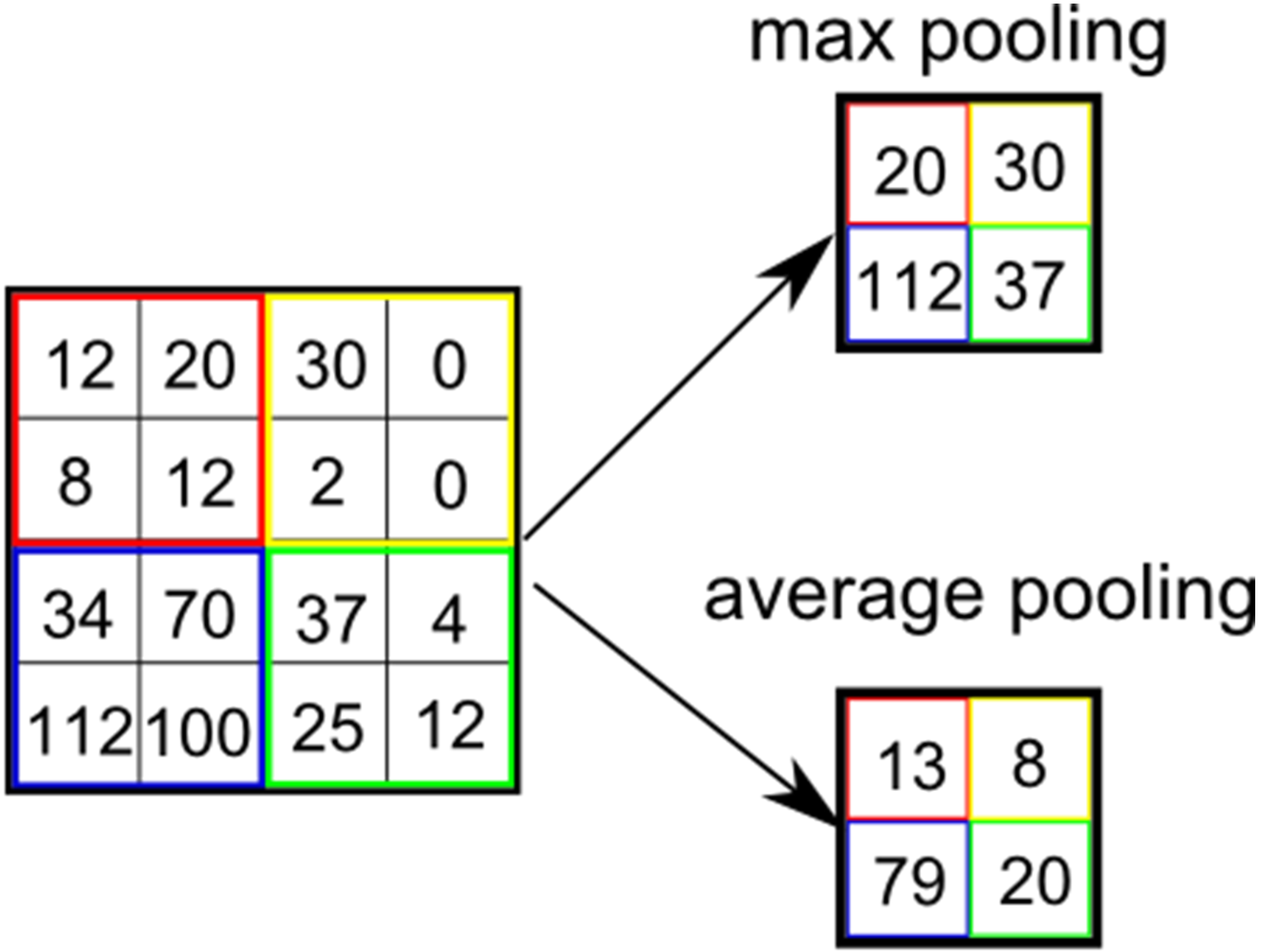
The types of pooling operations are:

- ❑ Max pooling: The maximum pixel value of the batch is selected.
- ❑ Average pooling: The average value of all the pixels in the batch is selected.

STEPS

1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually 2).
3. Walk your window across your filtered images.
4. From each window, take the maximum value.

Pooling


max pooling
average pooling
12
20
30
0
8
12
2
0
34
70
37
4
112
100
25
12
20
30
112
37
13
8
79
20

Why Pooling ?

- Subsampling pixels will not change the object

bird

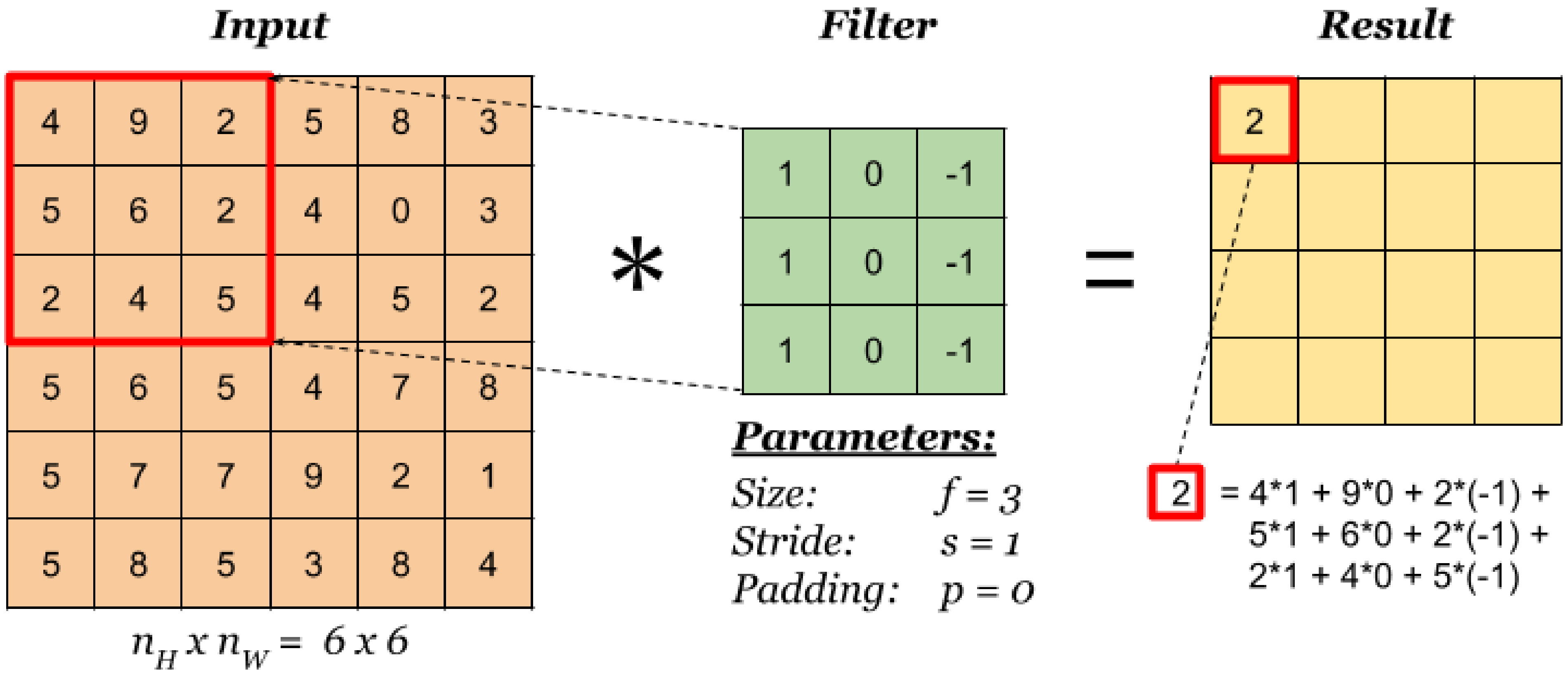
Subsampling

bird

We can subsample the pixels to make image smaller fewer parameters to characterize the image

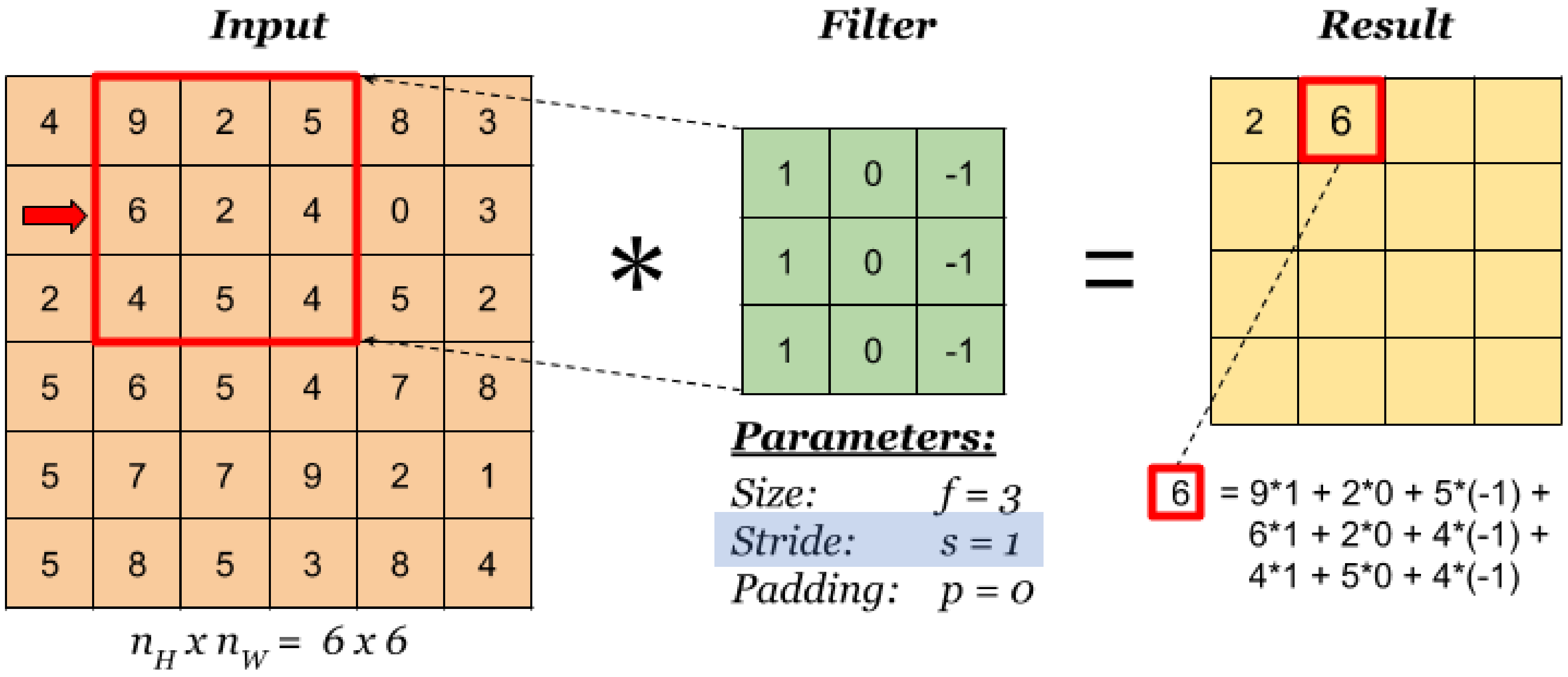
Stride

- number of cells the filter is moved to calculate the next output
- sample only every s pixels in each direction in the output



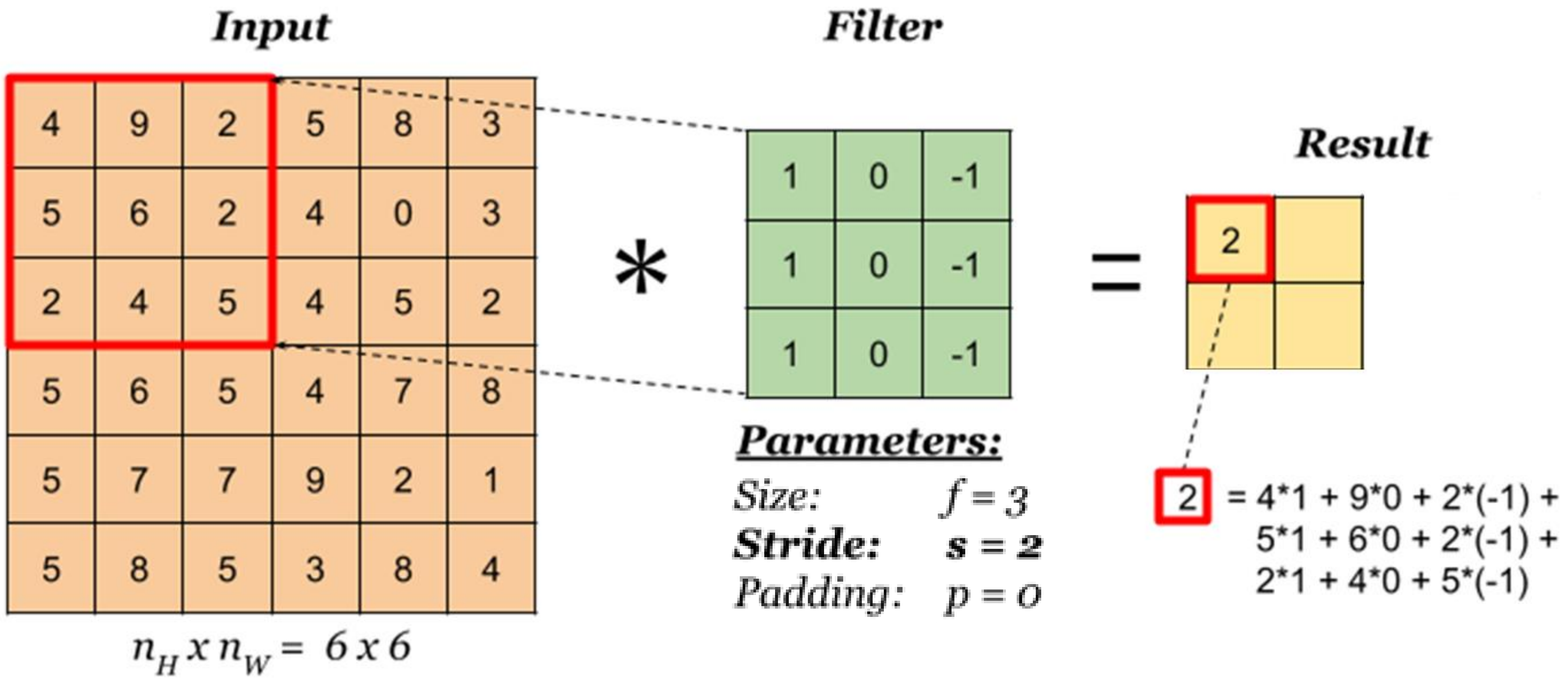
Stride

- number of cells the filter is moved to calculate the next output
- sample only every s pixels in each direction in the output



