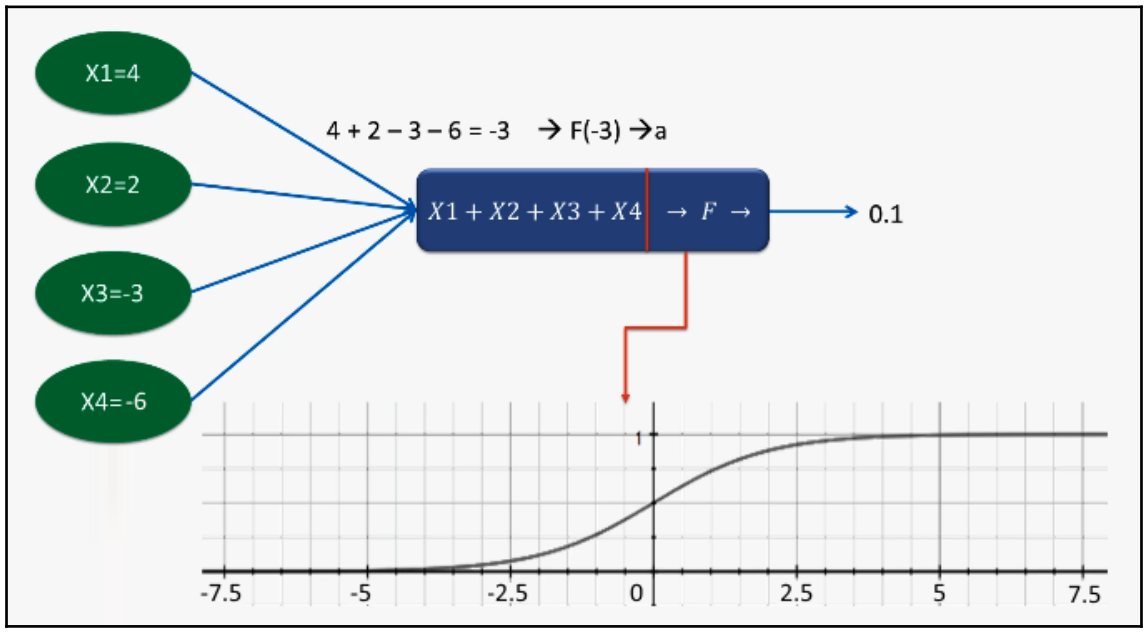
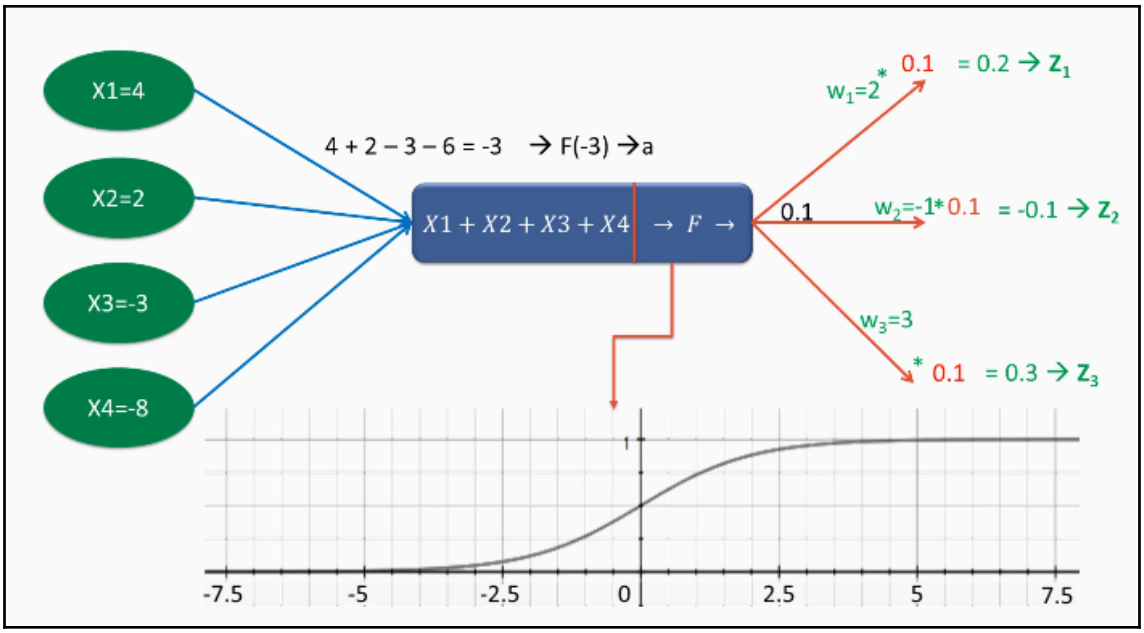
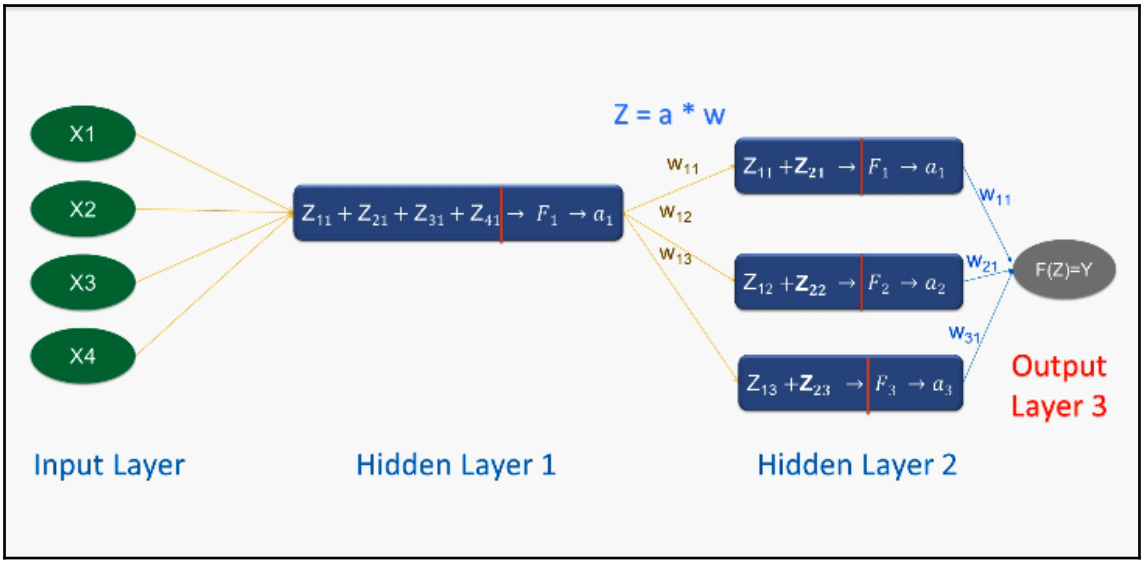
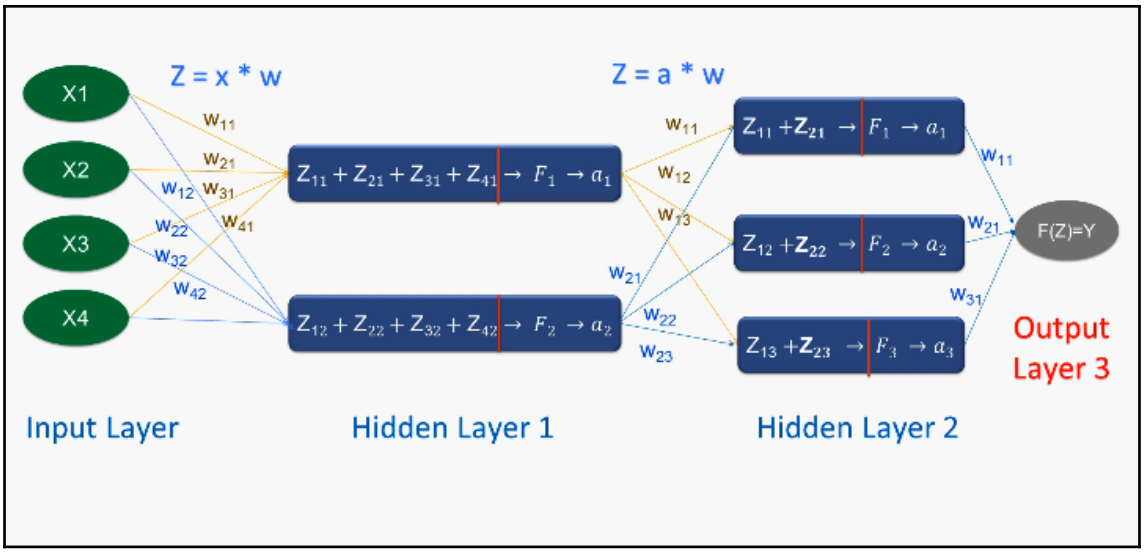
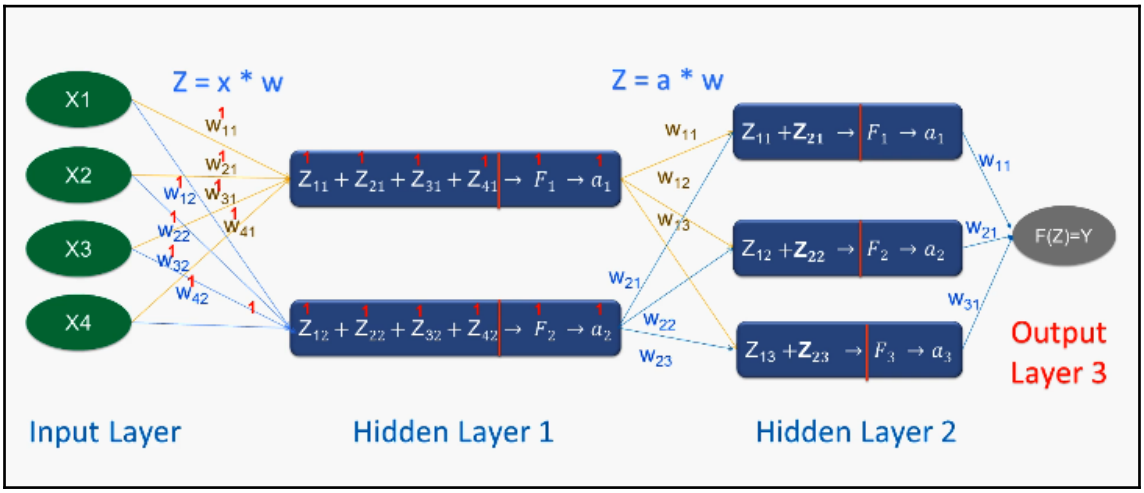
Chapter 1: Introduction to Computer Vision and Training Neural Networks

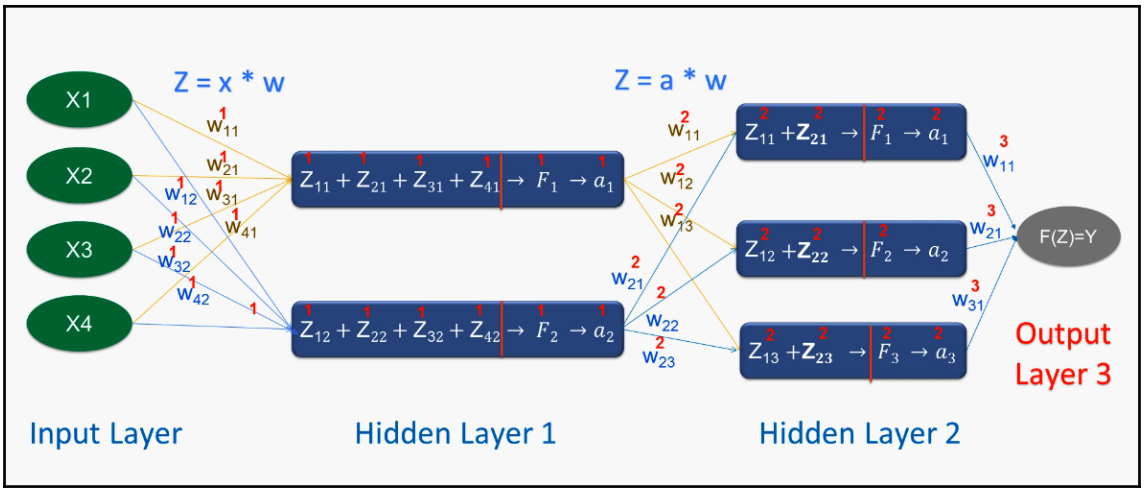


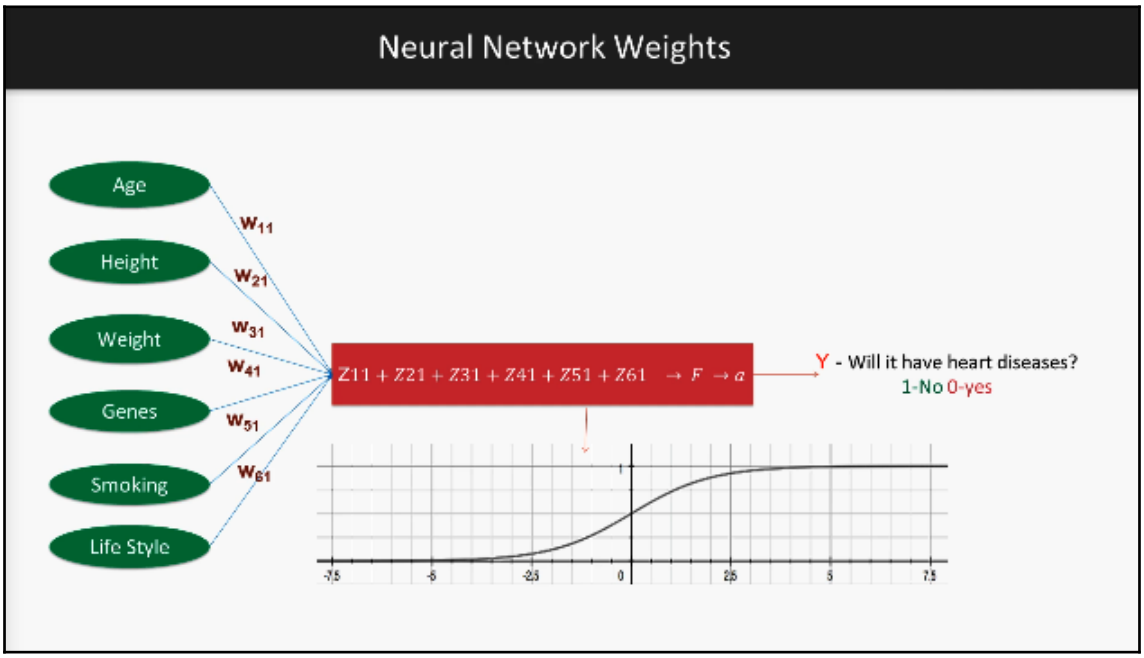

X1=4
X2=2
X3=-3
X4=-8
4 + 2 – 3 – 6 = -3 → F(-3) →a
X1 + X2 + X3 + X4 → F →
0.1
w1=2 * 0.1 = 0.2 → Z1
w2=-1* 0.1 = -0.1 → Z2
w3=3 * 0.1 = 0.3 → Z3
1
-7.5
-5
-2.5
0
2.5
5
7.5

