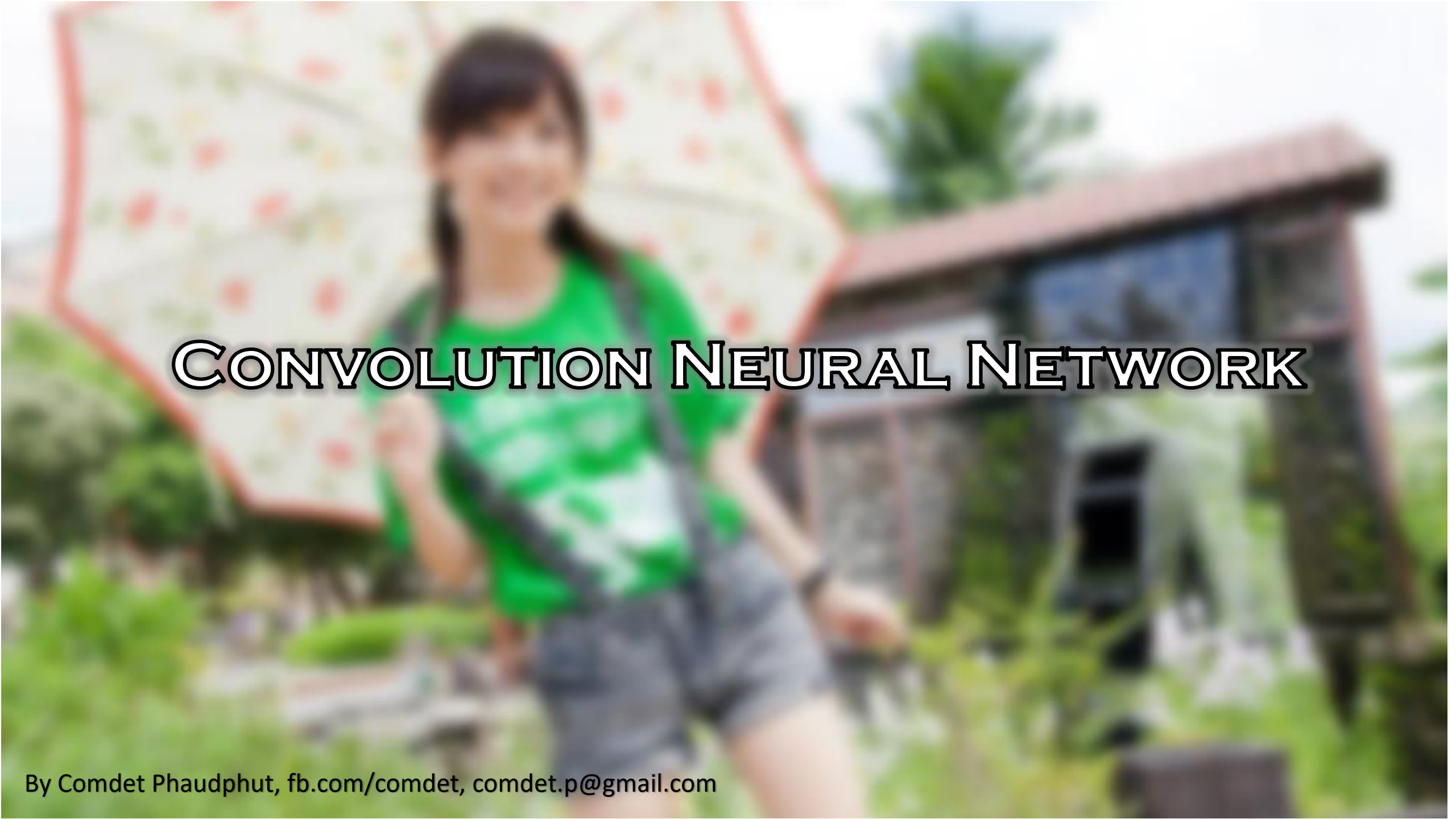
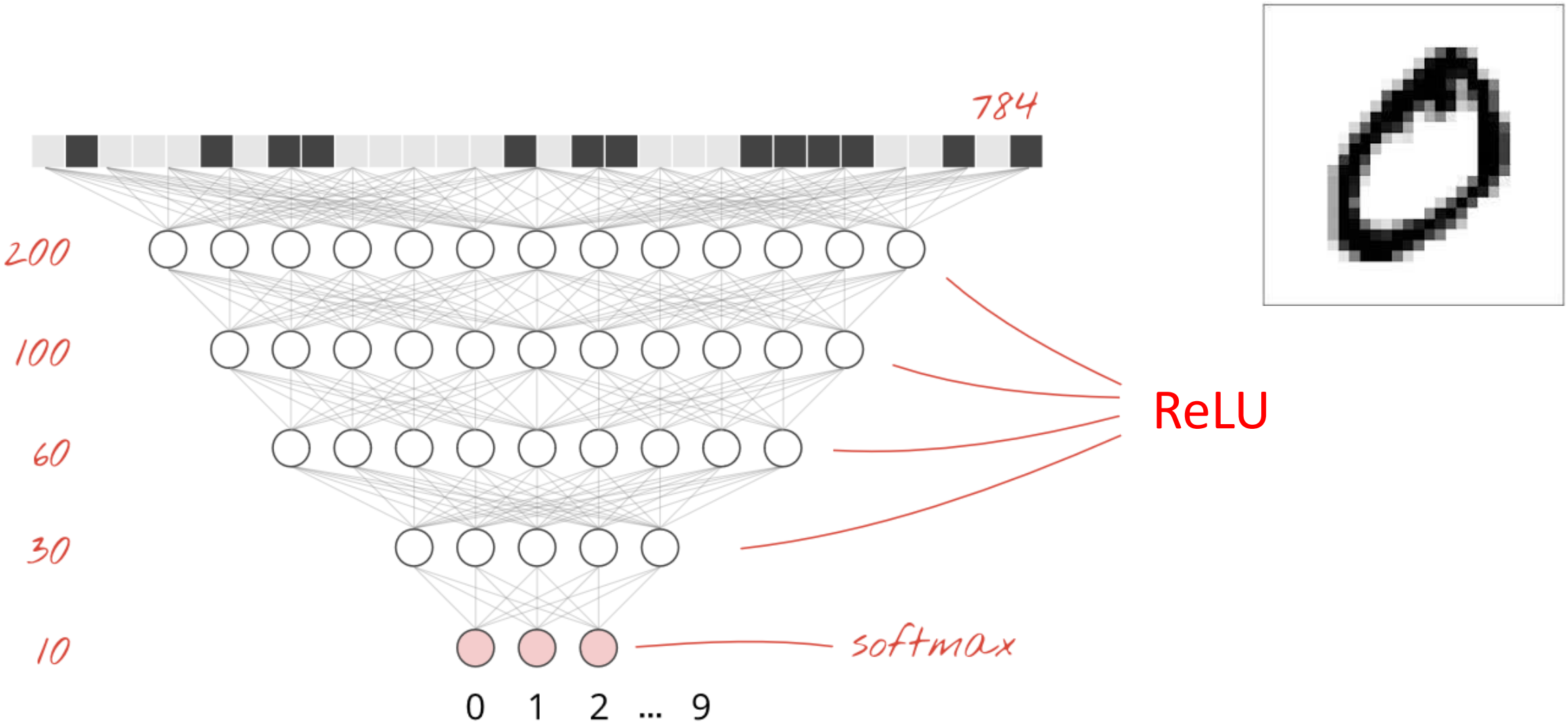


