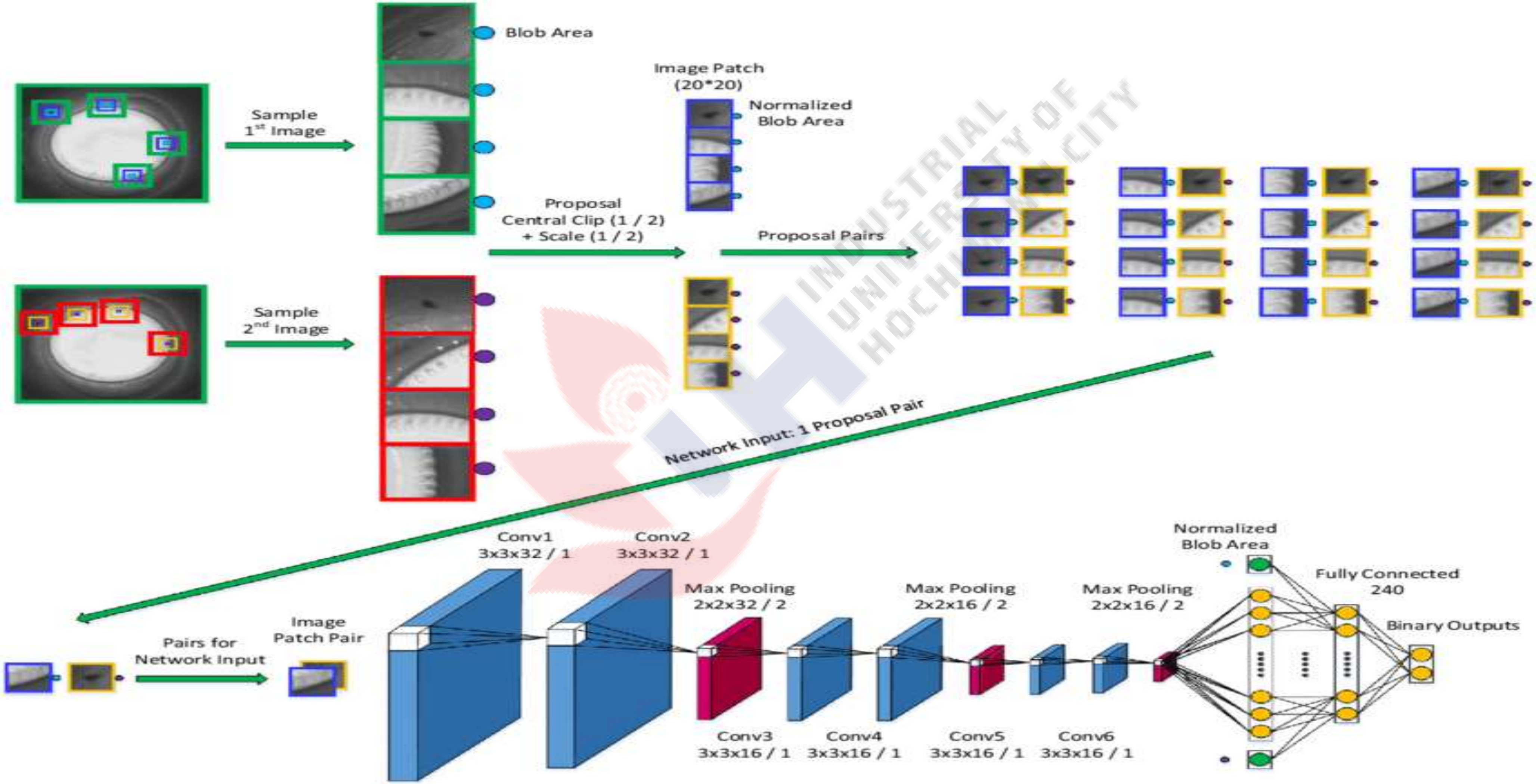
Convolutional Neural Network(CNN)

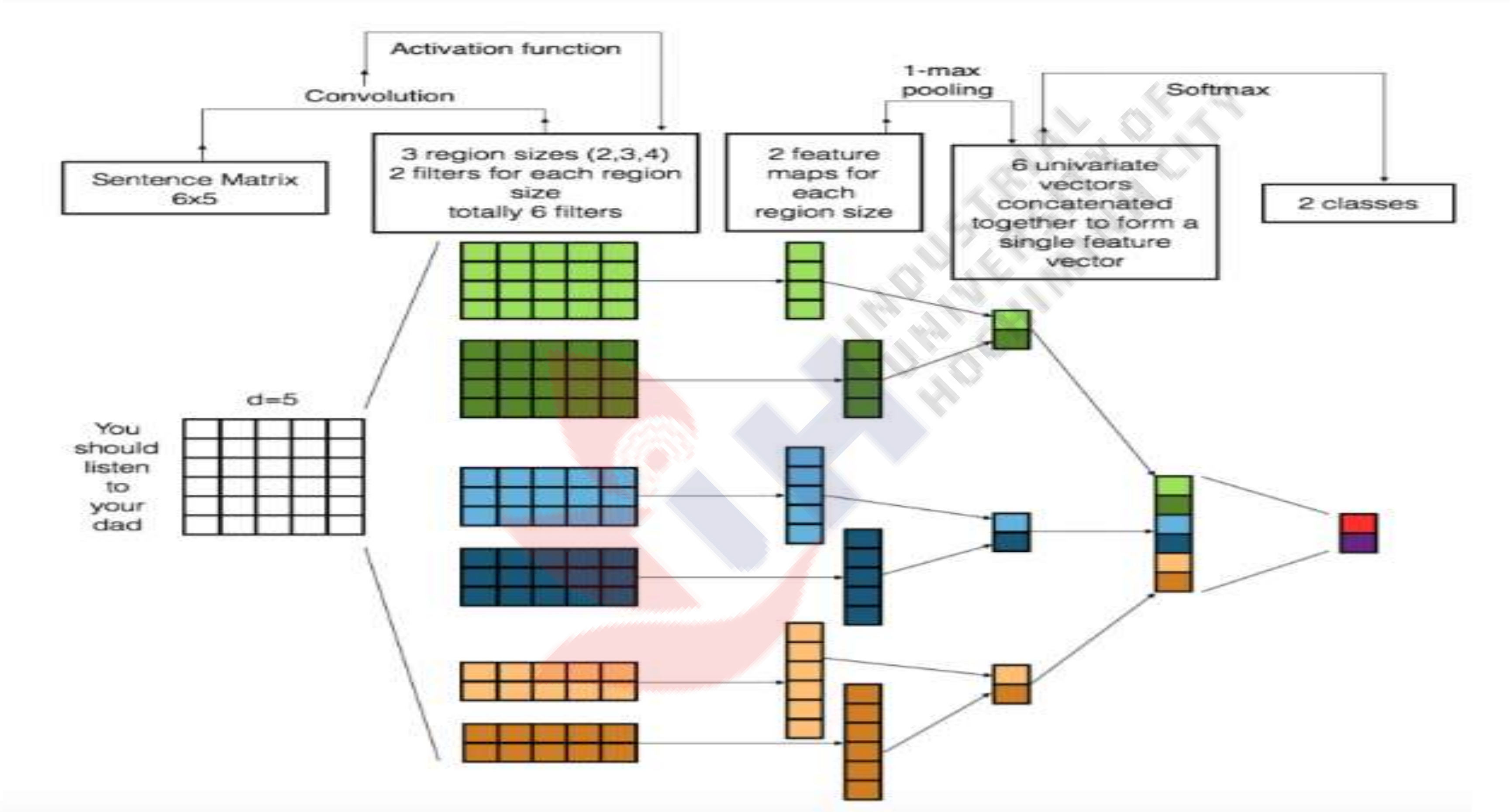
Các bài toán ứng dụng CNN

- Computer Vision



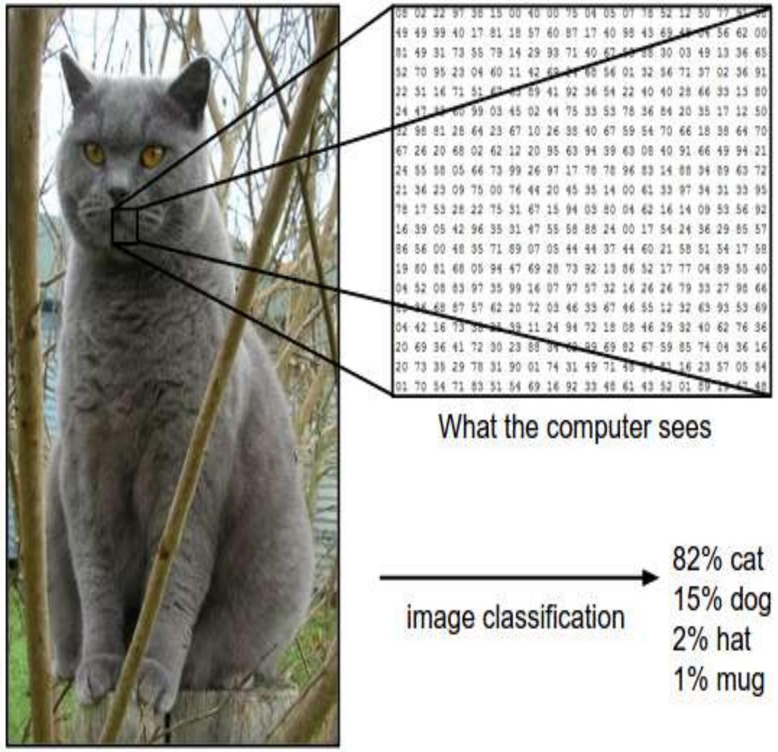
Các bài toán ứng dụng CNN

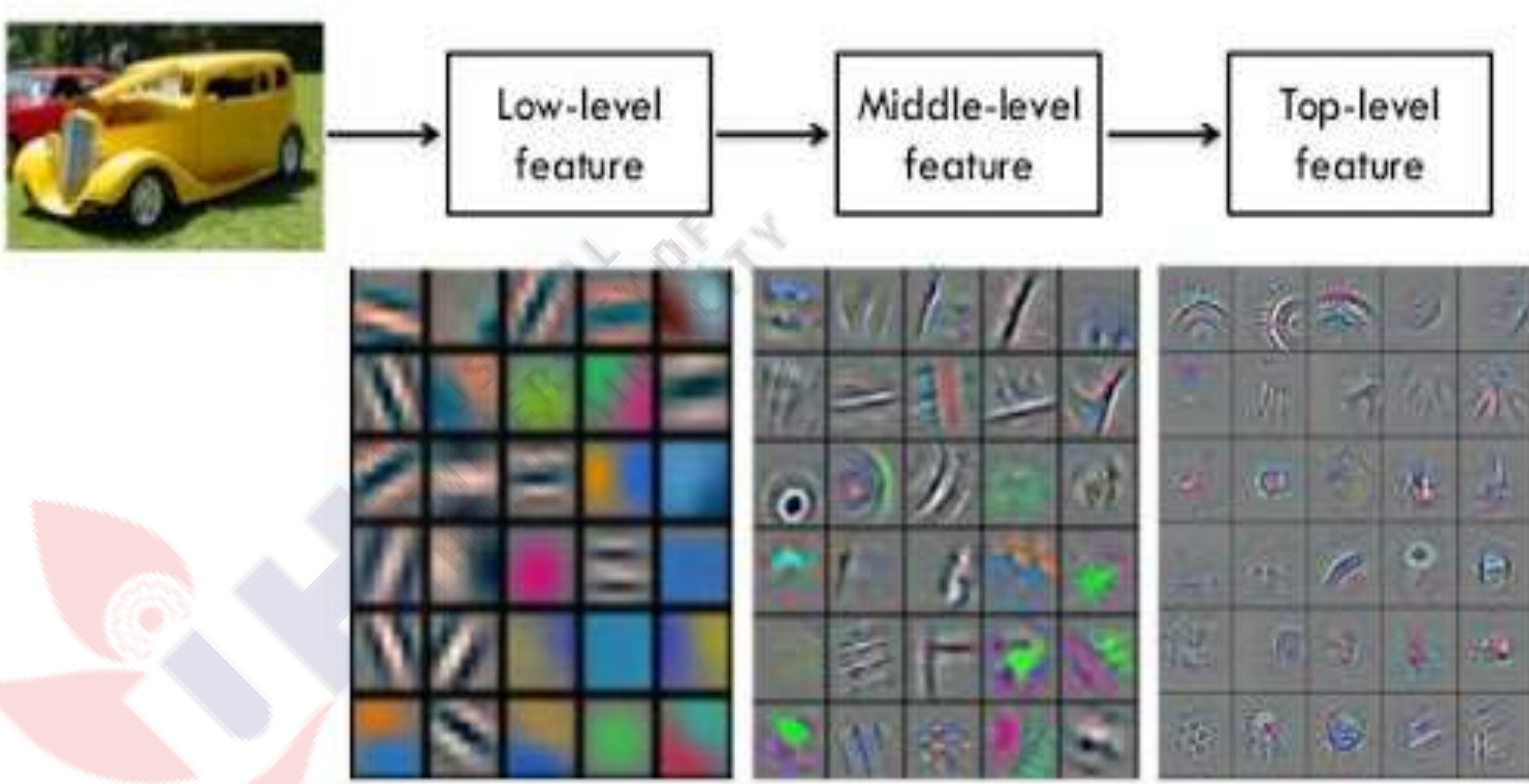
- NLP


Activation function
Convolution
1-max pooling
Softmax
Sentence Matrix 6x5
3 region sizes (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
You should listen to your dad

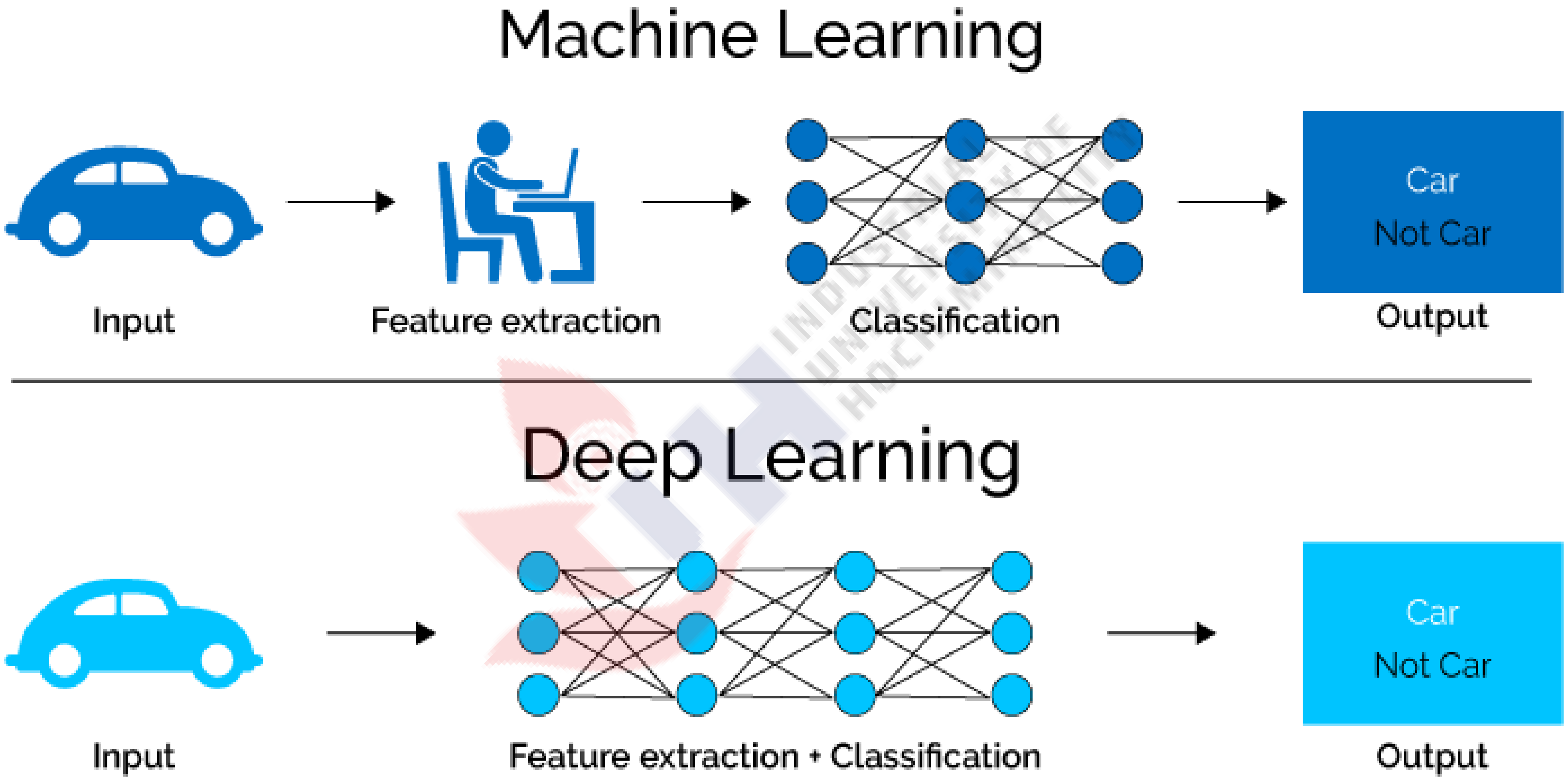
Computer Vision

- Feature extraction





Feature extraction



Machine Learning

Input

Feature extraction

Classification

Car

Not Car

Output

Deep Learning

Input

Feature extraction + Classification

Car

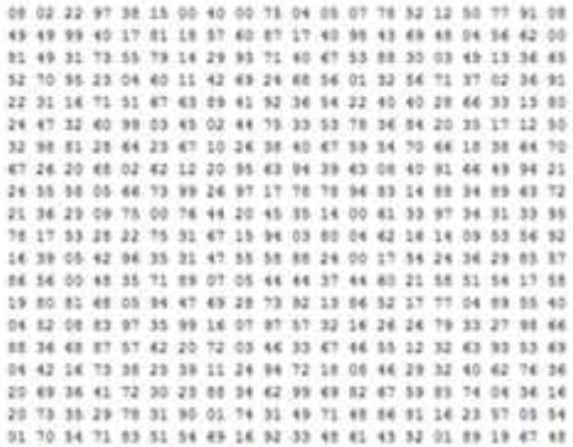
Not Car

Output

The Problem Space

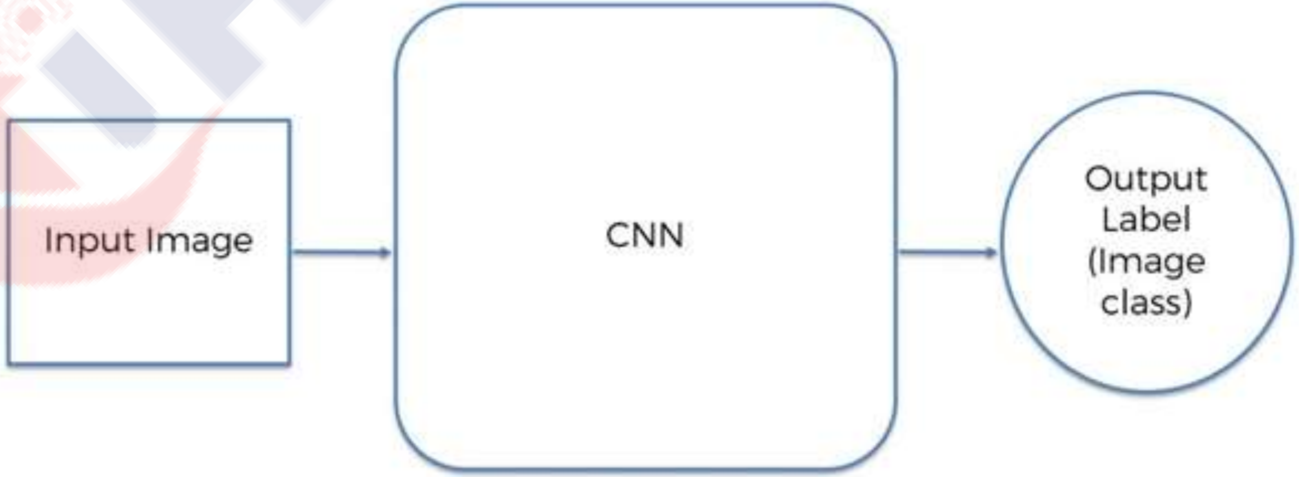
Image: 480 x 480 x 3
Pixel : 0 to 255
Problem when 20.000 cats, 40.000 dogs, 50.000 birds???