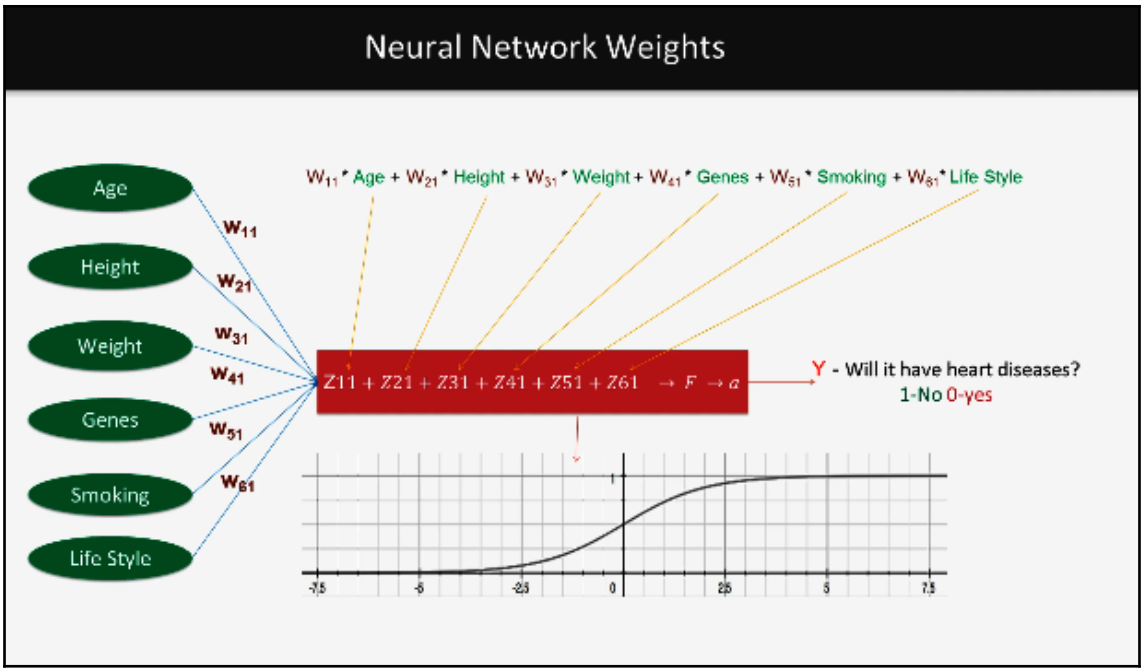

X1
X2
X3
X4
Z = a * w
Z11 + Z21 + Z31 + Z41 → F1 → a1
w11
w12
w13
Z11 + Z21 → F1 → a1
Z12 + Z22 → F2 → a2
Z13 + Z23 → F3 → a3
w11
w21
w31
F(Z)=Y
Output Layer 3
Input Layer
Hidden Layer 1
Hidden Layer 2

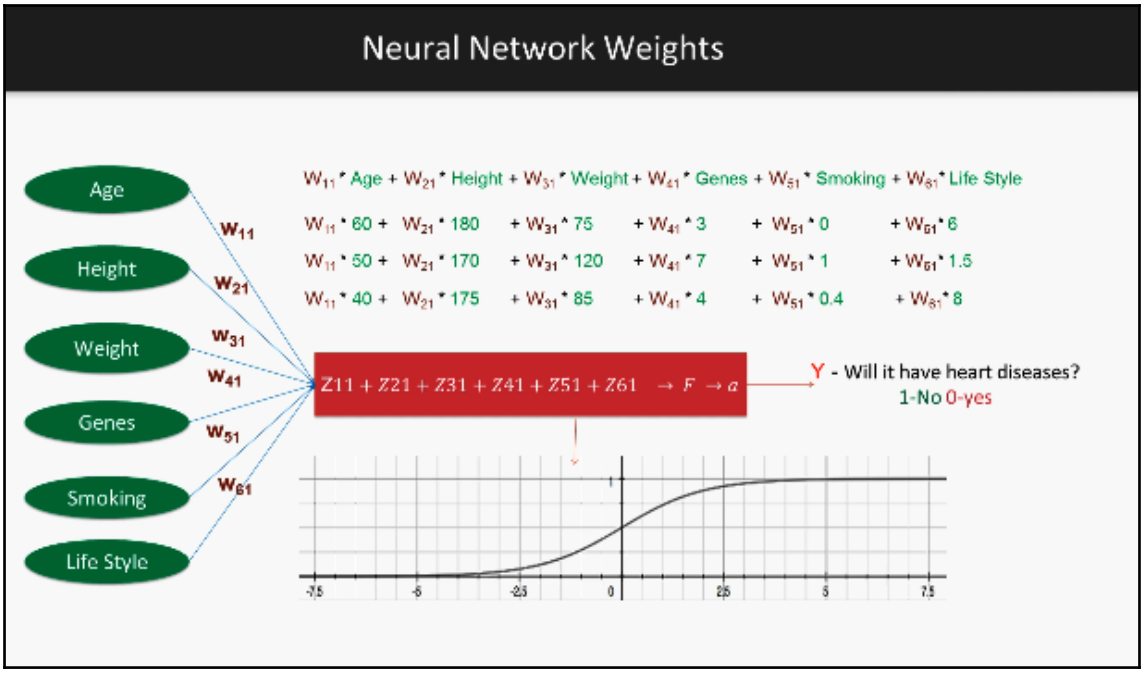

X1
X2
X3
X4
Z = x * w
Z = a * w
$Z_{11}+Z_{21}+Z_{31}+Z_{41} \rightarrow F_1 \rightarrow a_1$
$Z_{12}+Z_{22}+Z_{32}+Z_{42} \rightarrow F_2 \rightarrow a_2$
$Z_{11}+Z_{21} \rightarrow F_1 \rightarrow a_1$
$Z_{12}+Z_{22} \rightarrow F_2 \rightarrow a_2$
$Z_{13}+Z_{23} \rightarrow F_3 \rightarrow a_3$
F(Z)=Y
Output Layer 3
Input Layer
Hidden Layer 1
Hidden Layer 2


X1
X2
X3
X4
Z = x * w
Z = a * w
$Z_{11}+Z_{21}+Z_{31}+Z_{41} \rightarrow F_1 \rightarrow a_1$
$Z_{12}+Z_{22}+Z_{32}+Z_{42} \rightarrow F_2 \rightarrow a_2$
$Z_{11}+Z_{21} \rightarrow F_1 \rightarrow a_1$
$Z_{12}+Z_{22} \rightarrow F_2 \rightarrow a_2$
$Z_{13}+Z_{23} \rightarrow F_3 \rightarrow a_3$
F(Z)=Y
Output Layer 3
Input Layer
Hidden Layer 1
Hidden Layer 2

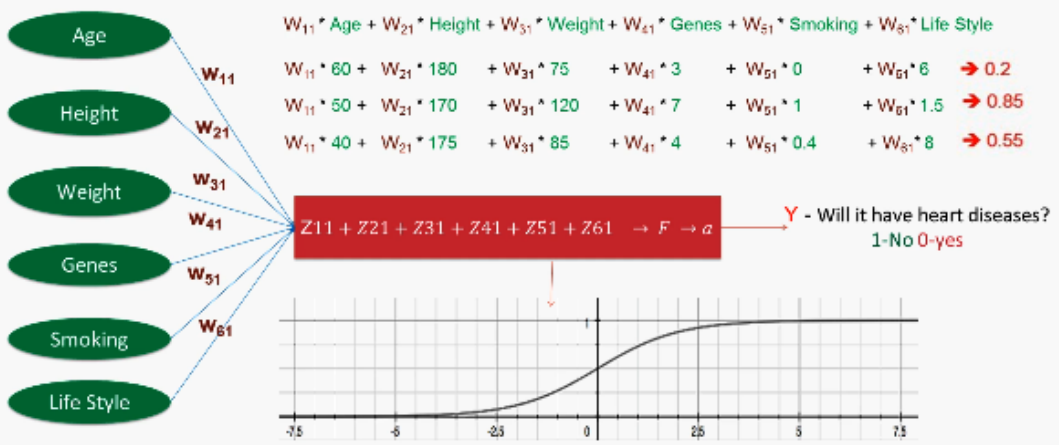

X1
X2
X3
X4
Z = x * w
Z = a * w
F(Z)=Y
Output Layer 3
Input Layer
Hidden Layer 1
Hidden Layer 2


Neural Network Weights
Age
Height
Weight
Genes
Smoking
Life Style
Y - Will it have heart diseases?
1-No 0-yes

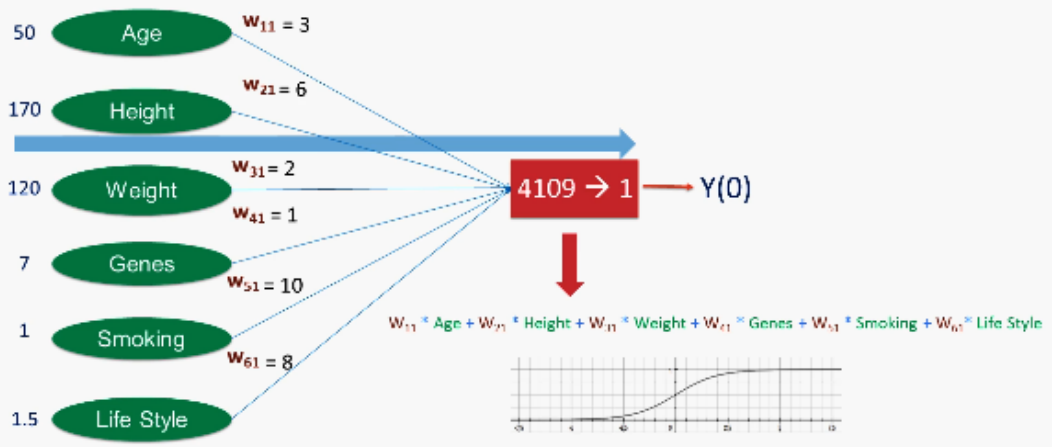

Neural Network Weights
Age
Height
Weight
Genes
Smoking
Life Style
W11 * Age + W21 * Height + W31 * Weight + W41 * Genes + W51 * Smoking + W61 * Life Style
Z11 + Z21 + Z31 + Z41 + Z51 + Z61 → F → a
Y - Will it have heart diseases?
1-No 0-yes

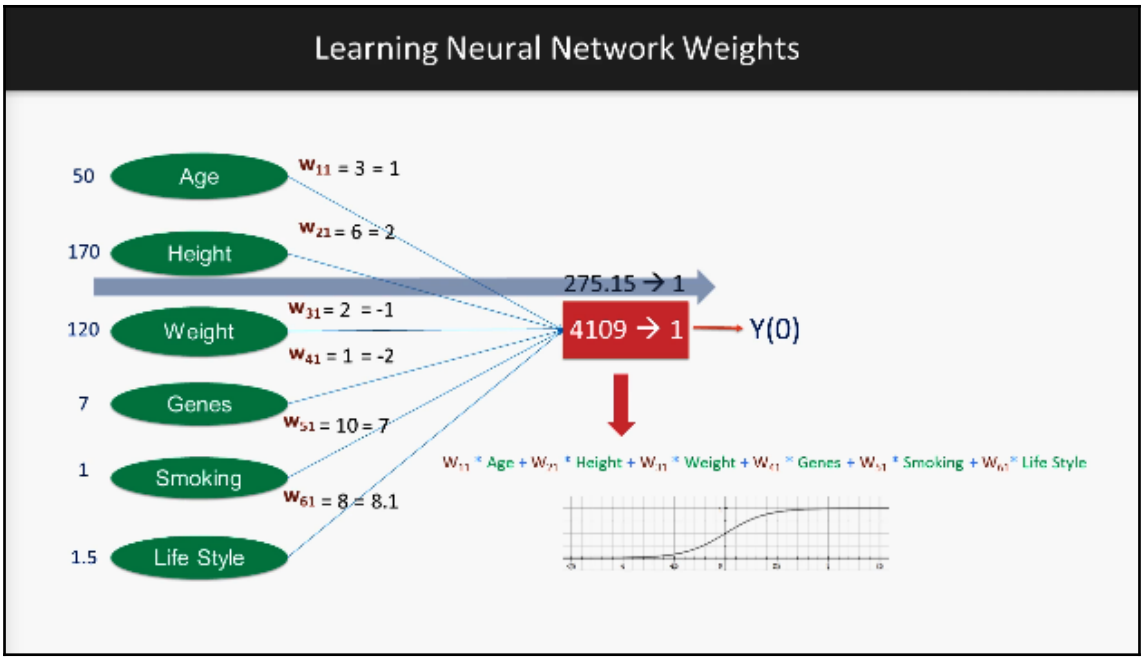

Neural Network Weights
Age
Height
Weight
Genes
Smoking
Life Style
W11 * Age + W21 * Height + W31 * Weight + W41 * Genes + W51 * Smoking + W61 * Life Style
W11 * 60 + W21 * 180 + W31 * 75 + W41 * 3 + W51 * 0 + W61 * 6
W11 * 50 + W21 * 170 + W31 * 120 + W41 * 7 + W51 * 1 + W61 * 1.5
W11 * 40 + W21 * 175 + W31 * 85 + W41 * 4 + W51 * 0.4 + W61 * 8
Z11 + Z21 + Z31 + Z41 + Z51 + Z61 → F → a
Y - Will it have heart diseases?
1-No 0-yes