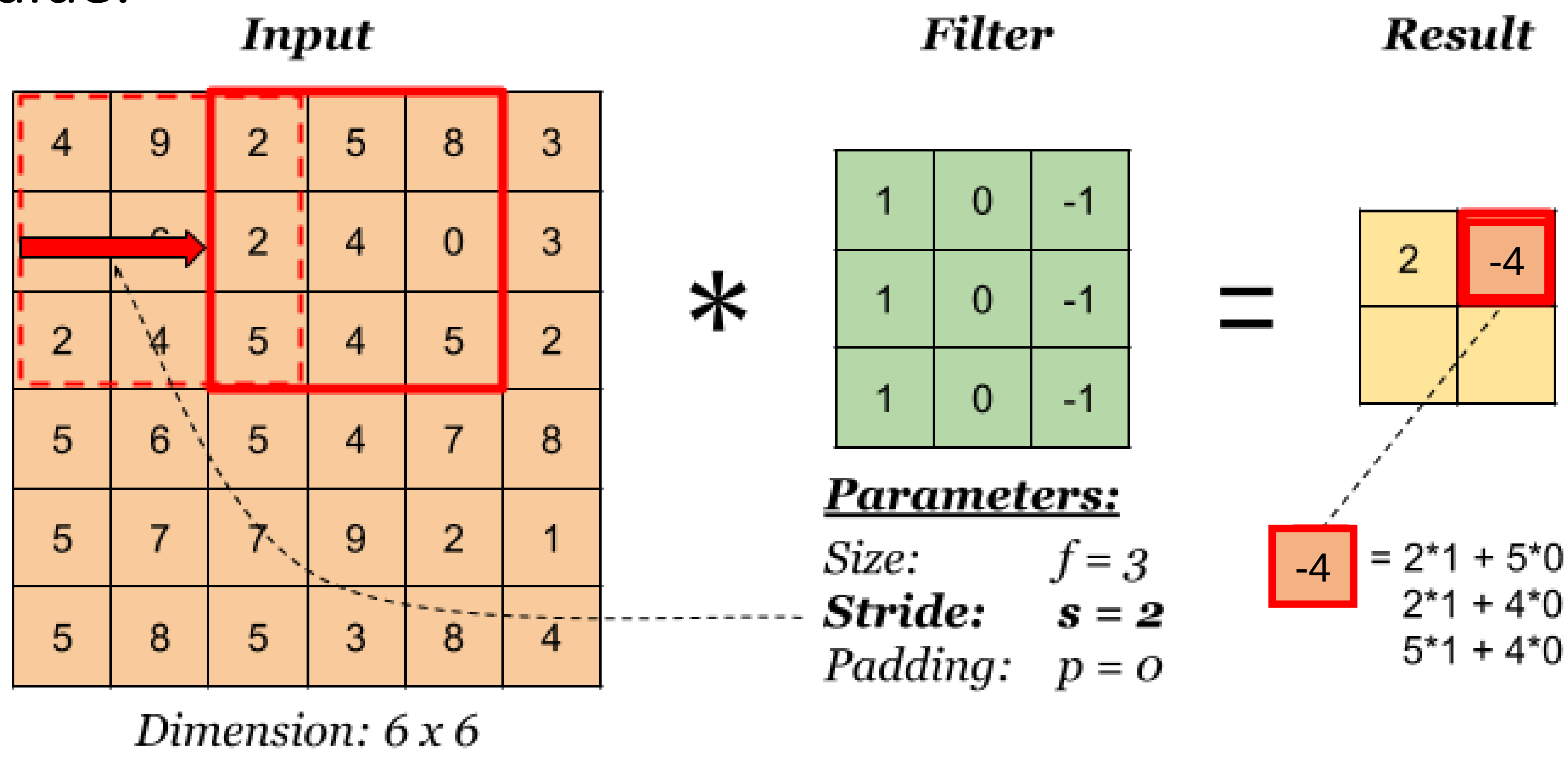
Stride

- Stride = 2
- First Value:



Stride

- Stride = 2
- Next Value:

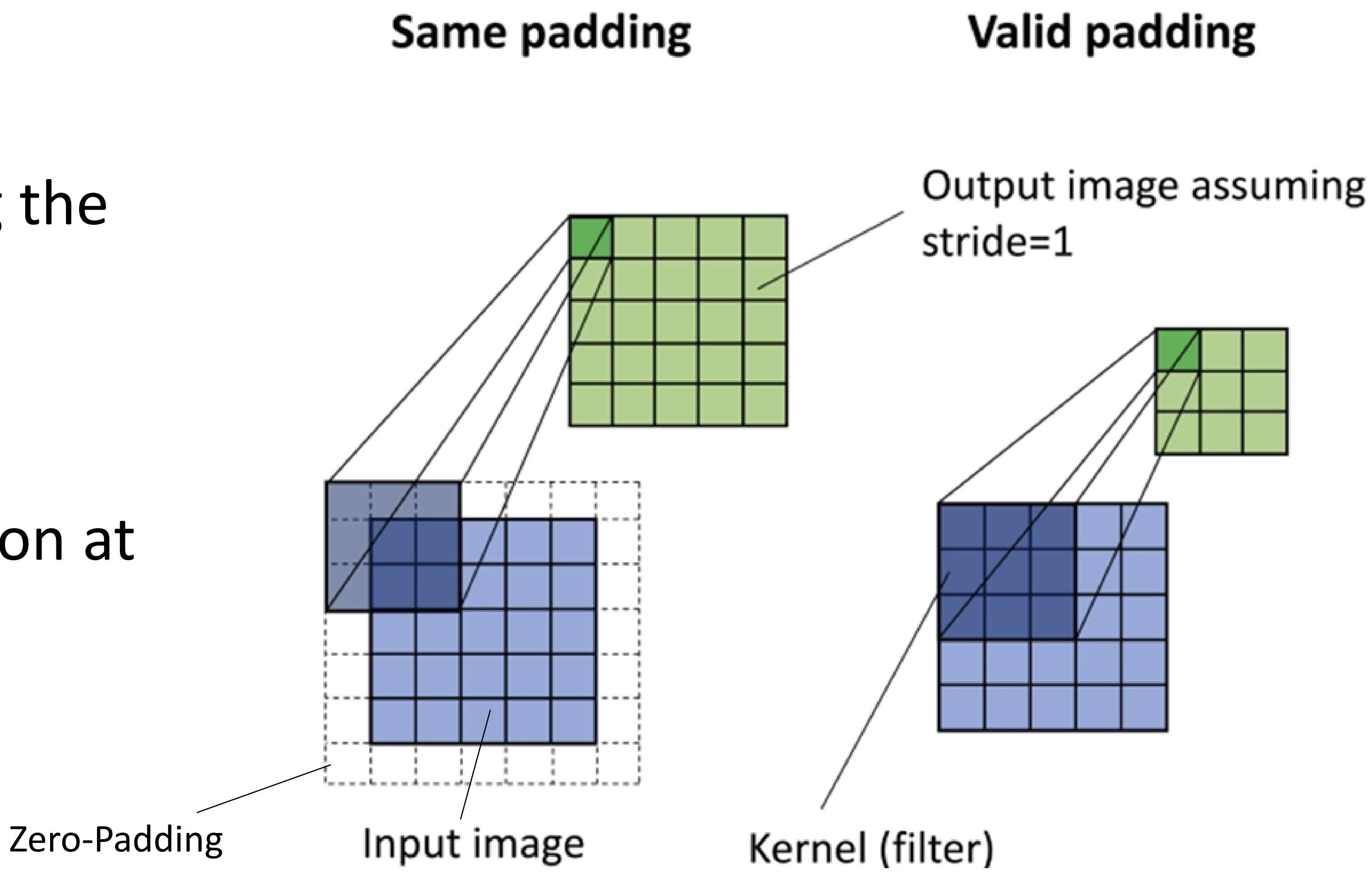


= 2*1 + 5*0 + 8*(-1) +
2*1 + 4*0 + 0*(-1) +
5*1 + 4*0 + 5*(-1)

Size of output feature map may decrease

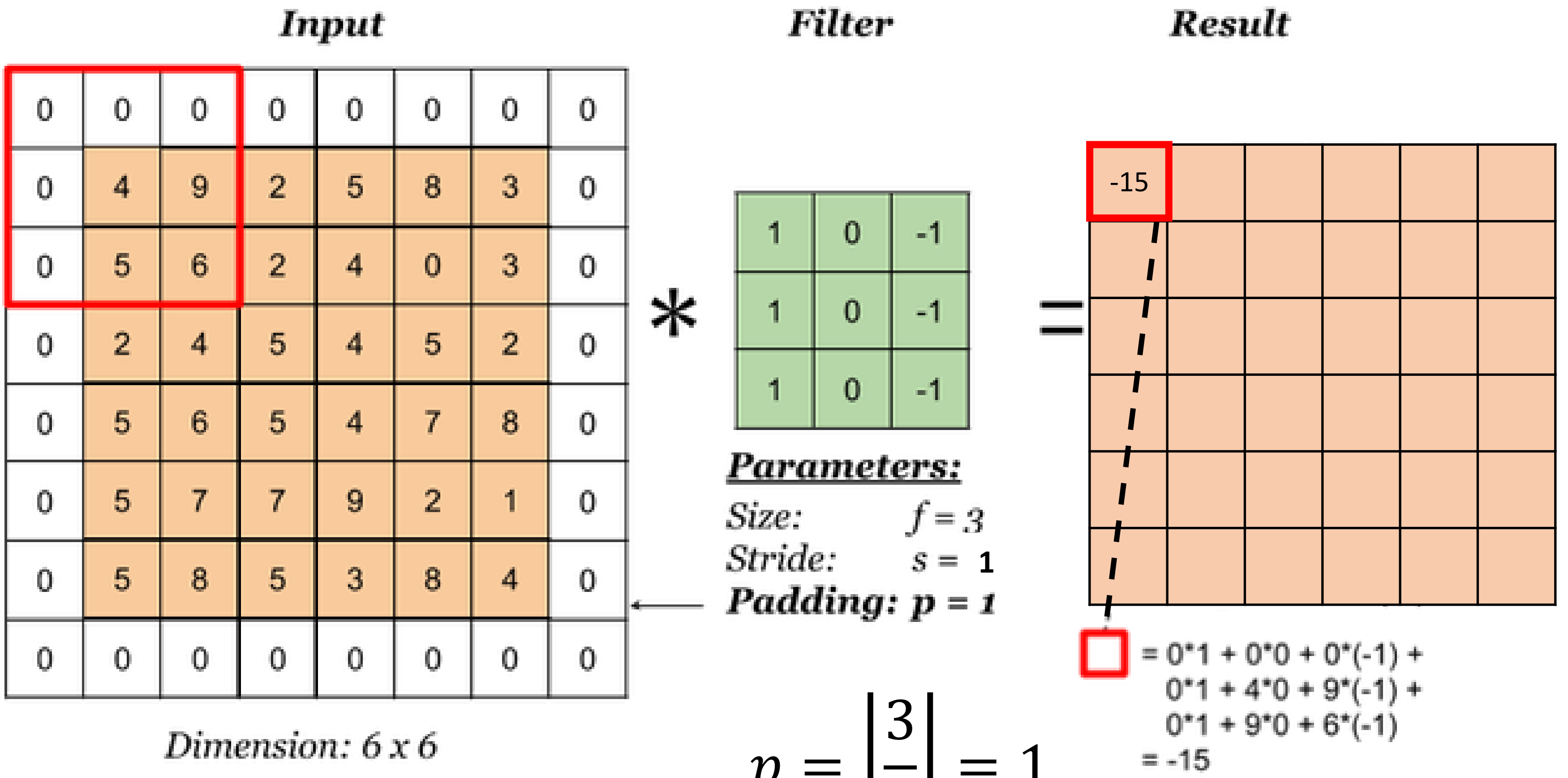
Padding

- Use Conv without shrinking the height and width
- Helpful in building deeper networks
- Keep more of the information at the border of an image



Same Padding

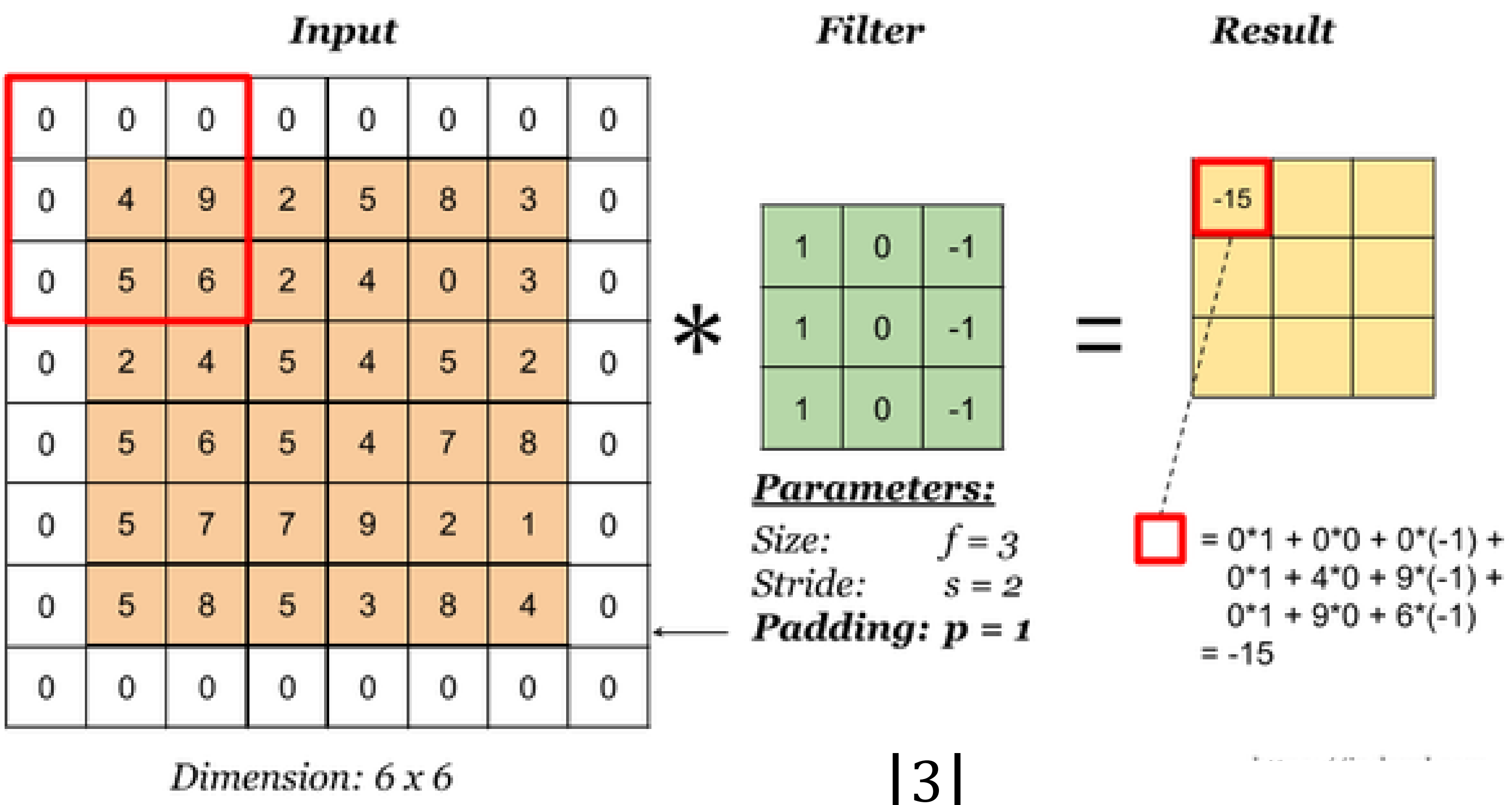
- Buffers the edge of the input with ⌊filter_size/2⌋ zeros (integer division)
- Output dimension is the same as the input for s=1



$$p = \left\lfloor \frac{3}{2} \right\rfloor = 1$$

Same Padding

- Buffers the edge of the input with ⌊filter_size/2⌋ zeros (integer division)
- Output dimension is the same as the input for s=1
- Output dimension reduces less for s>1



$$p = \left\lfloor \frac{3}{2} \right\rfloor = 1$$

HOW TO CALCULATE THE OUTPUT SIZE OF A CONVOLUTIONAL LAYER

To determine the output size of the convolution, the following equation can be applied:

$$n_{out} = \frac{n_{in} + 2p - k}{s} + 1$$

Putting it all together

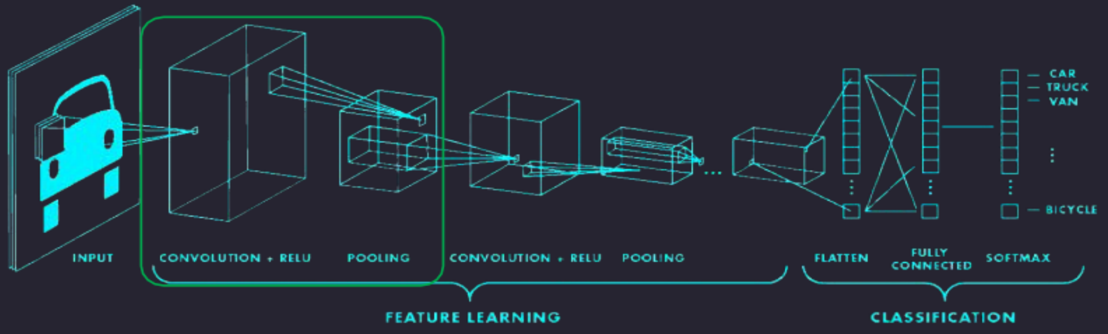
```
import tensorflow as tf

def generate_model():
    model = tf.keras.Sequential([
        # first convolutional layer
        tf.keras.layers.Conv2D(32, filter_size=3, activation='relu’),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

        # second convolutional layer
        tf.keras.layers.Conv2D(64, filter_size=3, activation='relu’),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

       # fully connected classifier
       tf.keras.layers.Flatten(),
       tf.keras.layers.Dense(1024, activation='relu’),
       tf.keras.layers.Dense(10, activation=‘softmax’)
       # 10 outputs

    ])
    return model
```

