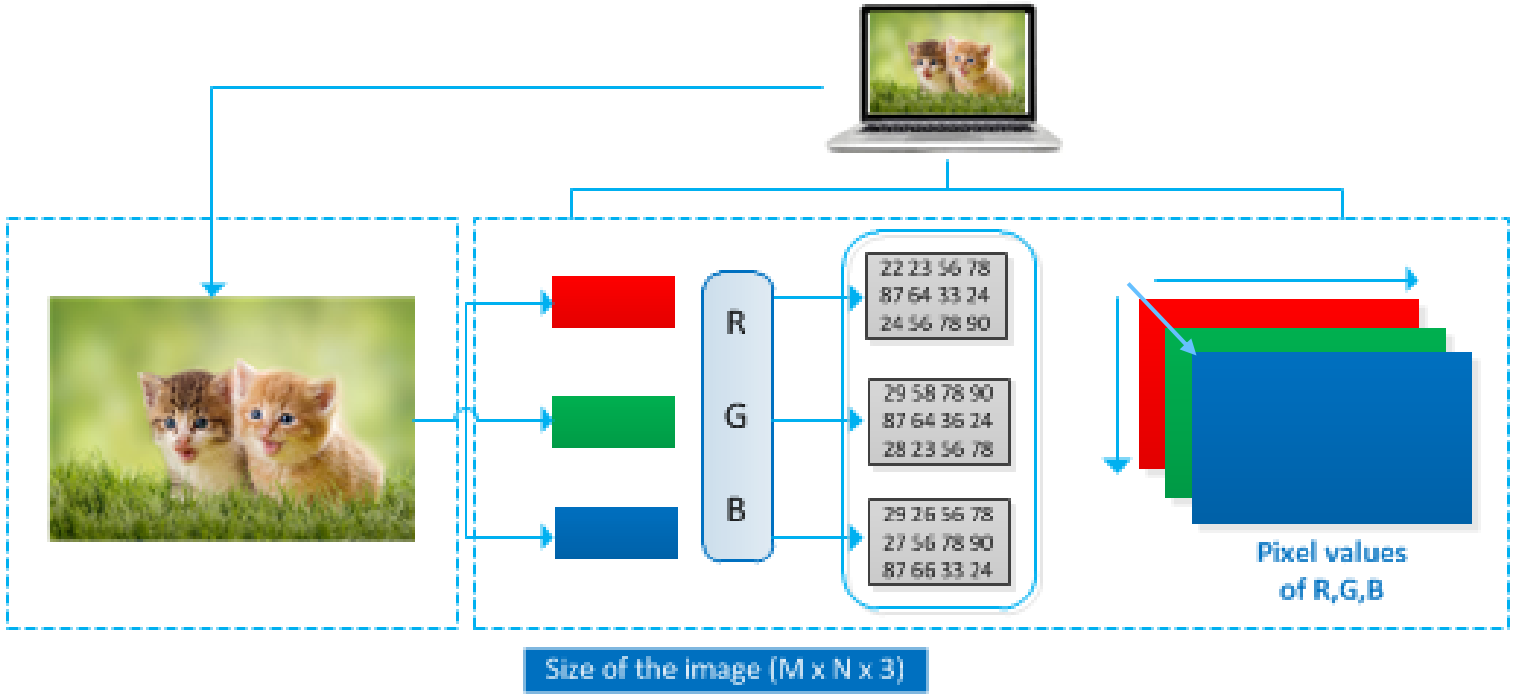
Image Classification

Dr. Su Su Maung
CEIT Department
Yangon Technological University

Outline of Lecture12

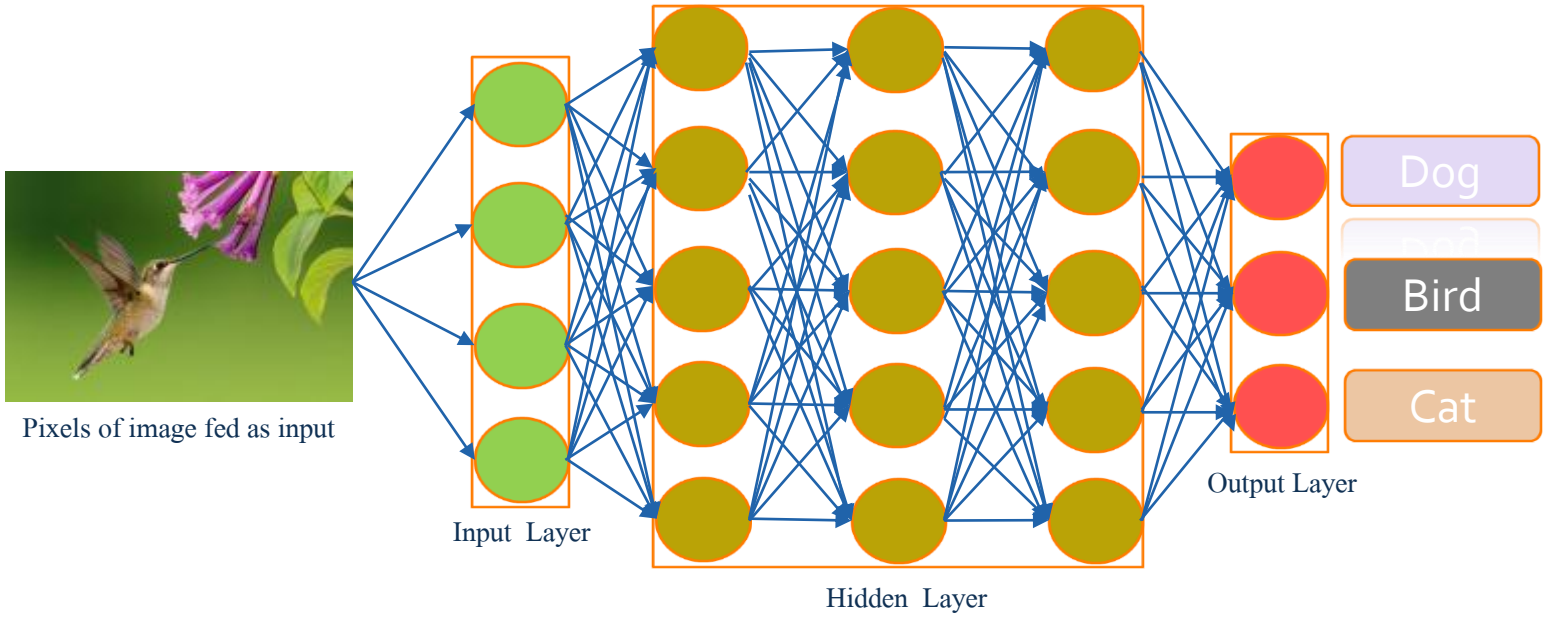
- How a Computer Reads an Image?
- How image recognition works?
- Convolutional Neural Network
- Layers in Convolution Neural Network

How a Computer Reads an Image?



How image recognition works?

Convolution Neutral Network recognizes the object in an image.