Convolution Neural Network

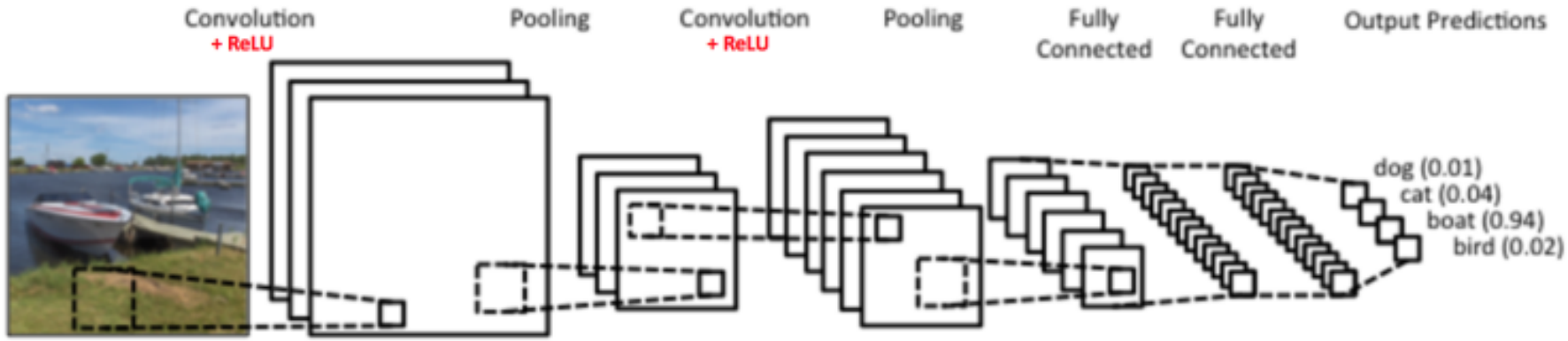
By Comdet Phaudphut, fb.com/comdet, comdet.p@gmail.com

Review Prev Lab

- We add 4 hidden layers.
- We change from sigmoid to ReLU
- But we didn’t gain more accuracy as expected.
- Why ?
  - code errors or bugs?
  - machine mistake?
  - Take a look closer, what happen.


