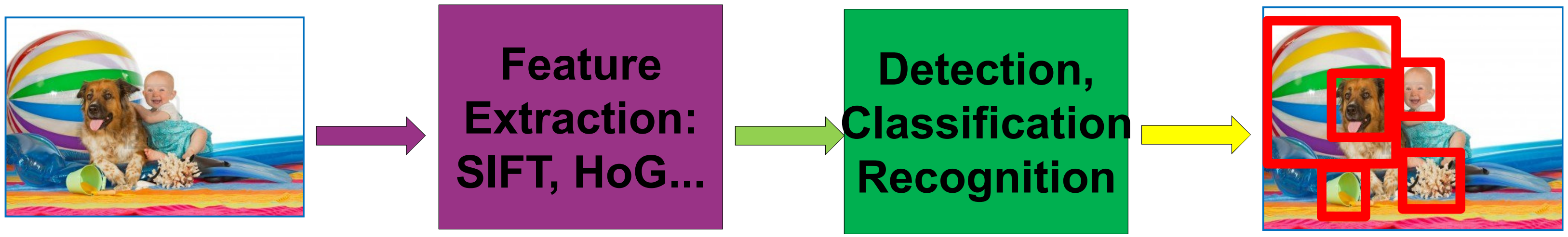
Image Source: deeplearning.ai

Computer Vision Problems

Classical Computer Vision Pipeline

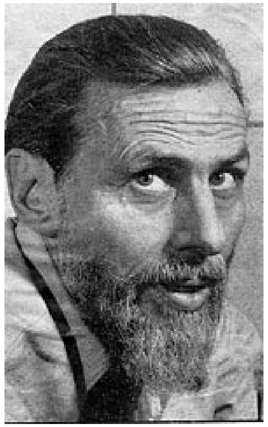
CV experts

1. Select / develop features: SURF, HoG, SIFT, RIFT, …
2. Add on top of this Machine Learning for multi-class recognition and train classifier

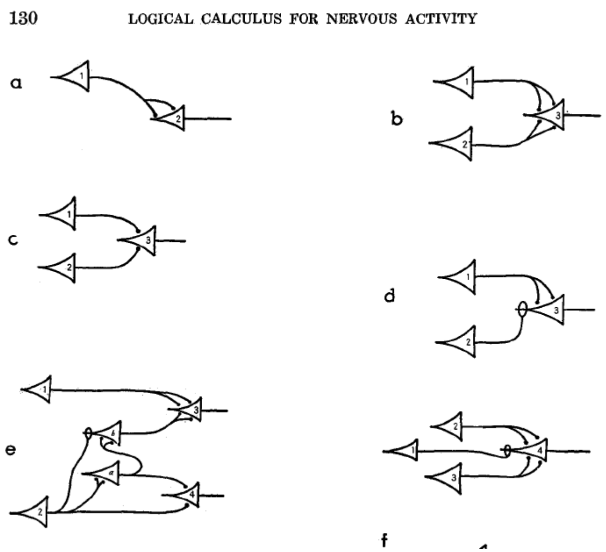


Classical CV feature definition is domain-specific and time-consuming

Neural Network



Warren McCulloch



Walter Pitts

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY, Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133 (1943).

Neural Network

Here x1 and x2 are normalized attribute value of data.

y is the output of the neuron , i.e the class label.

x1 and x2 values multiplied by weight values w1 and w2 are input to the neuron x.

Value of x1 is multiplied by a weight w1 and values of x2 is multiplied by a weight w2.

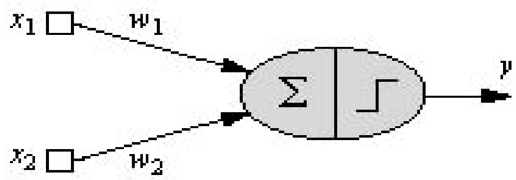


Fig1: an artificial neuron

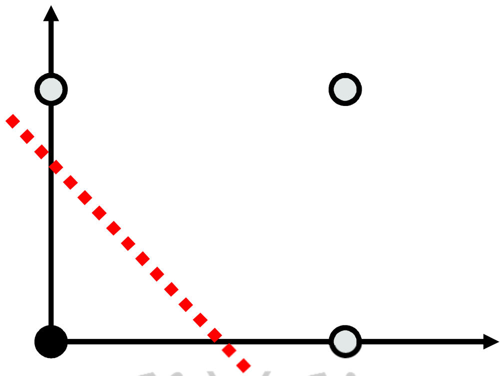
Given that

- w1 = 0.5 and w2 = 0.5
- Say value of x1 is 0.3 and value of x2 is 0.8,

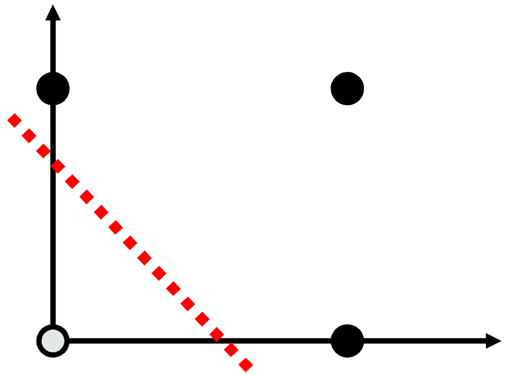
- So, weighted sum is :
- sum= w1 x x1 + w2 x x2 = 0.5 x 0.3 + 0.5 x 0.8 = 0.55

Why We Need Multi Layer ?

Linear Separable:

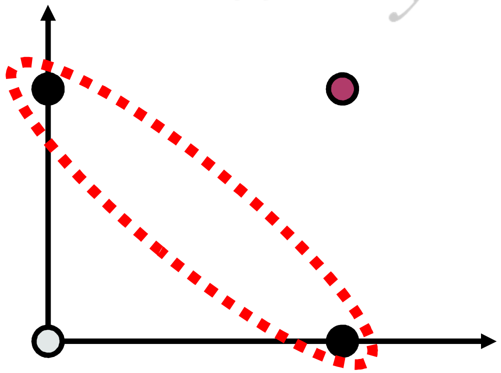


$x \vee y$



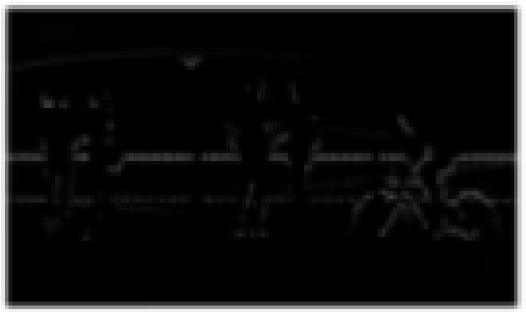
$x \wedge y$

Linear inseparable: $x \veebar y$



Edge Detection

Vertical edges



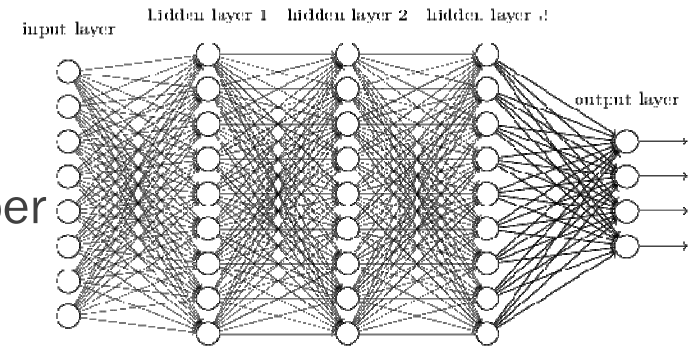
Horizontal edges

How do we detect these edges

Image Source: deeplearning.ai

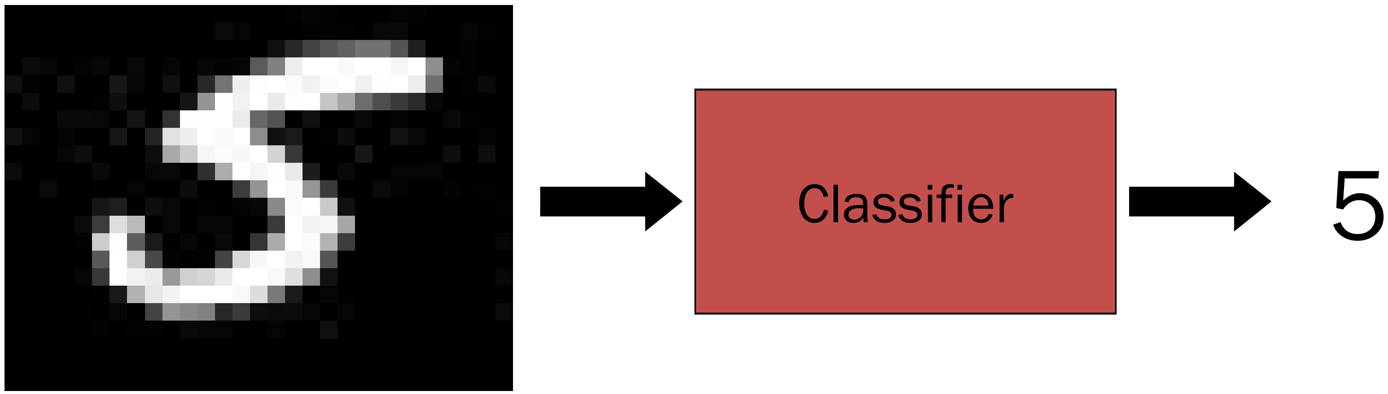
Neural Network?

- Suppose an image is of the size 68 X 68 X 3
  - Input feature dimension then becomes 12,288
- If Image size is of 720 X 720 X 3
  - Input feature dimension becomes 1,555,200
- Number of parameters will swell up to a HUGE number
- Result in more computational and memory requirements



Another Application

Digit Recognition



$X_1,\ldots,X_n \in \{0,1\}$ (Black vs. White pixels)

$Y \in \{5,6\}$ (predict whether a digit is a 5 or a 6)

The Bayes Classifier

In class, we saw that a good strategy is to predict:

$$\arg\max_{Y} P(Y|X_1,\ldots,X_n)$$

- (for example: what is the probability that the image represents a 5 given its pixels?)

So … how do we compute that?

The Bayes Classifier

Use Bayes Rule!

Likelihood Prior

$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

Normalization Constant

Why did this help? Well, we think that we might be able to specify how features are "generated" by the class label

The Bayes Classifier

Let’s expand this for our digit recognition task:

$$
\begin{aligned}
P(Y=5|X_1,\ldots,X_n) &= \frac{P(X_1,\ldots,X_n|Y=5)P(Y=5)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)} \\
P(Y=6|X_1,\ldots,X_n) &= \frac{P(X_1,\ldots,X_n|Y=6)P(Y=6)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}
\end{aligned}
$$

To classify, we’ll simply compute these two probabilities and predict based on which one is greater

Model Parameters

For the Bayes classifier, we need to “learn” two functions, the likelihood and the prior

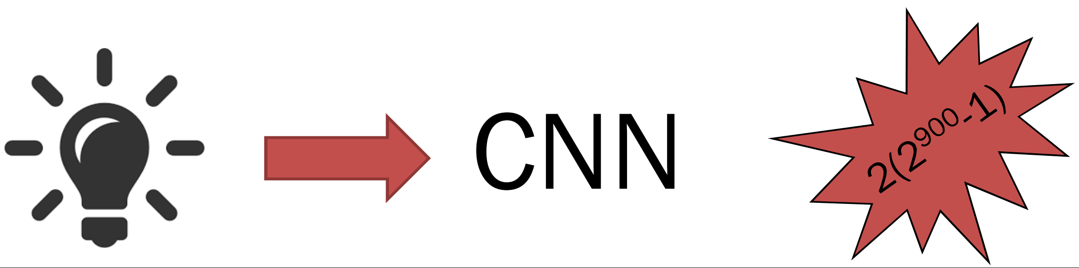
How many parameters are required to specify the prior for our digit recognition example?

Just 1

Model Parameters

How many parameters are required to specify the likelihood?

- (Supposing that each image is 30x30 pixels)



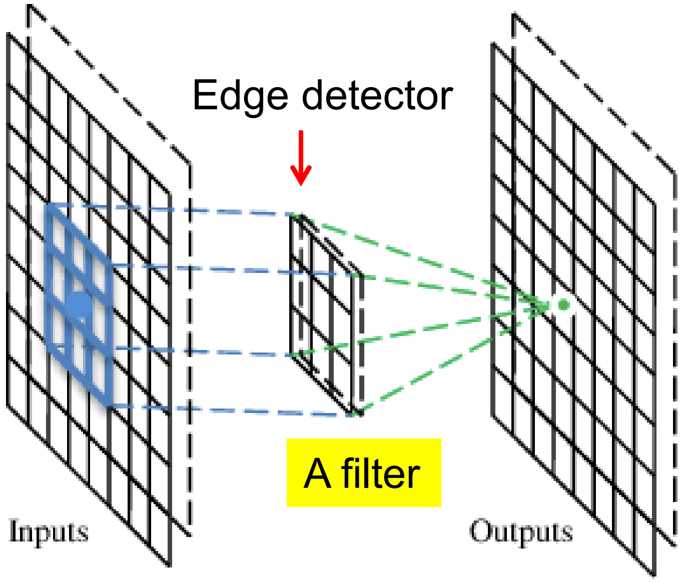
Drive into CNN

In a convolutional network (ConvNet), there are basically three types of layers:

1.Convolution layer
2.Pooling layer
3.Fully connected layer

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).