What We See



08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	91	08
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	24	68	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	63	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	32	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
88	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	25	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	62	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	86	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	19	67	48

What Computers See



Xf= 12288, số lượng nodes trong layer 1= 1000

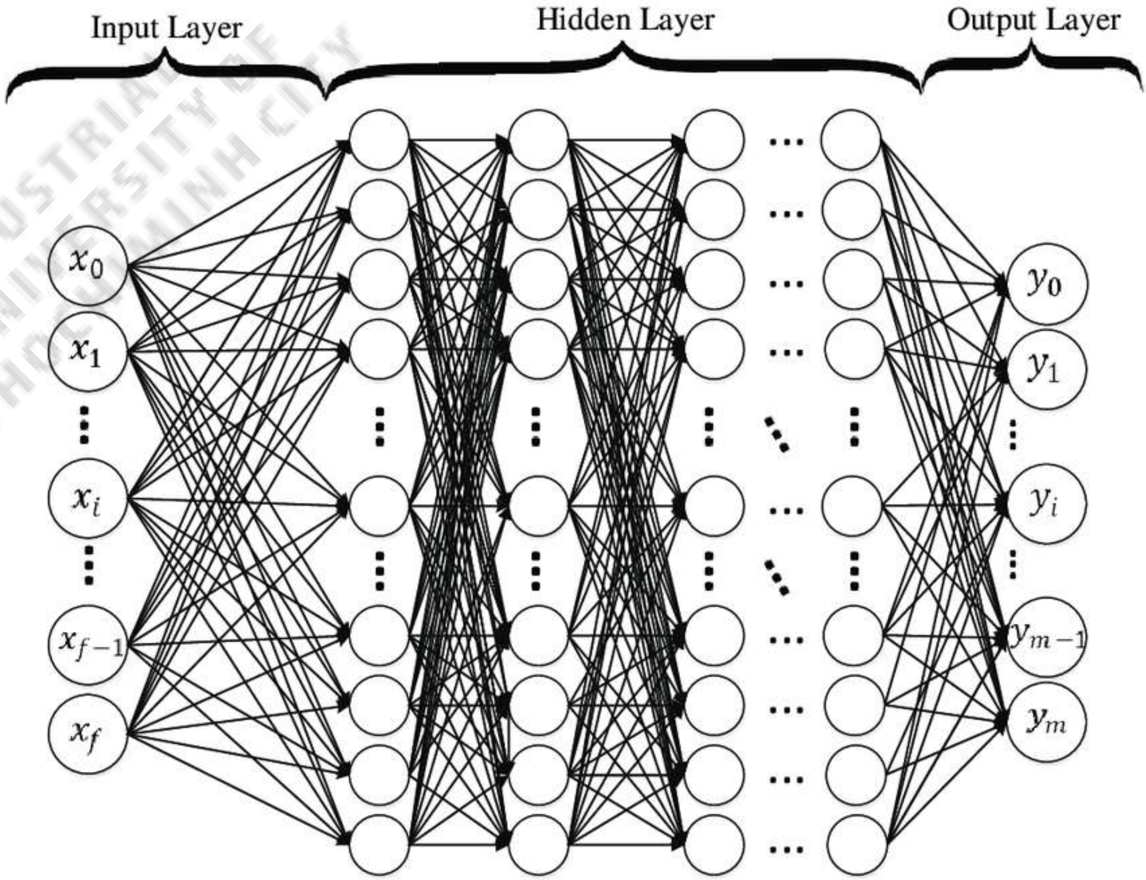
=>Số lượng weight = 12288x1000=12288000

Số lượng bias= 12289000

Layer 2,…layer n???

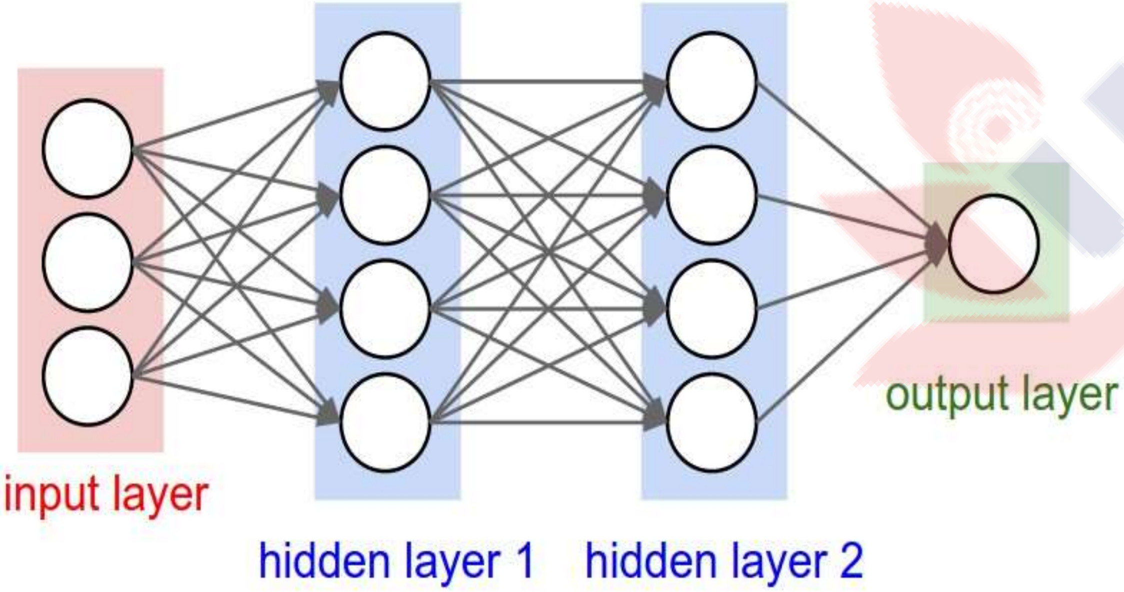
⇒*Số lượng quá lớn, tính toán chậm, cần giải pháp tốt hơn!*

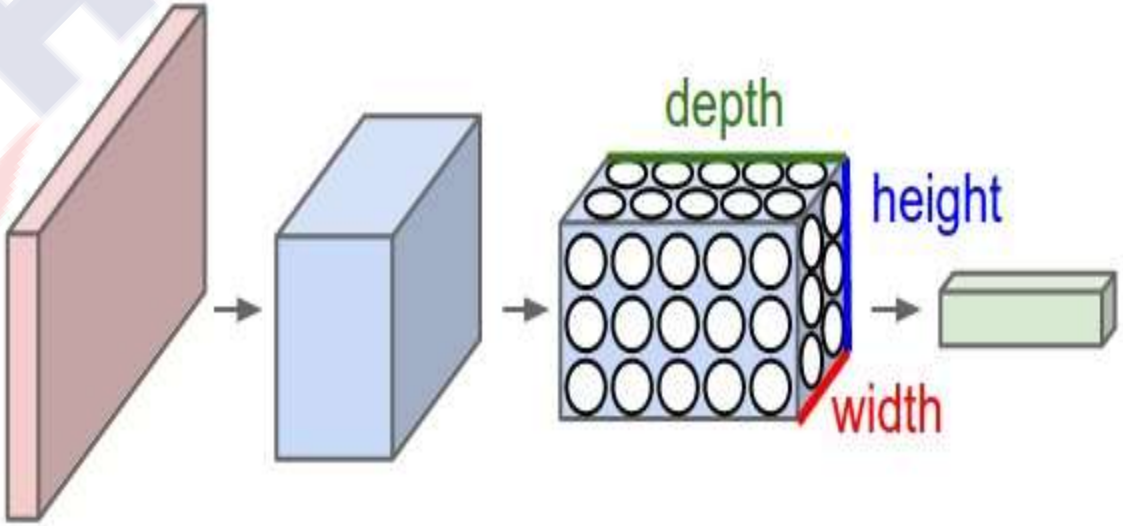
Áp dụng phép tính convolution vào các layer trong network giải quyết được vấn đề lượng lớn parameter mà vẫn lấy ra được các đặc trưng của ảnh.



Convolutional Neural Network

Neural Networks receive an input (a single vector)
3D volumes of neurons: the layers of a ConvNet have neurons arranged in 3 dimensions: **width, height, depth**

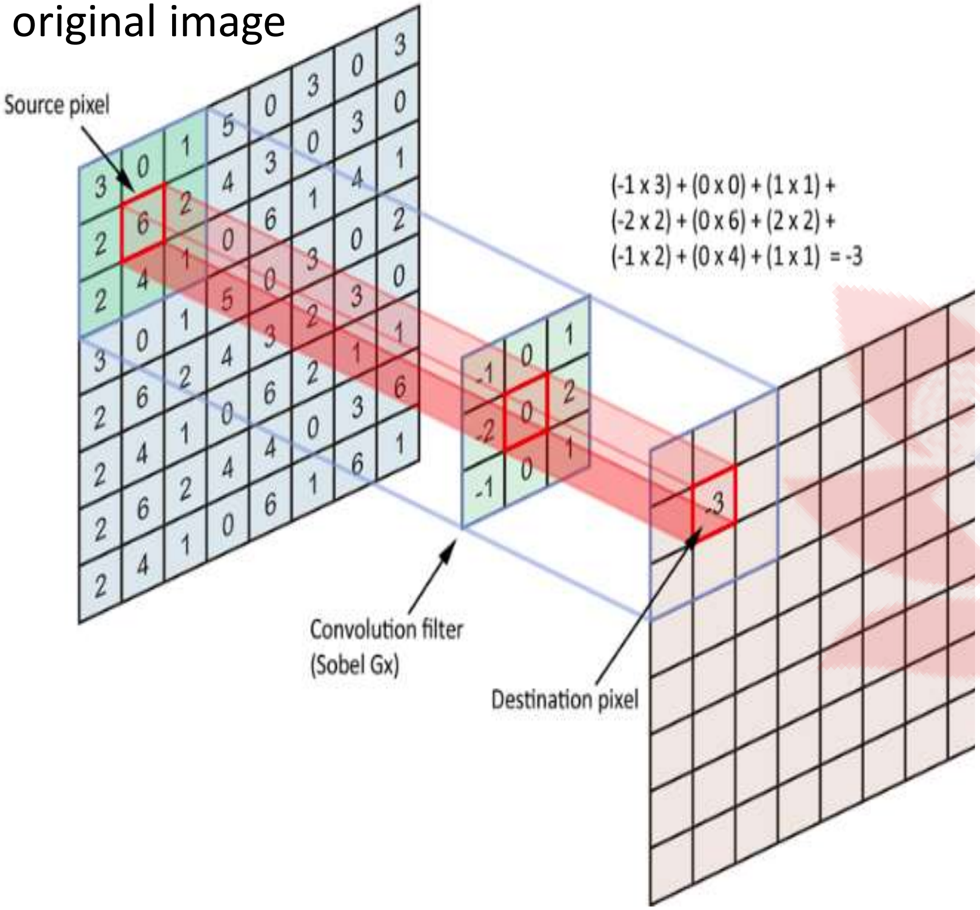


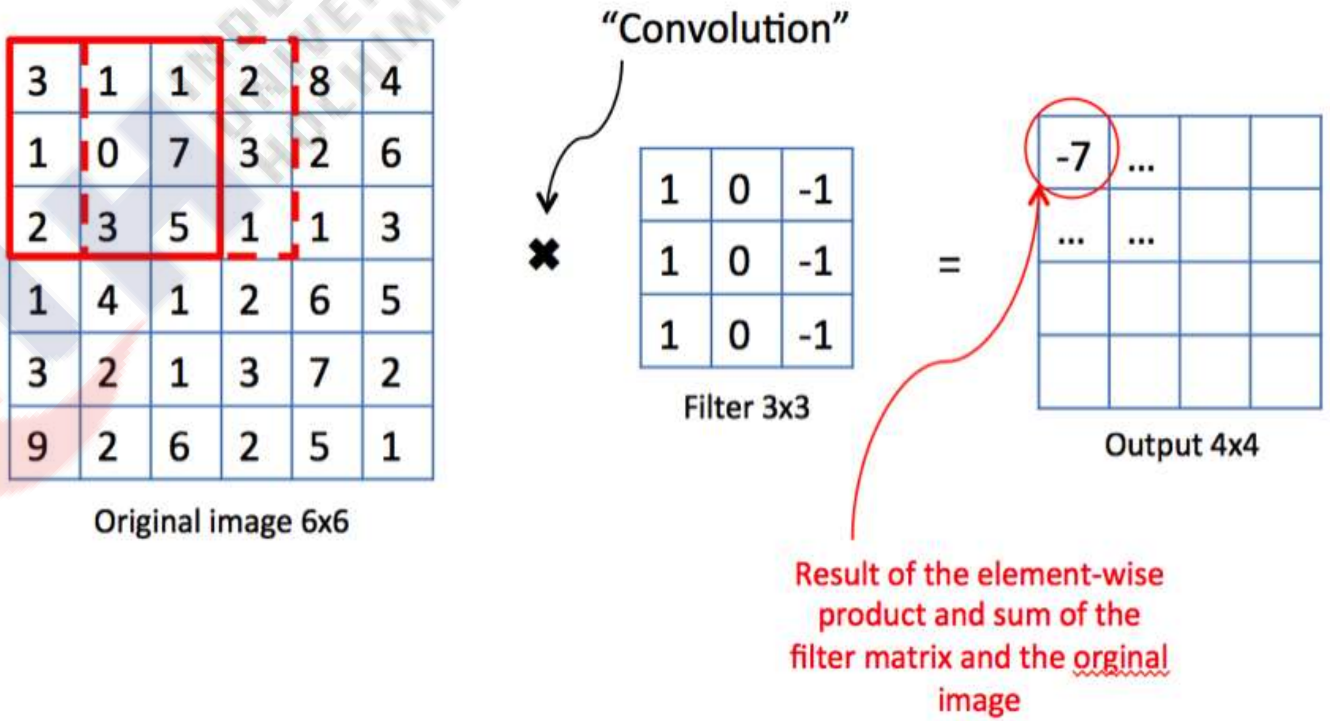


Convolutional Layer

A “*convolution*” is one of the building blocks of the Convolutional network.

