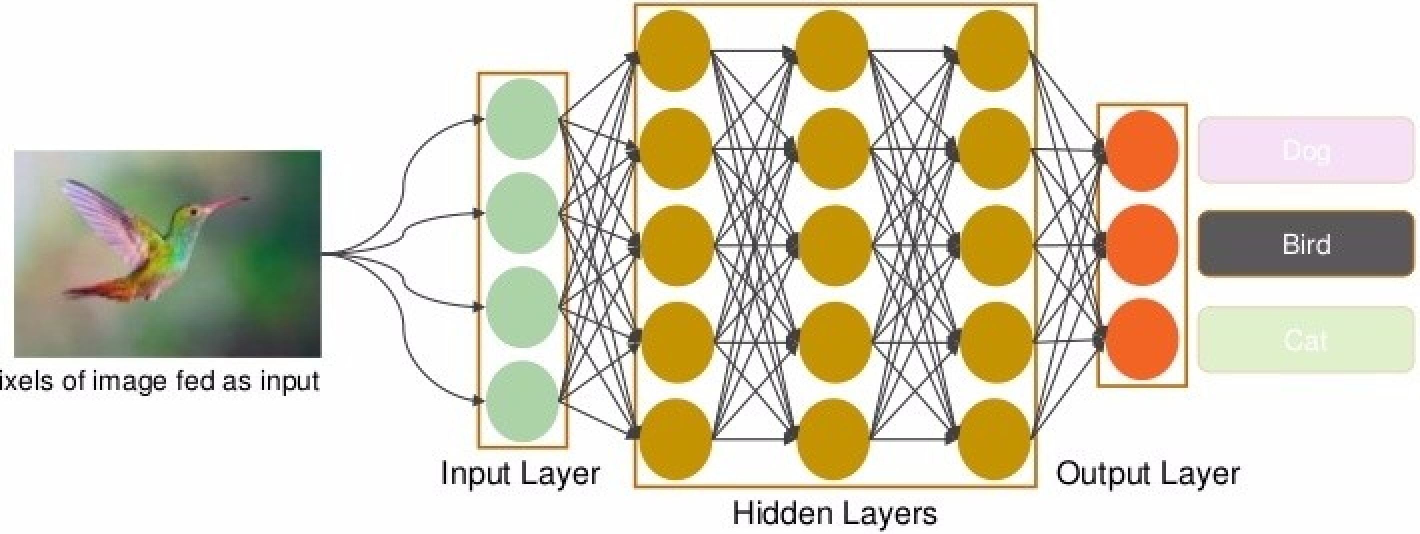
Convolution Neural Network Tutorial

How image recognition works?

Do you know how Deep Learning recognizes the objects in an image?

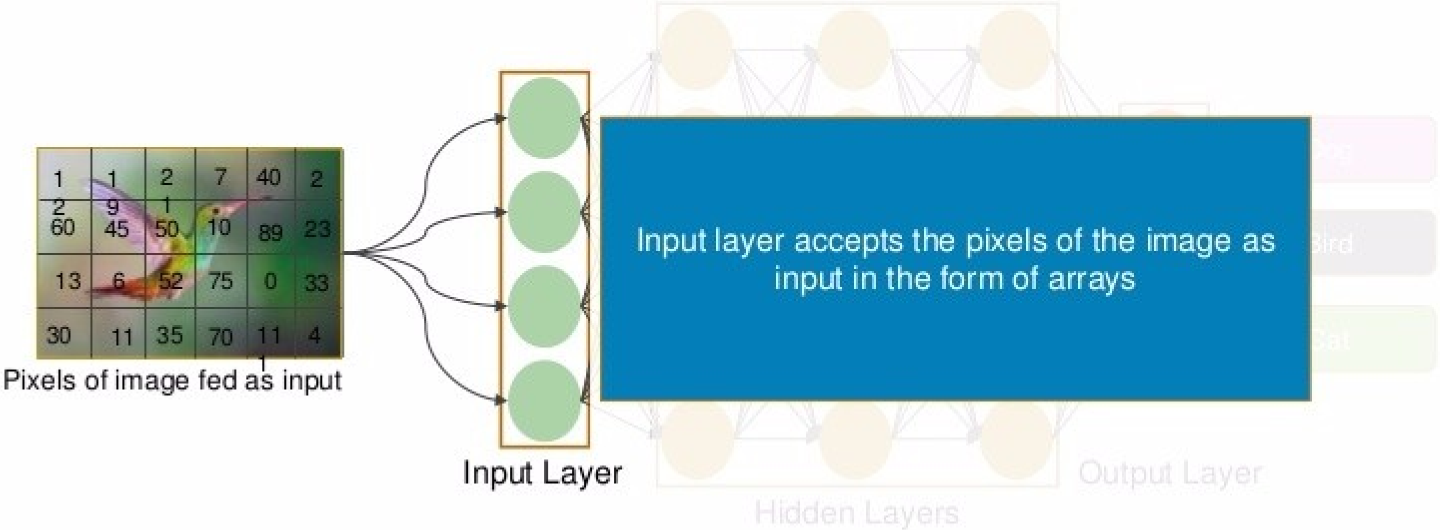
It does it using a Convolution Neural Network



Pixels of image fed as input

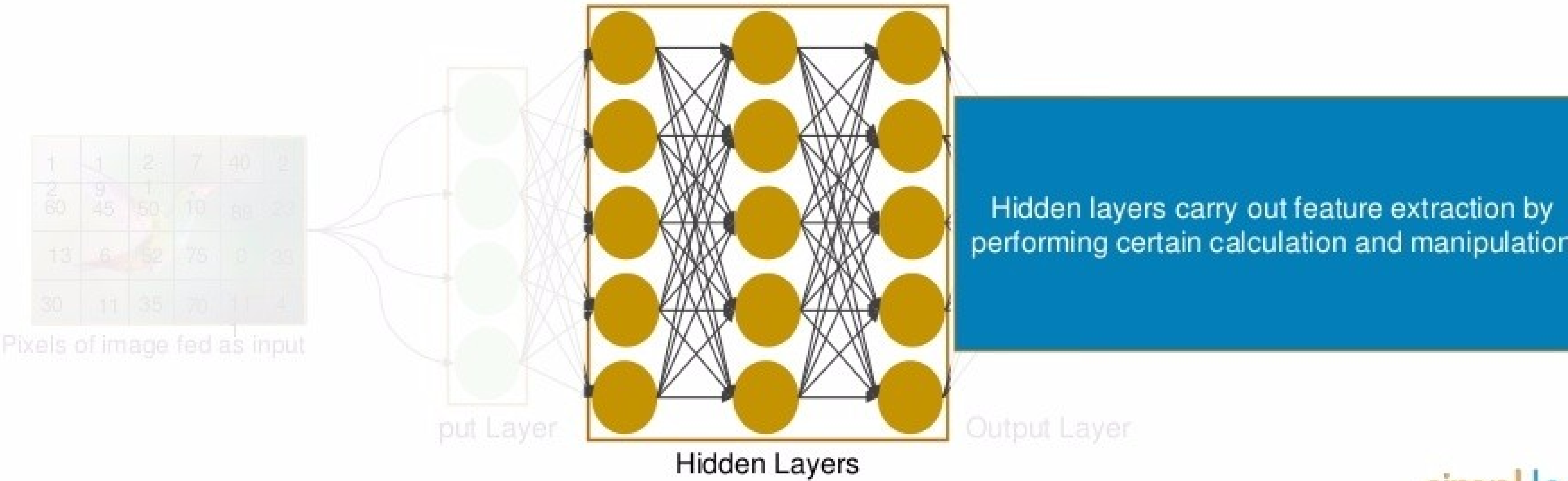
How image recognition works?

Let's see how CNN identifies the image of a bird



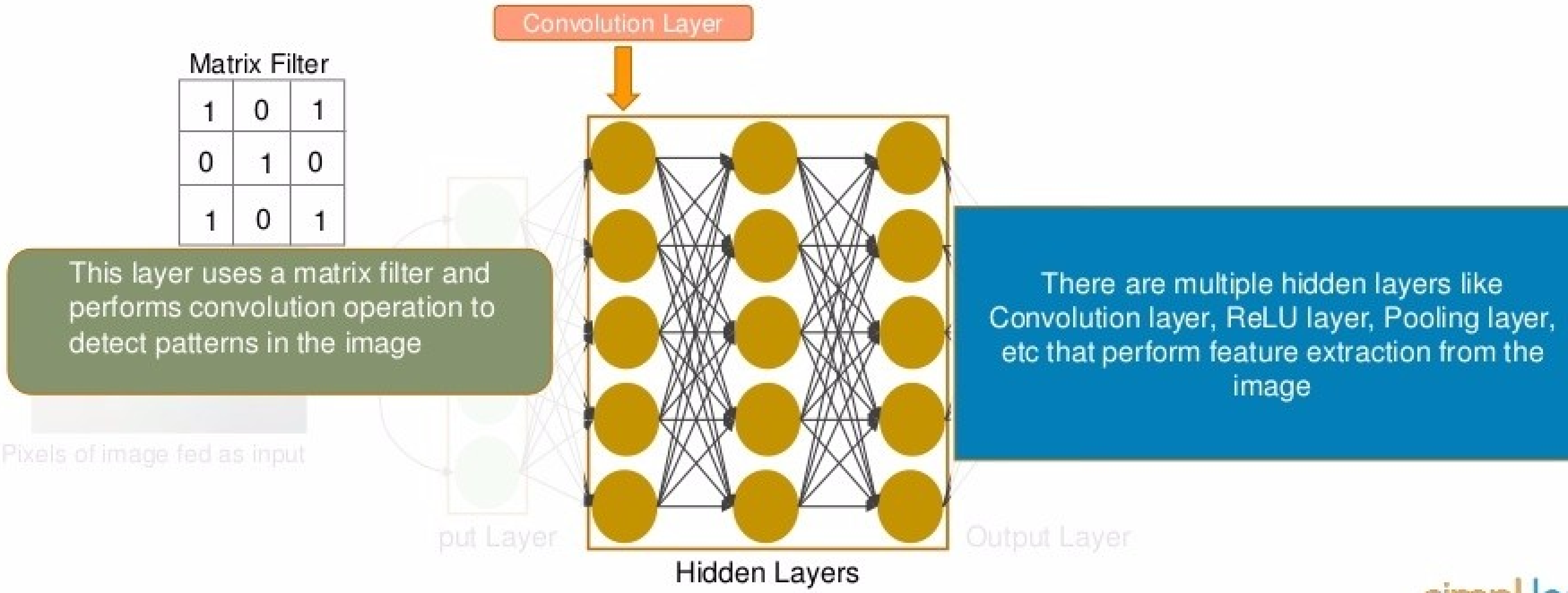
How image recognition works?

Let's see how CNN identifies the image of a bird



How image recognition works?

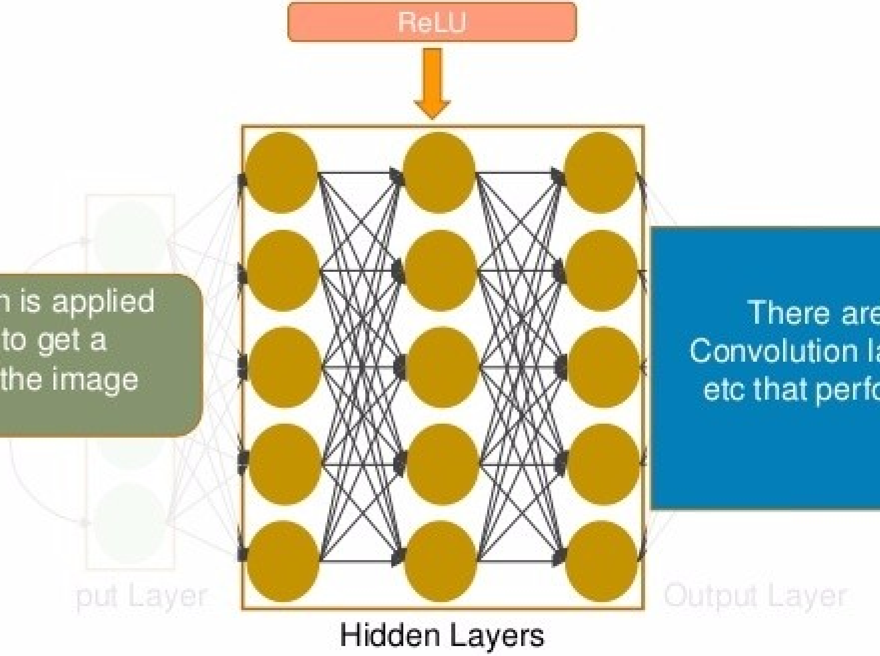
Let's see how CNN identifies the image of a bird



How image recognition works?