784
200
100
60
30
10
ReLU
softmax
0 1 2 ... 9

Convolution Neural Network

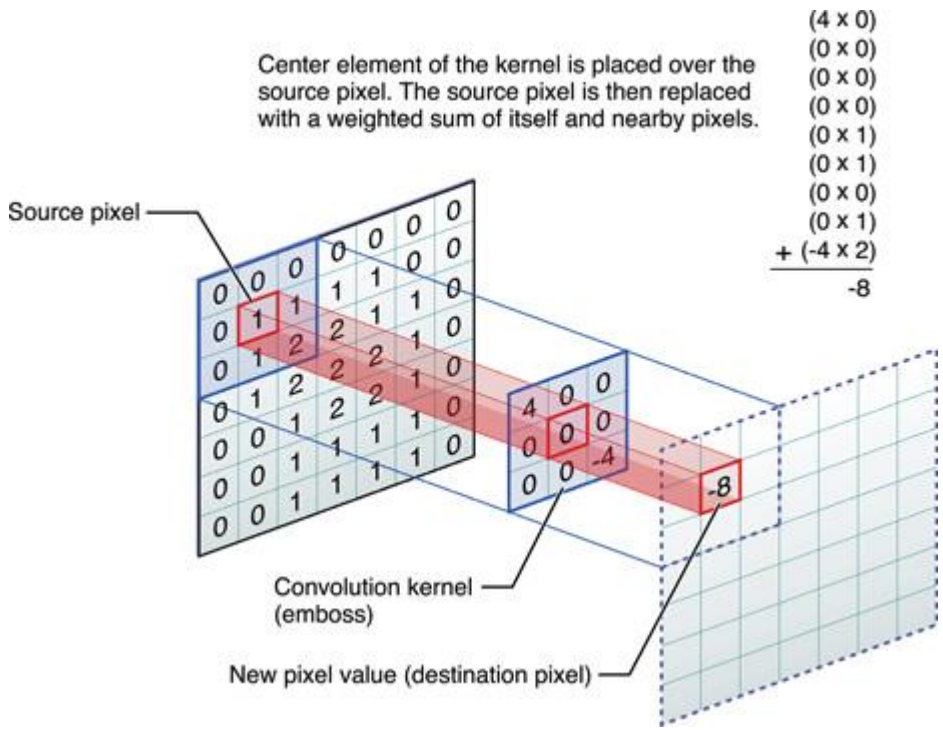


Wait!, Let’s learn some basic background

What’s convolutions

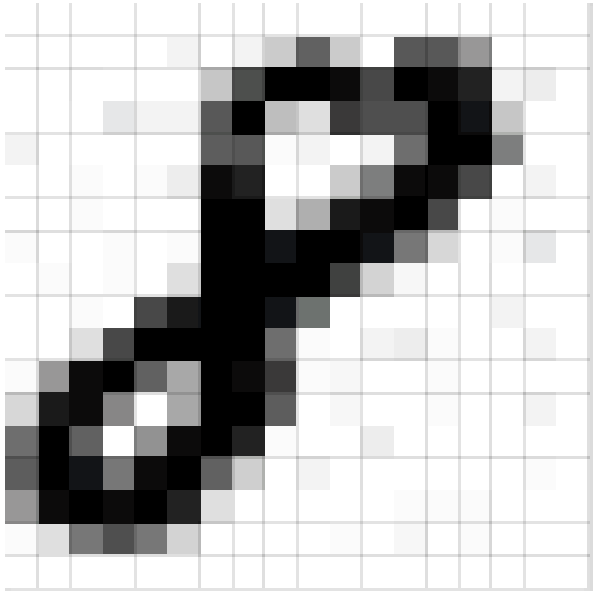
In Photoshop dose anyone try Image Effect before?
Yes! that is some case of Convolution

How convolution work



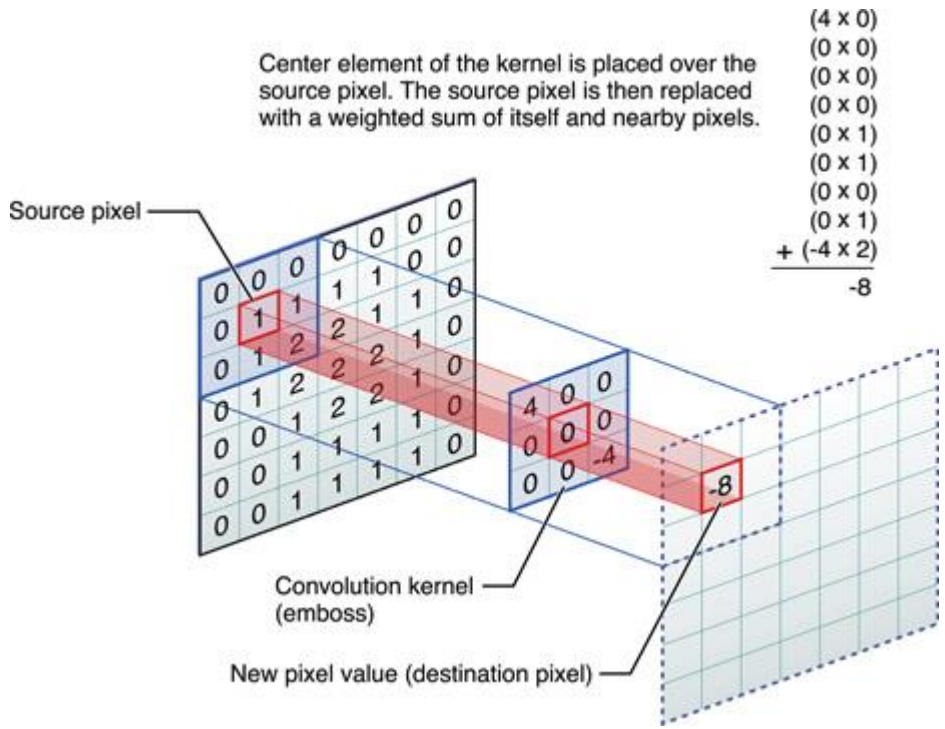
$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) \cdot g(x-\tau)\, d\tau$$

