CS409 : Neural Networks (Semester II - 2021/22)

Unit 6: Convolutional Neural Networks (CNNs) (1)

Dr. Ruwan Nawarathna
Department of Statistics & Computer Science
Faculty of Science
University of Peradeniya

Content

- Introduction to Convolutional Neural Networks (CNNs)
- What is Convolution?
- Types of layers of a CNN

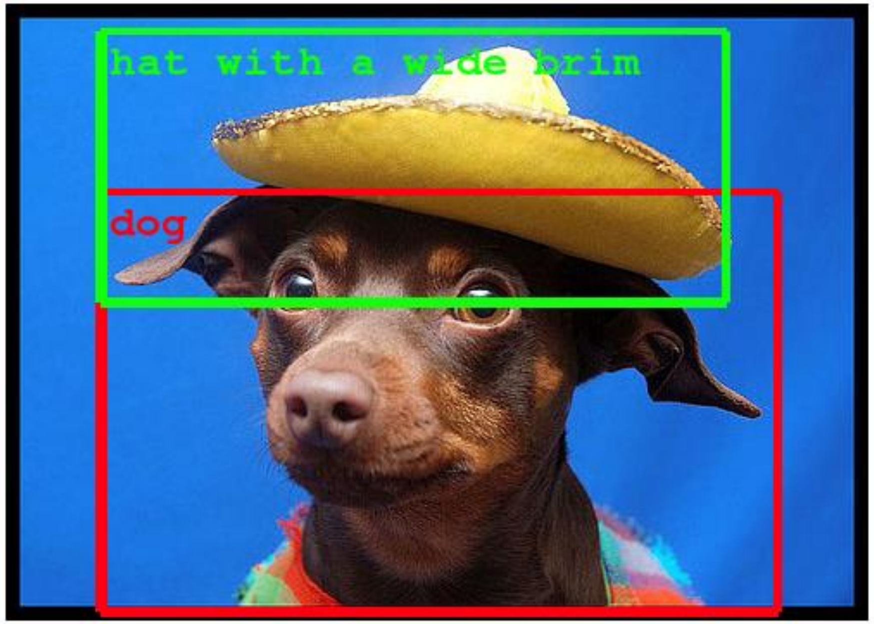
Convolutional Neural Networks (CNNs)

- **Convolutional networks**, also known as **convolutional neural networks** or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology.

- Examples include:
  - image data, which can be thought of as a 2D grid of pixels.
  - time-series data, which can be thought of as a 1D grid taking samples at regular time intervals.

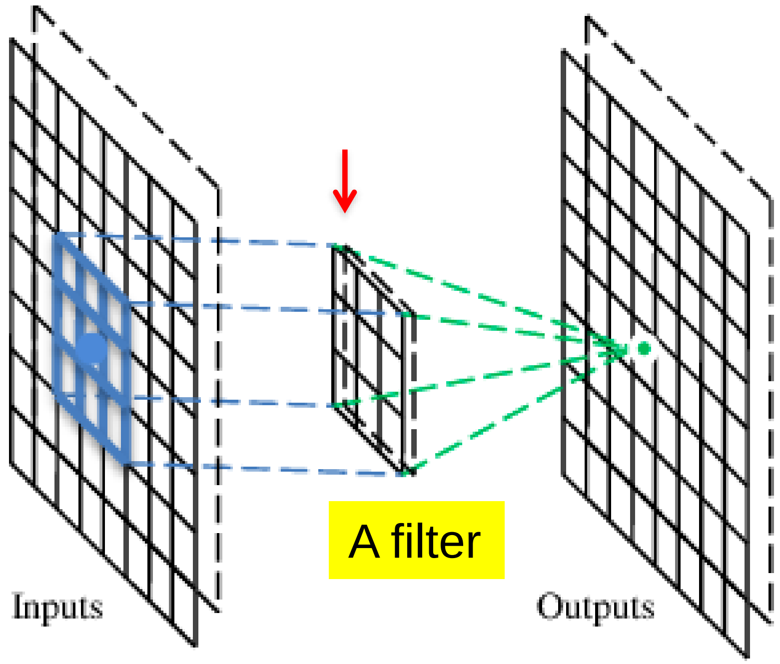
CNNs

- Convolutional networks have been tremendously successful in practical applications.
  - Computer vision
    - Object classification and detection in photographs
  - Natural language processing
  - Speech recognition



CNNs

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.



CNNs

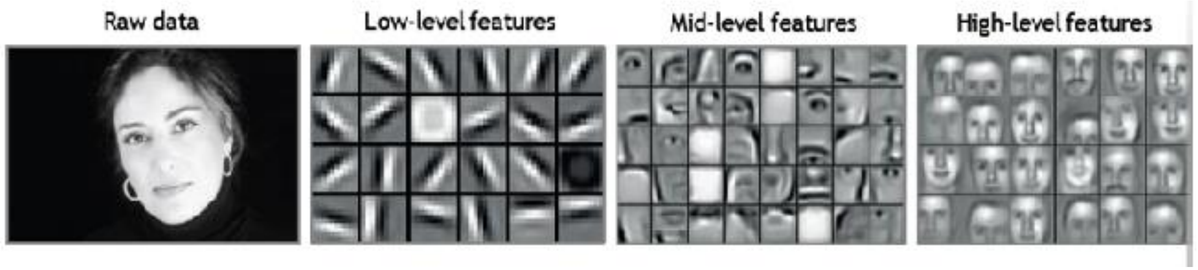
- The name “convolutional neural network” indicates that the network employs a mathematical operation called **convolution**.

- Convolution is a specialized kind of linear operation.

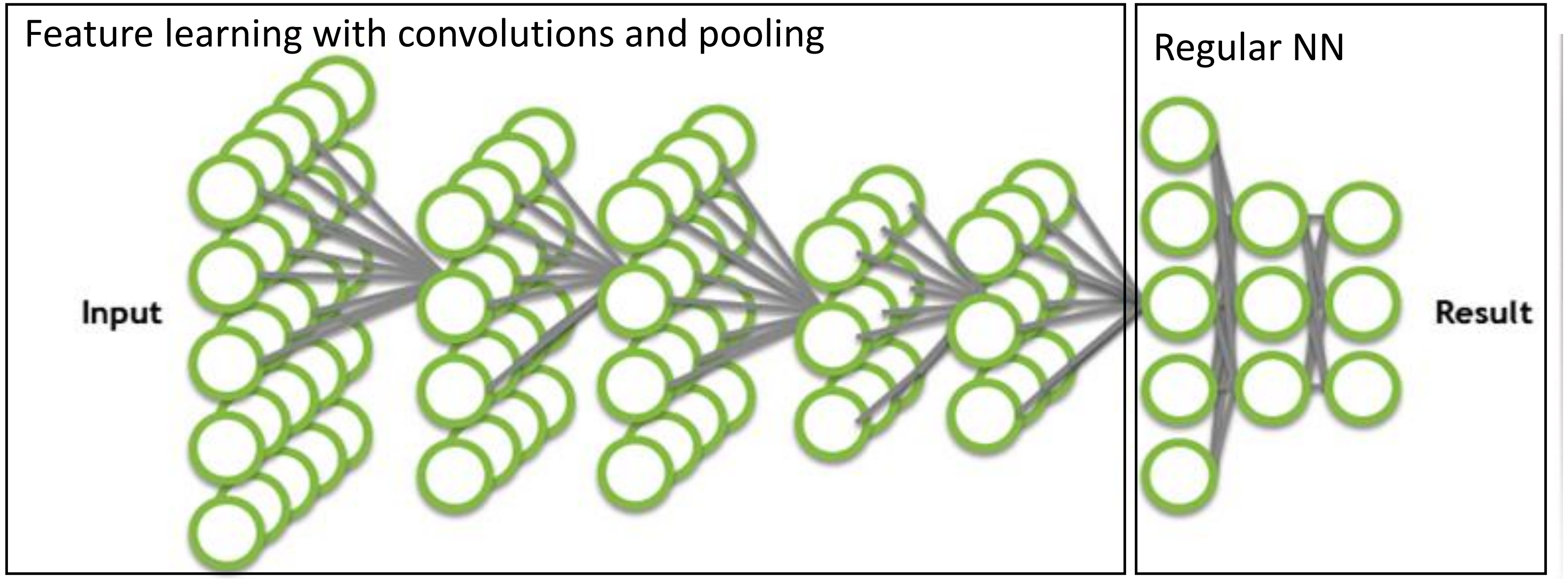
- Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

CNNs

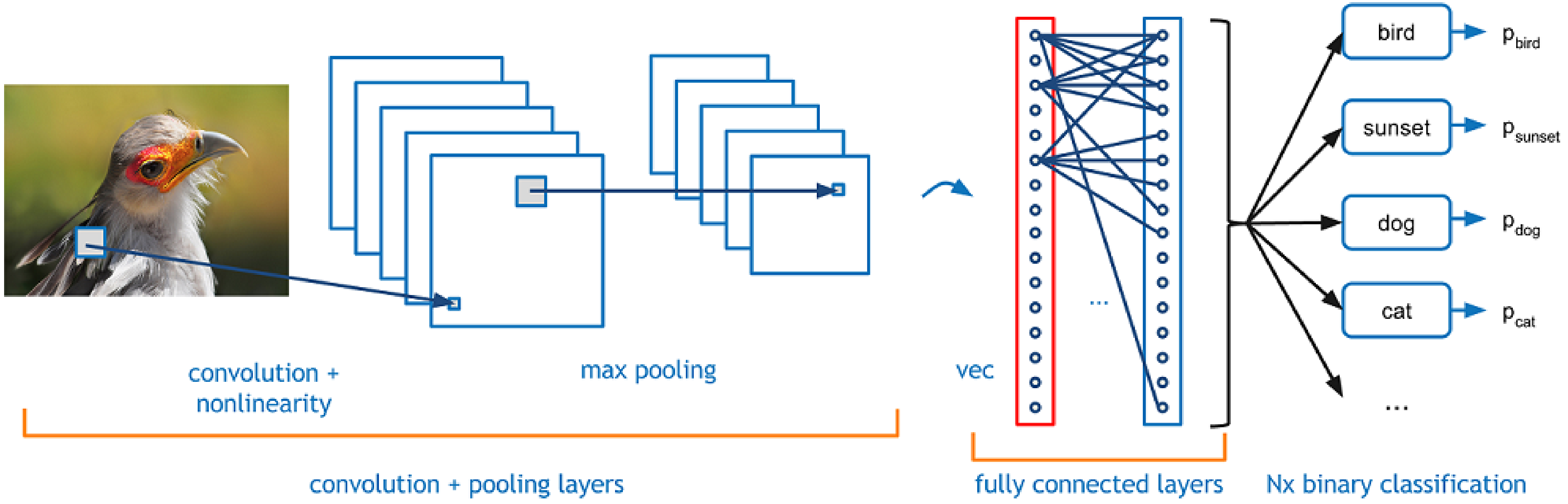
- Through series of convolutions (convolutional layers) feature learning is performed at various levels.