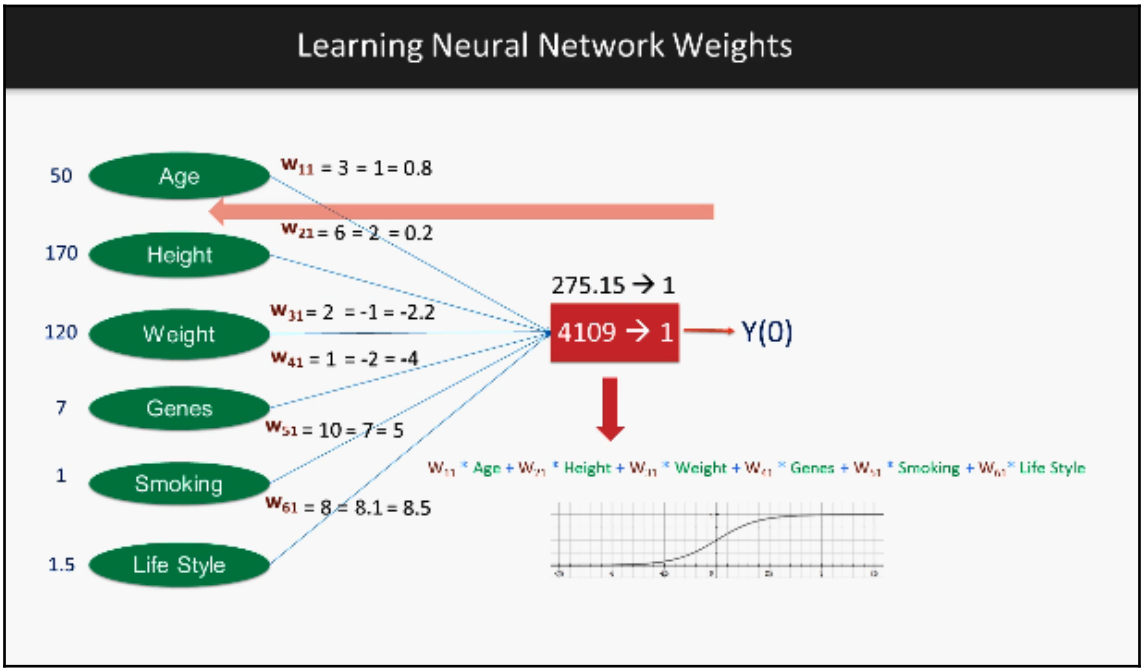
Neural Network Weights

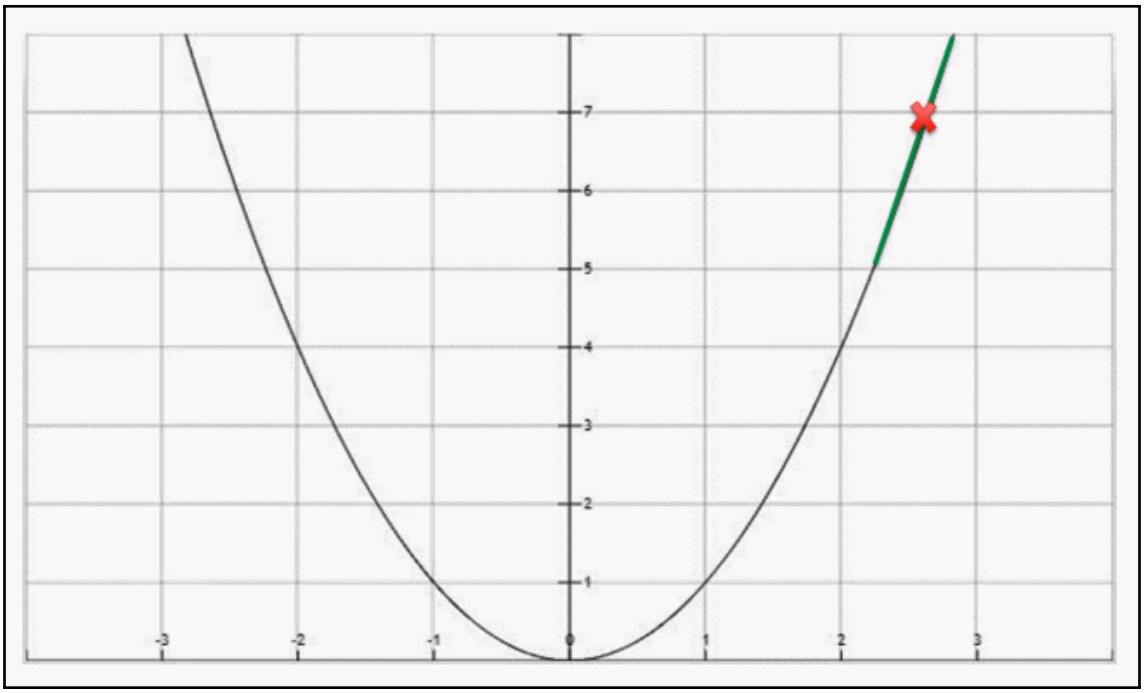


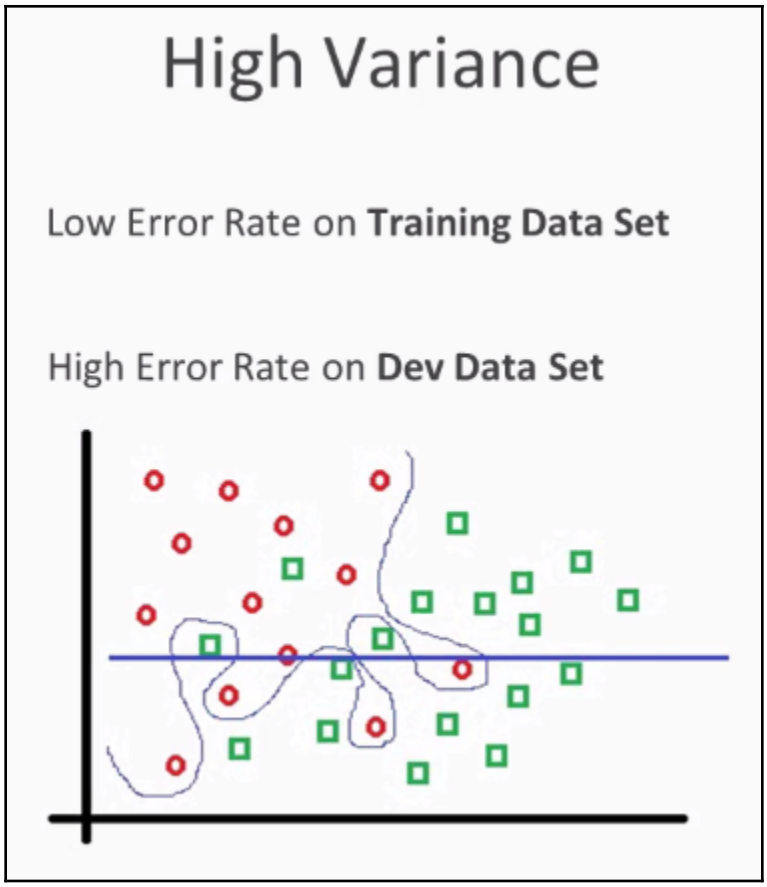
Learning Neural Network Weights



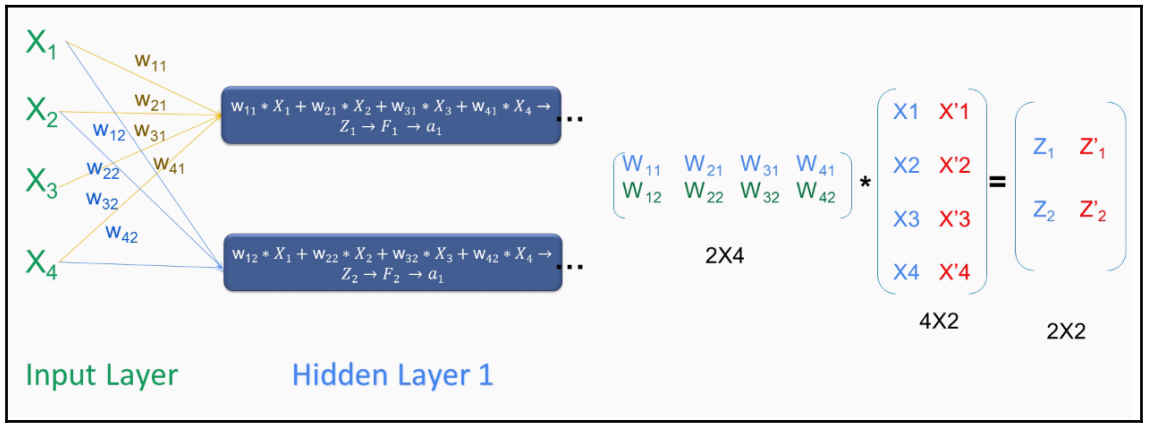

Learning Neural Network Weights
50
Age
w_{11} = 3 = 1
170
Height
w_{21} = 6 = 2
275.15 → 1
120
Weight
w_{31} = 2 = -1
4109 → 1
Y(0)
w_{41} = 1 = -2
7
Genes
w_{51} = 10 = 7
1
Smoking
w_{61} = 8 = 8.1
1.5
Life Style
W_{11} * Age + W_{21} * Height + W_{31} * Weight + W_{41} * Genes + W_{51} * Smoking + W_{61} * Life Style

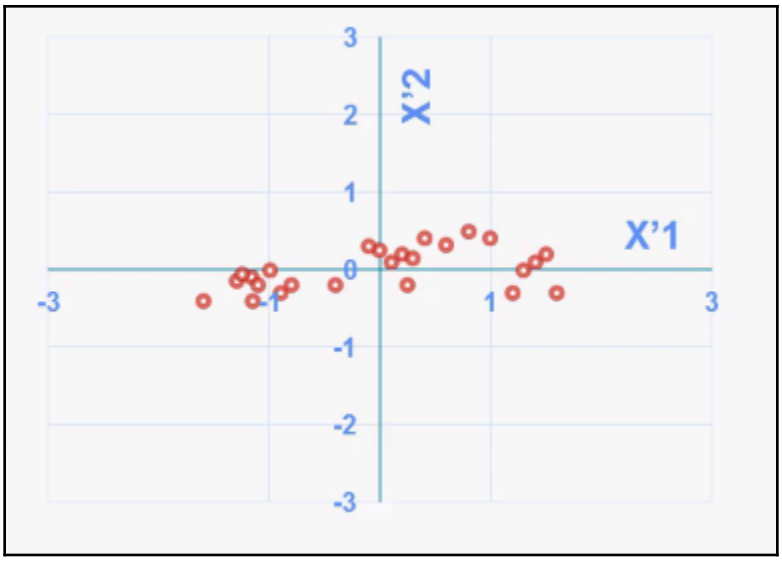

Learning Neural Network Weights
50
Age
w_{11} = 3 = 1 = 0.8
170
Height
w_{21} = 6 = 2 = 0.2
275.15 → 1
120
Weight
w_{31} = 2 = -1 = -2.2
4109 → 1
Y(0)
w_{41} = 1 = -2 = -4
7
Genes
w_{51} = 10 = 7 = 5
1
Smoking
w_{61} = 8 = 8.1 = 8.5
1.5
Life Style
W_{11} * Age + W_{21} * Height + W_{31} * Weight + W_{41} * Genes + W_{51} * Smoking + W_{61} * Life Style

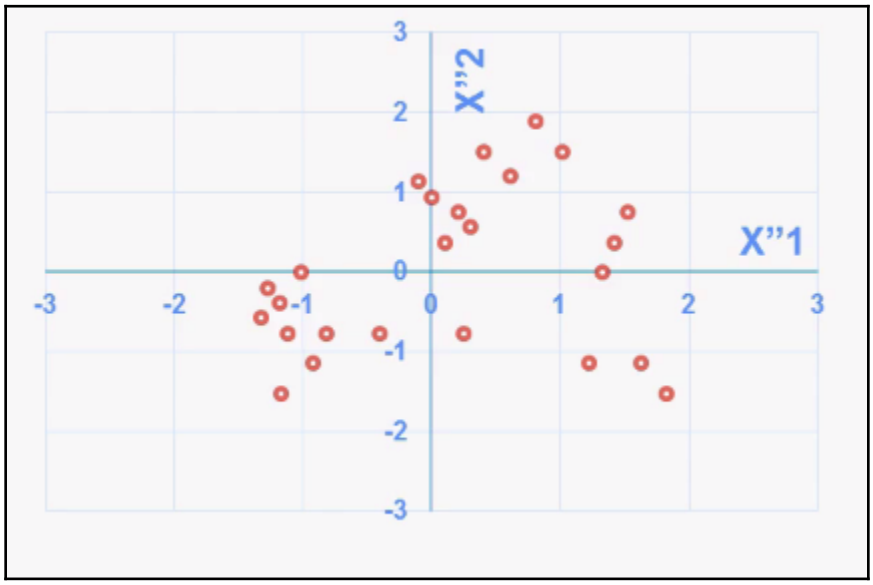

7
6
5
4
3
2
1
-3
-2
-1
0
1
2
3

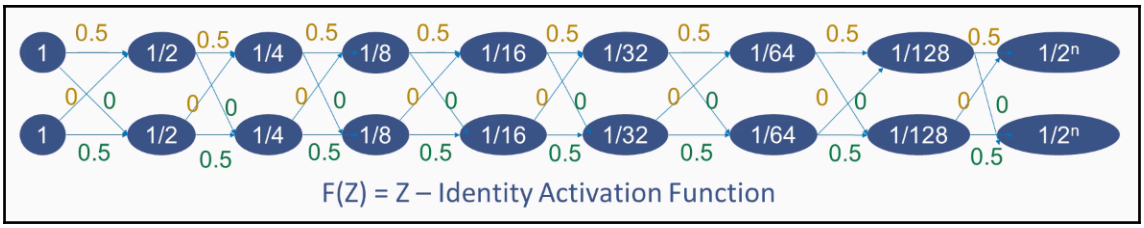

High Variance
Low Error Rate on **Training Data Set**
High Error Rate on **Dev Data Set**