How convolution work

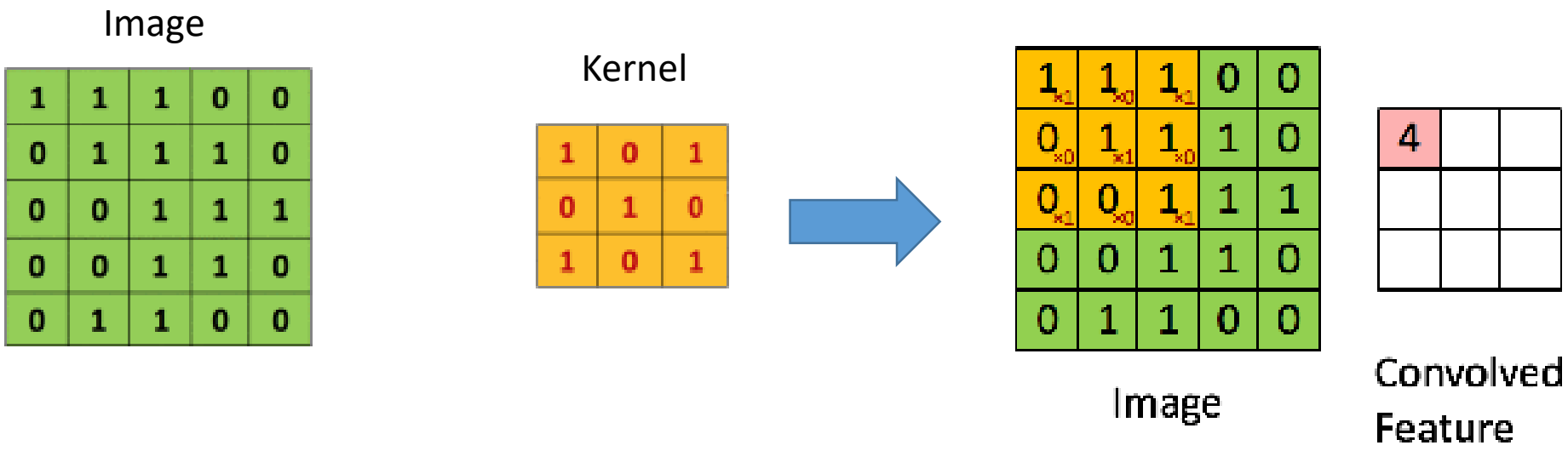


We has an images and represent as matrix of pixel

How convolution work

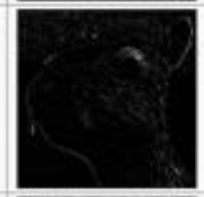


How convolution work



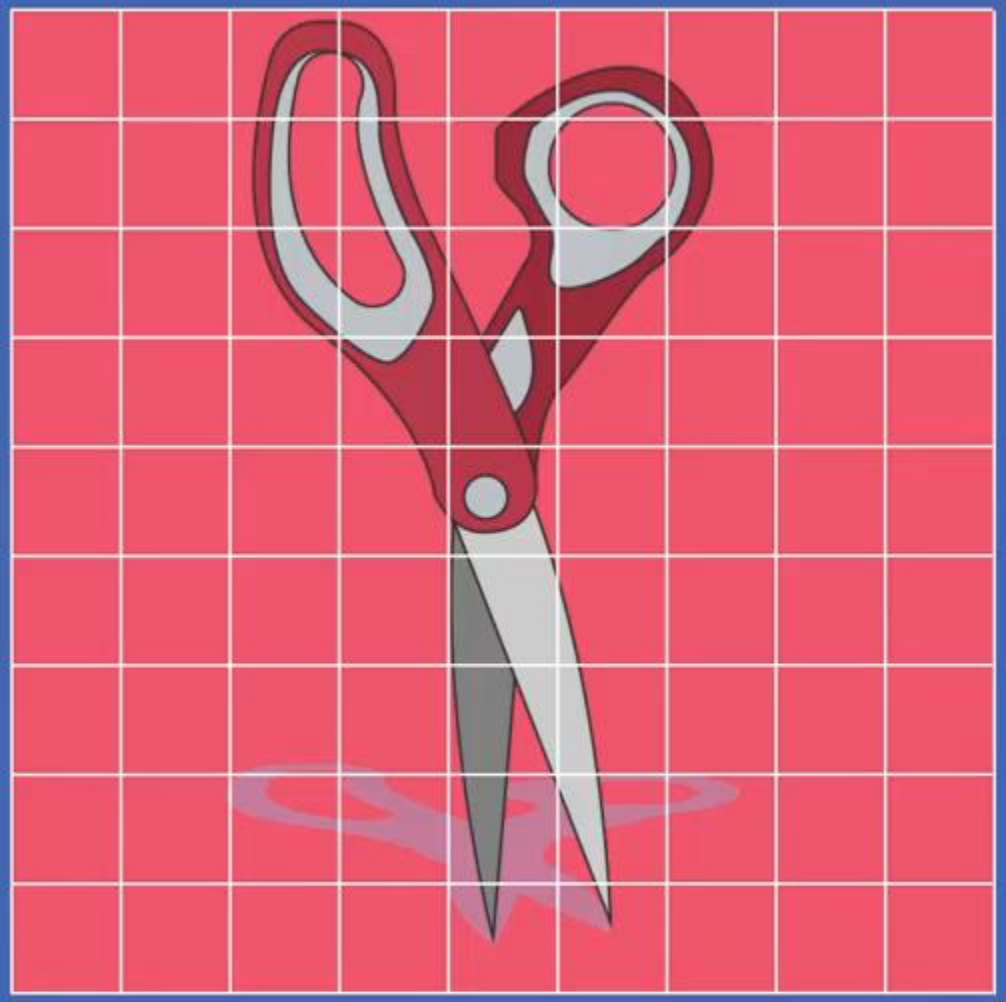
How convolution work

Example of same input image but difference kernel

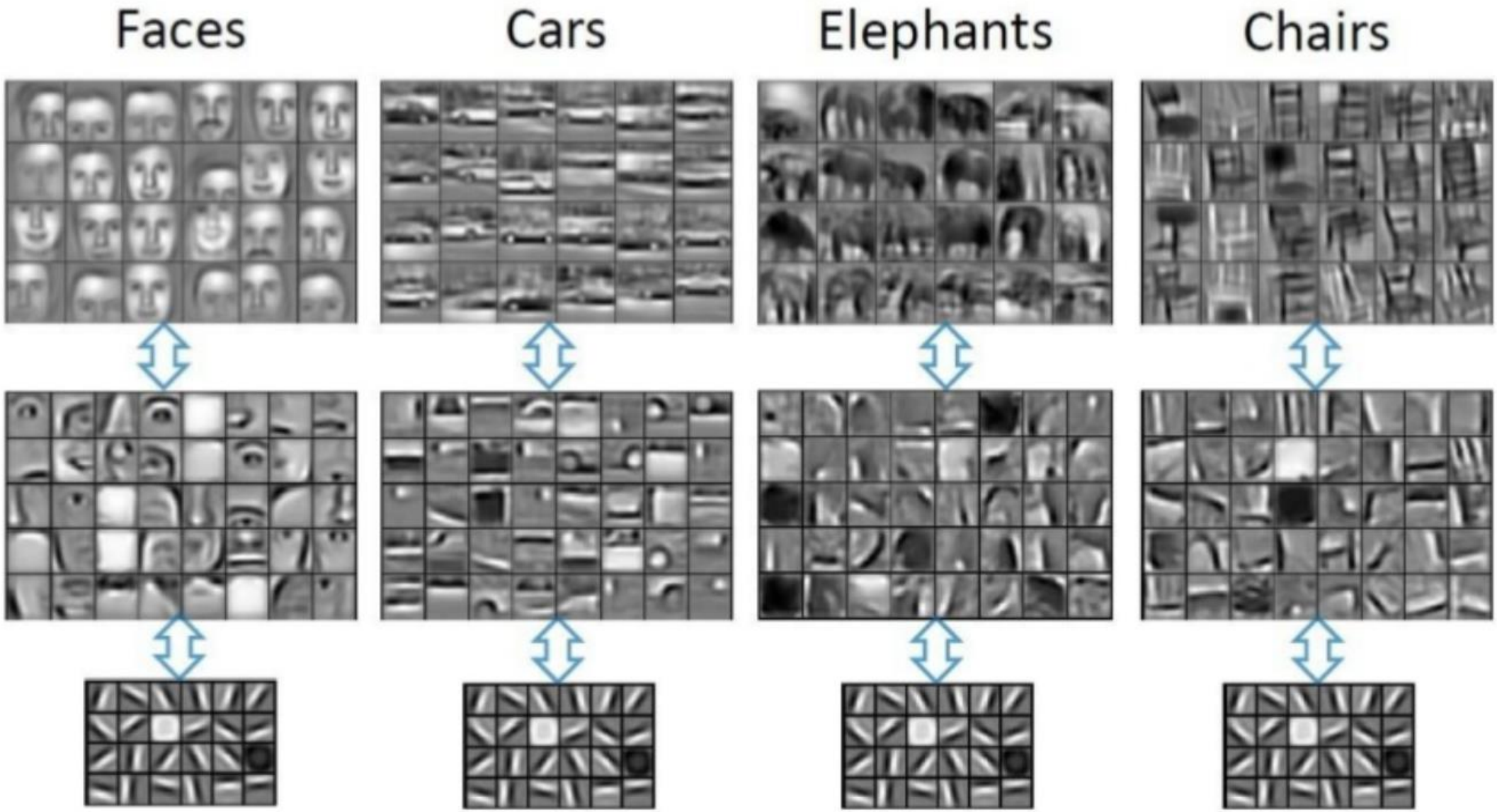
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

