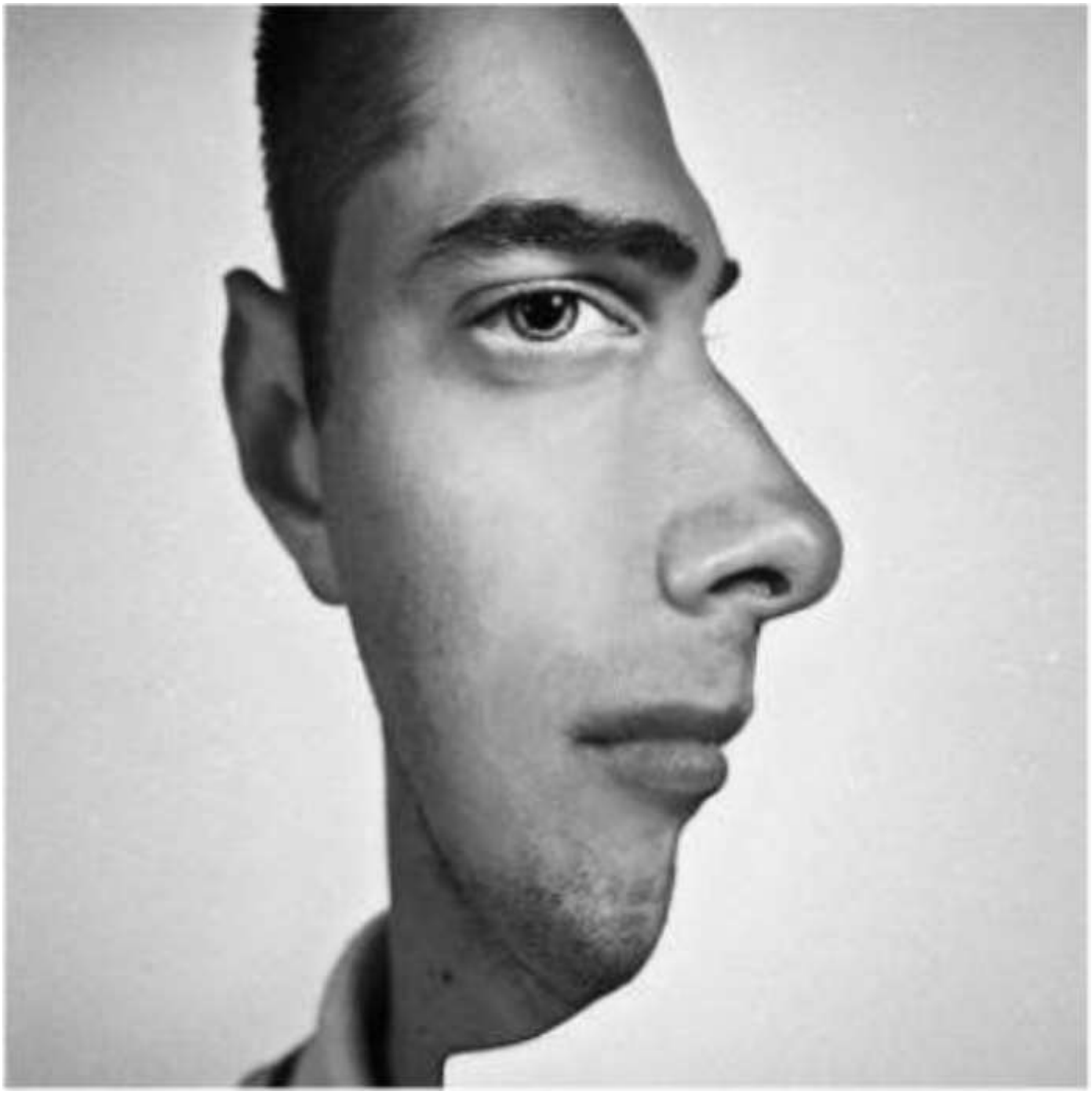
Convolutional Neural Network

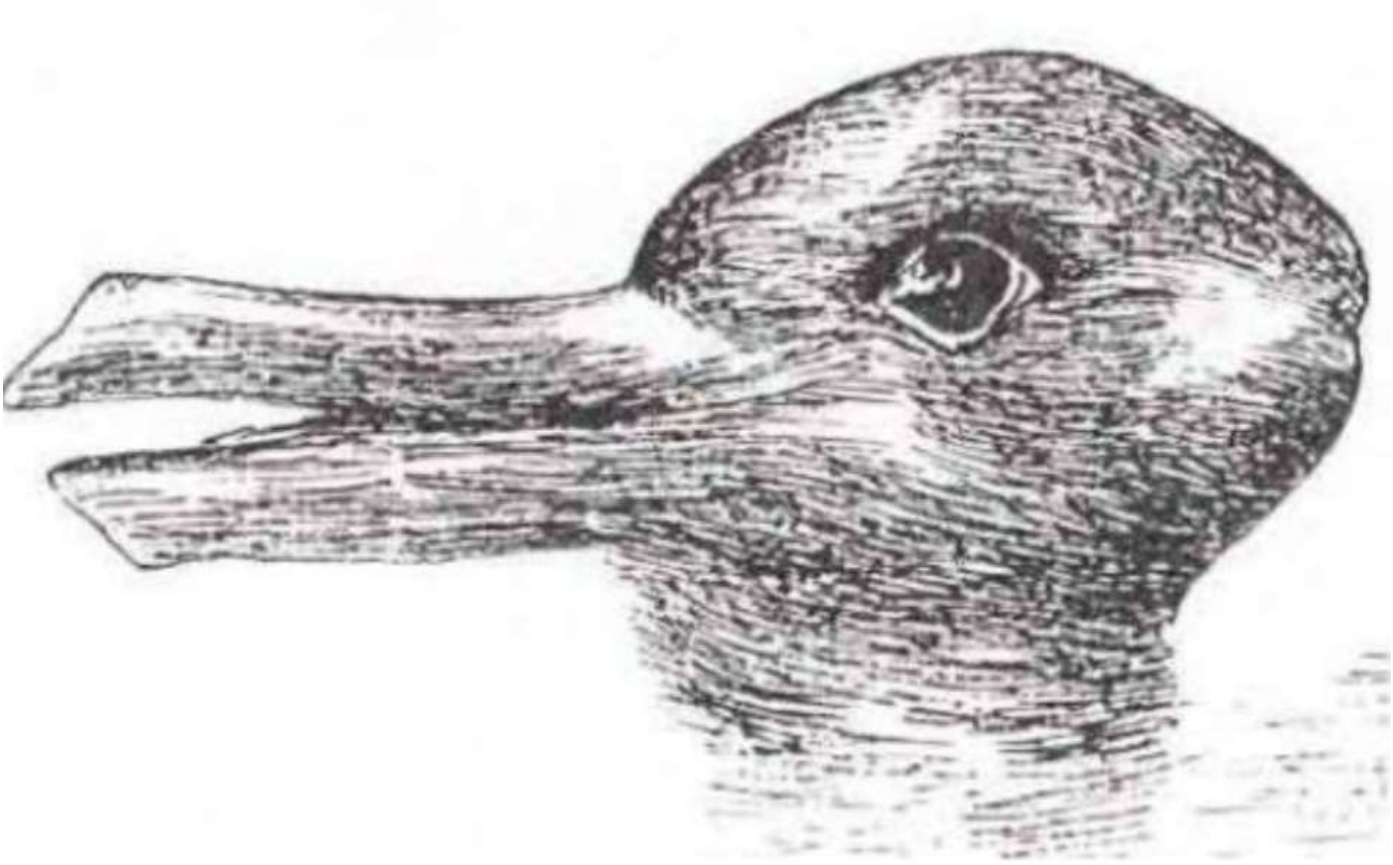
- What is convolutional neural Network
- Convolutional Operation
- Poolig Layer
- Flattening
- Full connection