2D original image


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel



3	1	1	2	8	4
1	0	7	3	2	6
2	3	5	1	1	3
1	4	1	2	6	5
3	2	1	3	7	2
9	2	6	2	5	1

Original image 6x6

“Convolution”

✖

1	0	-1
1	0	-1
1	0	-1

Filter 3x3

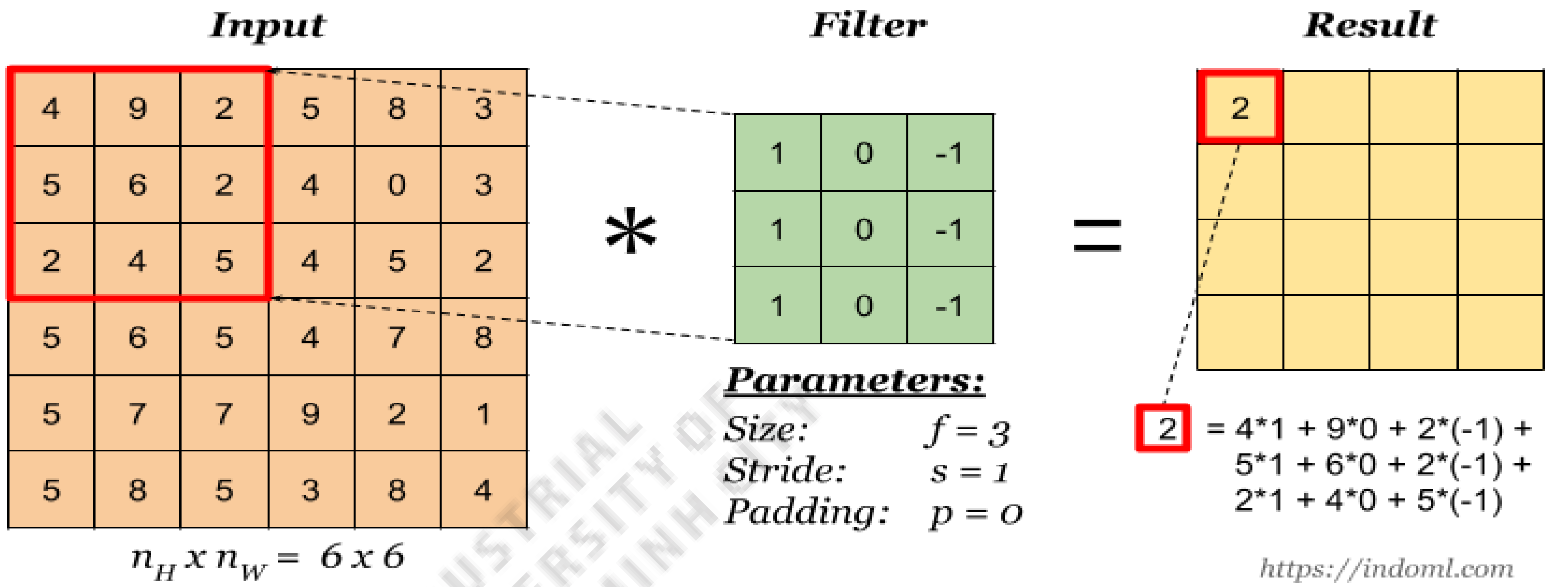
=

-7	...		
...	...		

Output 4x4

Result of the element-wise product and sum of the filter matrix and the orginal image

Step 1:



4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

$n_H \, x \, n_W = \; 6 \, x \, 6$

1	0	-1
1	0	-1
1	0	-1

Parameters:

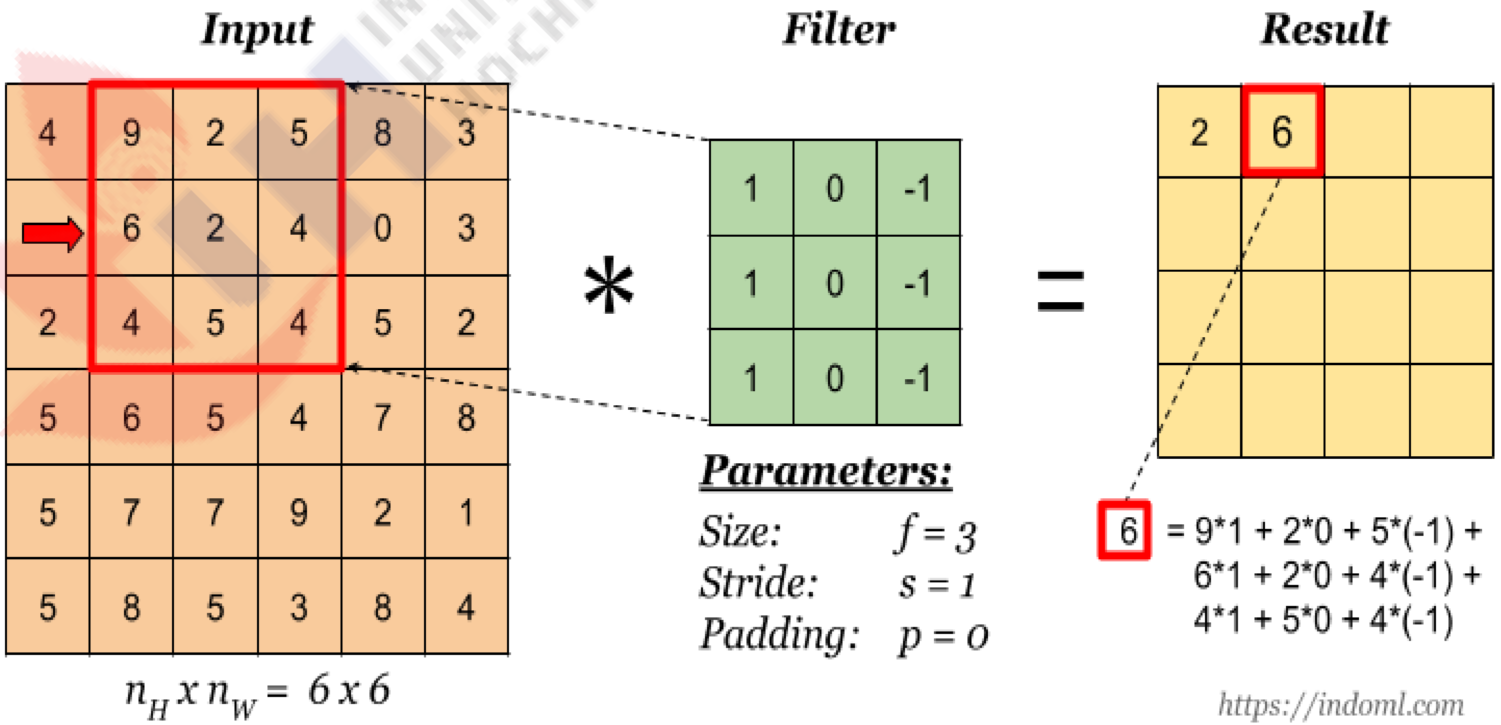
Size: $f = 3$
Stride: $s = 1$
Padding: $p = 0$

2			

2 = 4*1 + 9*0 + 2*(-1) +
5*1 + 6*0 + 2*(-1) +
2*1 + 4*0 + 5*(-1)

https://indoml.com

Step 2:



4	9	2	5	8	3
	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

$n_H \, x \, n_W = \; 6 \, x \, 6$

1	0	-1
1	0	-1
1	0	-1

Parameters:

Size: $f = 3$
Stride: $s = 1$
Padding: $p = 0$

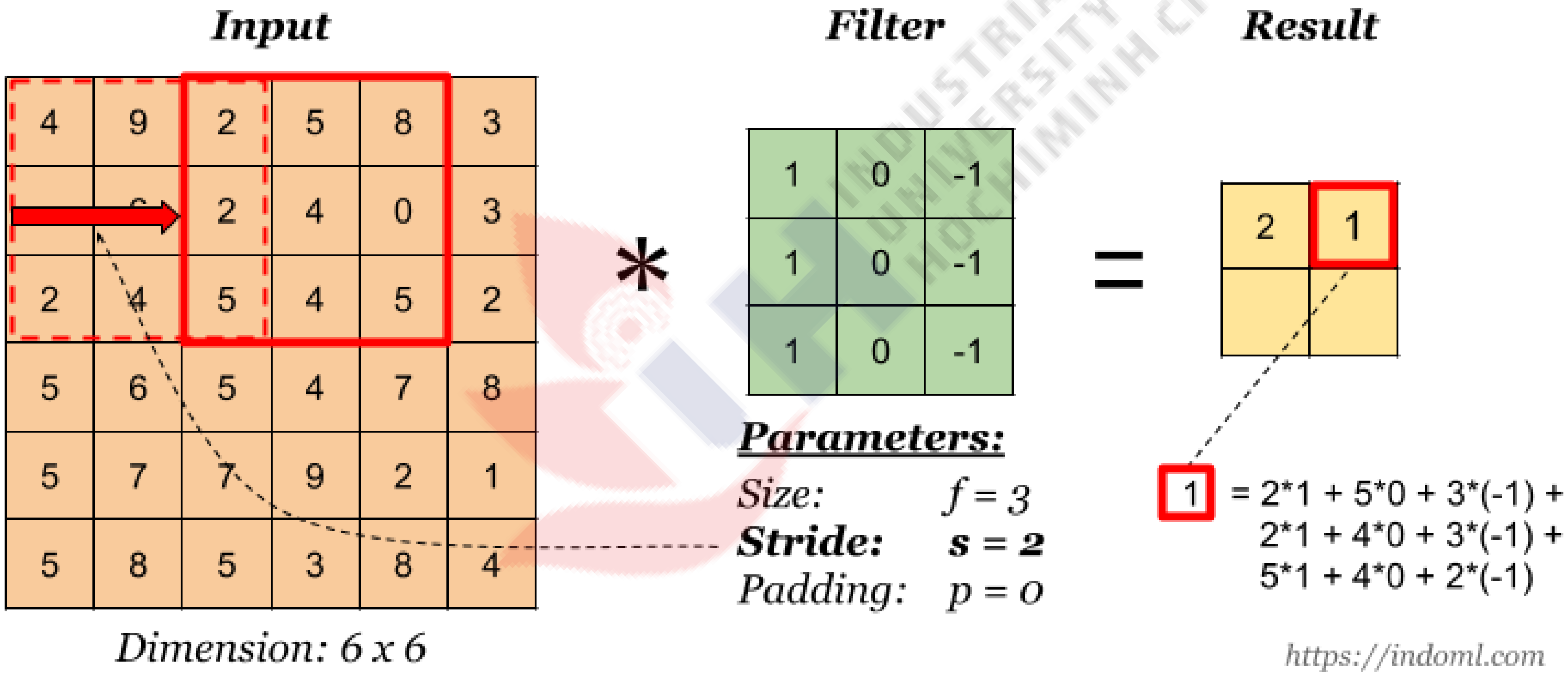
2	6		

6 = 9*1 + 2*0 + 5*(-1) +
6*1 + 2*0 + 4*(-1) +
4*1 + 5*0 + 4*(-1)

https://indoml.com

Stride

- Stride governs how many cells the filter is moved in the input to calculate the next cell in the result.



Input

4	9	2	5	8	3
[illegible]	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

Dimension: 6 x 6

*

Filter

1	0	-1
1	0	-1
1	0	-1

=

Result

2	1

Parameters:

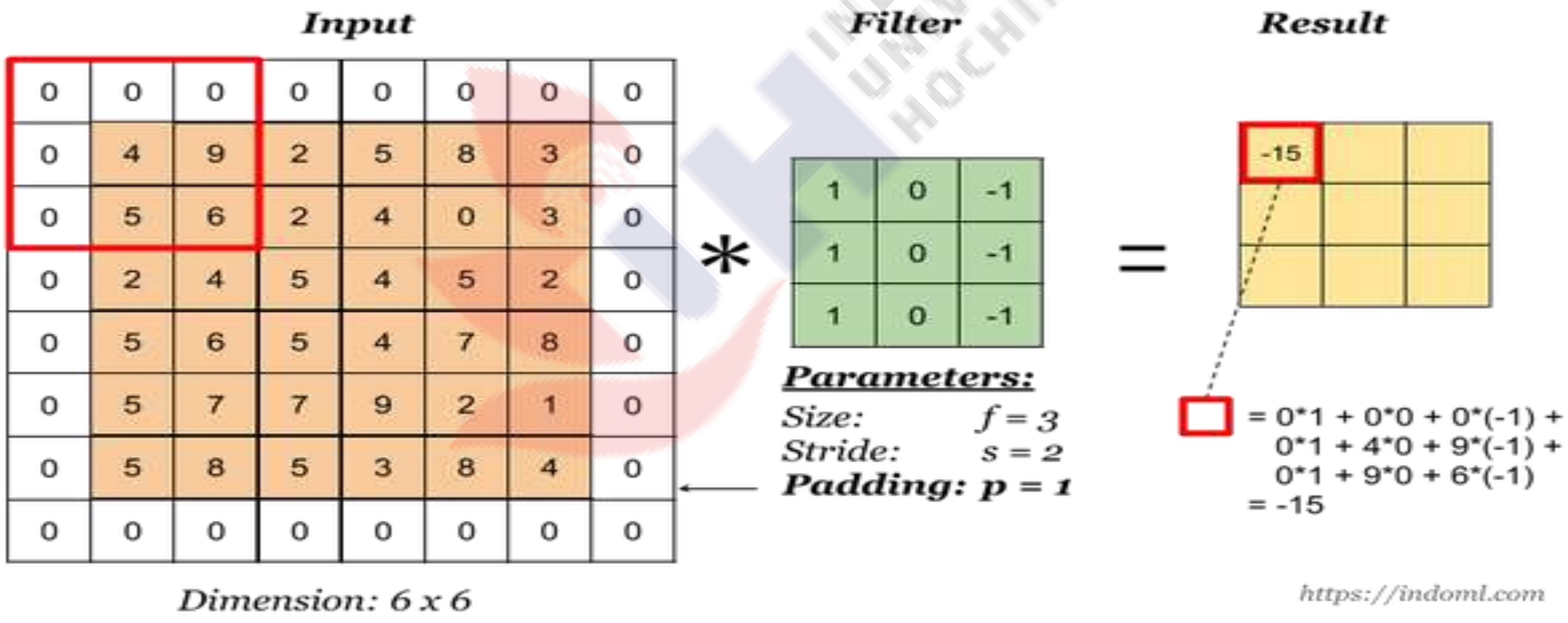
Size:	$f = 3$
Stride:	$\mathbf{s = 2}$
Padding:	$p = 0$

$\boxed{1}$ = 2*1 + 5*0 + 3*(-1) +
2*1 + 4*0 + 3*(-1) +
5*1 + 4*0 + 2*(-1)

https://indoml.com

Padding

- important for building deeper networks
- keep more of the information at the border of an image, Without padding, very few values at the next layer would be affected by pixels as the edges of an image.



0	0	0	0	0	0	0	0
0	4	9	2	5	8	3	0
0	5	6	2	4	0	3	0
0	2	4	5	4	5	2	0
0	5	6	5	4	7	8	0
0	5	7	7	9	2	1	0
0	5	8	5	3	8	4	0
0	0	0	0	0	0	0	0

Dimension: 6 x 6

1	0	-1
1	0	-1
1	0	-1

Parameters:

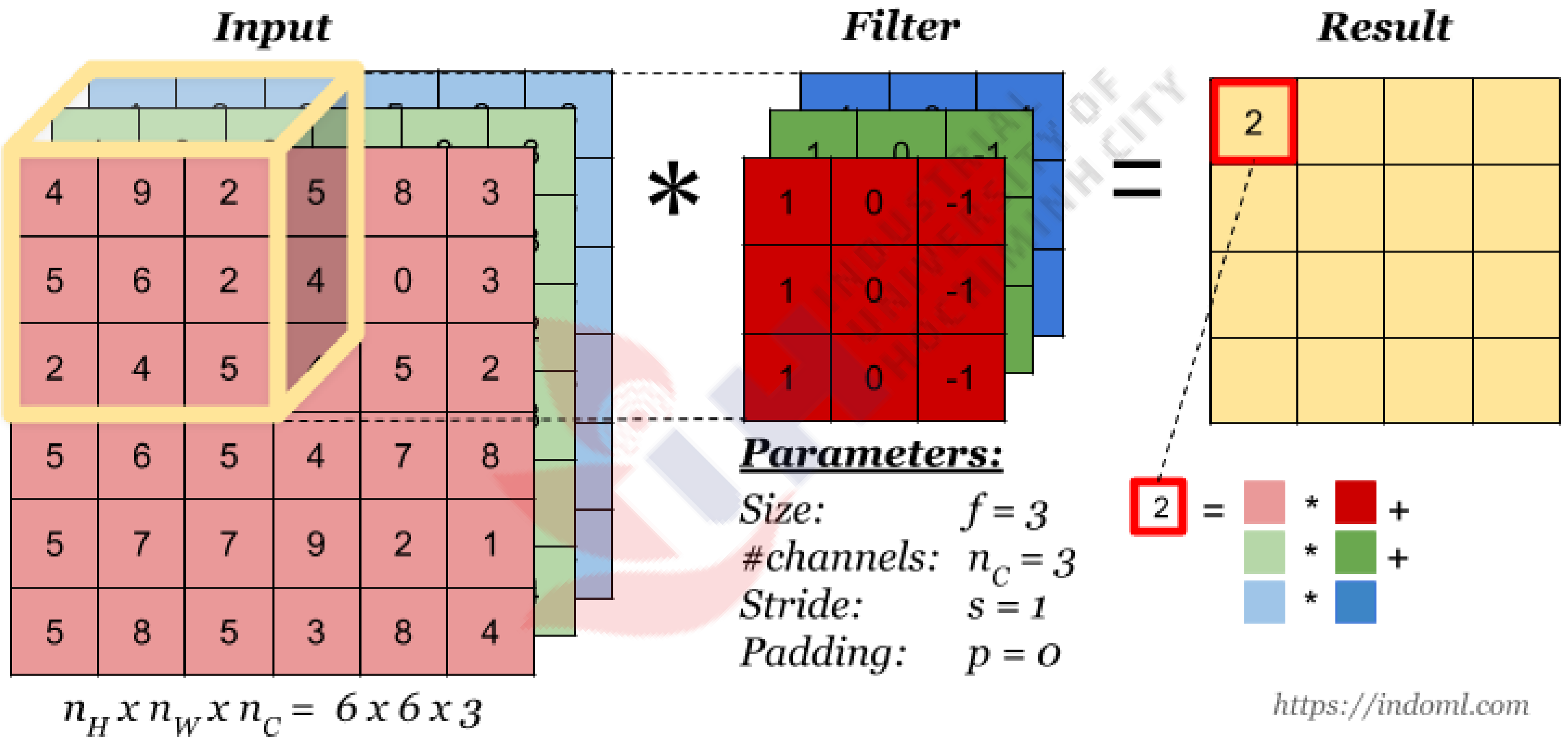
Size: $f = 3$
Stride: $s = 2$
Padding: $p = 1$

-15		

= 0*1 + 0*0 + 0*(-1) +
0*1 + 4*0 + 9*(-1) +
0*1 + 9*0 + 6*(-1)
= -15

https://indoml.com

Convolution Operation on Volume



4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

1	0	-1
1	0	-1
1	0	-1

Parameters:

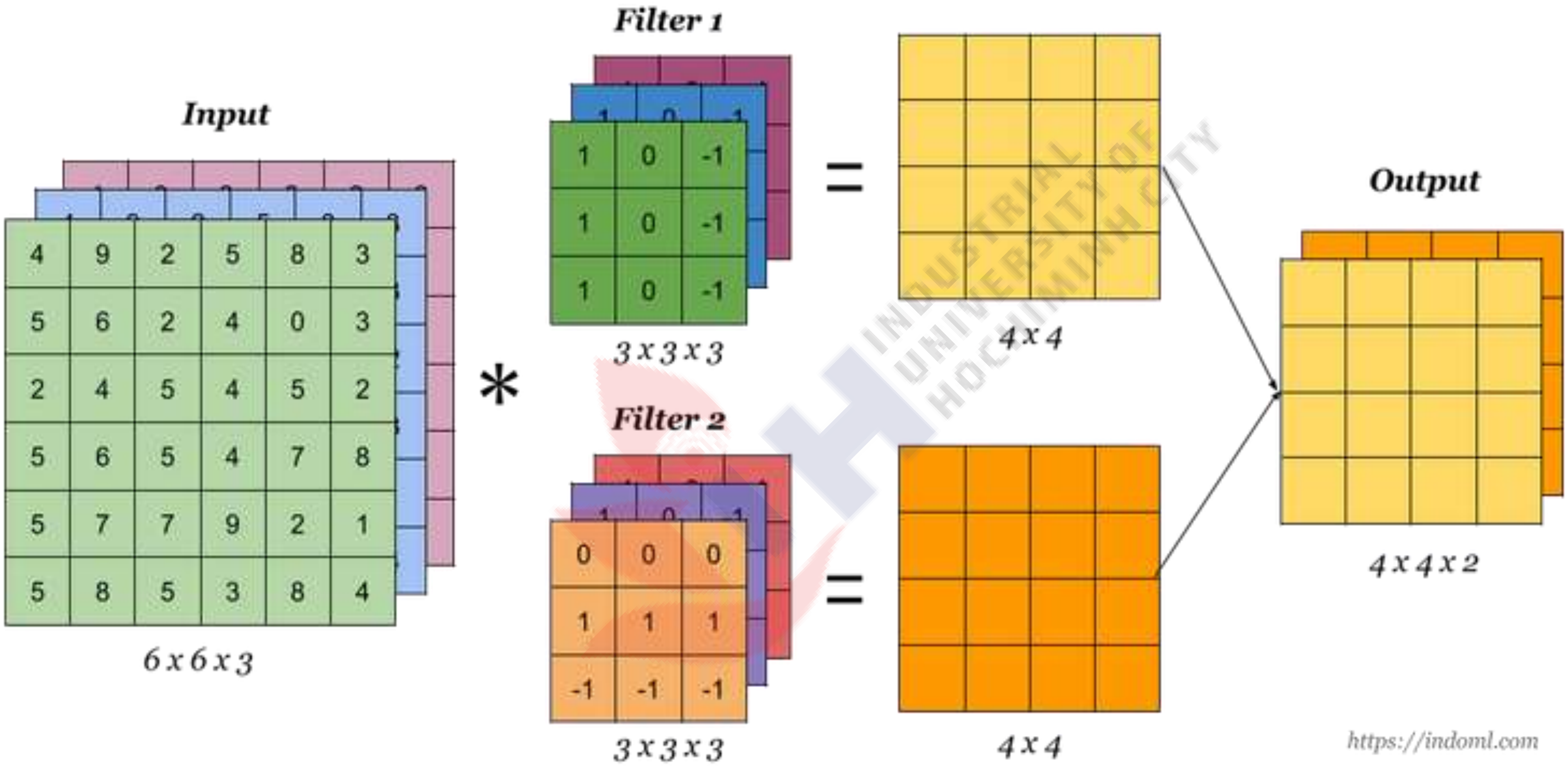
Size: $f = 3$

#channels: $n_C = 3$

Stride: $s = 1$

Padding: $p = 0$

Convolution Operation with Multiple Filters



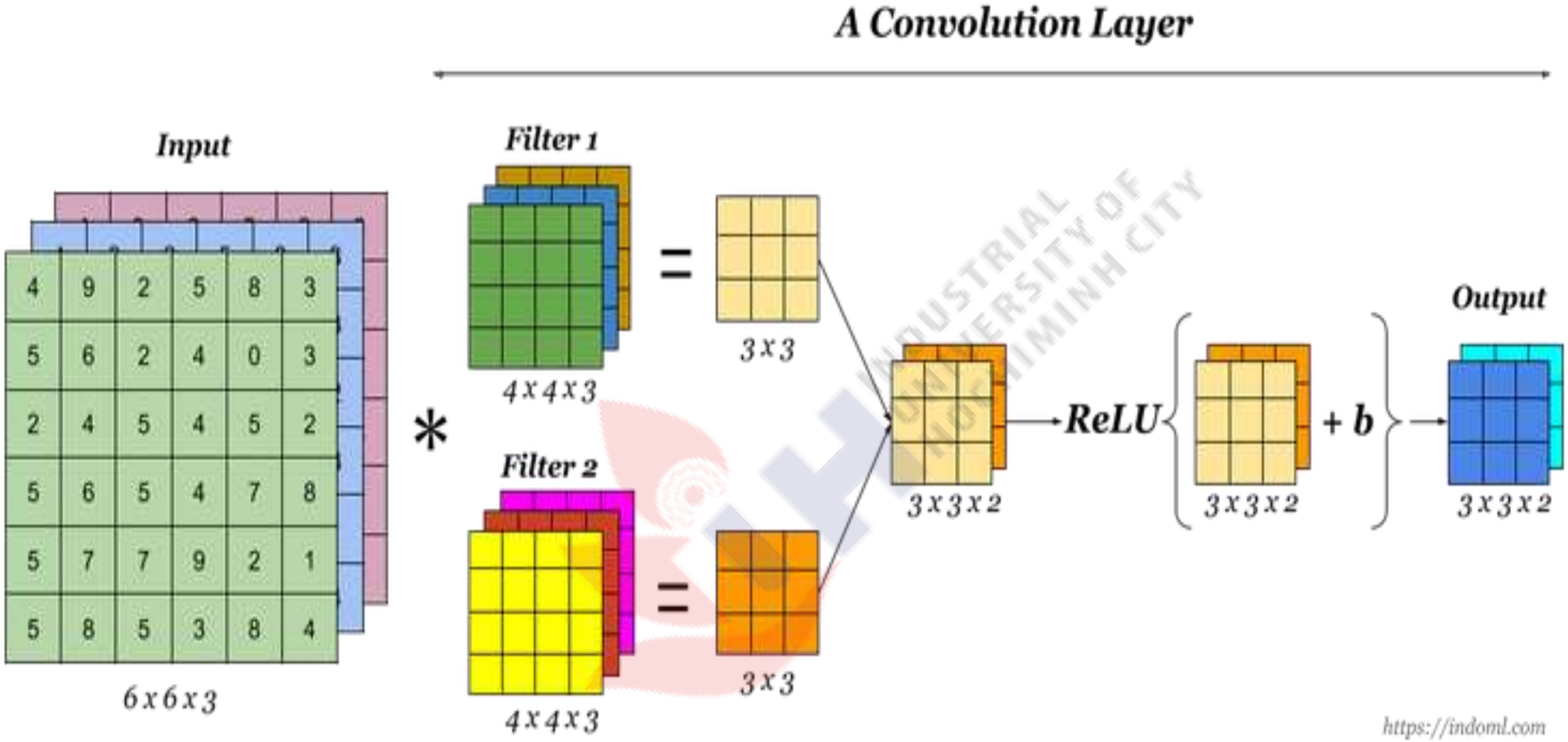
4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

1	0	-1
1	0	-1
1	0	-1

0	0	0
1	1	1
-1	-1	-1

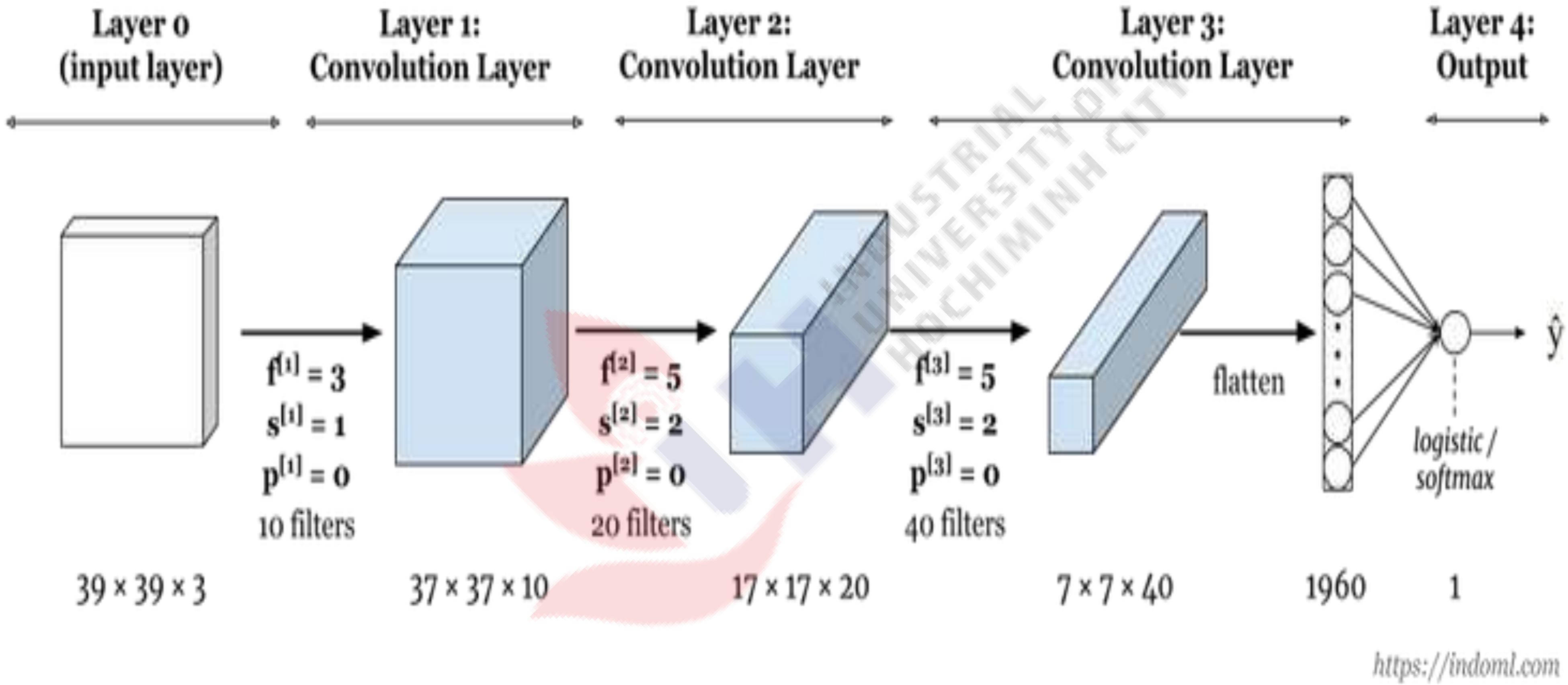
https://indoml.com

One Convolution Layer

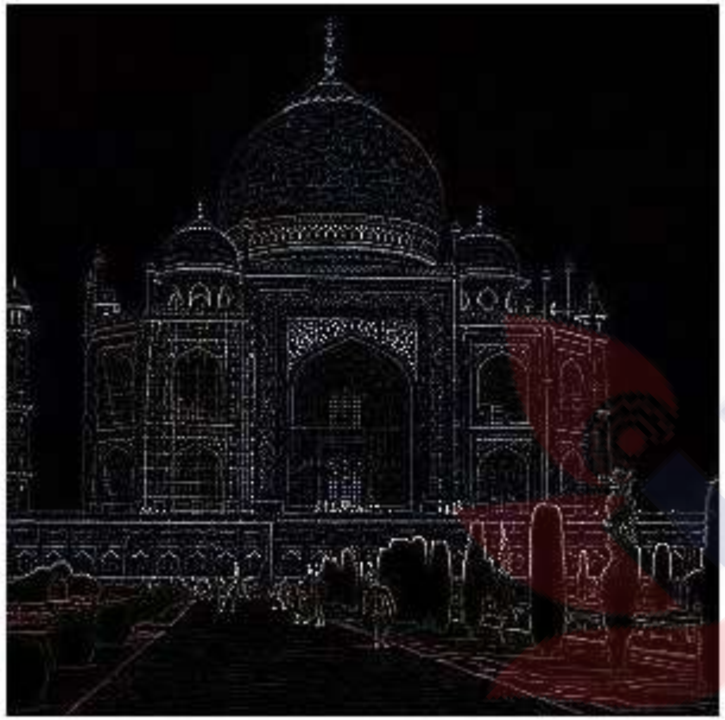


4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

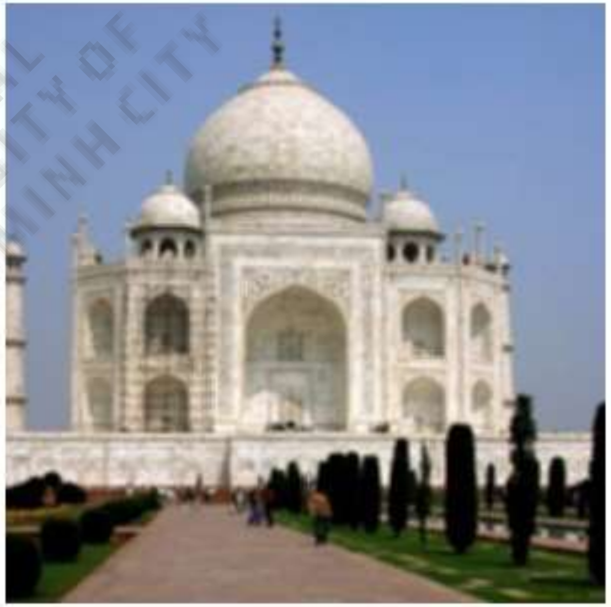
Sample Complete Network



	0	1	0	
	1	-4	1	
	0	1	0	

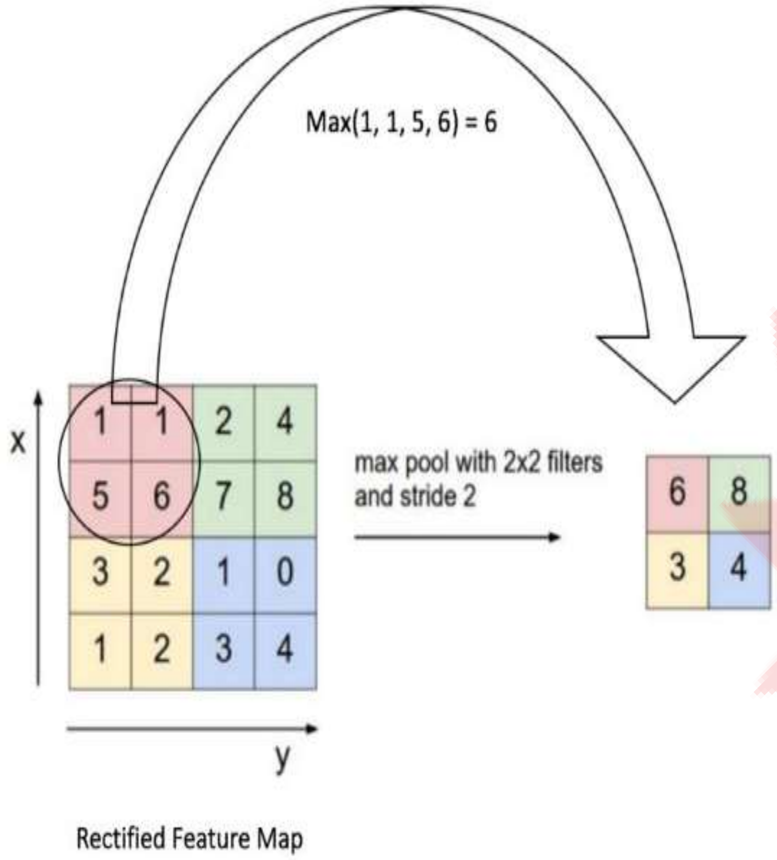


0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0



Pooling Layer

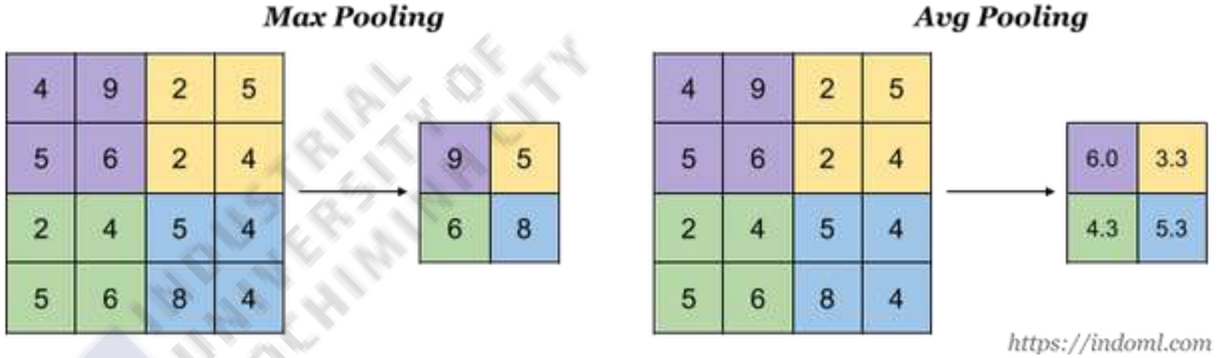
- Pooling layer is used to reduce the size of the representations and to speed up calculations