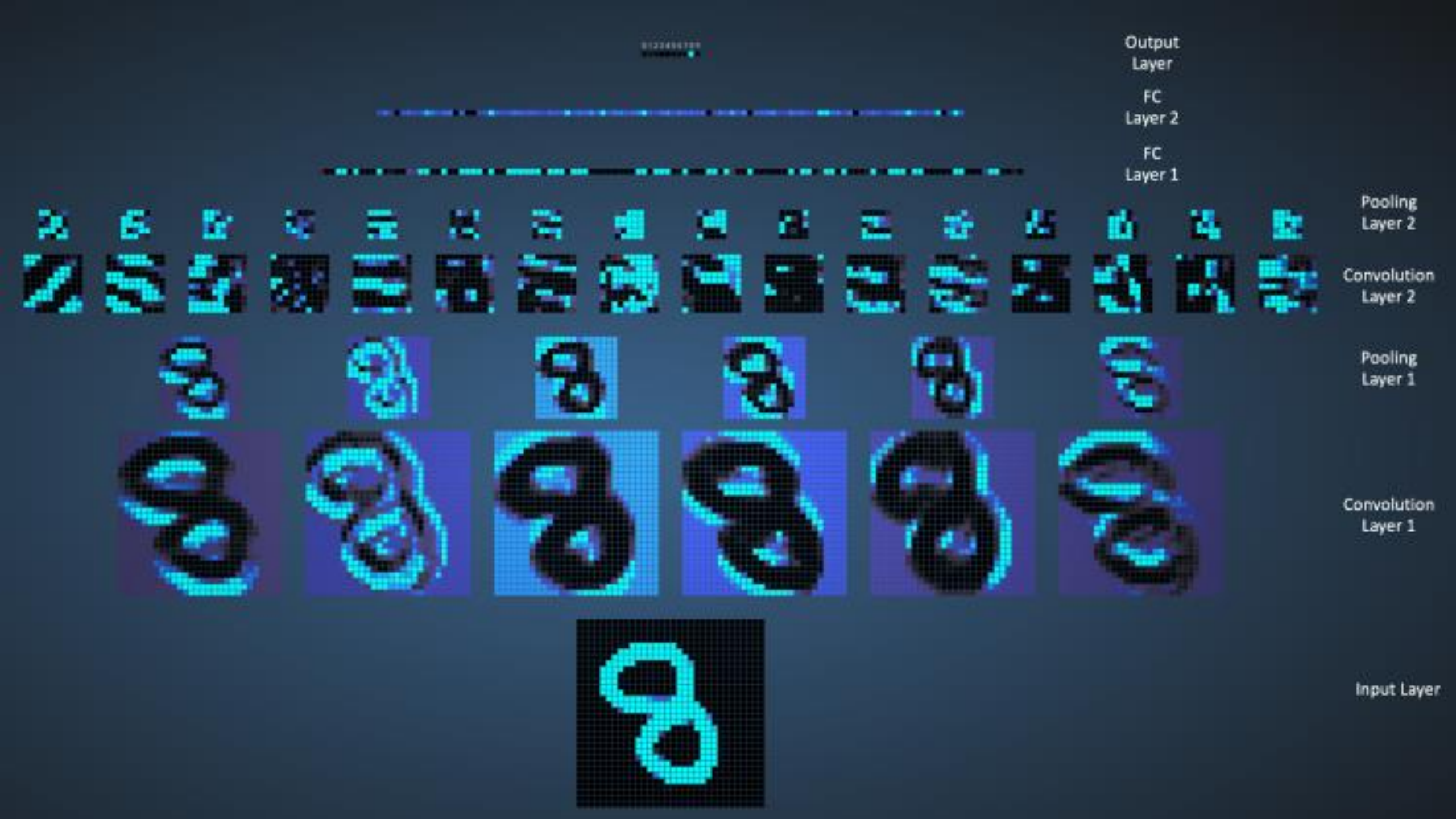
How convolution has been involved

Kernel presented as weight, if we update it to find proper filter
Extract features from image before training it, that can increase accuracy