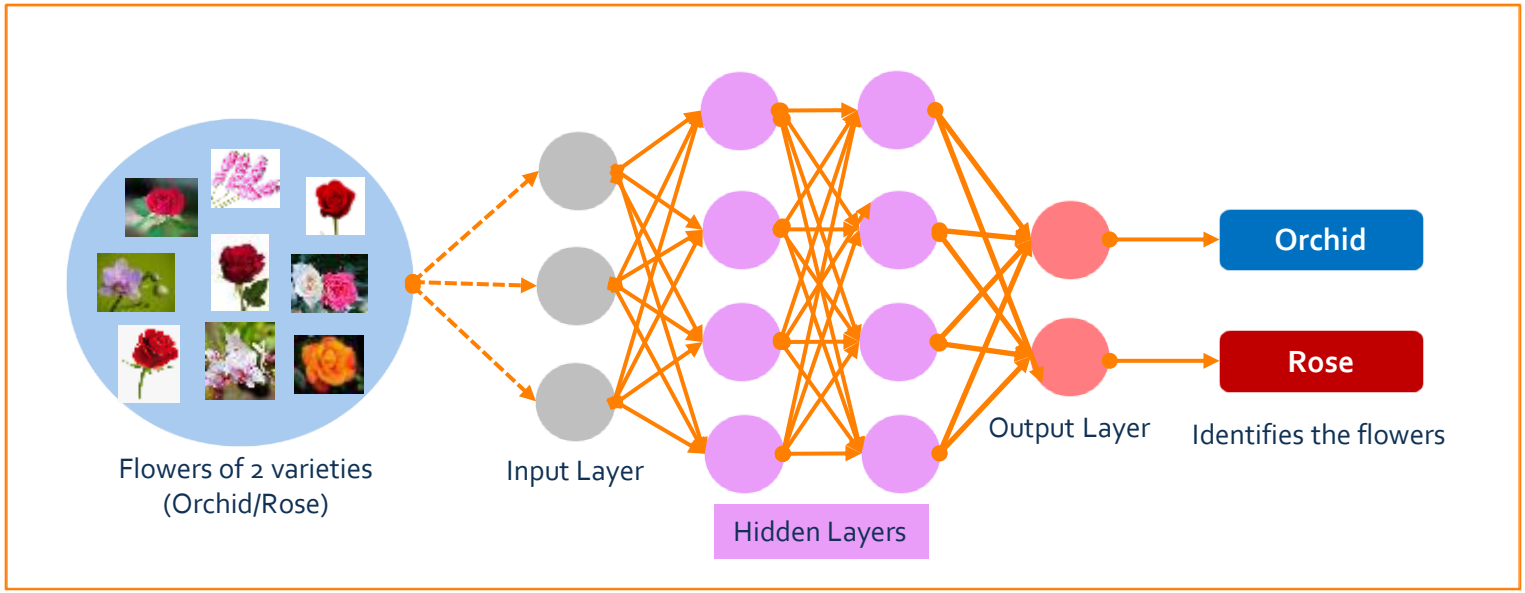


Convolutional Neural Network

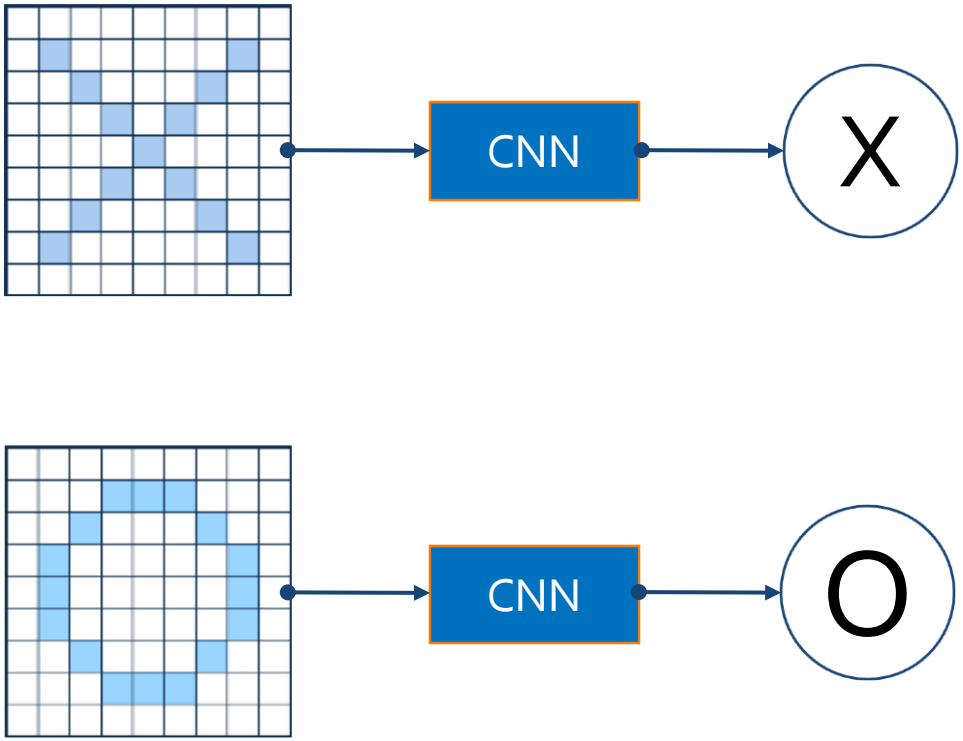
- Convolutional neural networks (CNN) and deep CNNs have recently shown an explosive popularity partially due to its success in image classification.
- They have also been successfully applied to other computer vision fields, such as object detection, face recognition, and pedestrian detection.

What is Convolutional Neural Network?

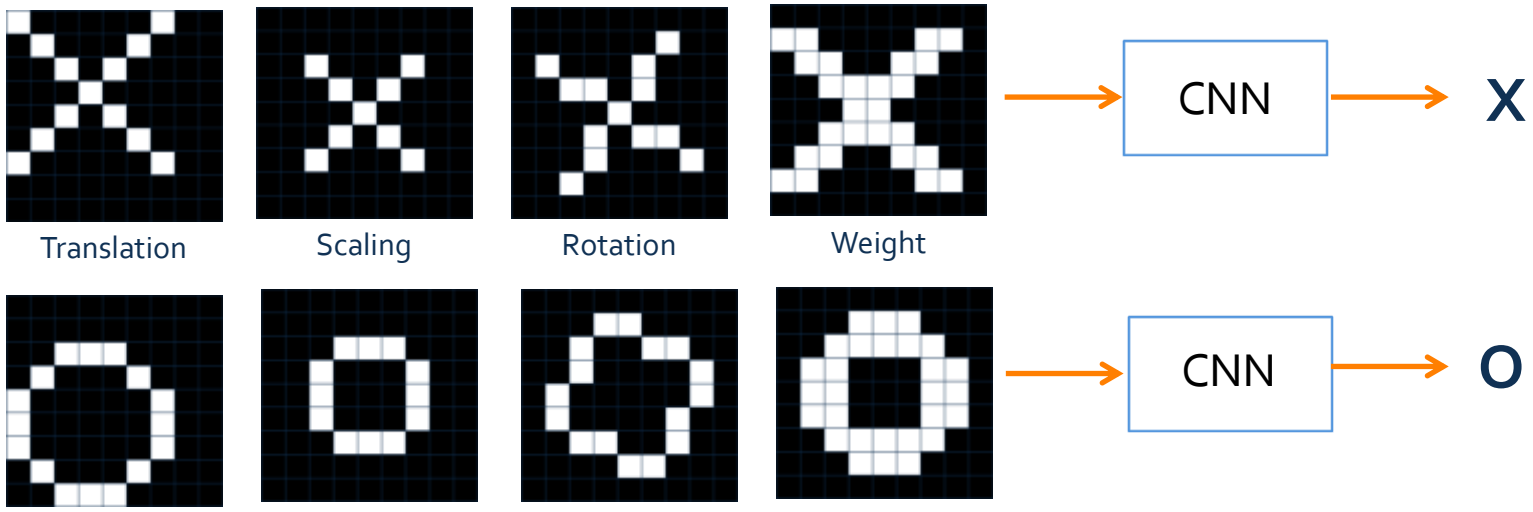
CNN is a feed forward neural network that is generally used to analyze visual images by Processing data with grid like topology. A CNN is also known as a "ConvNet".



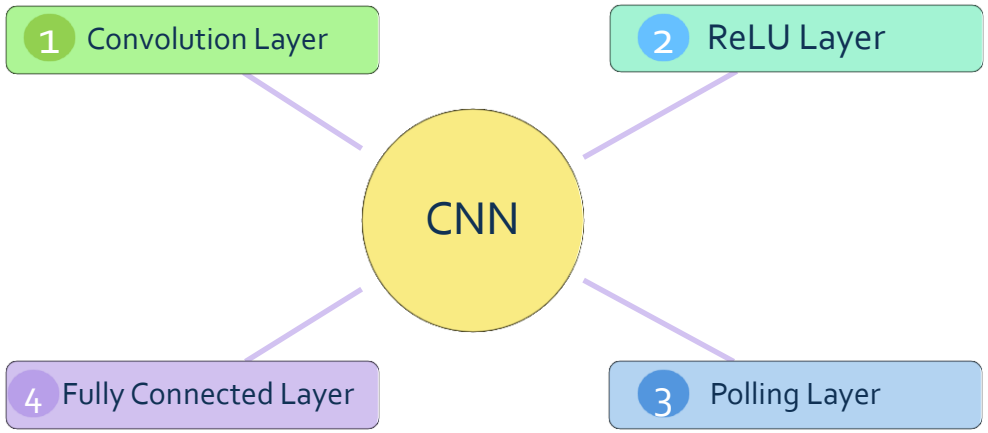
How CNN works?



How CNN works?