Let’s see how CNN identifies the image of a bird

ReLU activation function is applied to the convolution layer to get a rectified feature map of the image

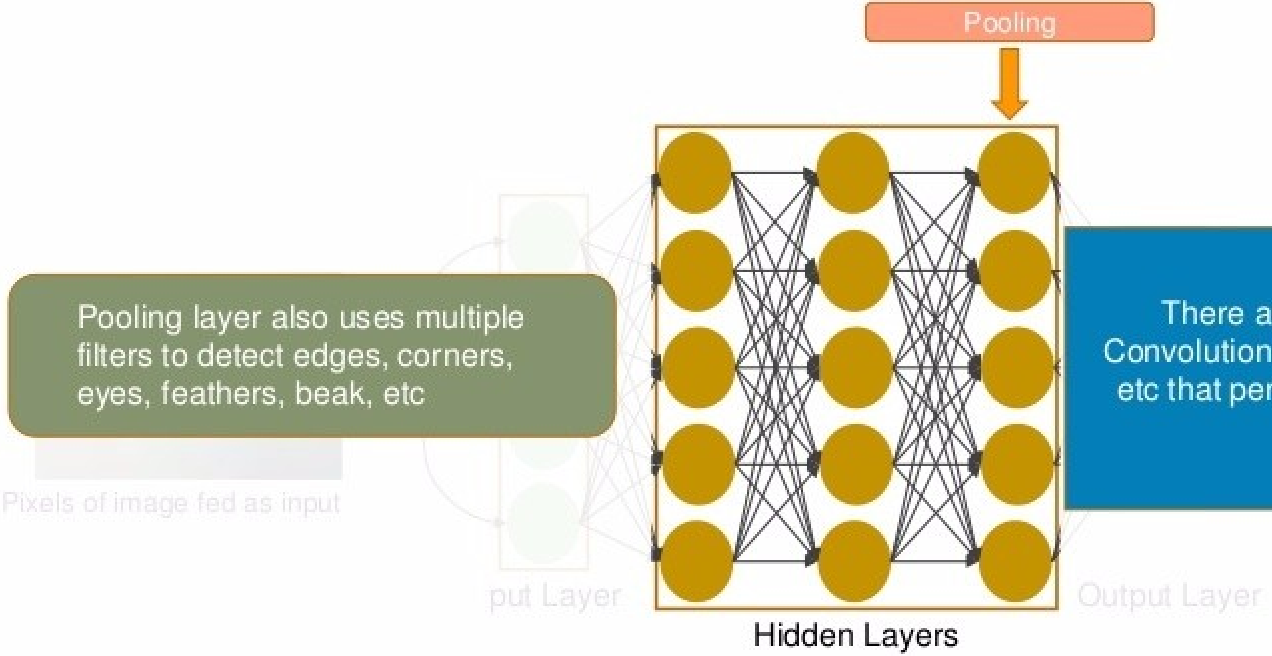


There are multiple hidden layers like Convolution layer, ReLU layer, Pooling layer, etc that perform feature extraction from the image

How image recognition works?

Let’s see how CNN identifies the image of a bird

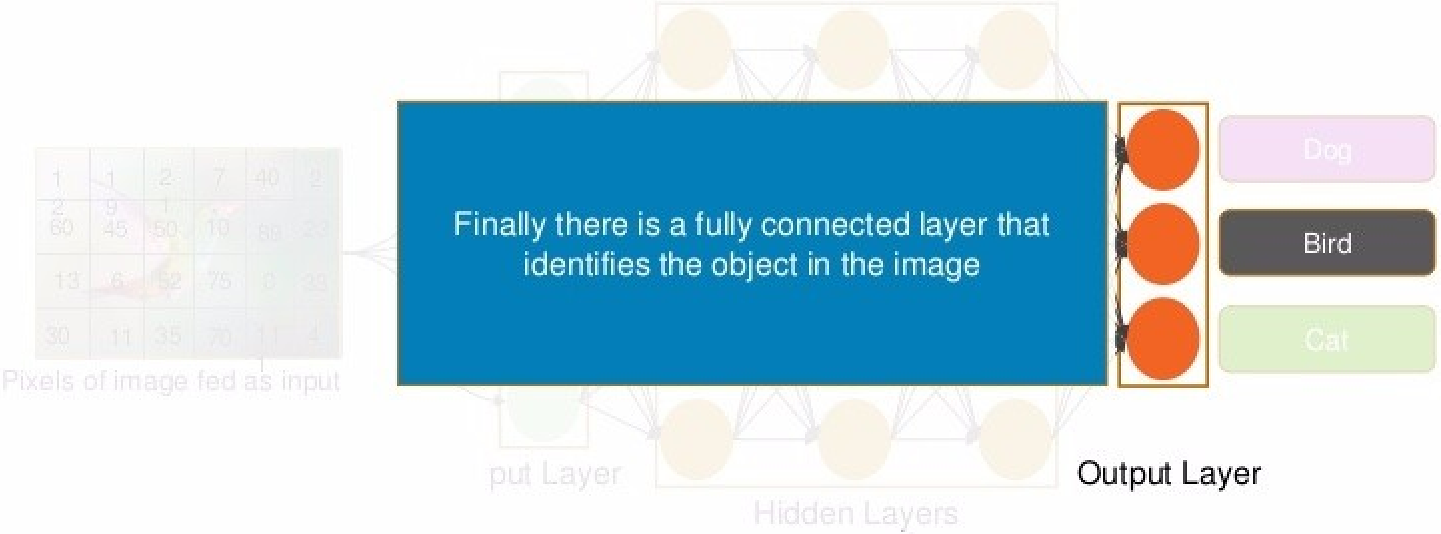
Pooling layer also uses multiple filters to detect edges, corners, eyes, feathers, beak, etc



There are multiple hidden layers like Convolution layer, ReLU layer, Pooling layer, etc that perform feature extraction from the image

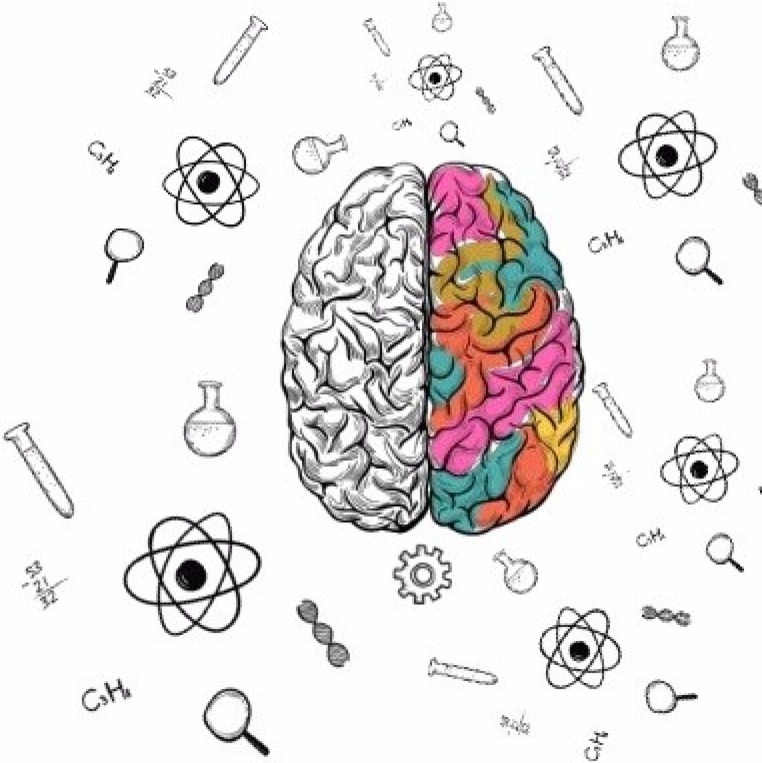
How image recognition works?

Let's see how CNN identifies the image of a bird



What's in it for you?

- Introduction to CNN
- What is Convolution neural network?
- How CNN recognizes images?
- Layers in convolution neural network
- Use case implementation using CNN



Introduction to CNN

Yann LeCun

Pioneer of Convolution Neural Network

Director of Facbook's AI Research Group

Built the first Convolution Neural Network called LeNet in 1988

It was used for character recognition tasks like reading zip codes, digits

Introduction to CNN

Yann LeCun

Pioneer of Convolution Neural Network

Director of Facbook's AI Research Group

Built the first Convolution Neural Network called LeNet in 1988

It was used for character recognition tasks like reading zip codes, digits

Introduction to CNN

Pioneer of Convolution Neural Network

Yann LeCun

Director of Facbook's AI Research Group

Built the first Convolution Neural Network called LeNet in 1988

It was used for character recognition tasks like reading zip codes, digits

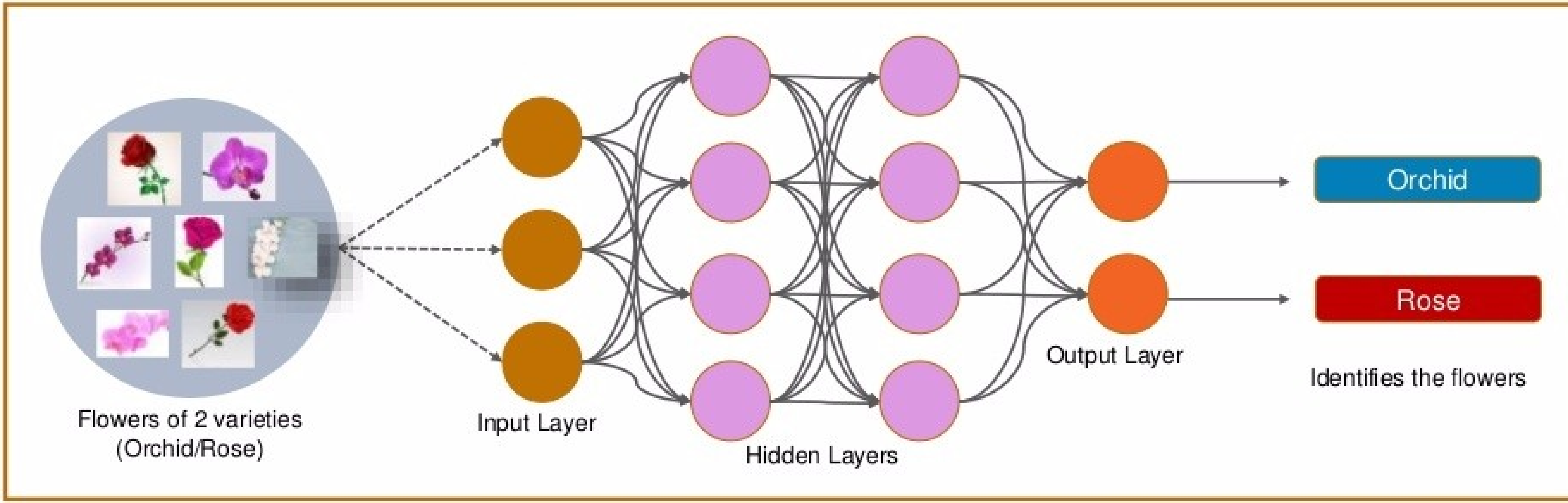
Introduction to CNN

Yann LeCun

- Pioneer of Convolution Neural Network
- Director of Facbook's AI Research Group
- Built the first Convolution Neural Network called LeNet in 1988
- It was used for character recognition tasks like reading zip codes, digits

What is a Convolution Neural Network?

CNN is a feed forward neural network that is generally used to analyze visual images by processing data with grid like topology. A CNN is also known as a "***ConvNet***"



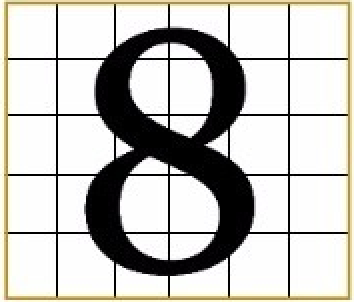
What is a Convolution Neural Network?

CNN is a feed forward neural network that is generally used to analyze visual images by processing data with grid like topology. A CNN is also known as a "***ConvNet***"

Convolution operation forms the basis of any Convolution Neural Network

In CNN, every image is represented in the form of arrays of pixel values

Real Image of the digit 8



Represented in the form of an array

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

Digit 8 represented in the fo
pixels of 0's and 1's

What is a Convolution Neural Network?

Let's understand the convolution operation using 2 matrices a and b of 1 dimension

a = [5, 3, 7, 5, 9, 7]

b = [1, 2, 3]

Matrix a and b

Convolution →

a * b

a = [5, 3, 2, 5, 9, 7]

b = [1, 2, 3]

What is a Convolution Neural Network?

Let's understand the convolution operation using 2 matrices a and b of 1 dimension

a = [5, 3, 7, 5, 9, 7]

b = [1, 2, 3]

Matrix a and b

Convolution →

a * b

Multiply the arrays element wise

[5, 6, 6]

Sum the product

17

a = [5, 3, 2, 5, 9, 7]

b = [1, 2, 3]

a * b = [17,]

What is a Convolution Neural Network?

Let's understand the convolution operation using 2 matrices a and b of 1 dimension

a = [5, 3, 7, 5, 9, 7]

b = [1, 2, 3]

Matrix a and b

Convolution →

a * b

a = [5, 3, 2, 5, 9, 7] b = [1, 2, 3]	Multiply the arrays element wise	Sum the product
	[5, 6, 6]	17
	[3, 4, 15]	22

a * b = [17, 22]

What is a Convolution Neural Network?

Let's understand the convolution operation using 2 matrices a and b of 1 dimension

a = [5, 3, 7, 5, 9, 7]

b = [1, 2, 3]

Matrix a and b

Convolution →

a * b

a = [5, 3, 2, 5, 9, 7]

b = [1, 2, 3]

Multiply the arrays element wise	Sum the product
[5, 6, 6]	17
[3, 4, 15]	22
[2, 10, 27]	39
⋮	⋮

a * b = [17, 22, 39,........]