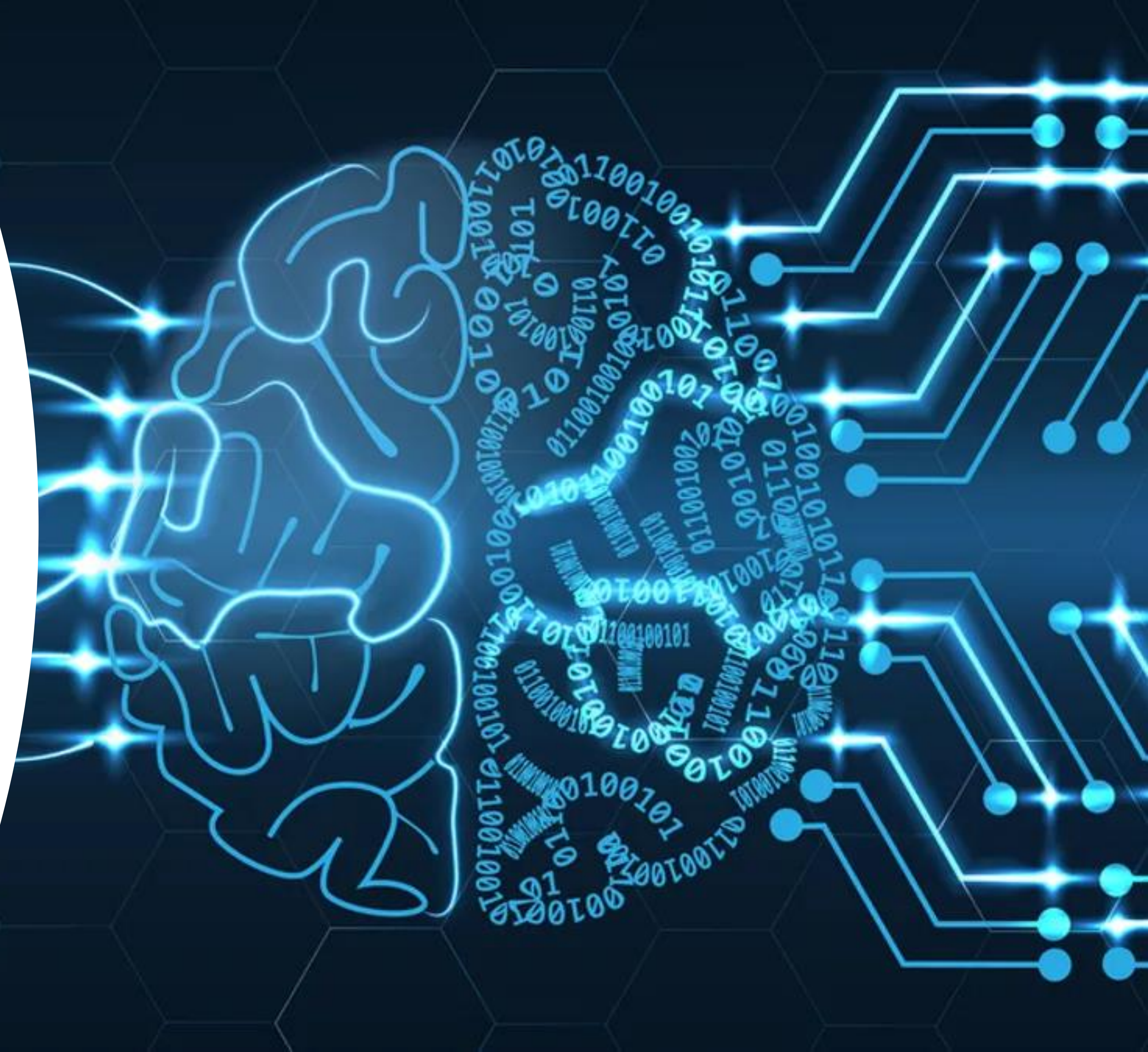


Deep learning

- Course Code:
- Unit 3

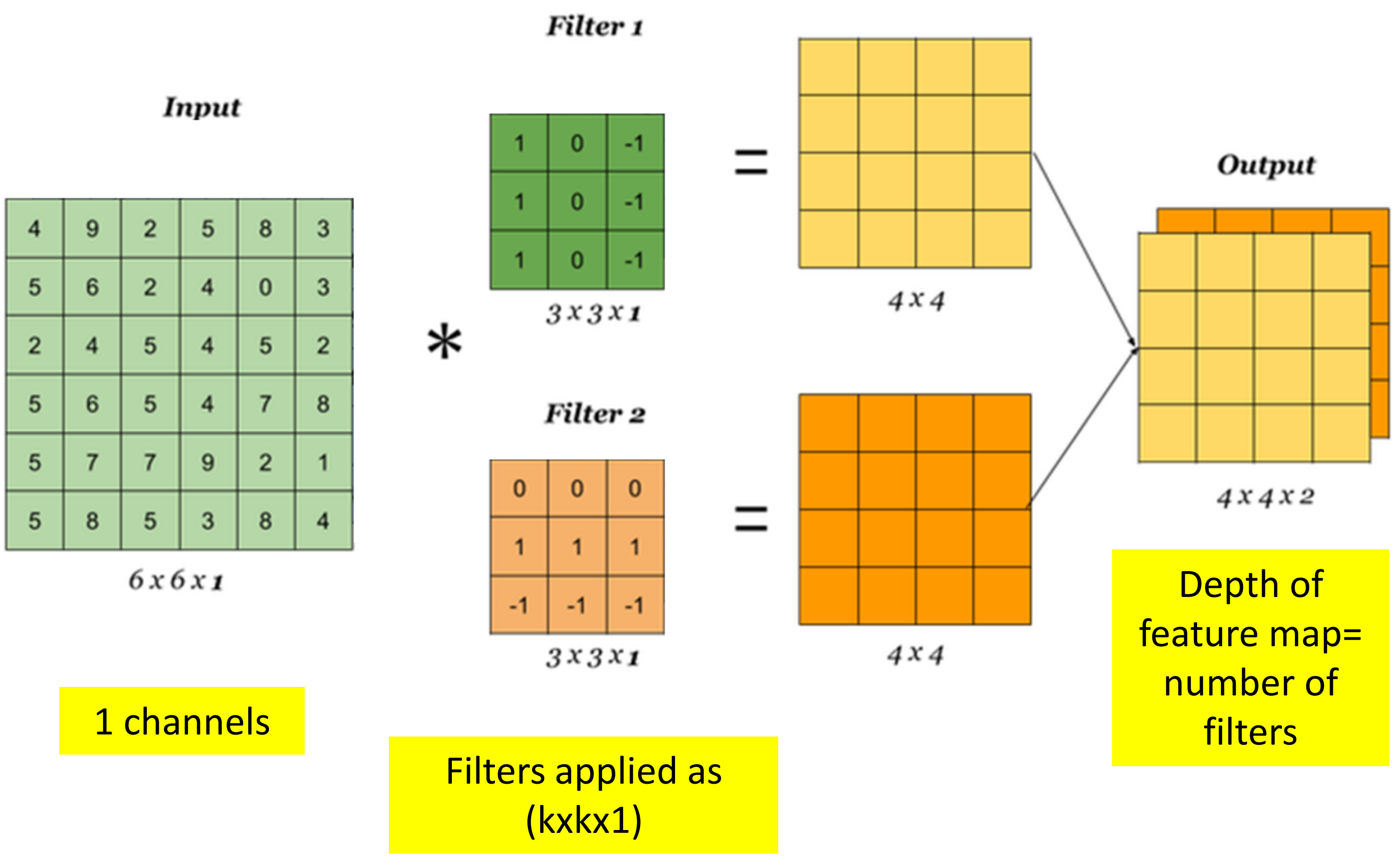
  Convolutional Neural Networks and Transfer Learning
- Lecture 4

  Convolutional Neural Network



Convolution with Single Channel and Multiple Filters

- Input with 1 channel (eg. grayscale image) then a 3×3 filter will be applied in 3x3x1 blocks

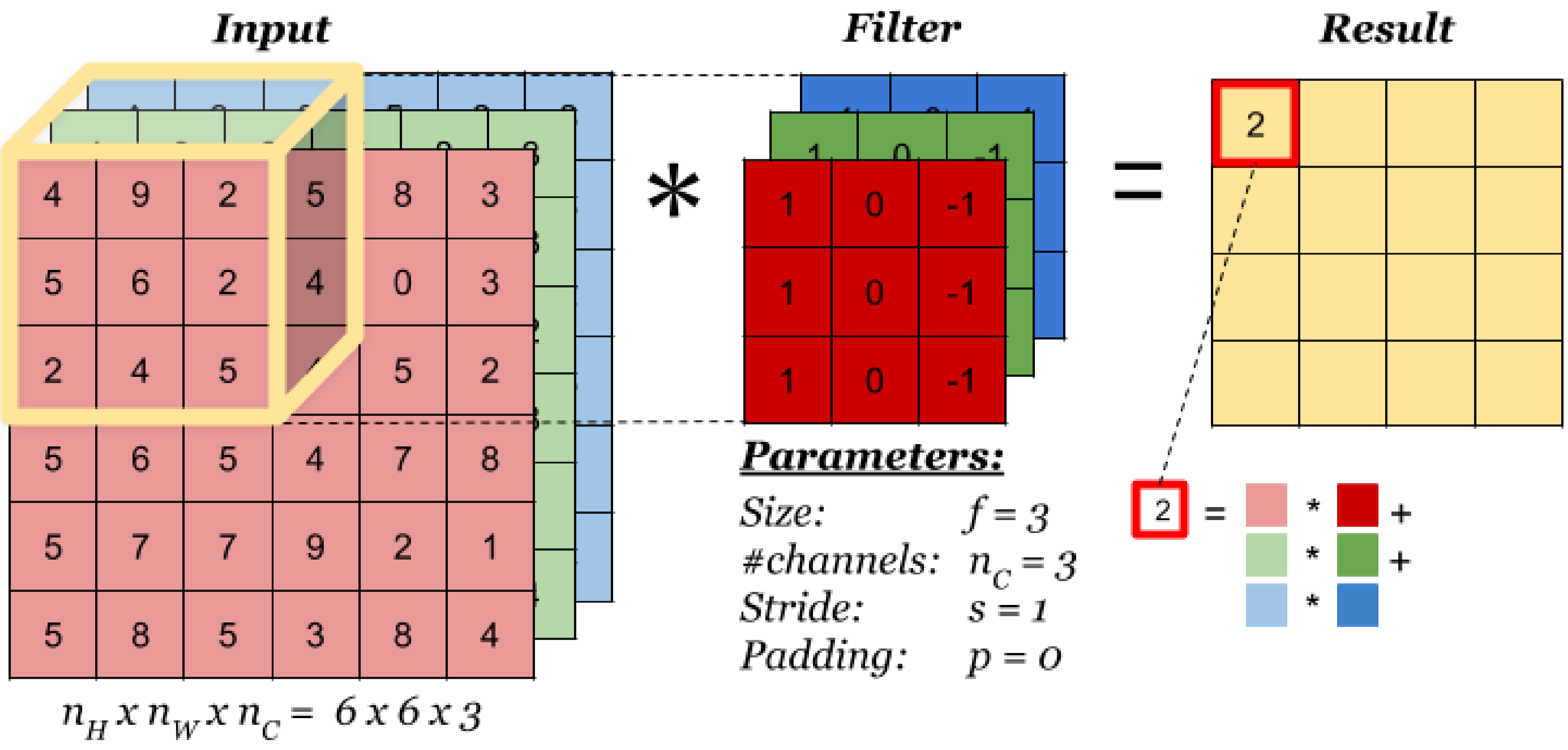


Convolution over Volume (Multiple Channels)

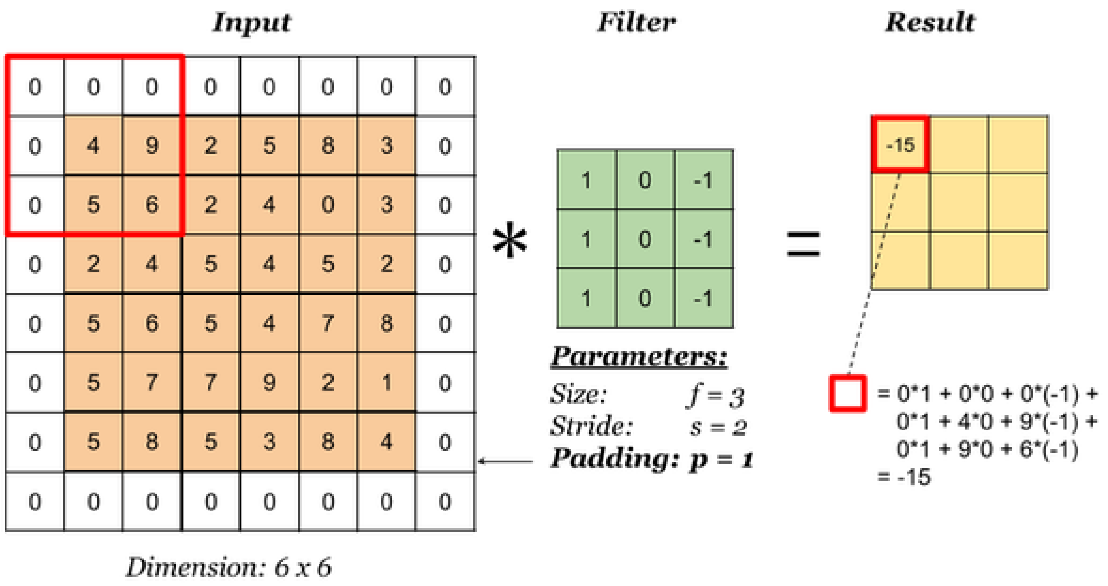
- RGB images has 3 channels:

  Red, Green, Blue

- One kernel for every input channel to the layer (each kernel is unique)
- Each filter = a *collection of kernels*



Output Dimension



$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

- $n_{out} = \left\lfloor \frac{6-3+2*1}{2} \right\rfloor + 1 = \left\lfloor \frac{5}{2} \right\rfloor + 1$
  $= 2 + 1 = 3$

Determine the Output Dimension

- $n_{out} = ?$

Input

4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

$n_H \, x \, n_W = 6 \, x \, 6$

$*$

Filter

1	0	-1
1	0	-1
1	0	-1

$=$

Result

Parameters:

Size: $f = 3$
Stride: $s = 1$
Padding: $p = 0$

Determine the Output Dimension

- $n_{out} = \left\lfloor \frac{6-3+2*0}{1} \right\rfloor + 1$

$= \left\lfloor \frac{3}{1} \right\rfloor + 1$

$= 4$

Input

4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

$n_H \times n_W = 6 \times 6$

$*$

Filter

1	0	-1
1	0	-1
1	0	-1

Parameters:

Size: $f = 3$
Stride: $s = 1$
Padding: $p = 0$

$=$

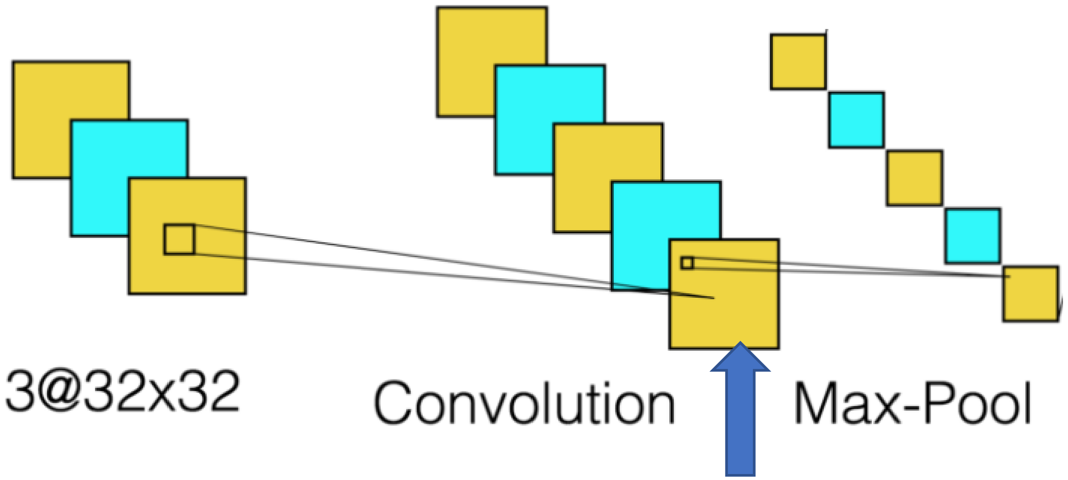
Result

2			

$\boxed{2} = 4*1 + 9*0 + 2*(-1) + 5*1 + 6*0 + 2*(-1) + 2*1 + 4*0 + 5*(-1)$

Determine the Output Dimension

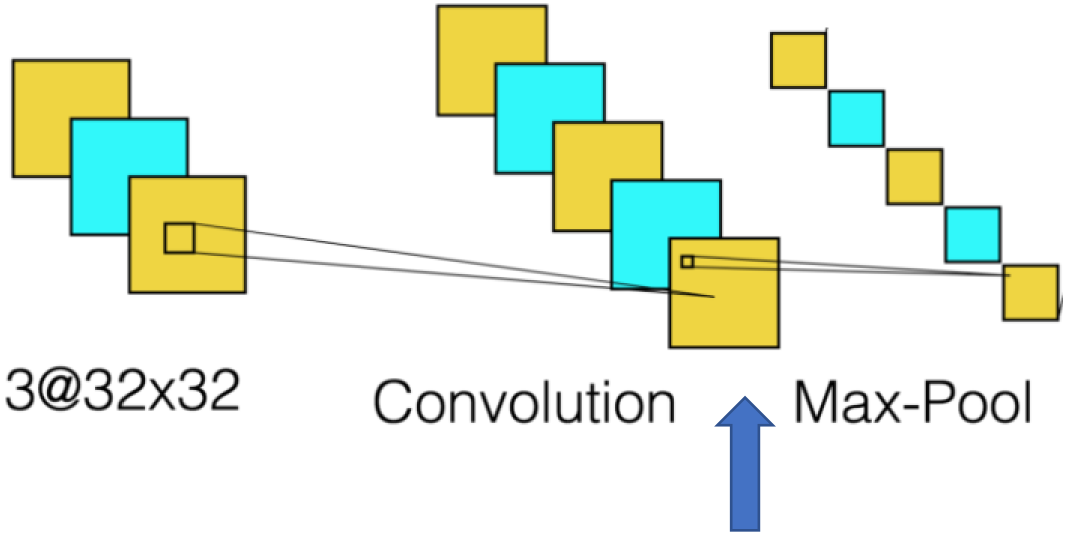
- Input image, I with dimensions (32x32x3)
- Convolution Layer
  - A filter size 3x3
  - Stride is 1
  - Valid padding, and
  - Depth/feature maps are 5 (D =5)
- **Output dimensions = ?**



$$n_{out} =$$

Determine the Output Dimension

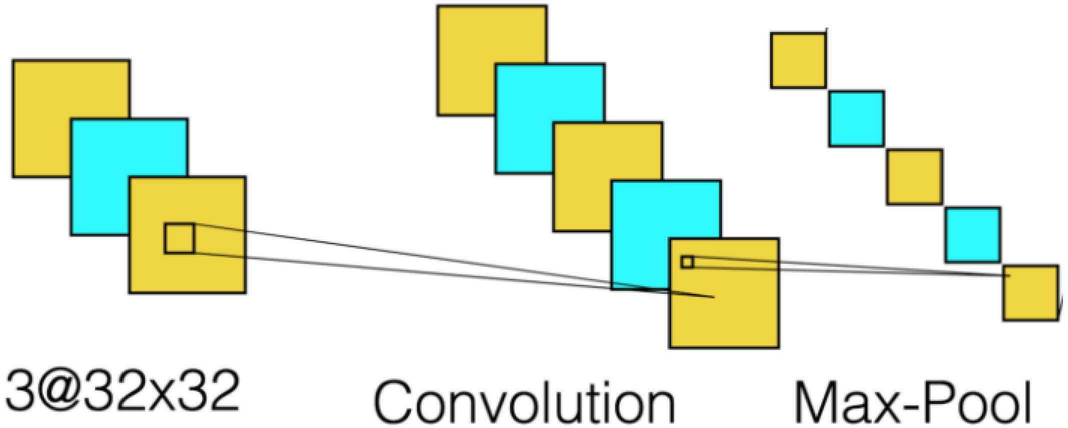
- Input image, I with dimensions (32x32x3)
- Convolution Layer
  - A filter size 3x3
  - Stride is 1 (s=1)
  - Valid padding (p=0), and
  - Depth/feature maps are 5 (D =5)
- **Output dimensions =** 30x30x5
- **After Pooling**?



$$n_{out} = \left\lfloor \frac{32 - 3 + 2 * 0}{1} \right\rfloor + 1 = \left\lfloor \frac{29}{1} \right\rfloor + 1 = 30$$

Determine the Output Dimension

- Input to Pooling Layer (30x30x5)
- After Pooling with
  - Filter size $k \times k$
  - Stride s
- $n_{out} = \left\lfloor \frac{n_{in} - k}{s} \right\rfloor + 1$
- **Eg,** Pooling with, Filter size 2×2, Stride 2
- Output dimensions =

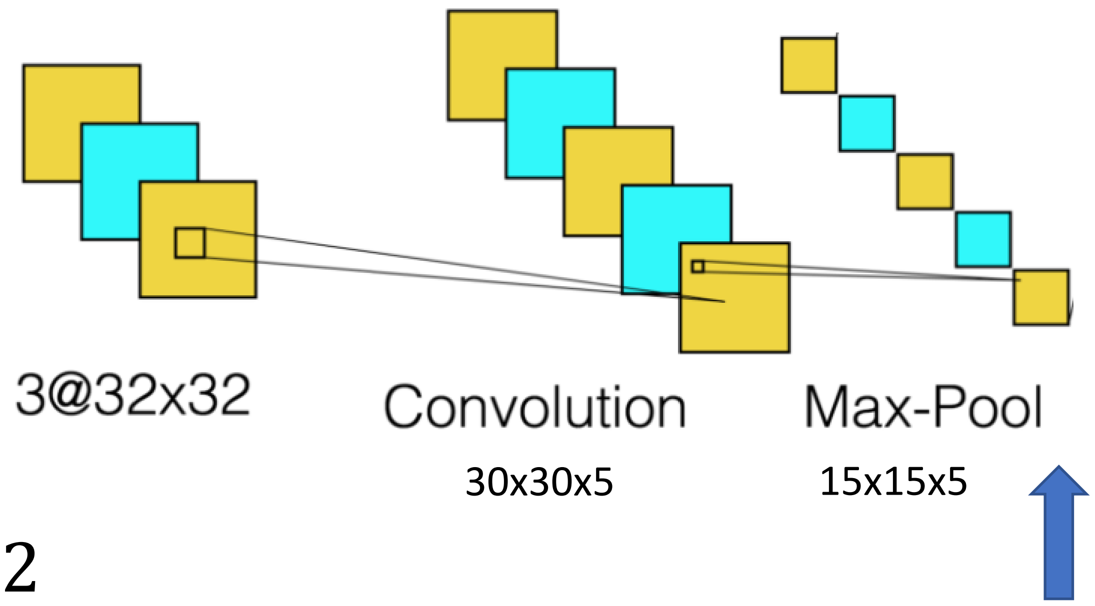


30x30x5

$n_{out} =$

Determine the Output Dimension

- Input to Pooling Layer (30x30x5)
- After Pooling with
  - Filter size $k \times k$
  - Stride s
- $n_{out} = \left\lfloor \frac{n_{in} - k}{s} \right\rfloor + 1$



- **Eg,** Pooling with, Filter size 2×2, Stride 2
- Output dimensions =15x15x5

$$n_{out} = \left\lfloor \frac{30-2}{2} \right\rfloor + 1 = \left\lfloor \frac{28}{2} \right\rfloor + 1 = 15$$

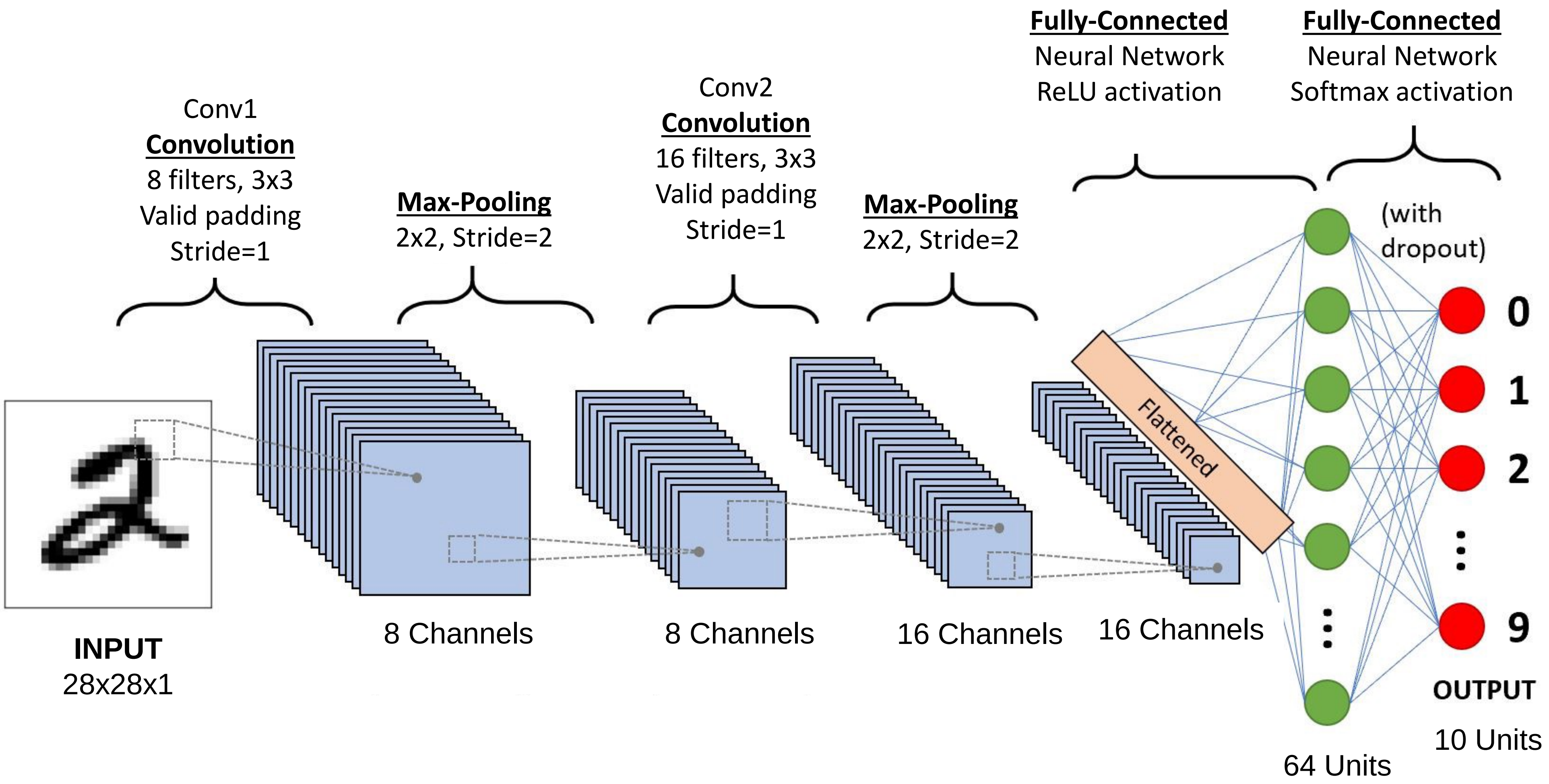
Typical CNN Model

- Conv1

$n_{out} = \left\lfloor \frac{n_{in} - k + 2*p}{s} \right\rfloor + 1$

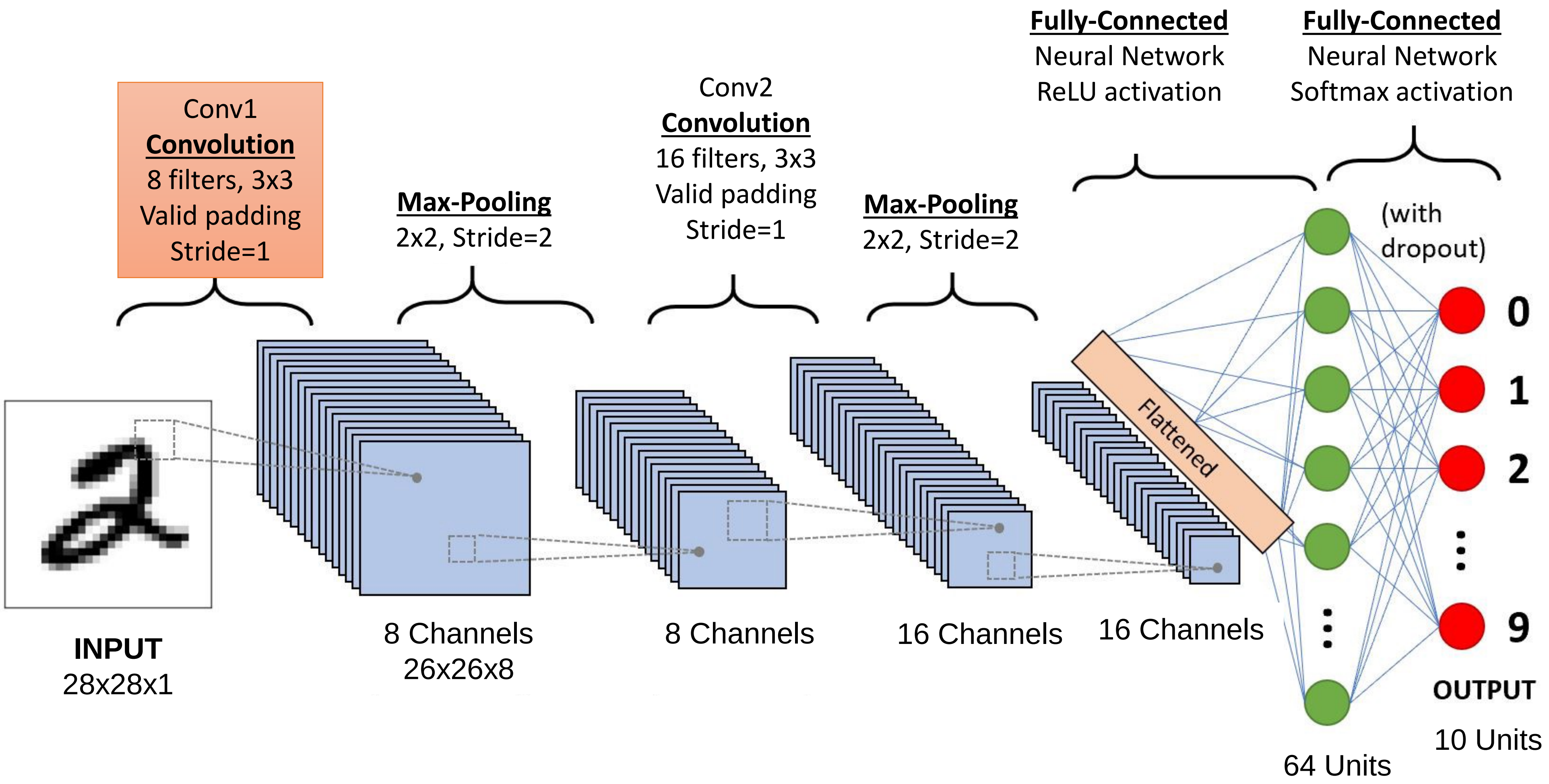
$= ?$

- o/p:



Typical CNN Model

- Conv1 $\left\lfloor \frac{28-3+2*0}{1} \right\rfloor + 1 = 26$
- o/p: $\mathbf{26 \times 26 \times 8}$
- Param: 3x3x1x8+8 = 80
- 3x3 filter for 1 channel, 8 such filters and 8 biases



Typical CNN Model

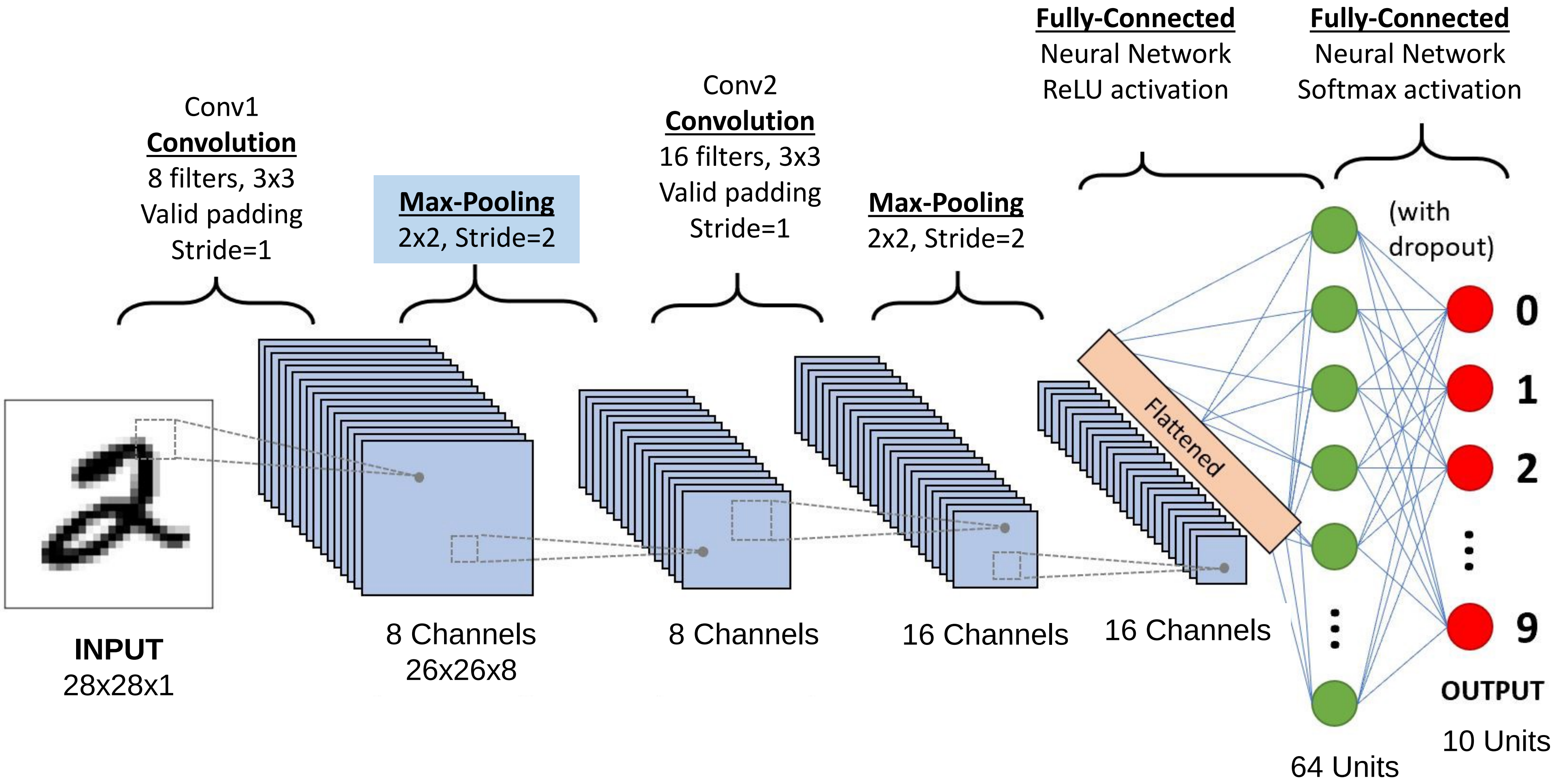
- Conv1:

  $26 \times 26 \times 8$

- Max-Pool

  $\left\lfloor \frac{n_{in} - k}{s} \right\rfloor + 1 \ = ?$

o/p: ?



Typical CNN Model

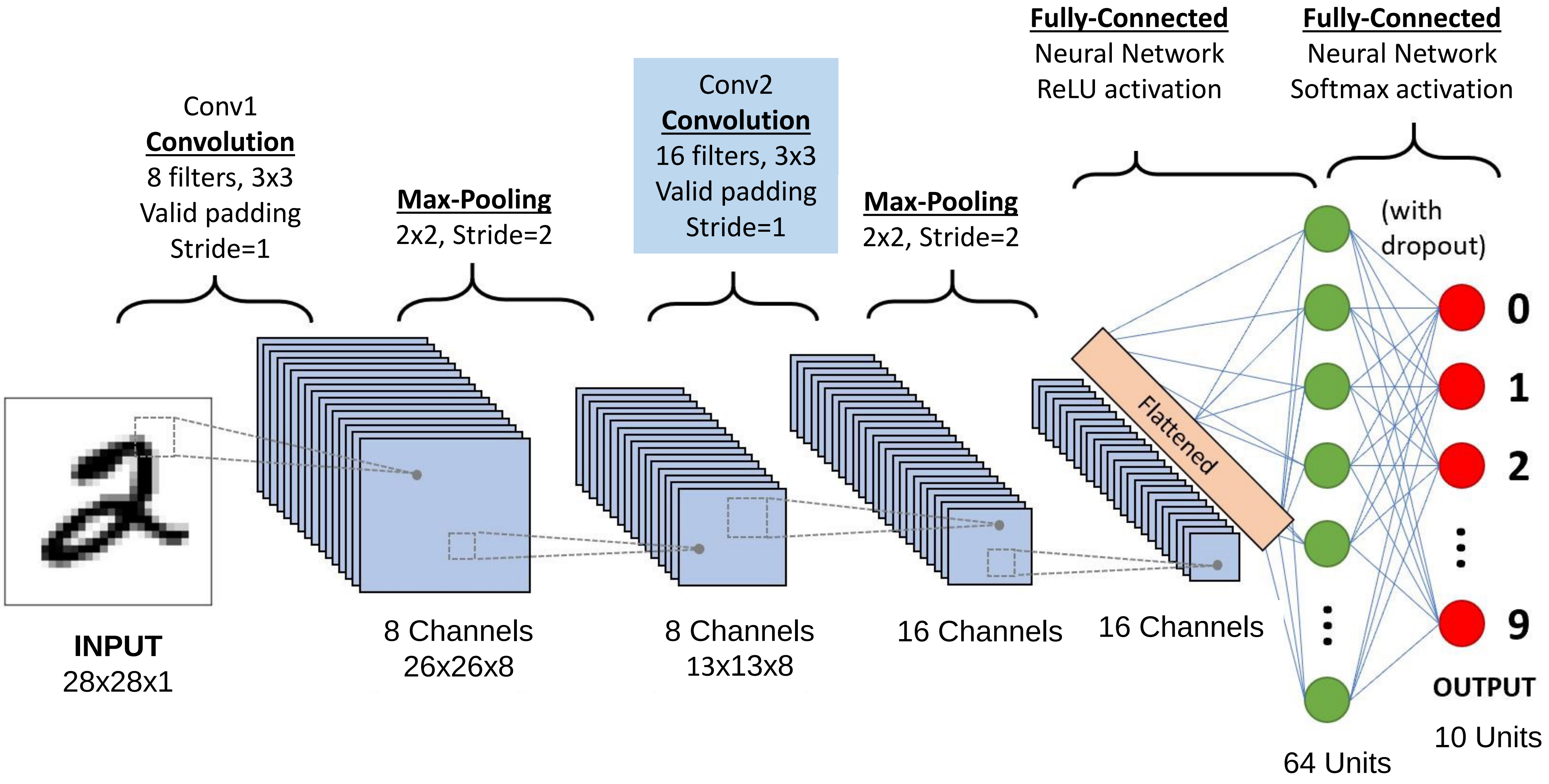
- Conv1:

  $26 \times 26 \times 8$

- Max-Pool
- $\left\lfloor \frac{26-2}{2} \right\rfloor + 1 = 13$

  o/p: $13 \times 13 \times 8$

- Conv2, $n_{out} = \left\lfloor \frac{n_{in}-k+2*p}{s} \right\rfloor + 1$

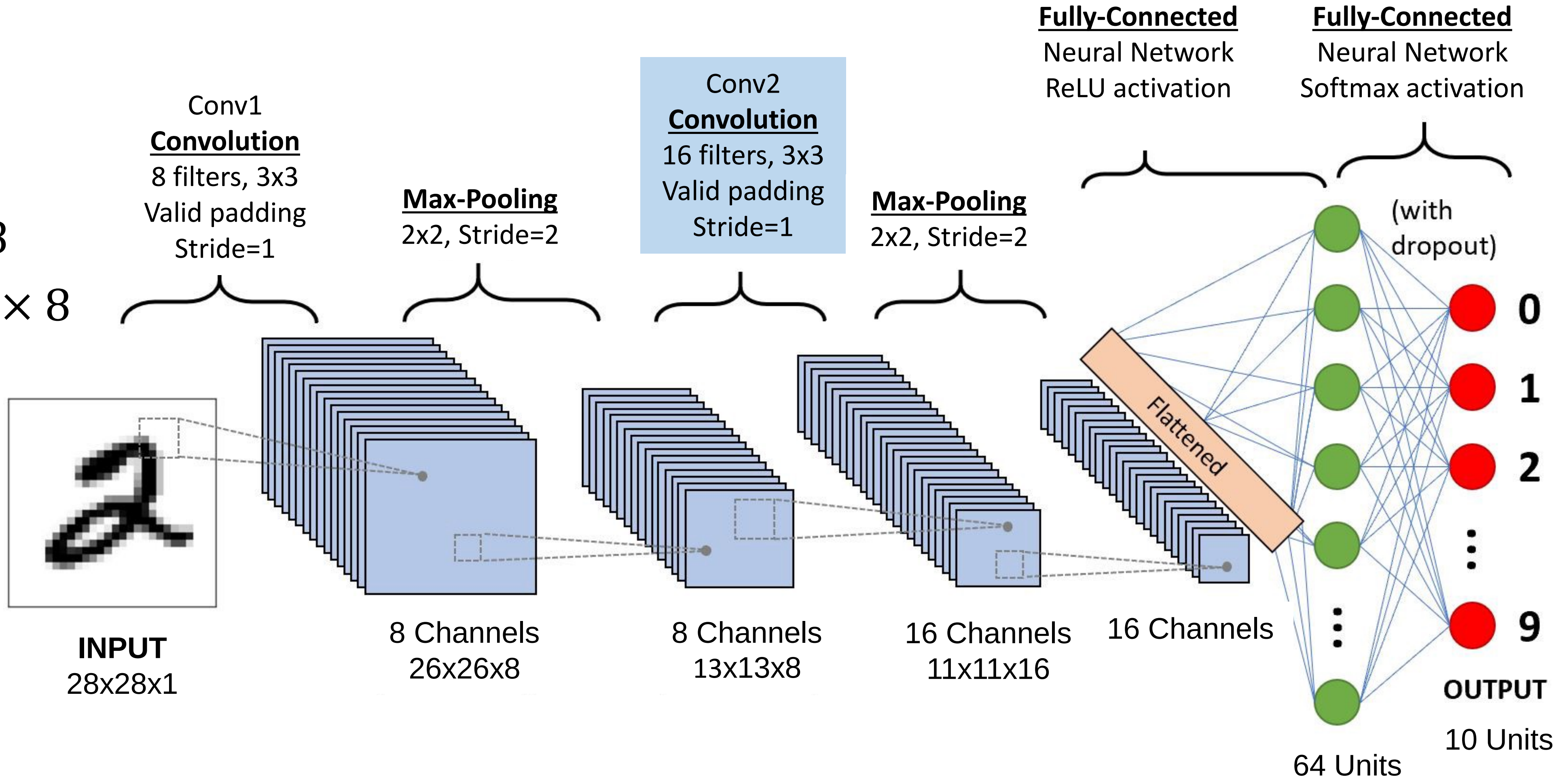


Typical CNN Model

- Conv1: $26 \times 26 \times 8$
- Max-Pool: $13 \times 13 \times 8$
- Conv2: $\left\lfloor \frac{13-3+2*0}{1} \right\rfloor + 1$
  $= 11$

o/p: $11 \times 11 \times 16$

- Param?



Typical CNN Model

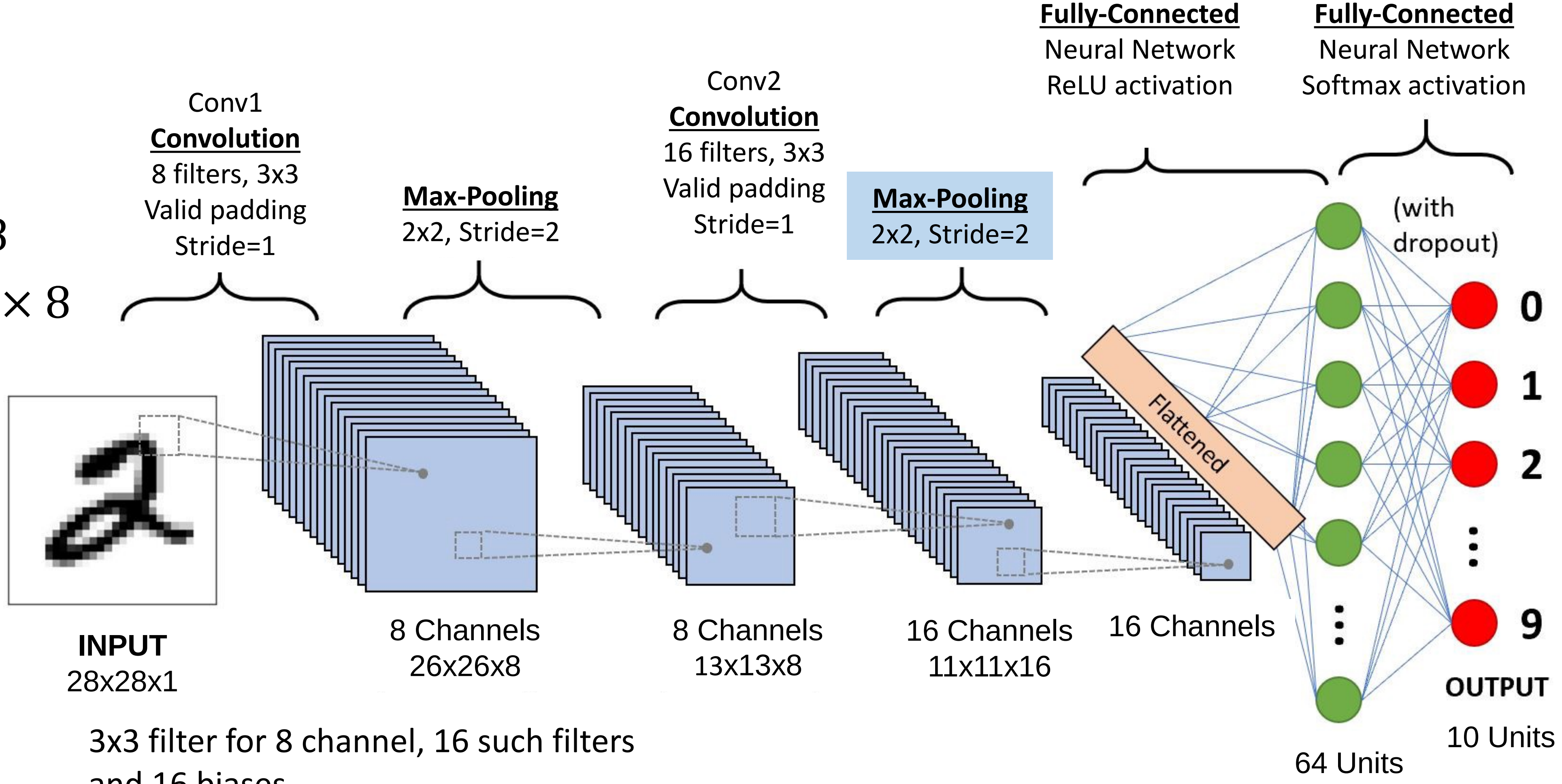
- Conv1: $26 \times 26 \times 8$
- Max-Pool: $13 \times 13 \times 8$
- Conv2:

$$\left\lfloor \frac{13-3+2*0}{1} \right\rfloor + 1 = 11$$

o/p: $11 \times 11 \times 16$

- Param=

3x3x8x 16+16=1168



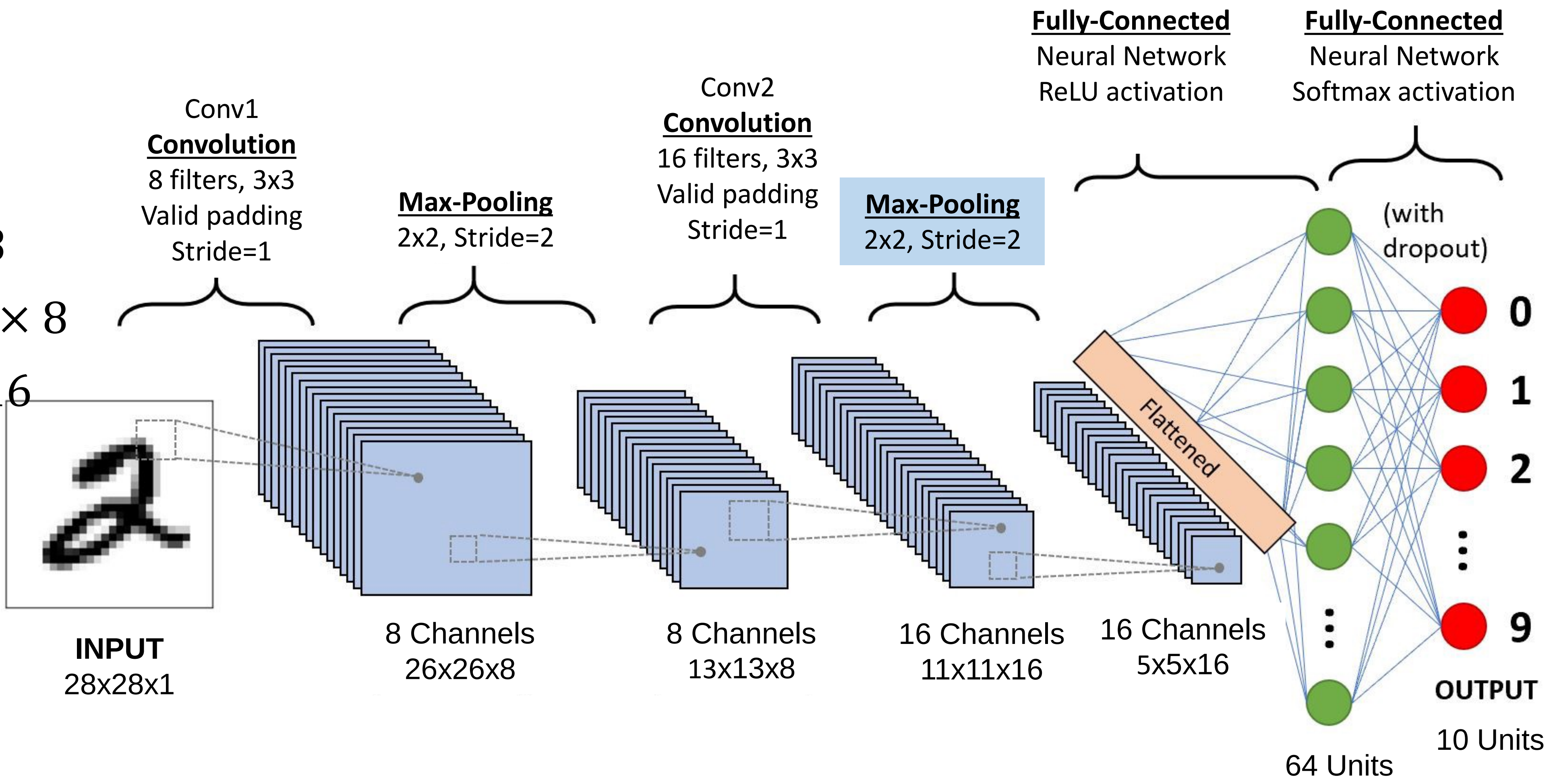
3x3 filter for 8 channel, 16 such filters and 16 biases

Typical CNN Model

- Conv1: $26 \times 26 \times 8$
- Max-Pool: $13 \times 13 \times 8$
- Conv2: $11 \times 11 \times 16$
- Max-Pool:

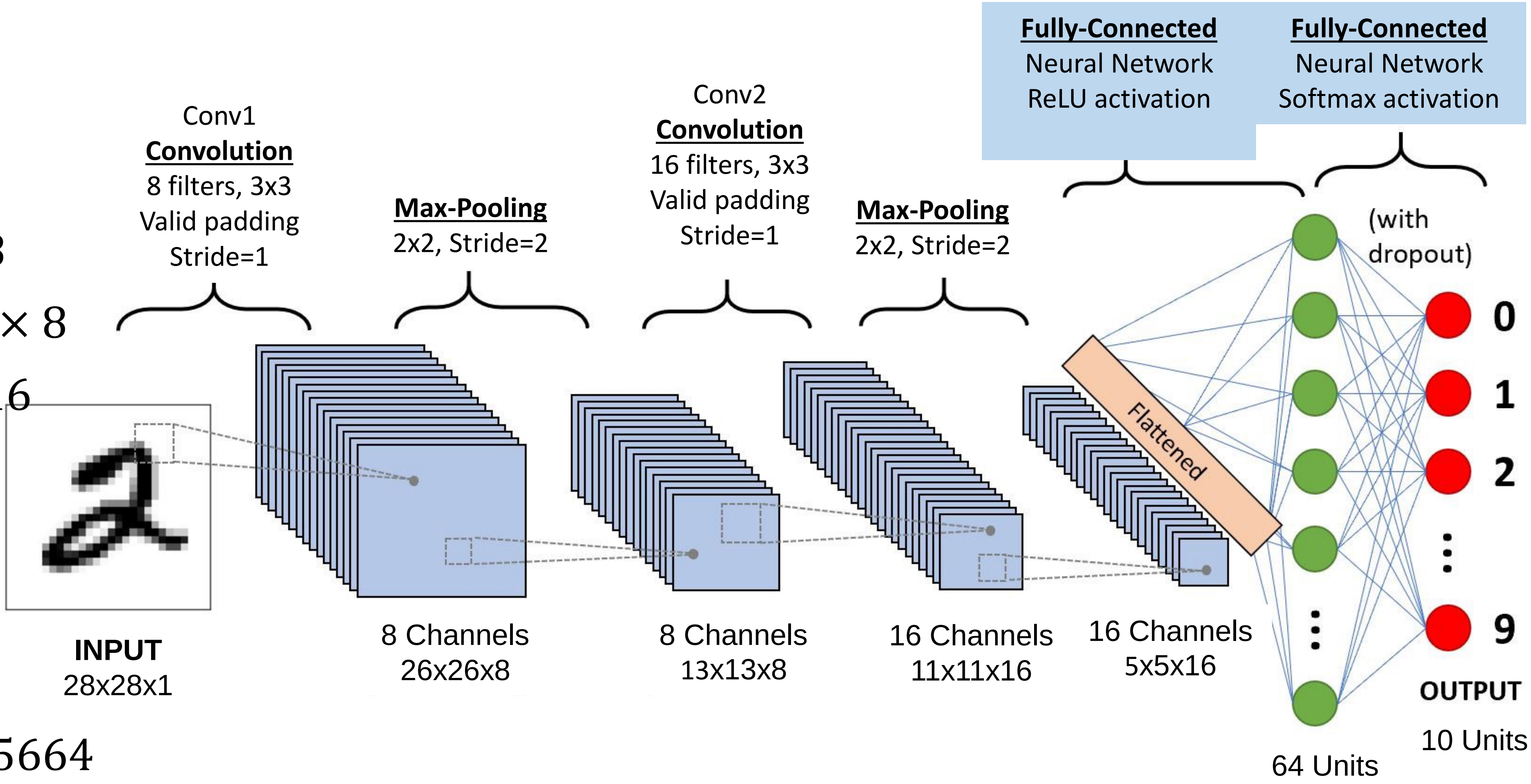
$5 \times 5 \times 16$

- $\left\lfloor \frac{11-2}{2} \right\rfloor + 1 = 5$



Typical CNN Model

- Conv1: $26 \times 26 \times 8$
- Max-Pool: $13 \times 13 \times 8$
- Conv2: $11 \times 11 \times 16$
- Max-Pool: $5 \times 5 \times 16$
- $5 \times 5 \times 16 = 400$
- FC1: $(400 + 1) \times 64 = 25664$
- FC2: (64+1)×10=650



Parameters and Hyperparameters

	Parameters	Hyperparameters
Convolution layer	Kernels	Kernel size, number of kernels, stride, padding, activation function
Pooling layer	None	Pooling method, filter size, stride, padding
Fully connected layer	Weights	Number of weights, activation function
Others		Model architecture, optimizer, learning rate, loss function, mini-batch size, epochs, regularization, weight initialization, dataset splitting

Note that a parameter is a variable that is automatically optimized during the training process and a hyperparameter is a variable that needs to be set beforehand

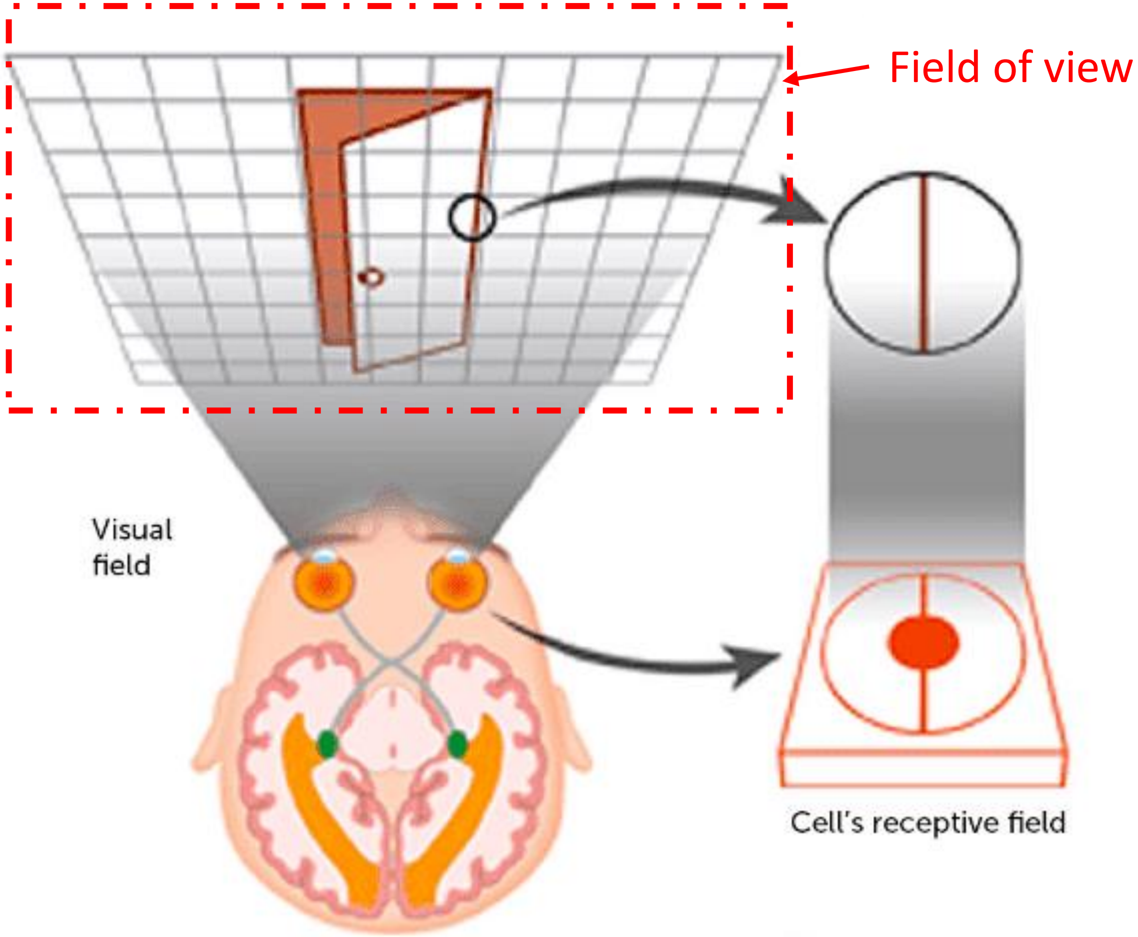
CNN Playground

Click the link below or copy paste the URL in your browser

https://poloclub.github.io/cnn-explainer/

With the CNN Explainer you can Learn and implement Convolutional Neural Network (CNN) in your browser! With real sample image dataset

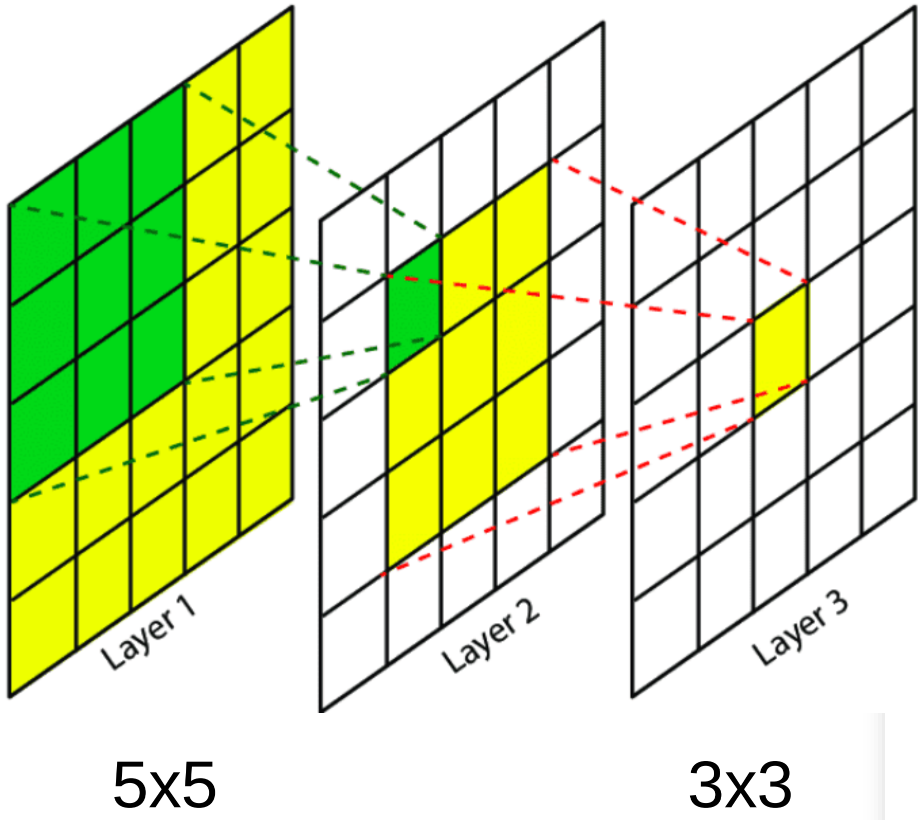
Understanding Receptive field



- The human visual system consists of millions of neurons, where each one captures different information.
- Defined as neuron's receptive field as the patch of the total field of view Or what information a single neuron has access to..