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Rectified Feature Map



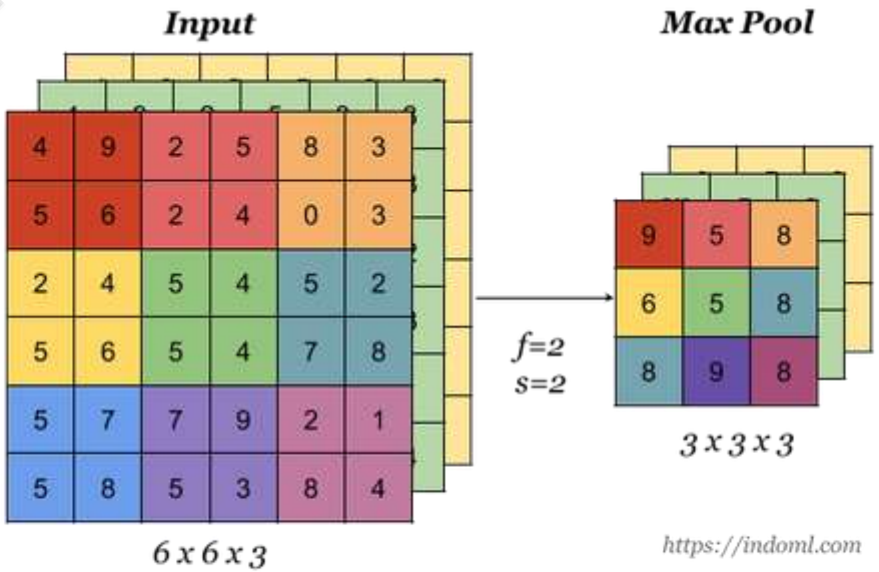
4	9	2	5
5	6	2	4
2	4	5	4
5	6	8	4

9	5
6	8

4	9	2	5
5	6	2	4
2	4	5	4
5	6	8	4

6.0	3.3
4.3	5.3

https://indoml.com



4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

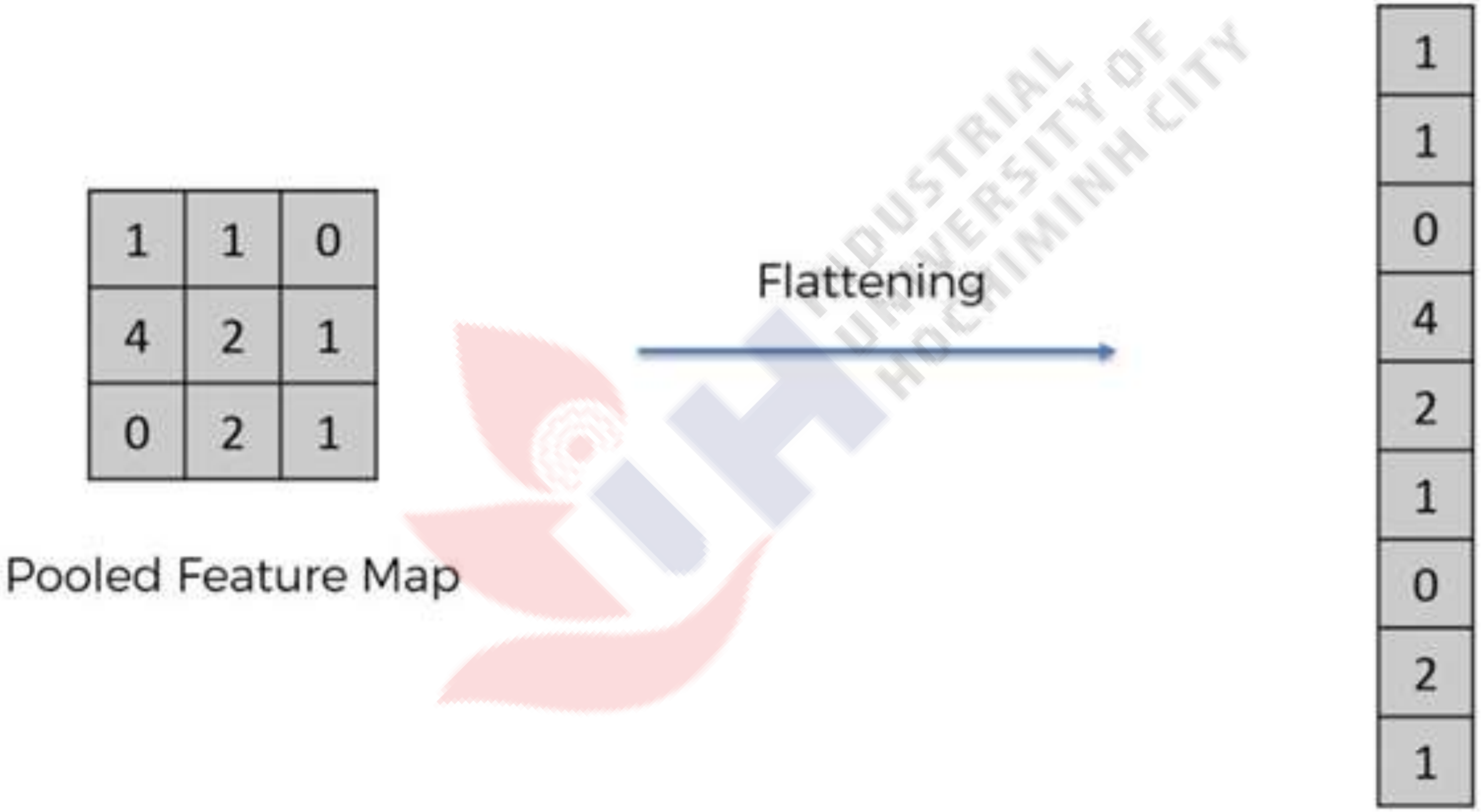
9	5	8
6	5	8
8	9	8

6 x 6 x 3

3 x 3 x 3

https://indoml.com

Flattening



1	1	0
4	2	1
0	2	1

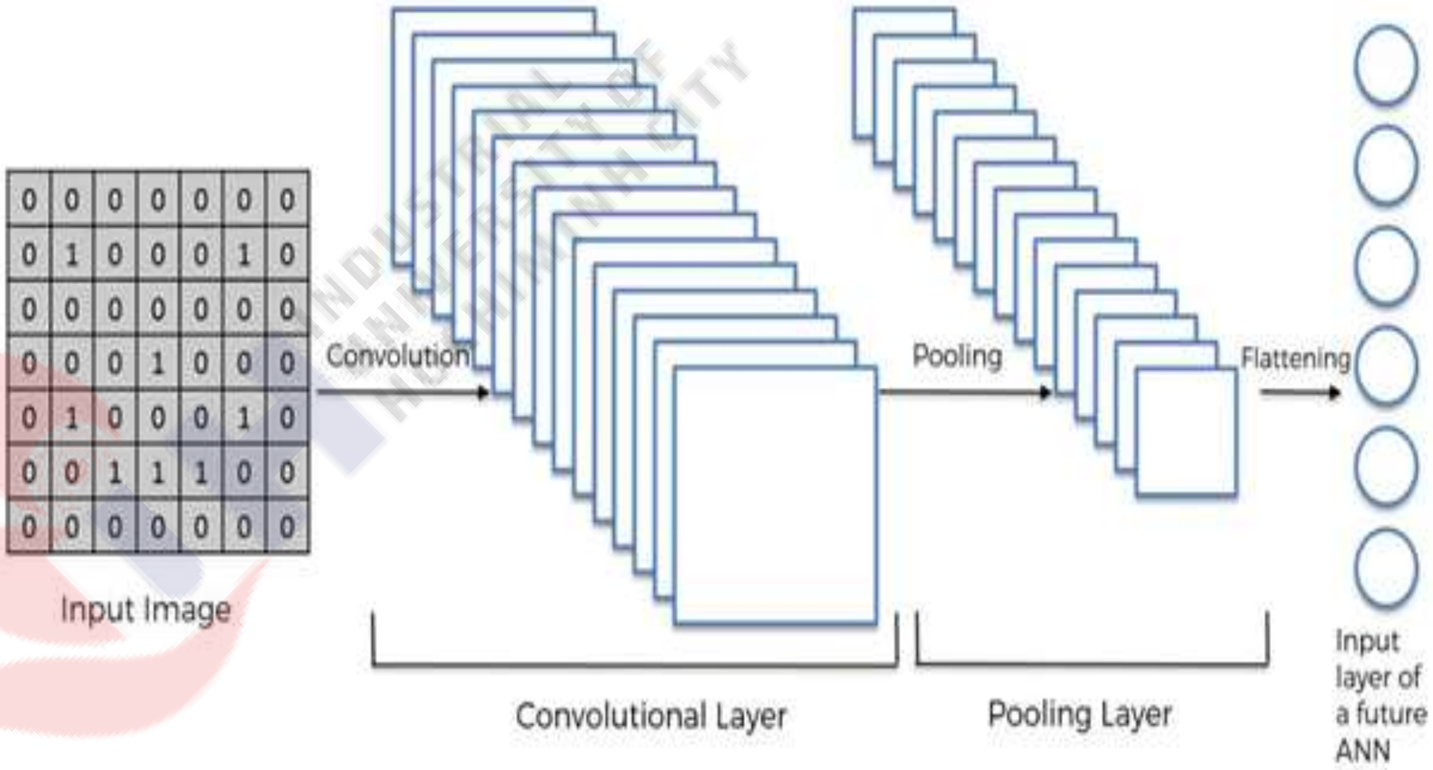
Pooled Feature Map

Flattening

1
1
0
4
2
1
0
2
1

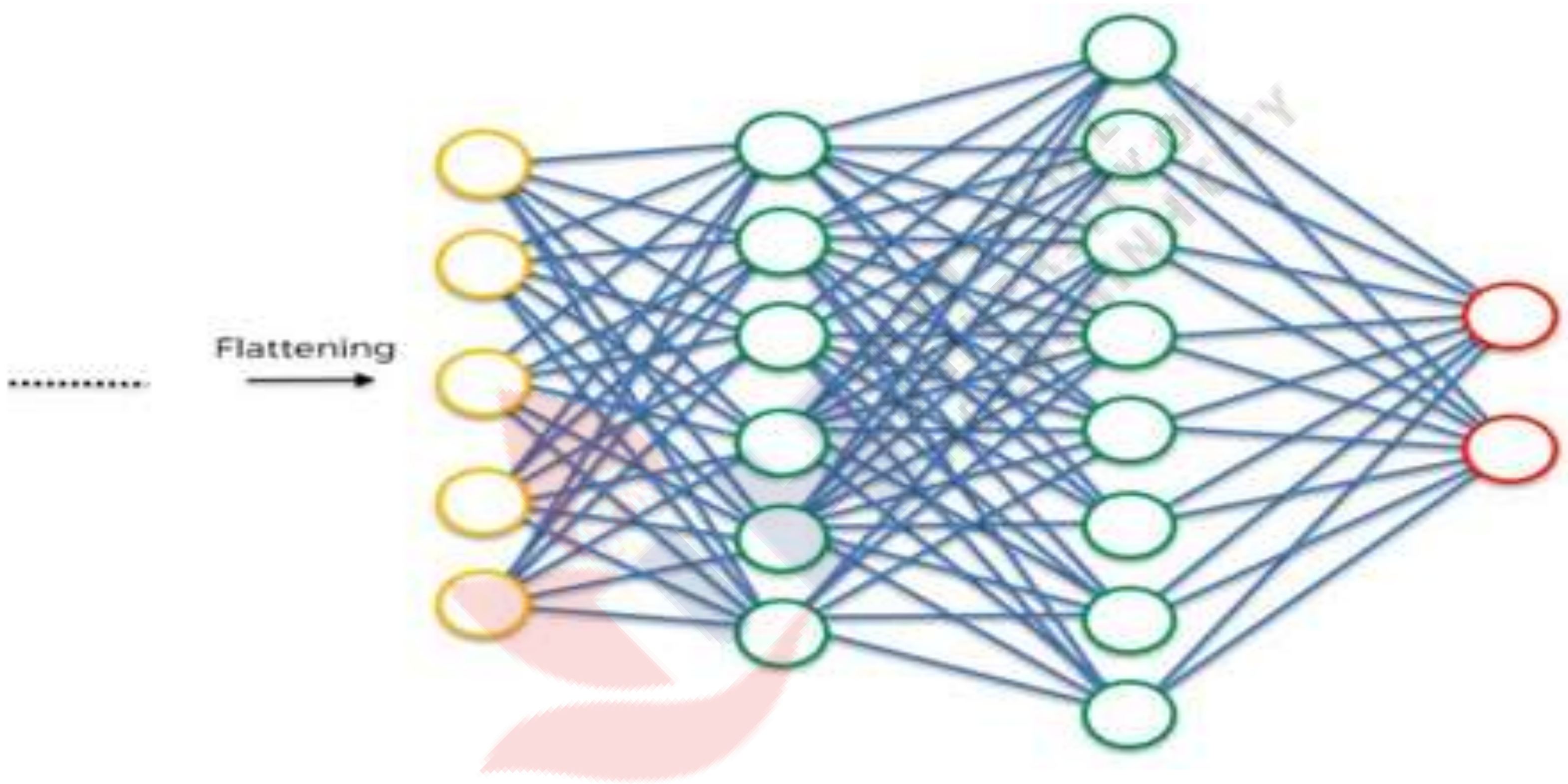
The process of building a CNN involves four major steps

- Convolution
- Pooling
- Flattening
- Full Connection

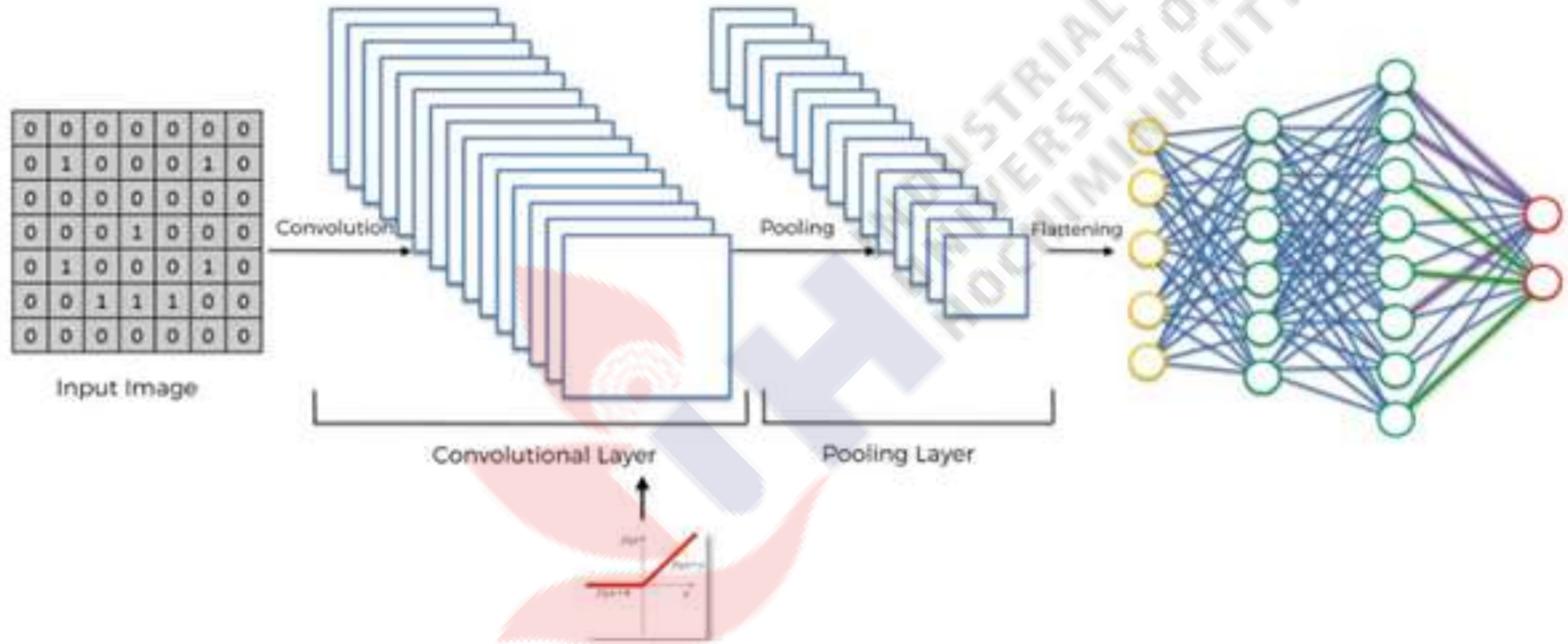


0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Full Connection

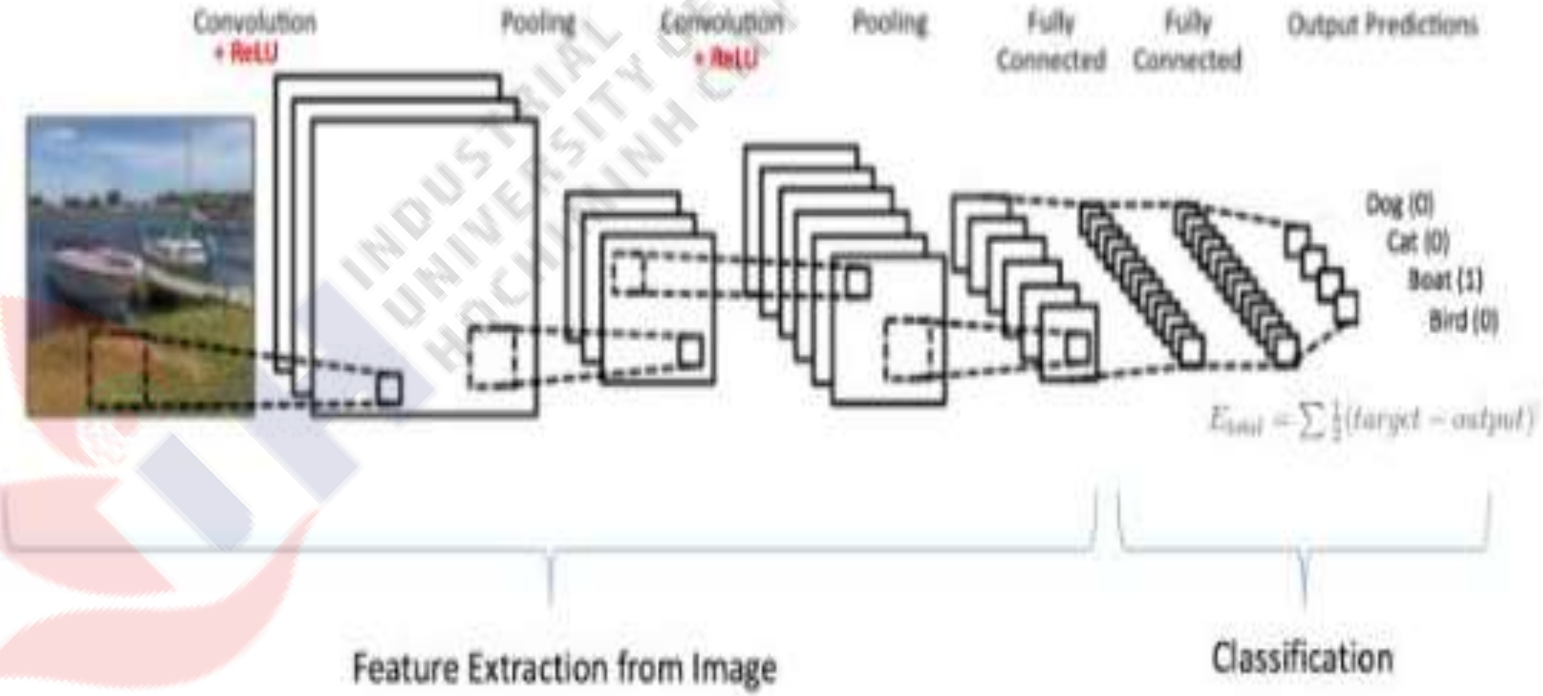


- **Putting it all together Training using Backpropagation**



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 = Boat
- Target Vector = [0, 0, 1, 0]



Training process

Step1: initialize all filters and parameters/weights with random values.