Image Source: internet

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

Dot product → 3 -1

6 x 6 image

Image Source: internet

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

Image Source: internet

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

Image Source: internet

Convolution

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

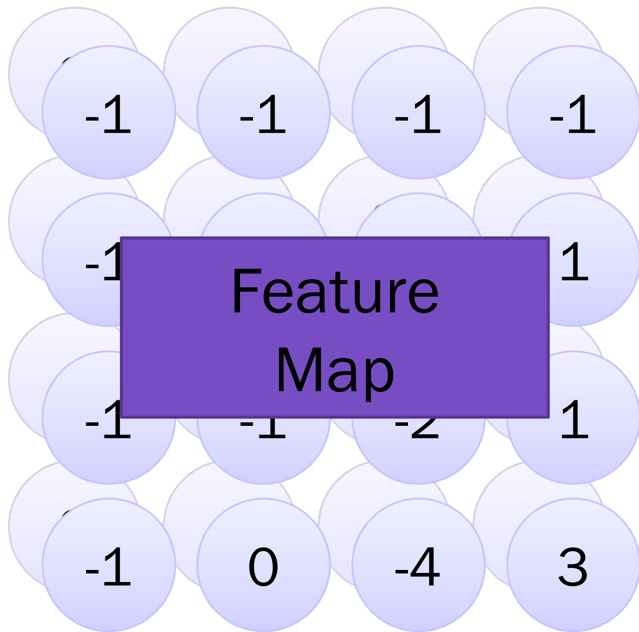
Filter 2

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

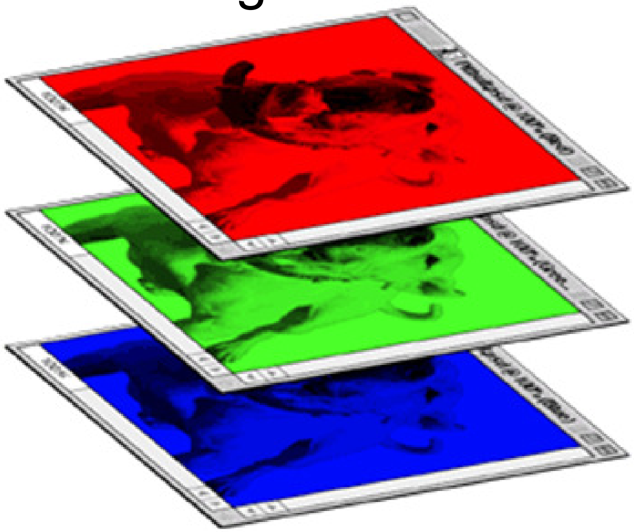


Two 4 x 4 images
Forming 4 x 4 x 2 matrix

Image Source: internet

Convolution over Volume

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

Filter 1

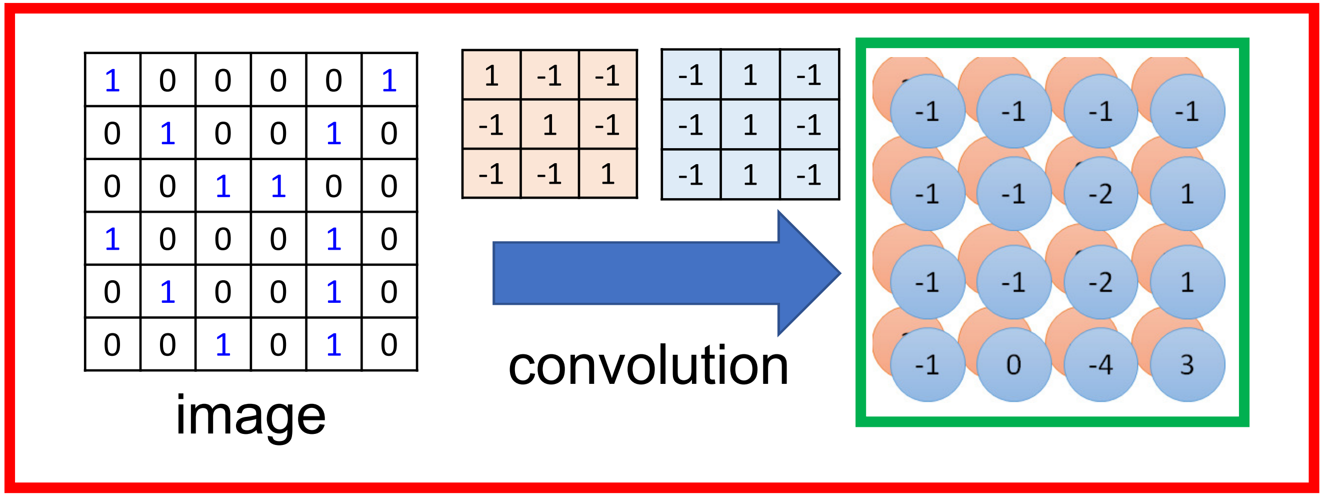
1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