Output Label



CNNs



bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution +
nonlinearity
max pooling
vec
...
convolution + pooling layers
fully connected layers
Nx binary classification

Types of layers

- Three main types of layers are used to build CNN architectures:

1. Convolutional layers (CONV)
   - Output: Feature Map
   - ReLU (Rectified linear unit) layers (RELU)
2. Pooling (or Subsampling) layers (POOL)
3. Fully connected layers (classification) (FC)
   - Multi-layer perceptron

Convolutional layer

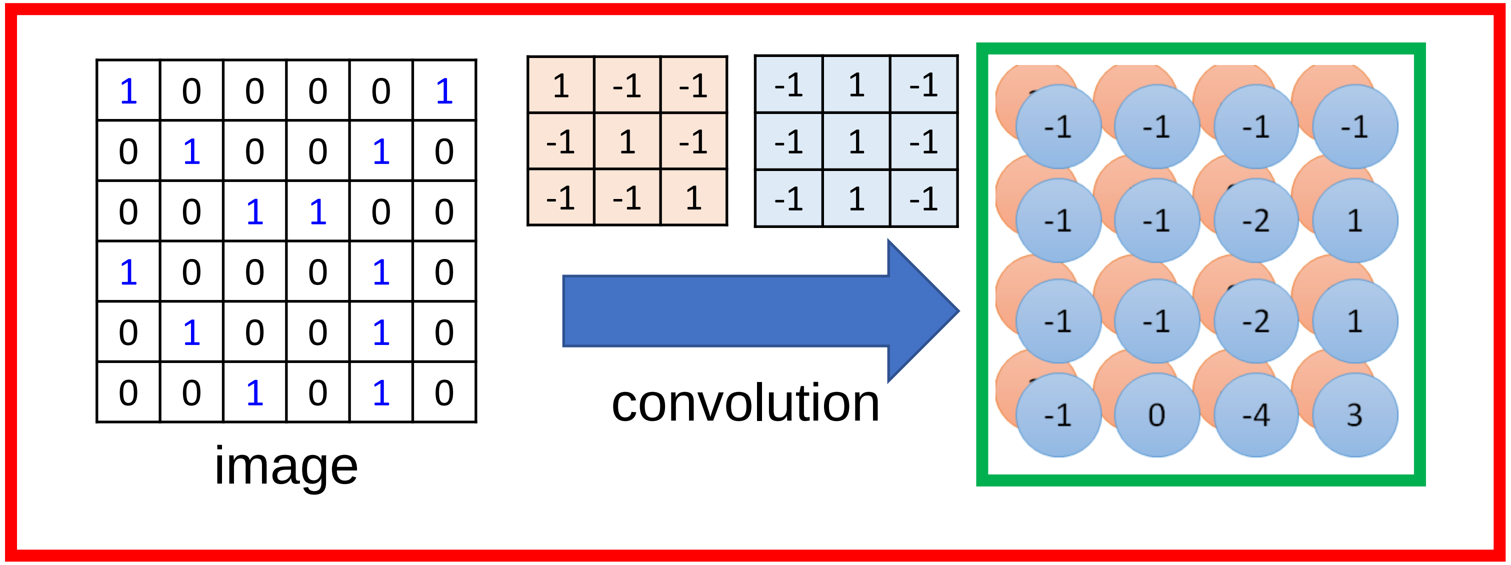
- Rectangular grid of neurons
- Input from a rectangular section of previous layer
- Weights are same for each neuron
- Weights specify convolutional filters
- Several grids in each layer, each grid takes inputs from all layers using different filters

Convolution

- Convolution is a common image processing technique that changes the intensities of a pixel to reflect the intensities of the surrounding pixels.

- A common use of convolution is to create image filters

Convolution



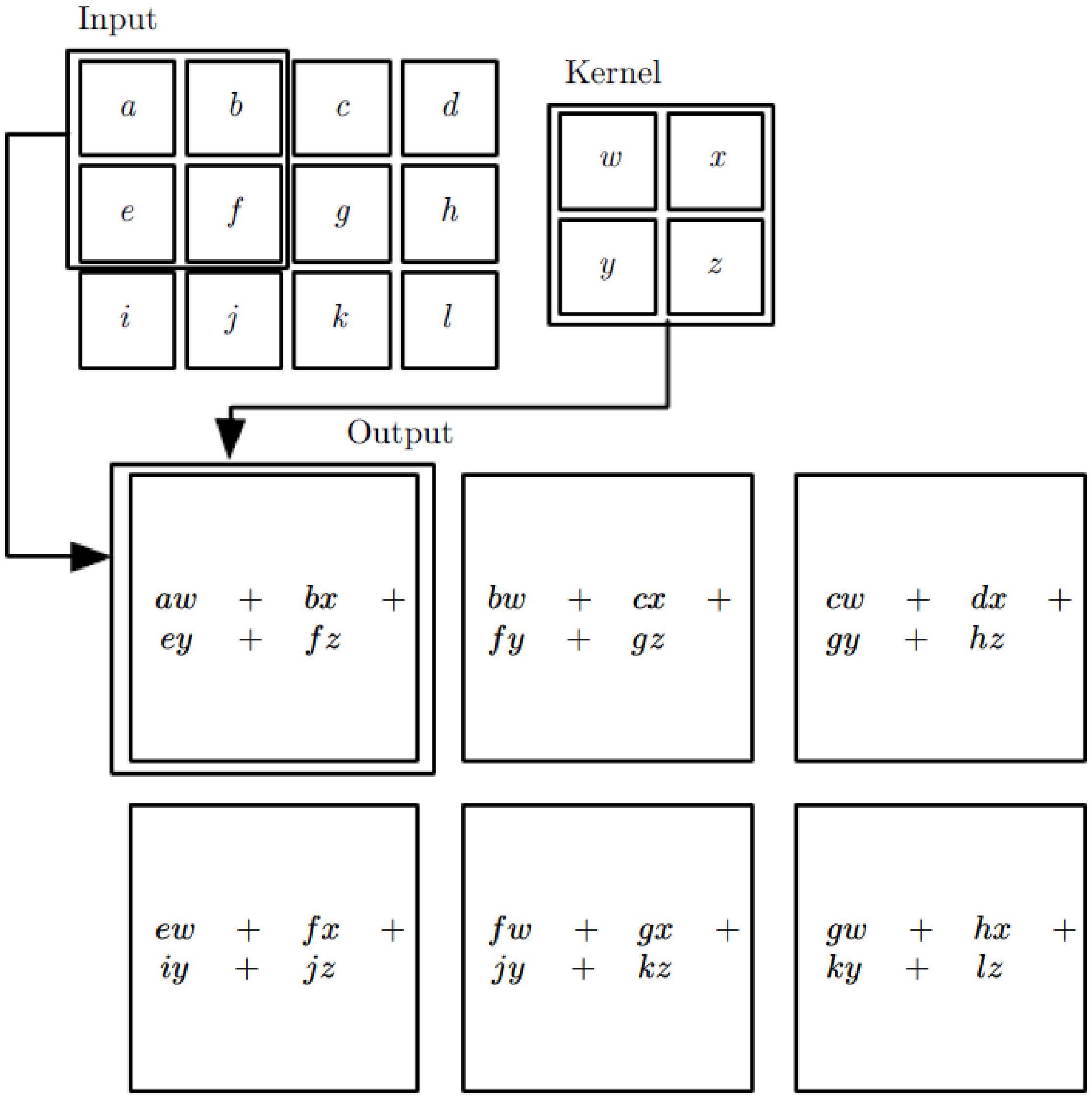
Effect of Convolution

Original Image

Convoluted image

This is an output of smoothing filter

Convolution



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

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

Convolution

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

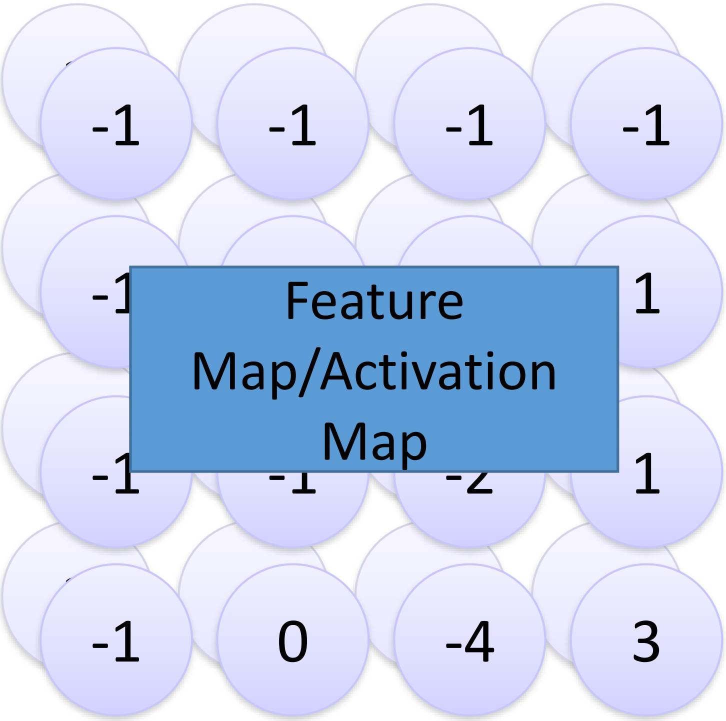
Filter 2

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 2 x 4 x 4 matrix

Convolution on Color images: RGB 3 channels

Now the filter is 3D

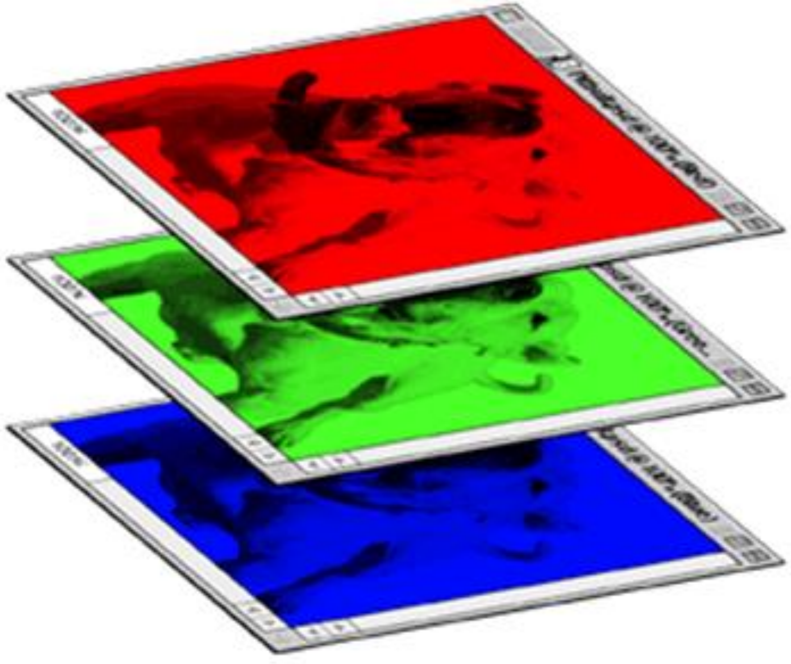
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2