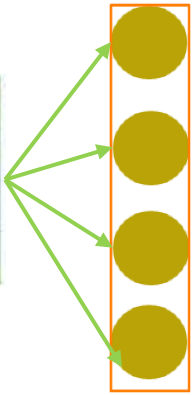


Layers in Convolution Neural Network



Input layer of CNN

Pixels of image fed as input

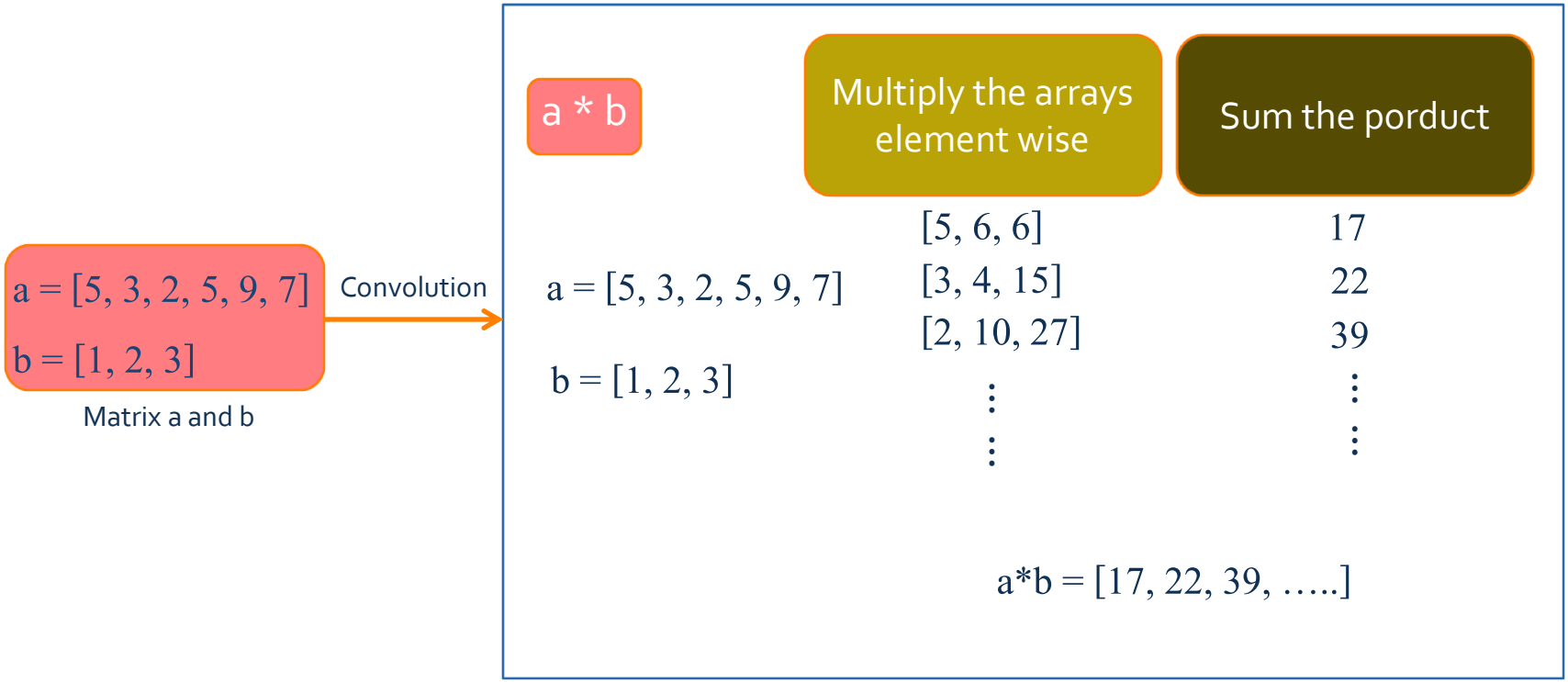


Input Layer

Input layer accepts the pixels of the image as input in the form of arrays

Convolution

The convolution operation using 2 matrices a and b of 1 dimension



Convolution (cont.)

CNN-Convolution

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

Filter 1

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

6 x6 image

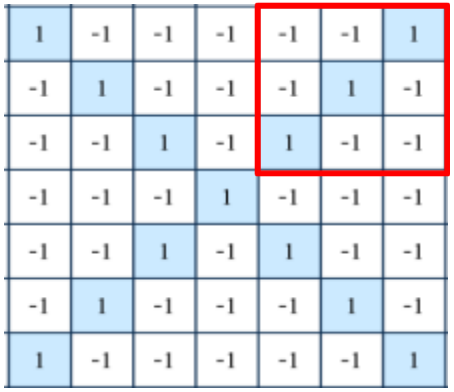
$$\frac{1+1+1+1+1+1+1+1+1}{9} = 1$$

1

Convolution (cont.)

CNN-Convolution

Stride=1



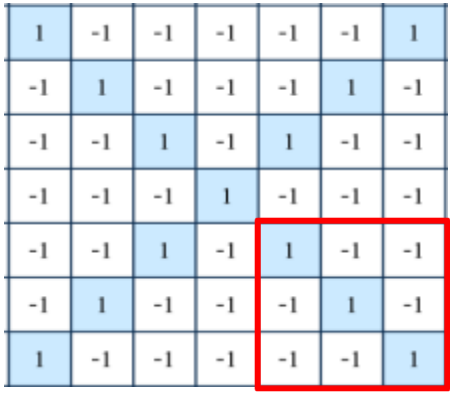
6 x6 image

Filter 1

Convolution (cont.)

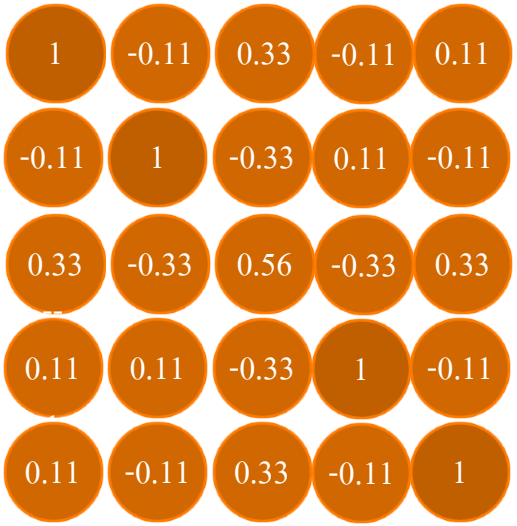
CNN-Convolution

Stride=1



6 x6 image

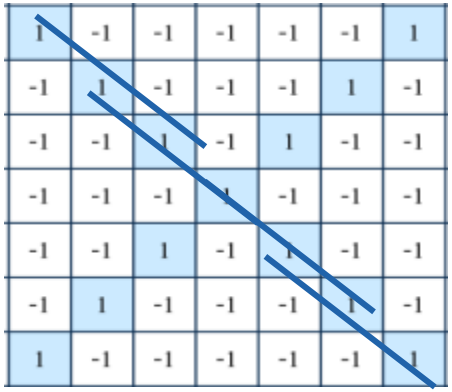
Filter 1



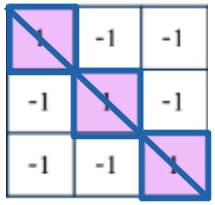
Convolution (cont.)