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

Image Source: internet

Convolution v.s. Fully Connected



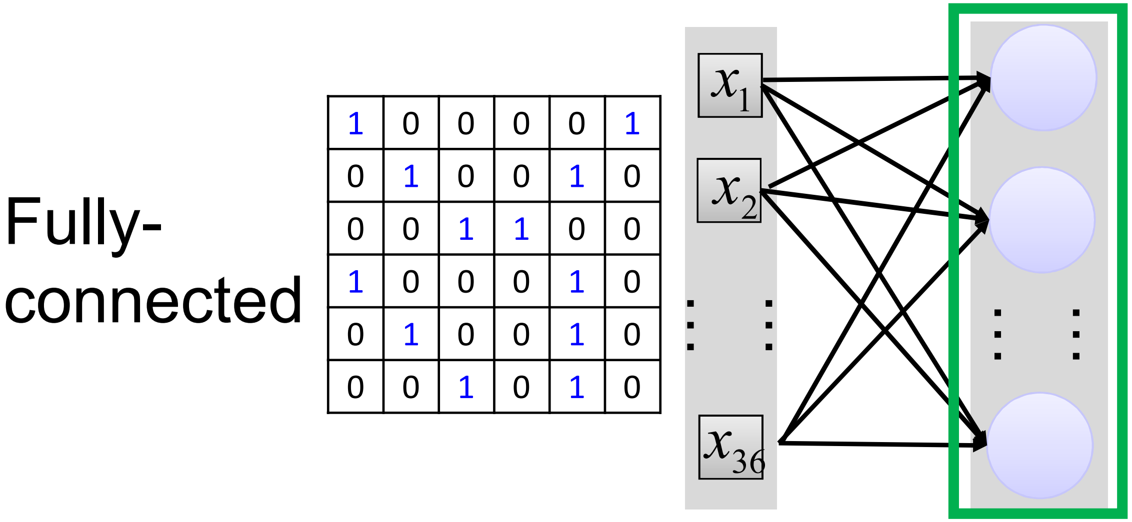


Image Source: internet

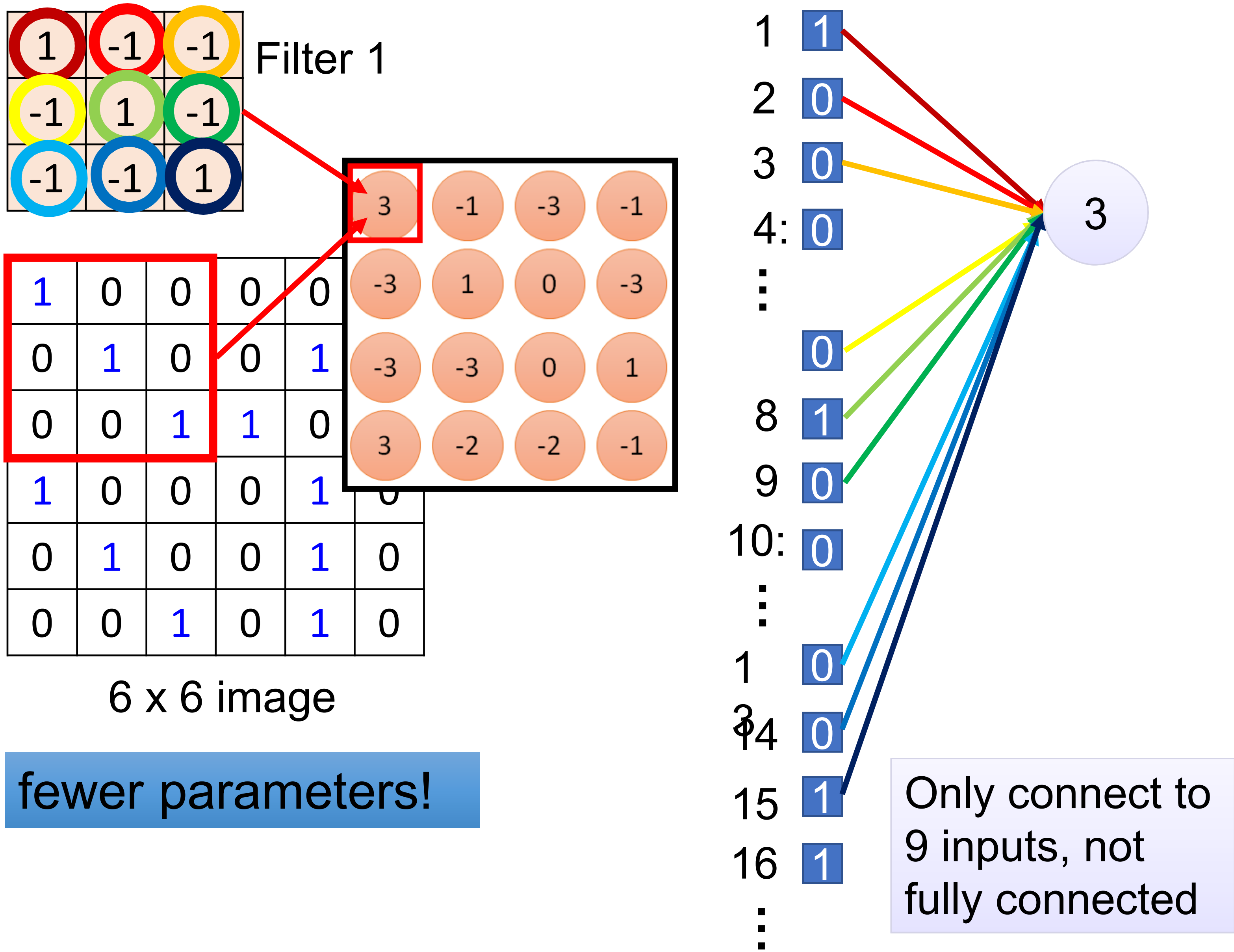


Image Source: internet

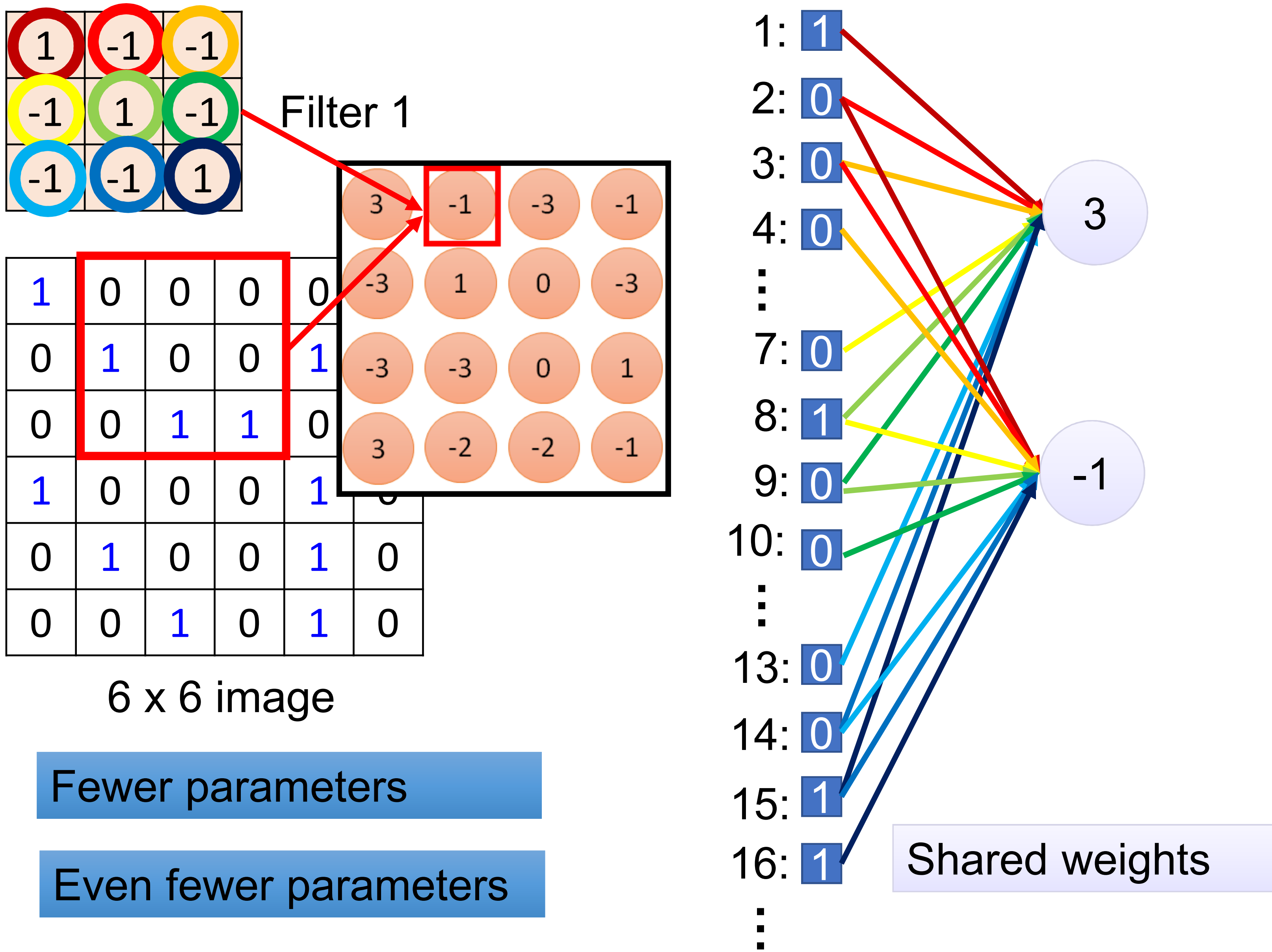


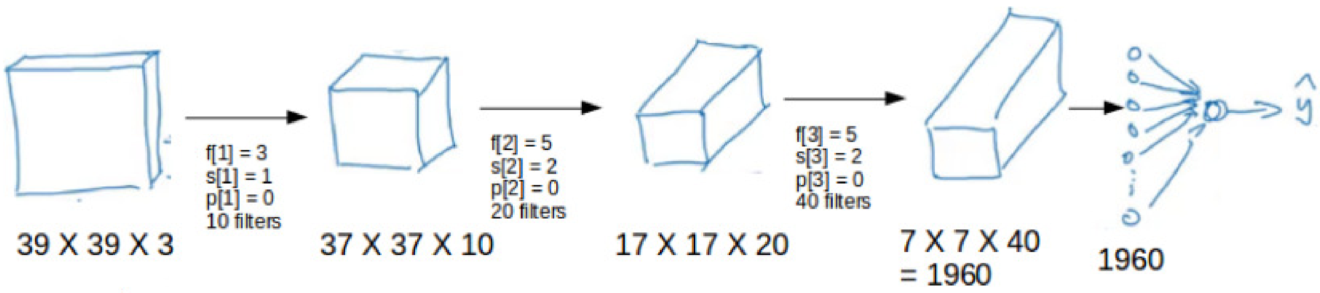
Image Source: internet

Suppose we have 10 filters applying on input (6 X 6 X 3), each of shape 3 X 3 X 3. What will be the number of parameters in that layer?

- Number of parameters for each filter = 3*3*3 = 27
- There will be a bias term for each filter, so total parameters per filter = 28
- As there are 10 filters, the total parameters for that layer = 28*10 = 280

Image Source: internet

Simple Convolutional Neural Network



- **Size of feature vector : (n+2p-f)/s +1**
- **n : dimension of matrix**
- **p : size of padding**
- **f : size of filter**
- **s : size of stride**