What is convolutional neural Network (CNN)

In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery(Images). CNNs use a variation of multilayer operations designed to require minimal preprocessing.

Lets take example of some images:

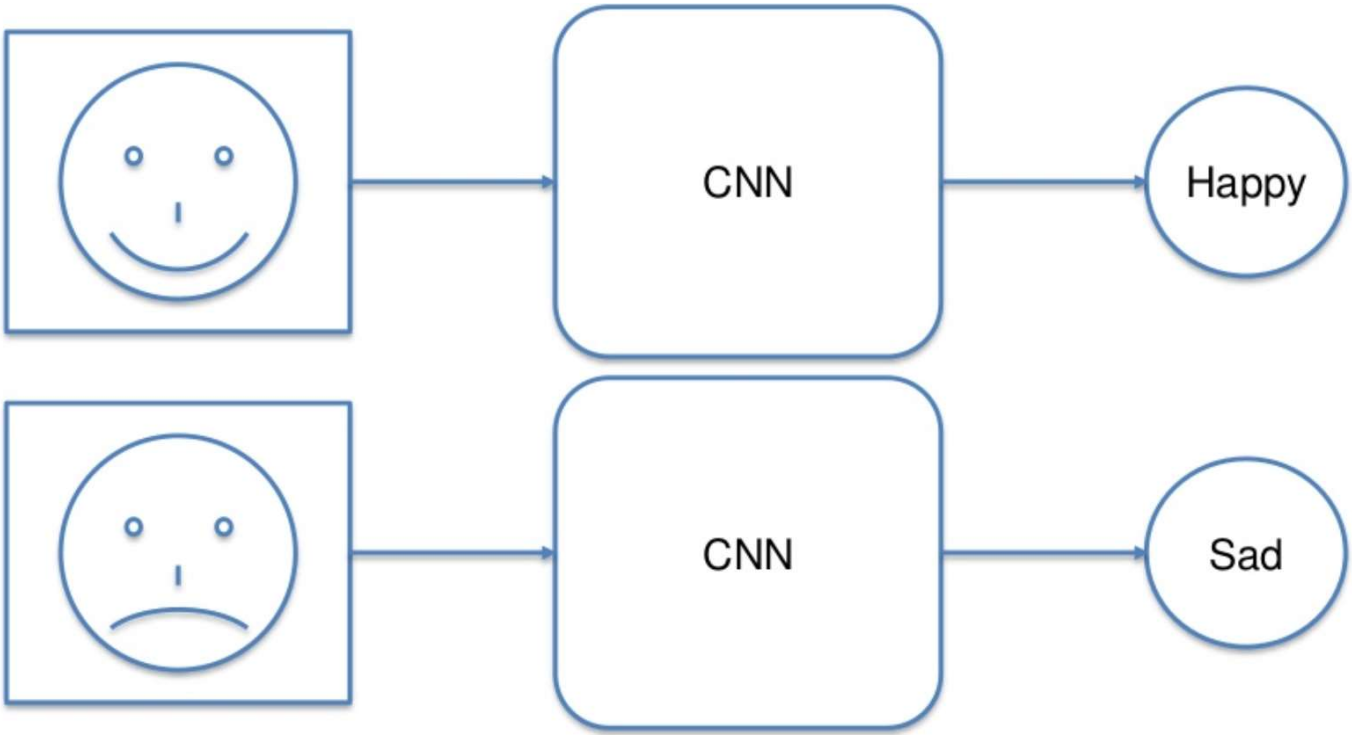




Even humans can diffreciate among images, so vast research is going on to make our Learning alogoritms to become stronger.

This is the drawback of current neural networks model, But they are efficient enough in 99% of the cases.