CNN-Convolution

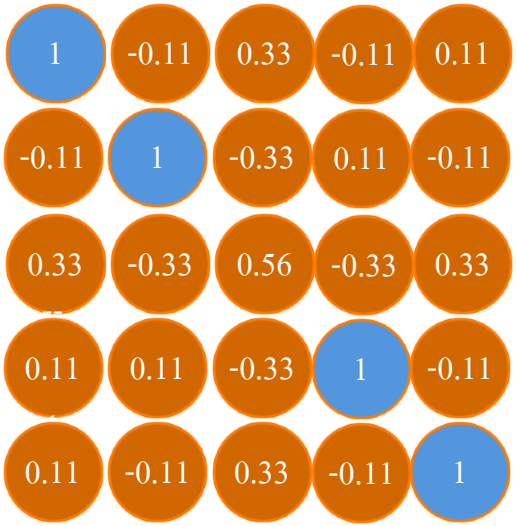
Stride=1



6 x6 image

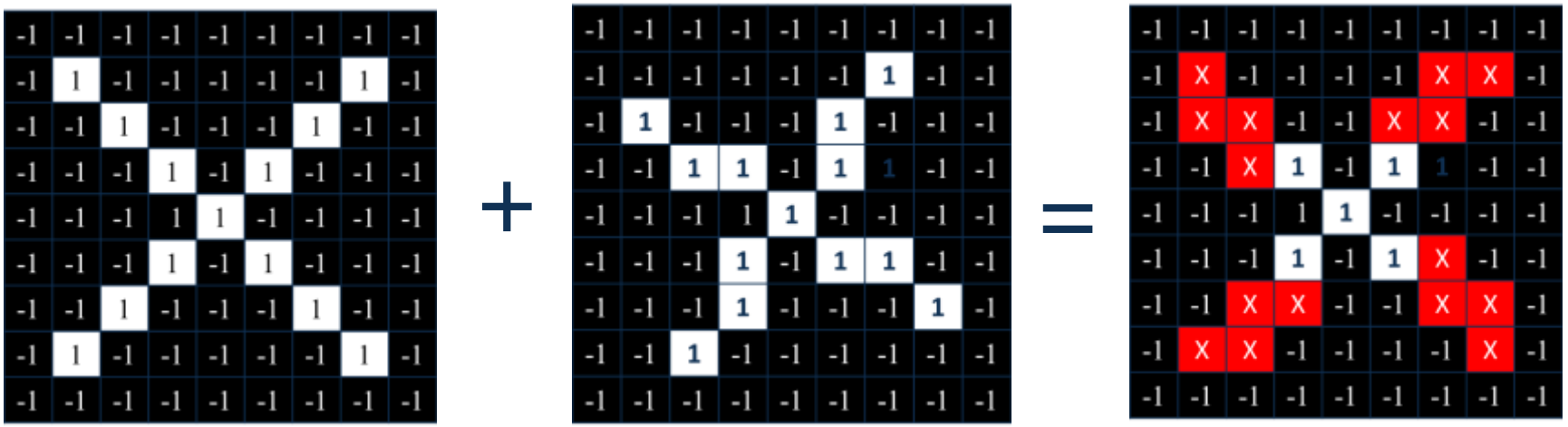


Filter 1



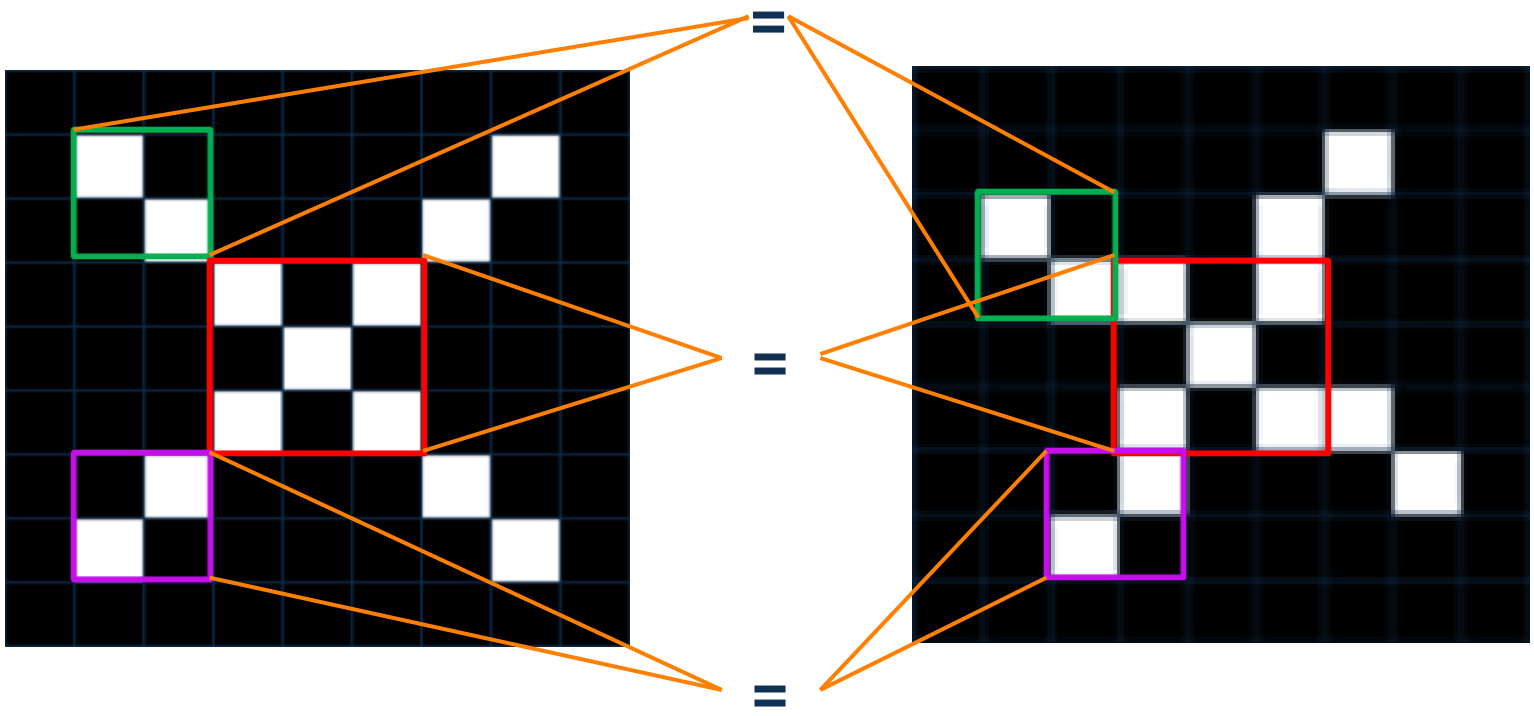
What computers see?

Using normal techniques, computers compare these images as:



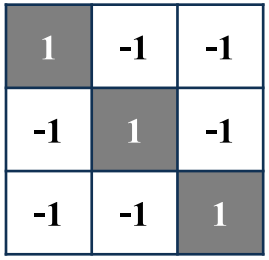
How CNN work?

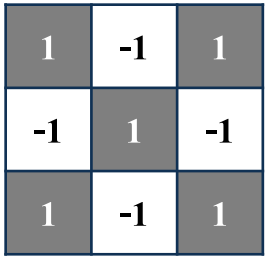
CNN compares the images piece by piece. The pieces that it looks for are called features. By finding rough feature matches, in roughly the same position in two images. CNN gets a lot better at seeing similarity than whole image matching schemes

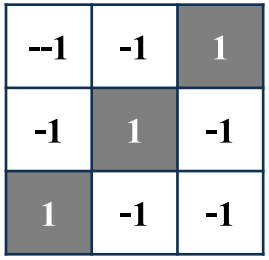


How CNN work?

ConvNet takes the following three filters as feature

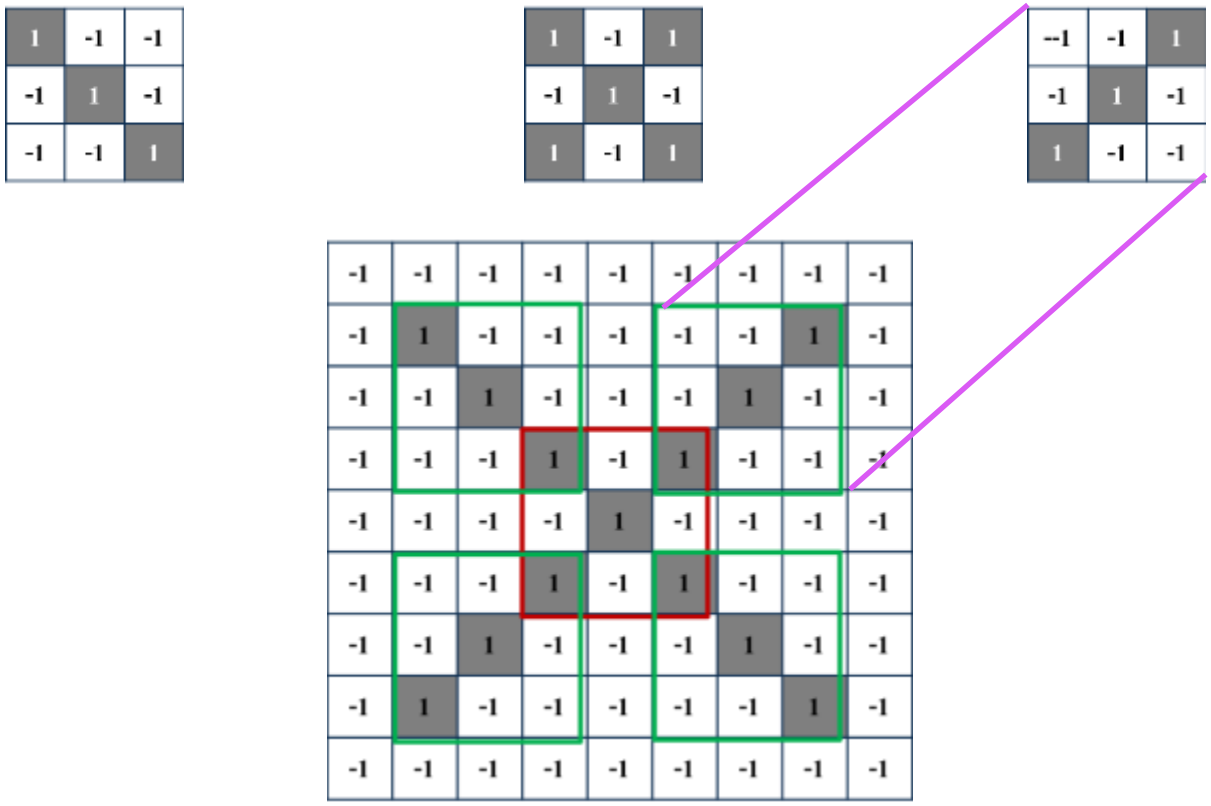






How CNN work?

- There are small pieces of the bigger image. We choose a feature and put it on the input image. If it matches then the image is classified correctly.