Image Source: deeplearning.ai

The whole CNN

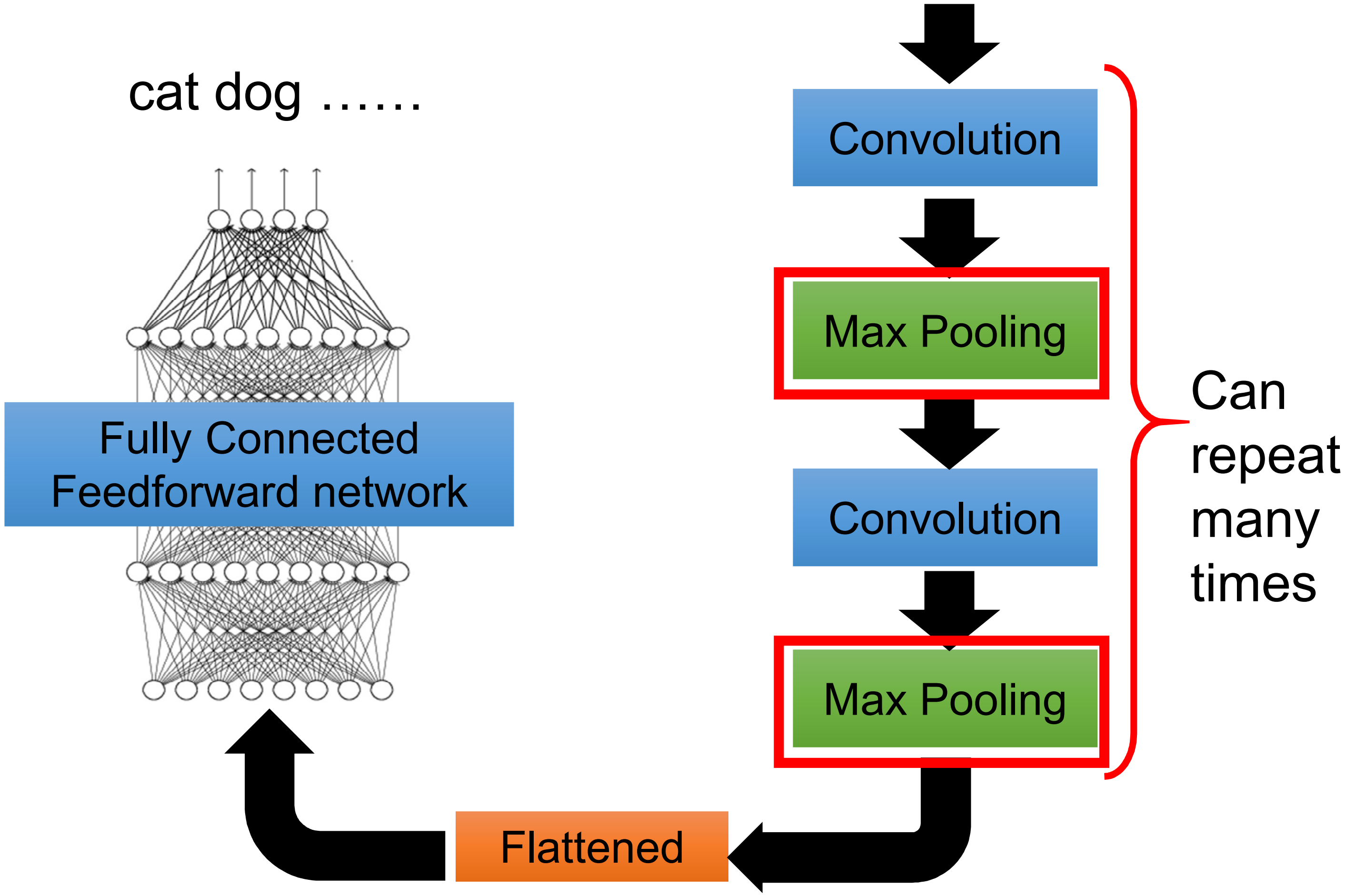


Image Source: internet

Max Pooling

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

Filter 1

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

Filter 2

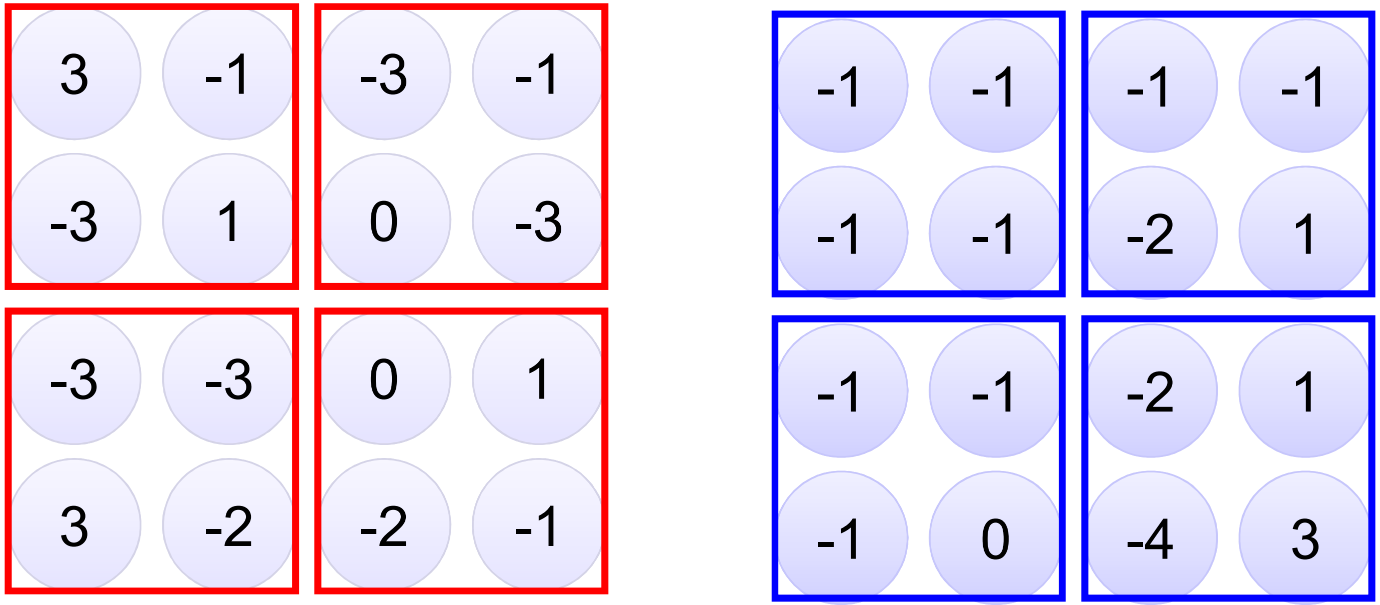
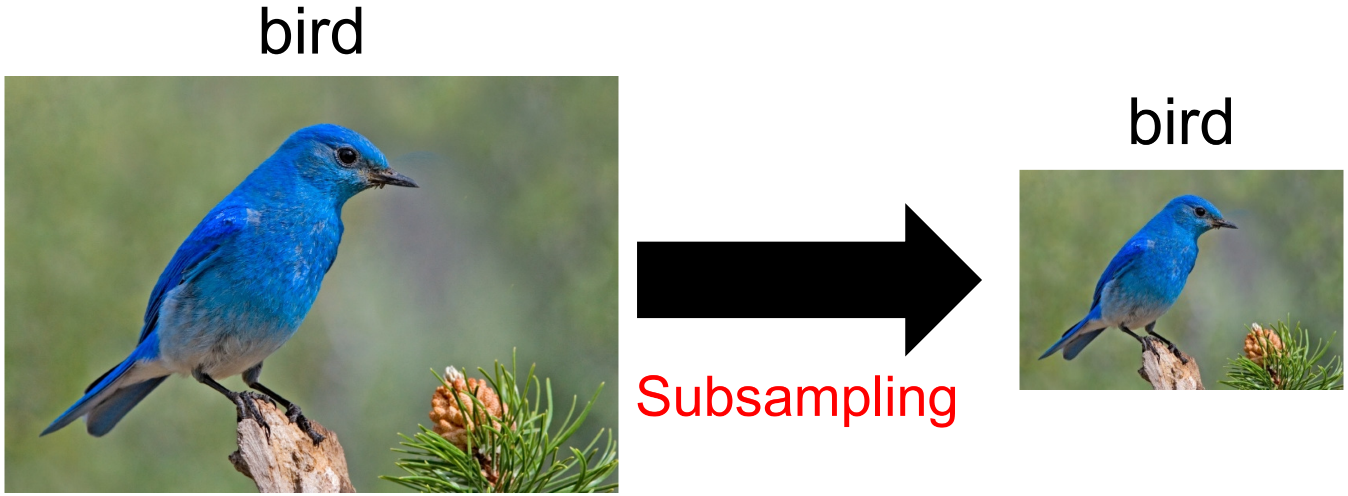


Image Source: internet

Why Pooling

- Subsampling pixels will not change the object



We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

Image Source: internet

A CNN compresses a fully connected network in two ways:

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

Max Pooling

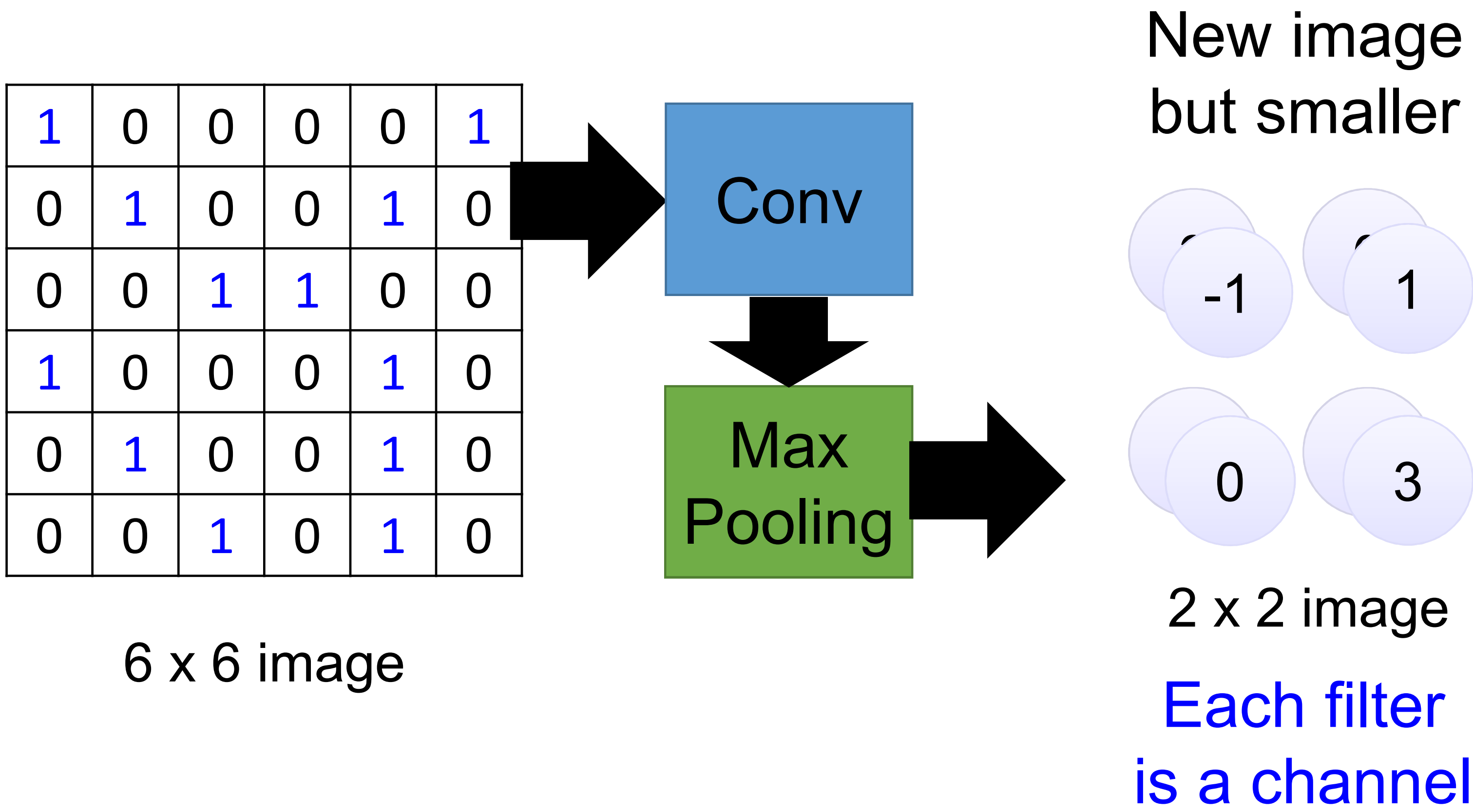
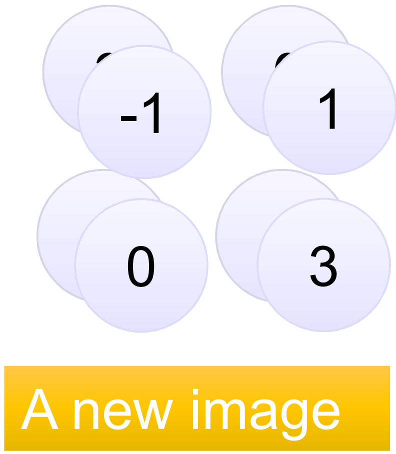
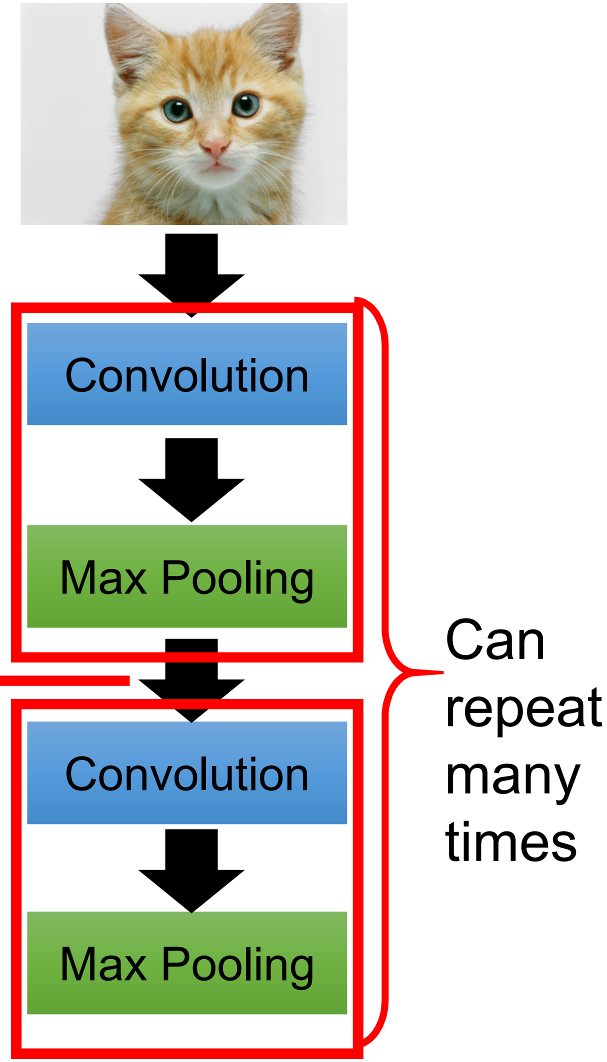