Step2: takes a training image as input => through the forward propagation step => finds the output probabilities for each class

Ex: output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3]

Step3: Calculate the total error at the output layer (summation over all 4 classes)

Total Error = $\sum \frac{1}{2}$ (target probability — output probability) ^{2}

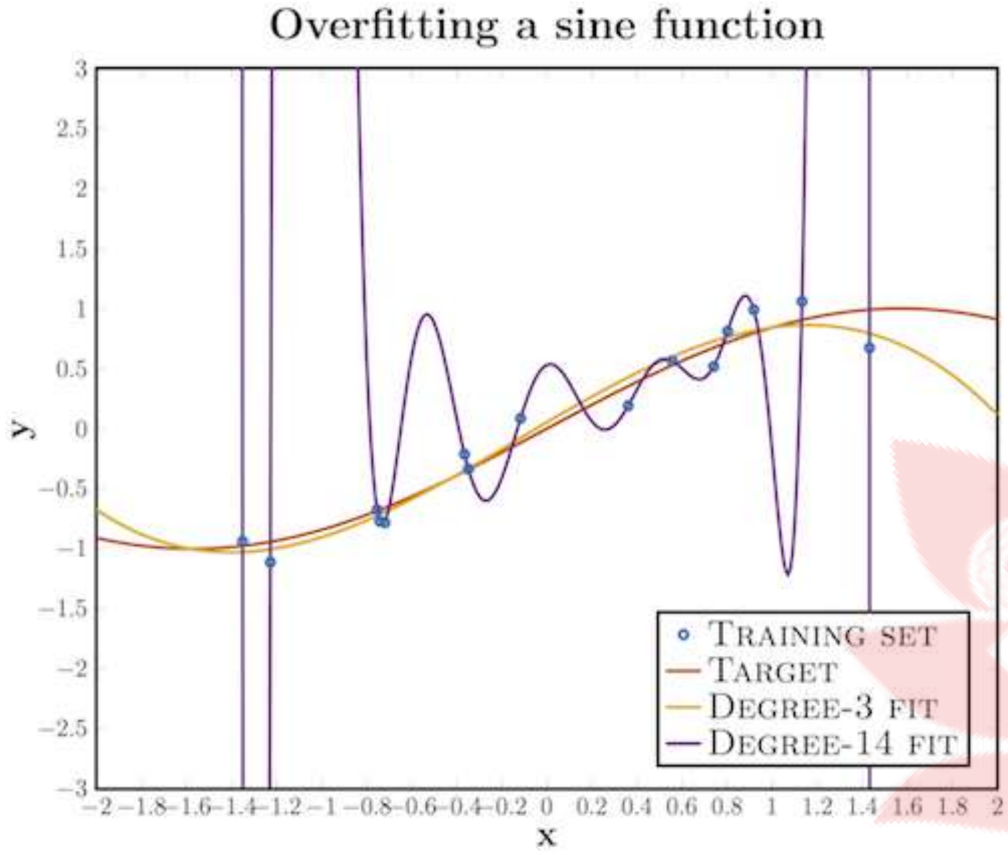
Step4: Use Backpropagation to calculate the *gradients* of the error concerning all weights in the network and use *gradient descent* to update all filter values/weights and parameter values to minimize the output error.

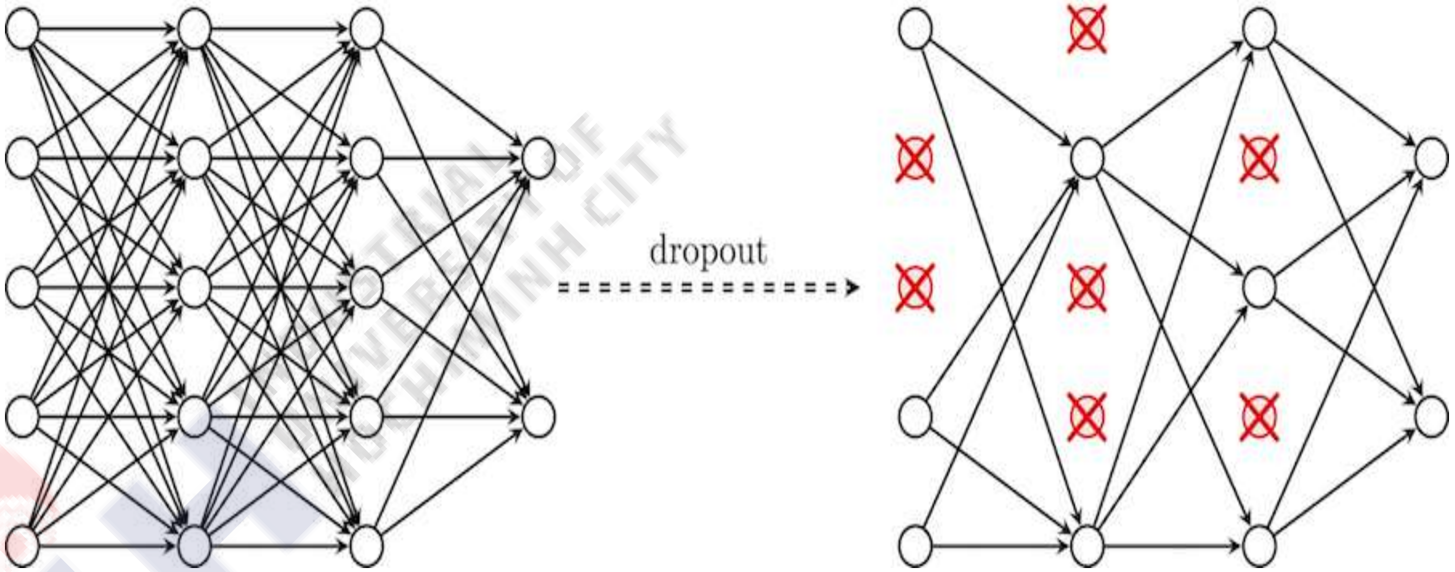
Ex: [0.1, 0.1, 0.7, 0.1] is closer to the target vector [0, 0, 1, 0].

Parameters like the number of filters, filter sizes, the architecture of the network, etc. have all been fixed before Step 1 and do not change during the training process — only the values of the filter matrix and connection weights get updated.

Step5: Repeat steps 2–4 with all images in the training set.

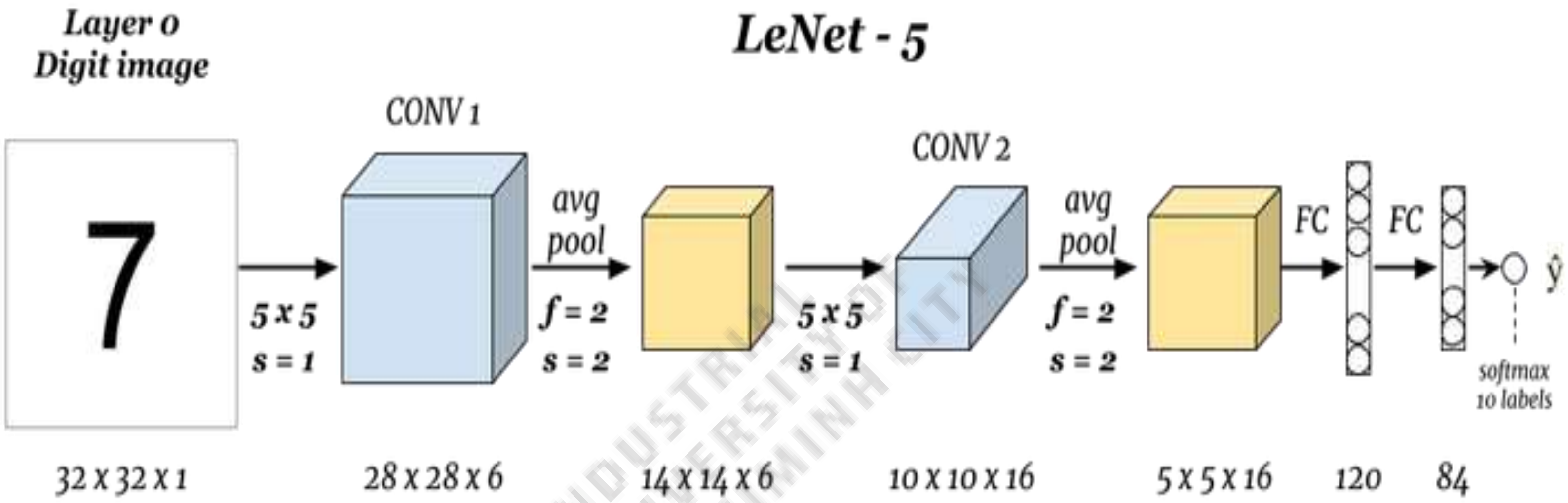
Overfitting and dropout

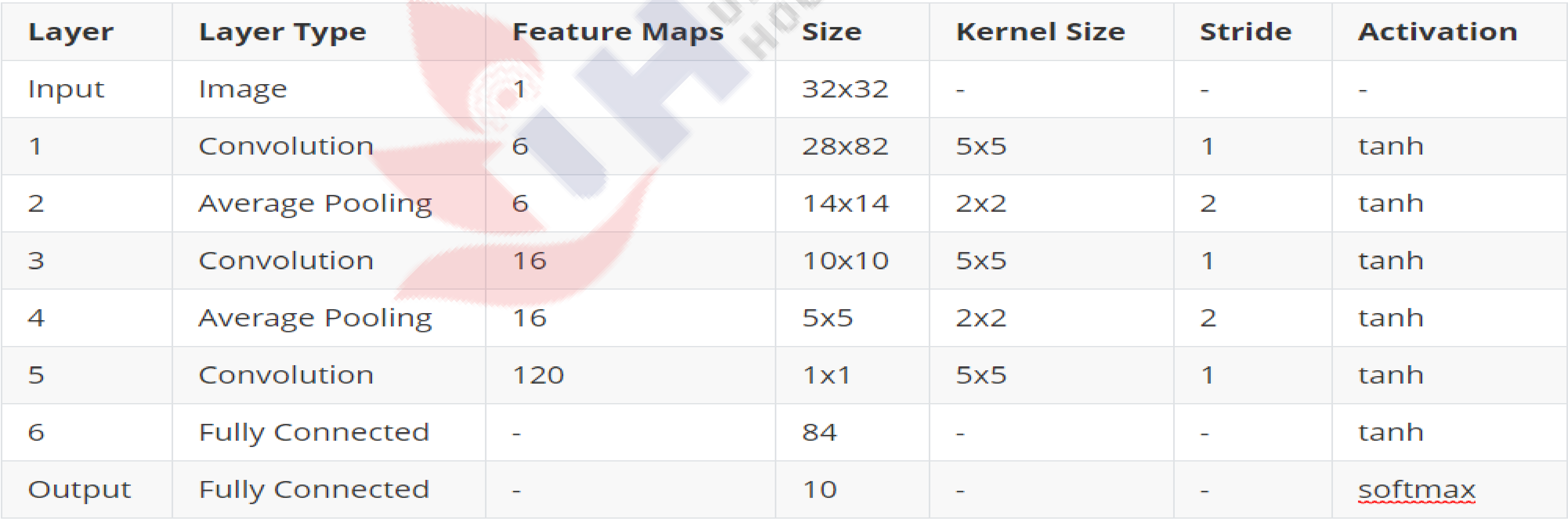




Well Known Architectures

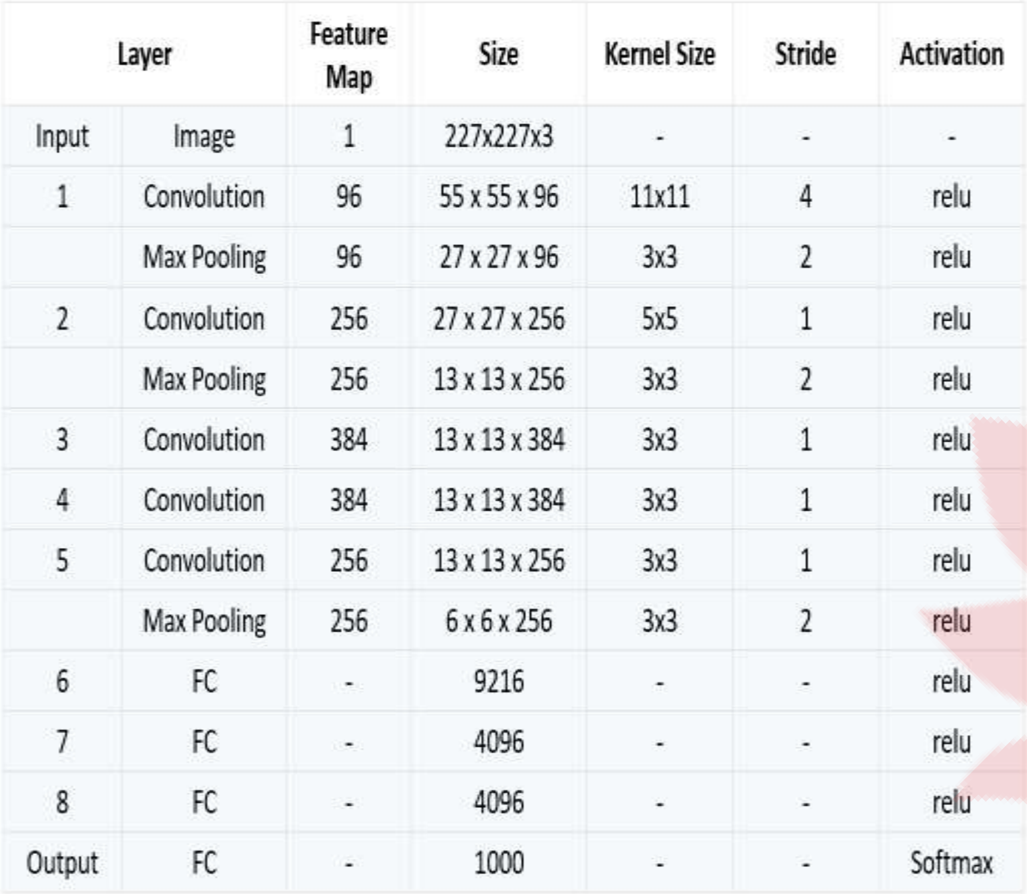
- **LeNet – 5**
-



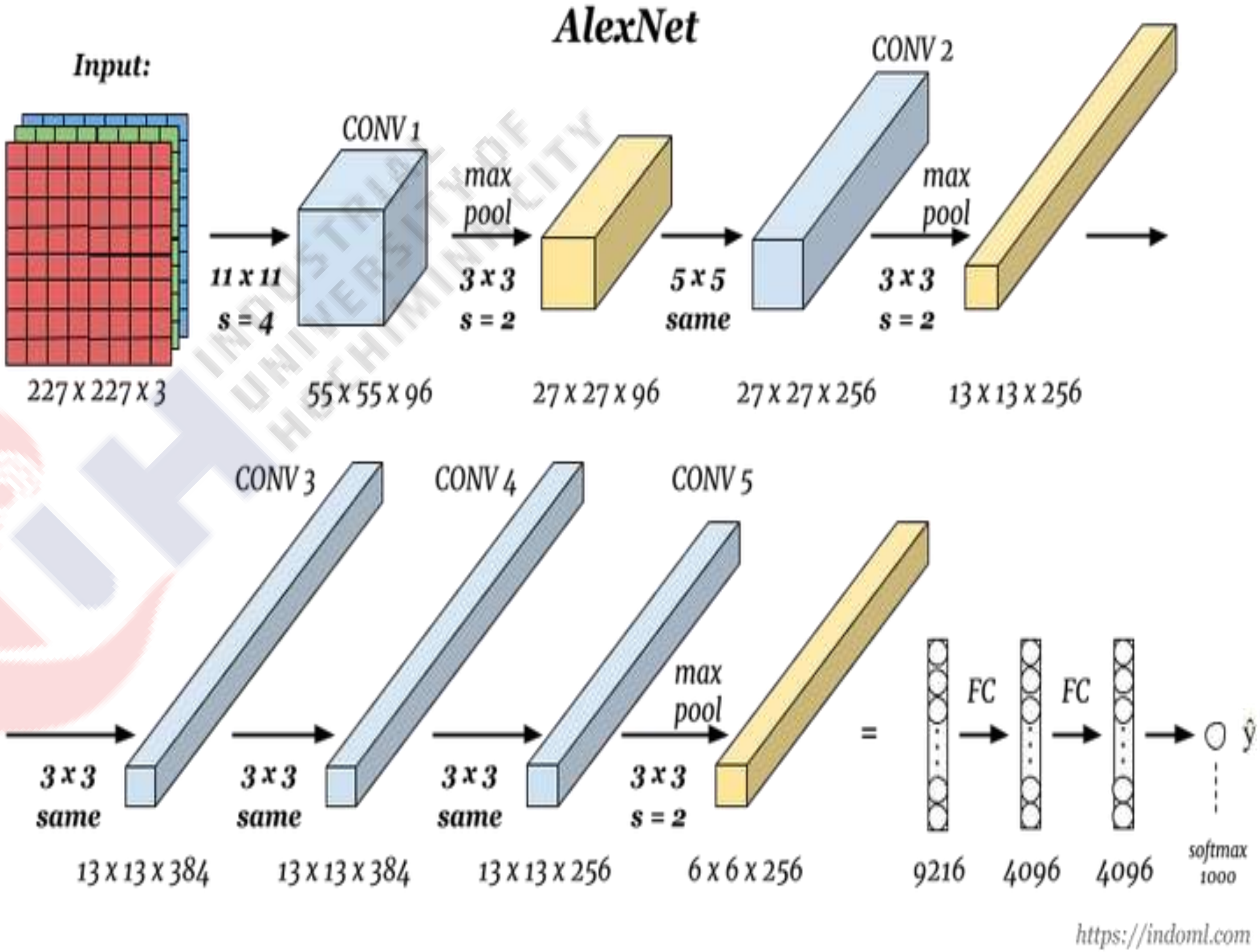


Layer	Layer Type	Feature Maps	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x82	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	Fully Connected	-	84	-	-	tanh
Output	Fully Connected	-	10	-	-	softmax