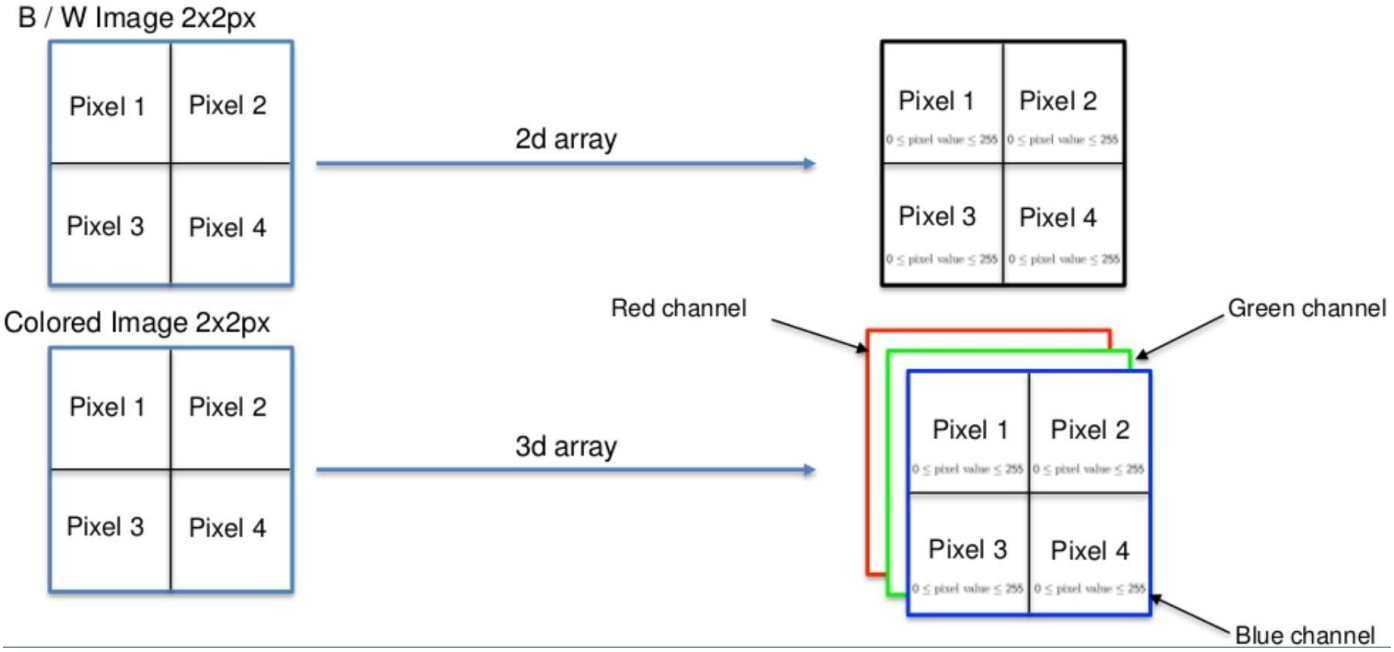
CNN Algoritm



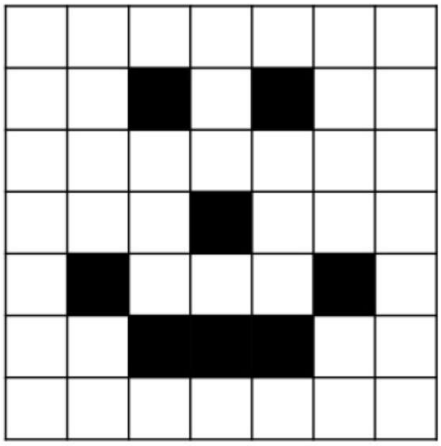
Images

A visible impression obtained by a camera, telescope, microscope, or other device, or displayed on a computer or video screen.

Resolution refers to the number of pixels in an image. Resolution is sometimes identified by the width and height of the image as well as the total number of pixels in the image