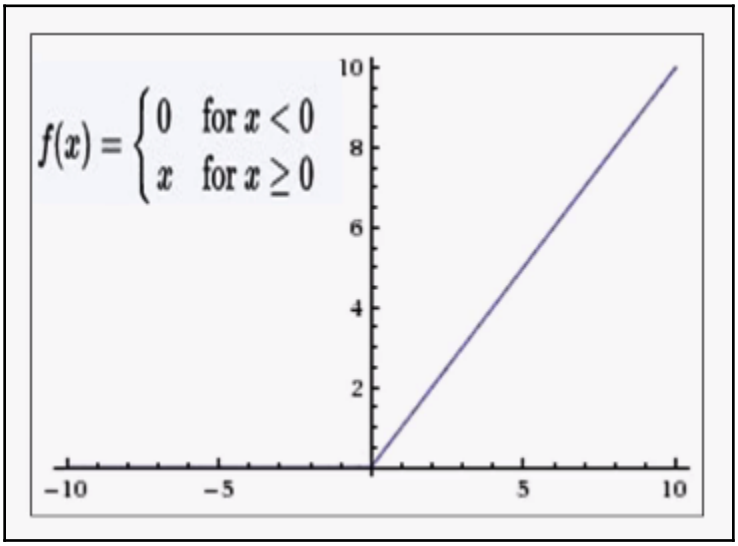

Both ☹
High Error Rate on **Training Data Set**
Even Higher Error Rate on **Dev Data Set**

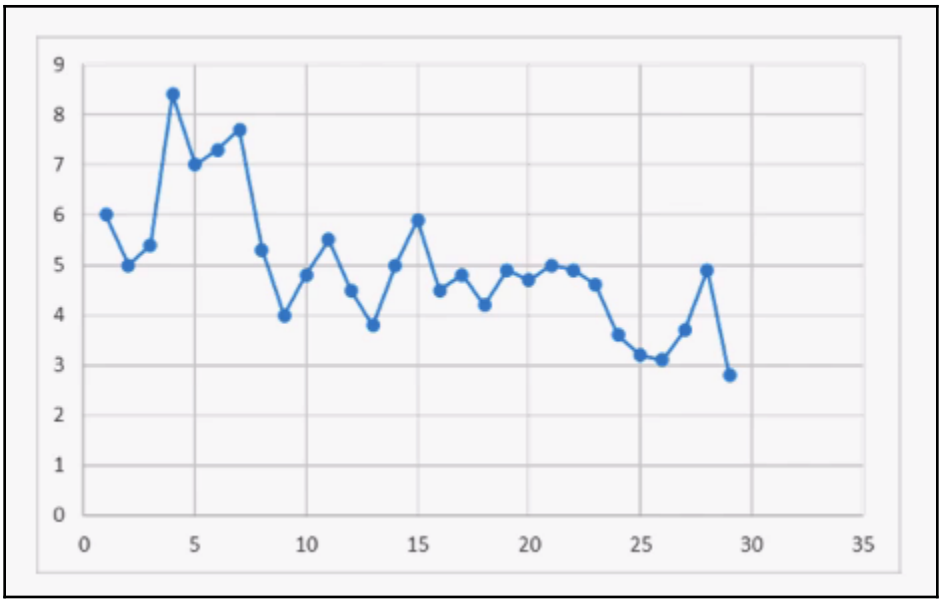

X1
X2
X3
X4
w11
w21
w12
w31
w22
w41
w32
w42
w11 * X1 + w21 * X2 + w31 * X3 + w41 * X4 → Z1 → F1 → a1
w12 * X1 + w22 * X2 + w32 * X3 + w42 * X4 → Z2 → F2 → a1
W11 W21 W31 W41
W12 W22 W32 W42
2X4
*
X1 X'1
X2 X'2
X3 X'3
X4 X'4
4X2
=
Z1 Z'1
Z2 Z'2
2X2
Input Layer
Hidden Layer 1

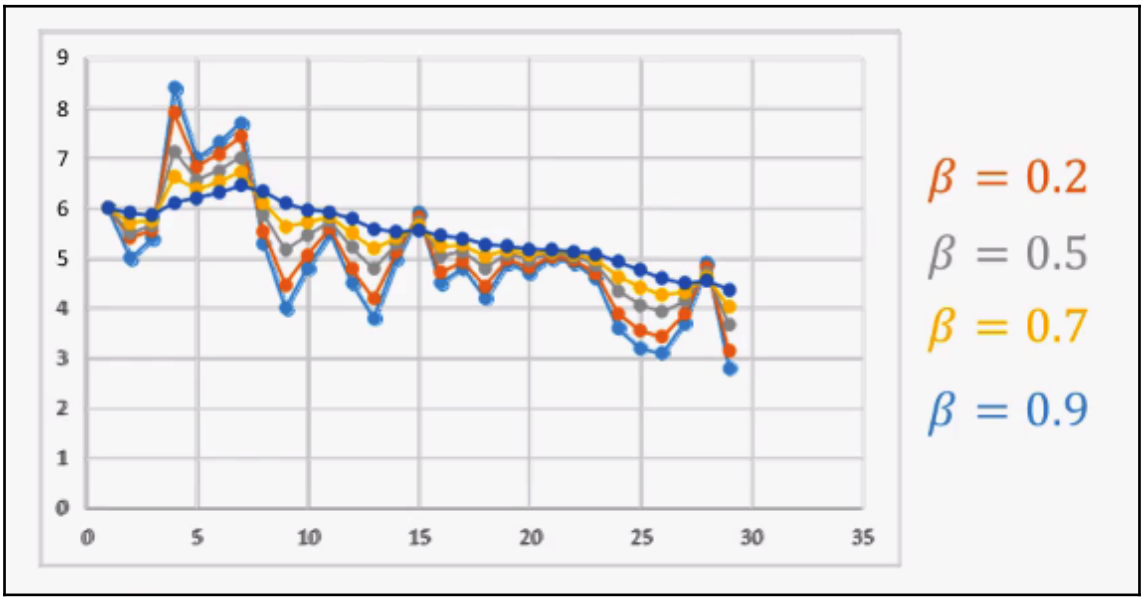

X'2
X'1
3
2
1
0
-1
-2
-3
-3
-1
1
3

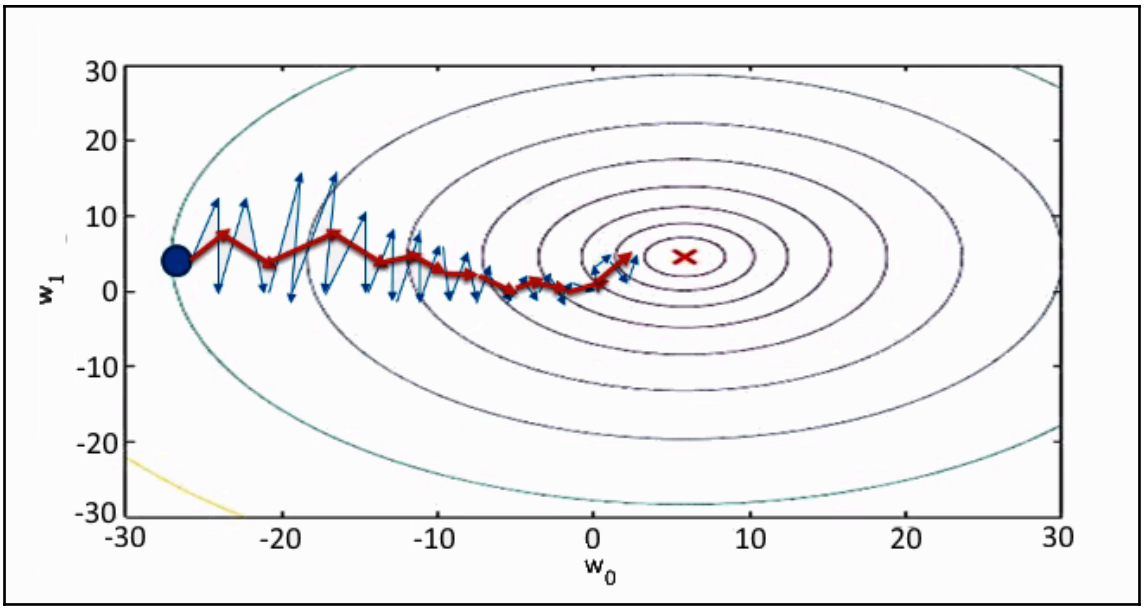

X''2
X''1
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


1
1/2
1/4
1/8
1/16
1/32
1/64
1/128
1/2ⁿ
0.5
0
F(Z) = Z – Identity Activation Function


f(x) = { 0 for x < 0
x for x ≥ 0
10
8
6
4
2
−10
−5
5
10


9
8
7
6
5
4
3
2
1
0
0
5
10
15
20
25
30
35


9
8
7
6
5
4
3
2
1
0
0
5
10
15
20
25
30
35
β = 0.2
β = 0.5
β = 0.7
β = 0.9



RMSProp

$$S_0 = 0$$

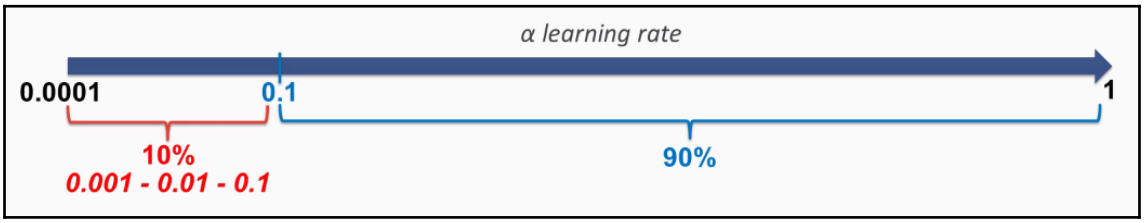
$$S_t = \beta * S_{t-1} + (1-\beta) * \left(\frac{dJ}{dW_{xy}}\right)^2$$

~~$w_{xy} = wxy - \alpha \frac{dJ}{dW_{xy}}$~~ $\quad w_{xy} = wxy - \alpha * \frac{\frac{dJ}{dw_{xy}}}{\sqrt{S_t}}$

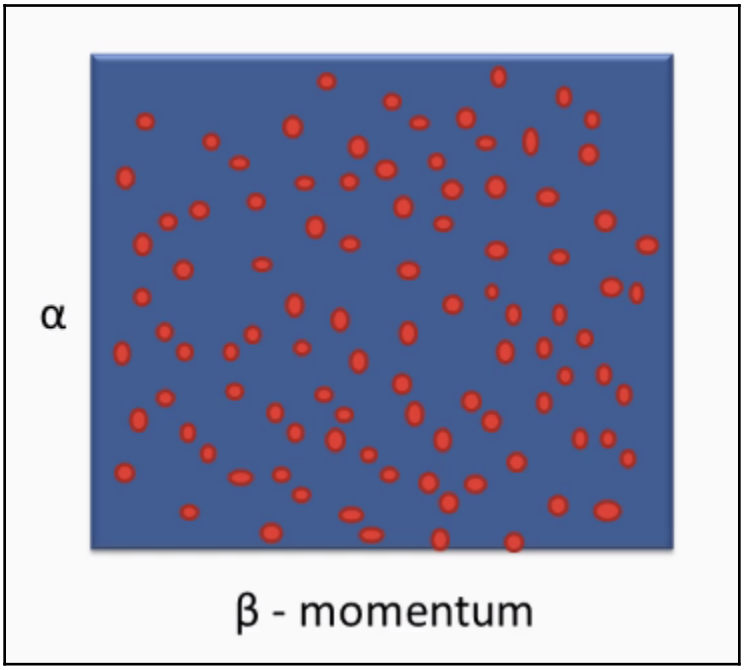
$$V_0 = 0 \quad S_0 = 0$$

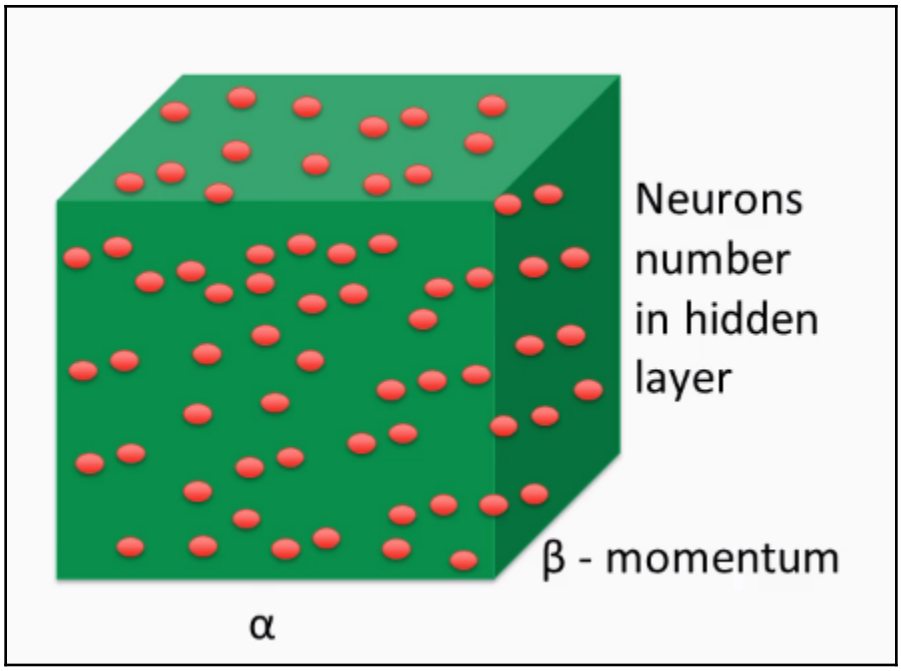
$$V_t = \beta * V_{t-1} + (1-\beta) * \frac{dJ}{dW_{xy}} \qquad S_t = \beta_2 * S_{t-1} + (1-\beta_2) * \left(\frac{dJ}{dW_{xy}}\right)^2$$