Classic Network: AlexNet



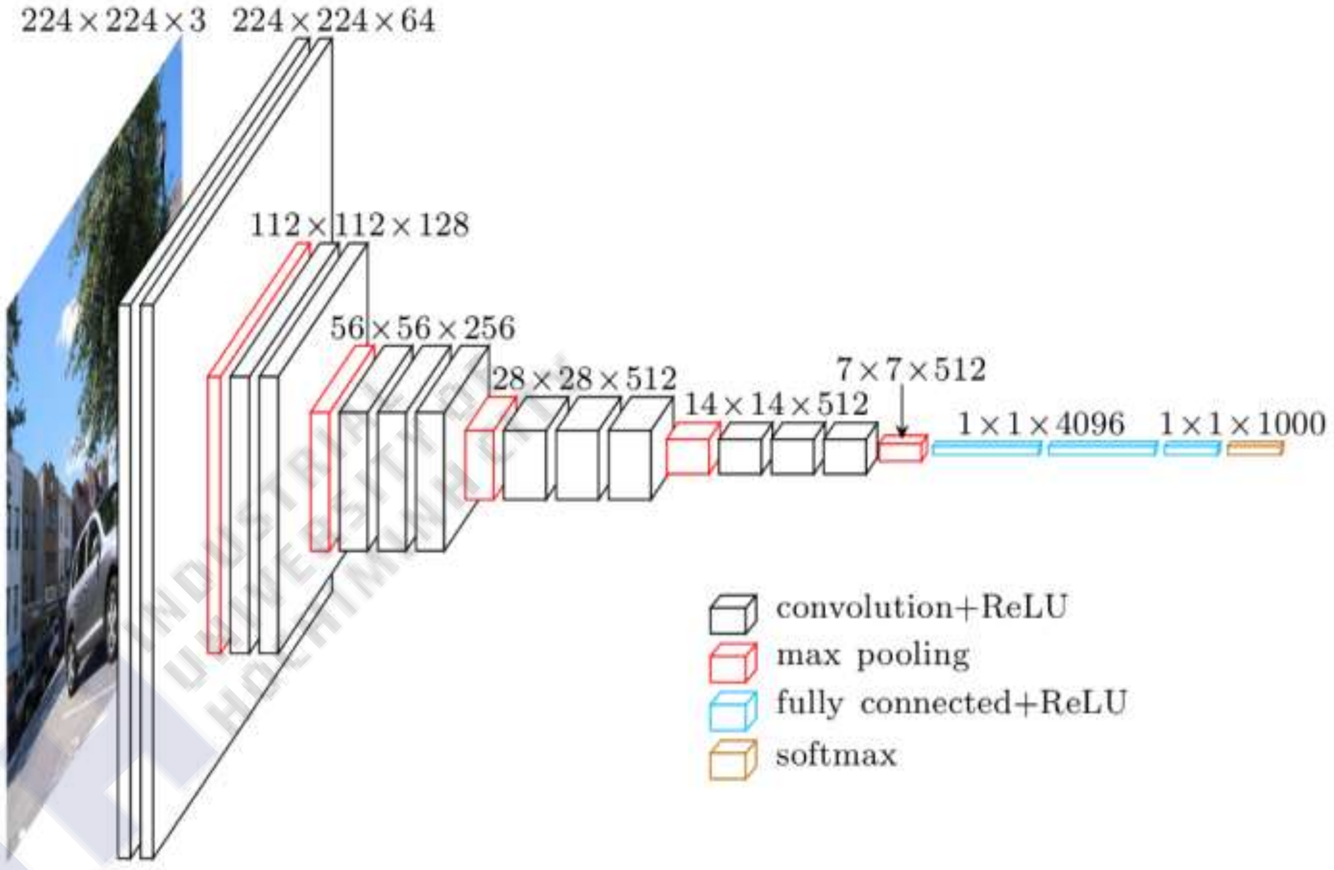
Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax



https://indoml.com

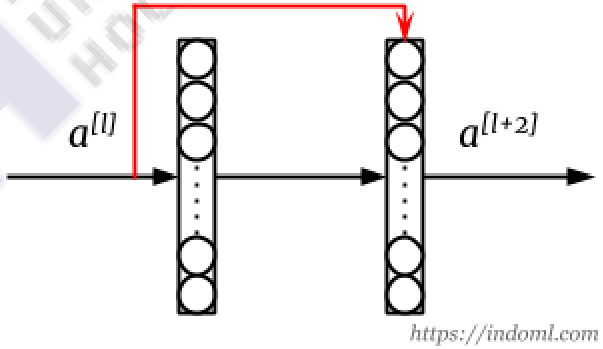
Classic Network: VGG-16

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

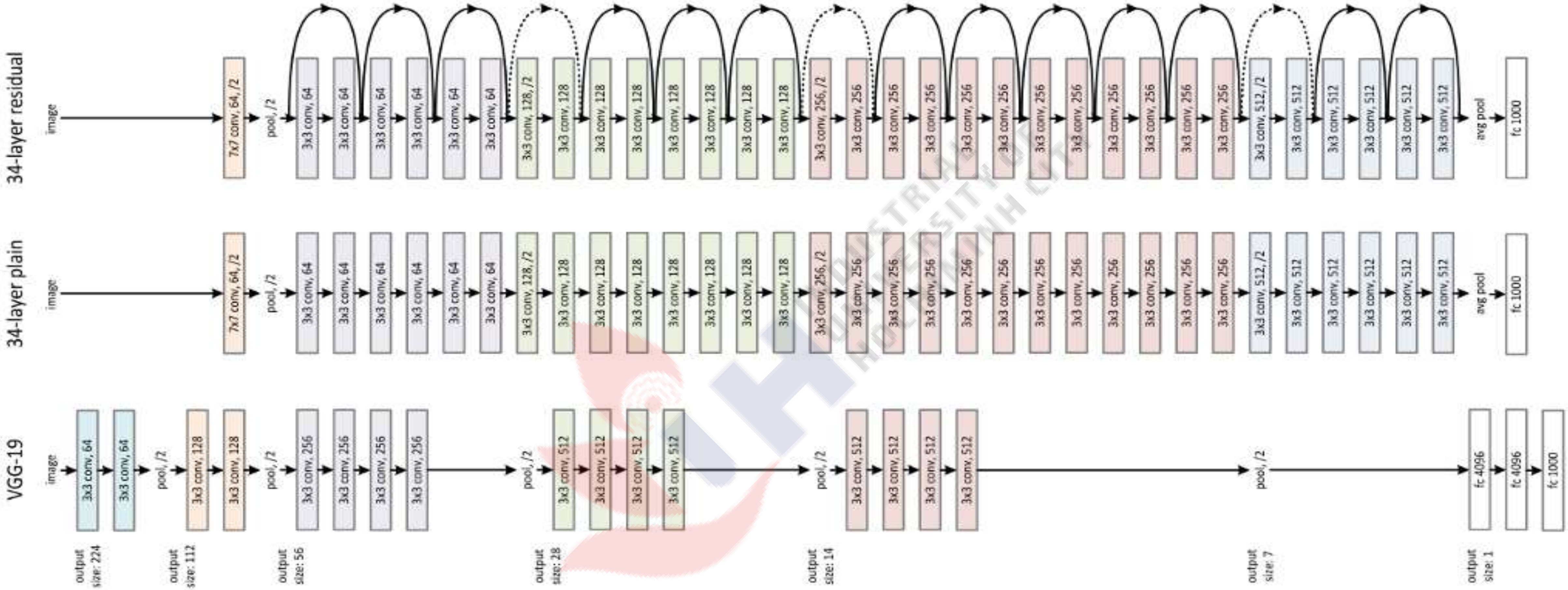


ResNet

- Vấn đề trong mạng deep neural networks sẽ khó huấn luyện khi đạt đến 1 cố layer nào đó, training error sẽ tăng trở lại, ngoài ra còn gặp vấn đề triệt tiêu hoặc đạo hàm quá lớn(exploding and vanishing gradients problem)
- Giải quyết vấn đề:
- $z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$
- $a^{[l+2]} = g^{[l+2]}(z^{[l+2]} + a^{[l]})$

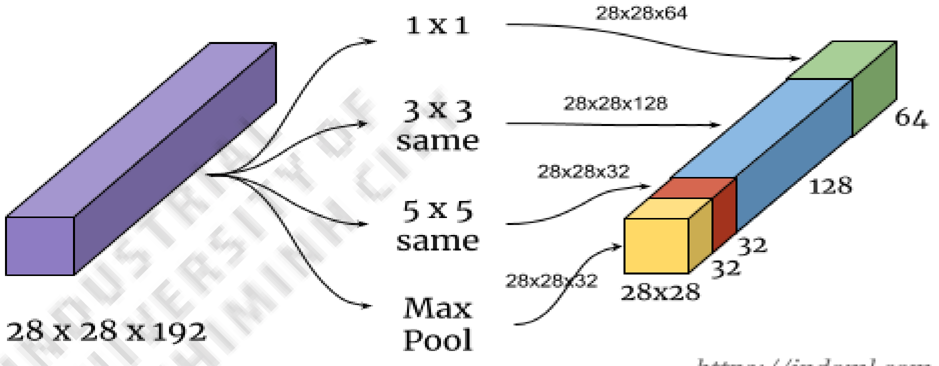


ResNet

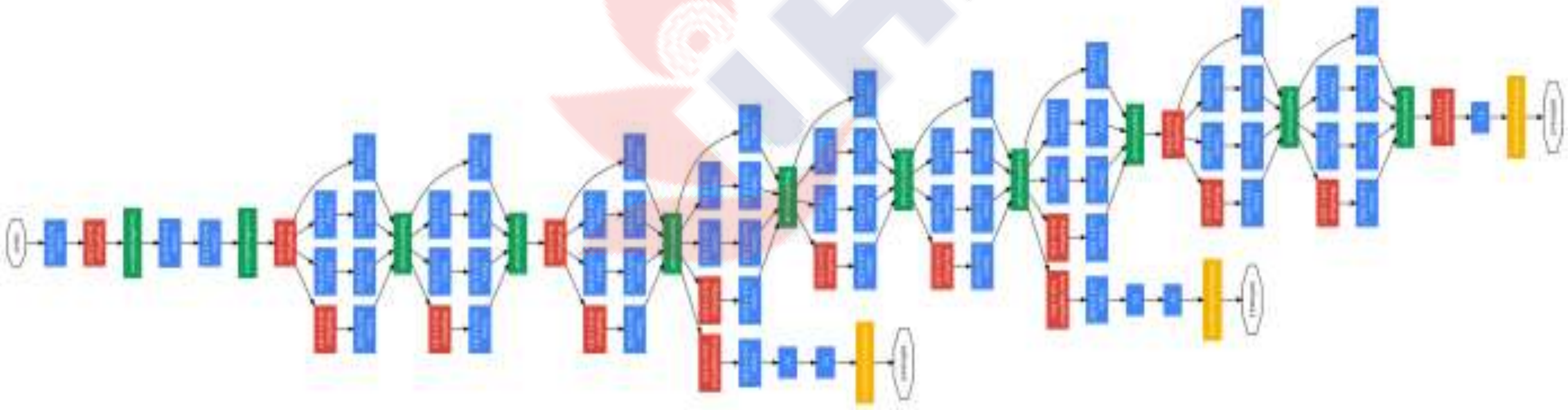

34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1

Inception

- Thay vì chọn kích thước filter 1 cách thủ công, network quyết định chọn cái tốt nhất để đưa vào layer
- **GoogLeNet**,