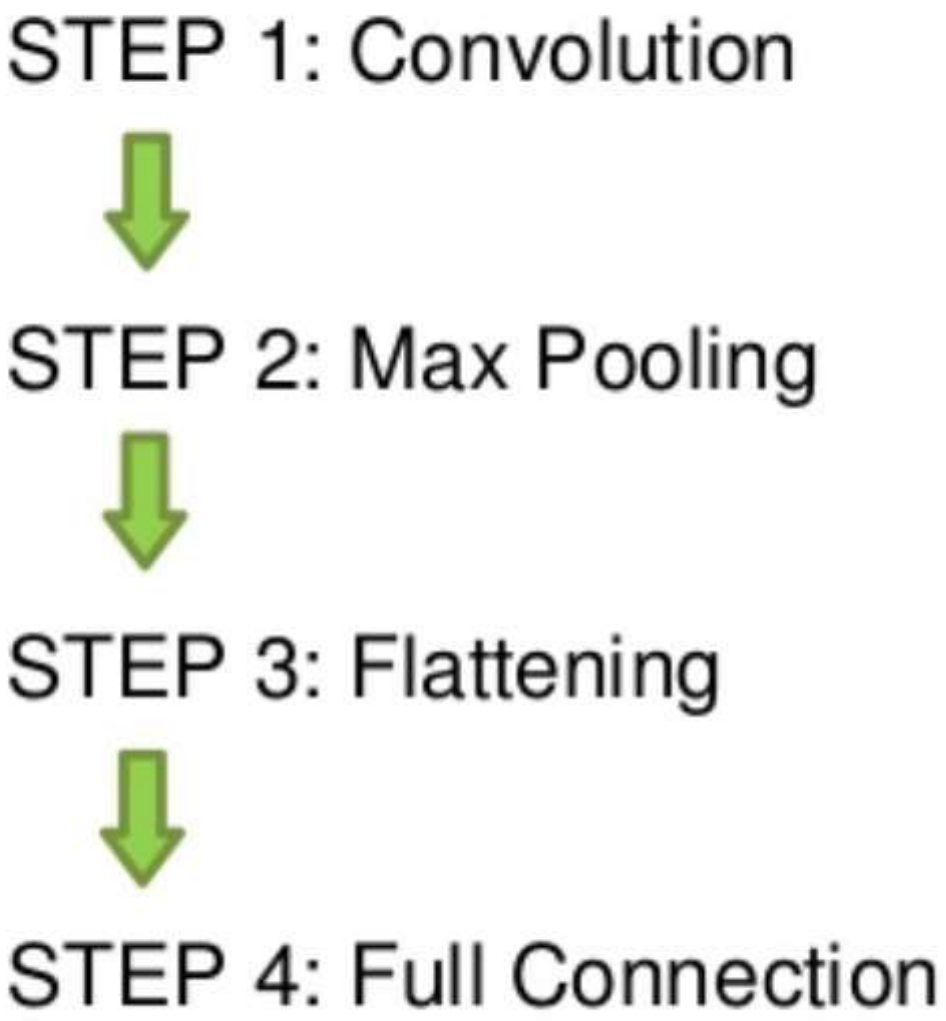


Example of Image we will be dealing in Deep Learning



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

Steps for Convolutional Neural Network

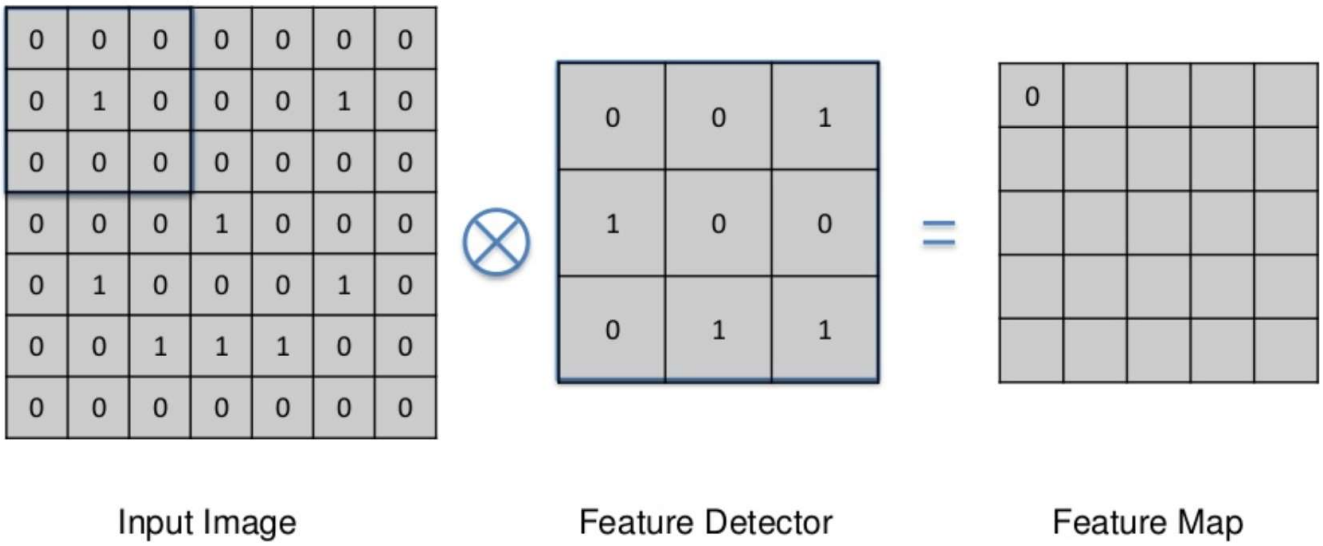


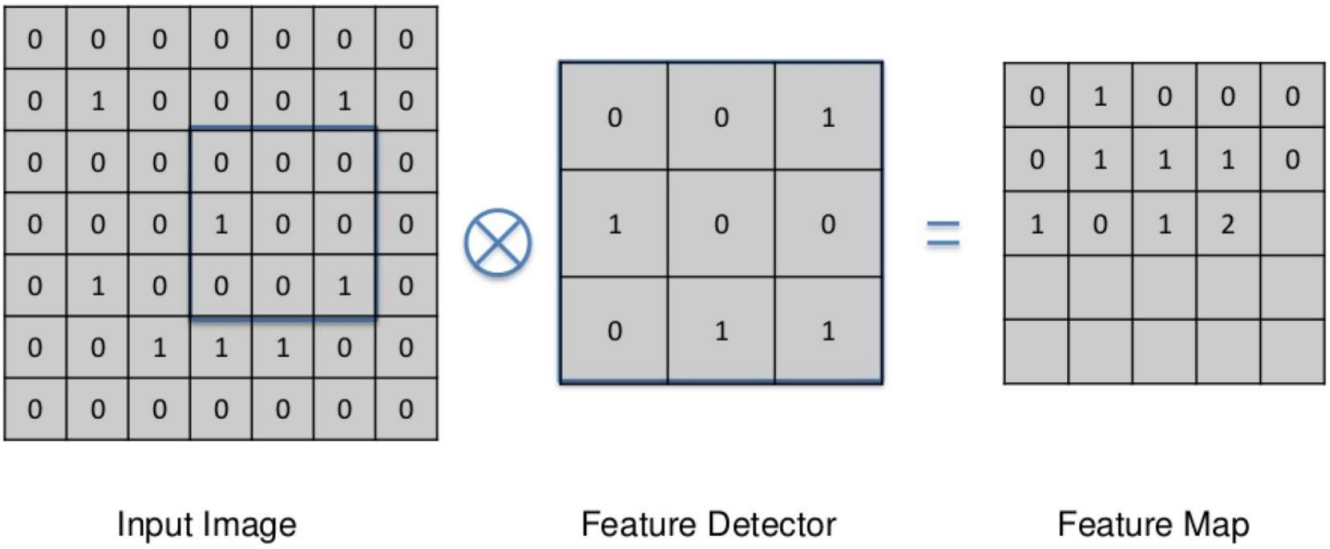
Step 1 - Convolution

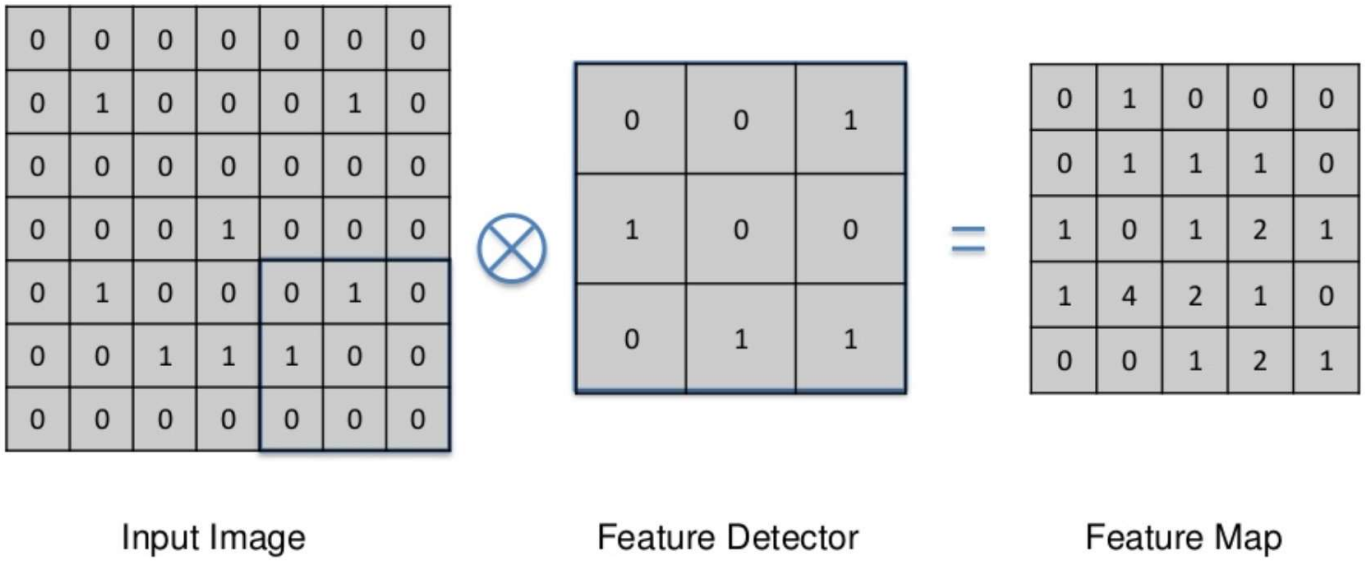
Convolution is a simple mathematical operation which is fundamental to many common image processing operators.

In image processing, a kernel, convolution matrix, or mask is a small matrix used for Convolution Operation.