Image Source: internet

The whole CNN





Smaller than the original image

The number of channels is the number of filters

Image Source: internet

The whole CNN

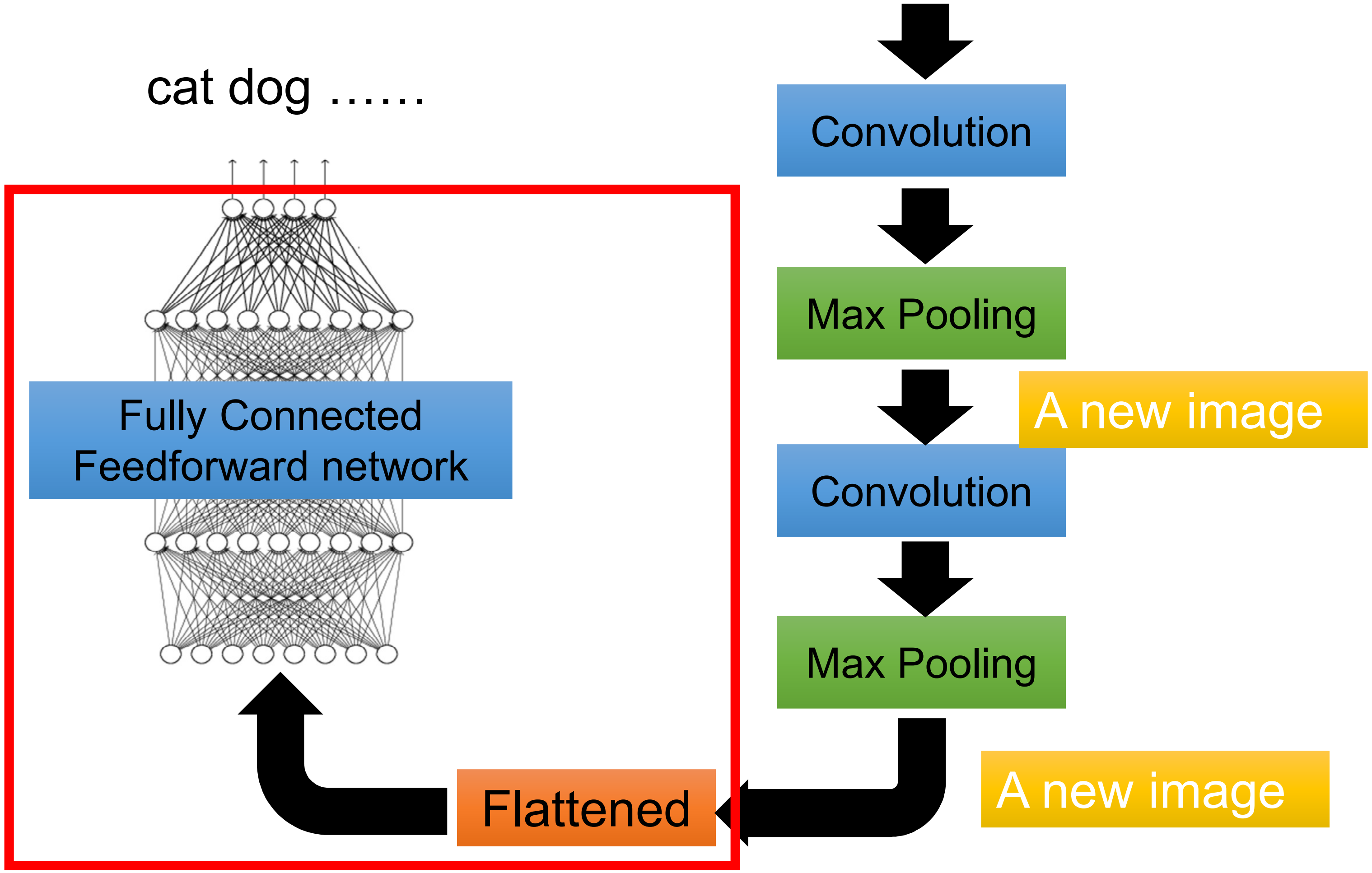


Image Source: internet

Flattening

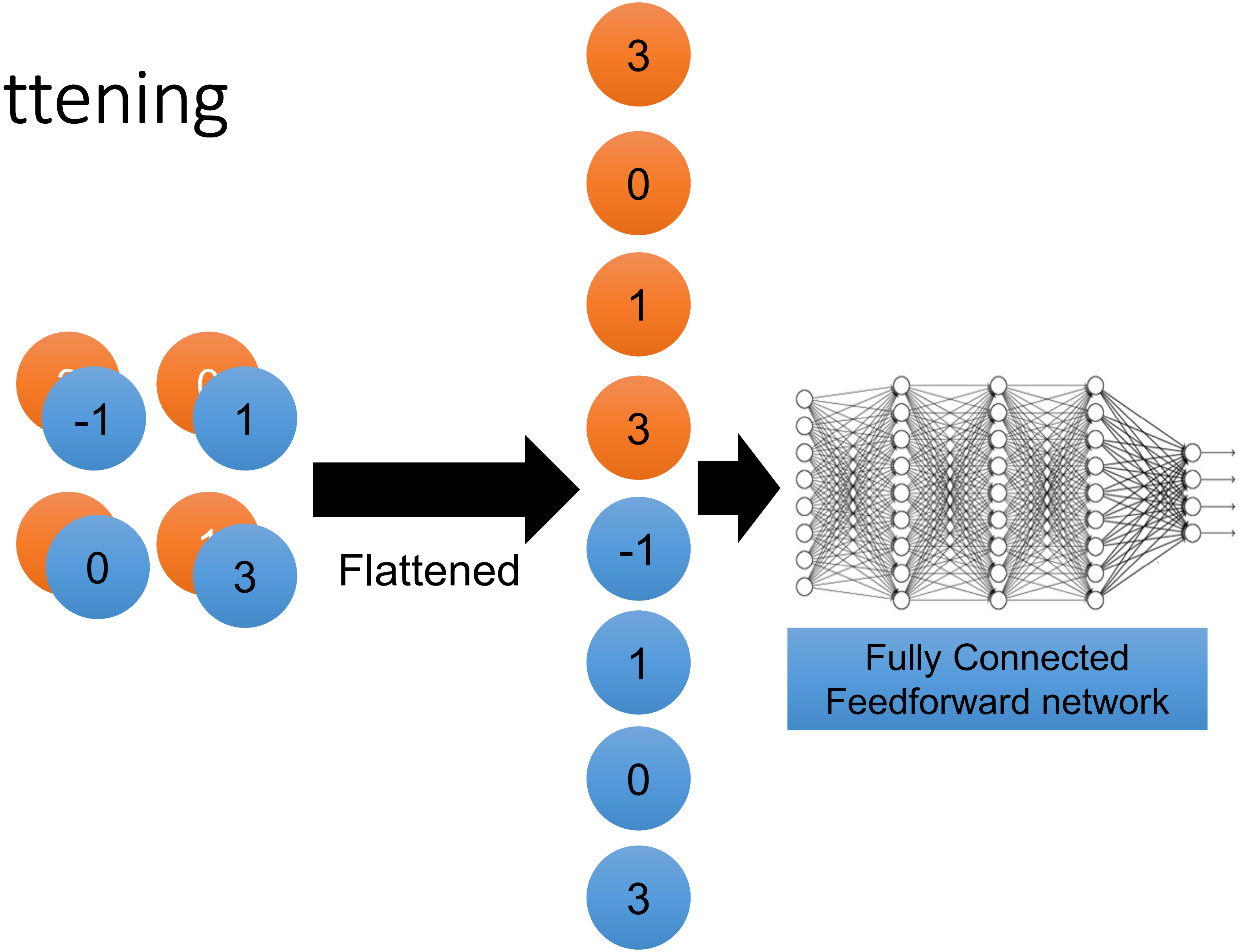
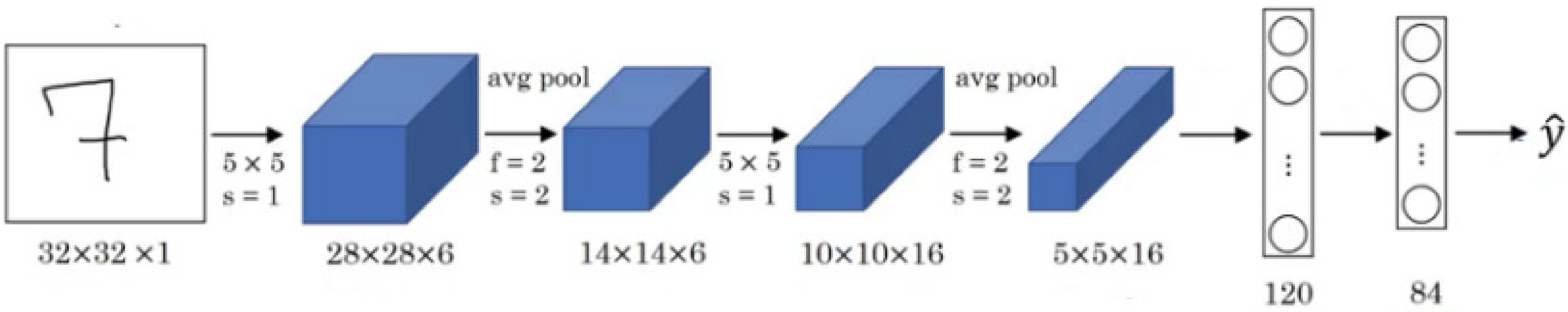


Image Source: internet

Classic Networks

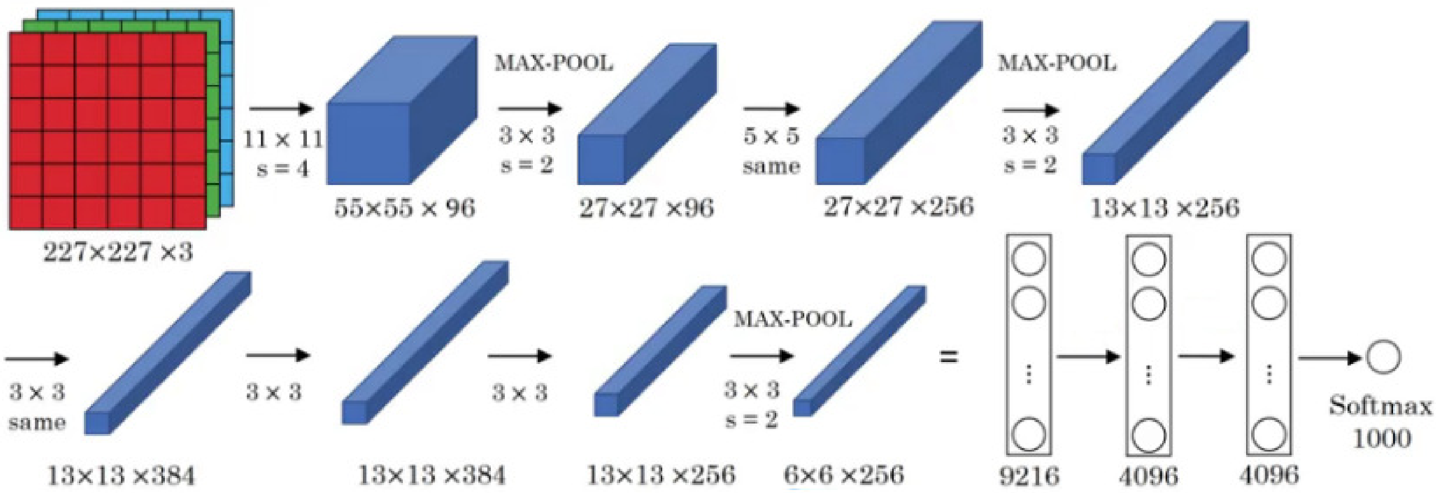
1. LeNet-5
2. AlexNet
3. VGG

LeNet-5



- **Parameters:** 60k
- **Layers flow:** Conv → Pool → Conv → Pool → FC → FC → Output
- **Activation functions:** Sigmoid/tanh and ReLU