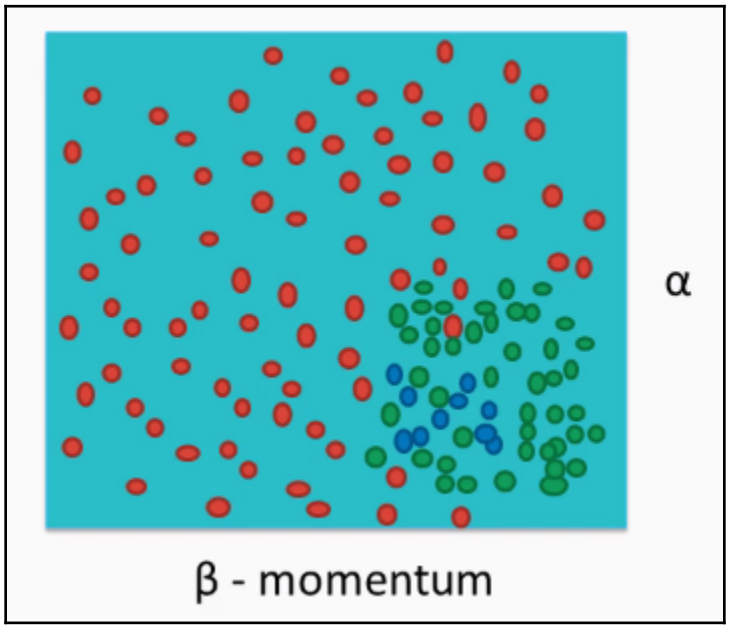
$$w_{xy} = wxy - \alpha \frac{dJ}{dW_{xy}} \qquad w_{xy} = wxy - \alpha * \frac{V_t}{\sqrt{S_t}}$$

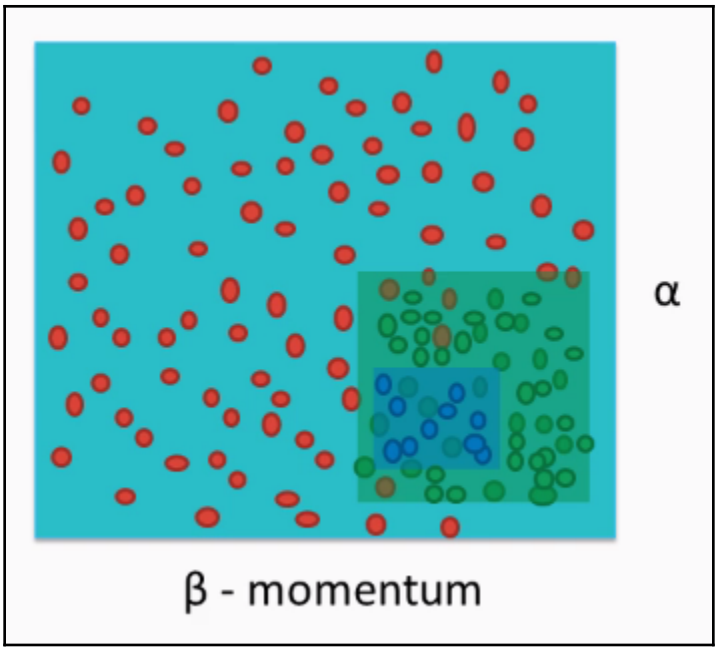


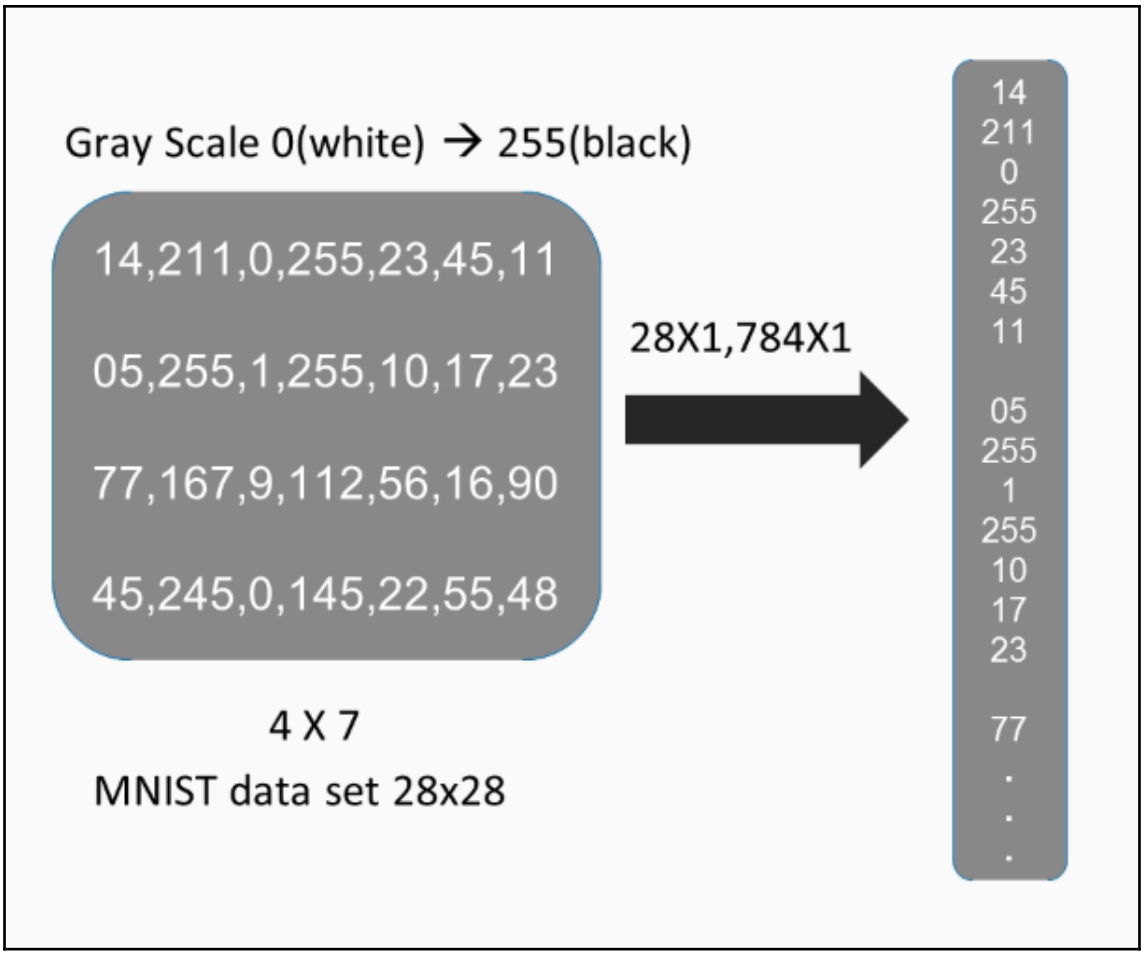

```
ThreadLocalRandom.current().nextDouble(-4, 0);
```

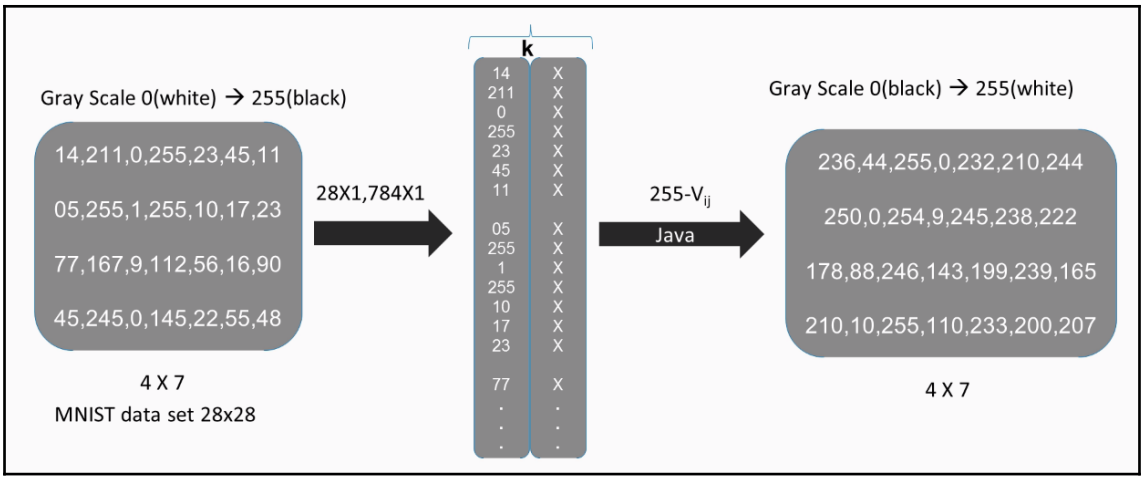


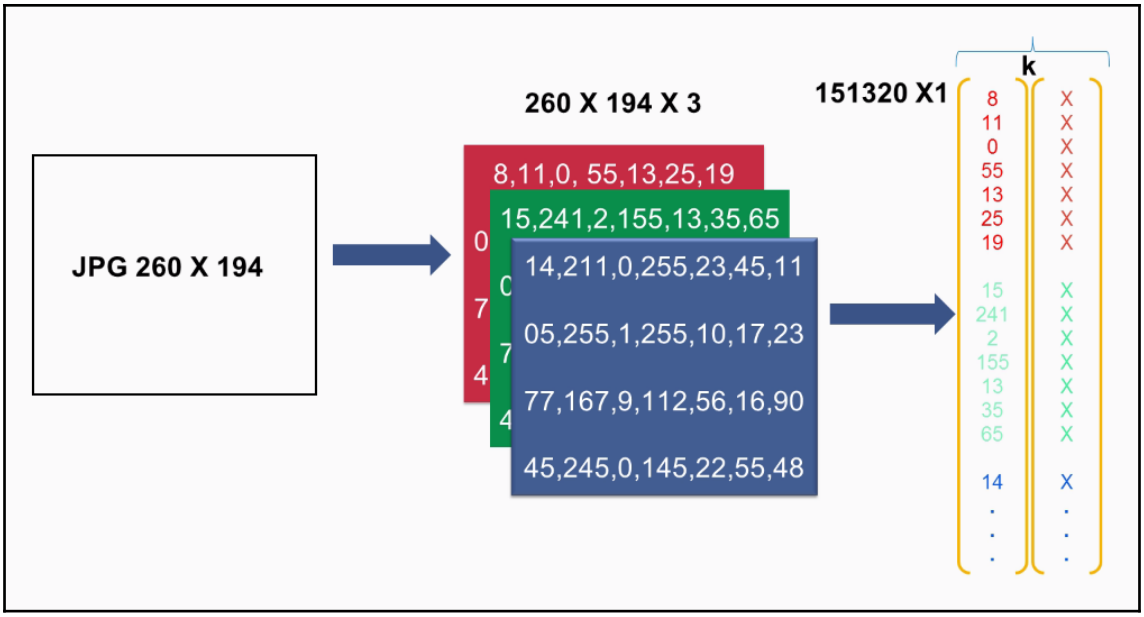

Neurons
number
in hidden
layer
β - momentum
α

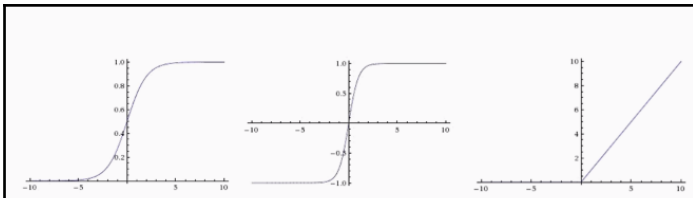

α
β - momentum


α
β - momentum


Gray Scale 0(white) → 255(black)
14,211,0,255,23,45,11
05,255,1,255,10,17,23
77,167,9,112,56,16,90
45,245,0,145,22,55,48
4 X 7
MNIST data set 28x28
28X1,784X1
14
211
0
255
23
45
11
05
255
1
255
10
17
23
77