Color 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 with Neurons

weights

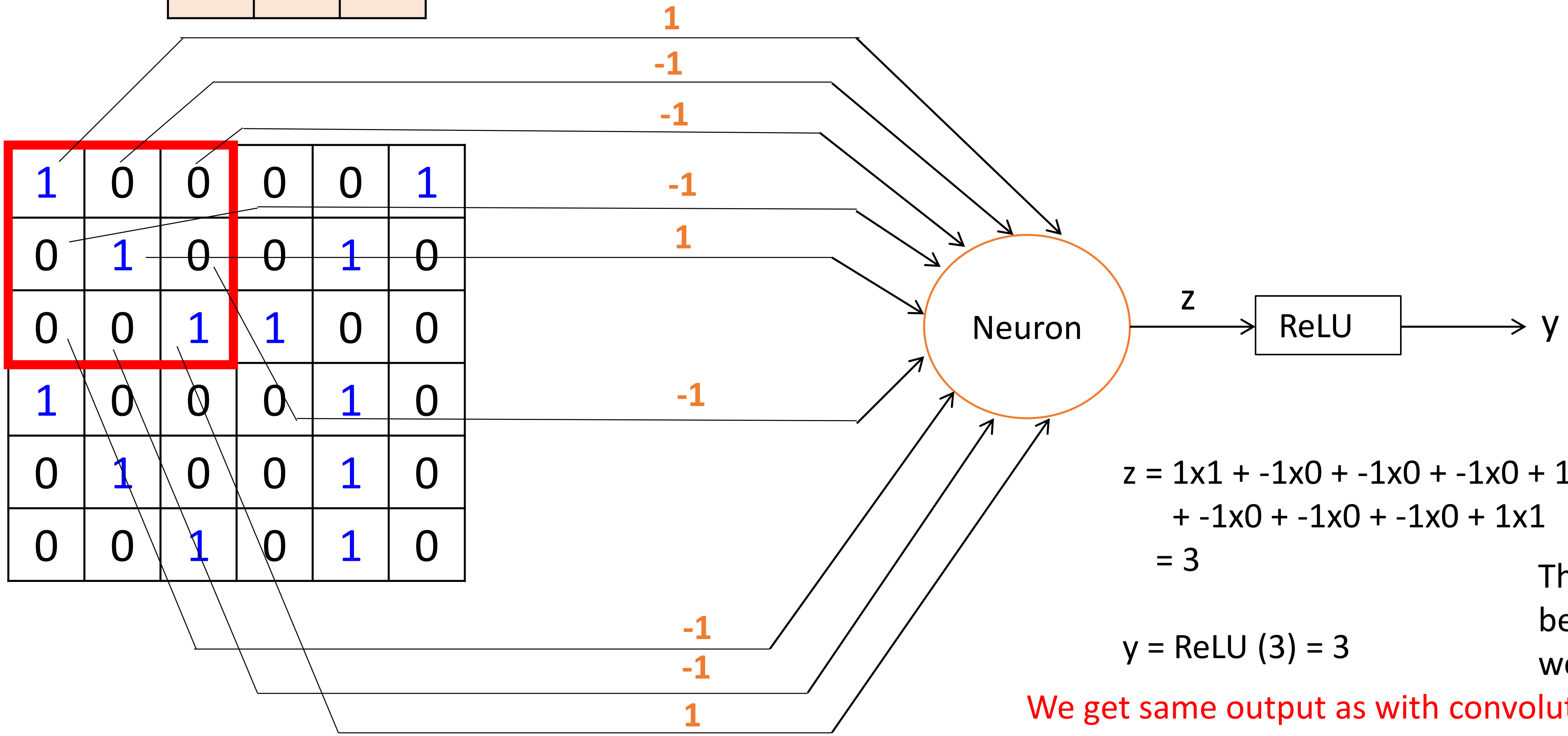
1	-1	-1
-1	1	-1
-1	-1	1

Convolution

$z = 1x1 + -1x0 + -1x0 + -1x0 + 1x1 + -1x0 + -1x0 + -1x0 + 1x1$
$= 3$

Inputs

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



$z = 1x1 + -1x0 + -1x0 + -1x0 + 1x1$
$+ -1x0 + -1x0 + -1x0 + 1x1$
$= 3$

$y = \text{ReLU}(3) = 3$

There could be a bias weight

We get same output as with convolution

Convolution with Neurons

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

Grid of Neurons (Activation Map)

ReLU

Considered as a ReLU layer

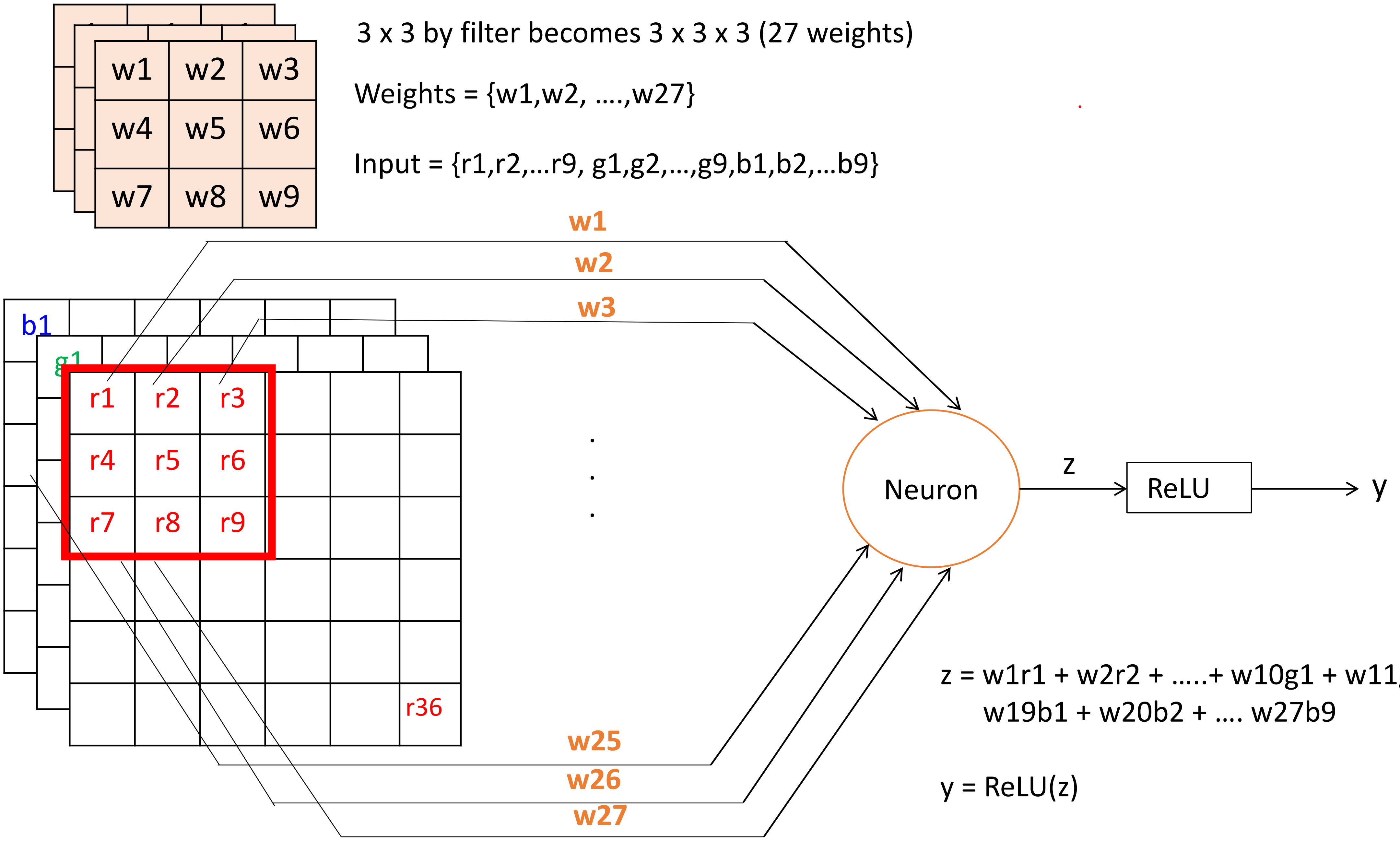
3	0	0	0
0	1	0	0
0	0	0	1
3	0	0	0

Convolution with Neurons – 3D Inputs

3 x 3 by filter becomes 3 x 3 x 3 (27 weights)

Weights = {w1,w2, ….,w27}

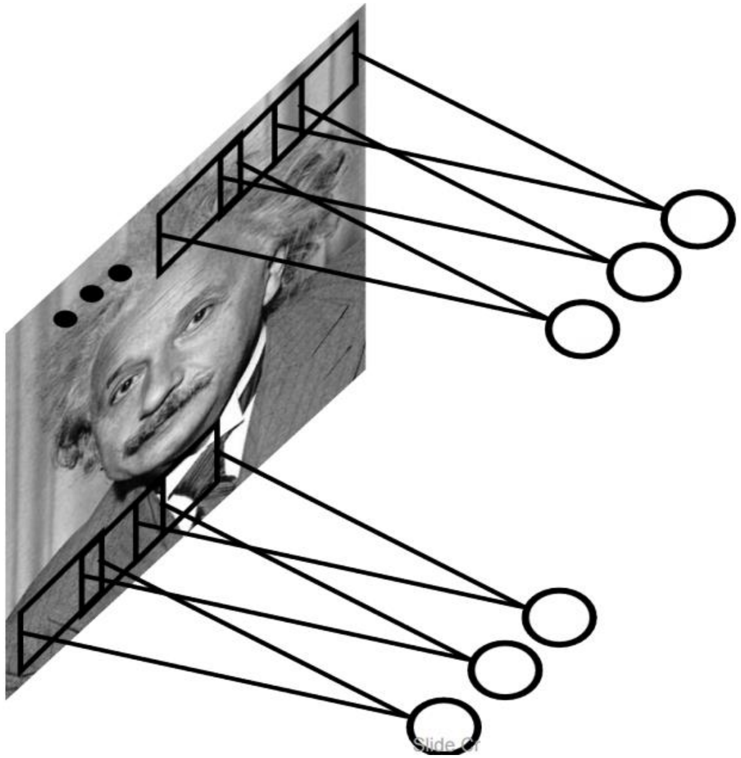
Input = {r1,r2,…r9, g1,g2,…,g9,b1,b2,…b9}



z = w1r1 + w2r2 + …..+ w10g1 + w11g2 + …
w19b1 + w20b2 + …. w27b9

y = ReLU(z)