ImageSource: https://www.brainhq.com/brain-resources/brain-connection

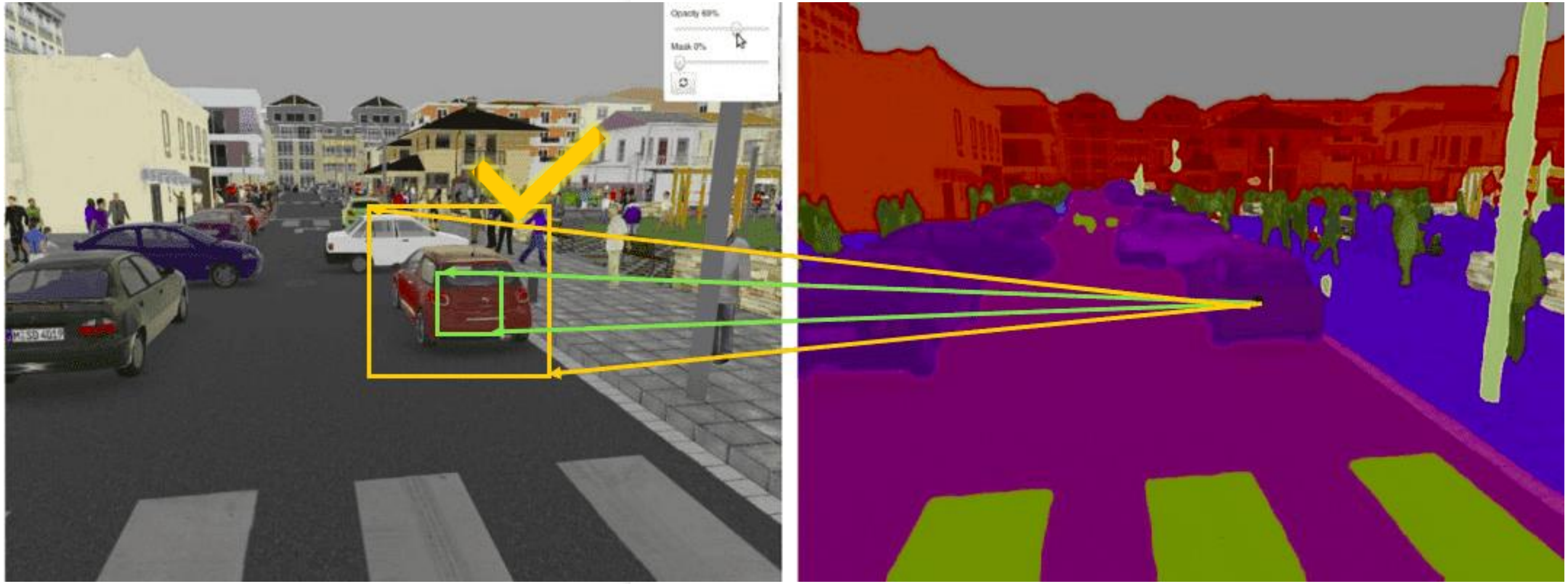
Receptive Field in Deep Learning



Effective Receptive Field

- Defined as a size of region in input that produces features. Basically, it is a measure of association of an output feature (of any layer) to the input region (patch).
- The idea of receptive fields applies to local operations (i.e. convolution, pooling).
- A convolutional unit only depends on a local region (patch) of the input.
- That's why RF never referred on fully connected layers since each unit has access to all the input region.

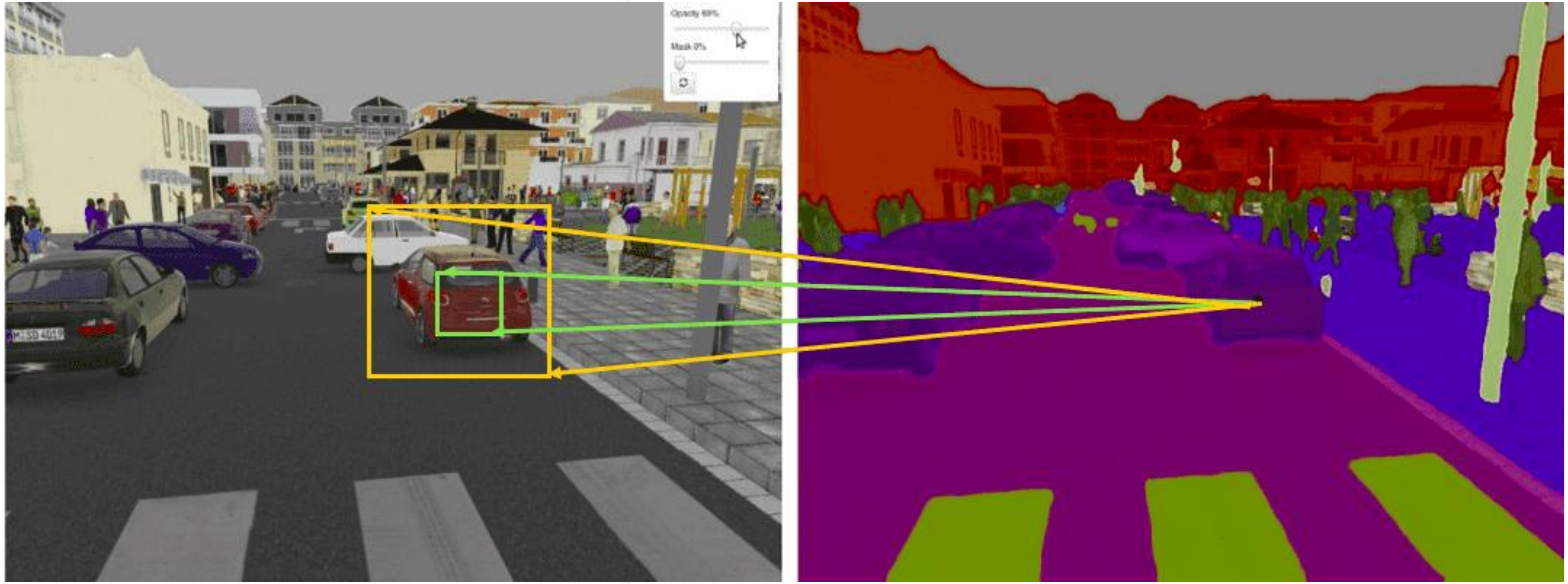
Why do we care for Receptive Field?



The green and the orange one. Which one would you like to have in your architecture?

Image Source: https://developer.nvidia.com/blog/image-segmentation-using-digits-5/

Why do we care for Receptive Field?



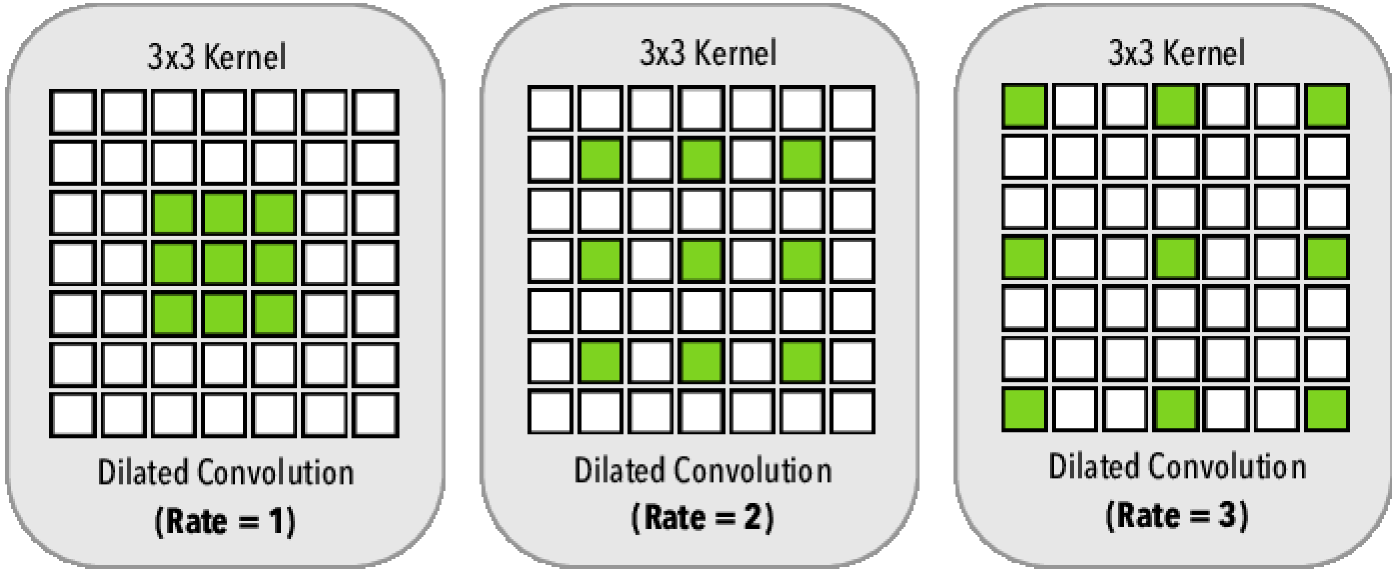
Therefore, our goal is to design a convolutional model so that we ensure that its RF covers the entire relevant input image region.

Image Source: https://developer.nvidia.com/blog/image-segmentation-using-digits-5/

How to increase receptive field in a convolutional network?

- Add more convolutional layers (make the network deeper)
- Add pooling layers or higher stride convolutions (sub-sampling)
- Use dilated convolutions

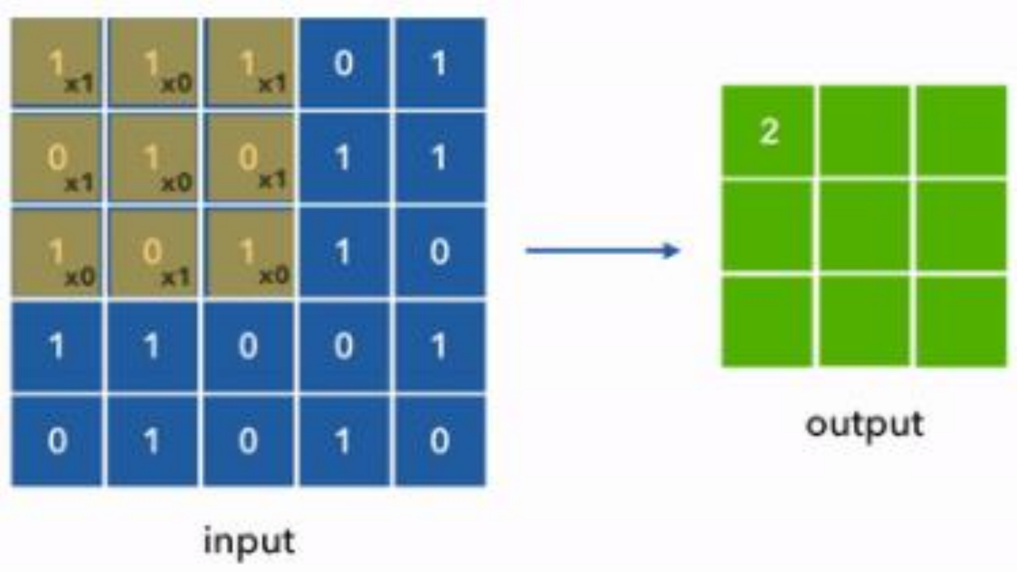
It is a technique that expands the kernel (input) by inserting holes between its consecutive elements.

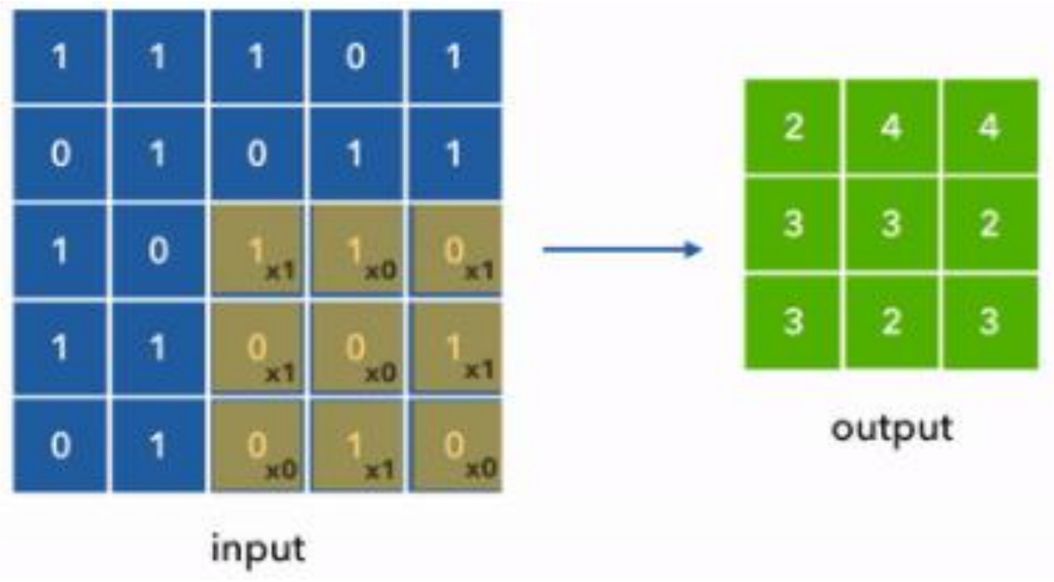


Parameter Sharing

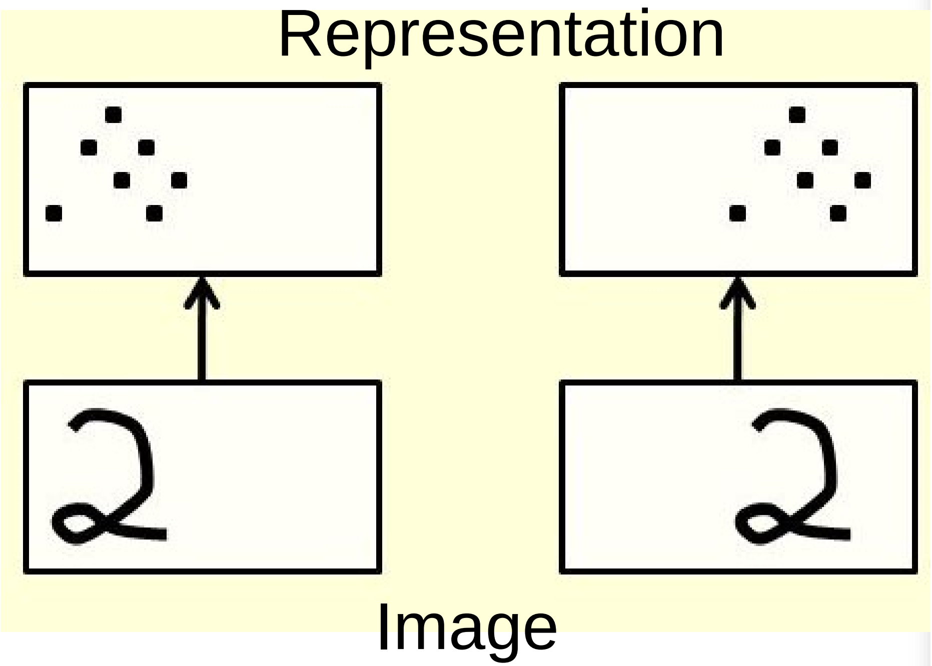
- Parameter sharing refers to using the same parameter for more than one function in a model

- Kernel is reused (by sliding) when calculating the layer o/p
- Less weights to store & train





Equivalent Representation



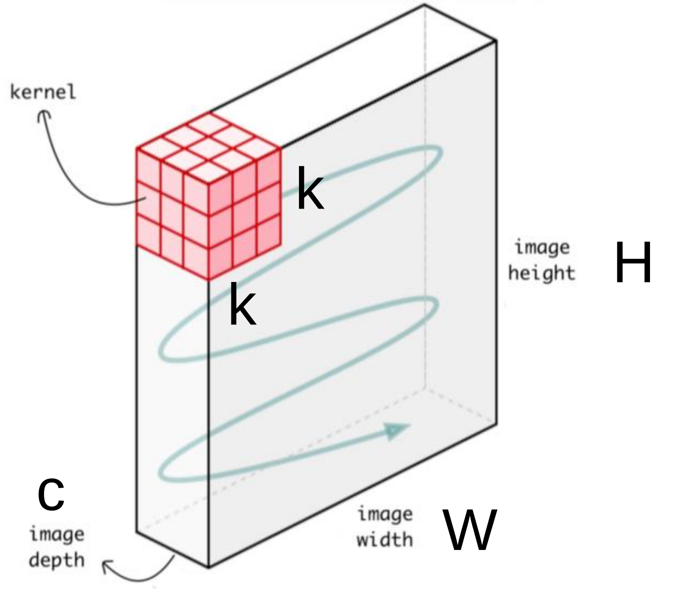
- Parameter sharing causes the layer to have a property called equivariance to translation
- Convolution creates 2-D map of where certain features appear in the input
- If we move the object in the input, its representation will move the same amount in the output
- Eg: Same kernel for Edge Detection wherever the edge occurs in the image