Convolution with one filter

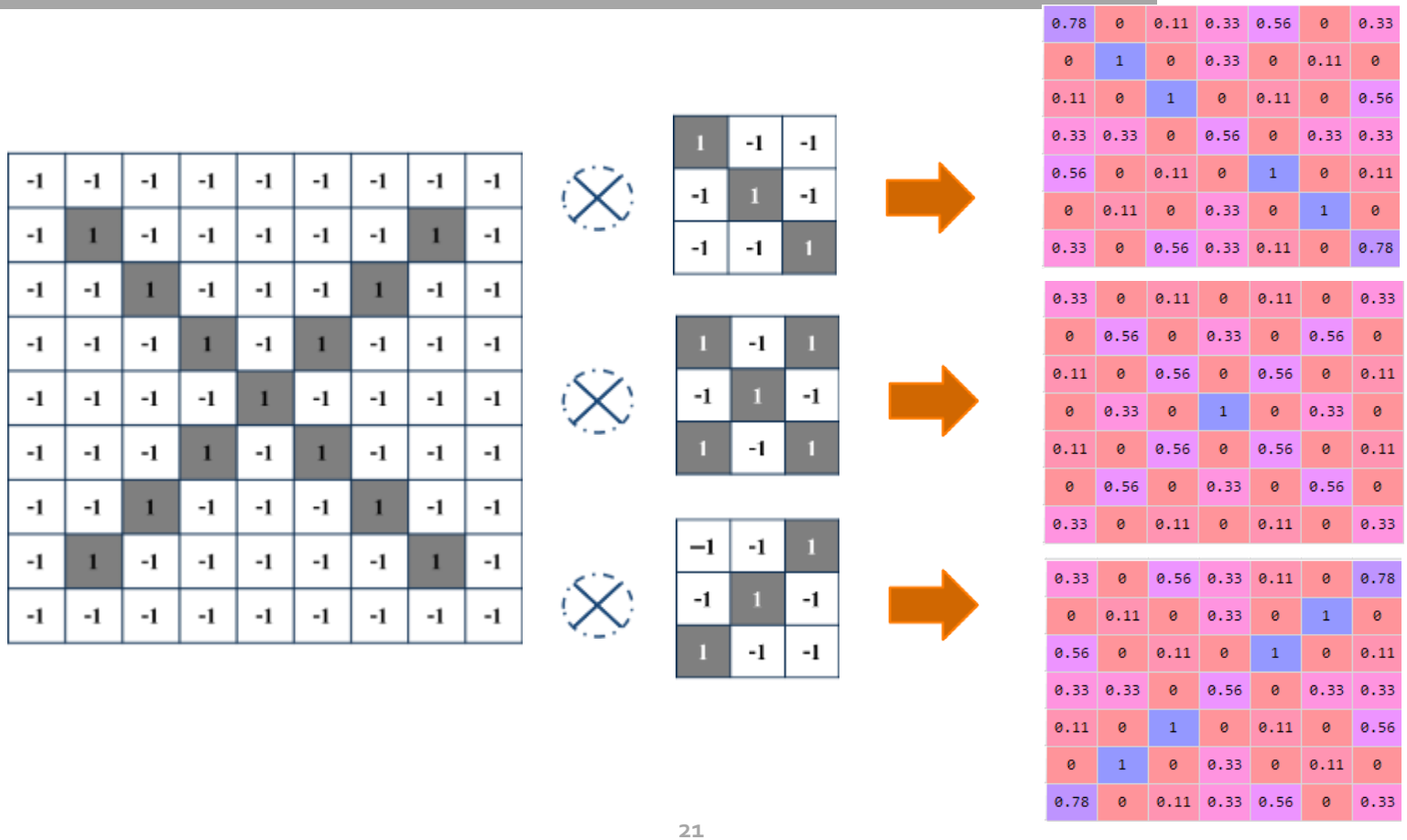
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

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

0.78	-0.11	0.11	0.33	0.56	-0.11	0.33
-0.11	1	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1	-0.33	0.11	-0.11	0.56
0.33	0.33	-0.33	0.56	-0.33	0.33	0.33
0.56	-0.11	0.11	-0.33	1	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1	-0.11
0.33	-0.11	0.56	0.33	0.11	-0.11	0.78

Convolution layer output

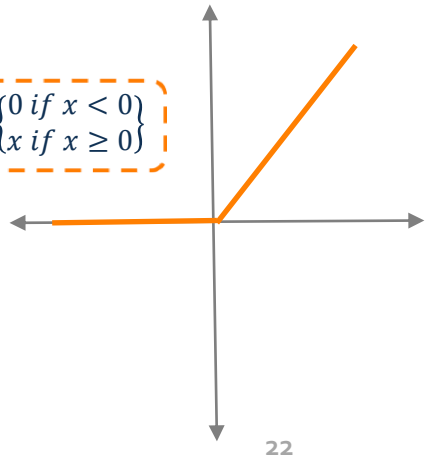
Similarly, we will perform the same convolution with every other filters



ReLU layer

- In this layer, every negative values from the filtered images are replaced with zeros.
- This is done to avoid values from summing up to zero.
- Rectified Linear Unit (ReLU) transform function only activates a node if the input is above a certain quantity, while the input is below zero, the output is zero, but when the input rises above certain threshold, it has a liner relationship with the dependent variable.

$$f(x) = \begin{cases} 0 \; if \; x < 0 \\ x \; if \; x \geq 0 \end{cases}$$



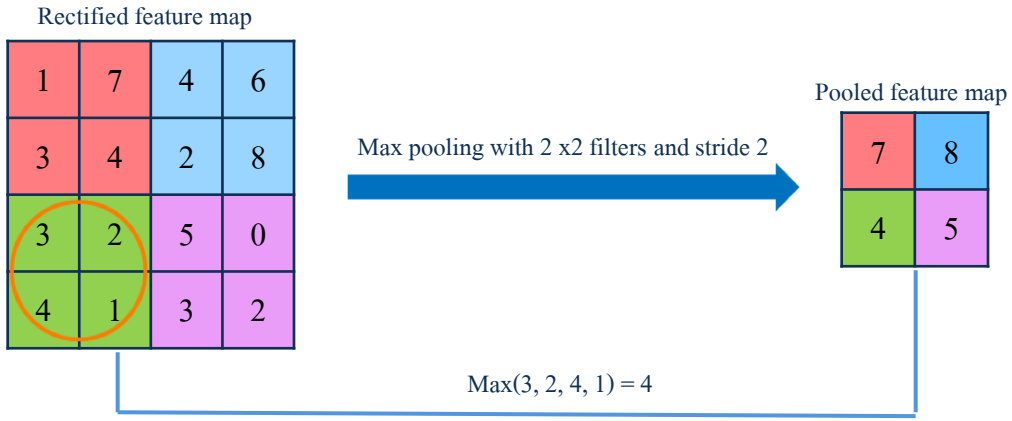
Output after passing through ReLU layer

0.78	-0.11	0.11	0.33	0.56	-0.11	0.33
-0.11	1	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1	-0.33	0.11	-0.11	0.56
0.33	0.33	-0.33	0.56	-0.33	0.33	0.33
0.56	-0.11	0.11	-0.33	1	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1	-0.11
0.33	-0.11	0.56	0.33	0.11	-0.11	0.78

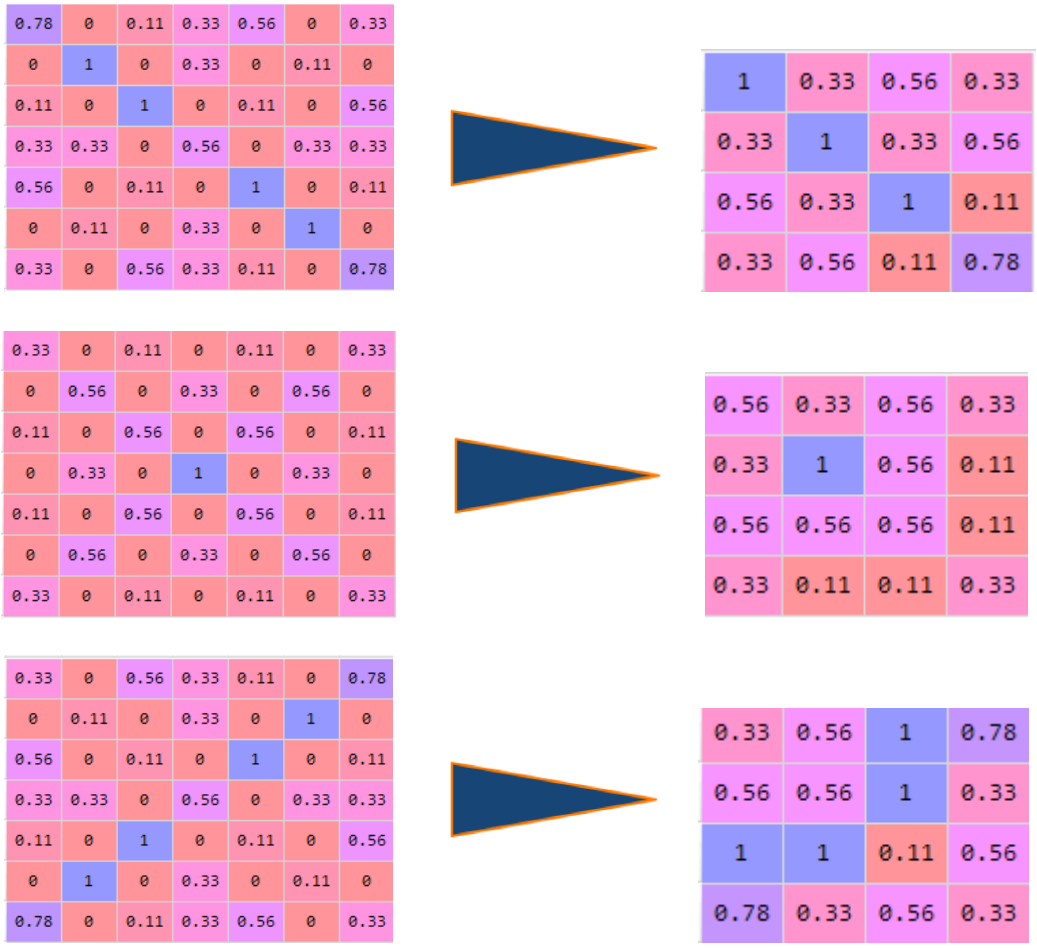
0.78	0	0.11	0.33	0.56	0	0.33
0	1	0	0.33	0	0.11	0
0.11	0	1	0	0.11	0	0.56
0.33	0.33	0	0.56	0	0.33	0.33
0.56	0	0.11	0	1	0	0.11
0	0.11	0	0.33	0	1	0
0.33	0	0.56	0.33	0.11	0	0.78