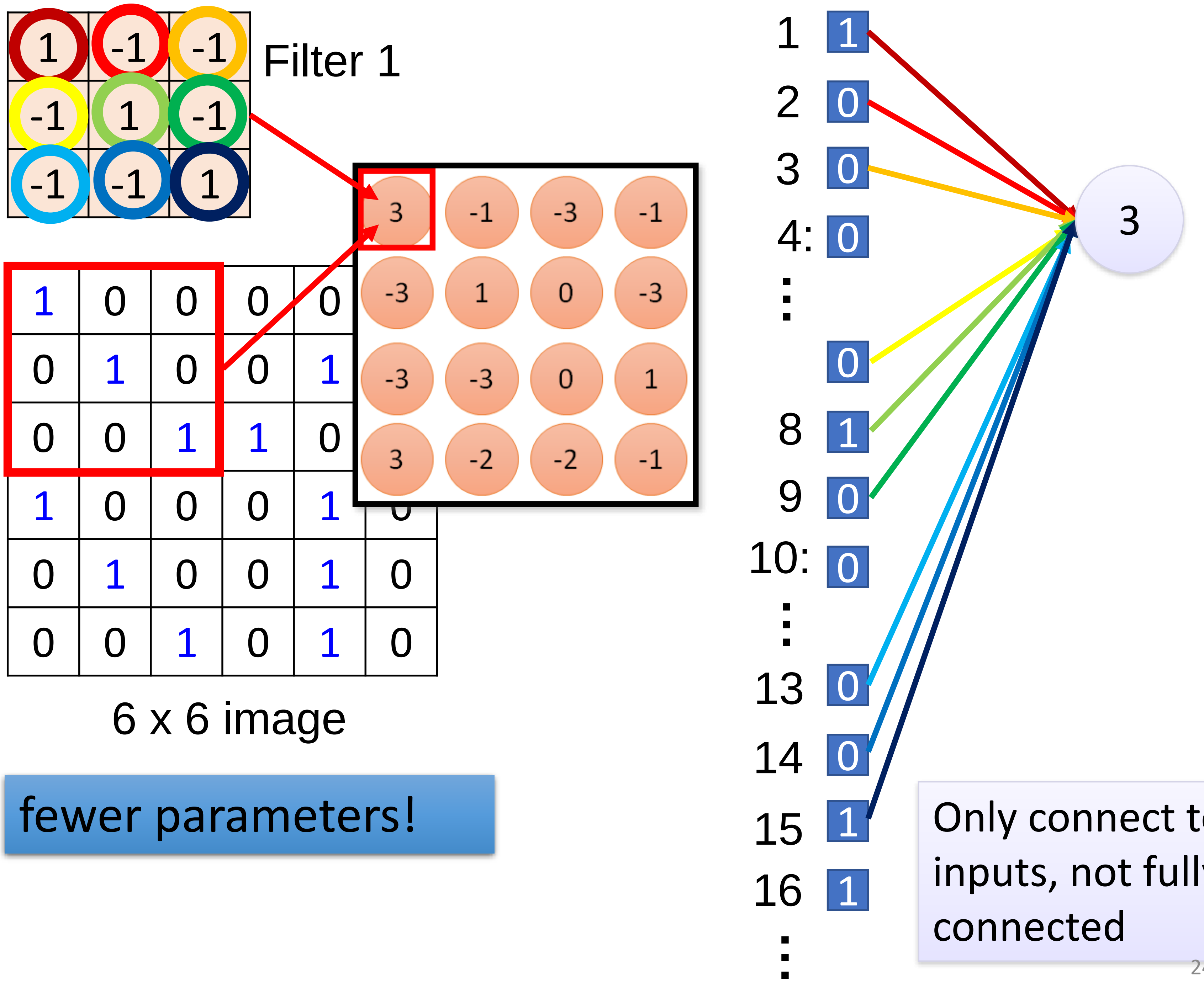
Convolutional Layers : Parameter Sharing and Local Connectivity



- Parameter sharing is sharing of weights by all neurons in a particular feature map.

- Local connectivity is the concept of each neural connected only to a subset of the input image (unlike a neural network where all the neurons are fully connected)

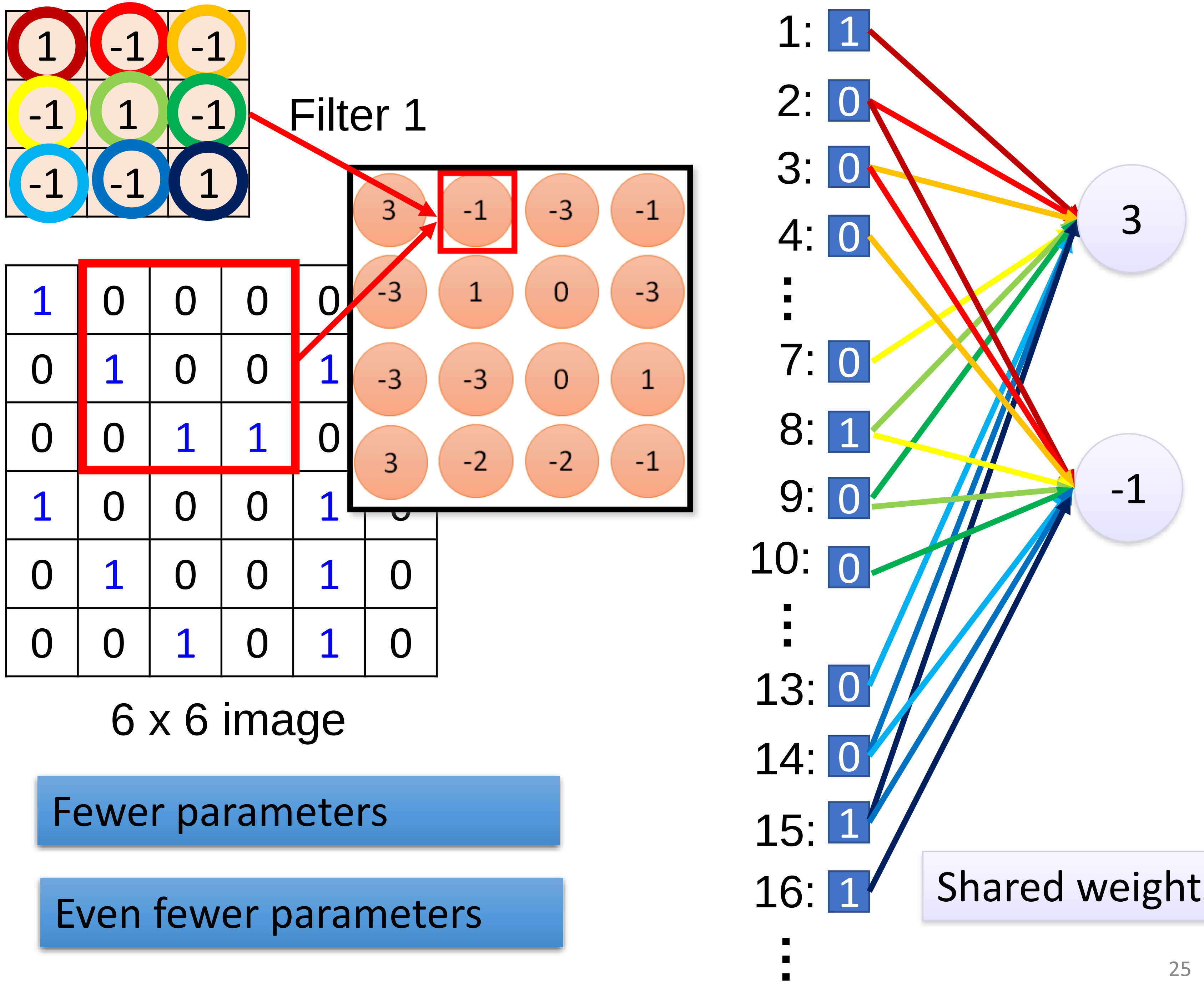
Parameter Sharing and Local Connectivity



fewer parameters!

Only connect to 9 inputs, not fully connected

Parameter Sharing and Local Connectivity



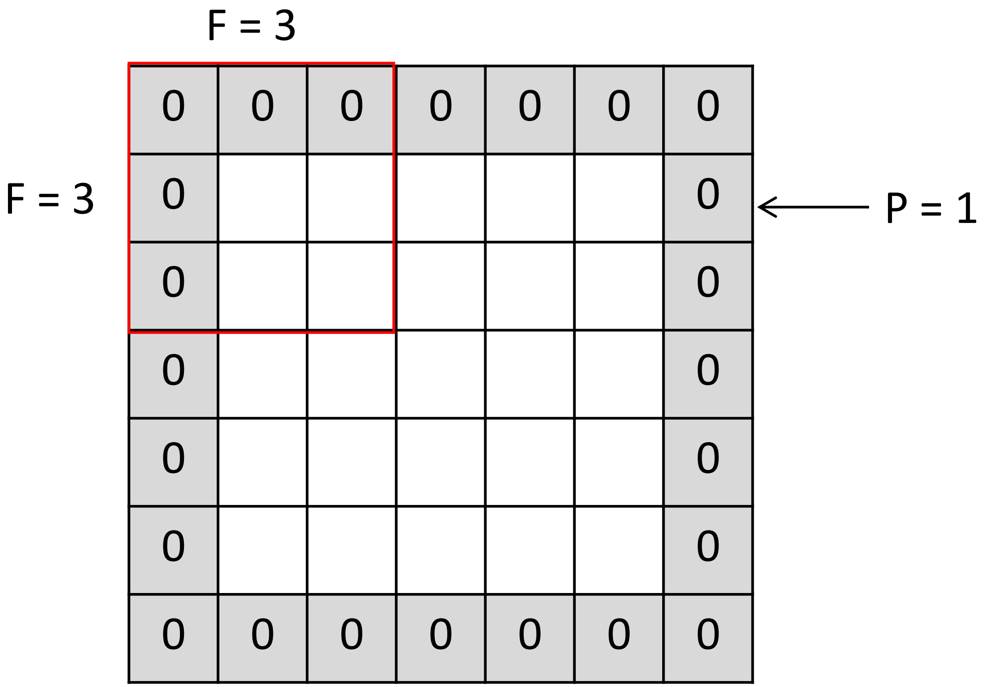
Fewer parameters

Even fewer parameters

Shared weights

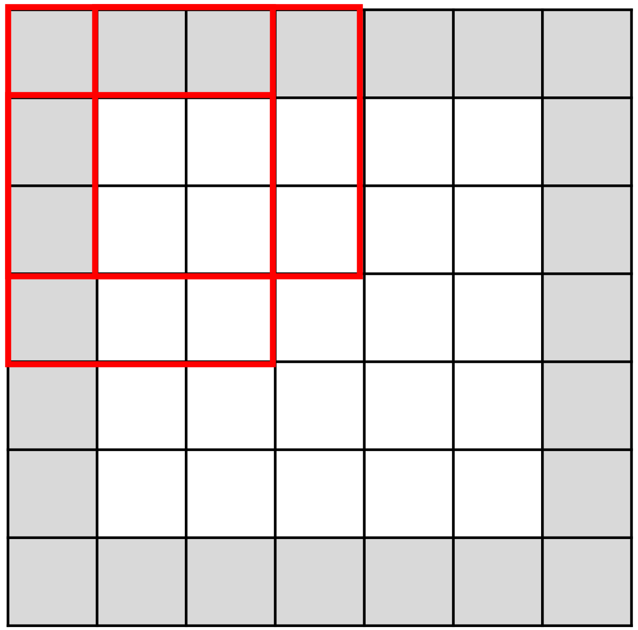
Zero Padding

- Adding zero padding will preserve the size spatially.
- In general, common to see CONV layers with stride 1, filter size F x F, and zero-padding with (F-1)/2.