How CNN recognizes images?

Consider the following 2 images:

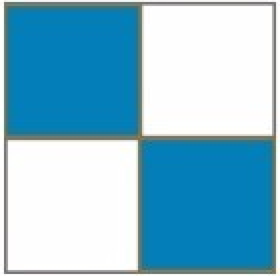
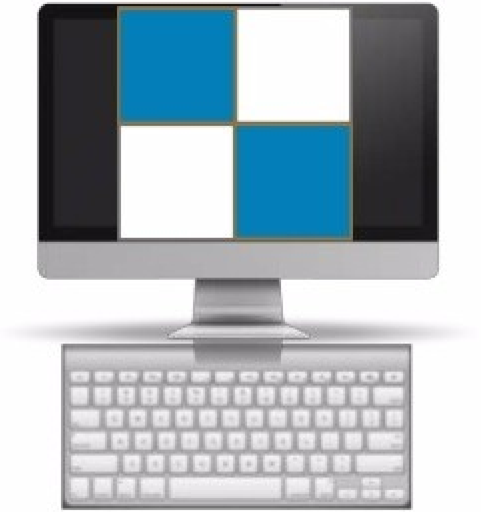


image for the symbol \



When you press \, the above image is processed

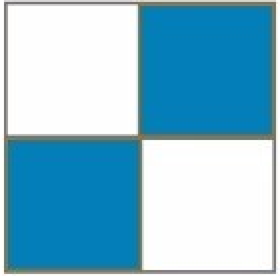


image for the symbol /

How CNN recognizes images?

Consider the following 2 images:

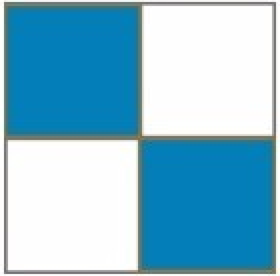
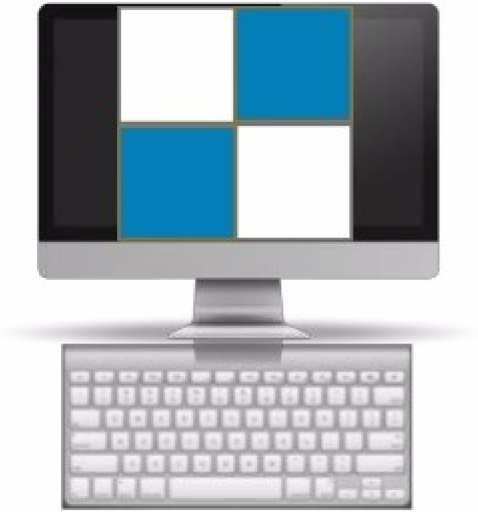


image for the symbol \



When you press /, the above image is processed

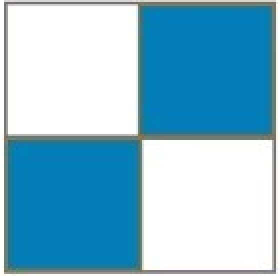
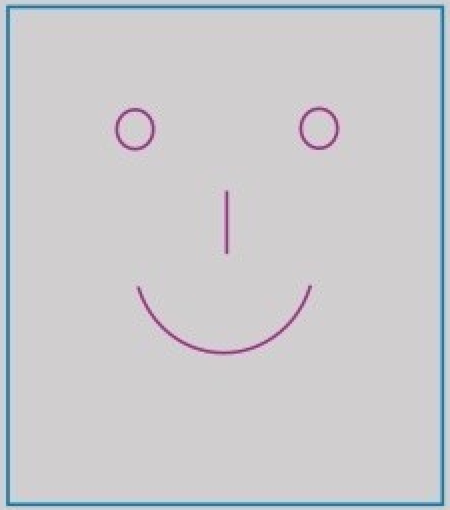
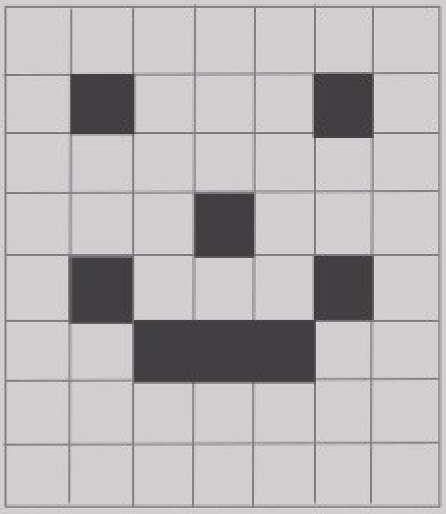


image for the symbol /

How CNN recognizes images?



Real Image

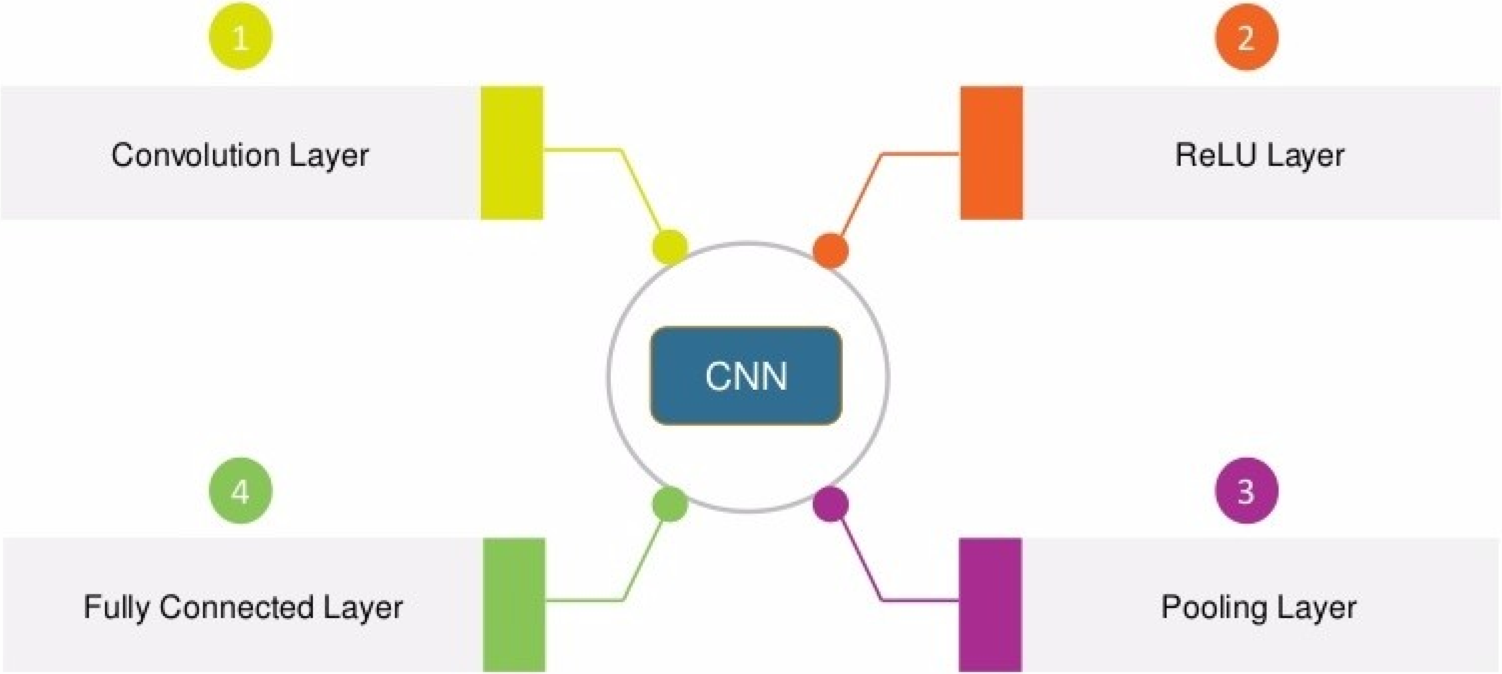


Represented in the form of black and white pixels

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

Image represented in the form of a matrix of numbers

Layers in Convolution Neural Network



Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

Image pixels

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

Filter

1	0	1
0	1	0
1	0	1

Convolved Feature

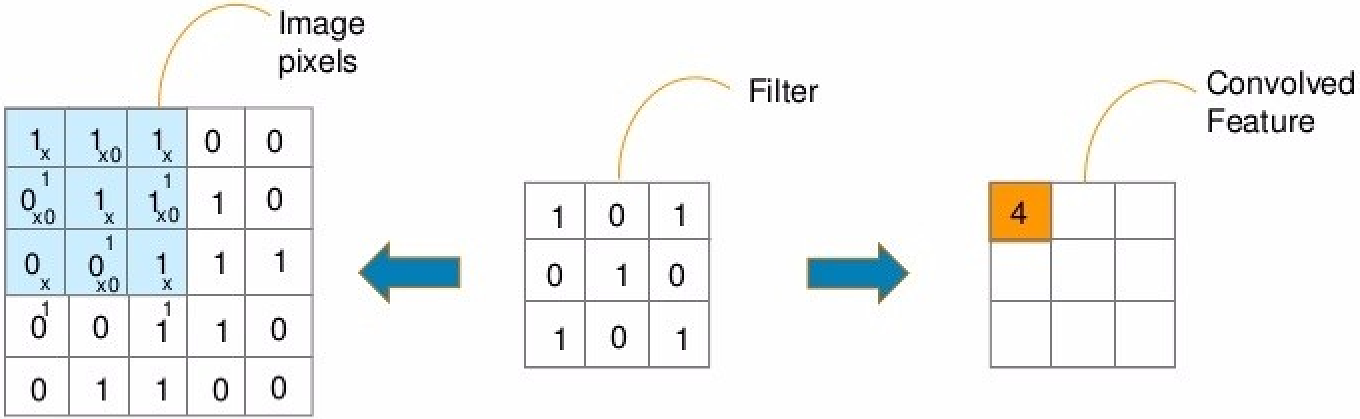
4	3	4
2	4	3
2	3	4

Sliding the filter matrix over the image and computing the dot product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

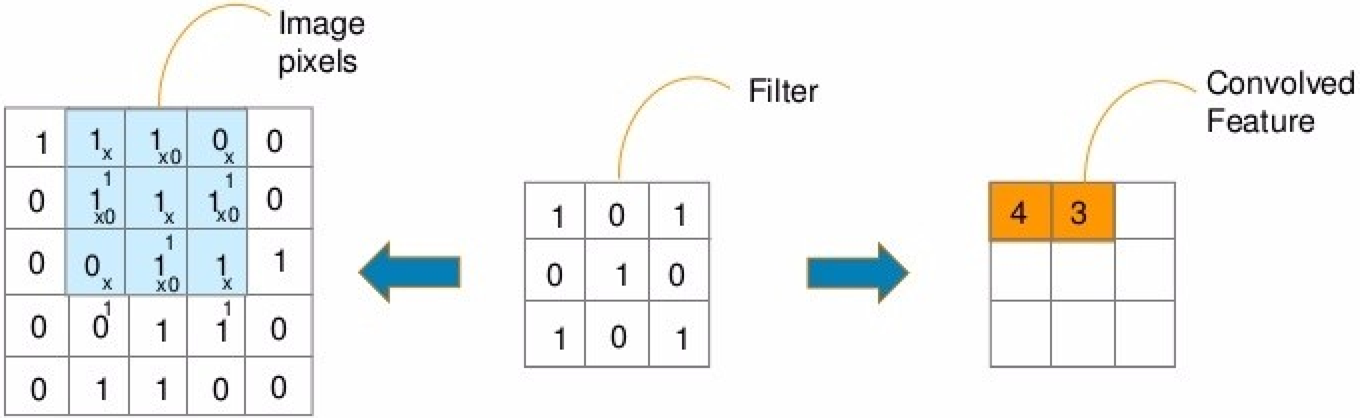


Sliding the filter matrix over the image and computing the dot product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

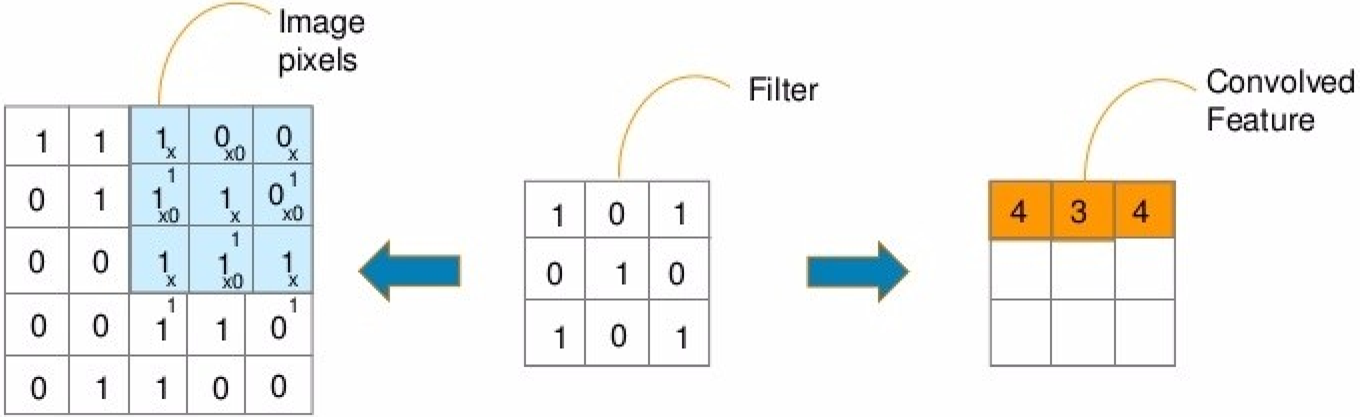


Sliding the filter matrix over the image and computing the dot product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

