Plan of Attack

Plan of Attack

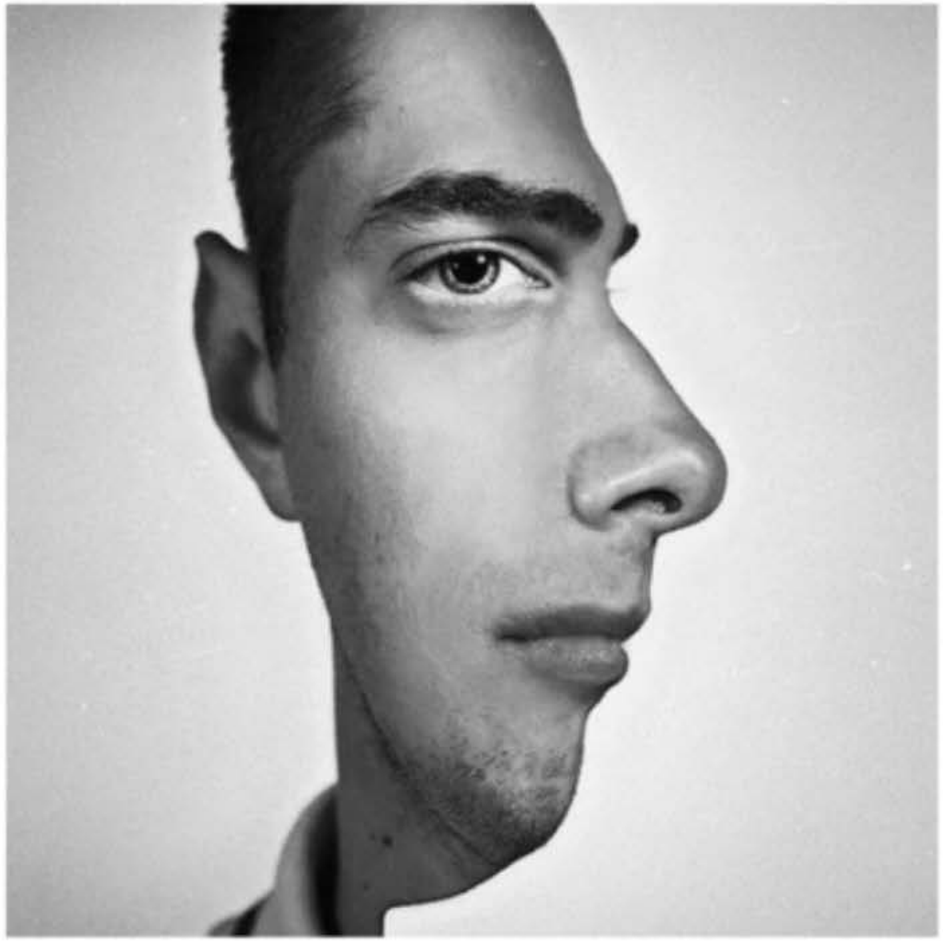
What we will learn in this section:

- What are Convolutional Neural Networks?
- Step 1 - Convolution Operation
- Step 1(b) - ReLU Layer
- Step 2 - Pooling
- Step 3 - Flattening
- Step 4 - Full Connection
- Summary

- EXTRA: Softmax & Cross-Entropy

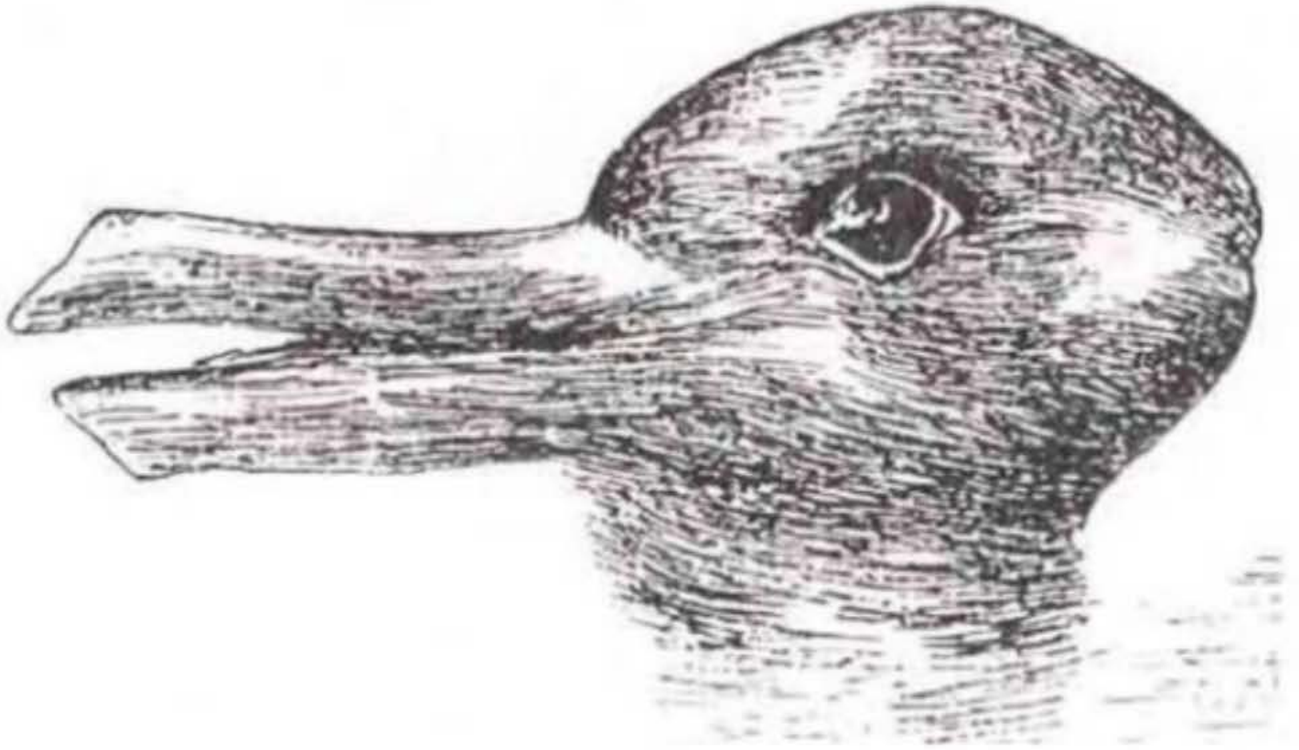
Convolutional Neural Networks

Convolutional Neural Networks

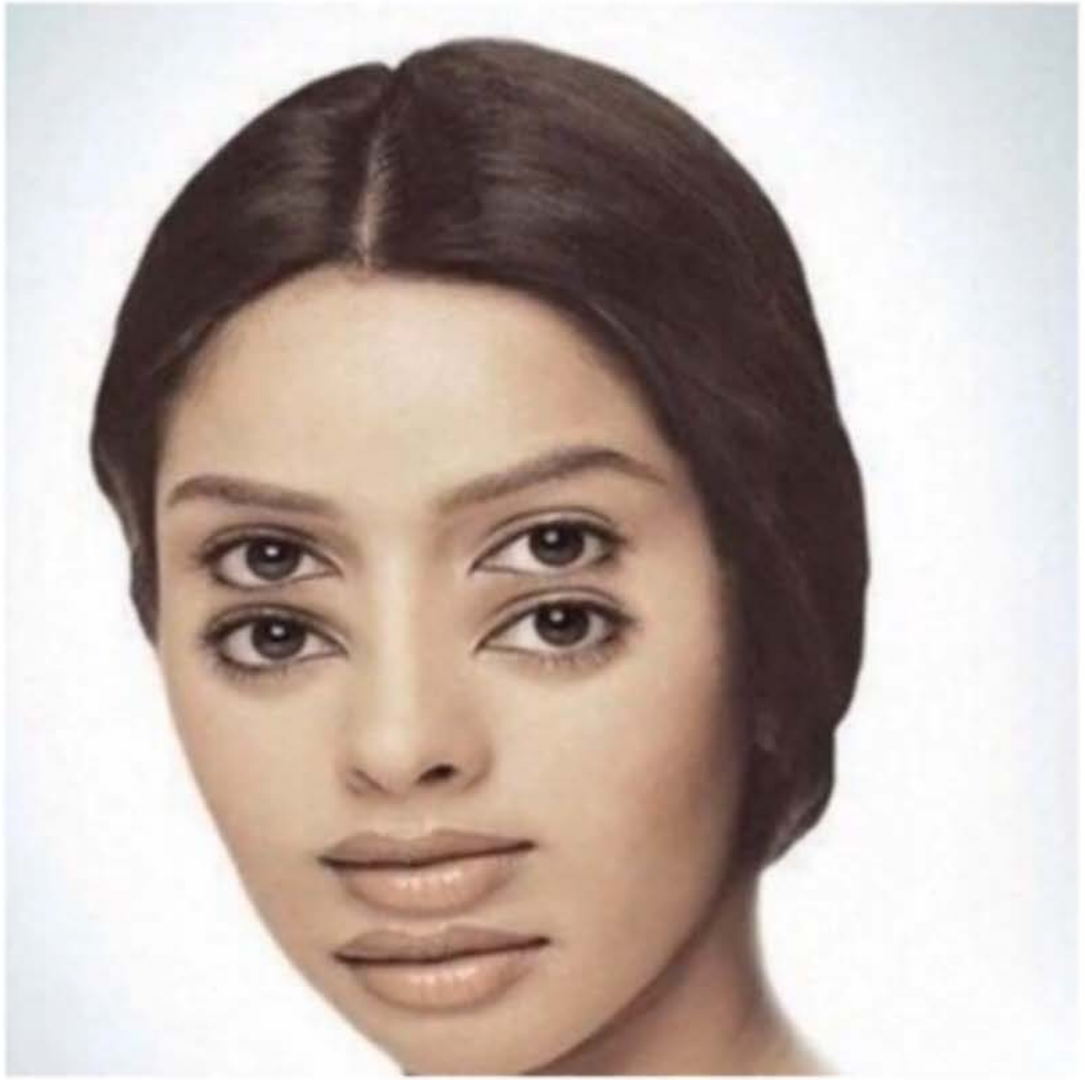


Convolutional Neural Networks

Convolutional Neural Networks



Convolutional Neural Networks



Convolutional Neural Networks

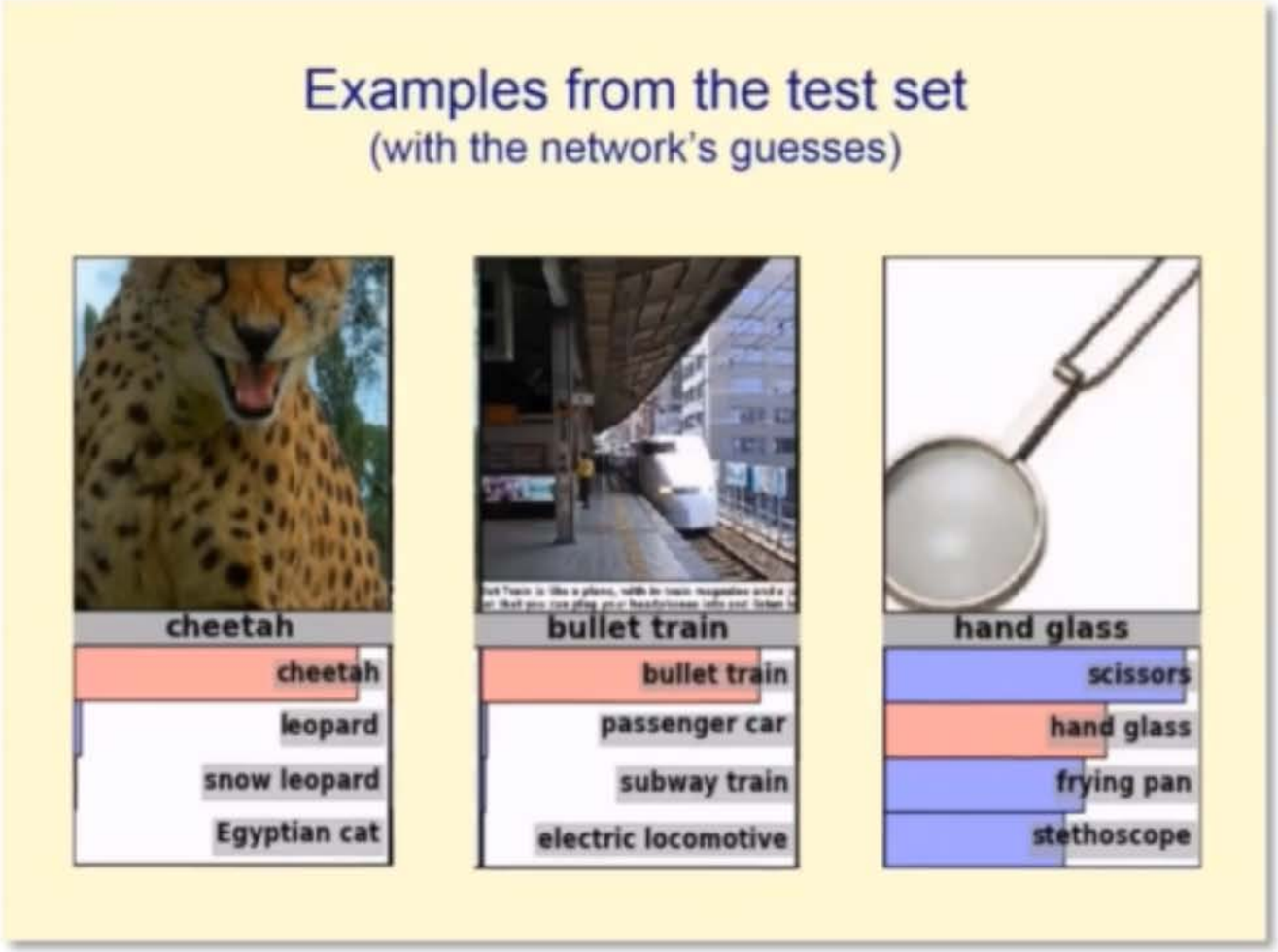
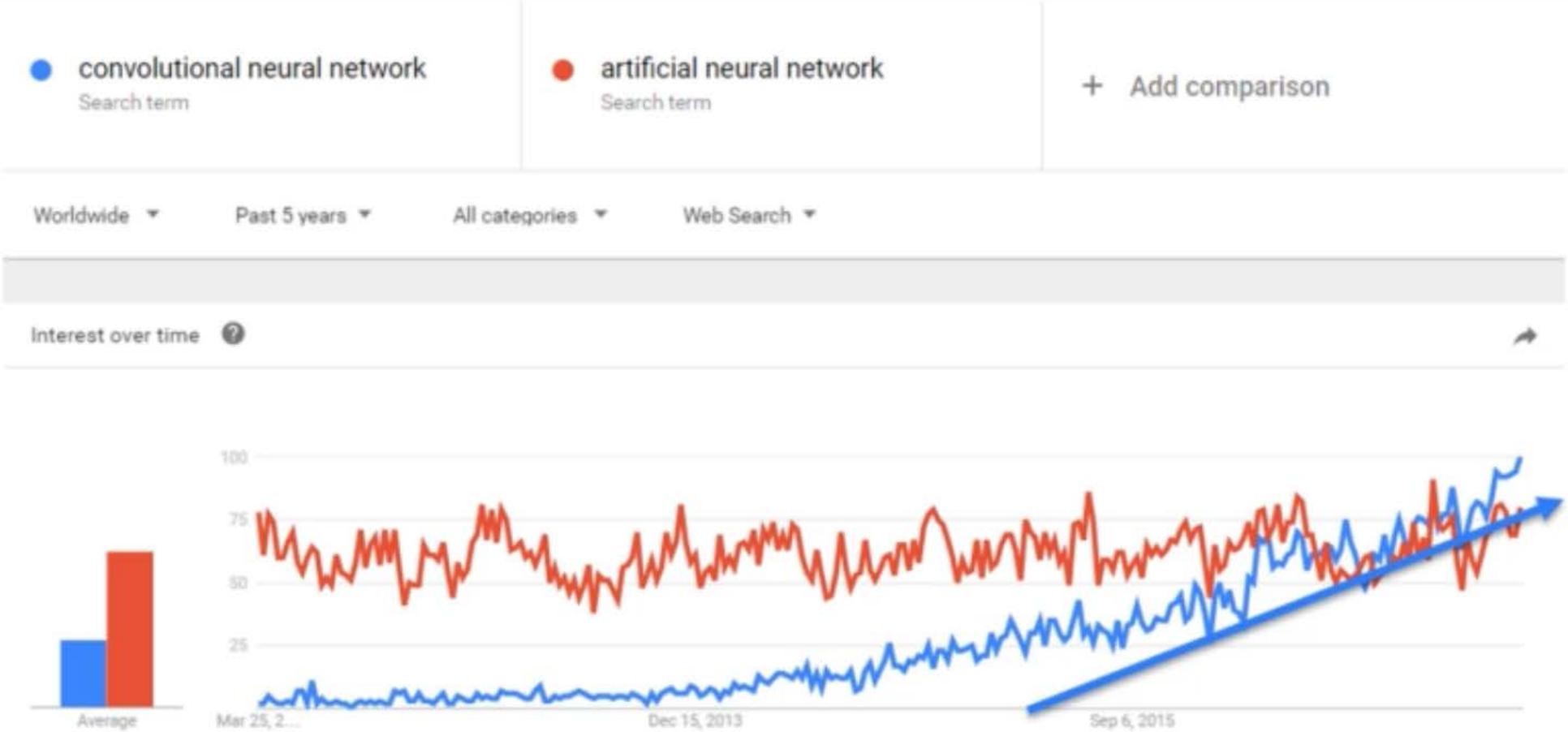