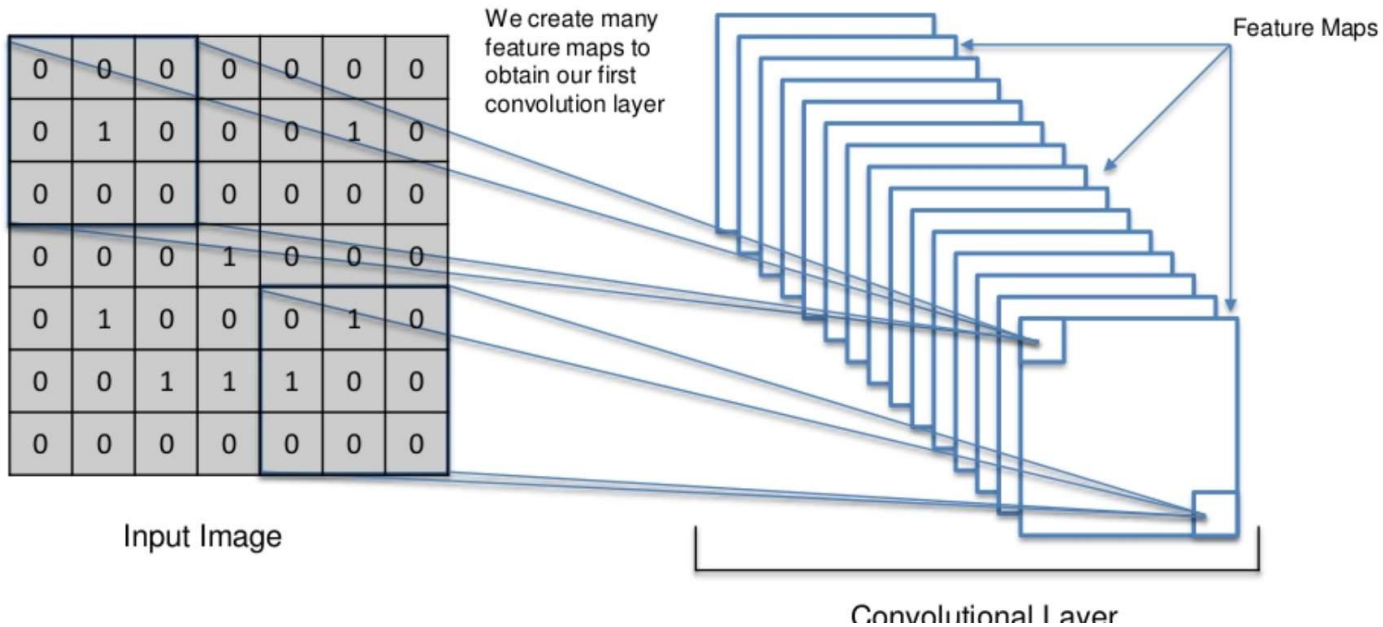
In deep learning it is also known as feature detector which gives us feature map. In a single convolution layer we can have as many number of feature maps as we want.







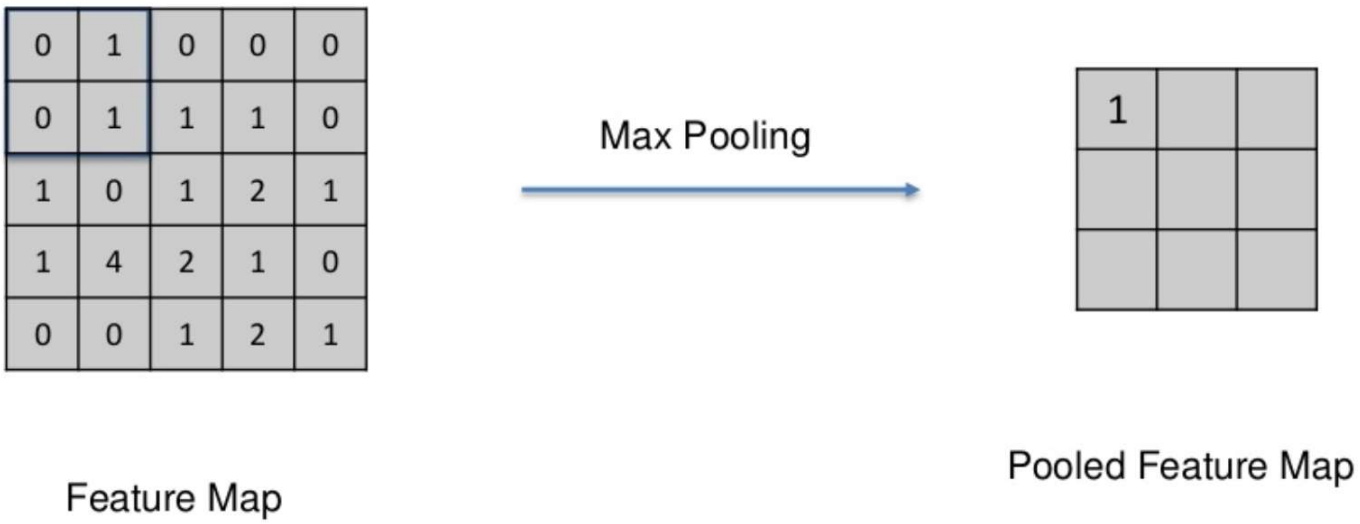
Now in Convolutional nural network we use convolutional layer which consist of convolution operation with fearture detectors to obtain feature maps.