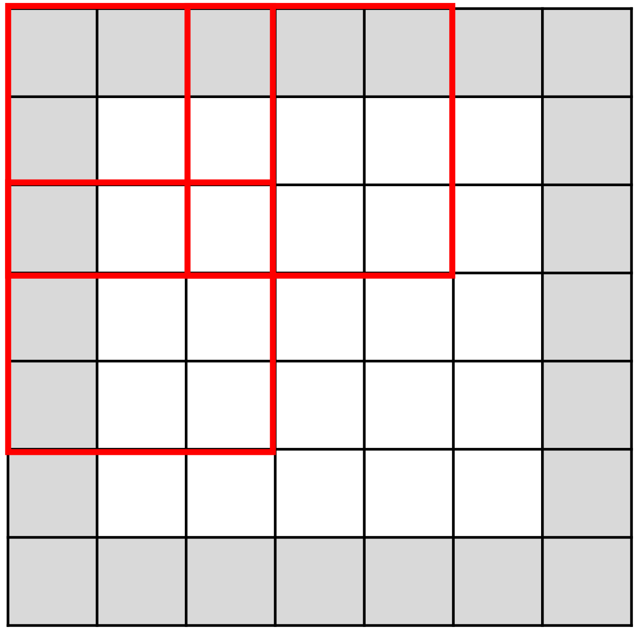


Stride

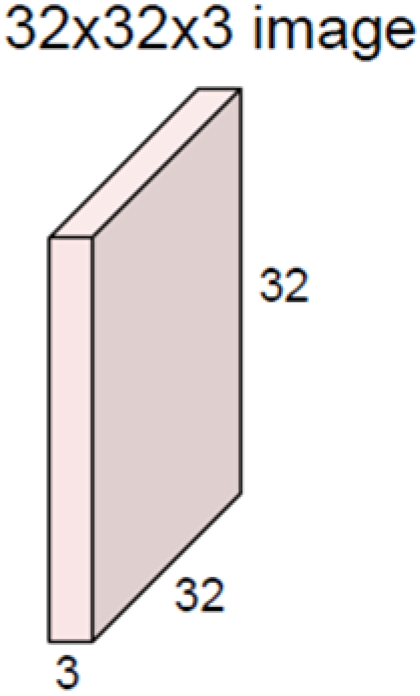
S = 1

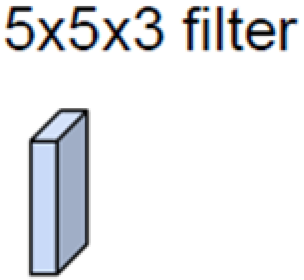


S = 2



Convolution Layer – Size of Activation Maps (Output Volume Size)

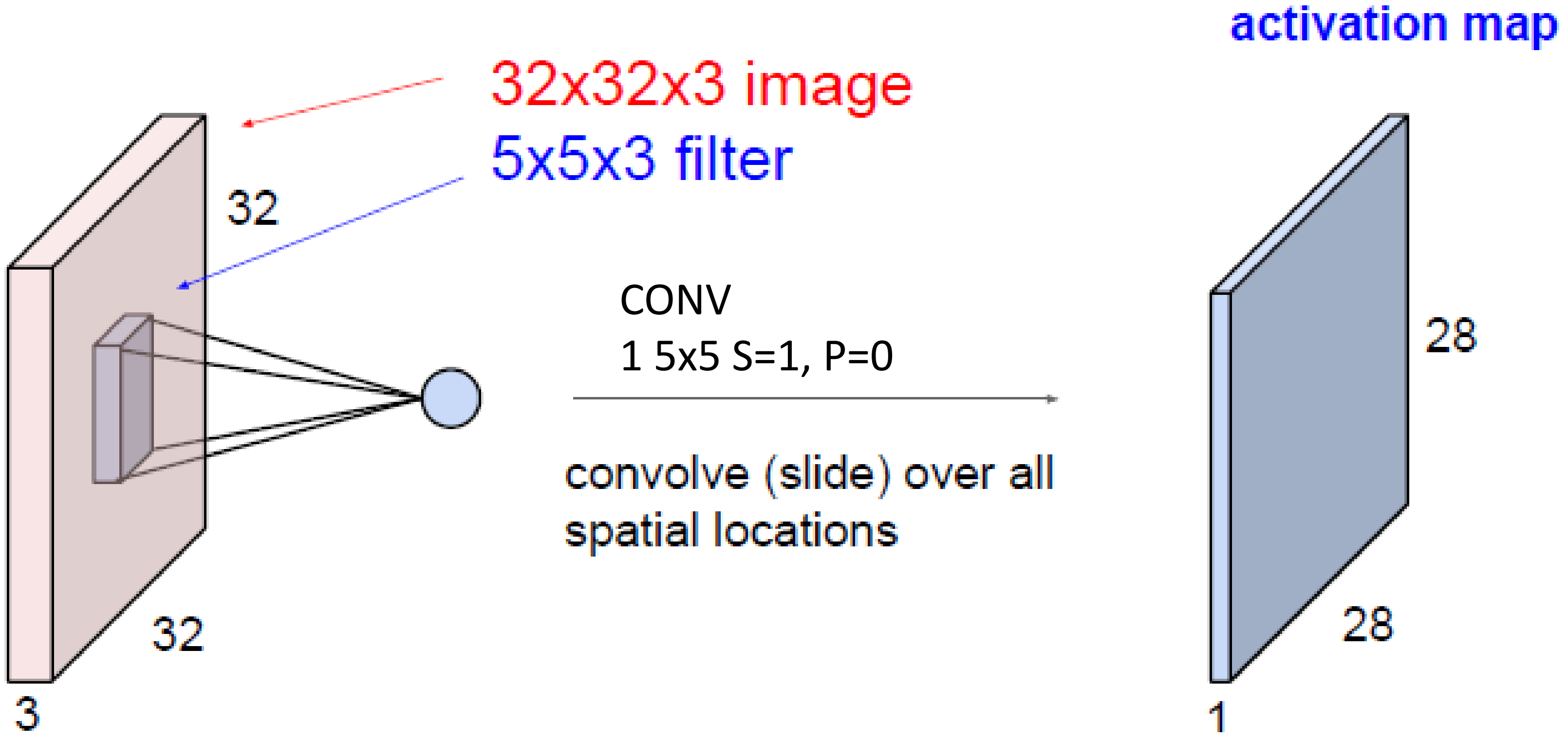




Since the image is 3-dimensional, the filter is also 3-dimensional

That is depth of the input = depth of the filter

Convolution Layer – Size of Activation Maps (Output Volume Size)



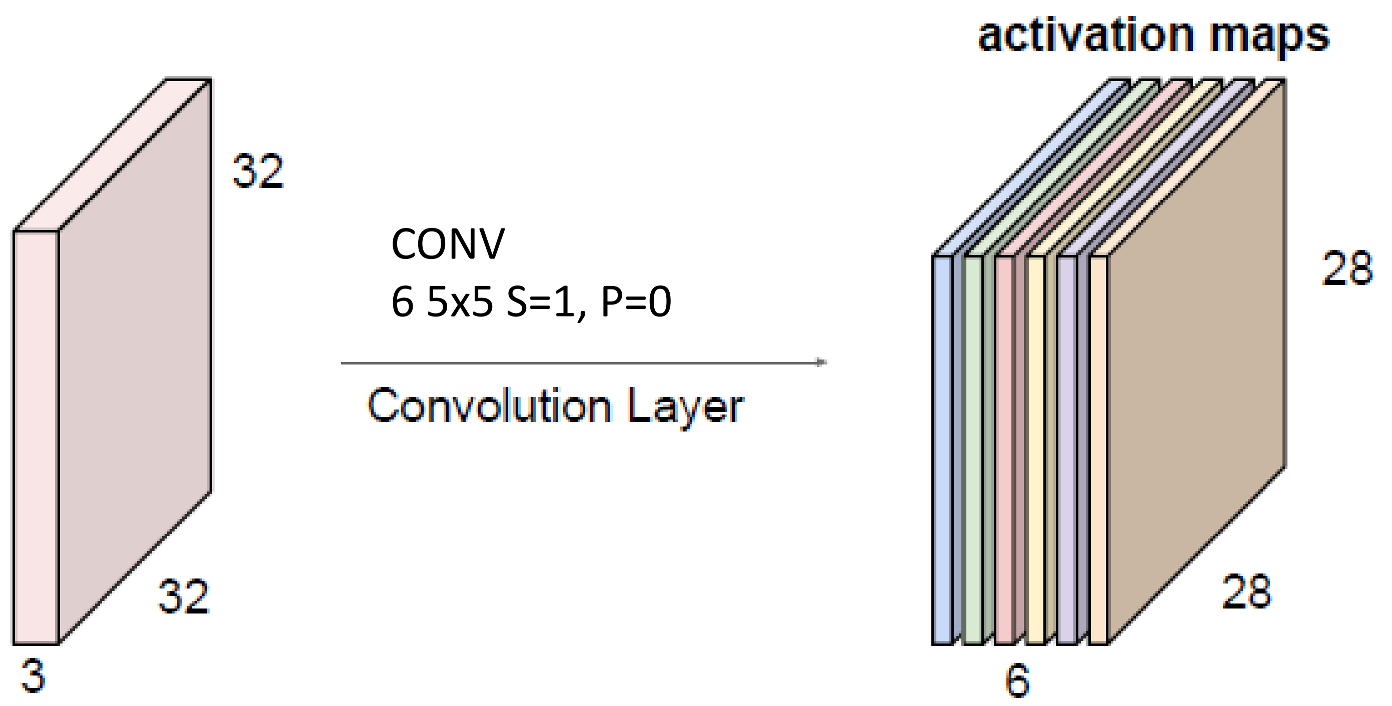
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

parameters = weights

Convolution Layer – Size of Activation Maps (Output Volume Size)

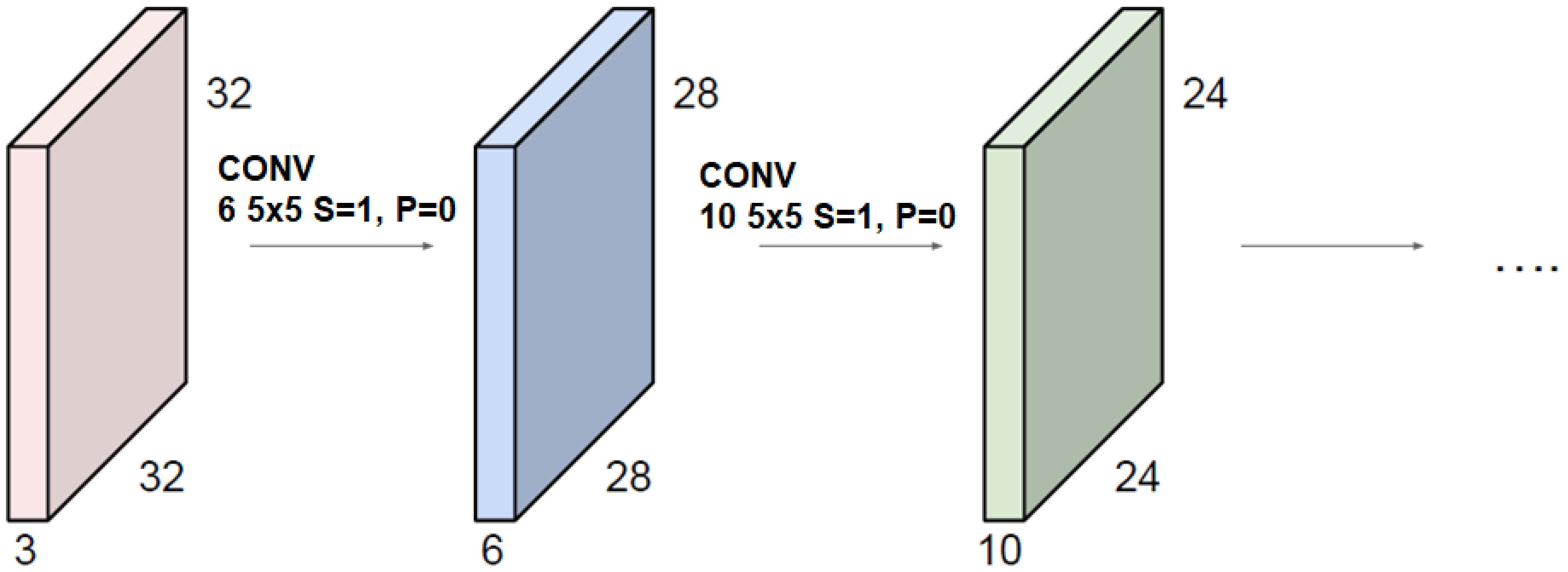
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