Image Source: a talk by Geoffrey Hinton

Convolutional Neural Networks



Source: google trends

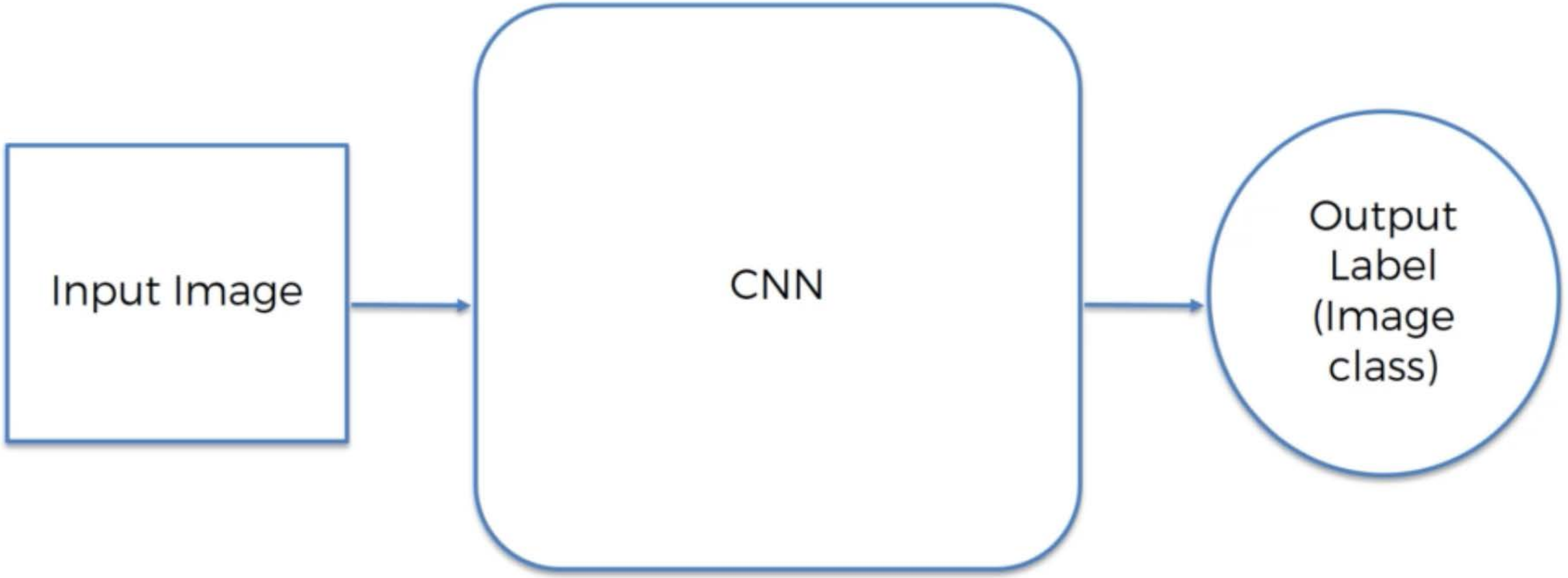
Convolutional Neural Networks

Yann Lecun

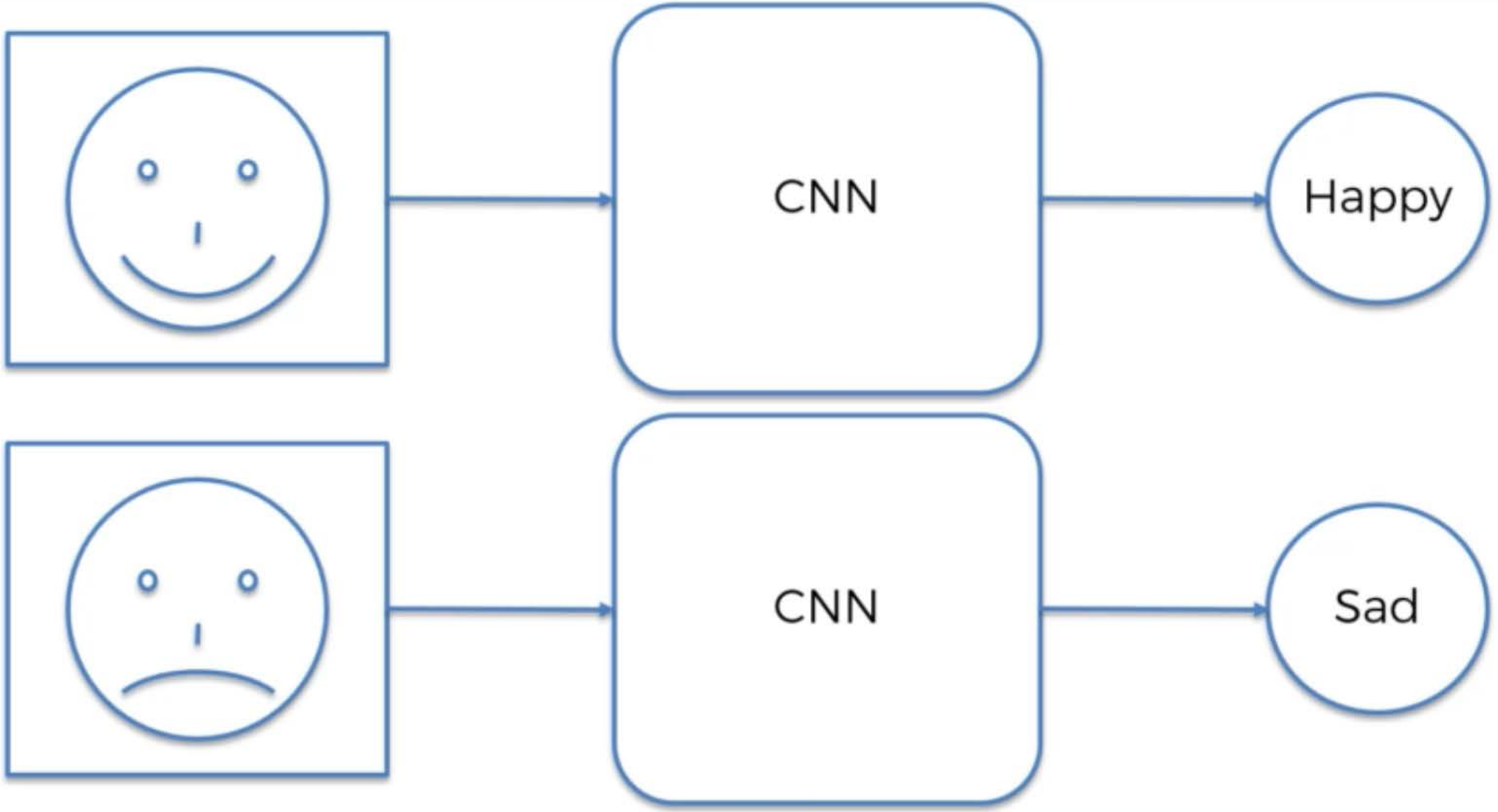
Convolutional Neural Networks

Google Facebook

Convolutional Neural Networks

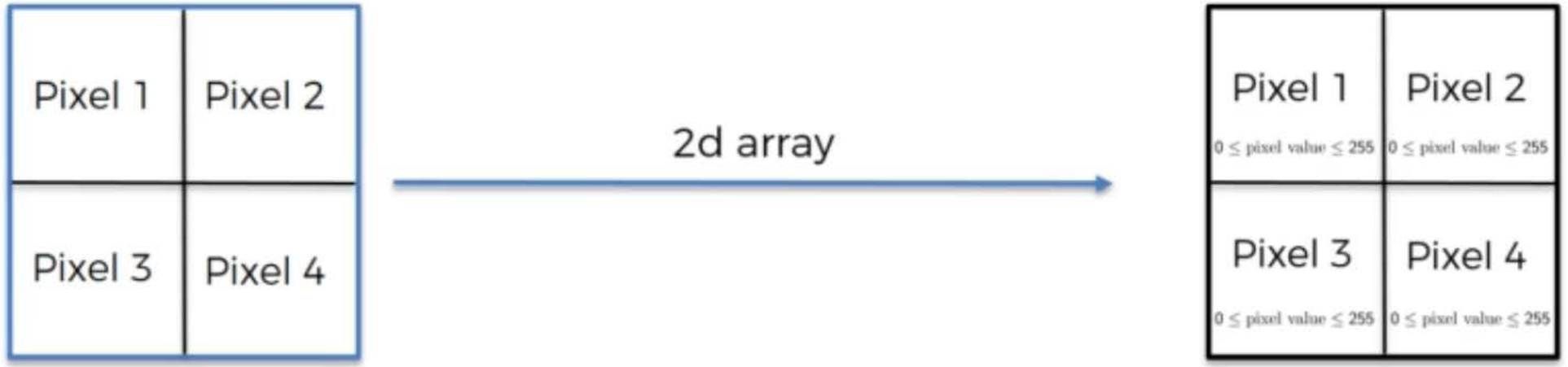


Convolutional Neural Networks

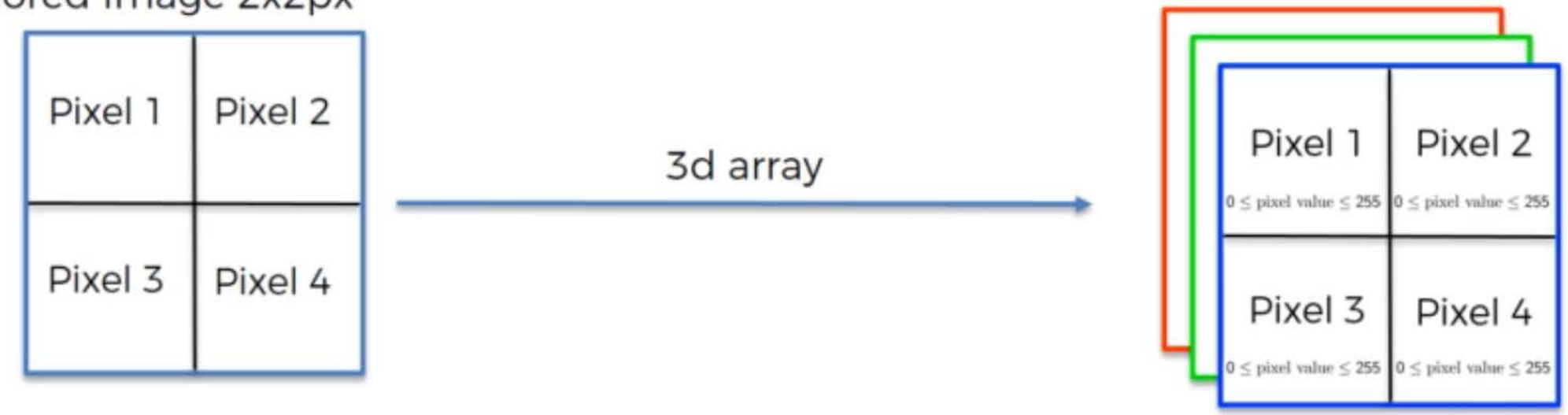


Convolutional Neural Networks

B / W Image 2x2px



Colored Image 2x2px



Convolutional Neural Networks

B / W Image 2x2px

Pixel 1	Pixel 2
Pixel 3	Pixel 4

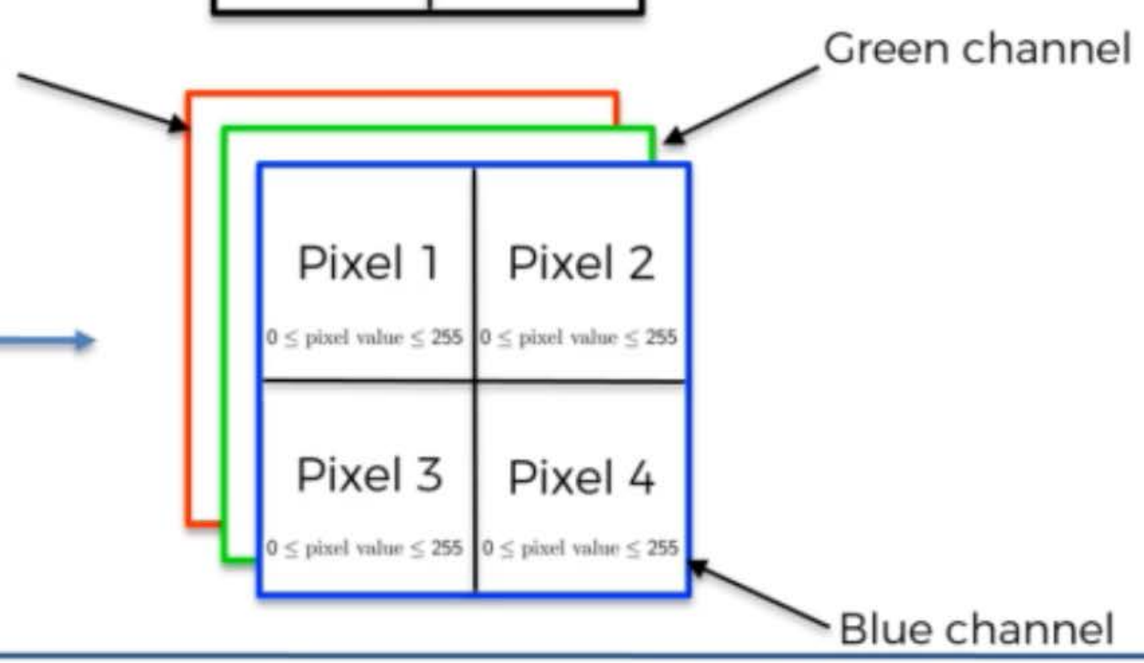
2d array

Pixel 1 $0 \leq$ pixel value ≤ 255	Pixel 2 $0 \leq$ pixel value ≤ 255
Pixel 3 $0 \leq$ pixel value ≤ 255	Pixel 4 $0 \leq$ pixel value ≤ 255