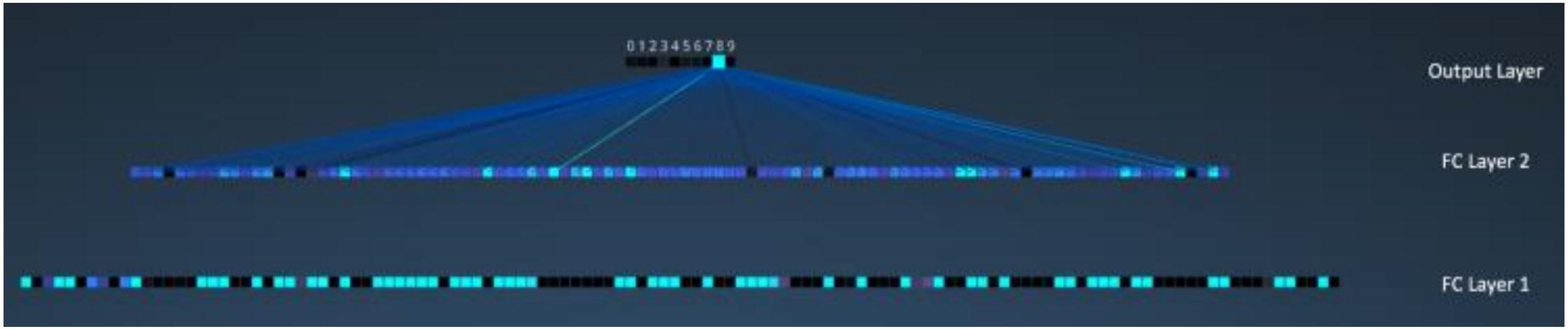


Hierarchical Feature Learning



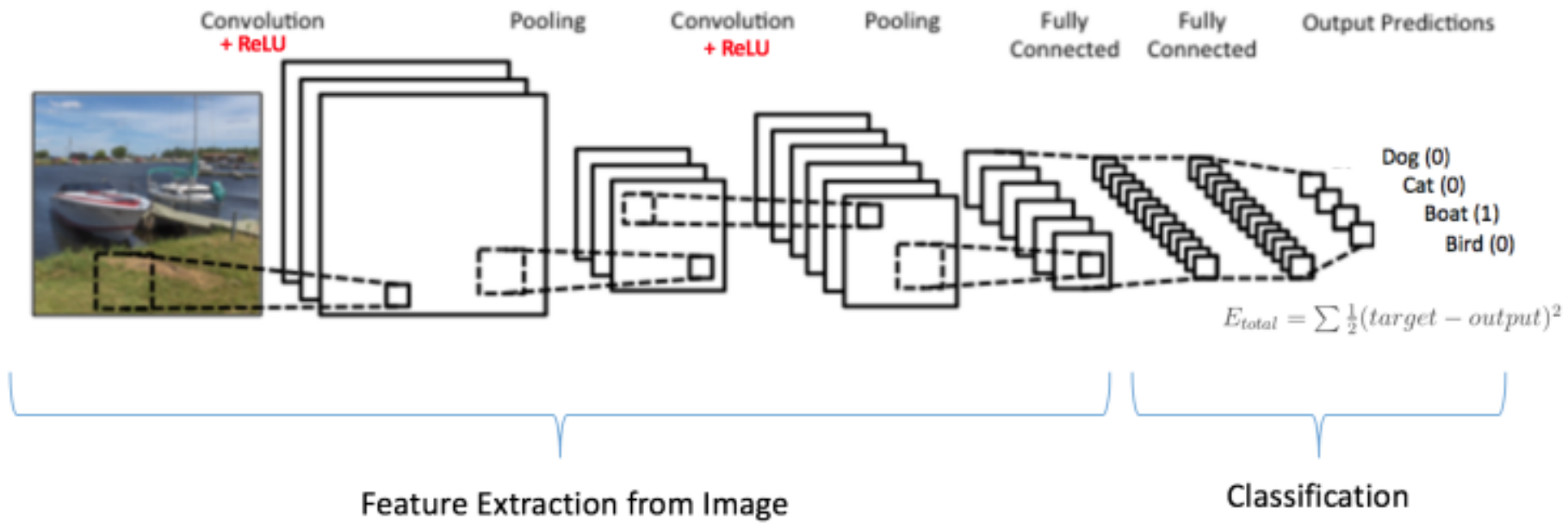

Output Layer
FC Layer 2
FC Layer 1
Pooling Layer 2
Convolution Layer 2
Pooling Layer 1
Convolution Layer 1
Input Layer

How convolution has been involved

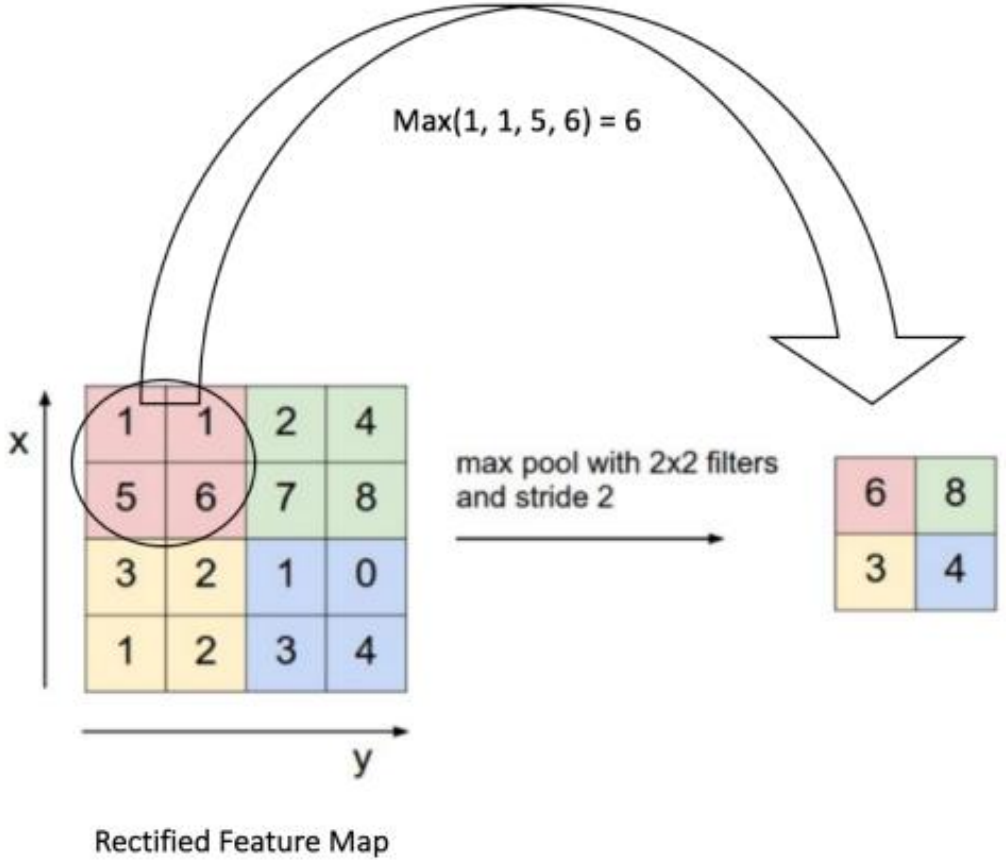


At the end we use fully connected (MLP)