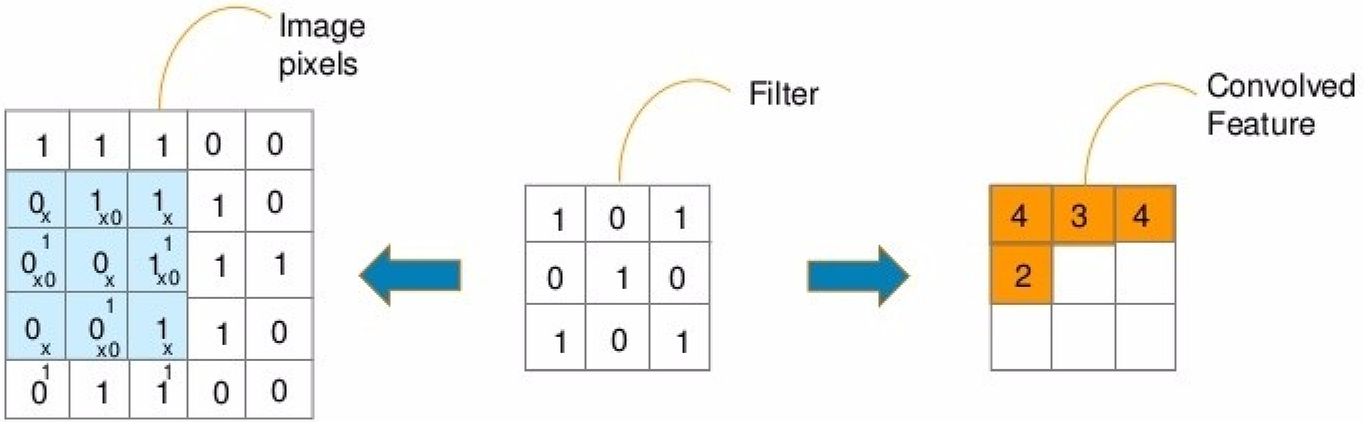


Sliding the filter matrix over the image and computing the dot product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

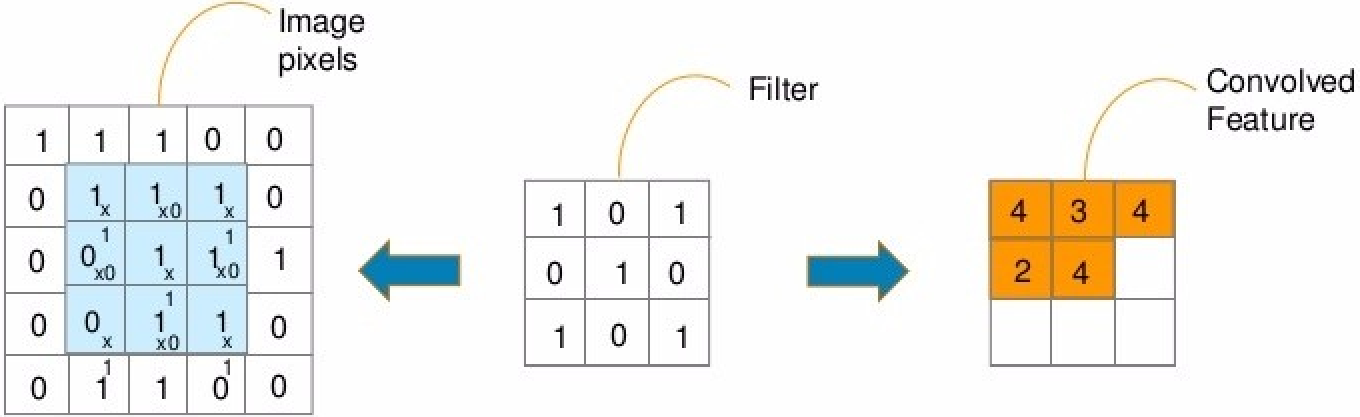


Sliding the filter matrix over the image and computing the dot product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

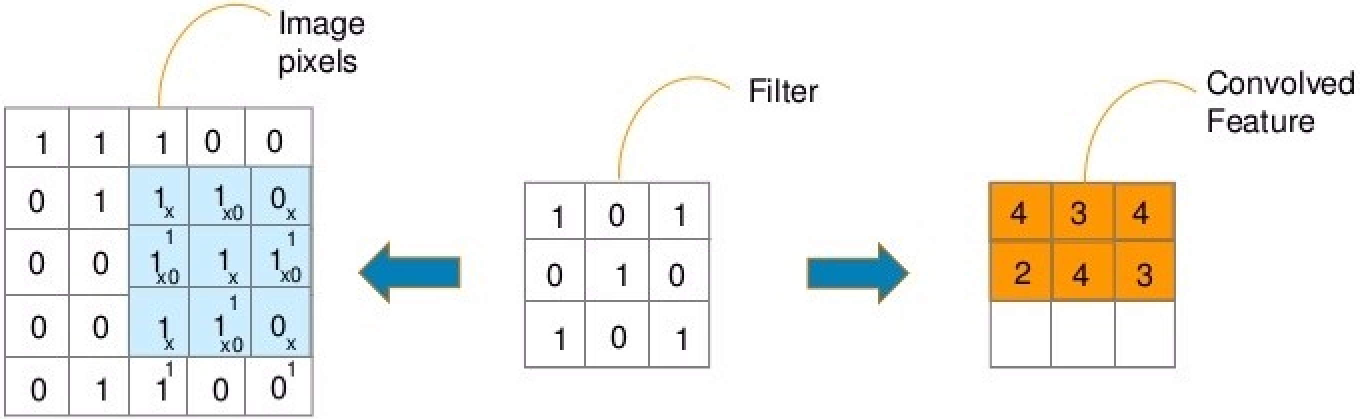


Sliding the filter matrix over the image and computing the dot product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

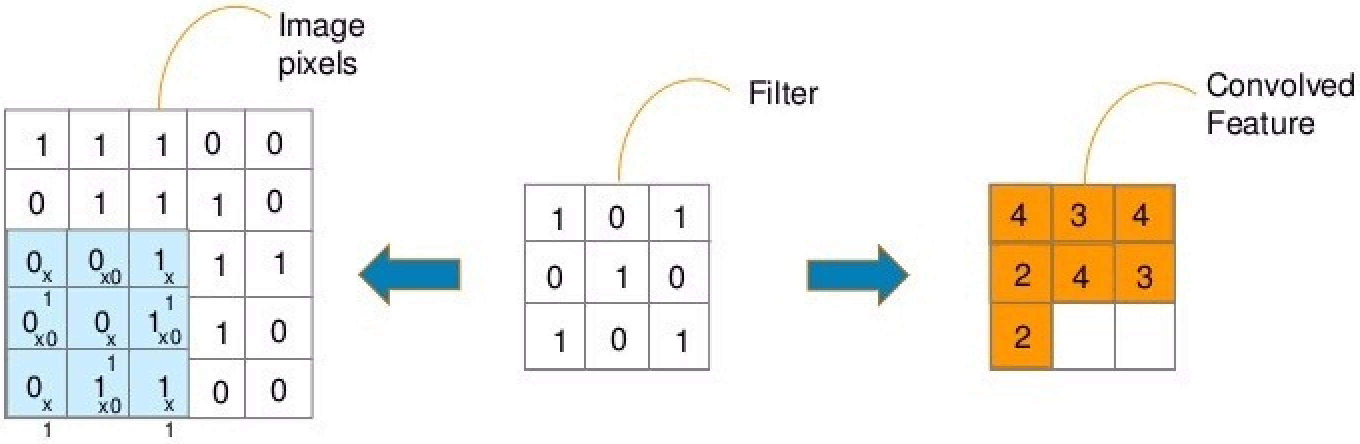


Sliding the filter matrix over the
image and computing the dot
product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

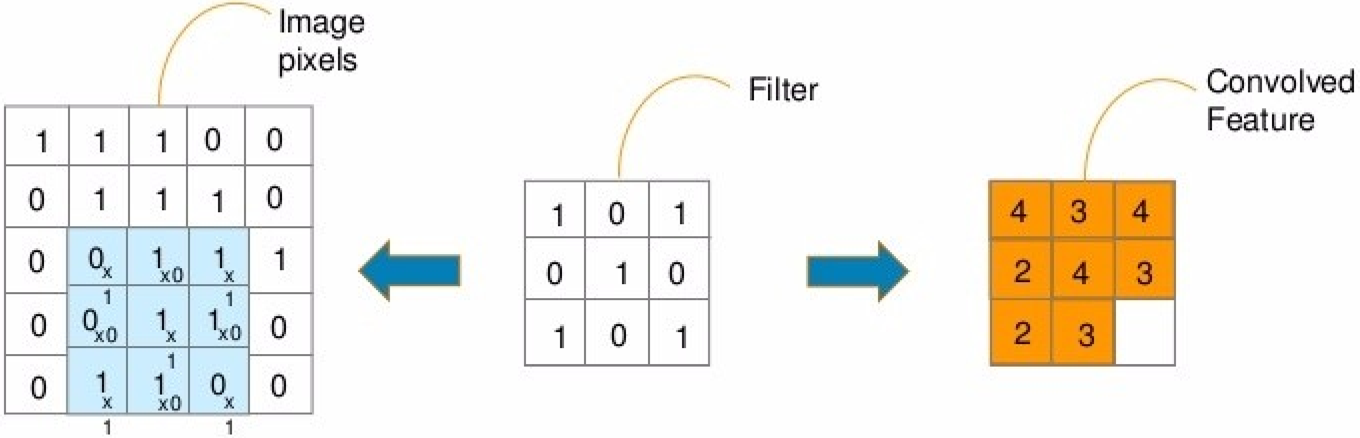


Sliding the filter matrix over the image and computing the dot product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1

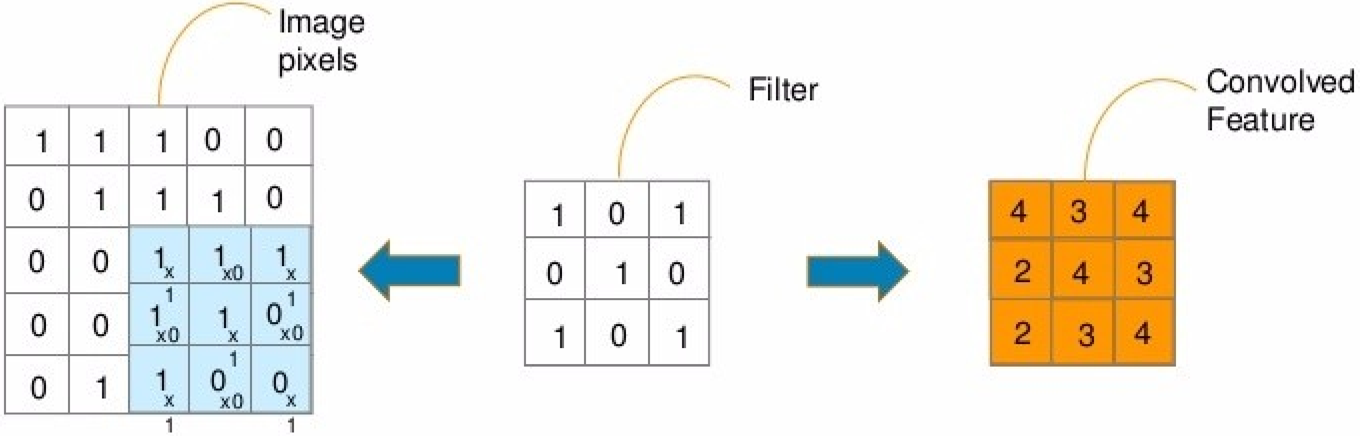


Sliding the filter matrix over the image and computing the dot product to detect patterns

Convolution Layer

A Convolution Layer has a number of filters that perform convolution operation

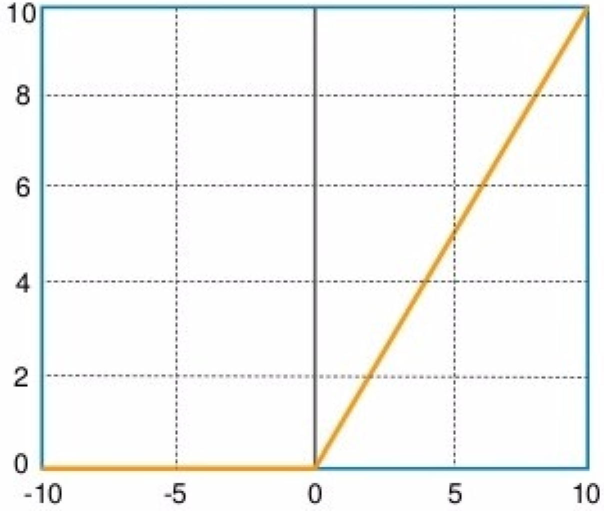
Every image is considered as a matrix of pixel values.
Consider the following 5 5 image whose pixel values are only 0 and 1