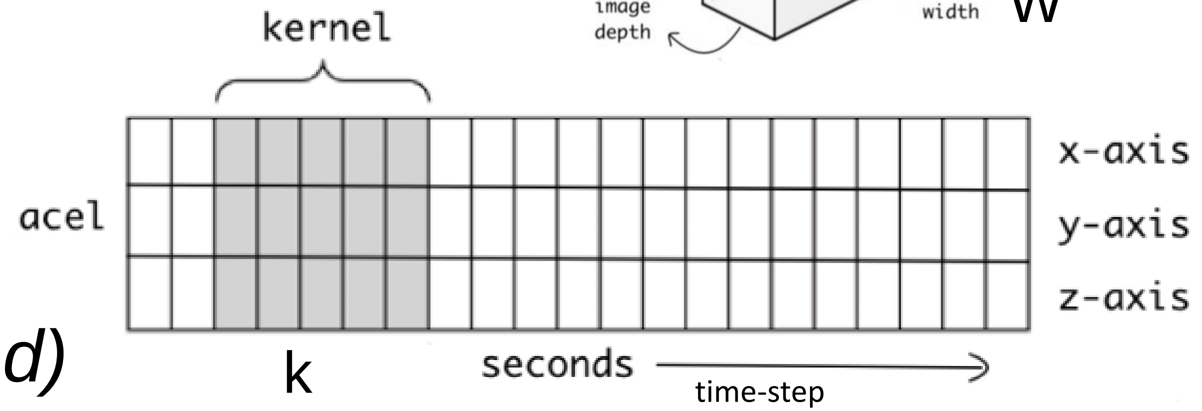
1D, 2D Convolution

- 2D Convolution
  - 2-directions (x,y) to calculate conv
  - input = (WxHxc), *d* filters (kxkxc) output = (W1xH1x*d*)
  - Eg: Image data (gray or color)
- 1D Convolution
  - 1-direction (time) to calculate conv
  - input = (time-step x c),

d filters (k x c) , output (time-step1 x *d)*

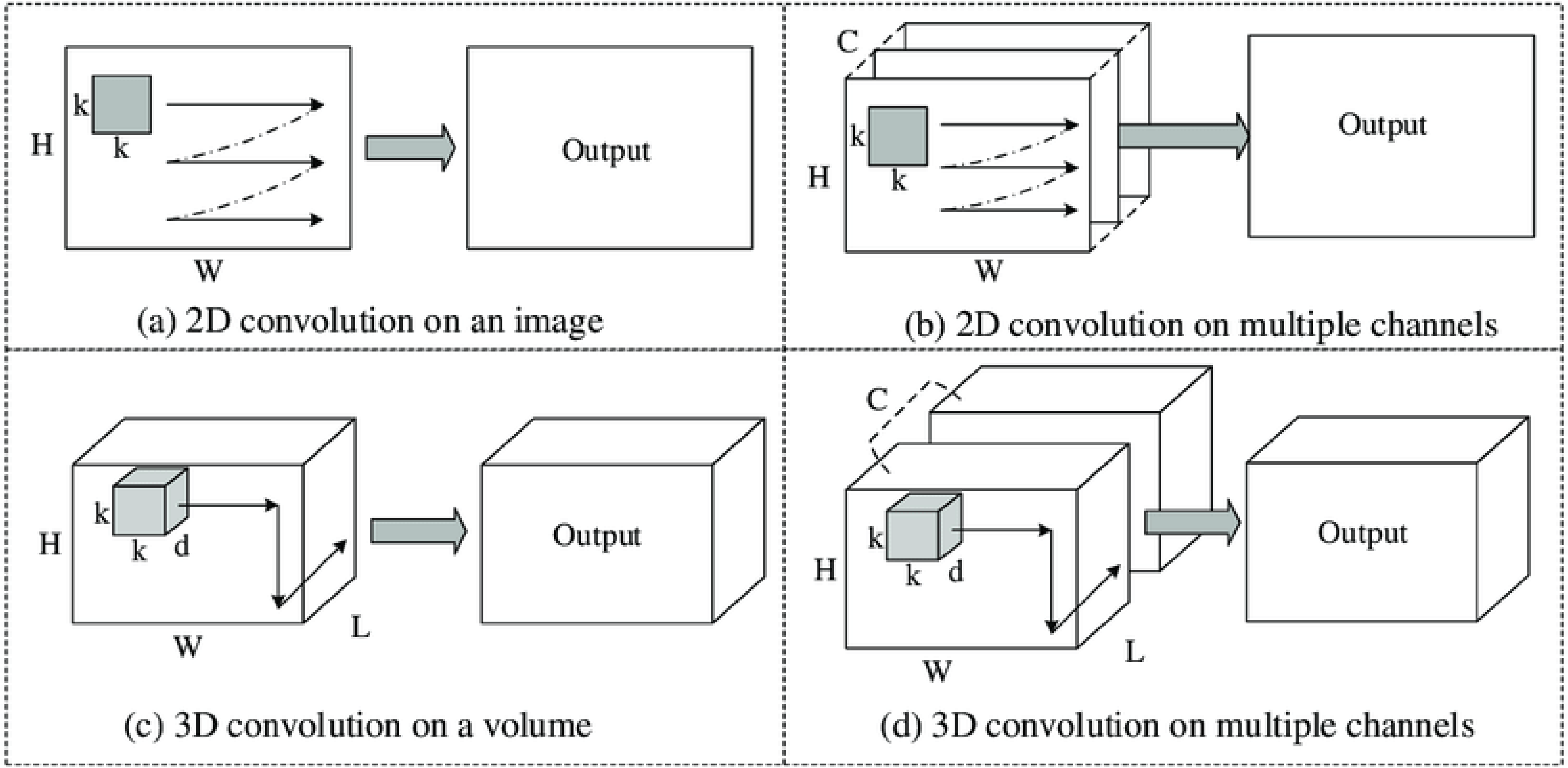
Eg: Time-series data, text analysis





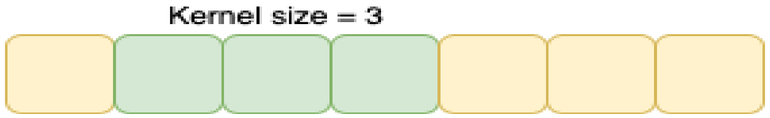
2D, 3D Convolution

- 3D Convolution
  - 3-directions (x,y,z) to calculate conv
  - input (WxHxLxC), *m* filters (kxkxd) output (W1xH1xL1xm)
  - Eg: MRI data, Videos



(a) 2D convolution on an image
(b) 2D convolution on multiple channels
(c) 3D convolution on a volume
(d) 3D convolution on multiple channels

1 D CNN



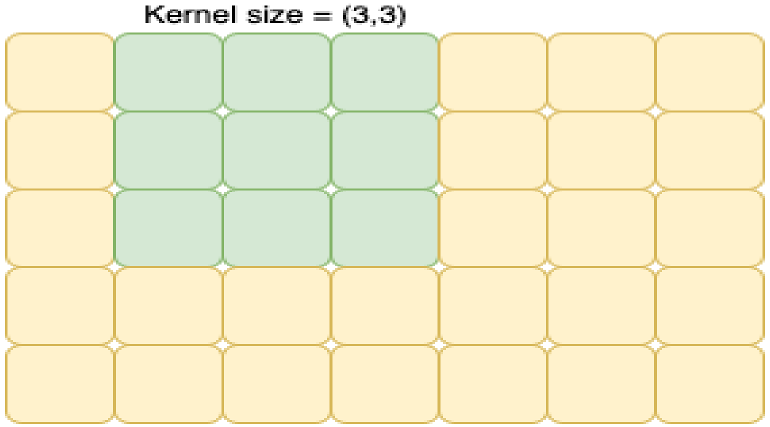
Input shape =2D

Batch size =None

Width = Time axis =7

Feature map/ channels =1

2 D CNN



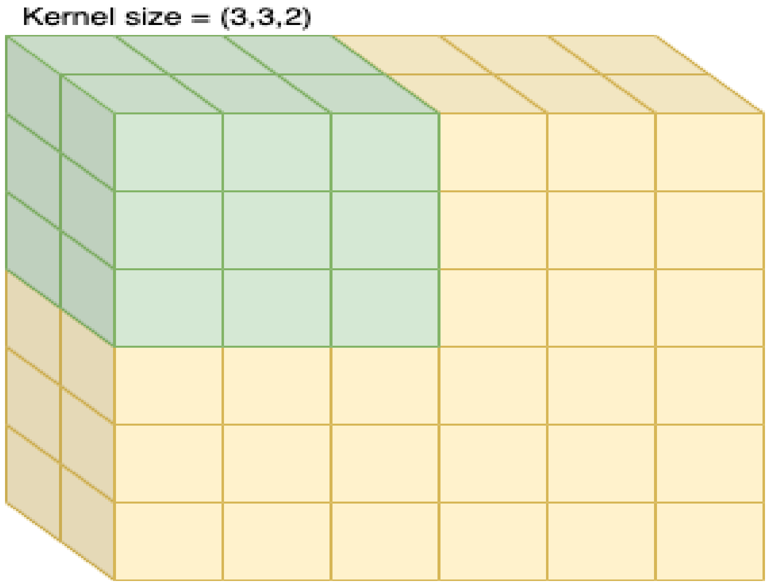
Input shape = 3D

Height=5

Width = 7

Feature map/ channels =1

3 D CNN



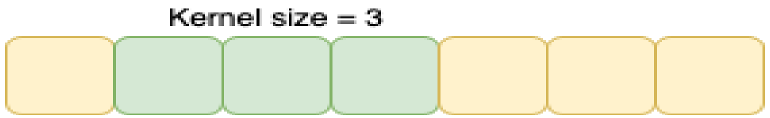
Input shape = 4D

Height=6

Width = 6

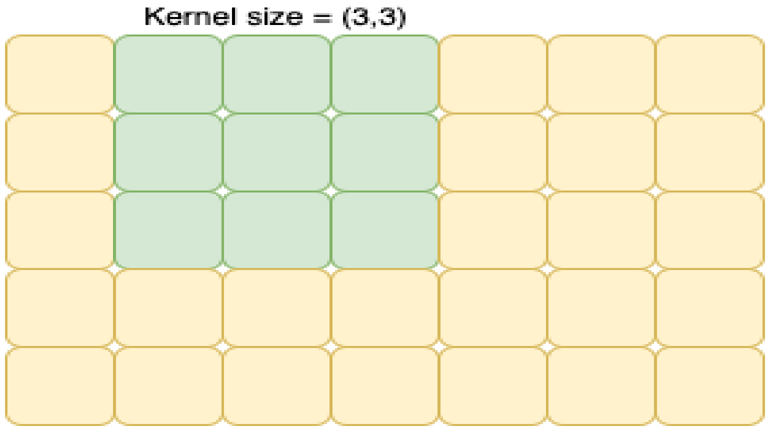
Feature map/ channels =depth=1

1 D CNN



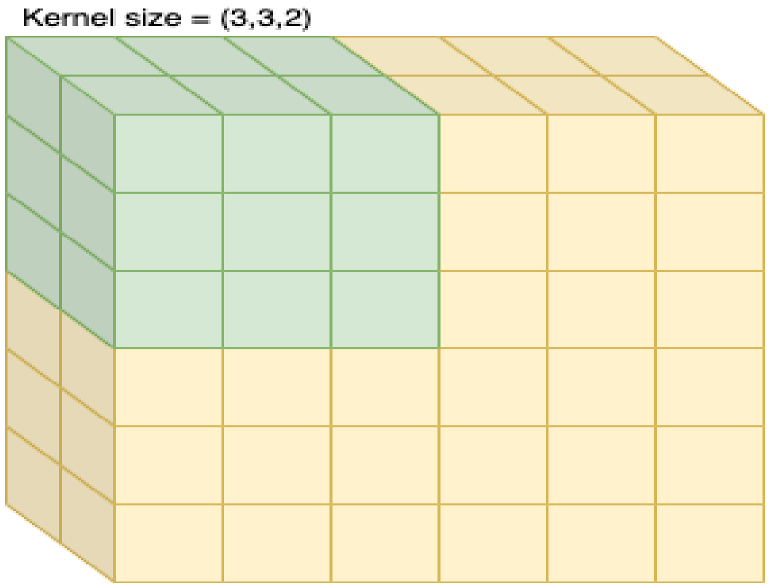
Input shape =2D
Batch size =None
Width = Time axis =7
Feature map/ channels =1

2 D CNN



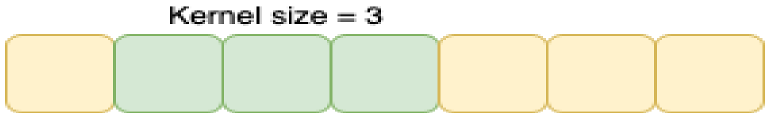
Input shape = 3D
Height=5
Width = 7
Feature map/ channels =1

3 D CNN



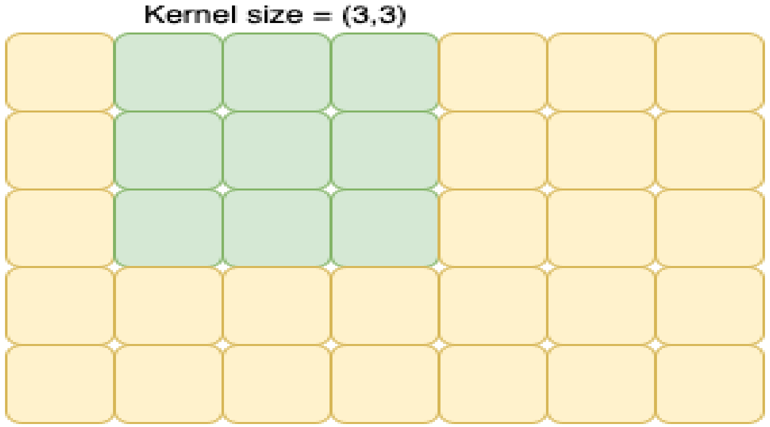
Input shape = 4D
Height=6
Width = 6
Feature map/ channels =depth=1

1 D CNN



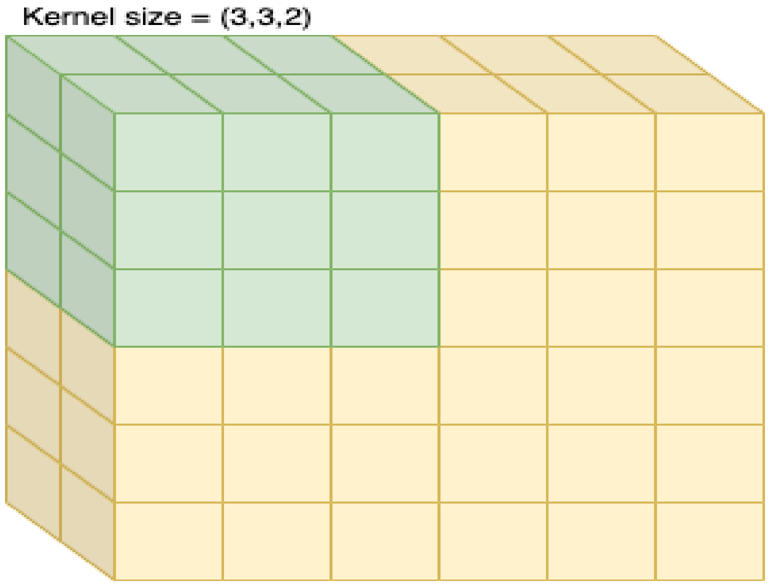
Input shape =2D
Batch size =None
Width = Time axis =7
Feature map/ channels =1

2 D CNN



Input shape = 3D
Height=5
Width = 7
Feature map/ channels =1

3 D CNN



Input shape = 4D
Height=6
Width = 6
Feature map/ channels =depth=1