Pooling layer

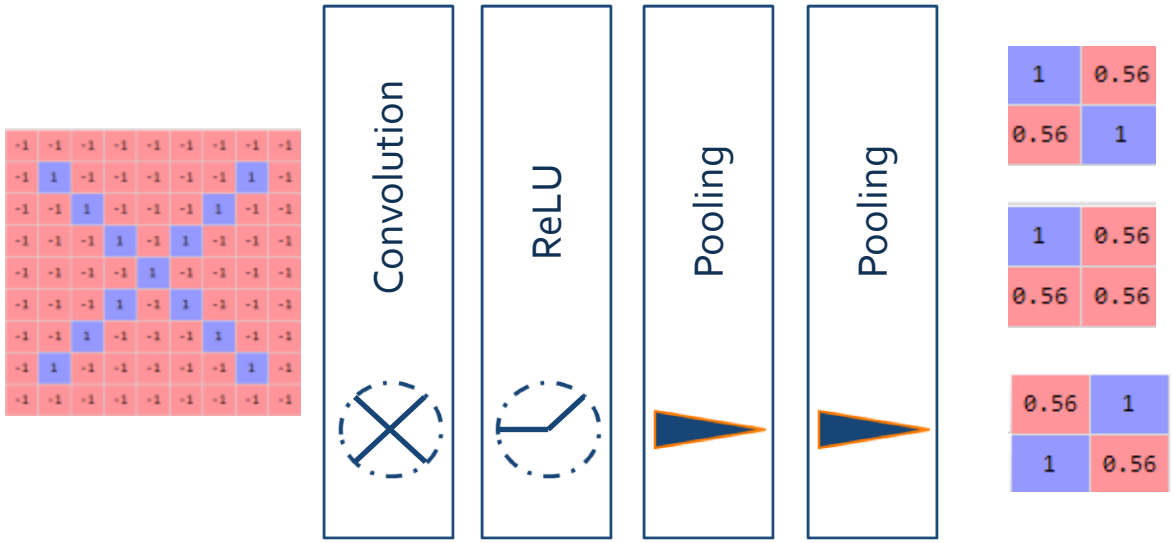
Pooling is a down-sampling operation that reduces the dimensionality of the feature map.



Output after passing through pooling layer

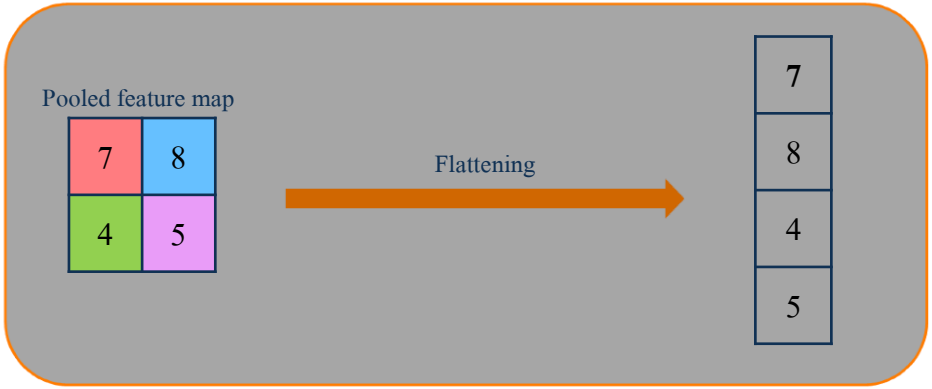


Stacking up the layers



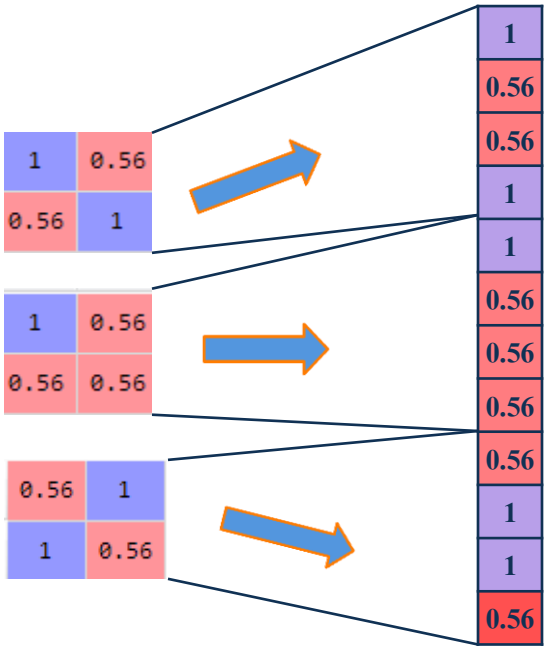
Flattening

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



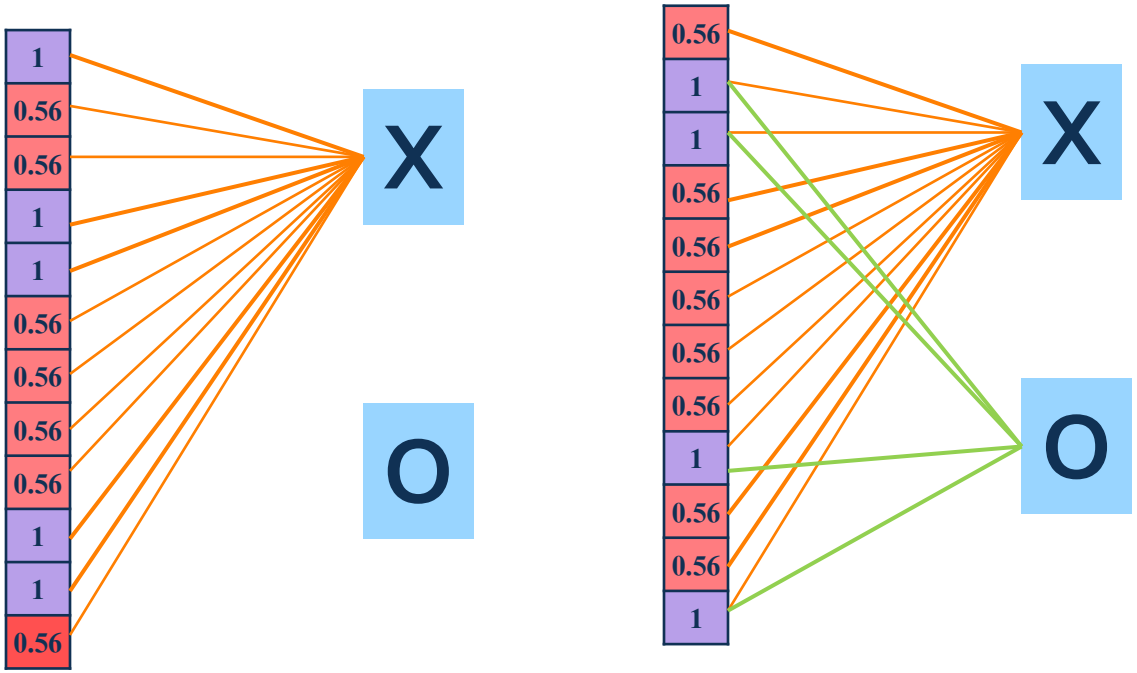
Fully connected layer

This is final layer where the actual classification happens. Every value gets a vote. Here we take our filtered and shirked images and put them into a single list



Fully connected layer (cont.)

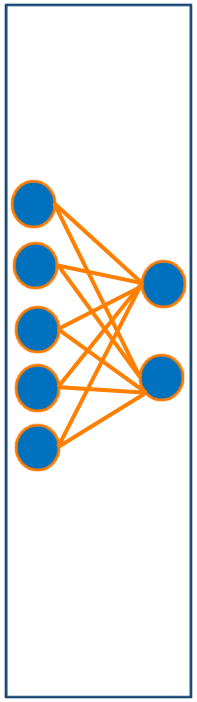
Vote depends on how strongly a value predicts X or O.



Fully connected layer (cont.)

A list of feature values becomes a list of votes.

0.78
0.56
0.56
0.78
1
0.33
0.33
0.33
0.56
0.78
0.78
0.56



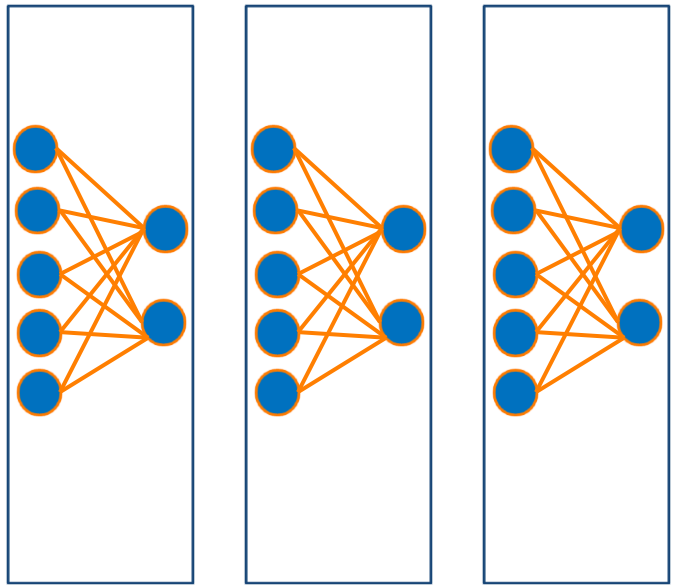
X

O

Fully connected layer (cont.)