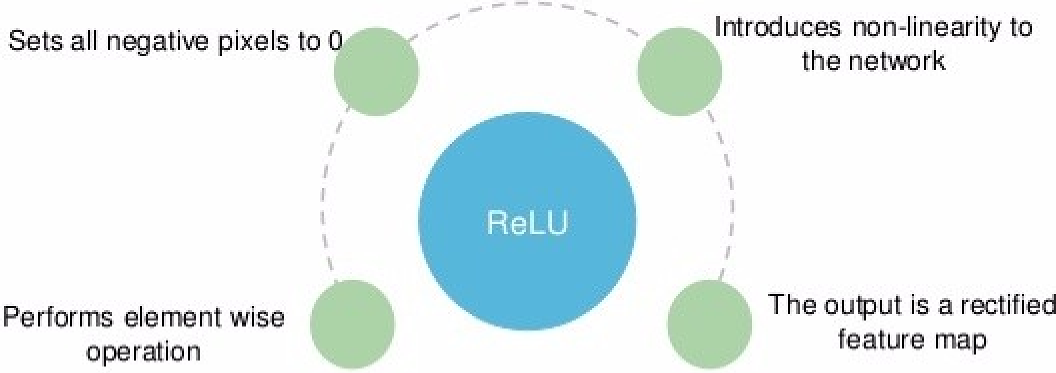
Sliding the filter matrix over the image and computing the dot product to detect patterns

ReLU Layer

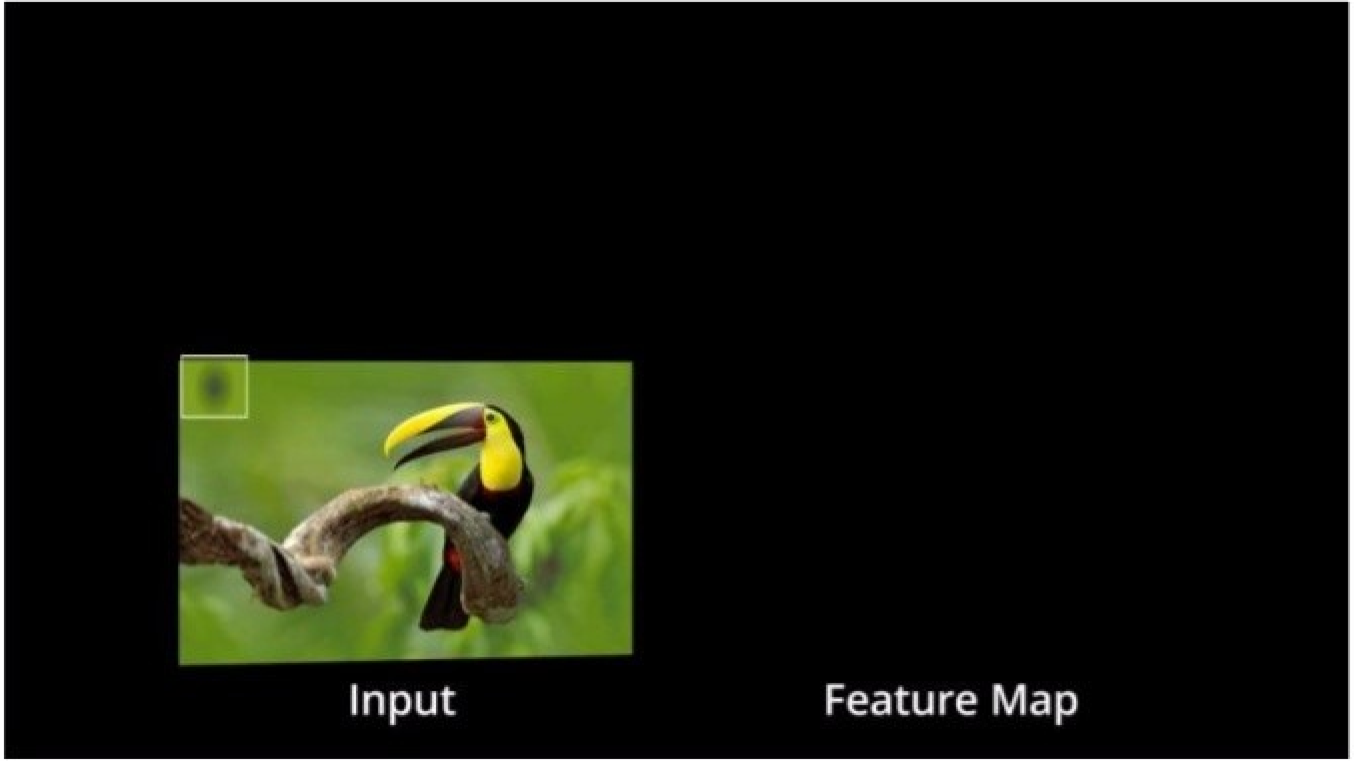
Once the feature maps are extracted, the next step is to move them to a ReLU layer



R(z) = max(0, z)

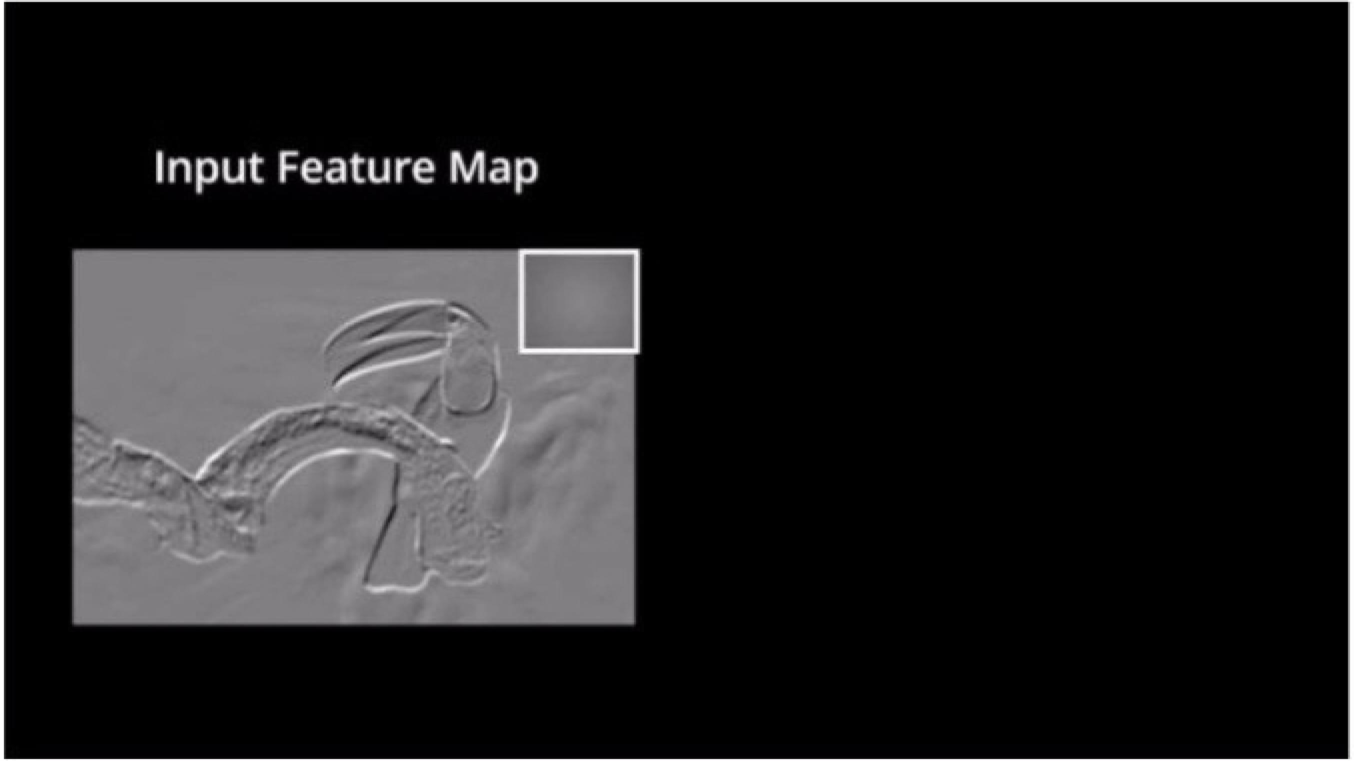


ReLU Layer



Real image is scanned in multiple convolution and ReLU layers for locating features

ReLU Layer



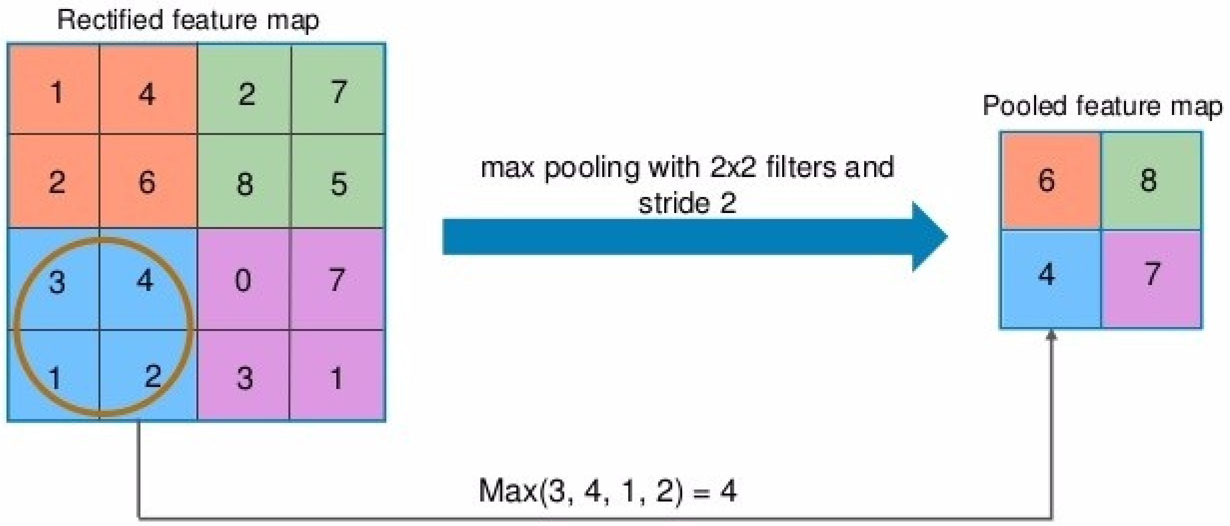
Real image is scanned in multiple convolution and ReLU layers for locating features

Note for the instructor

While explaining, please mention there are multiple Convolution, ReLU and Pooling layers connected one after another that carry out feature extraction in every layer. The input image is scanned multiple times to generate the input feature map.

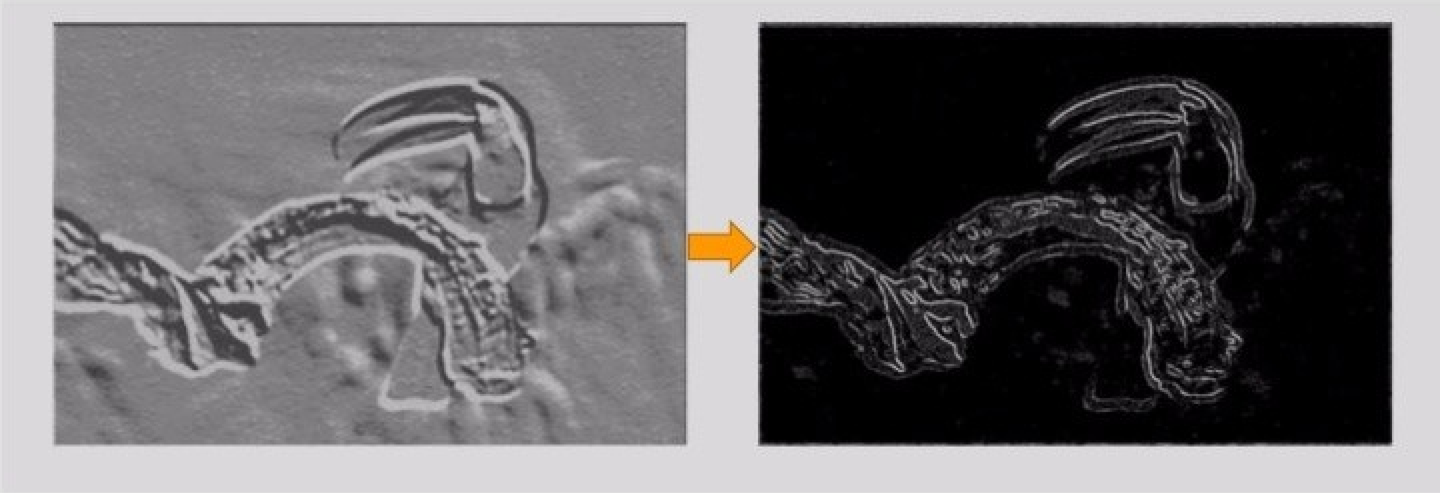
Pooling Layer

The rectified feature map now goes through a pooling layer. Pooling is a down-sampling operation that reduces the dimensionality of the feature map.