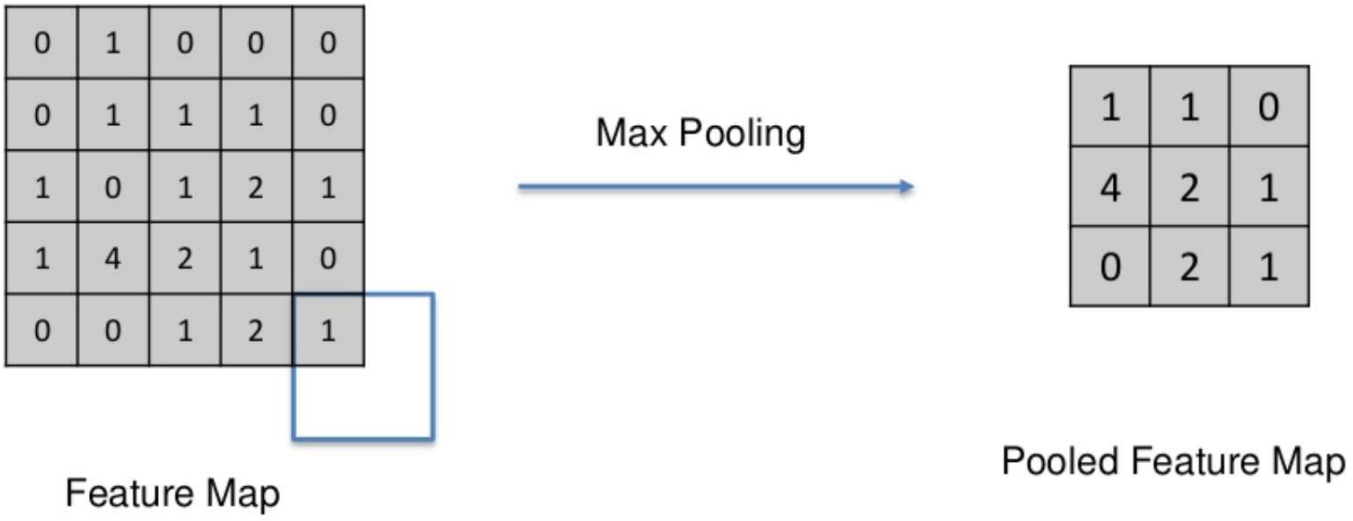


Maxpooling Operation

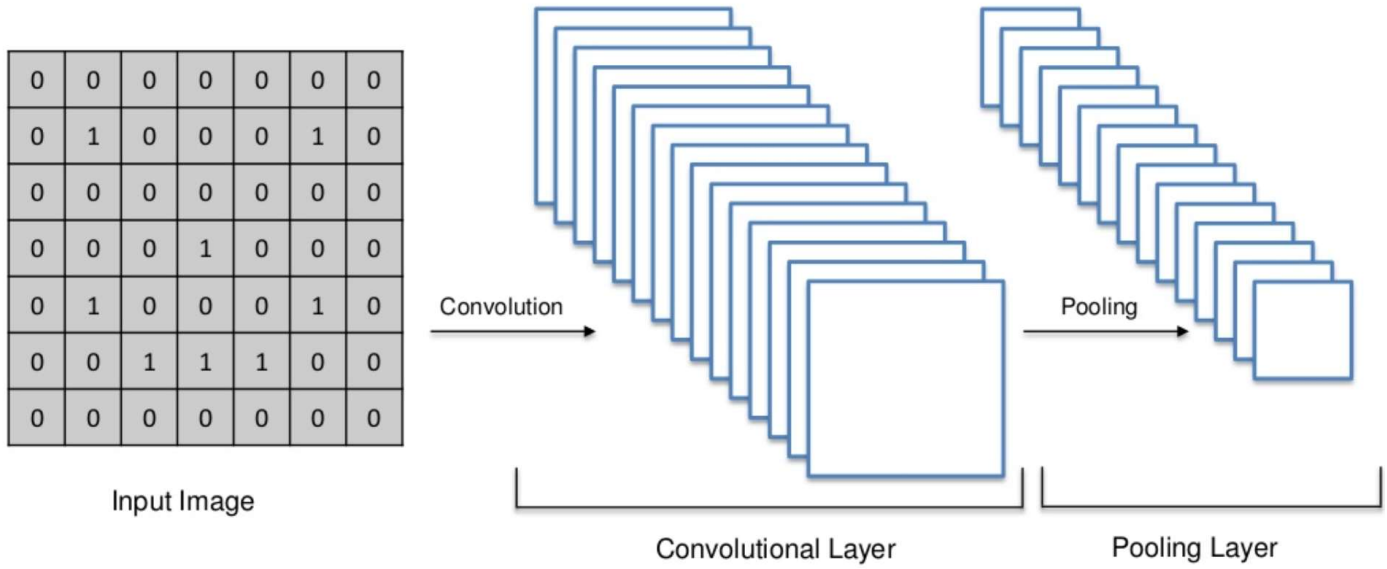
Then we perform maxpooling operation to our feature maps as below.

It chooses the maximum value among the window or kernel by traversing the whole image or array of image. In the below image kernel size is 2X2