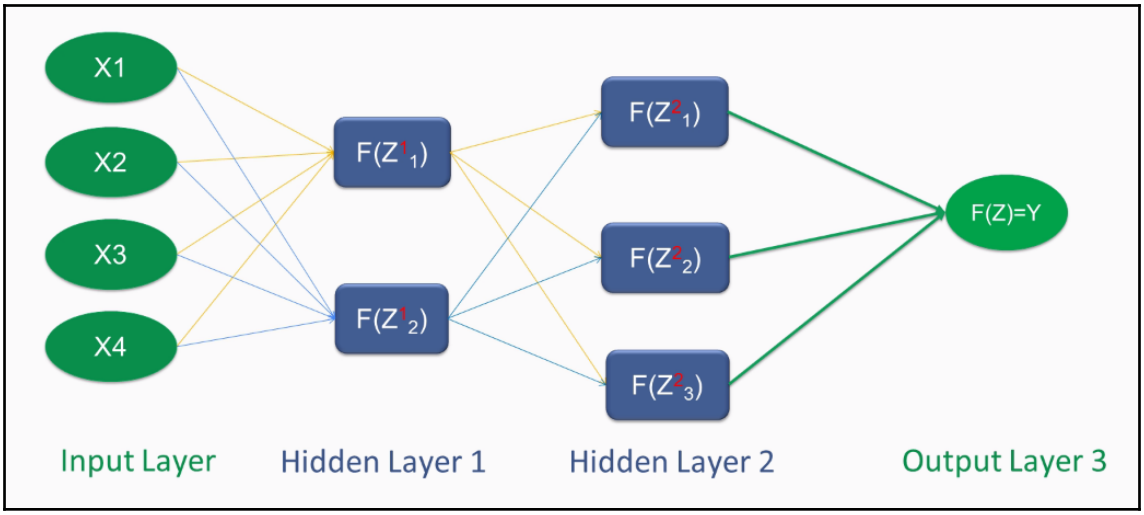


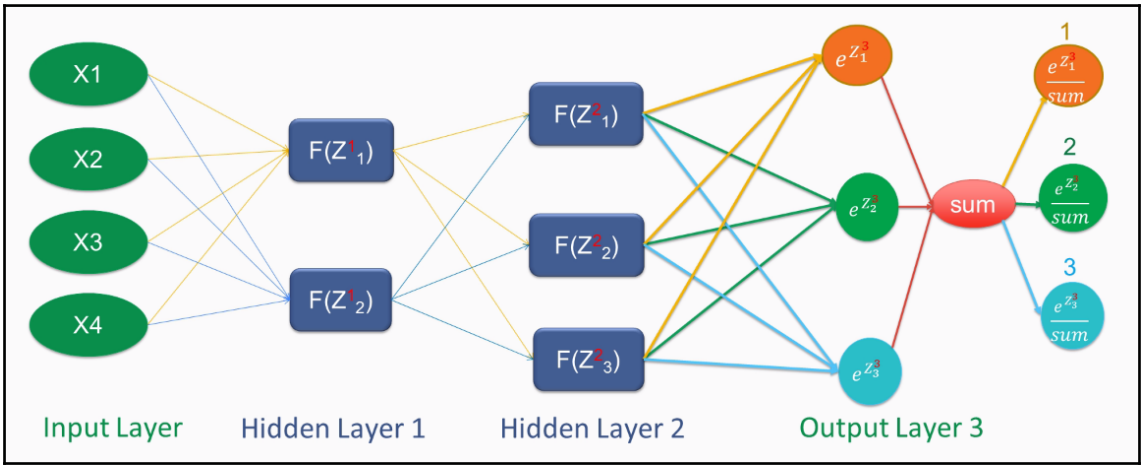


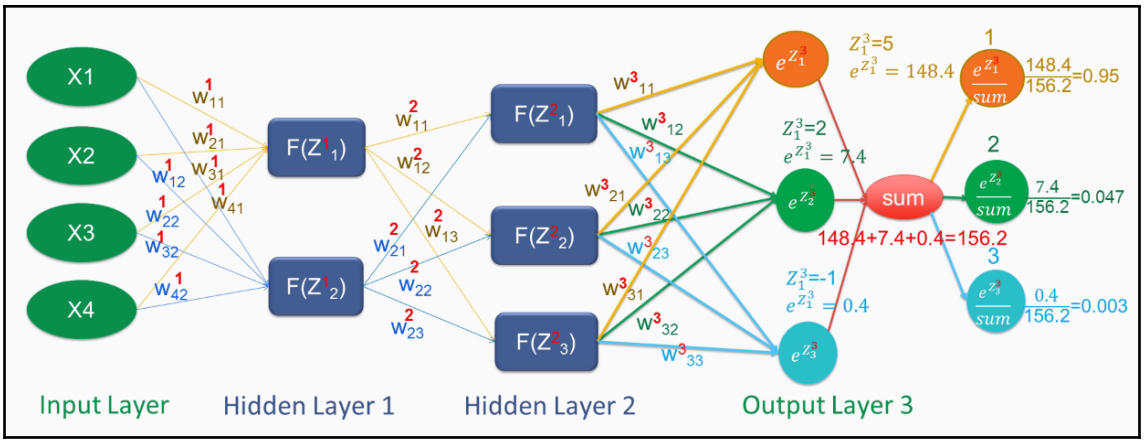
Will it have heart disease?
15% , 85%
Threshold→50%
1

0	1	2	3	4	5	6
26%	1%	8%	5%	38 %	12 %	10%

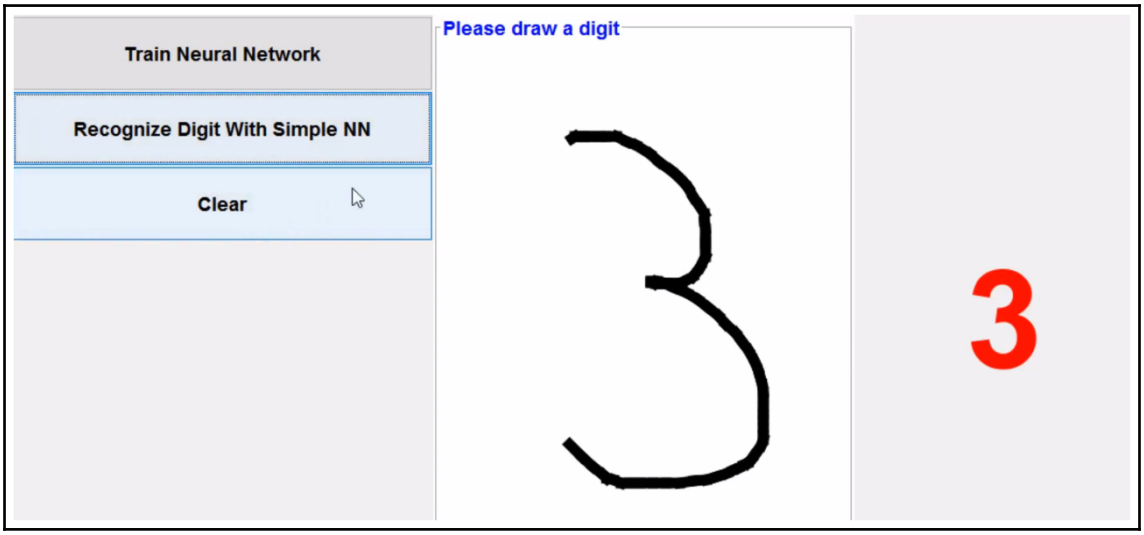
Take MAX



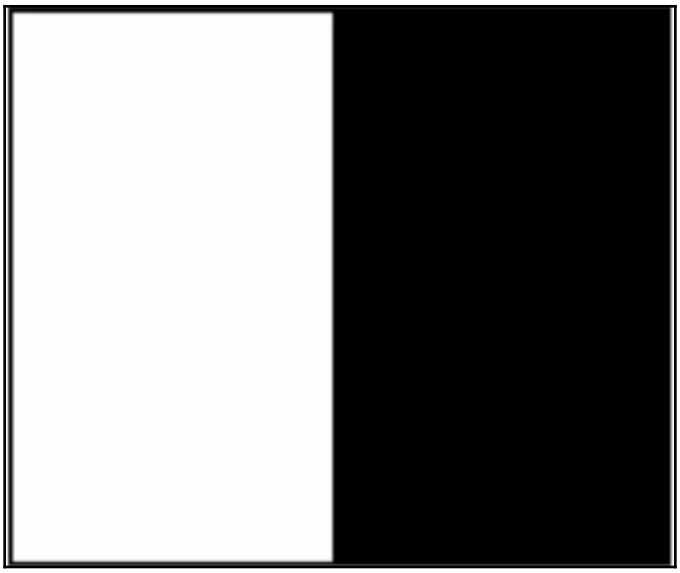




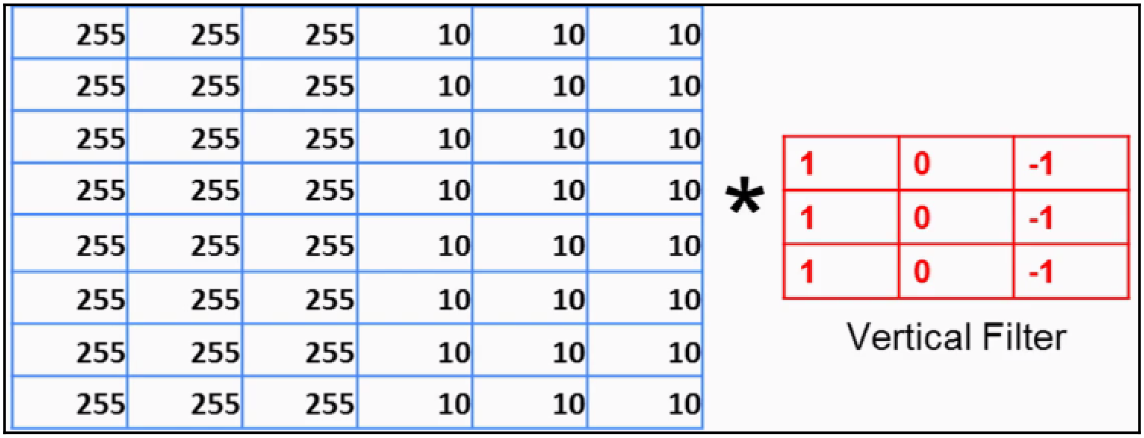

```
 # of classes:     10
 Accuracy:         0.9731
 Precision:        0.9731
 Recall:           0.9728
 F1 Score:         0.9729
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
================================================================================
[2018-04-07 01:55:04]Saving model at C:\devSources\ComputerVision\HandWrittenDigitReco
[2018-04-07 01:55:04]Congratulations,the desired score found,!
[2018-04-07 01:55:04]****************Example finished*******************

Process finished with exit code 0
```

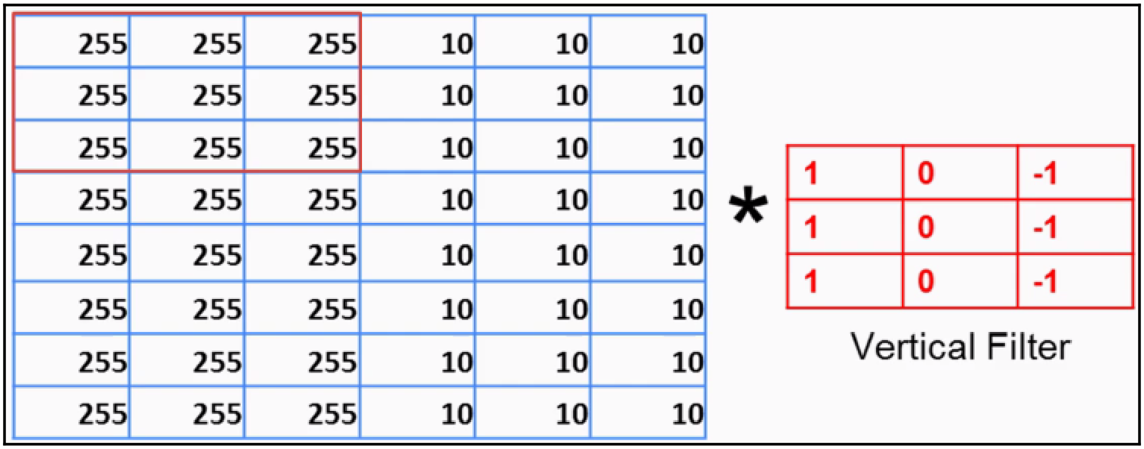


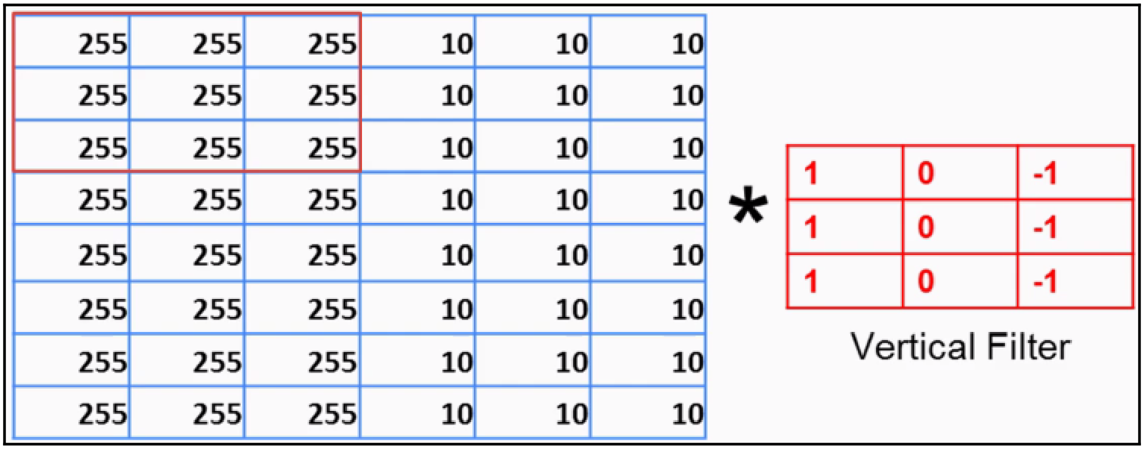
Chapter 2: Convolutional Neural Network Architectures

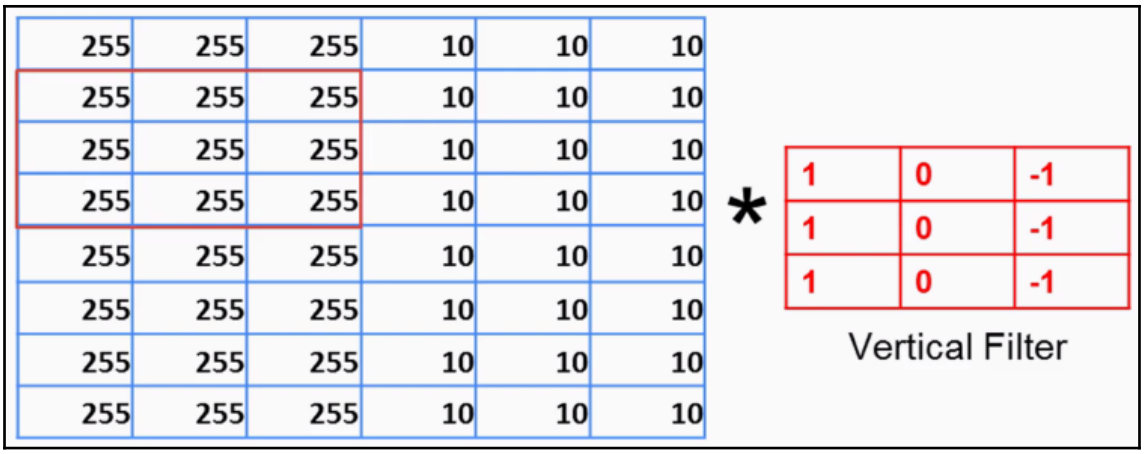


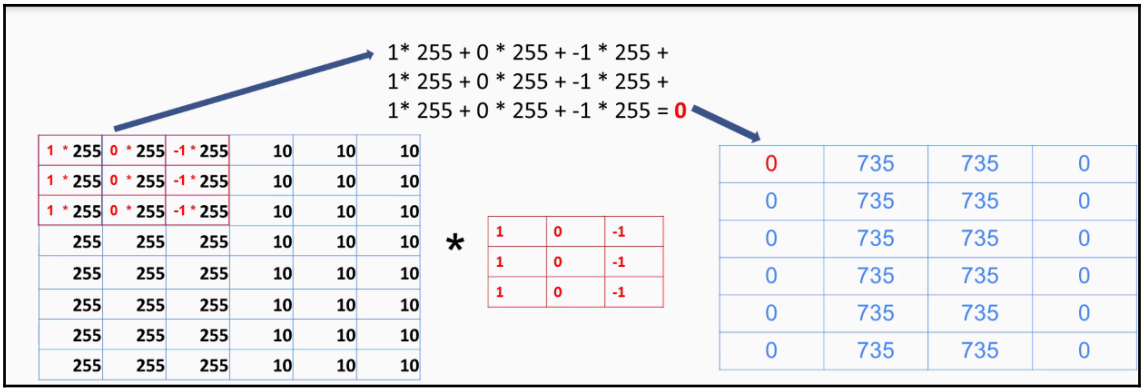
255	255	255	10	10	10
255	255	255	10	10	10
255	255	255	10	10	10
255	255	255	10	10	10
255	255	255	10	10	10
255	255	255	10	10	10
255	255	255	10	10	10
255	255	255	10	10	10