Pooling Layer

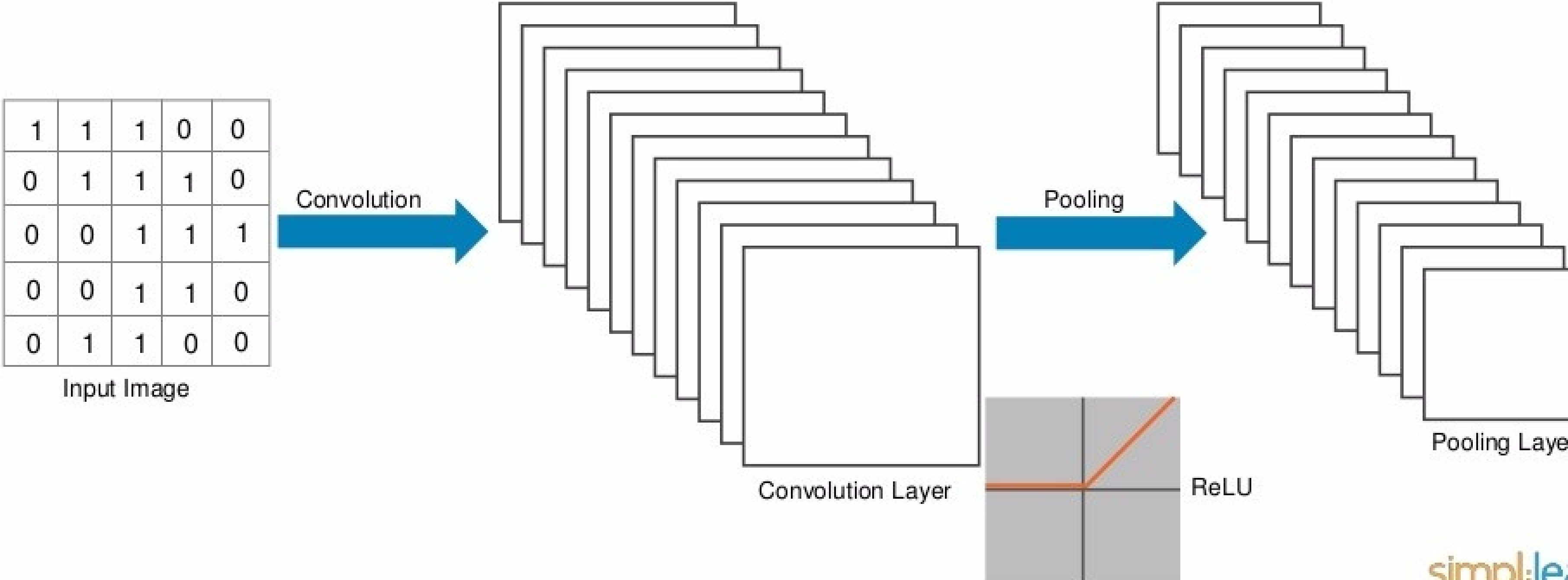
Pooling layer uses different filters to identify different parts of the image like edges, corners, body, feathers, eyes, beak, etc.



Identifies the edges, corners and other features of the bird

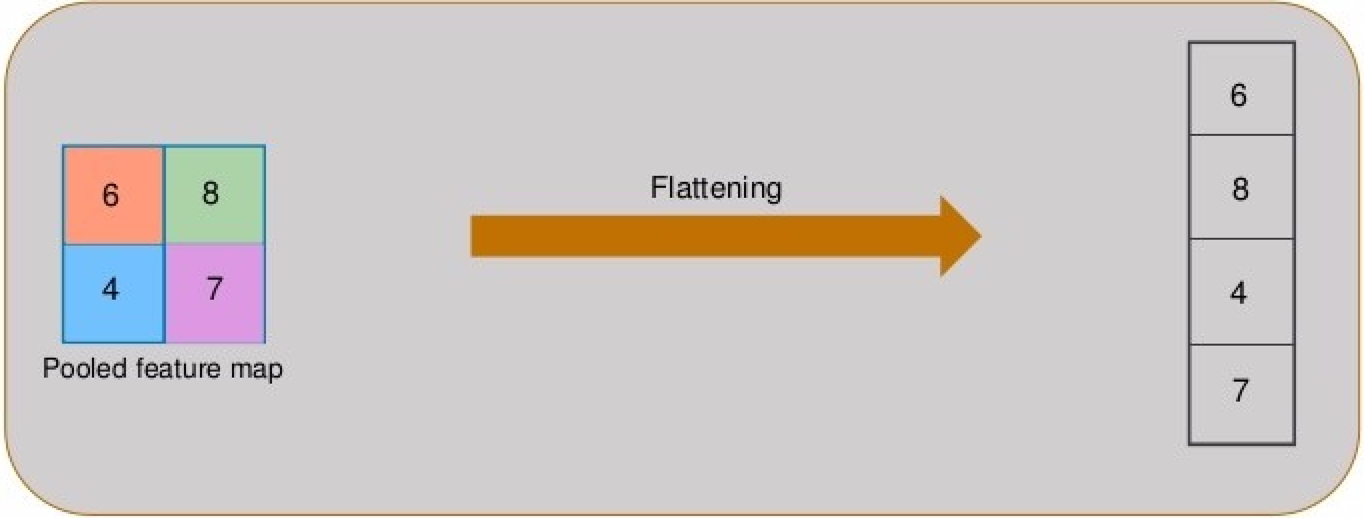
Pooling Layer

Structure of the Convolution Neural Network so far



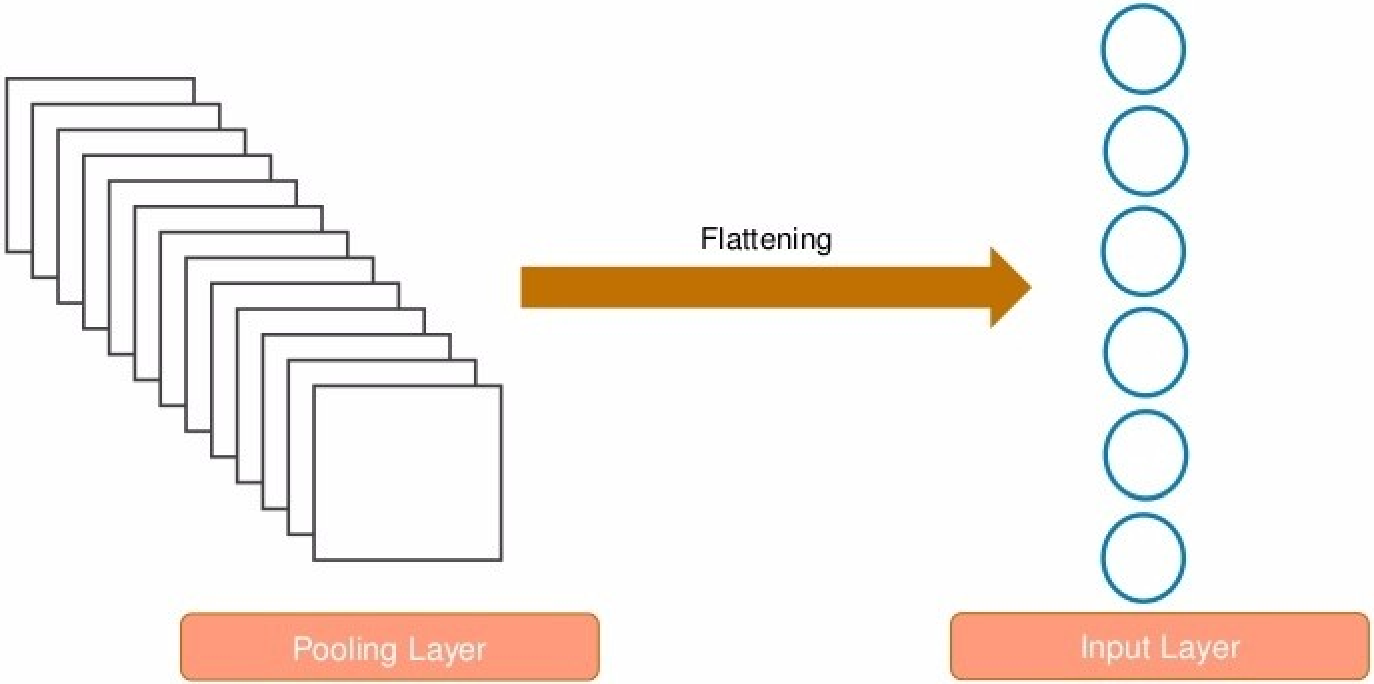
Flattening

Flattening is the process of converting all the resultant 2 dimensional arrays from pooled feature map into a single long continuous linear vector.



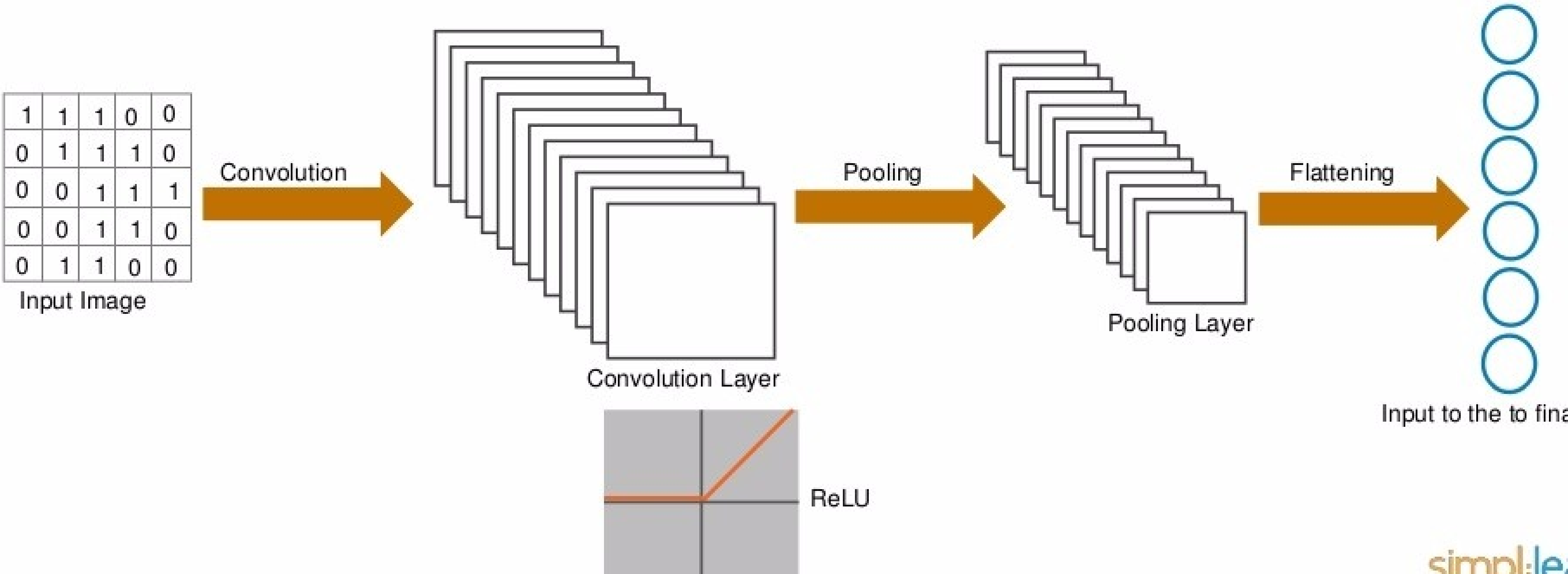
Flattening

Flattening is the process of converting all the resultant 2 dimensional arrays from pooled feature map into a single long continuous linear vector.