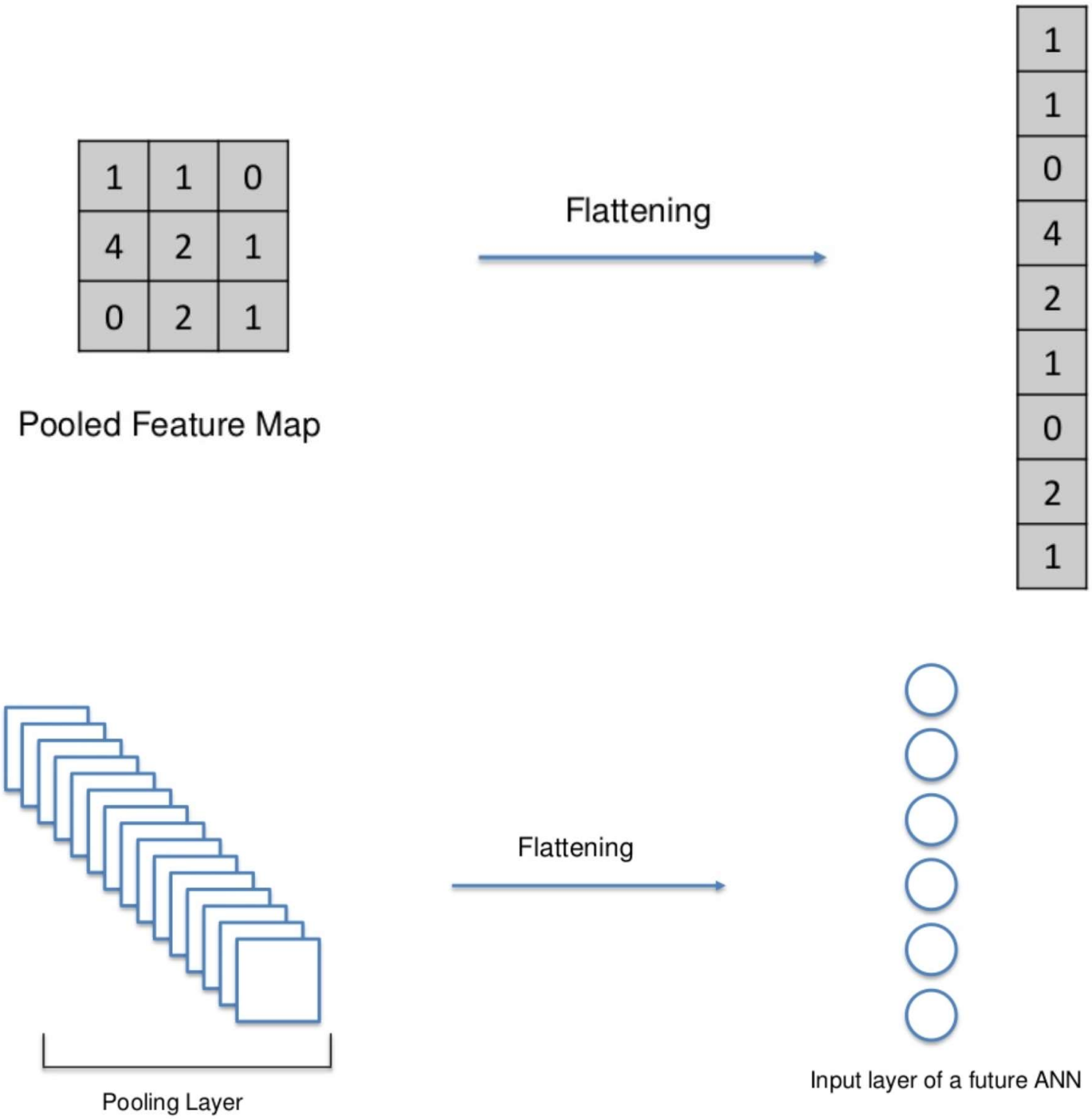


Resultant overview of Convolution and Maxpooling

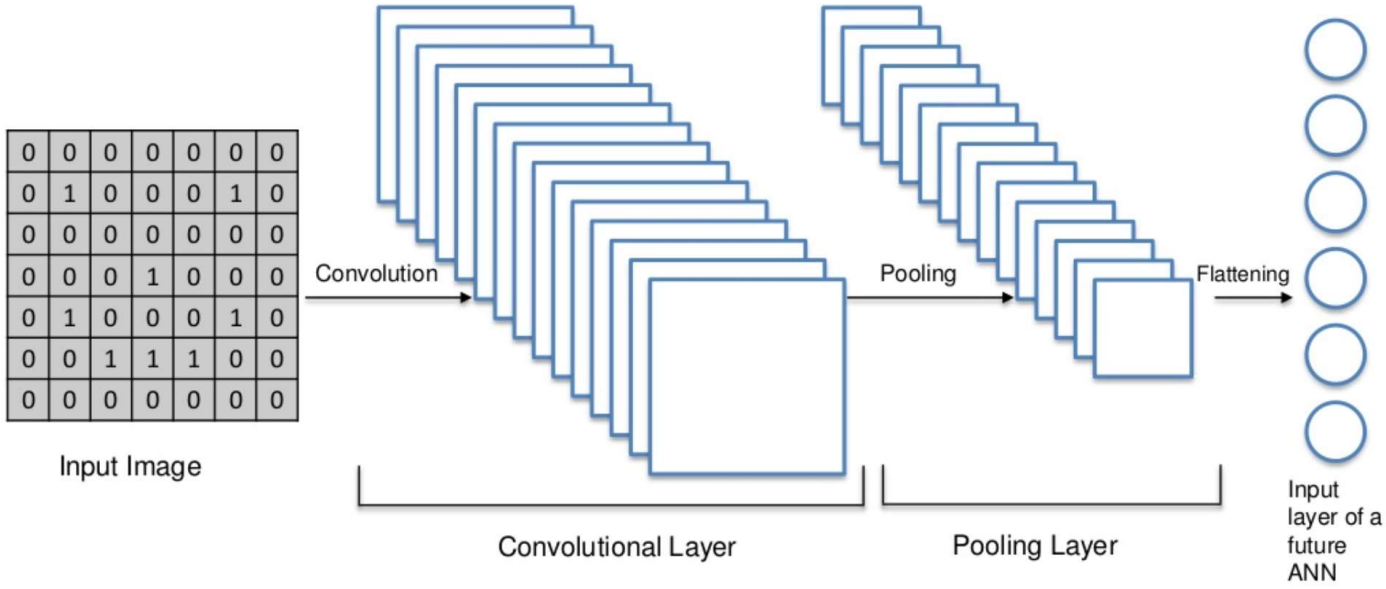


Flattening Operation

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

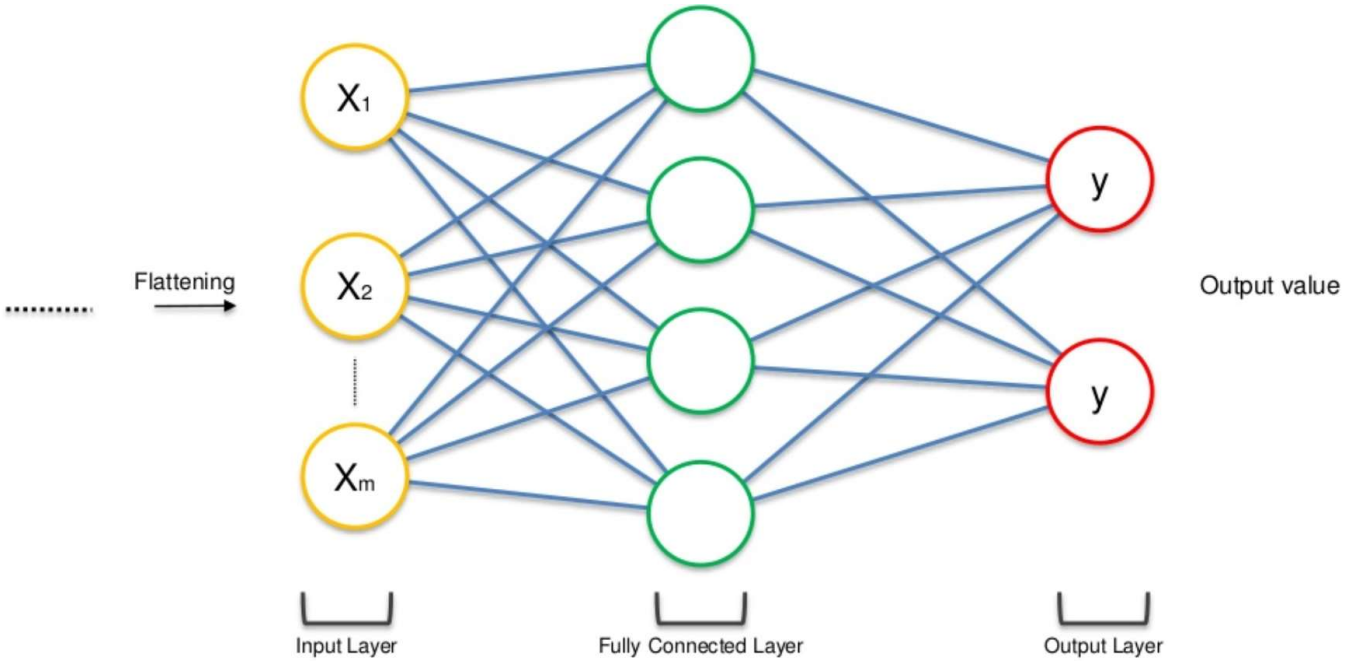


Whole convolutional neural network till now

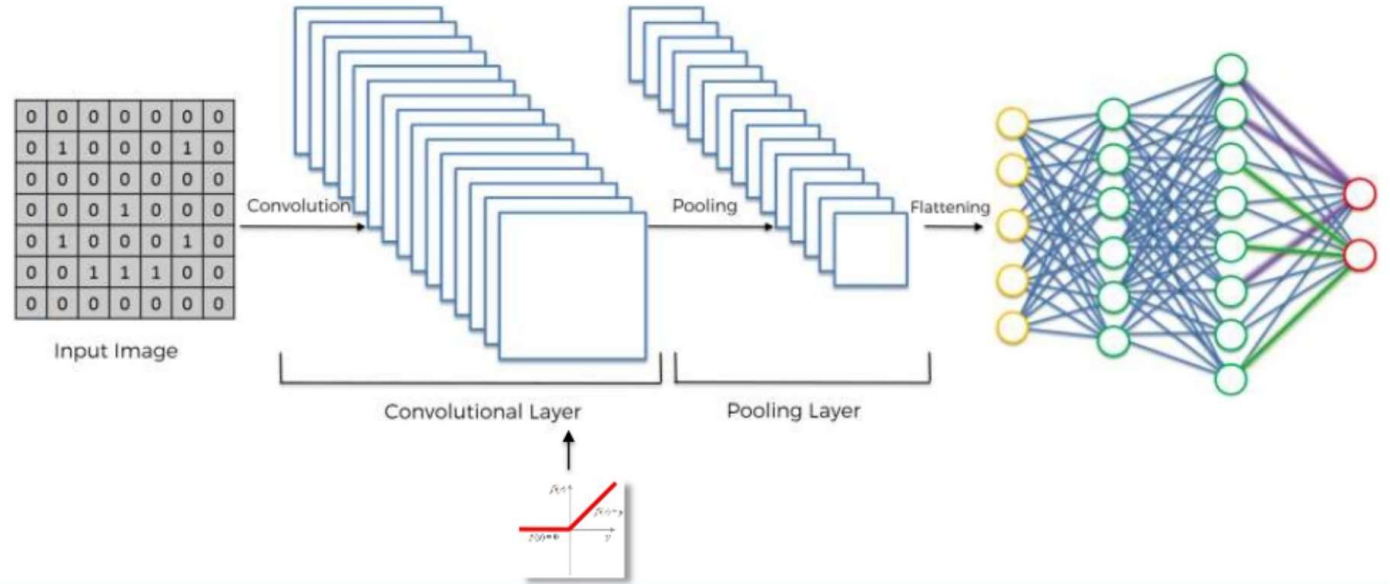


Full Connection

After flattening we add simple neural network layers as we saw in house prediction dataset to our convolutional neural network

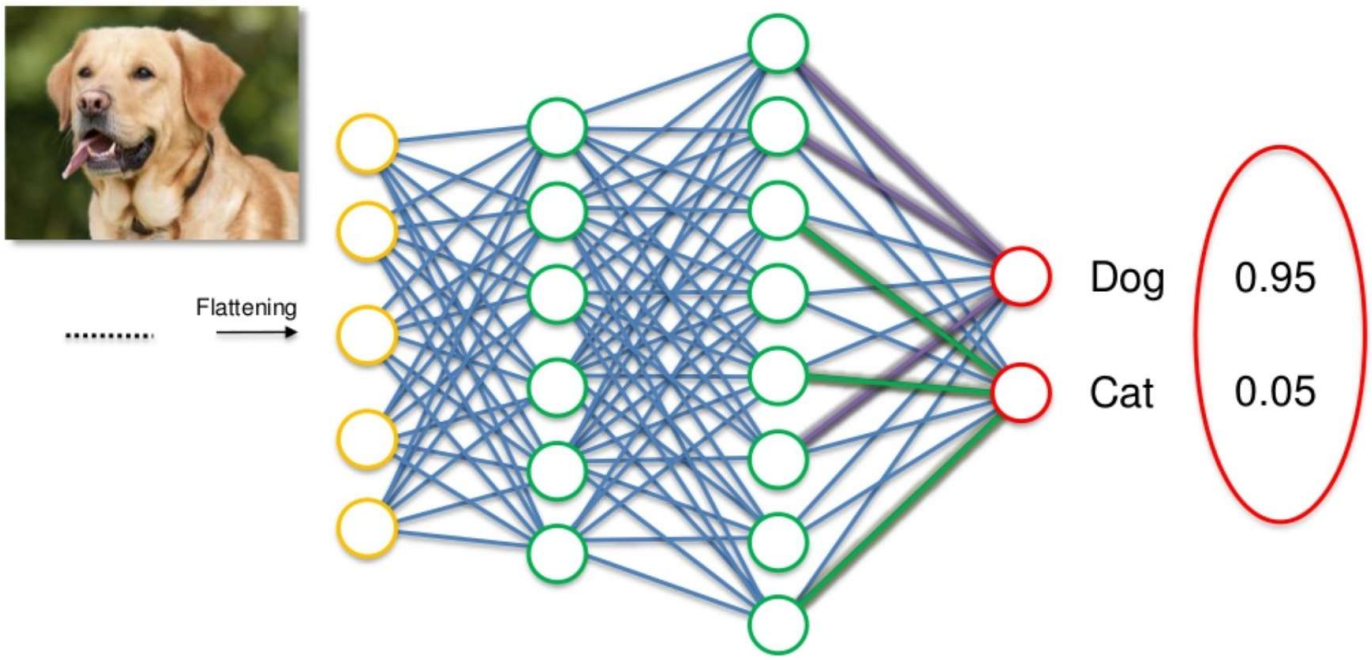


Complete Convolution Neural Network



In the final layer you will get the probability of the image form the dataset in which you have trained your network.

Example



MNIST Handwritten Digit Recognition in Keras

Setting up environment and dependencies

Run below commands in your terminal:

- conda install -y -c anaconda \ tensorflow-gpu h5py cudatoolkit=8
- pip install keras

Loading the modules

In [ ]:

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

# import matplotlib
# matplotlib.use('agg')
import matplotlib.pyplot as plt
%matplotlib inline
```

Initialising parameters

In [ ]:

```
batch_size = 16
num_classes = 10
epochs = 5

# input image dimensions
img_rows, img_cols = 28, 28
```

Loading and spilitting the 'MNIST' dataset

In [ ]:

```
# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Note: Here we are using dataset from the Keras package. You can use your own dataset.

Preprocessing the images

In [ ]:

```
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

Visualising the image samples of MNIST

In [ ]:

```
for i in range(9):
  plt.subplot(3,3,i+1)
  plt.tight_layout()
  plt.imshow(x_train[i,:,:,0], cmap='gray', interpolation='none')
  plt.title("Digit: {}".format(y_train[i]))