These can also be stacked.

0.78
0.56
0.56
0.78
1
0.33
0.33
0.33
0.56
0.78
0.78
0.56

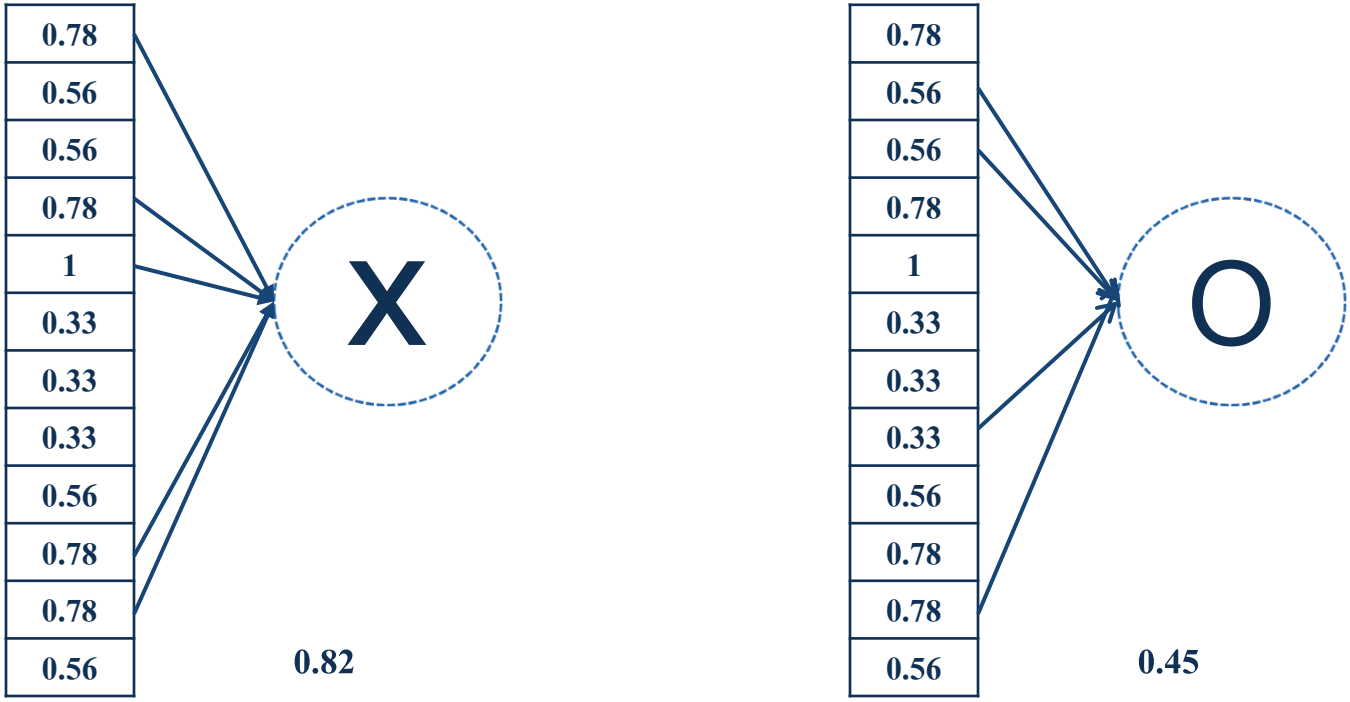


X

O

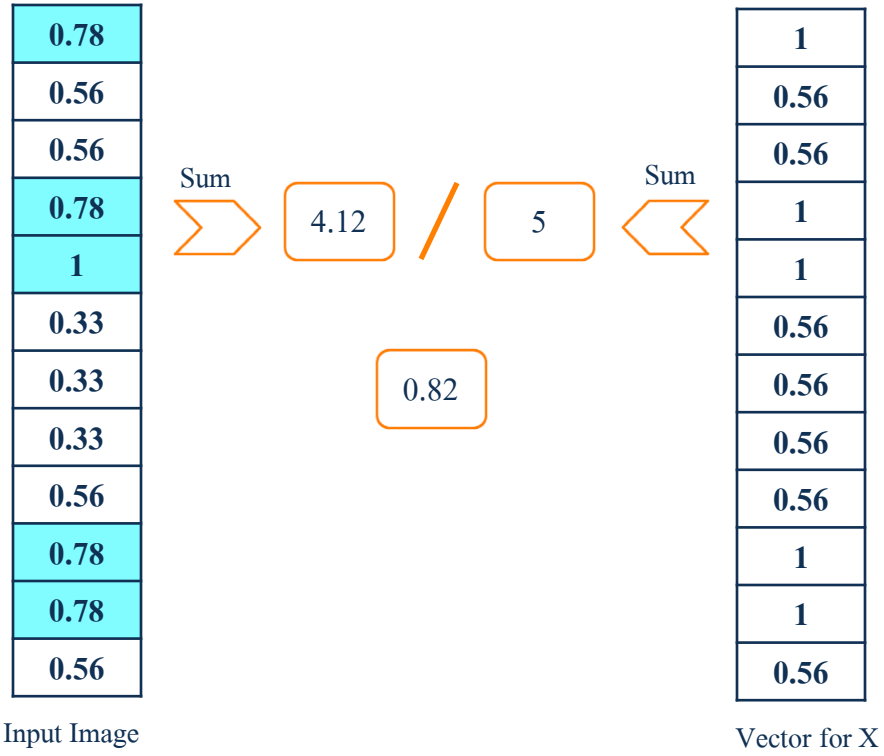
Result

When we feed in, 'X ' and 'O'. Then there will be some element in the vector that will be high. Consider the image below, as you can see for 'X' there are different elements that are high and similarly, for 'O' we have different elements that are high.



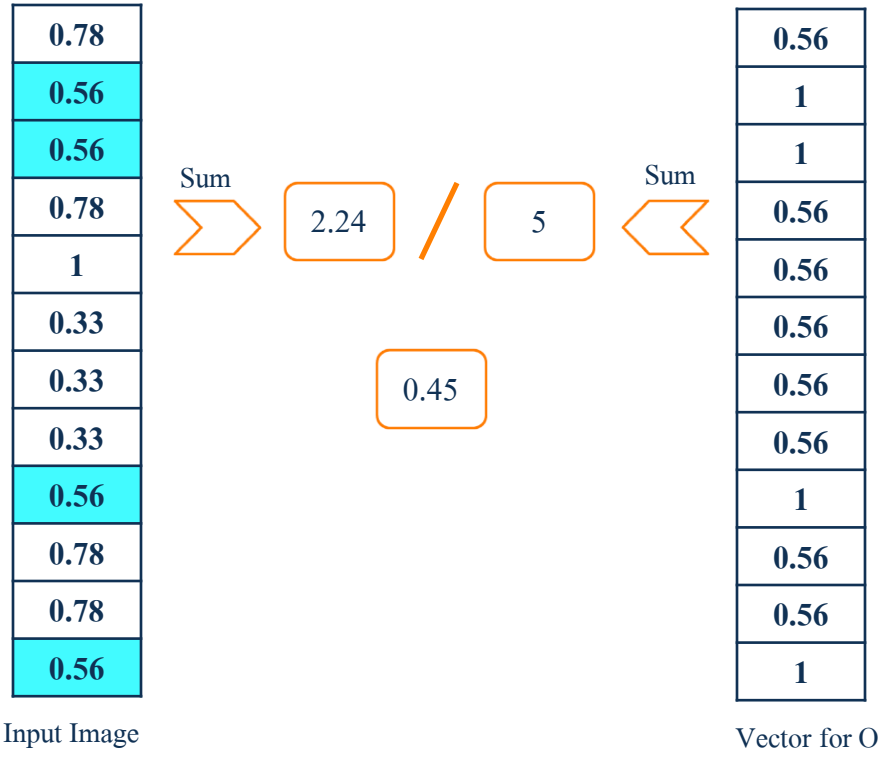
Prediction

Comparing the input Vector with X.