Convolution Neural Network

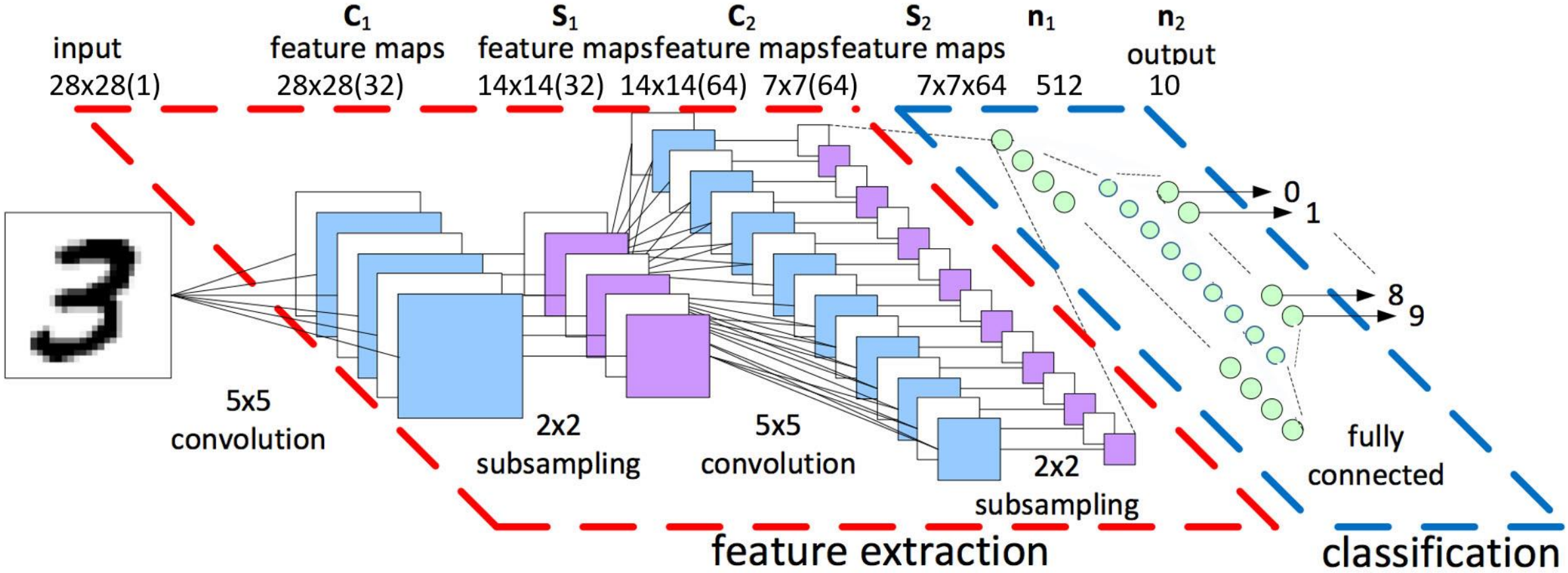


Max pooling



- makes the input representations (feature dimension) smaller and more manageable
- reduces the number of parameters and computations in the network
- makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling – since we take the maximum / average value in a local neighborhood).
- helps us arrive at an almost scale invariant representation of our image (the exact term is “equivariant”). This is very powerful since we can detect objects in an image no matter where they are located.

Max pooling


input
28x28(1)
C1
feature maps
28x28(32)
S1
feature maps
14x14(32)
C2
feature maps
14x14(64)
S2
feature maps
7x7(64)
n1
7x7x64
512
n2
output
10
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification

```
X = tf.placeholder(tf.float32,[None, 28,28, 1])
Y_ = tf.placeholder(tf.float32, [None, 10])

conv1_weights = tf.Variable(tf.truncated_normal([5, 5, 1, 32],  # 5x5 filter, depth 32.
                            stddev=0.1))
conv1_biases = tf.Variable(tf.zeros([32]))

conv2_weights = tf.Variable(tf.truncated_normal([5, 5, 32, 64], #5x5 filter, depth 32.
                                stddev=0.1))
conv2_biases = tf.Variable(tf.constant(0.1, shape=[64]))

fc1_weights = tf.Variable(  # fully connected, depth 512.
                            tf.truncated_normal([7 * 7 * 64, 512],stddev=0.1))
fc1_biases = tf.Variable(tf.constant(0.1, shape=[512]))


fc2_weights = tf.Variable(tf.truncated_normal([512, 10],
                                              stddev=0.1))
fc2_biases = tf.Variable(tf.constant(0.1, shape=[10]))
```

Input image 28x28 (ขาวดำ 1 มิติ)