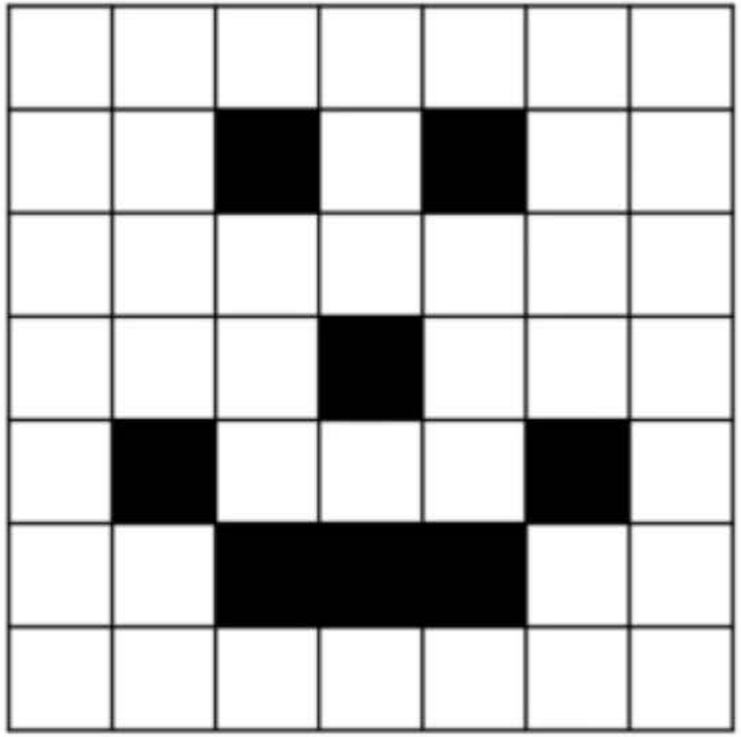
Colored Image 2x2px

Pixel 1	Pixel 2
Pixel 3	Pixel 4

3d array



Convolutional Neural Networks



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

Convolutional Neural Networks

STEP 1: Convolution

STEP 2: Max Pooling

STEP 3: Flattening

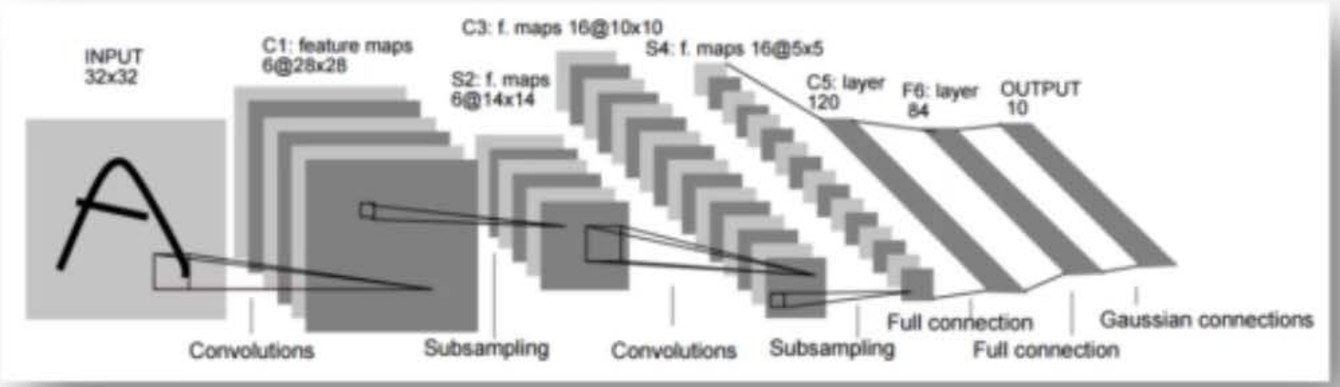
STEP 4: Full Connection

Convolutional Neural Networks

Additional Reading:

Gradient-Based Learning Applied to Document Recognition

By Yann LeCun et al. (1998)



Link:

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Step 1 - Convolution

Step 1 - Convolution

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t - \tau)\, d\tau$$

Step 1 - Convolution

Additional Reading:

Introduction to Convolutional Neural Networks

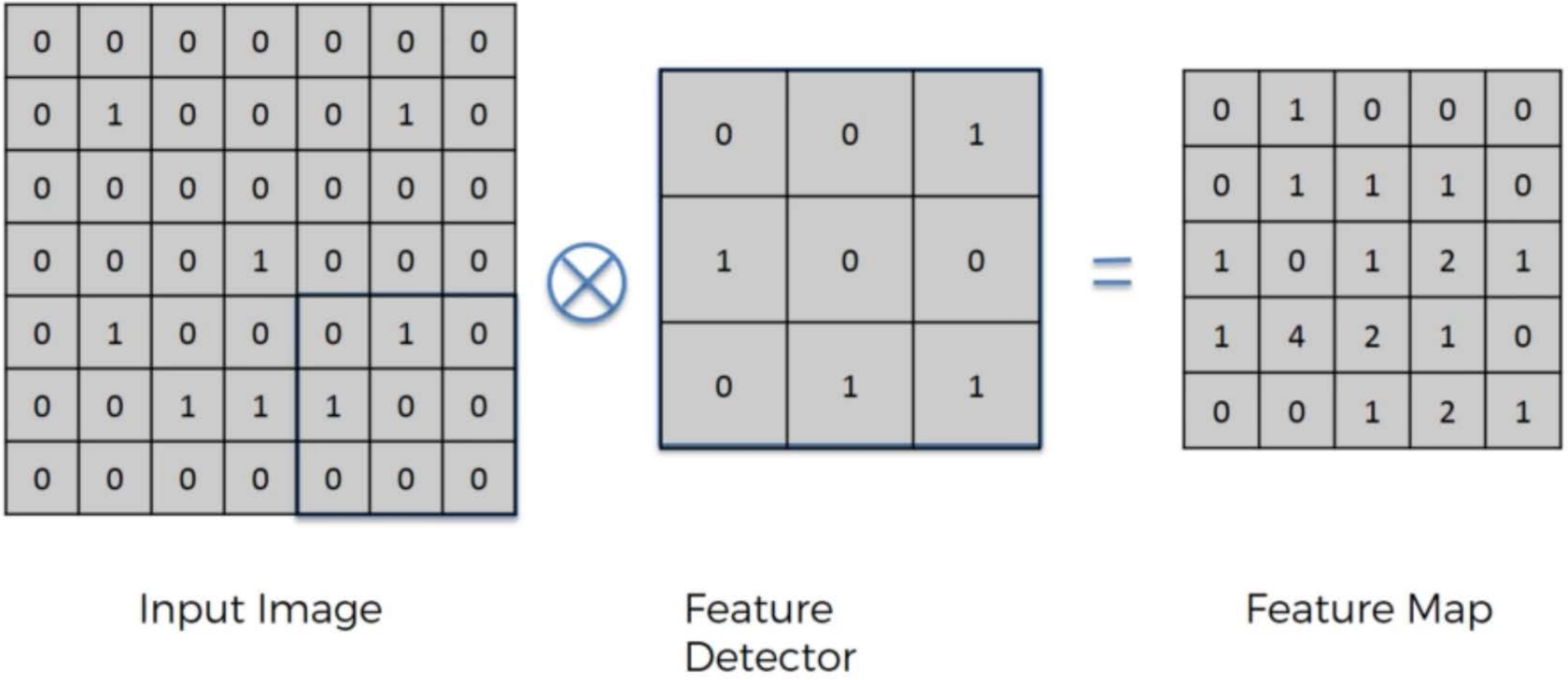
By Jianxin Wu (2017)

$$\begin{aligned}\frac{\partial z}{\partial(\mathrm{vec}(\boldsymbol{y})^T)}(F^T \otimes I) &= \left((F \otimes I)\frac{\partial z}{\partial \mathrm{vec}(\boldsymbol{y})}\right)^T \\ &= \left((F \otimes I)\,\mathrm{vec}\left(\frac{\partial z}{\partial Y}\right)\right)^T \\ &= \mathrm{vec}\left(I\frac{\partial z}{\partial Y}F^T\right)^T \\ &= \mathrm{vec}\left(\frac{\partial z}{\partial Y}F^T\right)^T,\end{aligned}$$

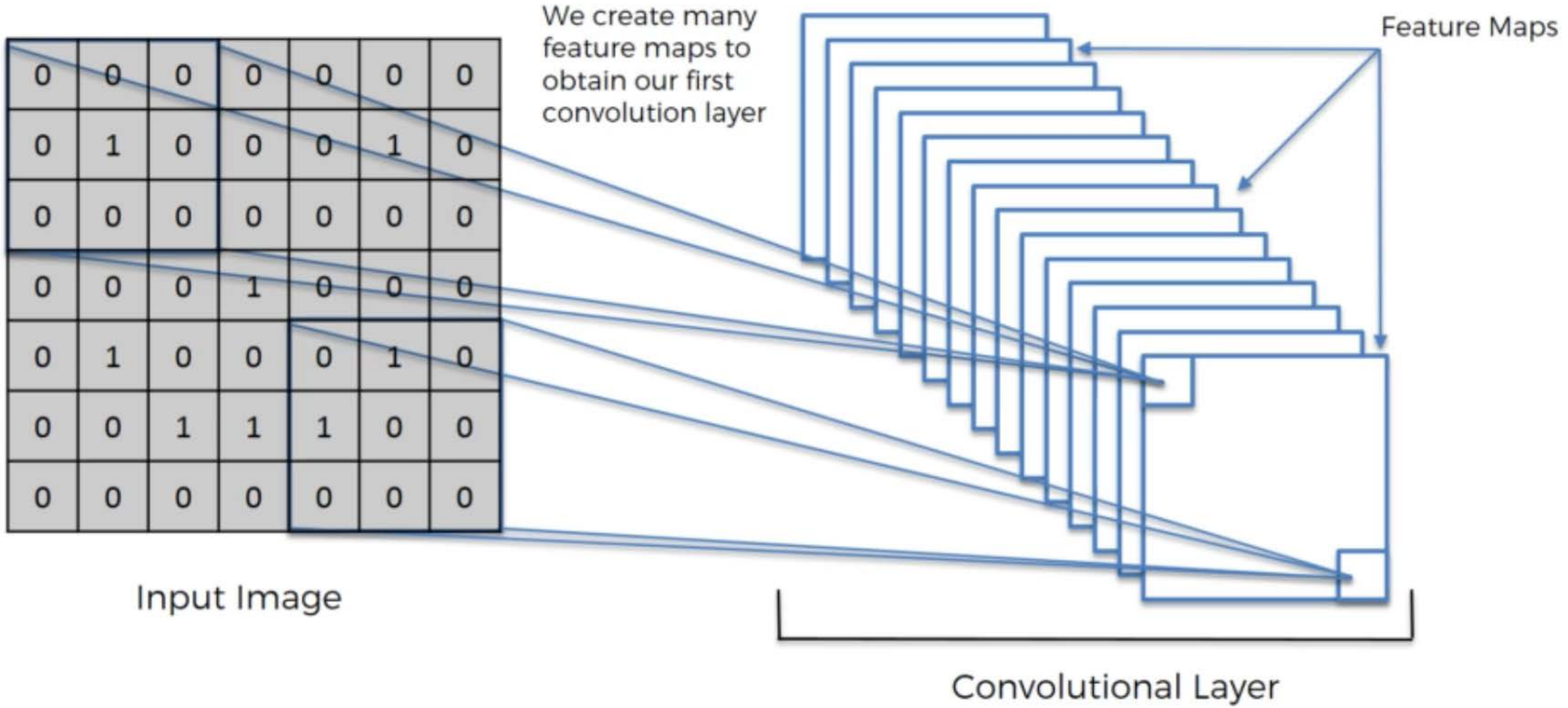
Link:

http://cs.nju.edu.cn/wujx/paper/CNN.pdf

Step 1 - Convolution



Step 1 - Convolution



Step 1 - Convolution

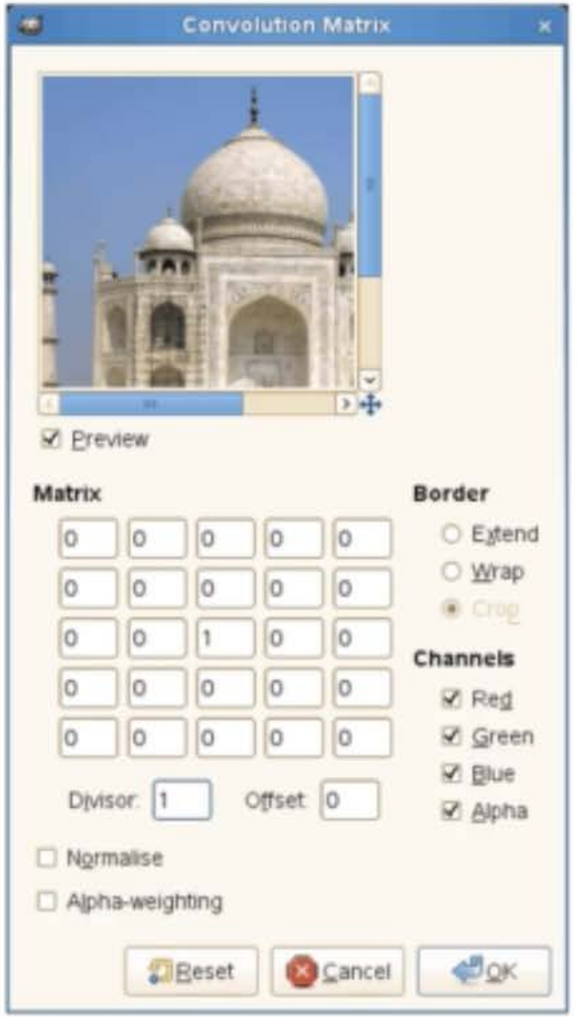


Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Sharpen:

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Blur:

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

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Edge Enhance:

	0	0	0	
	-1	1	0	
	0	0	0	

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Edge Detect:

	0	1	0	
	1	-4	1	
	0	1	0	

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Emboss:

	-2	-1	0	
	-1	1	1	
	0	1	2	

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

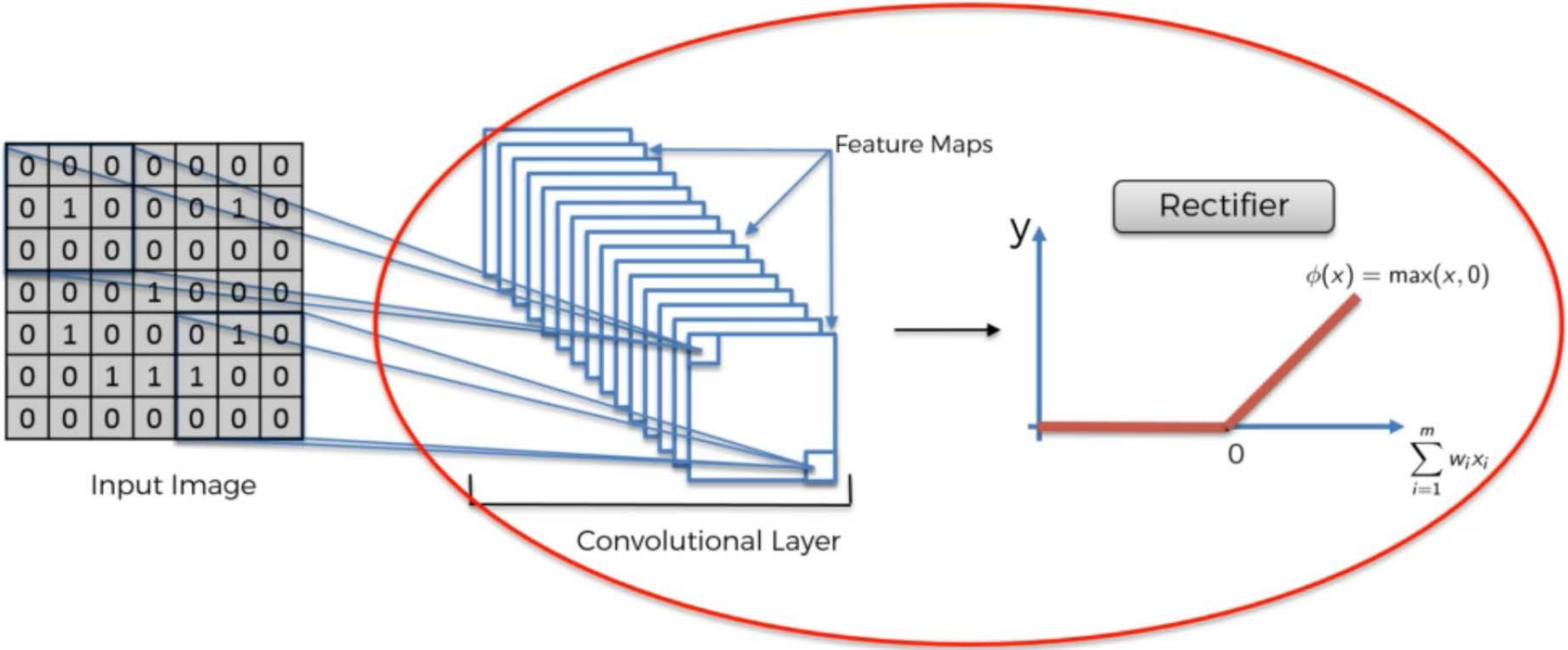
$*$

1	0	-1
2	0	-2
1	0	-1

Image Source: eonardoaraujosantos.gitbooks.io

Step 1 (B) – ReLU Layer

Step 1(B) - ReLU Layer



Step 1 (B) – ReLU Layer

Image Source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Step 1(B) - ReLU Layer

Image Source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Step 1(B) - ReLU Layer

Image Source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Step 1(B) – ReLU Layer

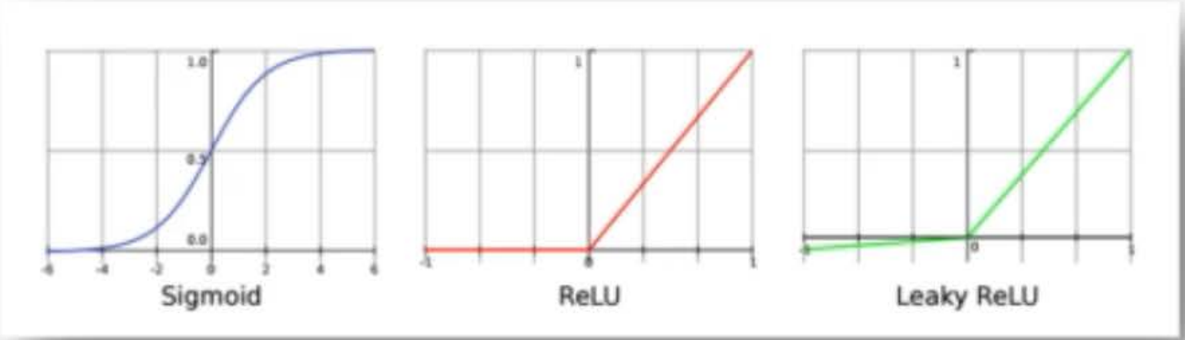
Additional Reading:

Understanding Convolutional Neural Networks with A Mathematical Model

By C.-C. Jay Kuo (2016)

Link:

https://arxiv.org/pdf/1609.04112.pdf



Step 1(B) – ReLU Layer

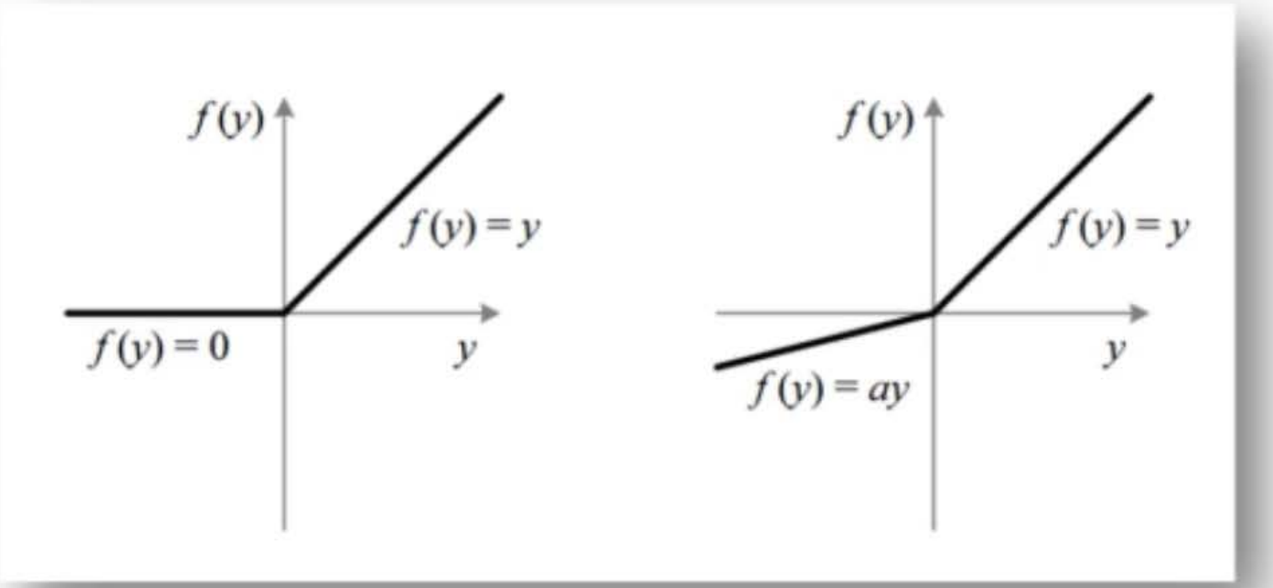
Additional Reading:

Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

By Kaiming He et al. (2015)

Link:

https://arxiv.org/pdf/1502.01852.pdf



Step 2 – Max Pooling

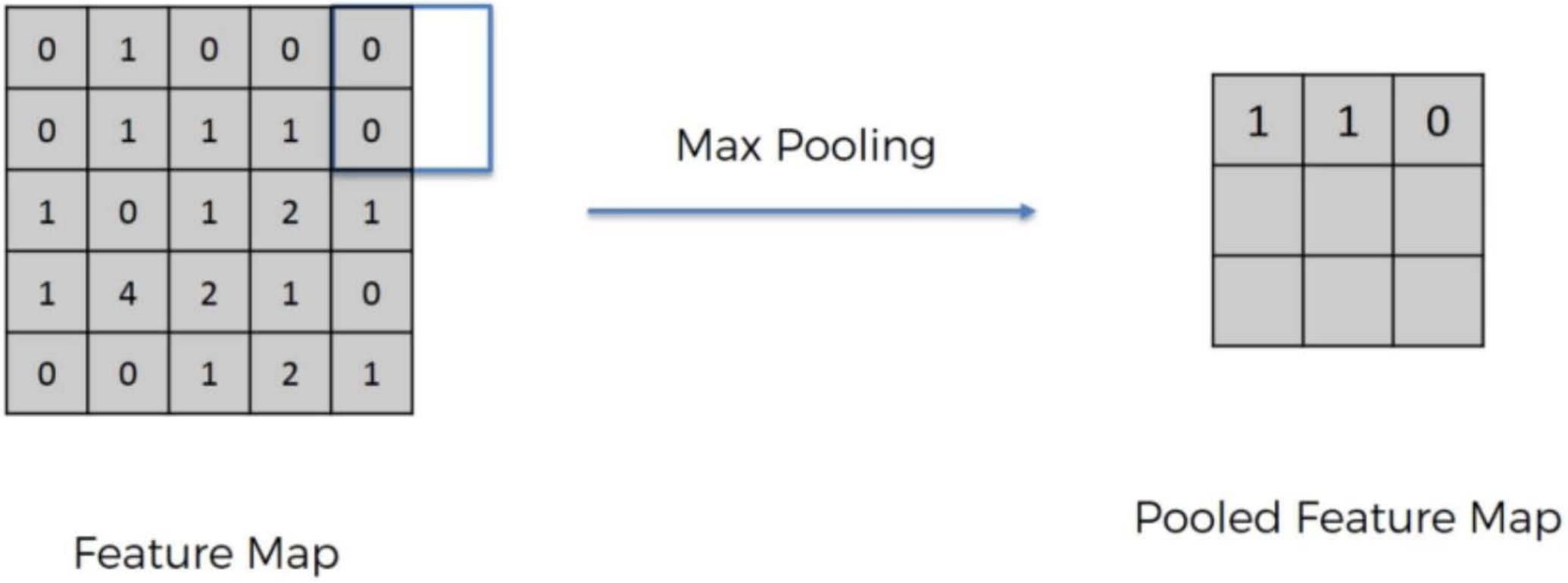
Step 2 - Max Pooling

Image Source: Wikipedia

Step 2 - Max Pooling

Image Source: Wikipedia

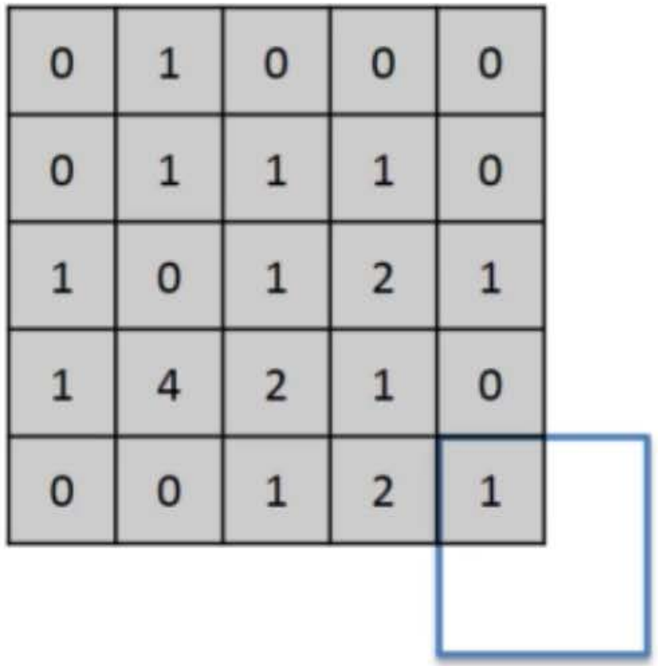
Step 2 - Max Pooling



Feature Map

Pooled Feature Map

Step 2 - Max Pooling



Feature Map

Max Pooling

Pooled Feature Map

Step 2 - Max Pooling

Additional Reading:

Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition

By Dominik Scherer et al. (2010)

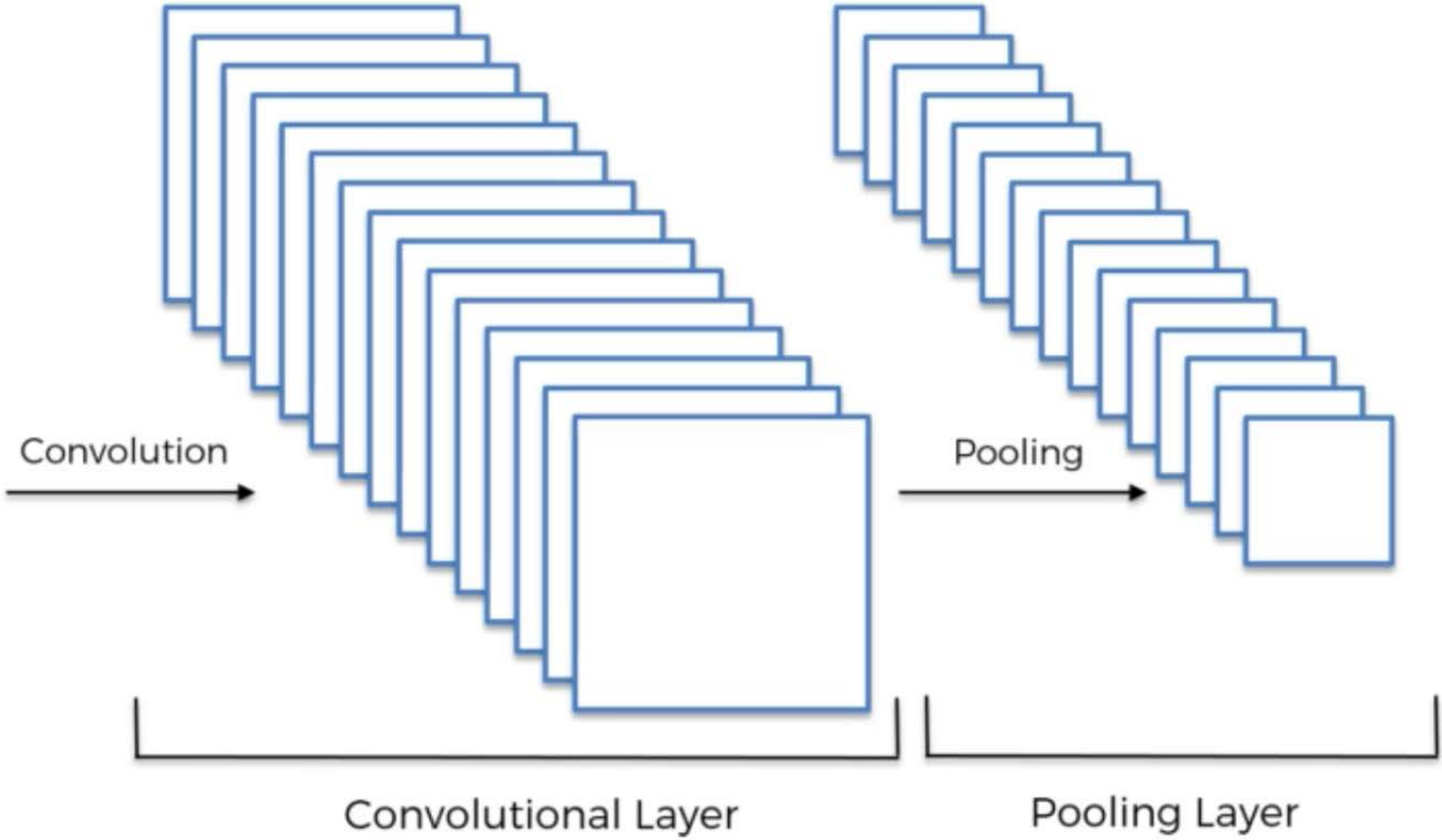
Link:

http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf

Step 2 - Max Pooling

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

Input Image



Example

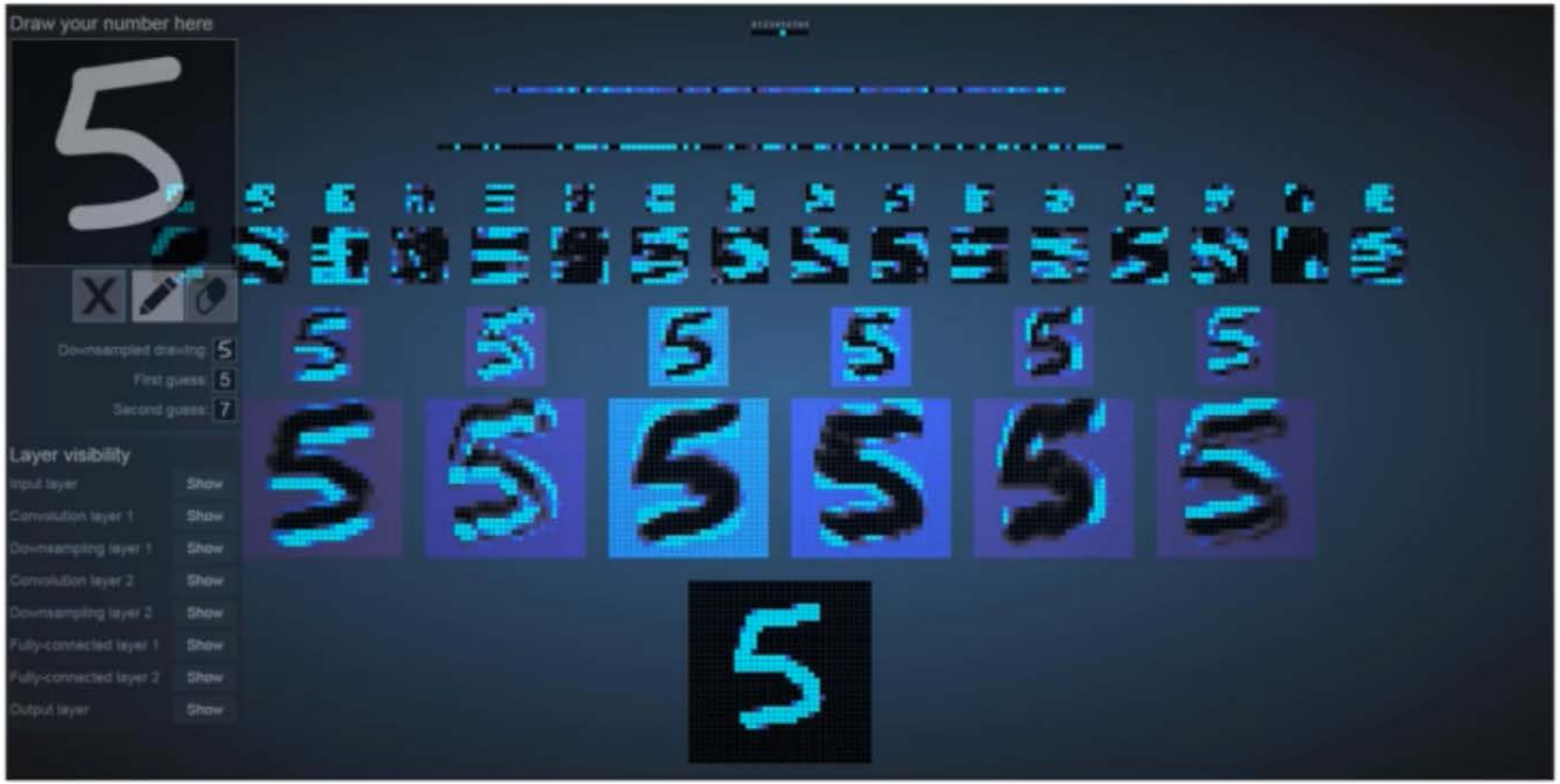
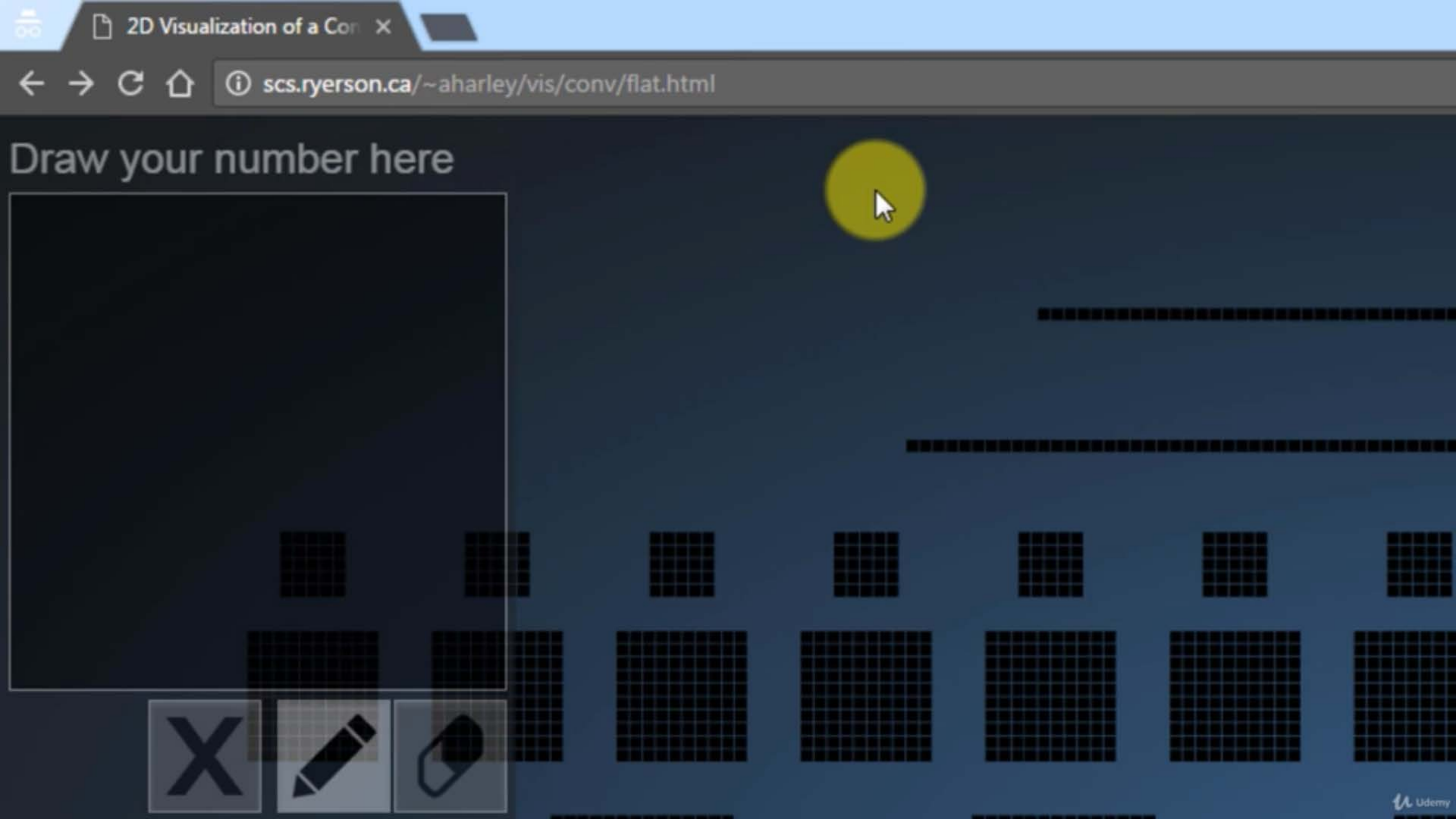


Image Source: scs.ryerson.ca/~aharley/vis/conv/flat.html


2D Visualization of a Con
scs.ryerson.ca/~aharley/vis/conv/flat.html
Draw your number here