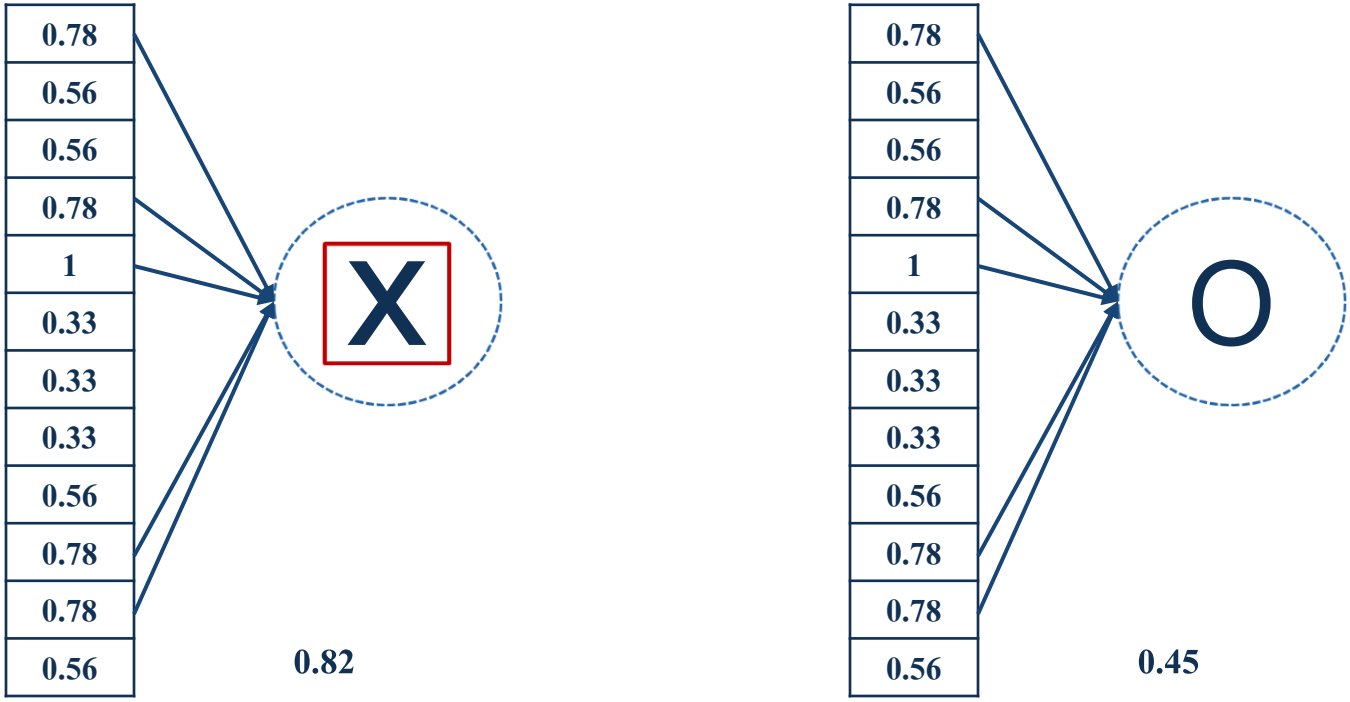


Prediction (cont.)

Comparing the input Vector with O.

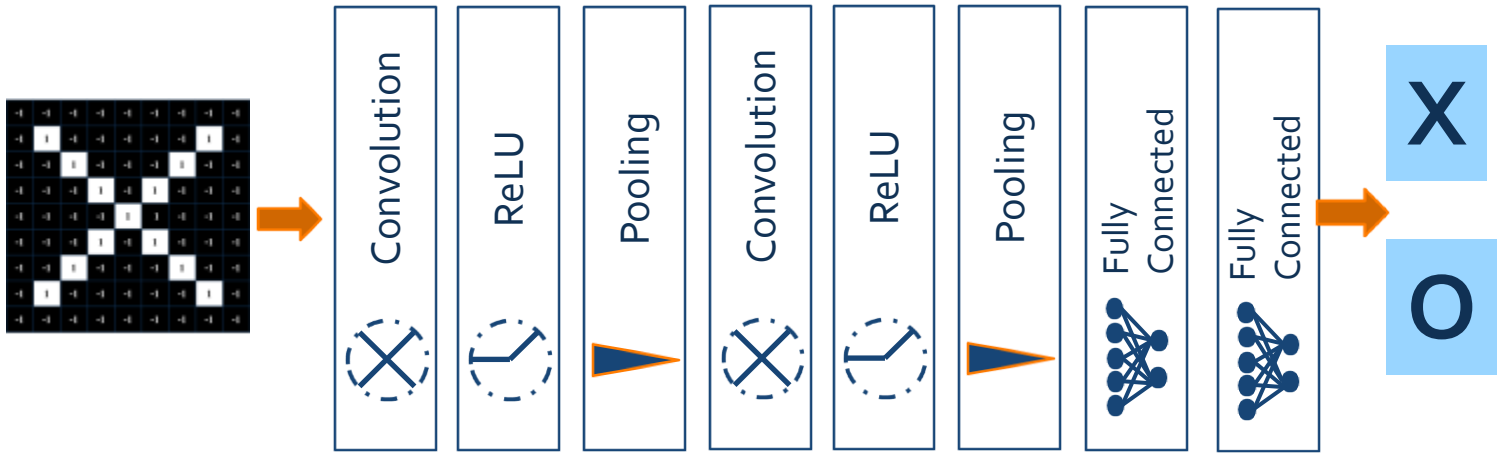


Result

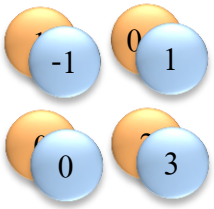

0.78
0.56
0.56
0.78
1
0.33
0.33
0.33
0.56
0.78
0.78
0.56
X
0.82
0.78
0.56
0.56
0.78
1
0.33
0.33
0.33
0.56
0.78
0.78
0.56
O
0.45

Putting it all together.

A set of pixels becomes a set of votes.



The whole CNN



Convolution

Max pooling

Convolution

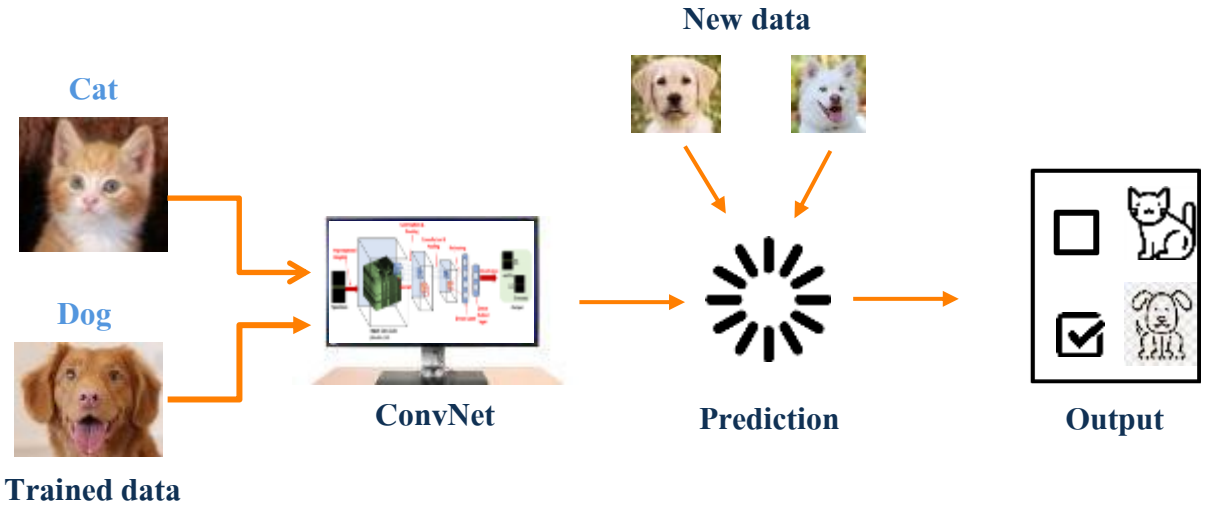
Max pooling

Can repeat
Many times

A new image

Smaller than the original image. The number of the channel is the number of filters.

Dog and cat classification



Classification hand written digits with Tensorflow's Keras API

- Load Data.
- Define Keras Model.
- Compile Keras Model.
- Fit Keras Model.
- Evaluate Keras Model.
- Tie It All Together.
- Make Predictions

Classification hand written digits with Tensorflow's Keras API

```
#import tensorflow and MNIST dataset
import tensorflow as tf
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()#groups as train and test
import matplotlib.pyplot as plt
image_index = 7777 # You may select anything up to 60,000
print(y_train[image_index]) # The label is 8
plt.imshow(x_train[image_index], cmap='Greys')
#shape of the dataset
x_train.shape #number of images in the train dataset with the size of the image: 28 x 28 pixel.
#Reshaping and Normalizing the Images
# Reshaping the array to 4-dims so that it can work with the Keras API
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
# Making sure that the values are float so that we can get decimal points after division
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# Normalizing the RGB codes by dividing it to the max RGB value.
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print('Number of images in x_train', x_train.shape[0])
print('Number of images in x_test', x_test.shape[0])
```

Classification hand written digits with Tensorflow's Keras API

```
#Building the Convolutional Neural Network
# Importing the required Keras modules containing model and layers
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
# Creating a Sequential Model and adding the layers
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten()) # Flattening the 2D arrays for fully connected layers
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10,activation=tf.nn.softmax))
#Compiling and Fitting the Model
model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
model.fit(x=x_train,y=y_train, epochs=10)
#Evaluating the Model
model.evaluate(x_test, y_test)
# test the model
image_index = 4444
plt.imshow(x_test[image_index].reshape(28, 28),cmap='Greys')
pred = model.predict(x_test[image_index].reshape(1, 28, 28, 1))
print(pred.argmax())
```

Thank You.