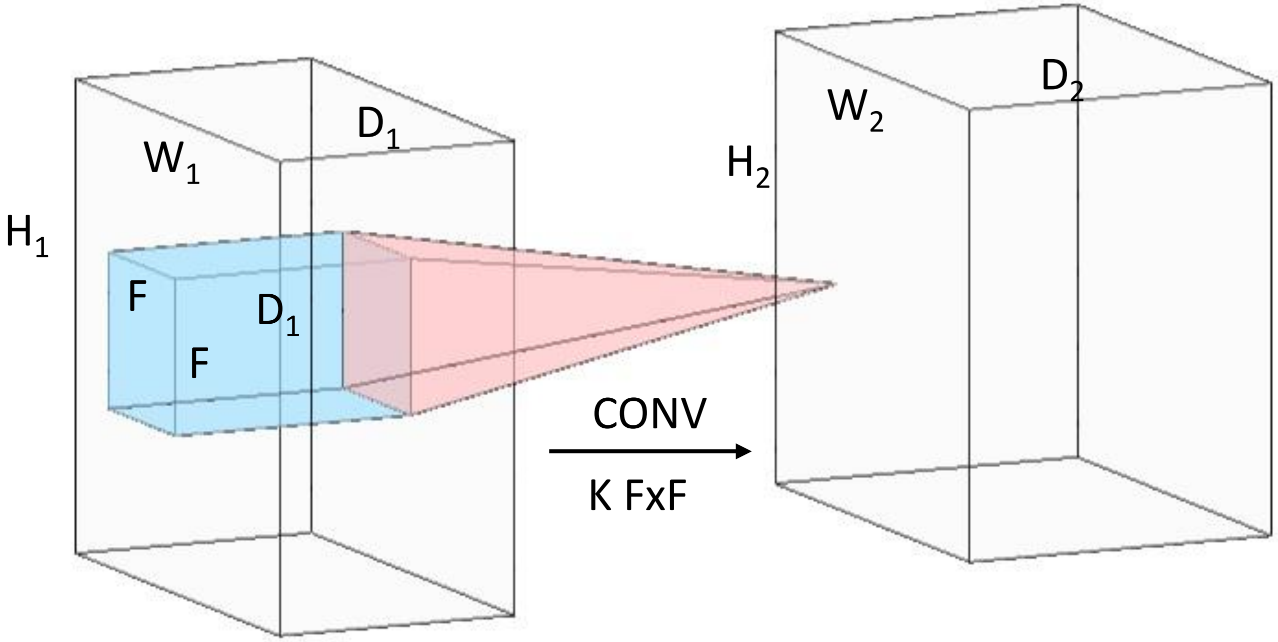
We stack these up to get a "new image" of size 28x28x6!

Convolution Layer – Size of Activation Maps (Output Volume Size)

- CNN is a sequence of Convolutional Layers, interspersed with activation functions



Calculating the Output Volume Size and Other Parameters



Filter Extent = F
Stride = S
Padding = P

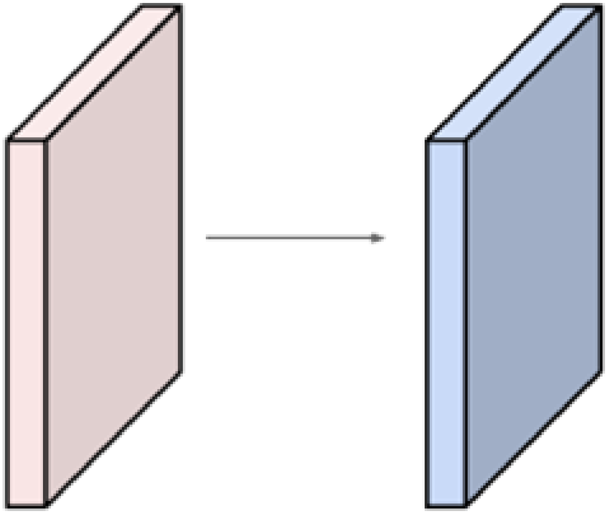
Produces a volume of size $W_2 x H_2 x D_2$ where:

$W_2 = (W_1\text{-}F+2P)/S +1$
$H_2 = (H_1\text{-}F+2P)/S + 1$
$D_2 = K$

With parameter sharing, it introduces $F.F.D_1$ weights per filter, for a total of $(F.F.D_1)$. K weights and k biases

Convolution Layer – Size of Activation Maps (Output Volume Size)

- Example



Input Volume of 32x32x3
Stride = 1
Padding = 2

Apply 10 5x5 filters

$$(W_2=H_2) = \frac{32-5+2*2}{1}+1$$

$$= 32$$

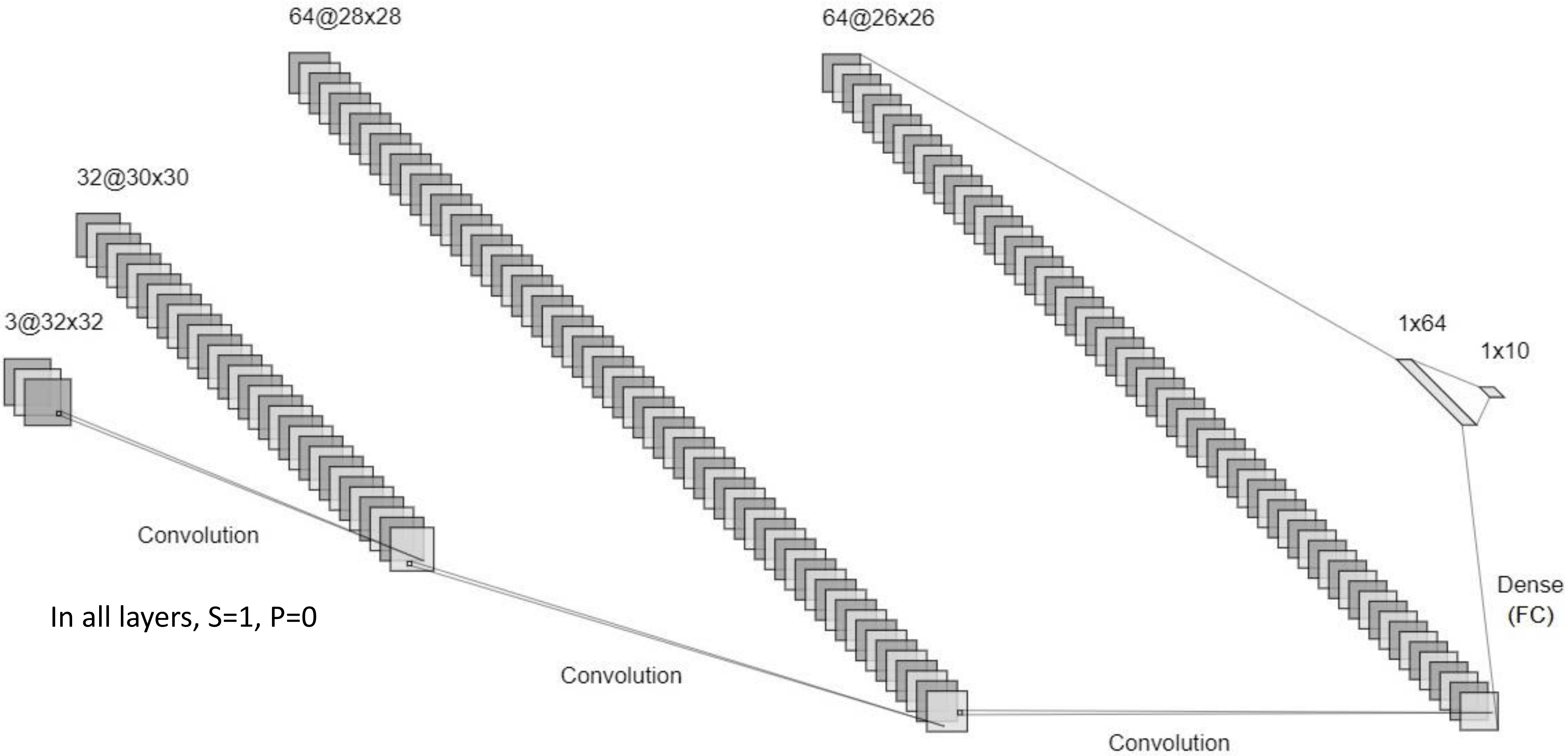
$D_2 = 10$

Output Volume Size = 32 x 32 x 10

Number of shared parameters (weights) per filter
= 5x5x3 + 1(bias) = 76

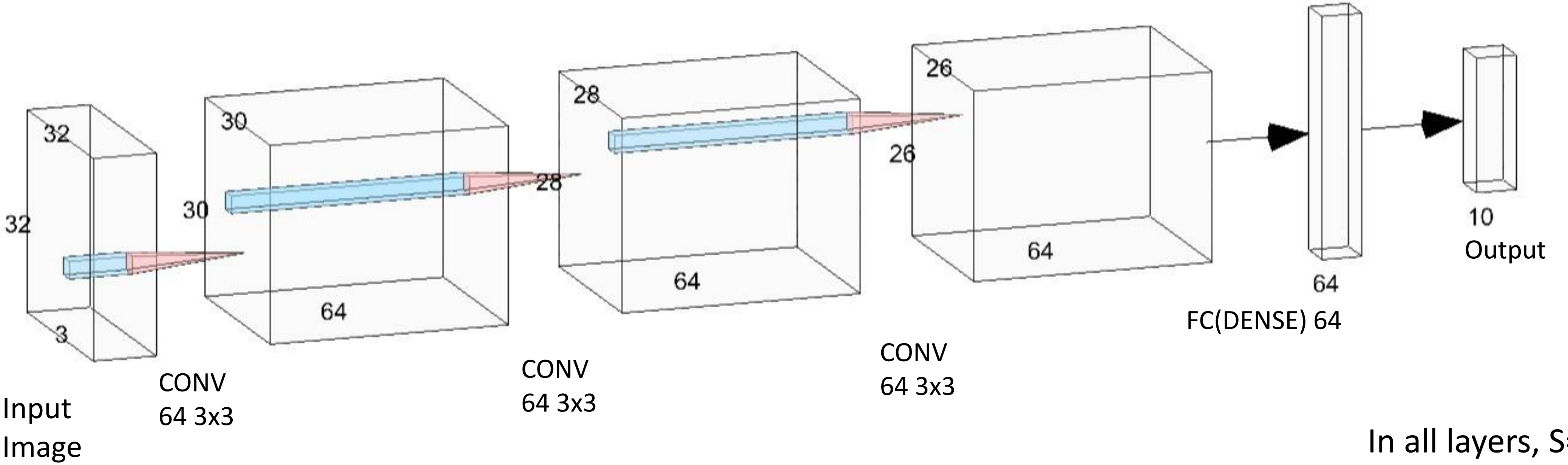
Total number of shared parameters (weights)
= 5x5x3 + 1(bias) = 76 x10 = 760