Step 3 – Flattening

Step 3 - Flattening

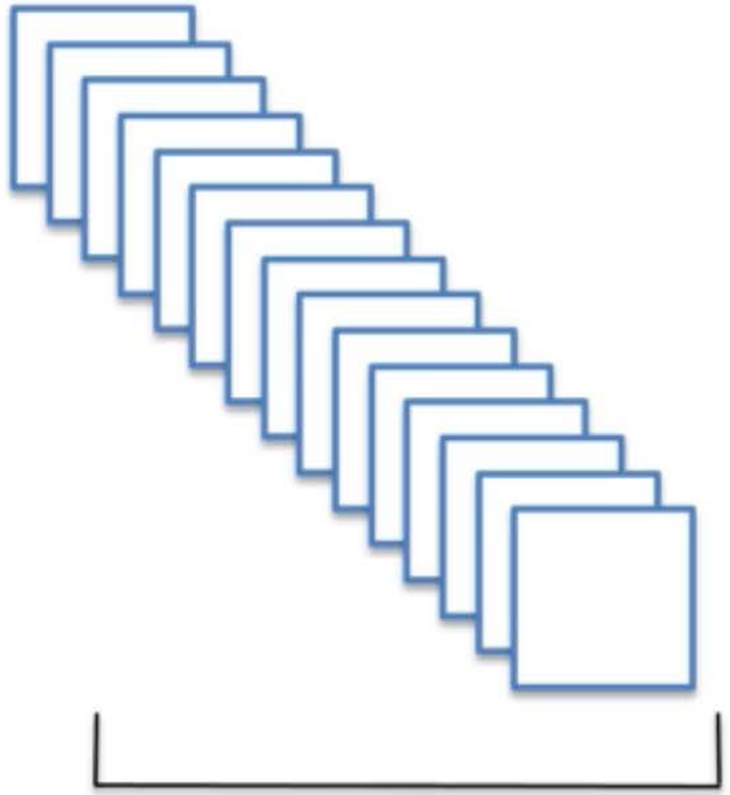
1	1	0
4	2	1
0	2	1

Pooled Feature Map

Flattening

1
1
0
4
2
1
0
2
1

Step 3 - Flattening

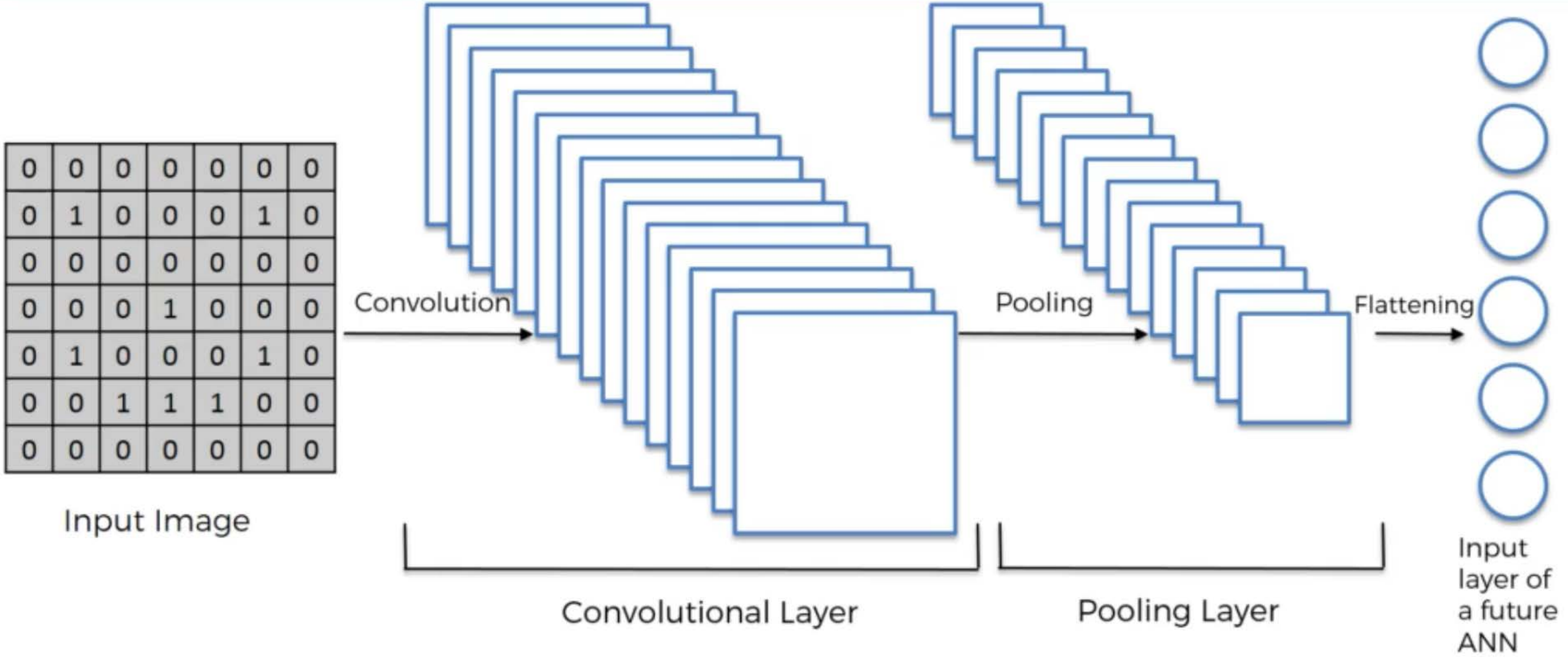


Pooling Layer

Flattening

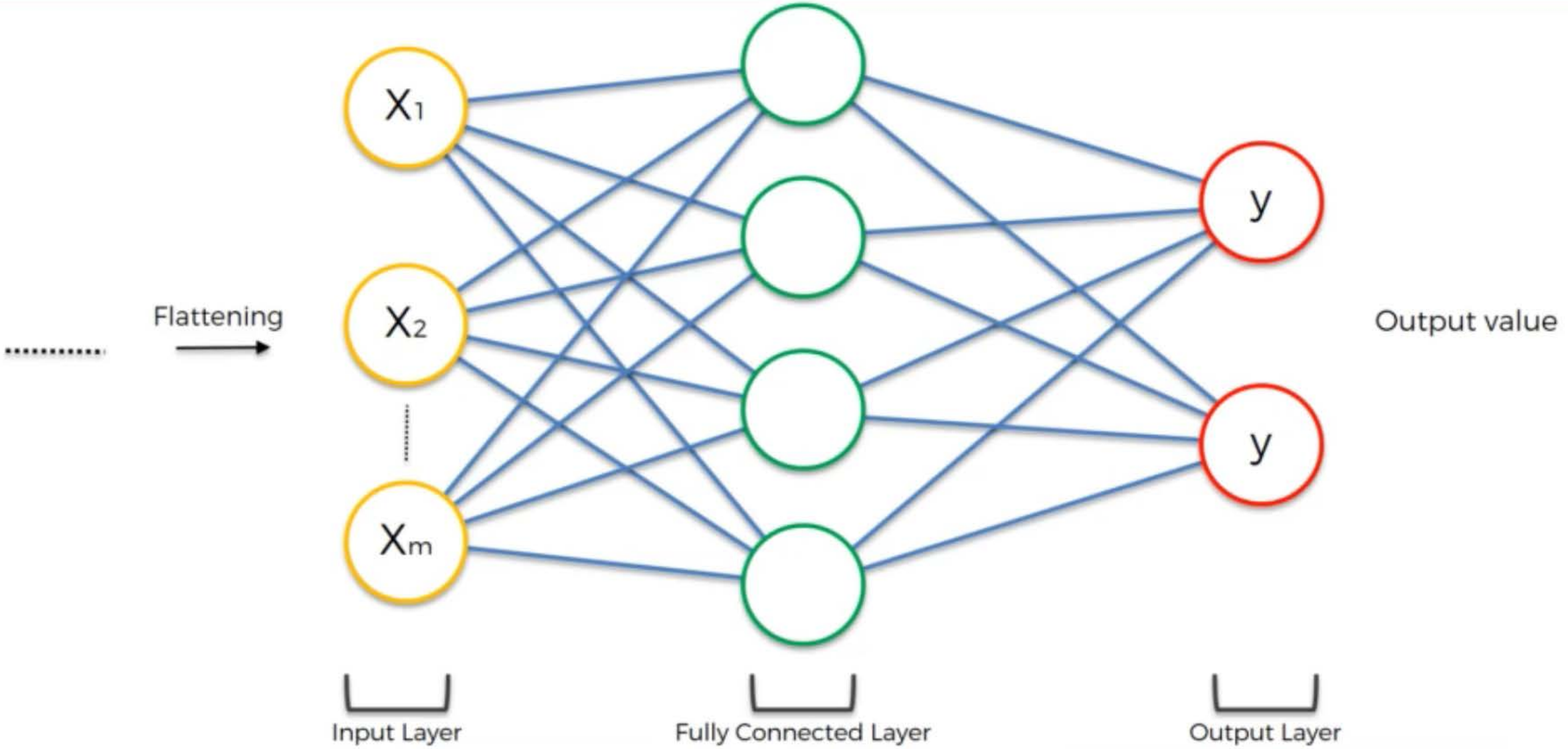
Input layer of a future ANN

Step 3 - Flattening

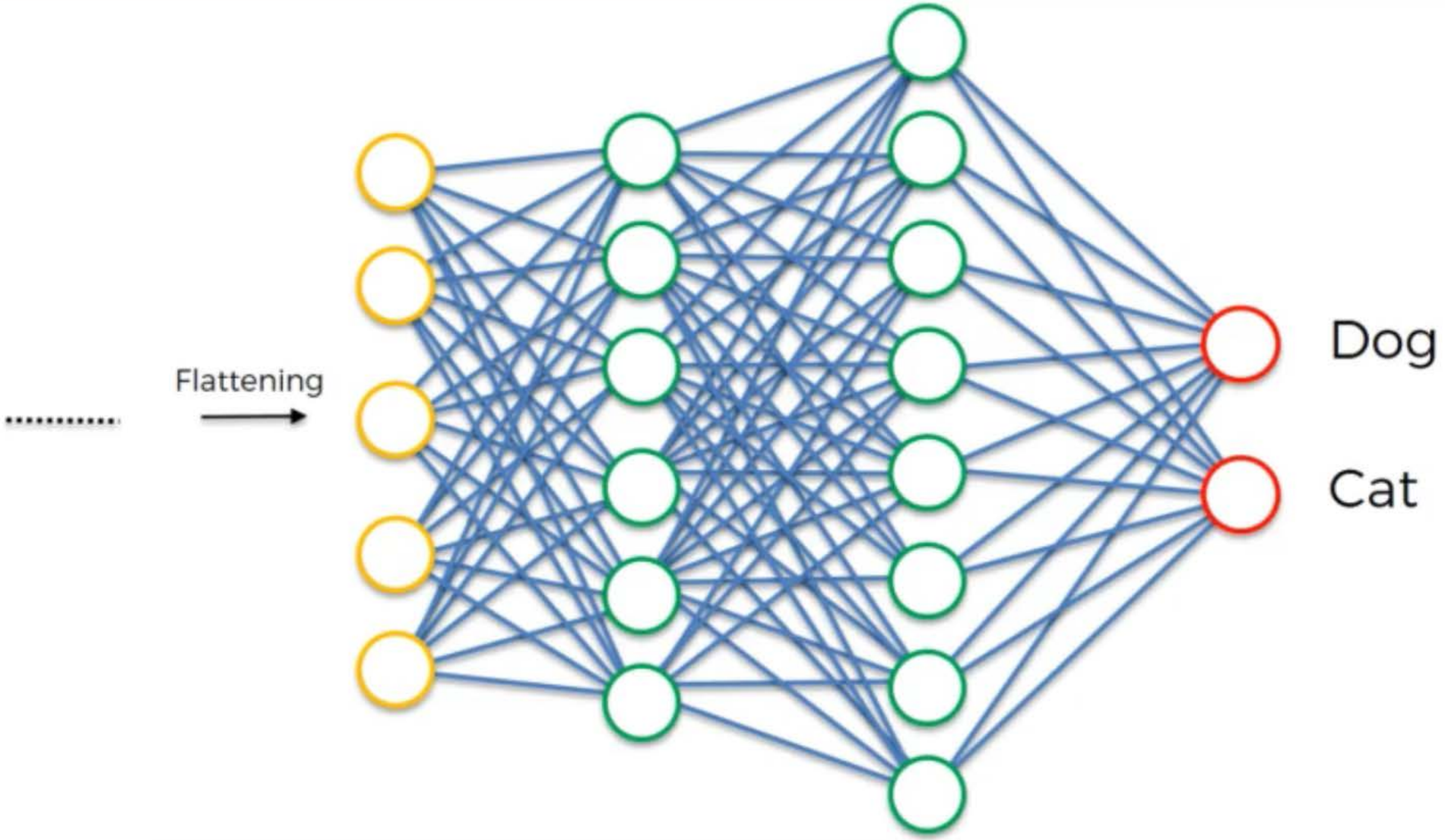


Step 4 – Full Connection

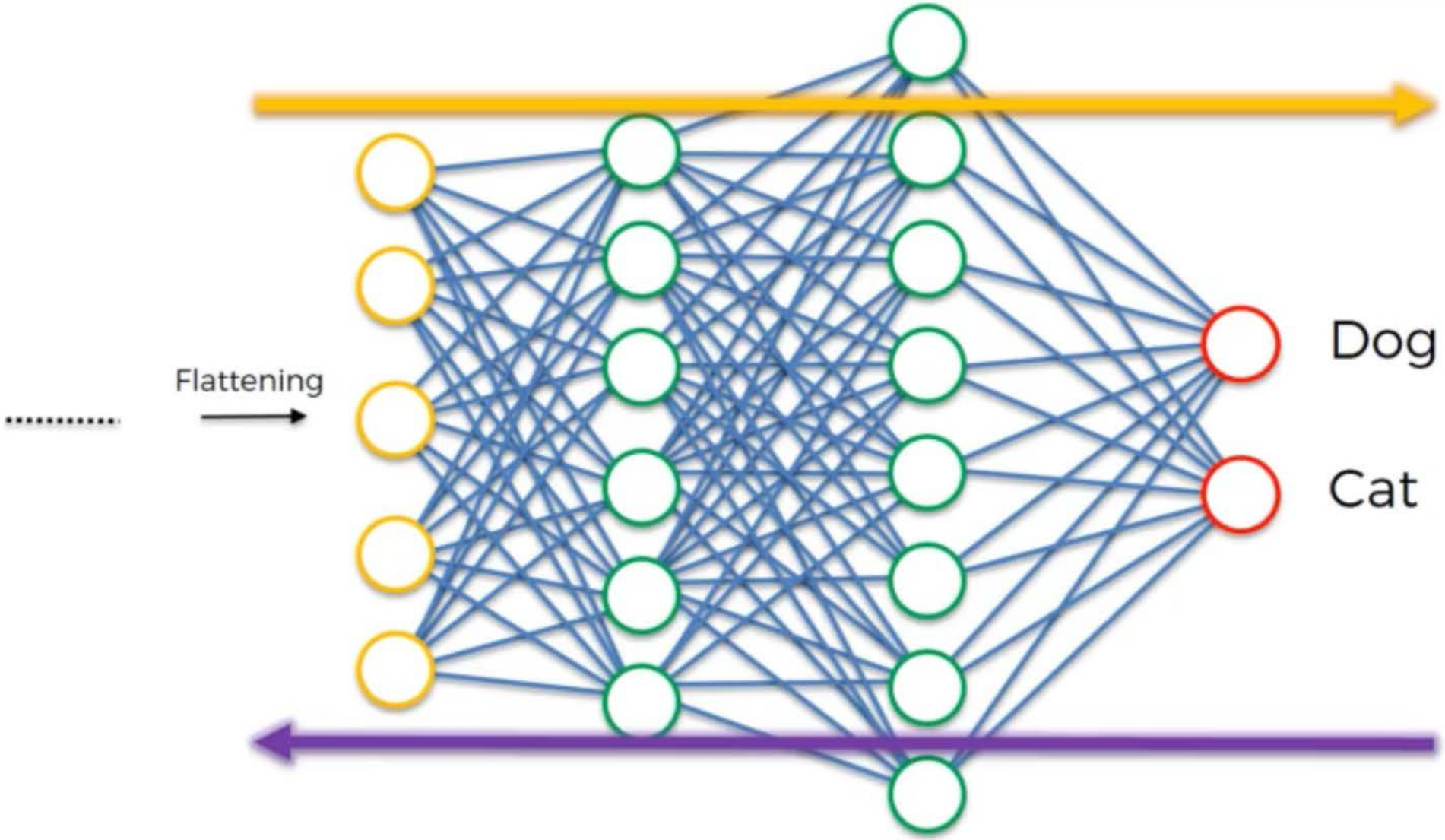
Step 4 - Full Connection



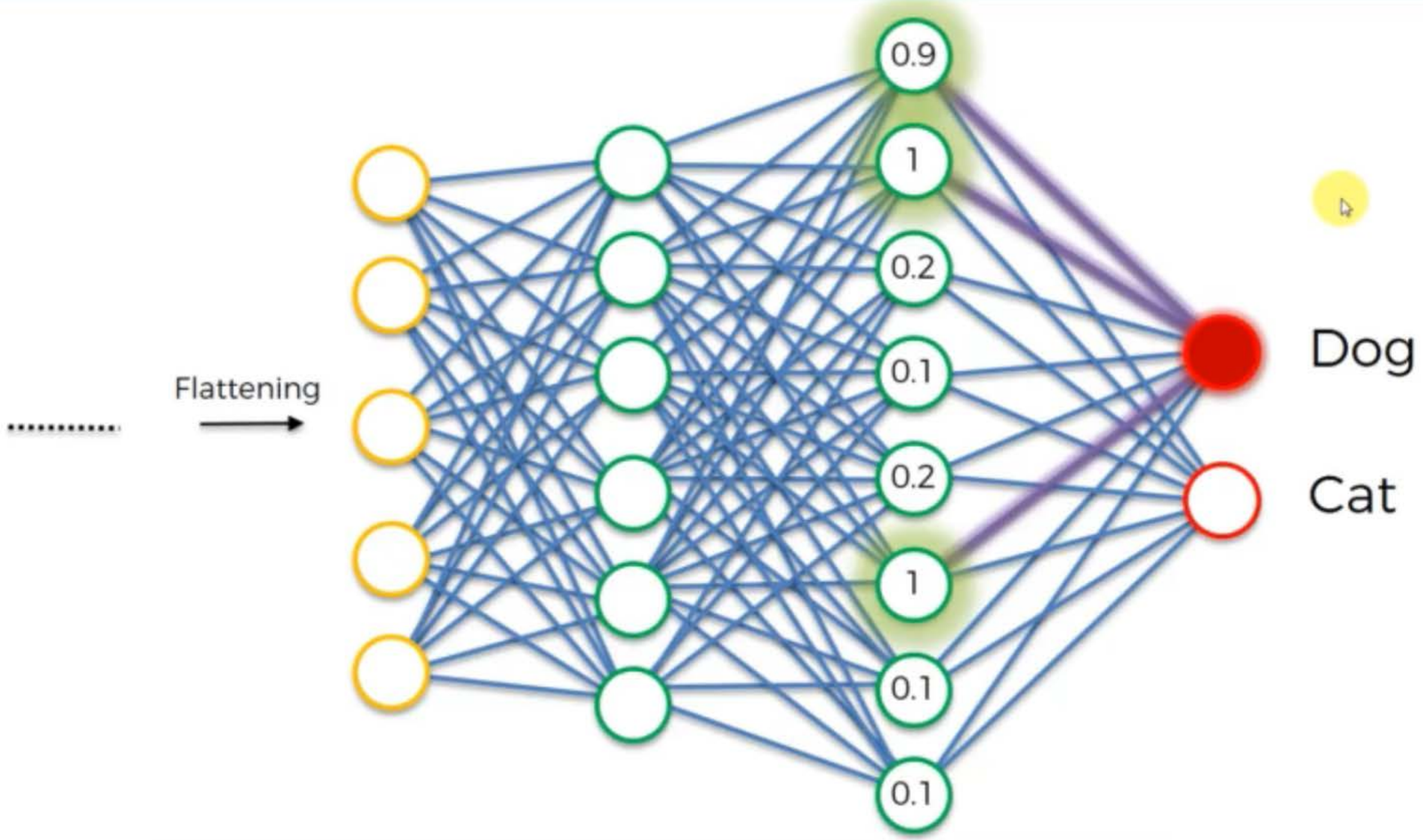