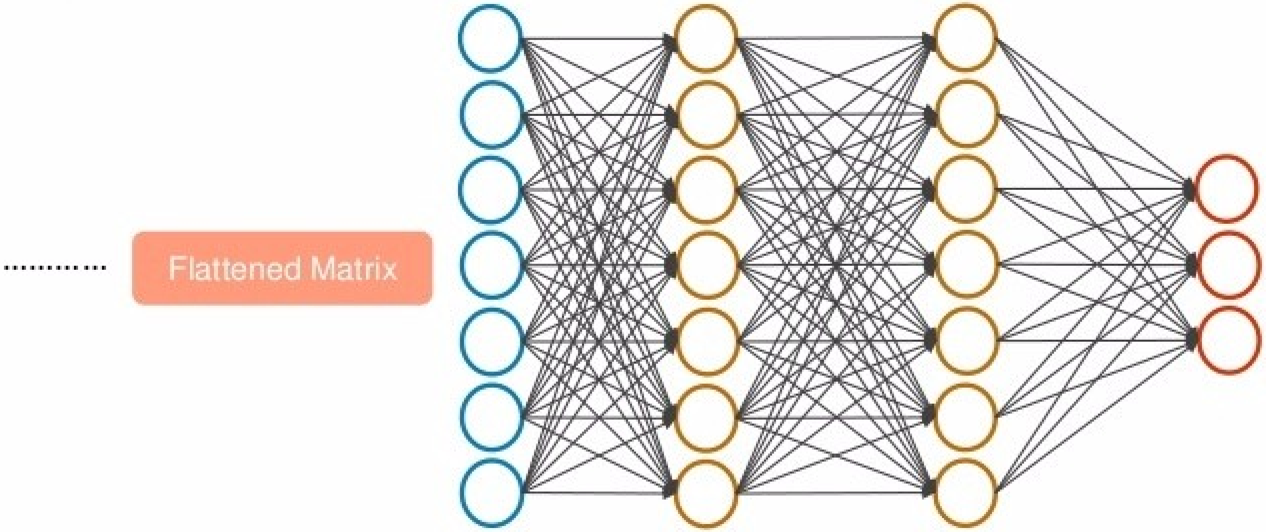
Flattening

Structure of the network so far



Fully Connected Layer

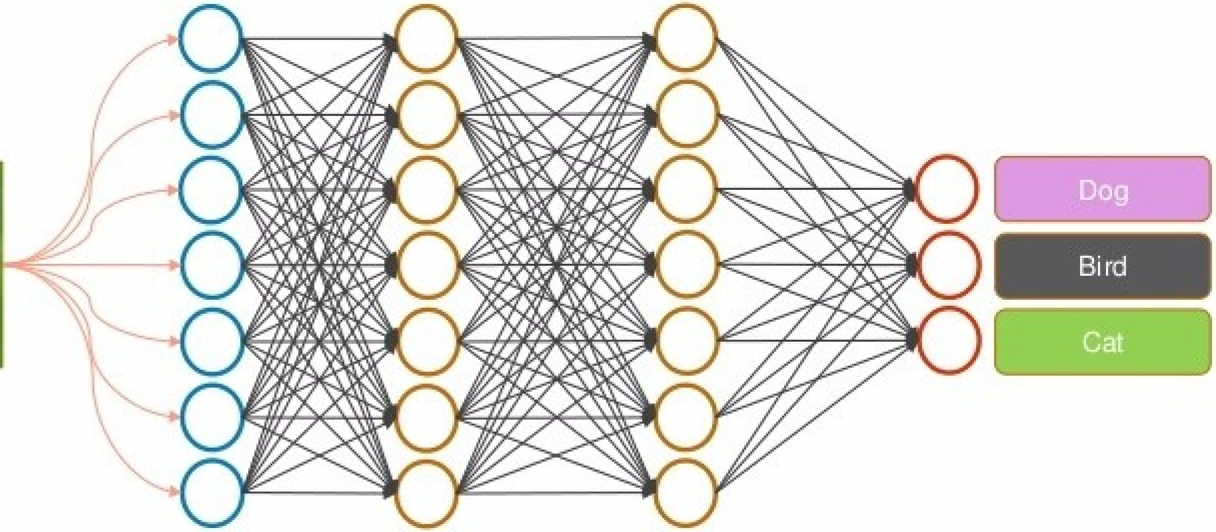
The Flattened matrix from the pooling layer is fed as input to the Fully Connected Layer to classify the image



Fully Connected Layer

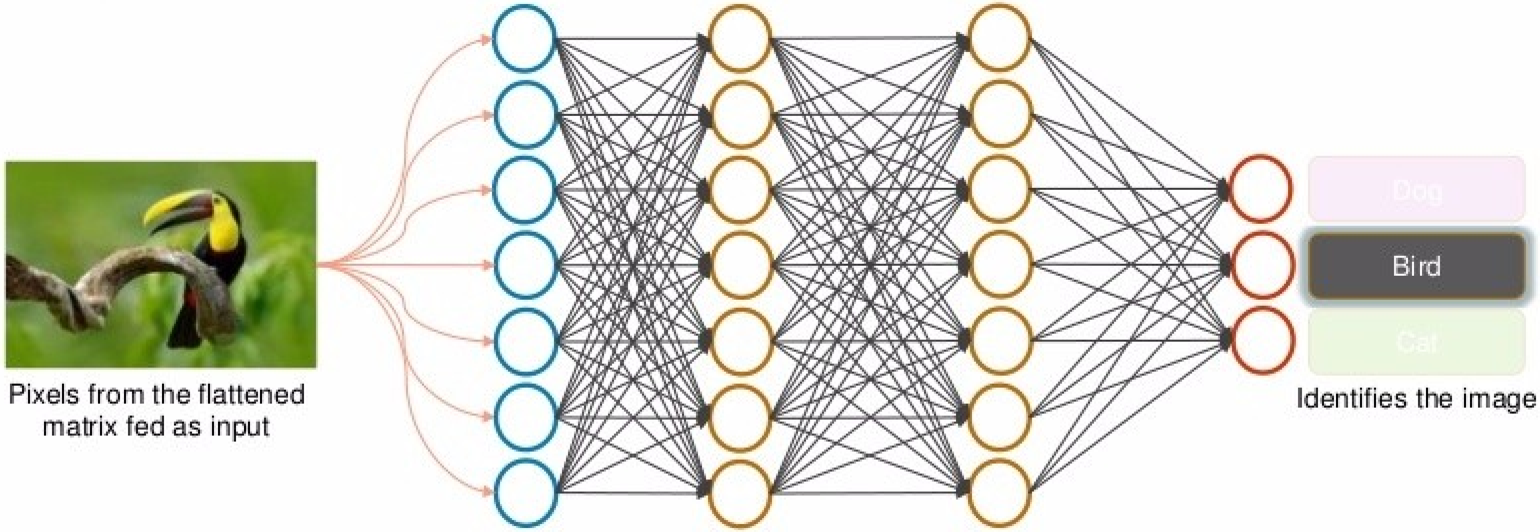
The Flattened matrix from the pooling layer is fed as input to the Fully Connected Layer to classify the image

Pixels from the flattened matrix fed as input



Fully Connected Layer

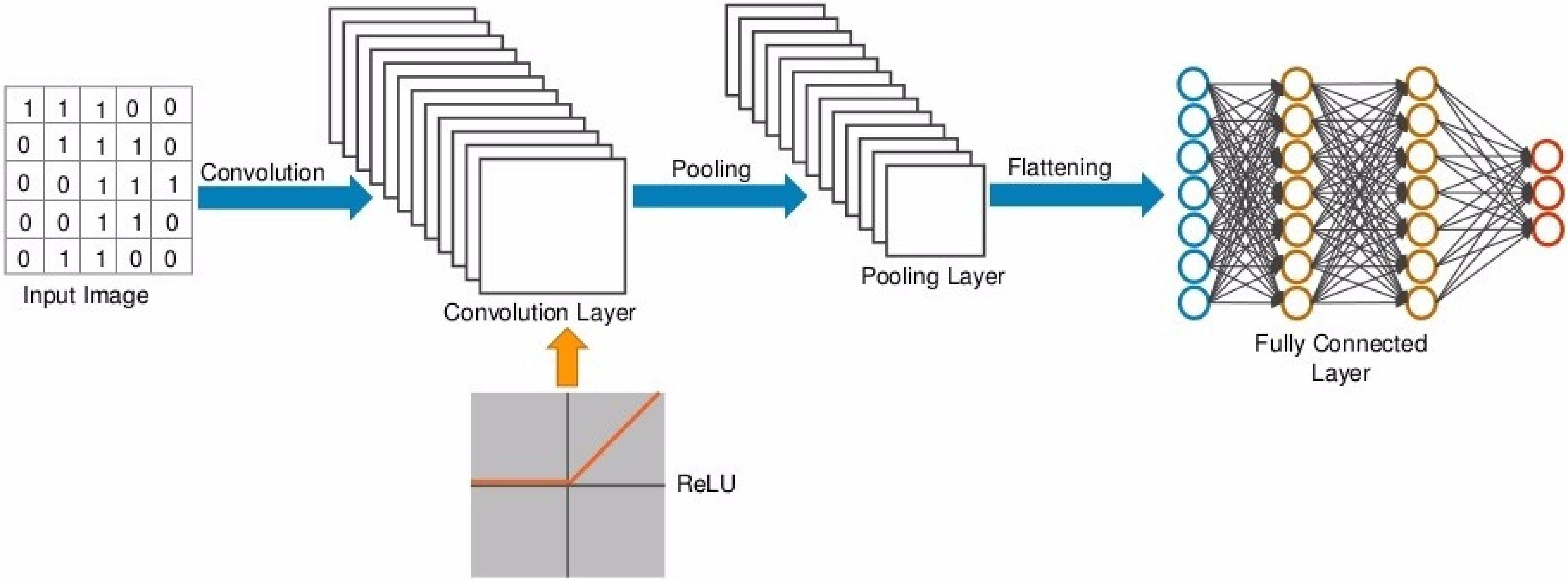
The Flattened matrix from the pooling layer is fed as input to the Fully Connected Layer to classify the image



Pixels from the flattened matrix fed as input

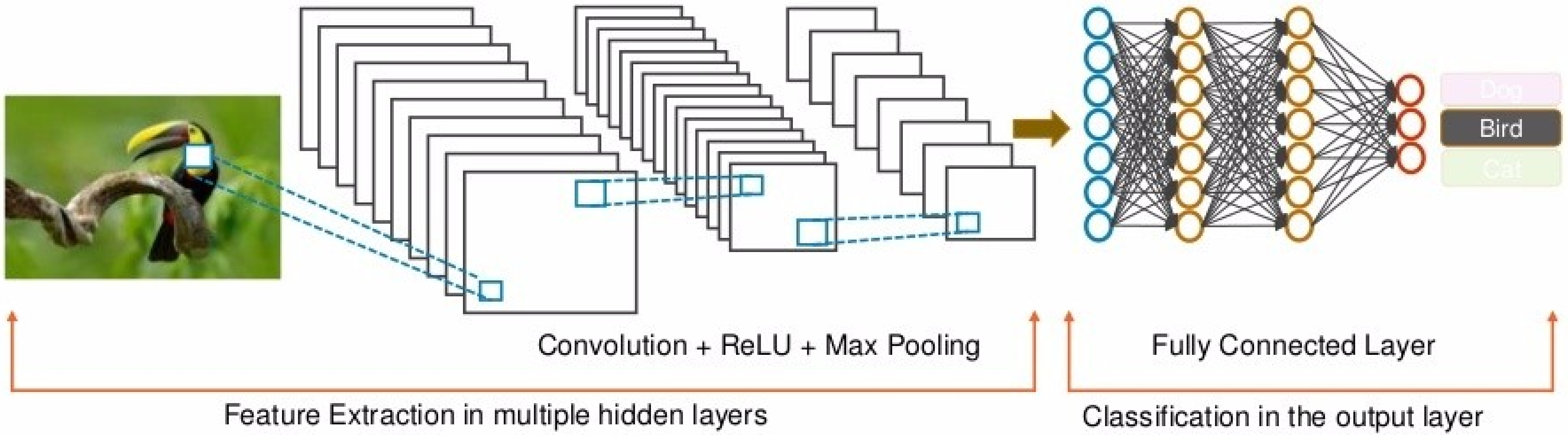
Identifies the image

Fully Connected Layer



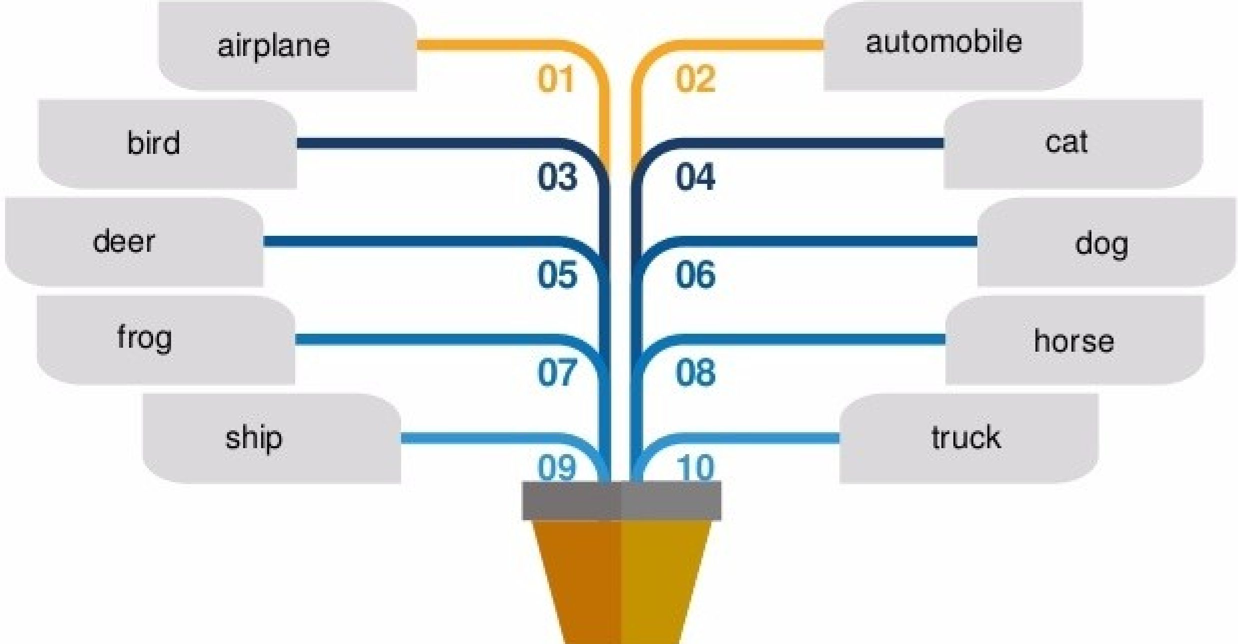
Fully Connected Layer

Lets see the entire process how CNN recognizes a bird



Use case implementation using CNN

We will be using CIFAR-10 data set (from Canadian Institute For Advanced Research) for classifying images across 10 categories