Example CNN – with CONV Layers Only



In all layers, S=1, P=0

Example CNN – with CONV Layers Only



In all layers, S=1, P=0

Same network with different Representation (AlexNet Style)

Pooling Layers

- Why Pooling: Subsampling pixels will not change the object

bird

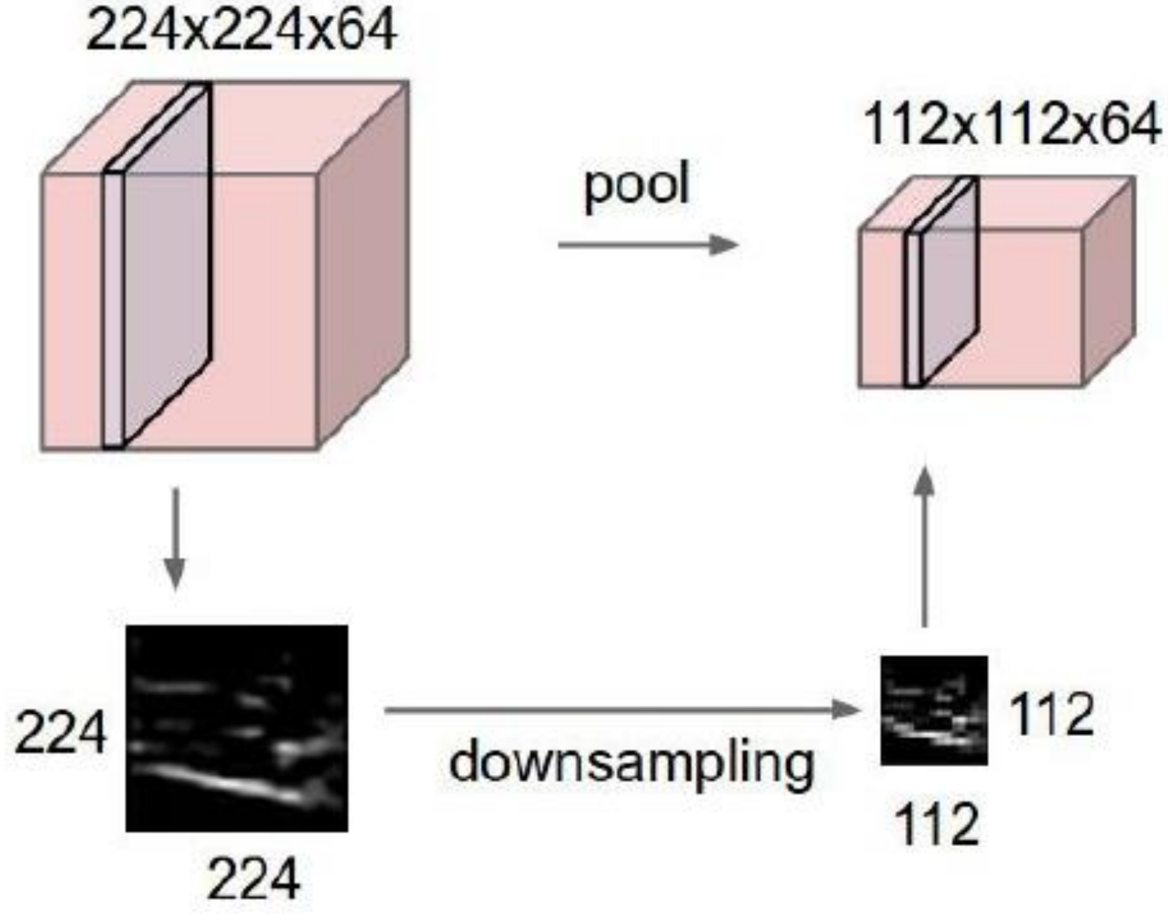
Subsampling
(Pooling)

bird

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

Pooling Layer

- makes the representations smaller and more manageable



Pooling layer

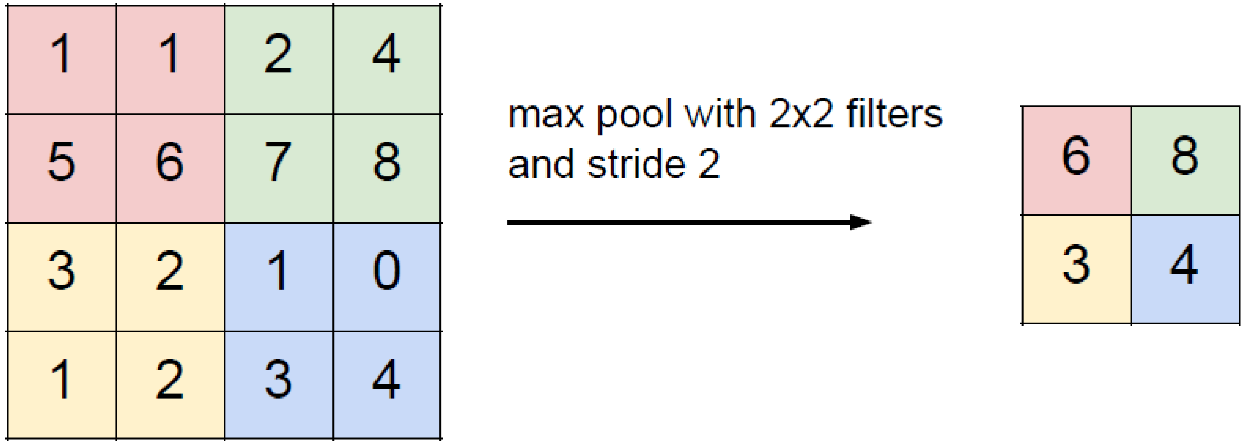
- Takes smaller blocks from convolutional layer
- Subsamples to produce single output from that block
- Several ways- average or maximum or learned linear combination of neurons
- For example, max pooling layers take maximum out of that block
- Pooling layers do not have trainable weights.

Types of Pooling

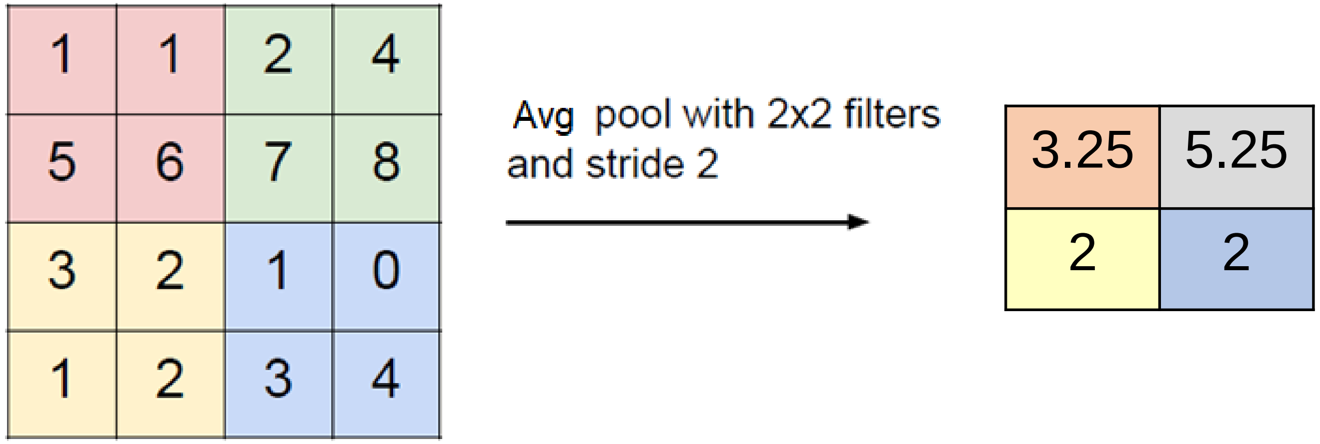
- Max pooling
- Average pooling
- Global max pooling
- Global average pooling