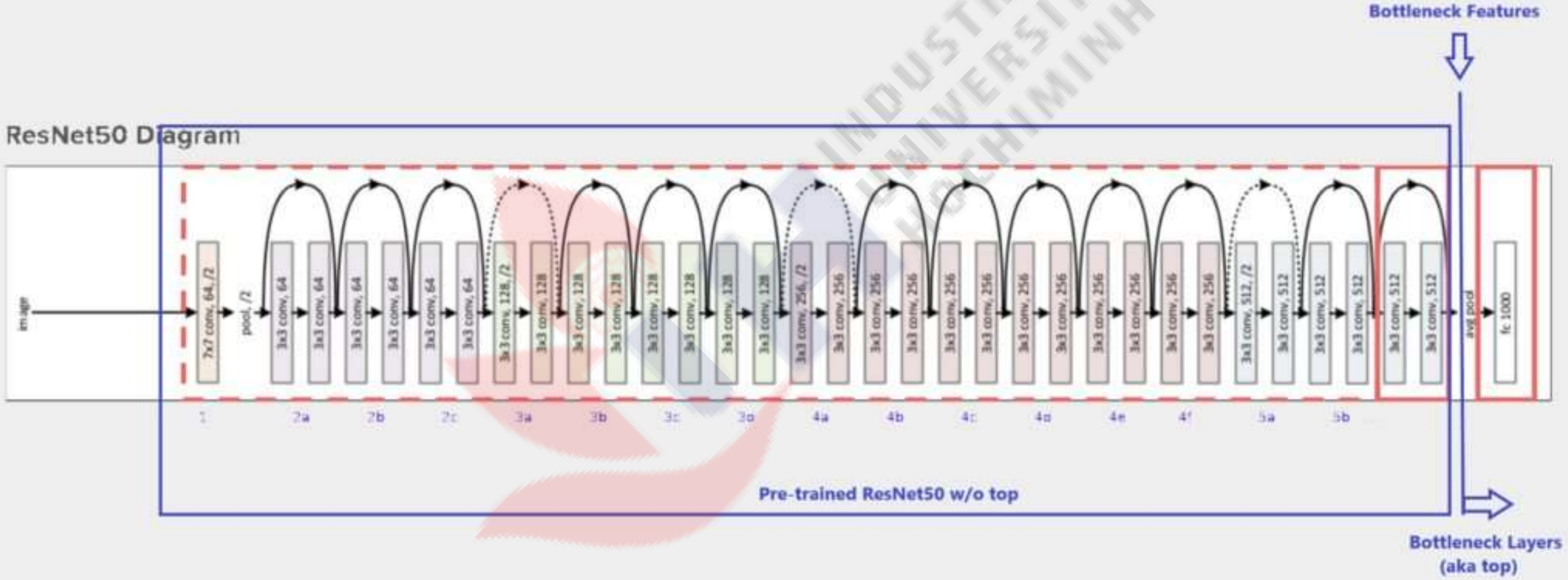


https://indoml.com



Retrain ResNet50

ResNet50 Diagram

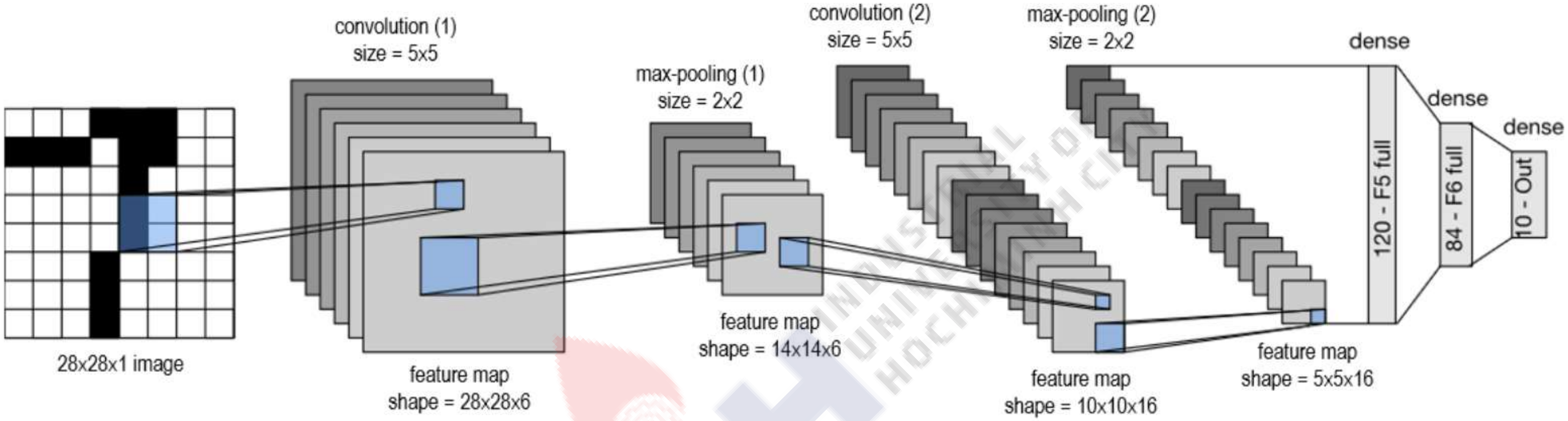


Bottleneck Features

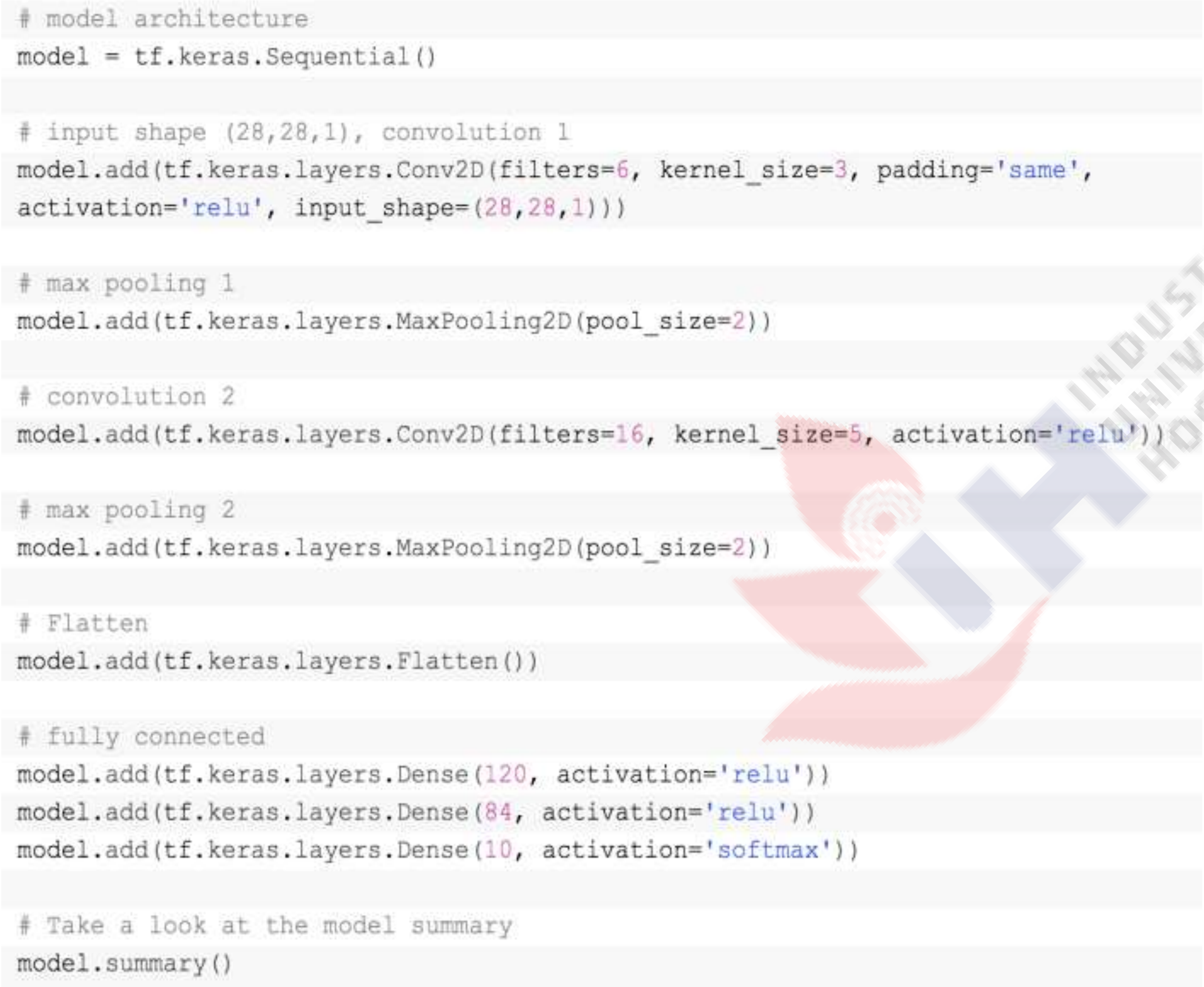
Pre-trained ResNet50 w/o top

Bottleneck Layers
(aka top)

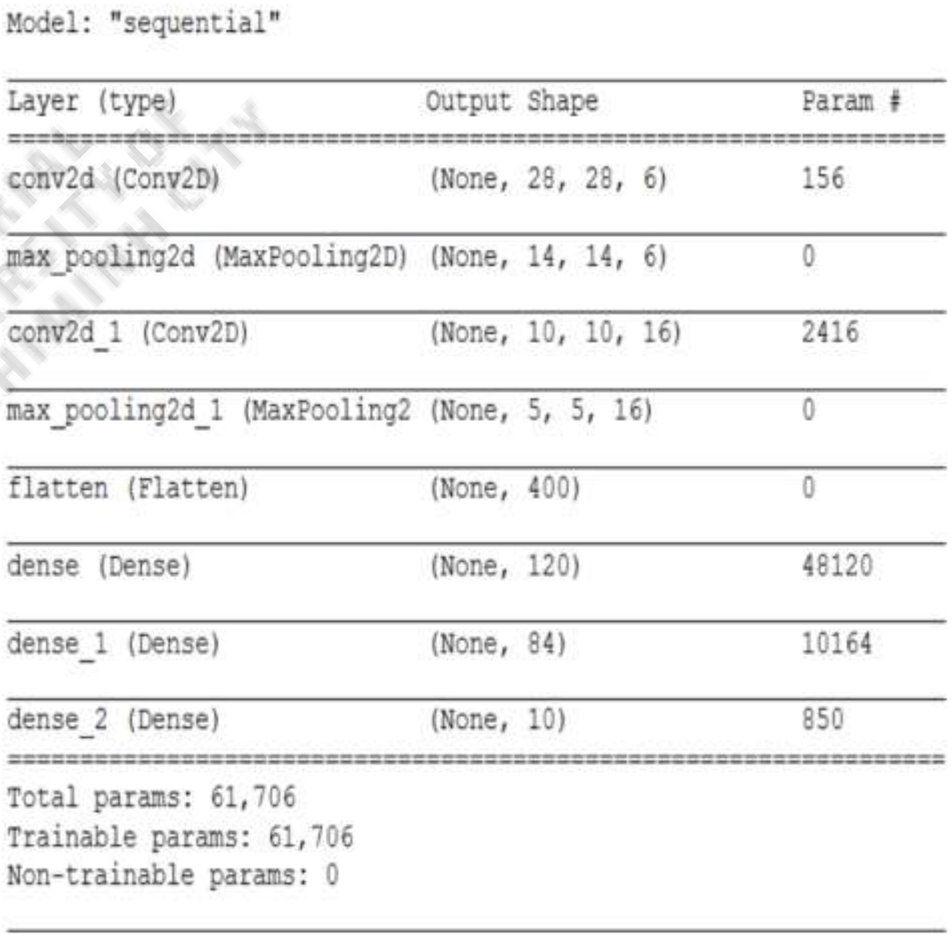
Thiết kế kiến trúc CNN



Thiết kế kiến trúc CNN



```
# model architecture
model = tf.keras.Sequential()

# input shape (28,28,1), convolution 1
model.add(tf.keras.layers.Conv2D(filters=6, kernel_size=3, padding='same',
activation='relu', input_shape=(28,28,1)))

# max pooling 1
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))

# convolution 2
model.add(tf.keras.layers.Conv2D(filters=16, kernel_size=5, activation='relu'))

# max pooling 2
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))

# Flatten
model.add(tf.keras.layers.Flatten())

# fully connected
model.add(tf.keras.layers.Dense(120, activation='relu'))
model.add(tf.keras.layers.Dense(84, activation='relu'))
model.add(tf.keras.layers.Dense(10, activation='softmax'))

# Take a look at the model summary
model.summary()
```



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 6)         156
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 120)               48120
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_2 (Dense)              (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

Thiết kế kiến trúc CNN

- model.add(tf.keras.layers.Conv2D(filters=6, kernel_size=5, padding='same', activation='relu', input_shape=(28,28,1)))
- model.add(layers.Conv2D(6, (5, 5), activation='relu', input_shape=(28, 28, 1)))

=> Tạo 6 filter, mỗi filter kích thước 5x5. Tổng số weigh: 6x5x5= 150, bias weight : 6x1. Tổng cộng: 6x5x5+6x1=156.

padding='same' => không có padding, không thay đổi kích thước ma trận input = > output (28x28x6)

- model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
- model.add(layers.MaxPooling2D((2, 2)))

=> Lớp maxPooling làm giảm kích thước ma trận theo 1 cửa sổ 2x2. Lớp này ko có weight. Từ input :(28x28x6) => output: (14x14x6)

- model.add(tf.keras.layers.Conv2D(filters=16, kernel_size=5, activation='relu'))
- model.add(layers.Conv2D(16, (5, 5), activation='relu'))

=>tạo 16 filter với kích thước mỗi filter là 5 x 5. Input : (14 x 14 x 6). Mỗi filter có 5x5 weights, lớp này có 5x5x6+1 = 151 weights cho mỗi phép tích chập. Tổng cộng có 16 filter: 151x16= 2416 weights.

padding = “SAME” => không có padding, output: (10x10x16)

- model.add(tf.keras.layers.MaxPooling2D(pool_size=2))

=> lớp maxPooling (2x2): output: (5x5x16), không có weight

- model.add(tf.keras.layers.Flatten())

=> sẽ flatten (5x 5 x 16) thành layer có 400 nodes, lớp này kết nối với FC có 120 nodes:

- model.add(tf.keras.layers.Dense(120, activation='relu'))

= 400x120+120=48120 weights

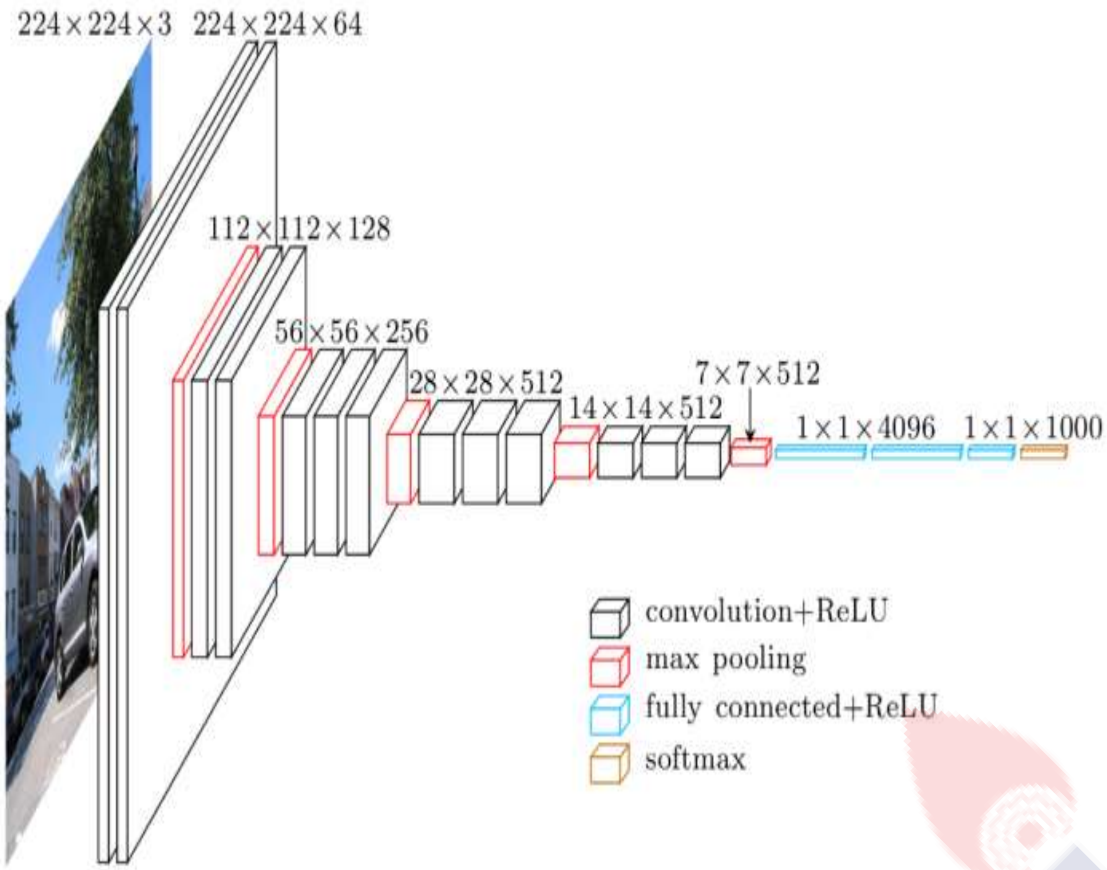
model.add(tf.keras.layers.Dense(84, activation='relu'))

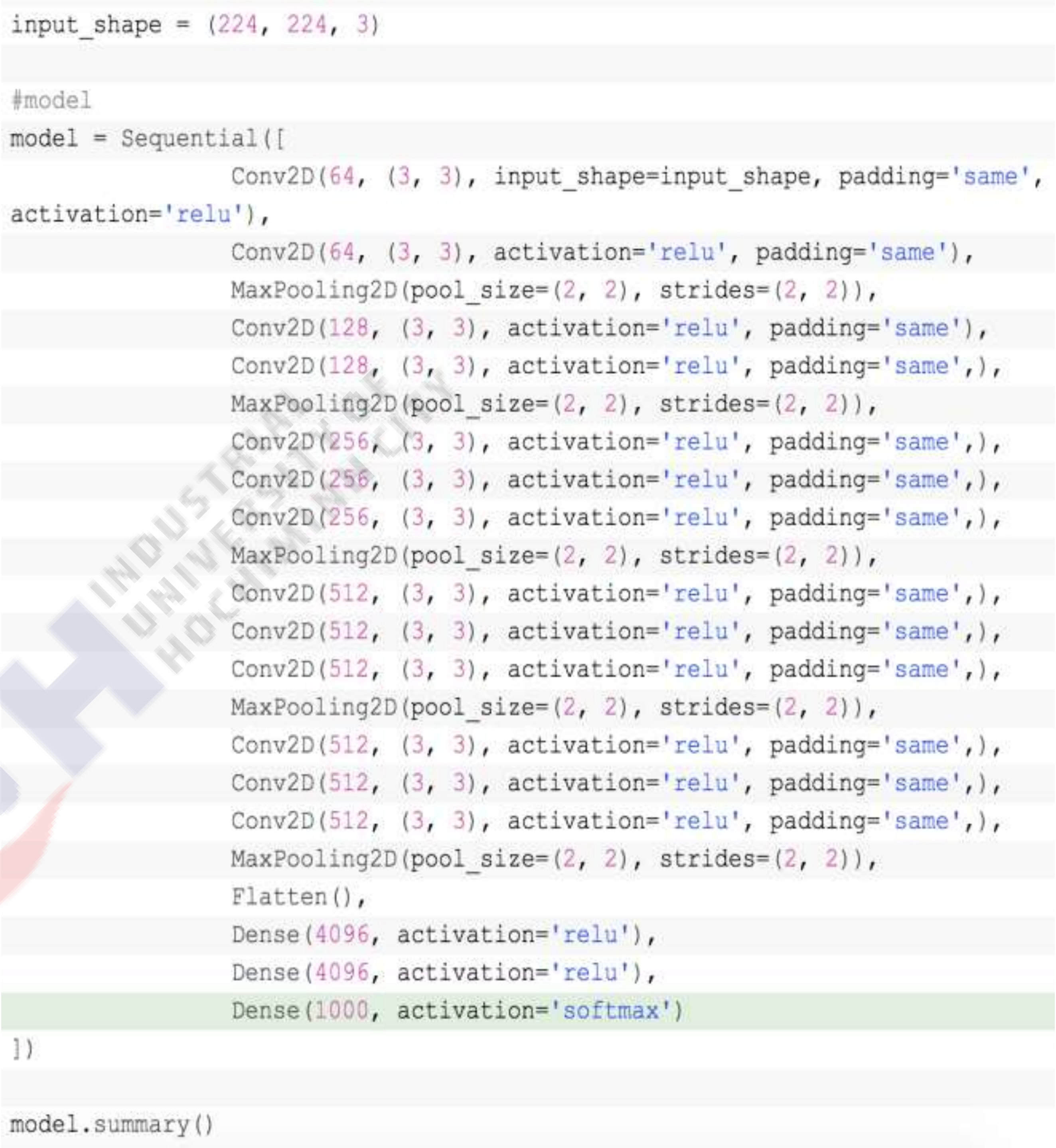
= 120x84+84=10164

model.add(tf.keras.layers.Dense(84, activation='relu'))

=84x10+10=850

Tổng cộng: 61,706





```
input_shape = (224, 224, 3)

#model
model = Sequential([
                    Conv2D(64, (3, 3), input_shape=input_shape, padding='same',
activation='relu'),
                    Conv2D(64, (3, 3), activation='relu', padding='same'),
                    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
                    Conv2D(128, (3, 3), activation='relu', padding='same'),
                    Conv2D(128, (3, 3), activation='relu', padding='same',),
                    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
                    Conv2D(256, (3, 3), activation='relu', padding='same',),
                    Conv2D(256, (3, 3), activation='relu', padding='same',),
                    Conv2D(256, (3, 3), activation='relu', padding='same',),
                    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
                    Conv2D(512, (3, 3), activation='relu', padding='same',),
                    Conv2D(512, (3, 3), activation='relu', padding='same',),
                    Conv2D(512, (3, 3), activation='relu', padding='same',),
                    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
                    Conv2D(512, (3, 3), activation='relu', padding='same',),
                    Conv2D(512, (3, 3), activation='relu', padding='same',),
                    Conv2D(512, (3, 3), activation='relu', padding='same',),
                    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
                    Flatten(),
                    Dense(4096, activation='relu'),
                    Dense(4096, activation='relu'),
                    Dense(1000, activation='softmax')
])

model.summary()
```