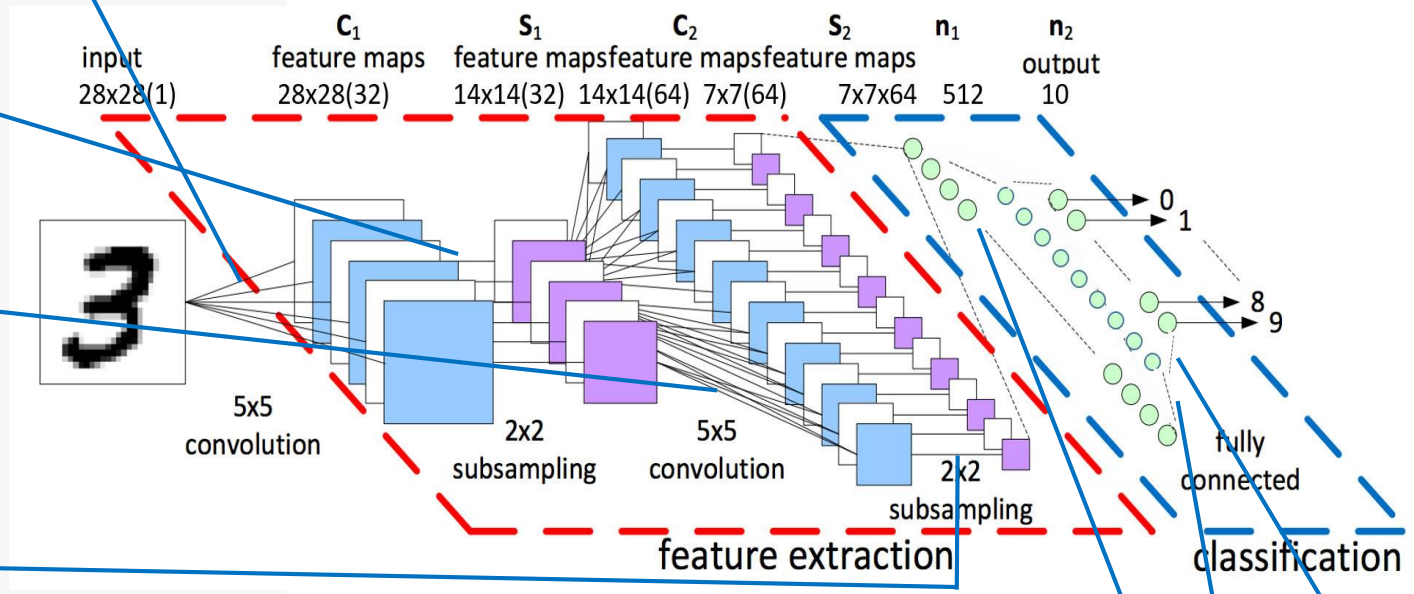
Weight convo1 5x5 input 1 output 32

Weight convo2 5x5 input 32 output 64

Fully connected from 7x7x64

Output 512 hidden to 10 class numbers

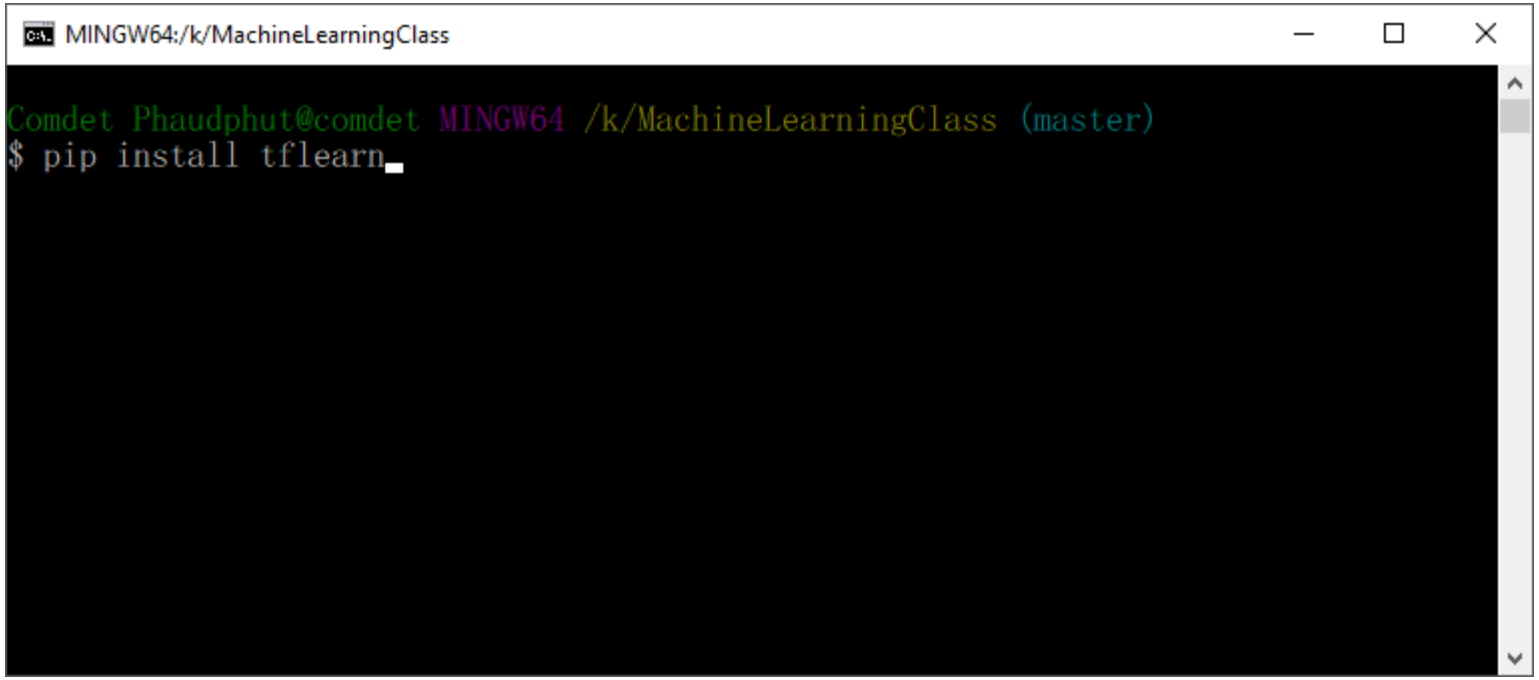
```
conv = tf.nn.conv2d(X,
                    conv1_weights,strides=[1, 1, 1, 1],
                    padding='SAME')

relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))

pool = tf.nn.max_pool(relu,
                          ksize=[1, 2, 2, 1],
                          strides=[1, 2, 2, 1],
                          padding='SAME')

conv = tf.nn.conv2d(pool,
                         conv2_weights,
                         strides=[1, 1, 1, 1],
                         padding='SAME')

relu = tf.nn.relu(tf.nn.bias_add(conv, conv2_biases))

pool = tf.nn.max_pool(relu,
                          ksize=[1, 2, 2, 1],
                          strides=[1, 2, 2, 1],
                          padding='SAME')

reshape = tf.reshape(pool, shape=[-1,7 * 7 * 64])

hidden = tf.nn.relu(tf.matmul(reshape, fc1_weights) + fc1_biases)

Ylogits = tf.matmul(hidden, fc2_weights) + fc2_biases

Y = tf.nn.softmax(Ylogits)
```



Run & open tensorboard
tensorboard –logdir=“logs”
and see what happen!

Ummm…we should know helper.

tflearn


MINGW64:/k/MachineLearningClass
Comdet Phaudphut@comdet MINGW64 /k/MachineLearningClass (master)
$ pip install tflearn