Use case implementation using CNN

1. Download data set

Download the data for CIFAR from here: https://www.cs.toronto.edu/~kriz/cifar.html

Specifically the CIFAR-10 python version link: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

Remember the directory you save the file in!

```
# Put file path as a string here
CIFAR_DIR = 'cifar-10-batches-py/'
```

2. Import the CIFAR data set

```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        cifar_dict = pickle.load(fo, encoding='bytes')
    return cifar_dict
```

```
dirs = ['batches.meta','data_batch_1','data_batch_2','data_batch_3','data_batch_4','data_batch_5','test_batch']
```

```
all_data = [0,1,2,3,4,5,6]
```

```
print(CIFAR_DIR+direc)
```

```
cifar-10-batches-py/batches.meta
```

Use case implementation using CNN

```
for i,direc in zip(all_data,dirs):
    all_data[i] = unpickle(CIFAR_DIR+direc)
```

```
batch_meta = all_data[0]
data_batch1 = all_data[1]
data_batch2 = all_data[2]
data_batch3 = all_data[3]
data_batch4 = all_data[4]
data_batch5 = all_data[5]
test_batch = all_data[6]
```

3. Reading the label names

```
batch_meta
```

```
{b'label_names': [b'airplane',
  b'automobile',
  b'bird',
  b'cat',
  b'deer',
  b'dog',
  b'frog',
  b'horse',
  b'ship',
  b'truck'],
 b'num_cases_per_batch': 10000,
 b'num_vis': 3072}
```

Use case implementation using CNN

4. Display images using matplotlib

```
import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np
```

```
X = data_batch1[b"data"]
```

```
X = X.reshape(10000, 3, 32, 32).transpose(0,2,3,1).astype("uint8")
```

```
X[0].max()
```

```
(X[0]/255).max()
```

```
plt.imshow(X[0])
```

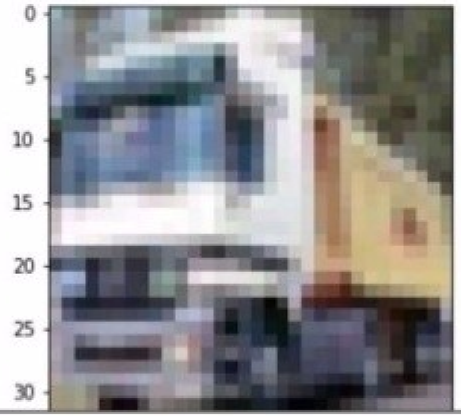
```
<matplotlib.image.AxesImage at 0x7fa87d412b70>
```

Use case implementation using CNN

4. Display images using matplotlib

```
plt.imshow(X[1])
```

```
<matplotlib.image.AxesImage at 0x7fa87d3fe588>
```

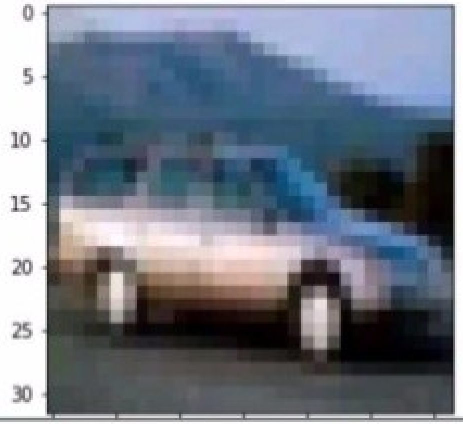


Use case implementation using CNN

4. Display images using matplotlib

```
plt.imshow(X[4])
```

```
<matplotlib.image.AxesImage at 0x7f56d0a24080>
```



Use case implementation using CNN

5. Helper function to handle data

```
def one_hot_encode(vec, vals=10):
    '''
    For use to one-hot encode the 10- possible labels
    '''
    n = len(vec)
    out = np.zeros((n, vals))
    out[range(n), vec] = 1
    return out
```

Use case implementation using CNN

5. Helper function to handle data

```
class CifarHelper():

    def __init__(self):
        self.i = 0

        self.all_train_batches = [data_batch1,data_batch2,data_batch3,data_batch4,data_batch5]
        self.test_batch = [test_batch]

        self.training_images = None
        self.training_labels = None

        self.test_images = None
        self.test_labels = None

    def set_up_images(self):

        print("Setting Up Training Images and Labels")

        self.training_images = np.vstack([d[b"data"] for d in self.all_train_batches])
        train_len = len(self.training_images)

        self.training_images = self.training_images.reshape(train_len,3,32,32).transpose(0,2,3,1)/255
        self.training_labels = one_hot_encode(np.hstack([d[b"labels"] for d in self.all_train_batches]), 10)

        print("Setting Up Test Images and Labels")

        self.test_images = np.vstack([d[b"data"] for d in self.test_batch])
        test_len = len(self.test_images)

        self.test_images = self.test_images.reshape(test_len,3,32,32).transpose(0,2,3,1)/255
```

Use case implementation using CNN

6. To use the previous code, run the following

```
# Before Your tf.Session run these two lines
ch = CifarHelper()
ch.set_up_images()
```

7. Creating the model

```
import tensorflow as tf
```

```
x = tf.placeholder(tf.float32,shape=[None,32,32,3])
y_true = tf.placeholder(tf.float32,shape=[None,10])
```

```
hold_prob = tf.placeholder(tf.float32)
```

Use case implementation using CNN

8. Applying the helper functions

```
def init_weights(shape):
    init_random_dist = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(init_random_dist)

def init_bias(shape):
    init_bias_vals = tf.constant(0.1, shape=shape)
    return tf.Variable(init_bias_vals)

def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2by2(x):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                          strides=[1, 2, 2, 1], padding='SAME')

def convolutional_layer(input_x, shape):
    W = init_weights(shape)
    b = init_bias([shape[3]])
    return tf.nn.relu(conv2d(input_x, W) + b)

def normal_full_layer(input_layer, size):
    input_size = int(input_layer.get_shape()[1])
    W = init_weights([input_size, size])
    b = init_bias([size])
    return tf.matmul(input_layer, W) + b
```

Use case implementation using CNN

8. Create the layers

```
convo_1 = convolutional_layer(x,shape=[4,4,3,32])
convo_1_pooling = max_pool_2by2(convo_1)
```

```
convo_2 = convolutional_layer(convo_1_pooling,shape=[4,4,32,64])
convo_2_pooling = max_pool_2by2(convo_2)
```

9. Create the flattened layer by reshaping the pooling layer

```
8*8*64
```

```
4096
```

```
convo_2_flat = tf.reshape(convo_2_pooling,[-1,8*8*64])
```

10. Create the fully connected layer

```
full_layer_one = tf.nn.relu(normal_full_layer(convo_2_flat,1024))
```

```
full_one_dropout = tf.nn.dropout(full_layer_one,keep_prob=hold_prob)
```

Use case implementation using CNN

11. Set output to y_pred

```
y_pred = normal_full_layer(full_one_dropout,10)
y_pred
```

```
<tf.Tensor 'add_9:0' shape=(?, 10) dtype=float32>
```

12. Apply the Loss function

```
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_true,logits=y_pred))
```

13. Create the optimizer

```
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)
train = optimizer.minimize(cross_entropy)
```

14. Create a variable to initialize all the global tf variables

```
init = tf.global_variables_initializer()
```

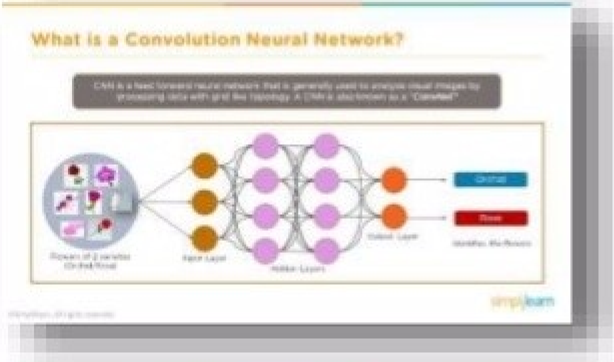
Use case implementation using CNN

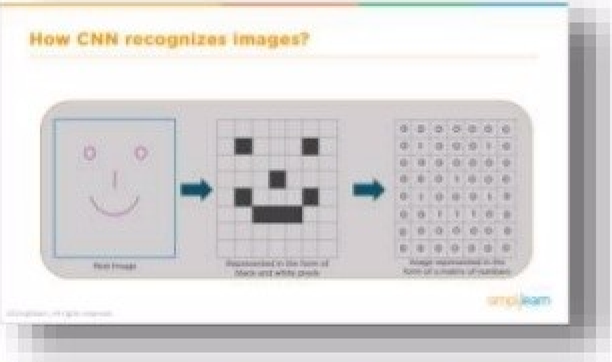
15. Run the model by creating a Graph Session

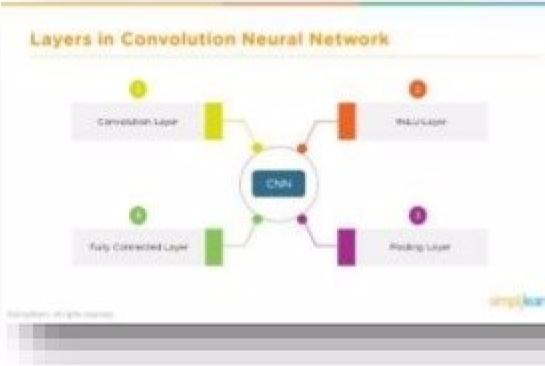
```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for i in range(500):
        batch = ch.next_batch(100)
        sess.run(train, feed_dict={x: batch[0], y_true: batch[1], hold_prob: 0.5})

        # PRINT OUT A MESSAGE EVERY 100 STEPS
        if i%100 == 0:

            print('Currently on step {}'.format(i))
            print('Accuracy is:')
            # Test the Train Model
            matches = tf.equal(tf.argmax(y_pred,1),tf.argmax(y_true,1))

            acc = tf.reduce_mean(tf.cast(matches,tf.float32))

            print(sess.run(acc,feed_dict={x:ch.test_images,y_true:ch.test_labels,hold_prob:1.0}))
            print('\n')
```

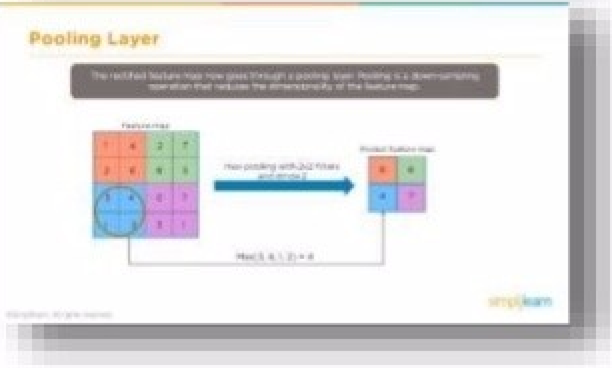
```
Currently on step 0
Accuracy is:
0.0979


Currently on step 100
Accuracy is:
0.4065


Currently on step 200
Accuracy is:
0.4654


Currently on step 300
Accuracy is:
0.5065


Currently on step 400
Accuracy is:
0.5251
```

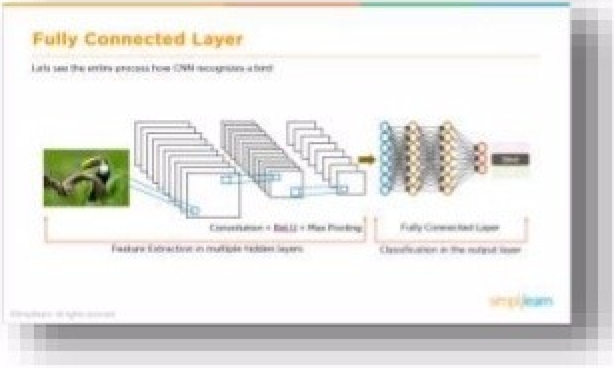
Key Takeaways

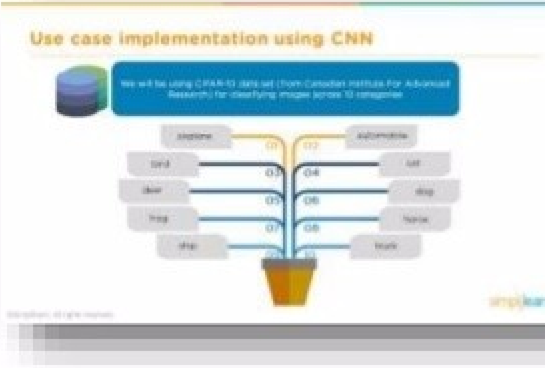












THANK YOU

For more information, visit

www.simplilearn.com

simplilearn