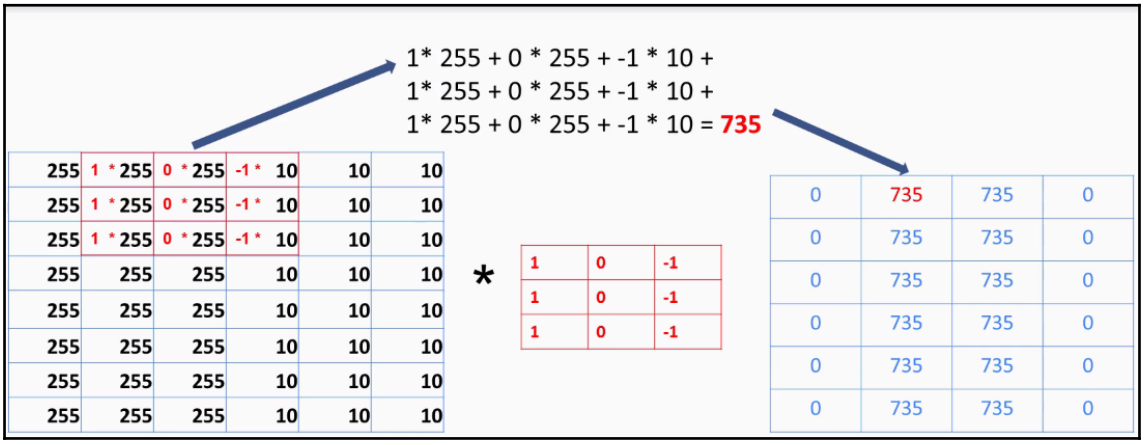

255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
Vertical Filter

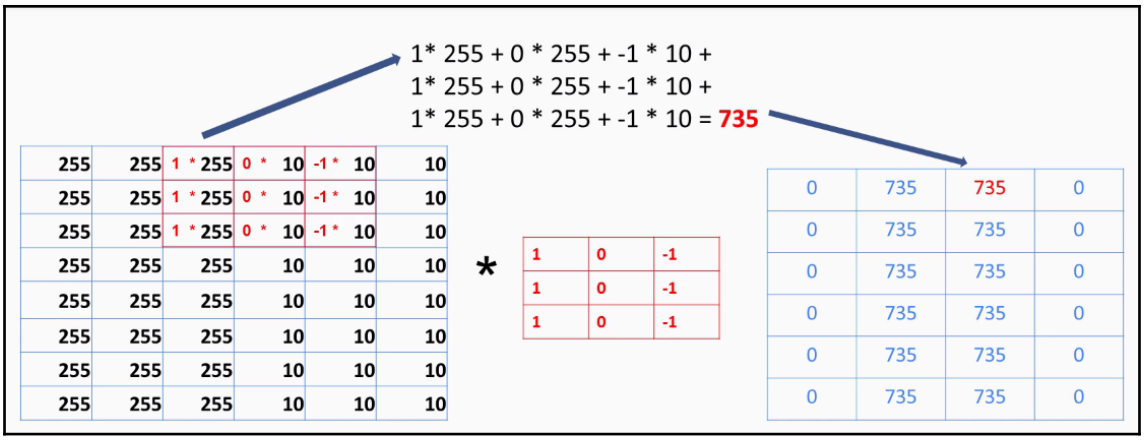

255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
Vertical Filter

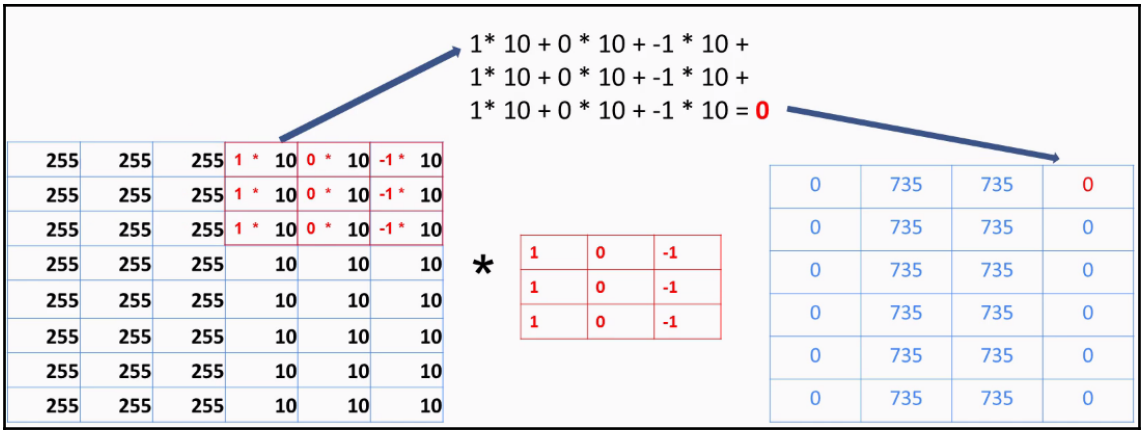

255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
Vertical Filter

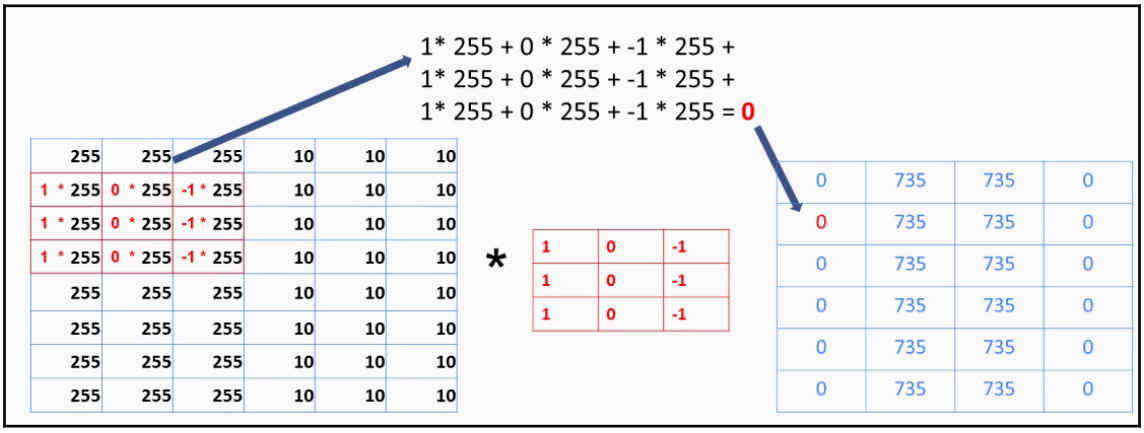

255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
Vertical Filter

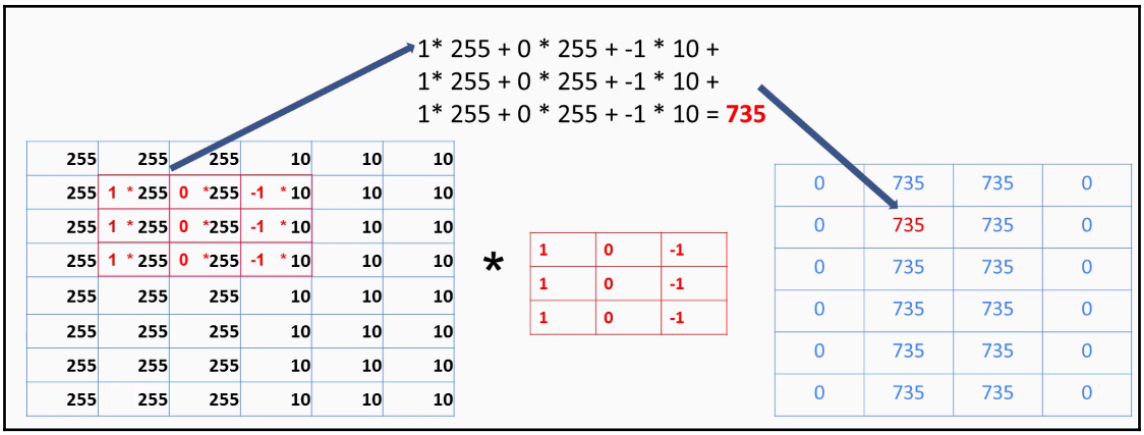

1* 255 + 0 * 255 + -1 * 255 +
1* 255 + 0 * 255 + -1 * 255 +
1* 255 + 0 * 255 + -1 * 255 = 0
1 * 255 0 * 255 -1 * 255 10 10 10
1 * 255 0 * 255 -1 * 255 10 10 10
1 * 255 0 * 255 -1 * 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0

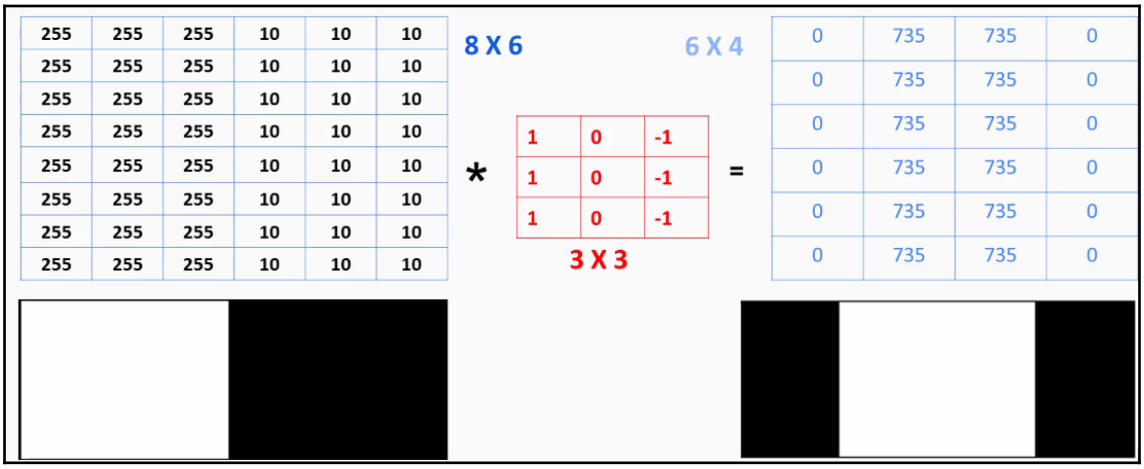

1* 255 + 0 * 255 + -1 * 10 +
1* 255 + 0 * 255 + -1 * 10 +
1* 255 + 0 * 255 + -1 * 10 = 735
255 1 * 255 0 * 255 -1 * 10 10 10
255 1 * 255 0 * 255 -1 * 10 10 10
255 1 * 255 0 * 255 -1 * 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0


1* 255 + 0 * 255 + -1 * 10 +
1* 255 + 0 * 255 + -1 * 10 +
1* 255 + 0 * 255 + -1 * 10 = 735
255 255 1 * 255 0 * 10 -1 * 10 10
255 255 1 * 255 0 * 10 -1 * 10 10
255 255 1 * 255 0 * 10 -1 * 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0


1* 10 + 0 * 10 + -1 * 10 +
1* 10 + 0 * 10 + -1 * 10 +
1* 10 + 0 * 10 + -1 * 10 = 0
255 255 255 1 * 10 0 * 10 -1 * 10
255 255 255 1 * 10 0 * 10 -1 * 10
255 255 255 1 * 10 0 * 10 -1 * 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0


1* 255 + 0 * 255 + -1 * 255 +
1* 255 + 0 * 255 + -1 * 255 +
1* 255 + 0 * 255 + -1 * 255 = 0
255 255 255 10 10 10
1 * 255 0 * 255 -1 * 255 10 10 10
1 * 255 0 * 255 -1 * 255 10 10 10
1 * 255 0 * 255 -1 * 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0


1* 255 + 0 * 255 + -1 * 10 +
1* 255 + 0 * 255 + -1 * 10 +
1* 255 + 0 * 255 + -1 * 10 = 735
255 255 255 10 10 10
255 1 * 255 0 * 255 -1 * 10 10 10
255 1 * 255 0 * 255 -1 * 10 10 10
255 1 * 255 0 * 255 -1 * 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
*
1 0 -1
1 0 -1
1 0 -1
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0


255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
255 255 255 10 10 10
8 X 6
6 X 4
*
1 0 -1
1 0 -1
1 0 -1
3 X 3
=
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0
0 735 735 0

255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255
10	10	10	10	10	10
10	10	10	10	10	10
10	10	10	10	10	10
10	10	10	10	10	10

8 X 6

*

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

3 X 3

=

6 X 4

0	0	0	0
0	0	0	0
735	735	735	735
735	735	735	735
0	0	0	0
0	0	0	0

[redacted]