  plt.xticks([])
  plt.yticks([])
plt.show()
```

Binary Class to Categorical Class conversion

In [ ]:

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Building the model

In [ ]:

```
model = Sequential()
# Convolving the input feature vectors
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))

# Max-pooling the convolved feature vector
model.add(MaxPooling2D(pool_size=(2, 2)))

# Applying dropout
model.add(Dropout(0.25))

# Flattening the feature vector, i.e. the image into a 1-dimentional vector
model.add(Flatten())

# Adding a neural network fully connected layer
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Compiling the model

In [ ]:

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

Training the model

In [ ]:

```
model.fit(x_train[:5000], y_train[:5000],
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test[:1000], y_test[:1000]))
```

Evaluating the model

In [ ]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Testing the final model

In [ ]:

```
for i in range(9):
    img = x_test[i,:,:,0]
    plt.subplot(3,3,i+1)
    plt.tight_layout()
    plt.imshow(img, cmap='gray', interpolation='none')
    plt.title("Digit: {}".format(y_train[i]))
    plt.xticks([])
    plt.yticks([])
    y_pred = model.predict(img)
    print('Ground truth:', y_test)
    print('Predicted value:', y_pred)
plt.show()
```

References and Resources:

- https://www.coursera.org/specializations/deep-learning (https://www.coursera.org/specializations/deep-learning)
- https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py (https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py)
- https://towardsdatascience.com/tagged/deep-learning (https://towardsdatascience.com/tagged/deep-learning)