Step 4 - Full Connection



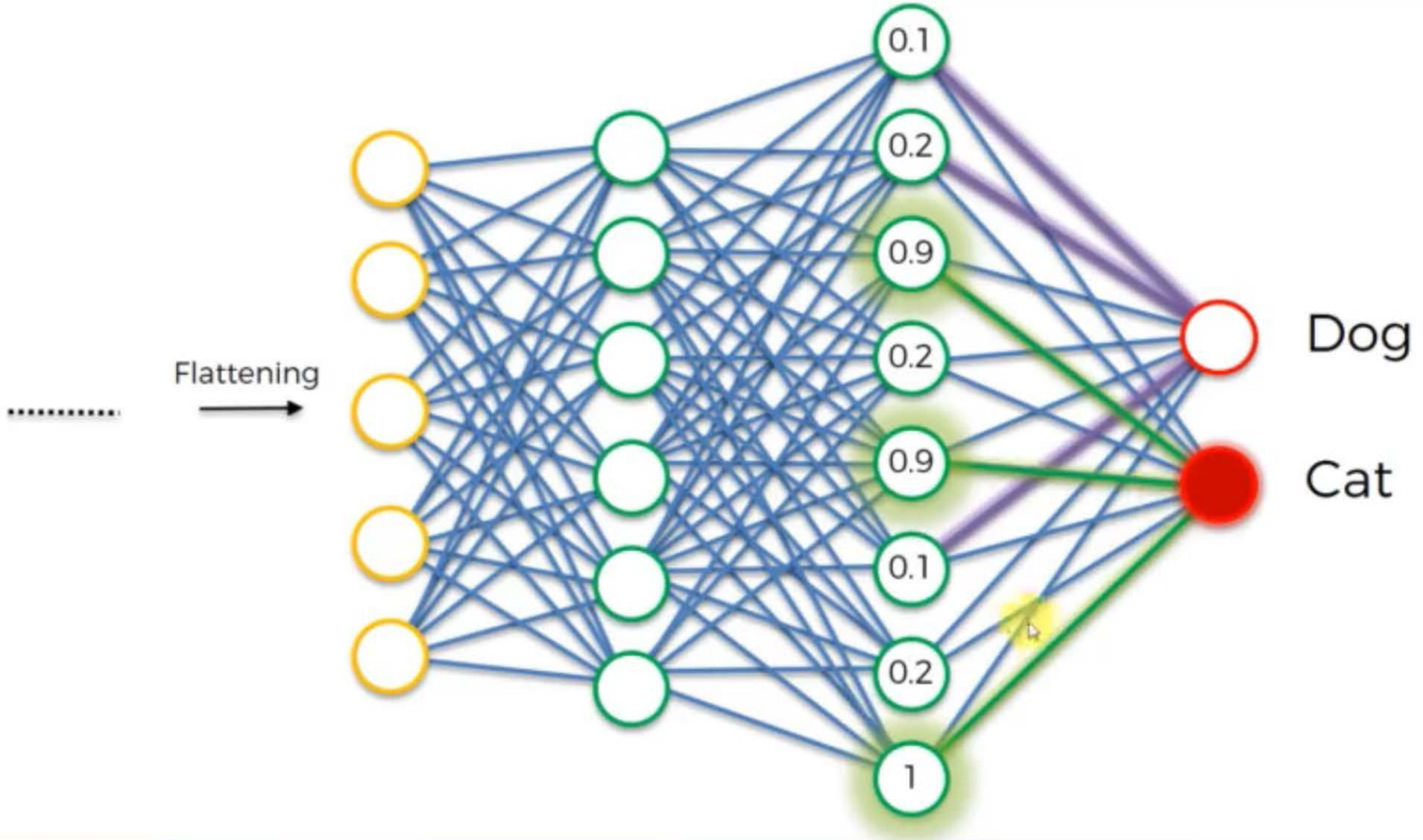
Step 4 - Full Connection



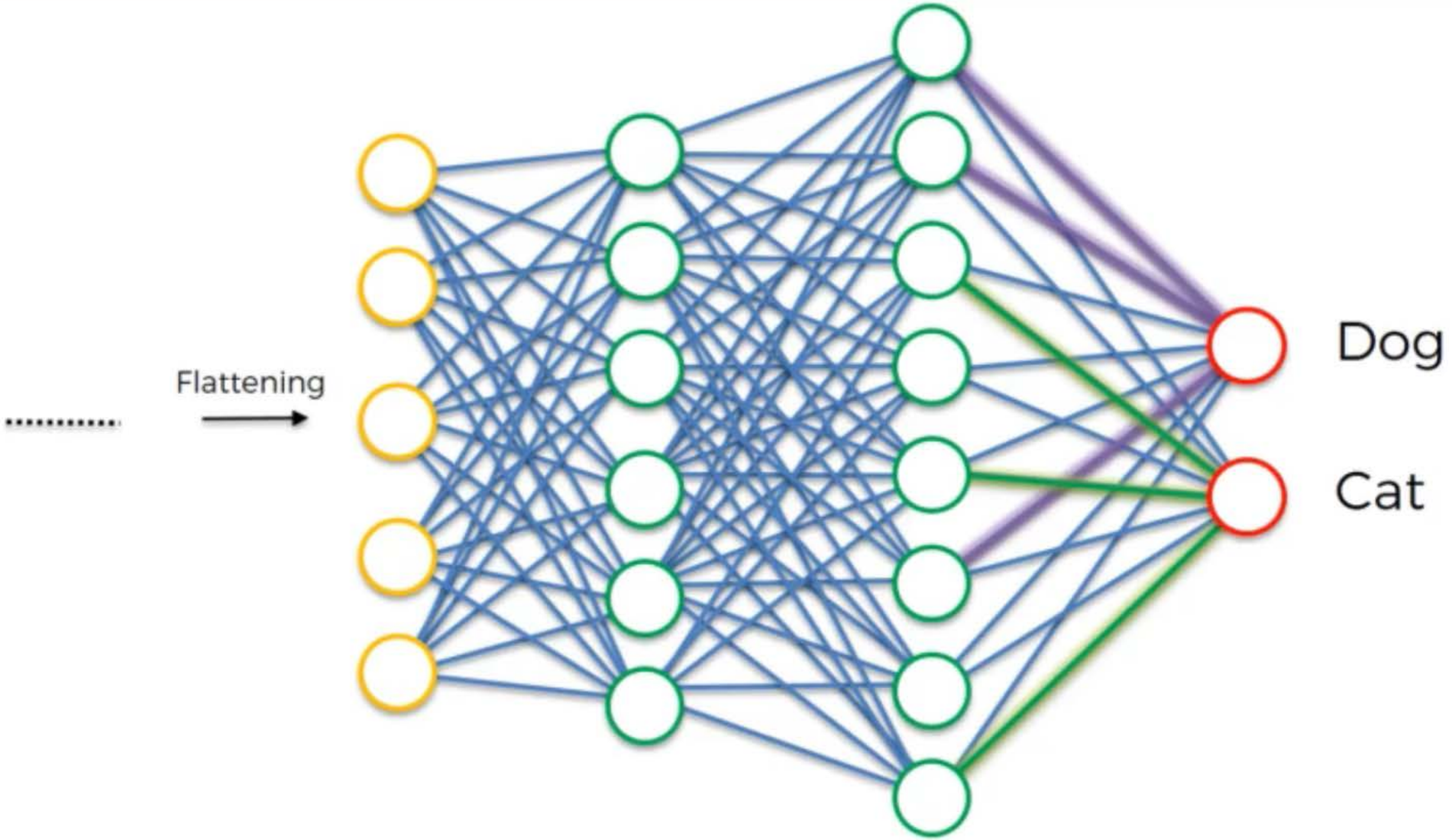
Step 4 - Full Connection



Step 4 - Full Connection

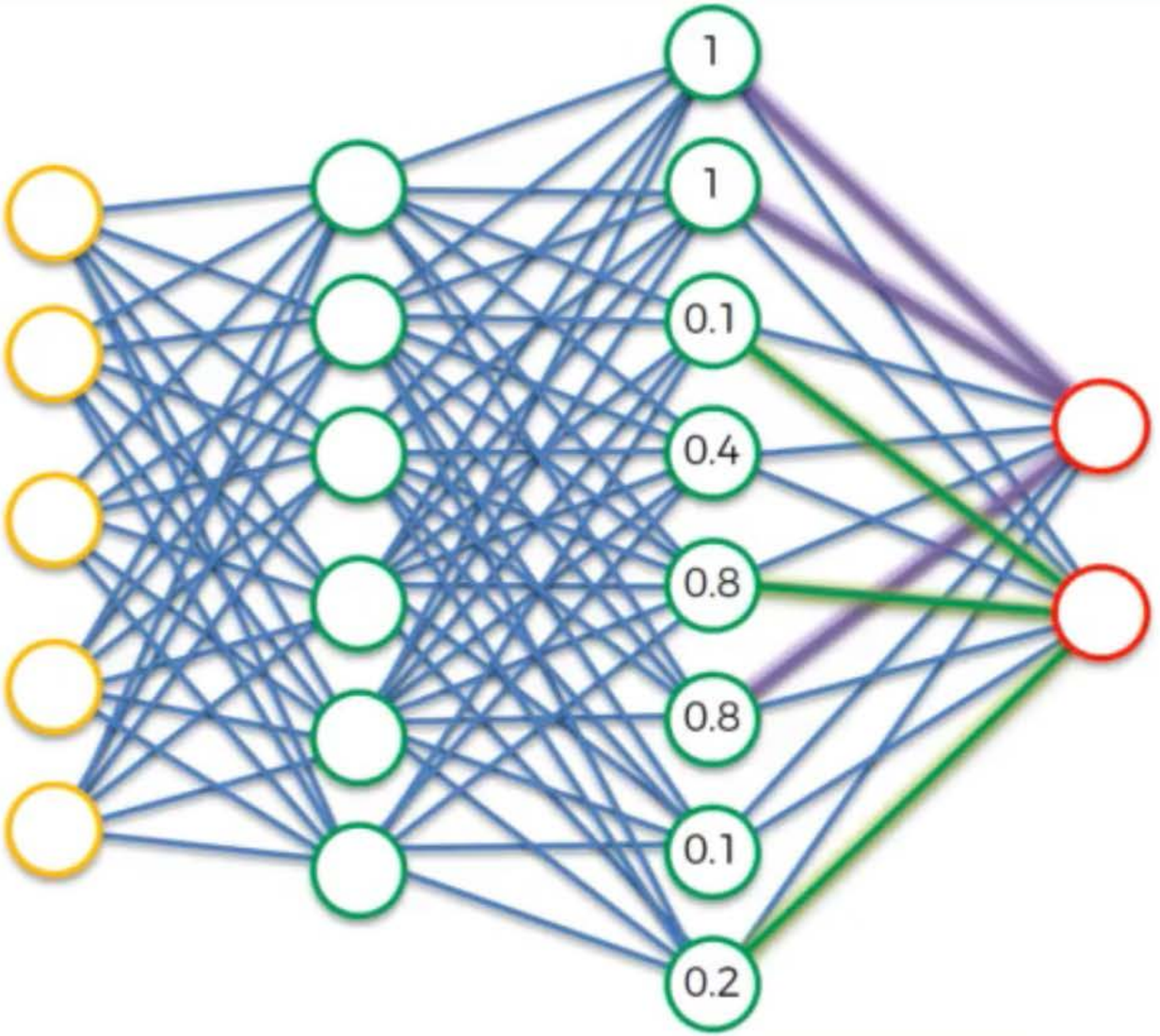


Step 4 - Full Connection



Step 4 - Full Connection

Flattening

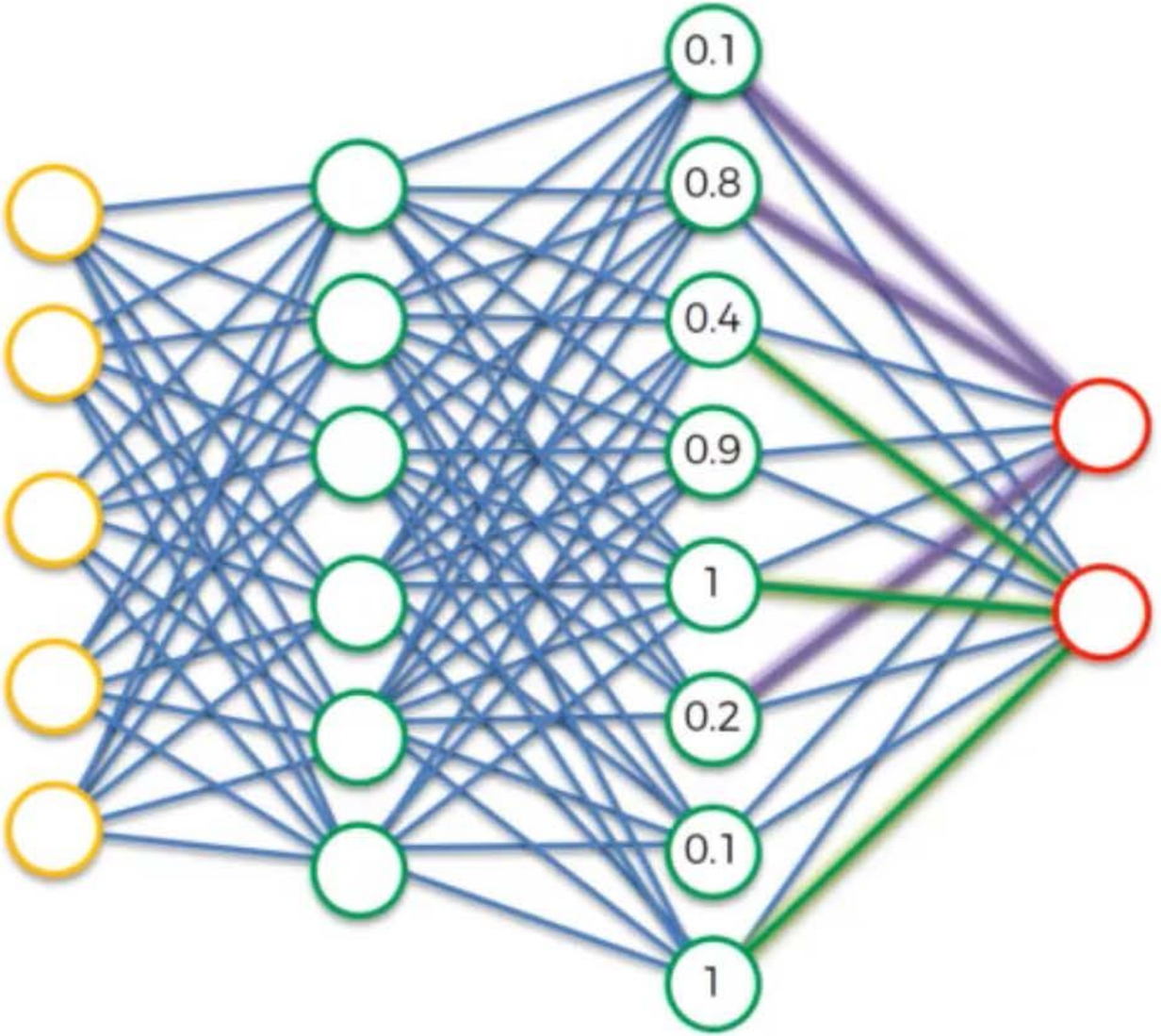


Dog 0.95

Cat 0.05

Step 4 - Full Connection

Flattening



Dog 0.21

Cat 0.79

Step 4 - Full Connection

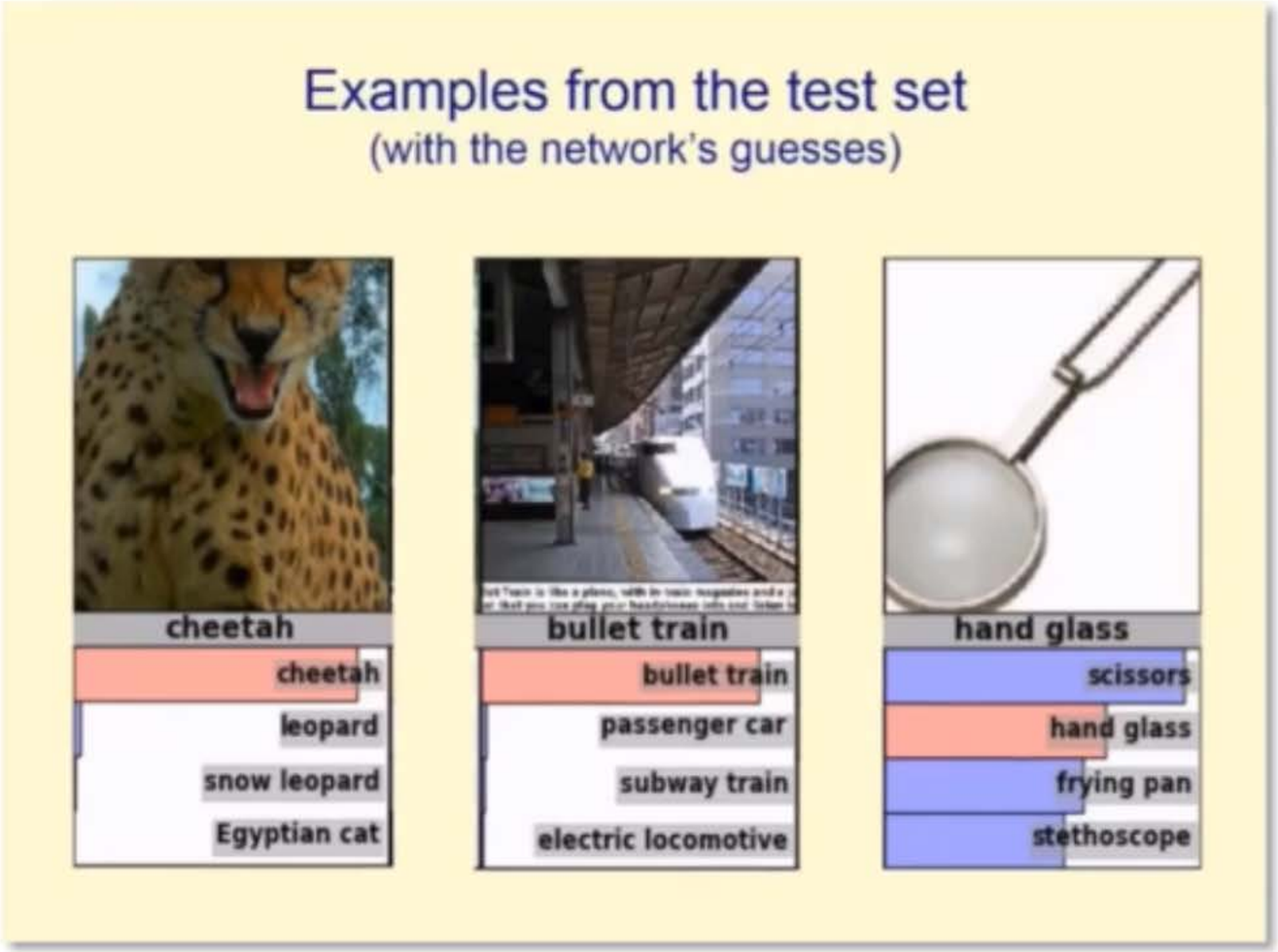
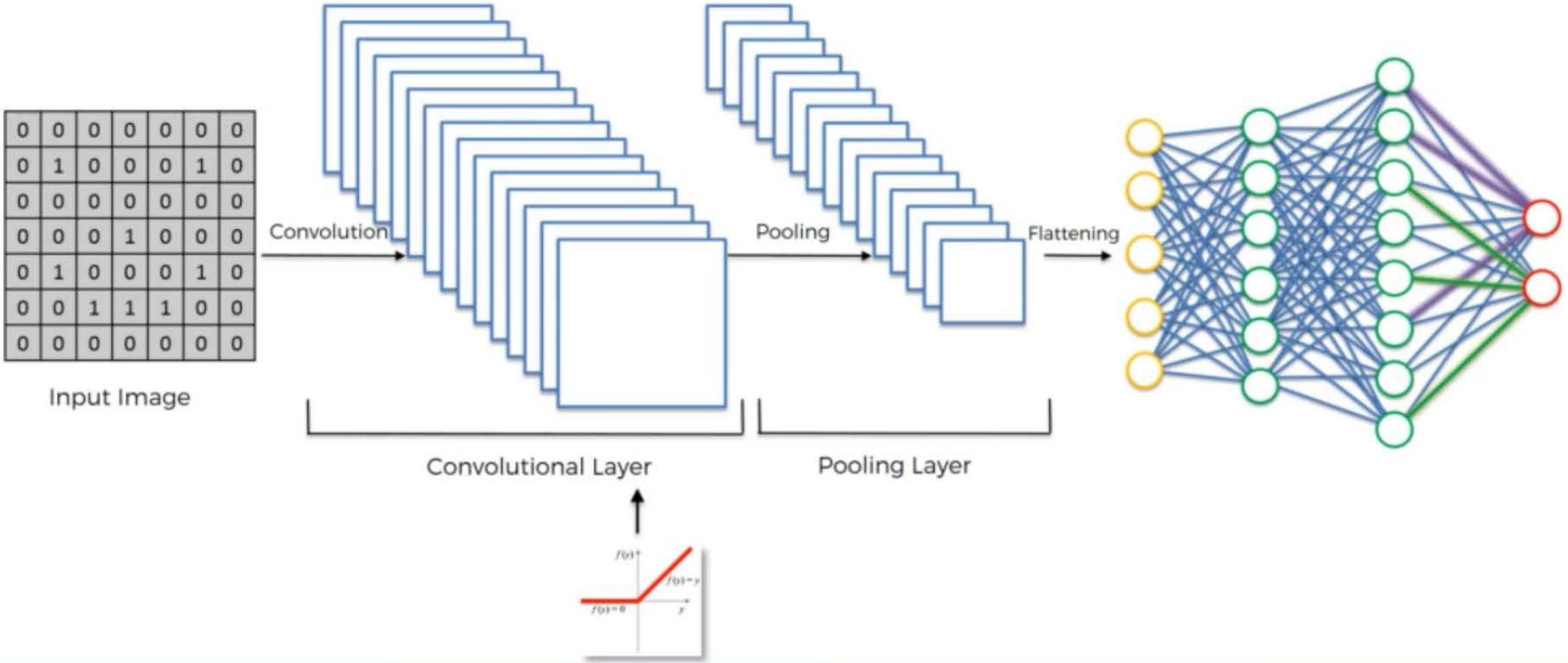