Pooling Layer

- Max Pooling



- Average Pooling

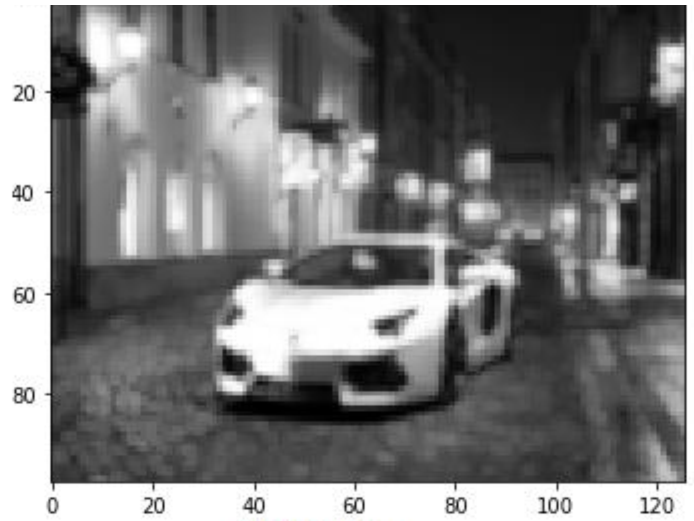


Effect of Pooling

Original Image

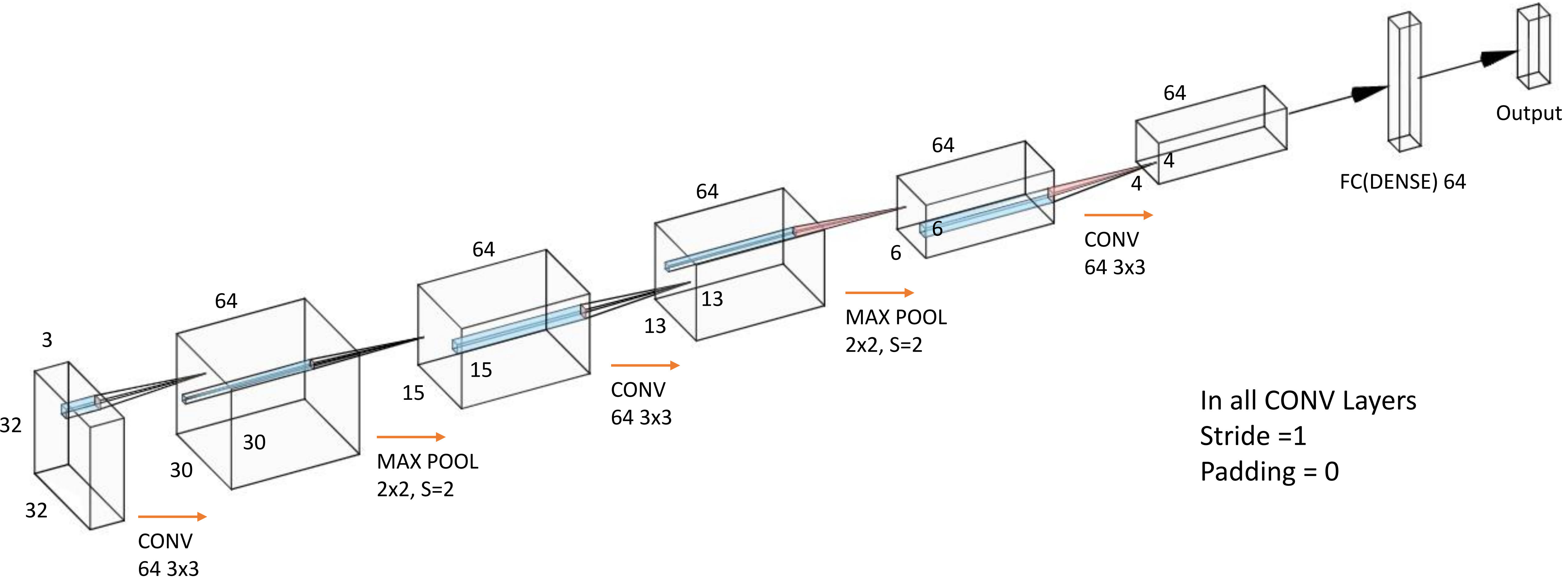
Convoluted image

Convoluted image



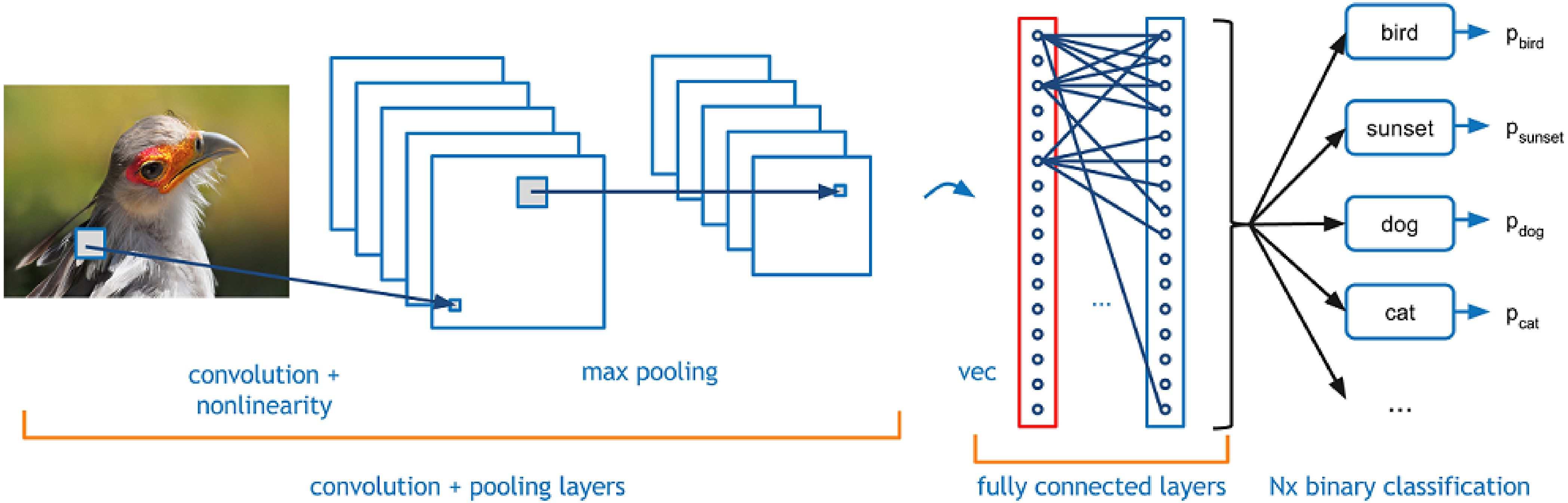
MAX Pooling

Example CNN – with CONV and Pooling Layers



In all CONV Layers
Stride =1
Padding = 0

Fully-connected layer

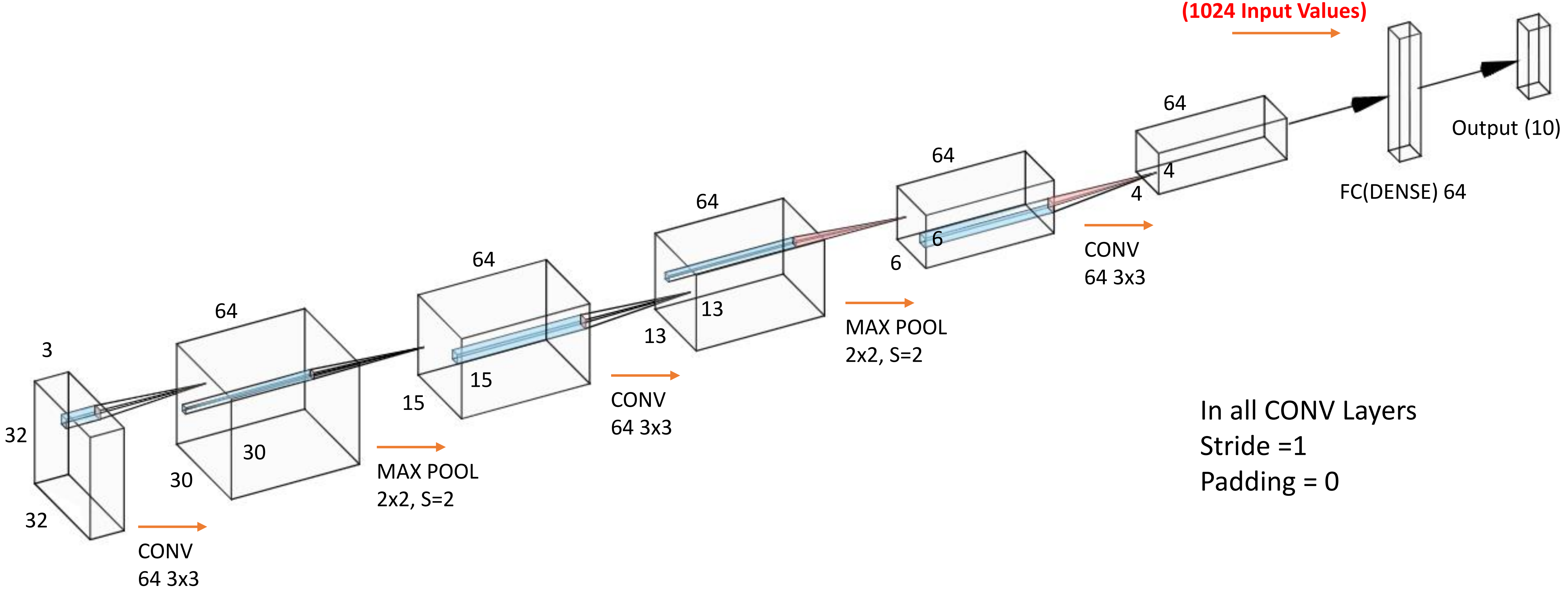


Fully-connected layer

- Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers.

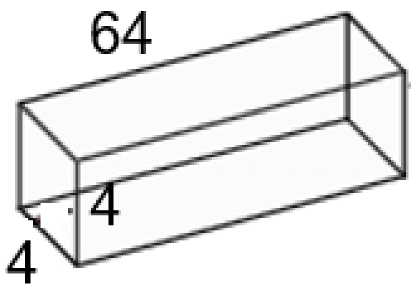
- Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks.

Example CNN

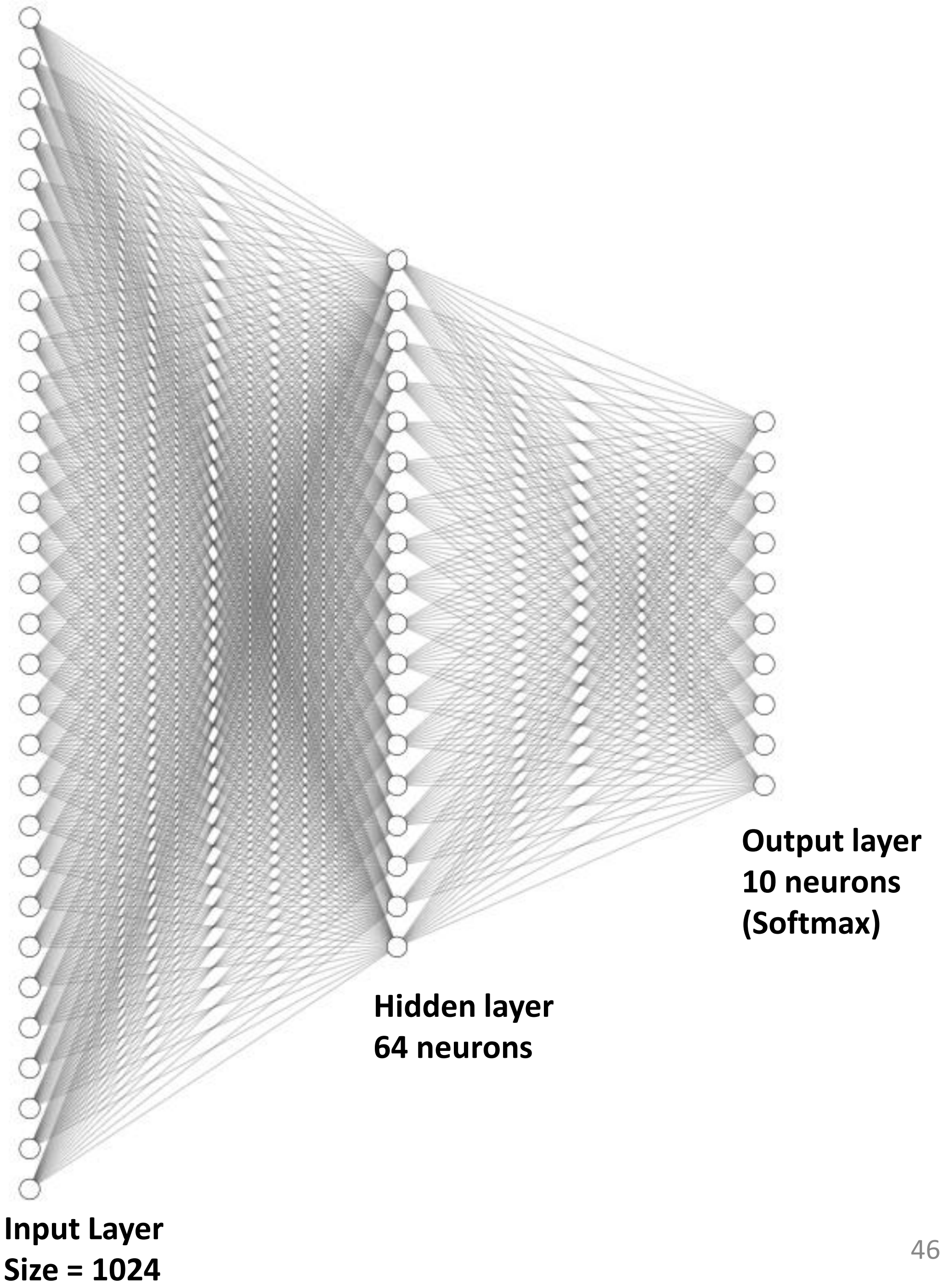


In all CONV Layers
Stride =1
Padding = 0

Fully-connected layer



Flattening



Last output volume will be flattened and use it for the input layer of the fully connected network.

In this case, flattening gives 64x4x4 = 1024 Input values which are the outputs of 1024 neurons of the output volume

Example Architecture – LeNet 5

- Exercise: Explain the architecture of LeNet-5