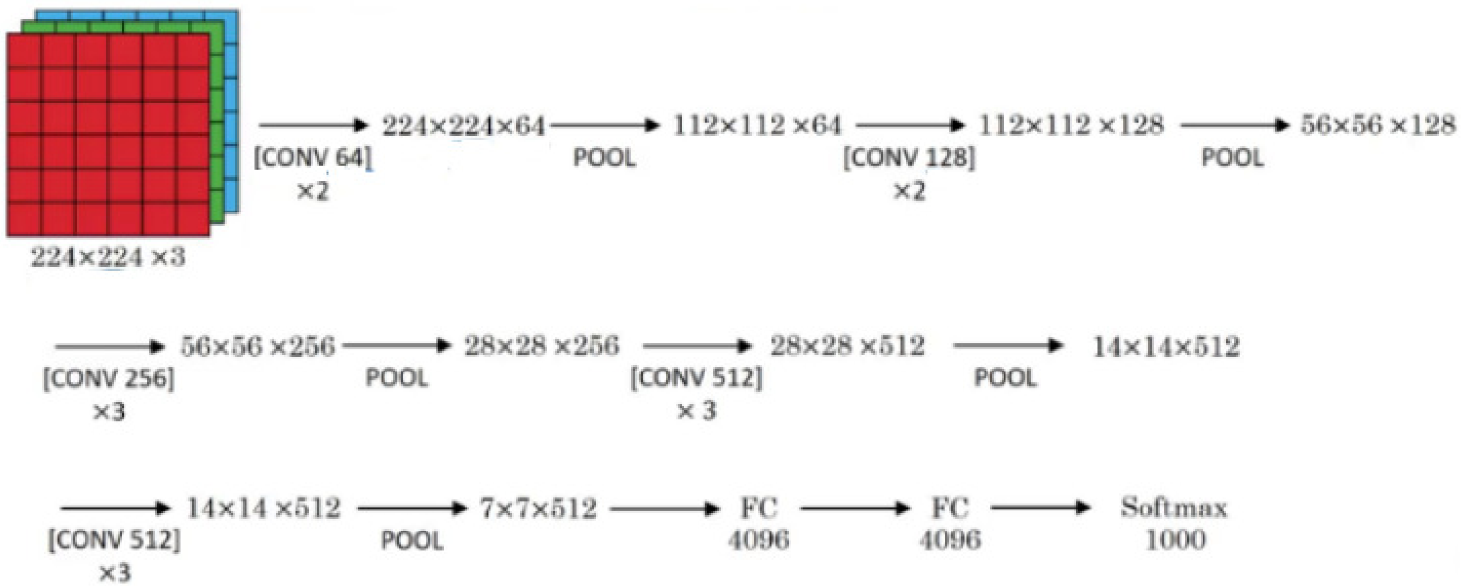
AlexNet



- **Parameters:** 60 million
- **Activation functions:** ReLU

Image Source: deeplearning.ai

VGG-16



- **Parameters:** 138 million
- **Pool:** MAX with stride 2
- **CONV layer**: stride 1

Image Source: deeplearning.ai

CNN in Keras

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

Input

1 x 28 x 28

Convolution

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

How many parameters for each filter? **9**

25 x 26 x 26

Max Pooling

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

25 x 13 x 13

Convolution

```
model2.add(Convolution2D(50,3,3))
```

How many parameters for each filter? **225= 25x9**

50 x 11 x 11

Max Pooling

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

50 x 5 x 5

Image Source: internet

CNN in Keras

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

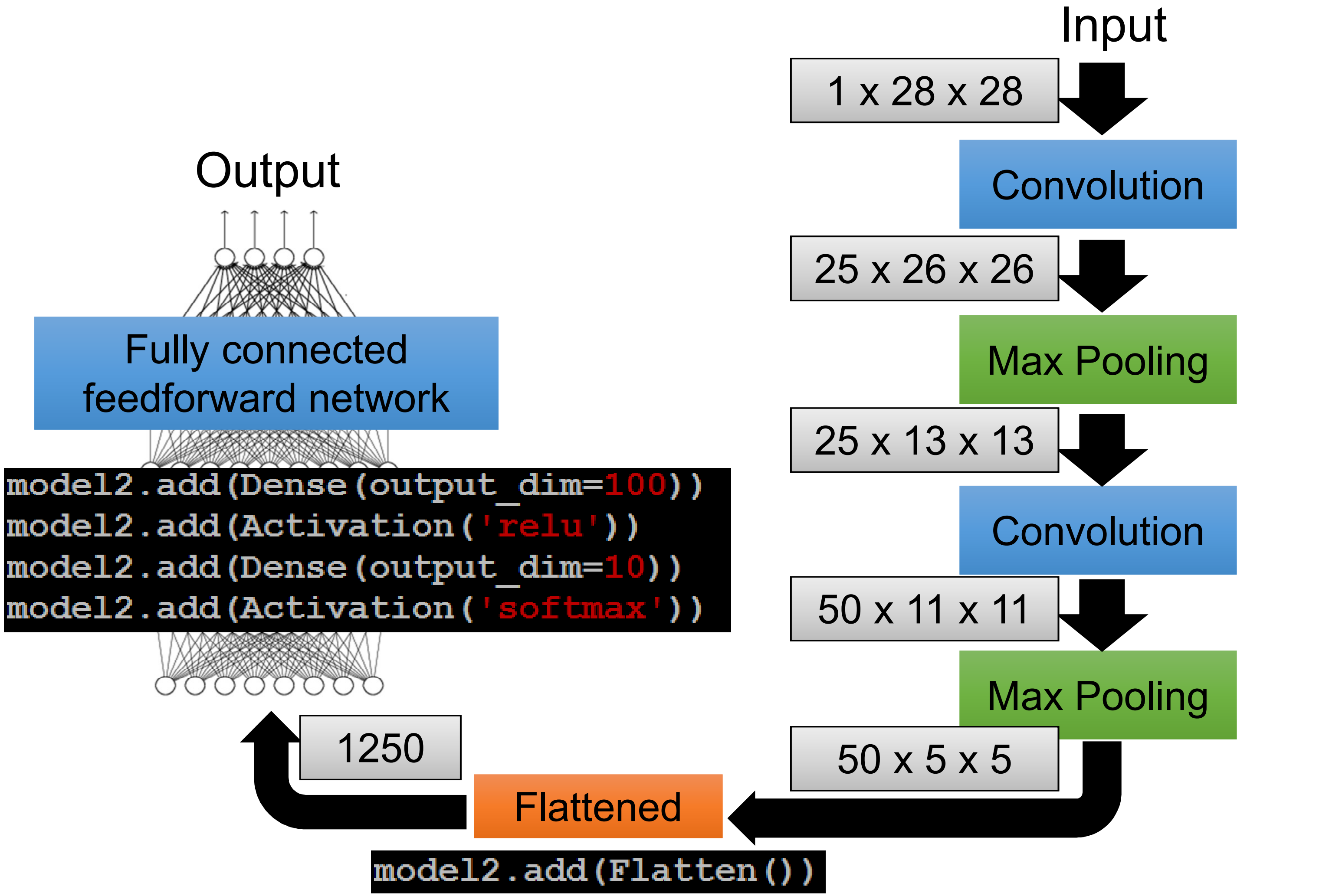


Image Source: internet