Image Source: a talk by Geoffrey Hinton

Summary

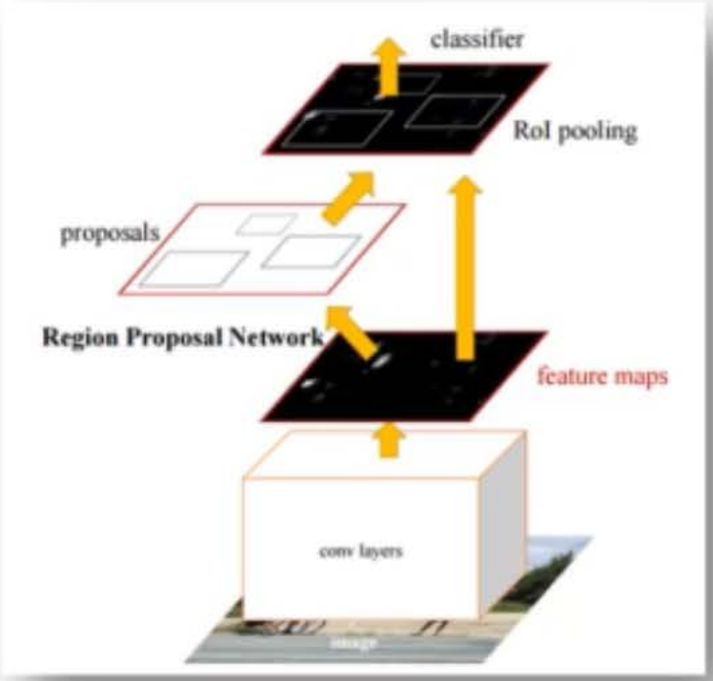


Summary

Additional Reading:

The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)

Adit Deshpande (2016)



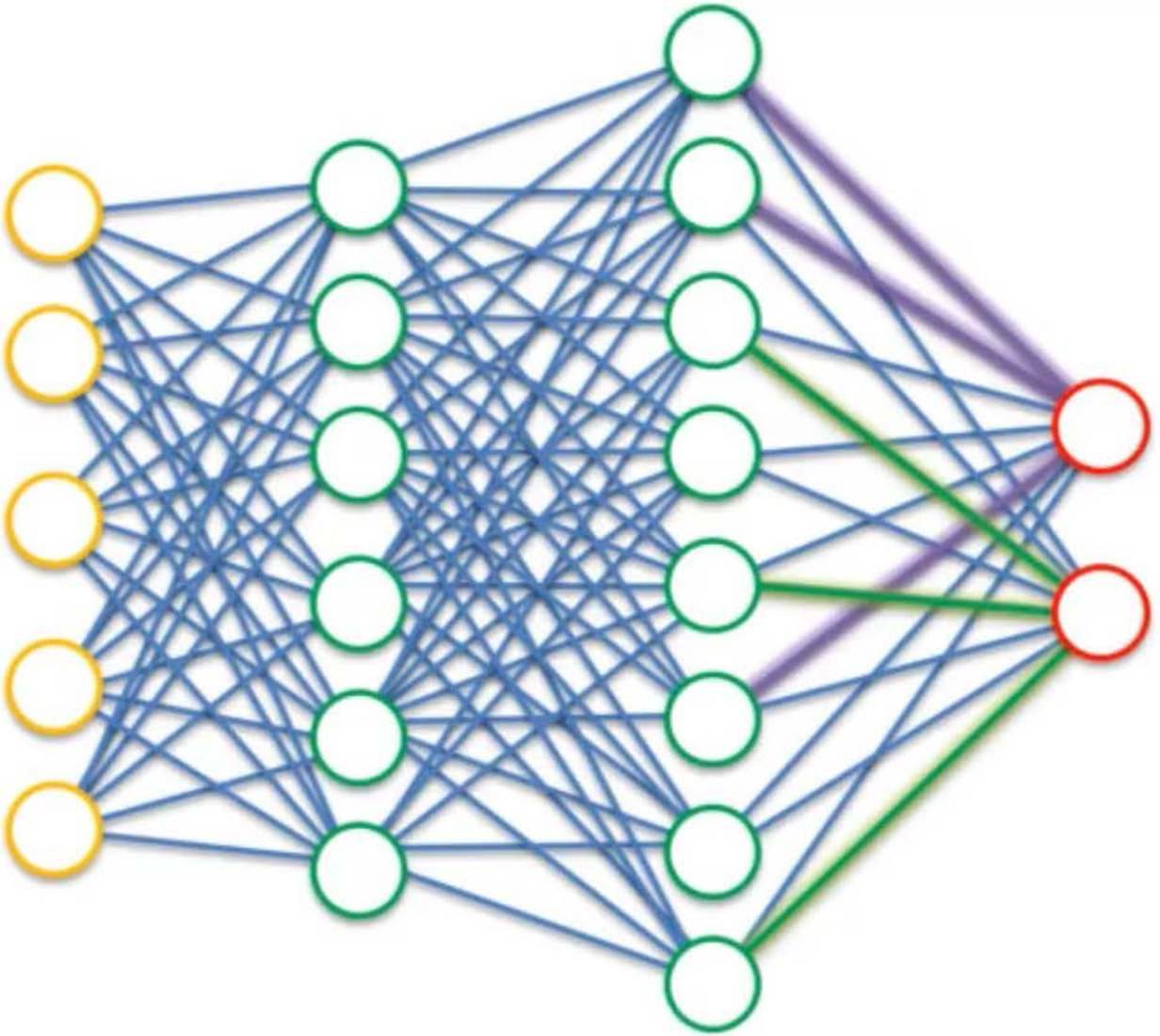
Link:

https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

Softmax & Cross-Entropy

Softmax & Cross-Entropy

Flattening



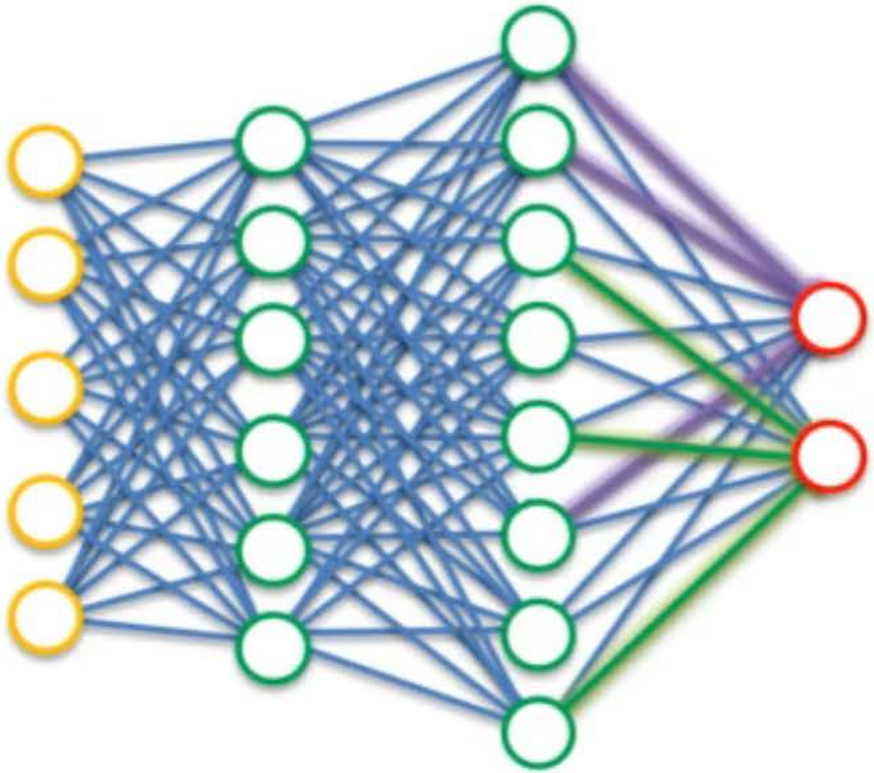
Dog 0.95

Cat 0.05

Softmax & Cross-Entropy

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Flattening



Dog → z_1 → 0.95

Cat → z_2 → 0.05

Softmax & Cross-Entropy

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

$$H(p,q) = -\sum_x p(x)\,\log q(x)$$

Softmax & Cross-Entropy

Dog 0.9

Cat 0.1

$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

1
0

Softmax & Cross-Entropy

			NN1	NN2
	Dog	1	0.9	0.6
	Cat	0	0.1	0.4
	Dog	0	0.1	0.3
	Cat	1	0.9	0.7
	Dog	1	0.4	0.1
	Cat	0	0.6	0.9

Softmax & Cross-Entropy

NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

Classification Error

1/3 = 0.33

1/3 = 0.33

Mean Squared Error

0.25

0.71

Cross-Entropy

0.38

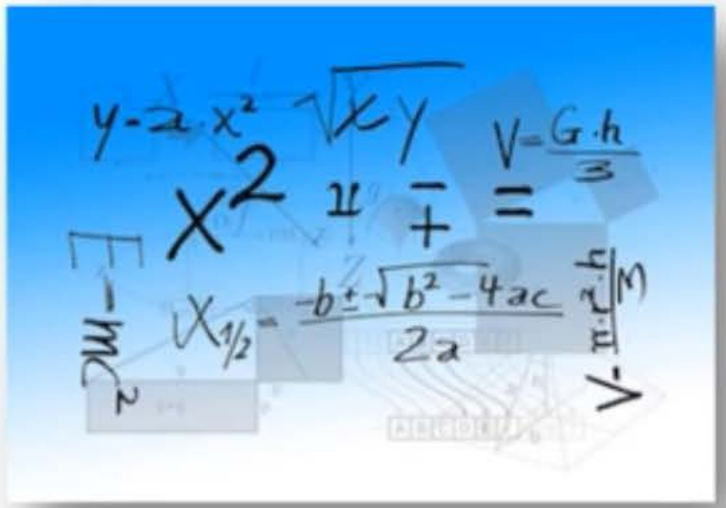
1.06

Softmax & Cross-Entropy

Additional Reading:

A Friendly Introduction to Cross-Entropy Loss

By Rob DiPietro (2016)



Link:

https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

Softmax & Cross-Entropy

Additional Reading:

How to implement a neural network Intermezzo 2

By Peter Roelants (2016)

$$\begin{aligned}
\frac{\partial \xi}{\partial z_i} &= -\sum_{j=1}^{C} \frac{\partial t_j log(y_j)}{\partial z_i} = -\sum_{j=1}^{C} t_j \frac{\partial log(y_j)}{\partial z_i} = -\sum_{j=1}^{C} t_j \frac{1}{y_j} \frac{\partial y_j}{\partial z_i} \\
&= -\frac{t_i}{y_i} \frac{\partial y_i}{\partial z_i} - \sum_{j \neq i}^{C} \frac{t_j}{y_j} \frac{\partial y_j}{\partial z_i} = -\frac{t_i}{y_i} y_i (1 - y_i) - \sum_{j \neq i}^{C} \frac{t_j}{y_j} (-y_j y_i) \\
&= -t_i + t_i y_i + \sum_{j \neq i}^{C} t_j y_i = -t_i + \sum_{j=1}^{C} t_j y_i = -t_i + y_i \sum_{j=1}^{C} t_j \\
&= y_i - t_i
\end{aligned}$$

Link:

http://peterroelants.github.io/posts/neural_network_implementation_intermezzo02/

Convolutional Neural Network

Importing the libraries

In [2]:
```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

In [3]:
```
tf.__version__
```

Out[3]:
```
'2.3.0'
```

Part 1 - Data Preprocessing

Preprocessing the Training set

In [4]:
```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
```

```
Found 8000 images belonging to 2 classes.
```

Preprocessing the Test set

In [5]:
```
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

```
Found 2000 images belonging to 2 classes.
```

Part 2 - Building the CNN

Initialising the CNN

In [6]:
```
cnn = tf.keras.models.Sequential()
```

Step 1 - Convolution

In [7]:
```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=[64, 64, 3]))
```

Step 2 - Pooling

In [8]:
```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

Adding a second convolutional layer

In [9]:
```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

Step 3 - Flattening

In [10]:
```
cnn.add(tf.keras.layers.Flatten())
```

Step 4 - Full Connection

In [11]:
```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

Step 5 - Output Layer

In [12]:
```
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Part 4 - Making a single prediction

In [15]:
```
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('dataset/single_prediction/cat_or_dog_1.jpg', target_size = (64, 64))
test_image = image.img_to_array(test_image)
#for the extra dimension of batch size 32
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image)
#we get the right index for each class
training_set.class_indices
#we have only one batch and one image hence,[0][0]
if result[0][0] == 1:
  prediction = 'dog'
else:
  prediction = 'cat'
```

In [16]:
```
print(prediction)
```

```
dog
```