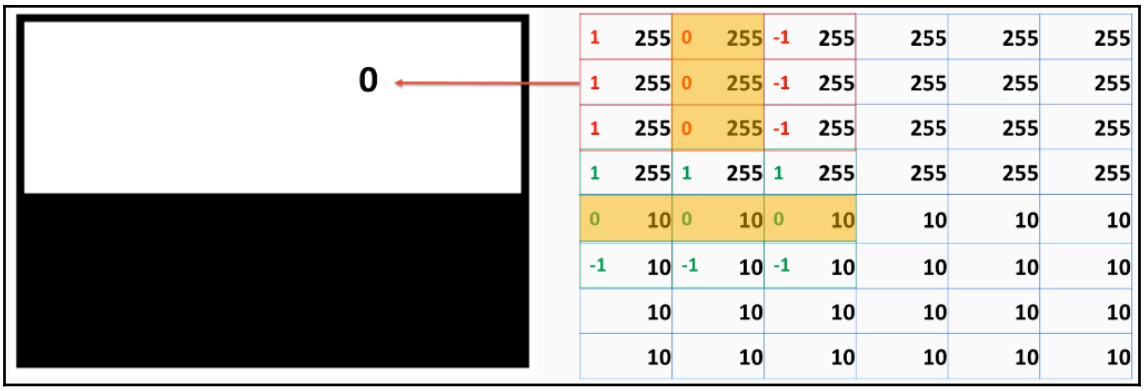
[redacted]

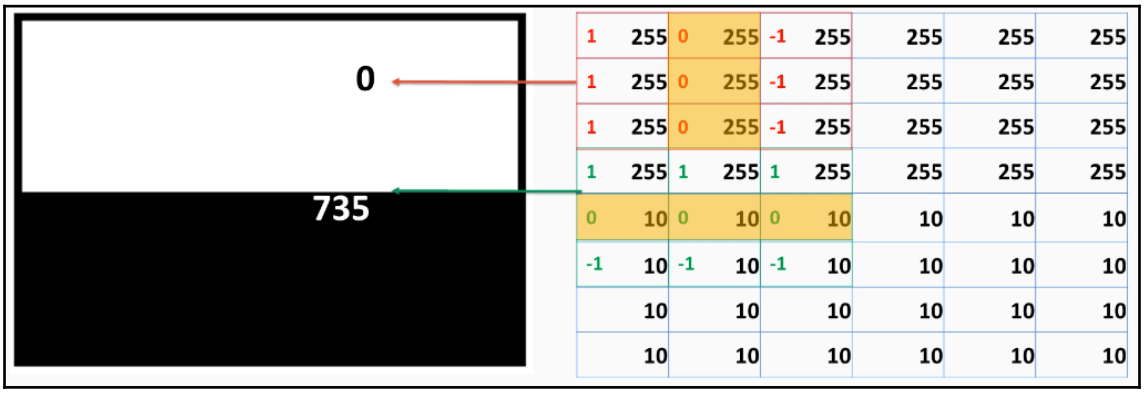
[redacted]

1	255	0	255	-1	255	255	255	255
1	255	0	255	-1	255	255	255	255
1	255	0	255	-1	255	255	255	255
1	255	1	255	1	255	255	255	255
0	10	0	10	0	10	10	10	10
-1	10	-1	10	-1	10	10	10	10
	10		10		10	10	10	10
	10		10		10	10	10	10

[redacted]

1	255	0	255	-1	255	255	255	255
1	255	0	255	-1	255	255	255	255
1	255	0	255	-1	255	255	255	255
1	255	1	255	1	255	255	255	255
0	10	0	10	0	10	10	10	10
-1	10	-1	10	-1	10	10	10	10
	10		10		10	10	10	10
	10		10		10	10	10	10





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

Vertical

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

Sobel Vertical

3	0	-3
10	0	-10
3	0	-3

Scharr Vertical

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

Horizontal

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

Vertical

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

Sobel Vertical

3	0	-3
10	0	-10
3	0	-3

Scharr Vertical

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

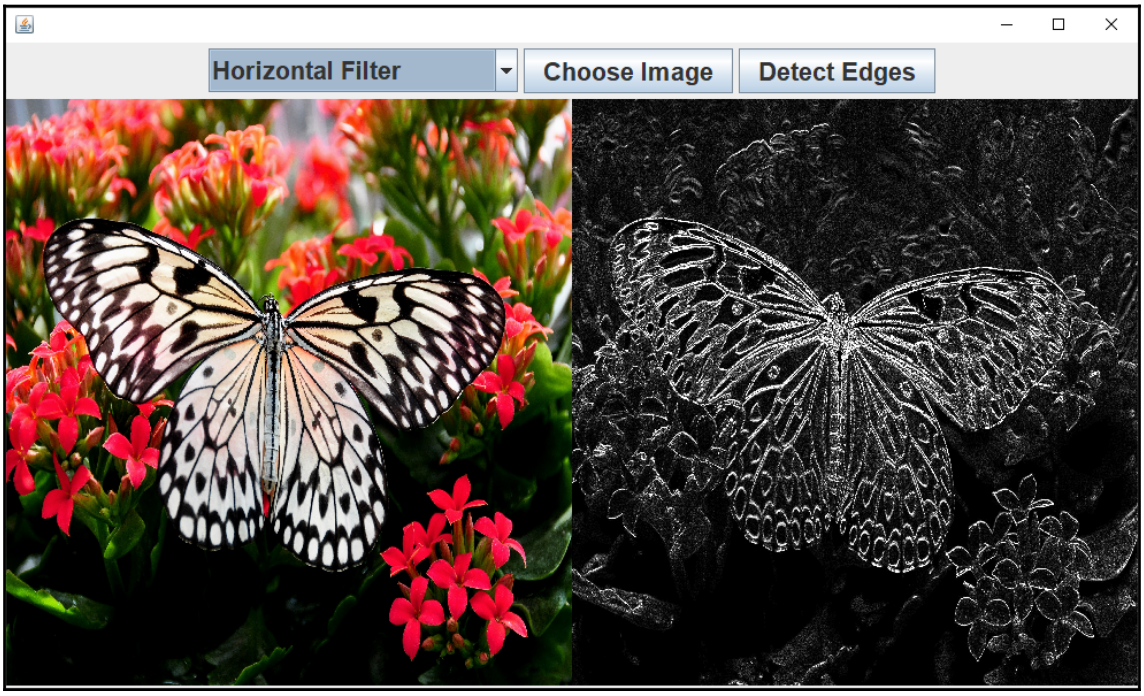
Horizontal

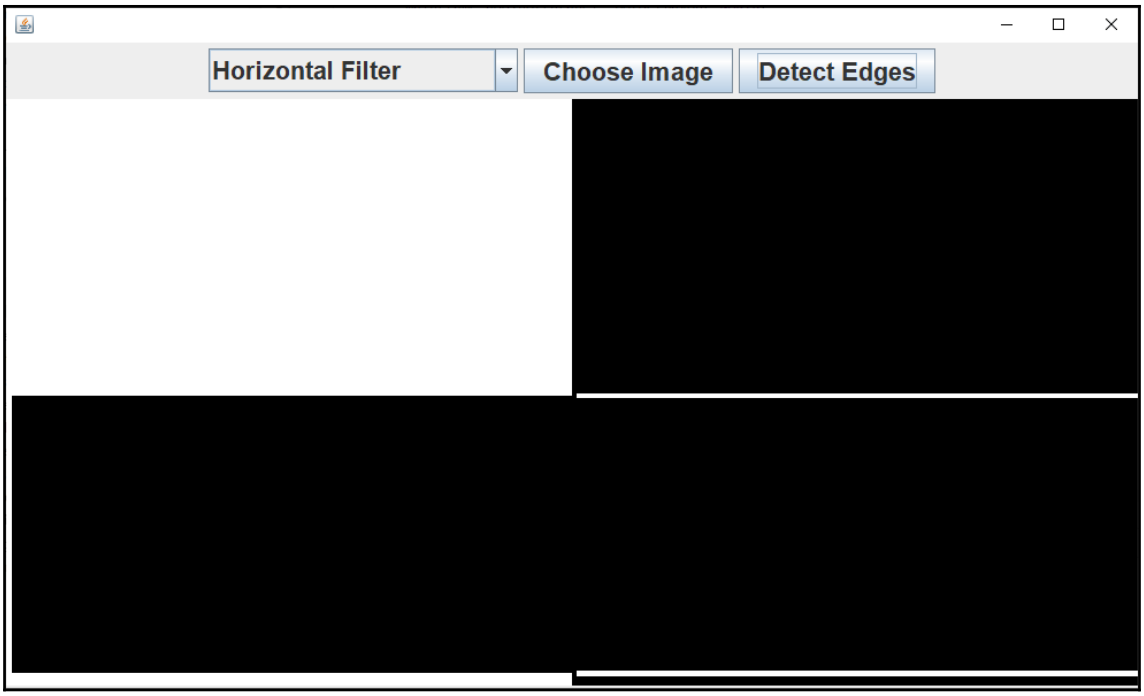
1	2	1
0	0	0
-1	-2	-1

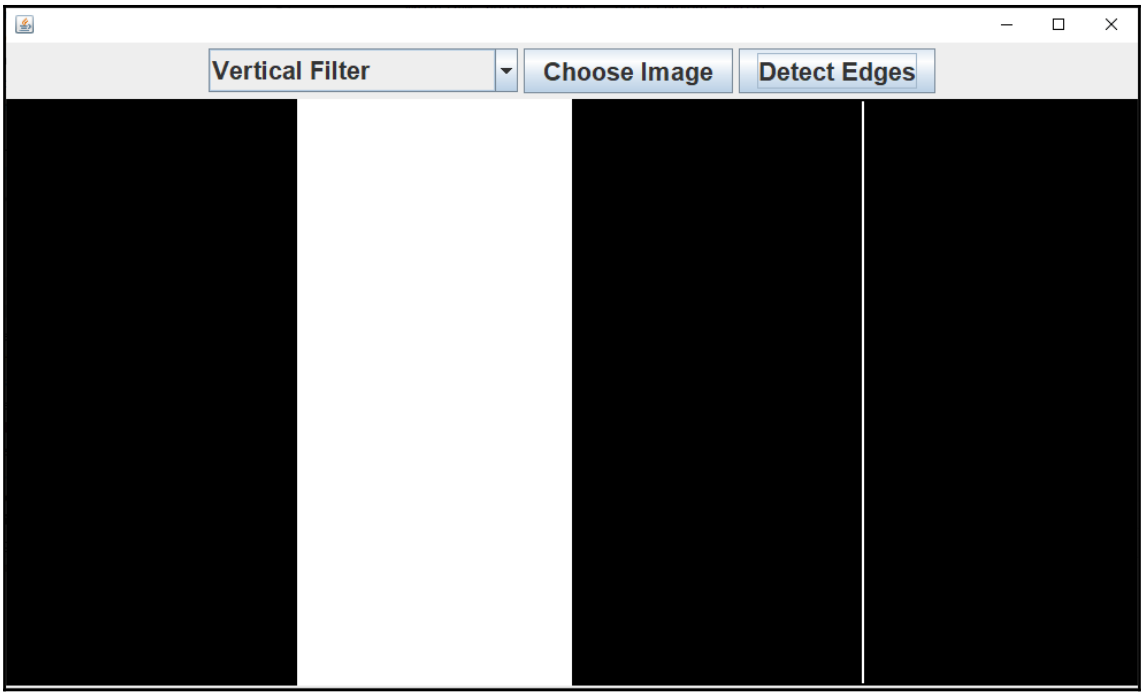
Sobel Horizontal

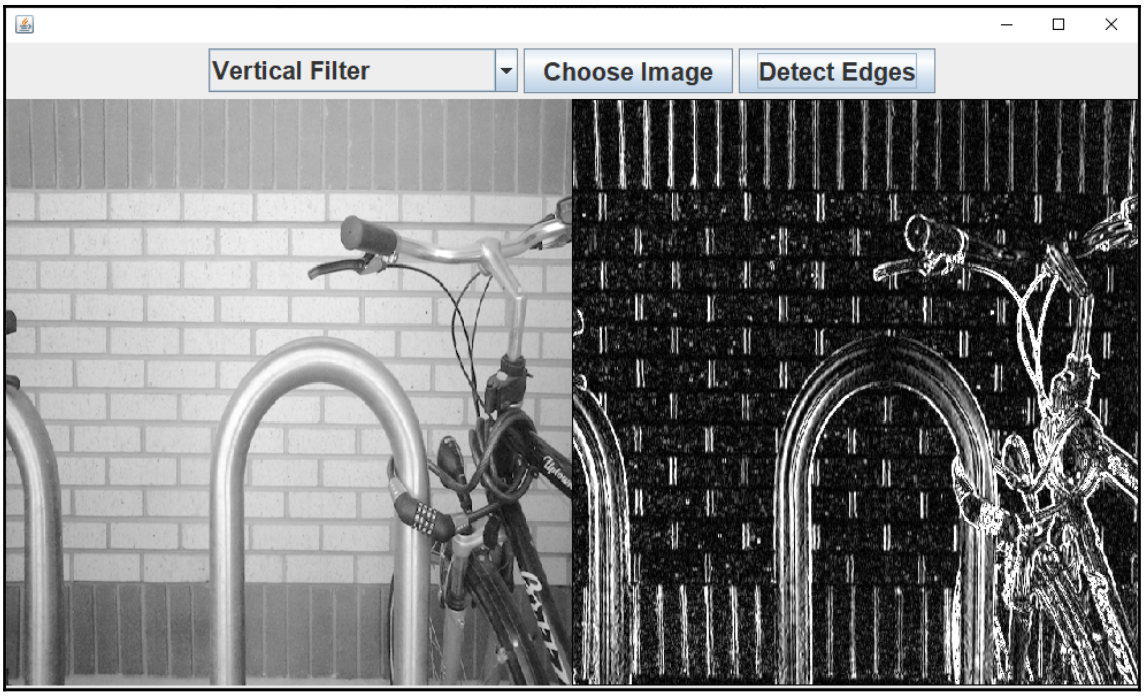
3	10	3
0	0	0
-3	-10	-3

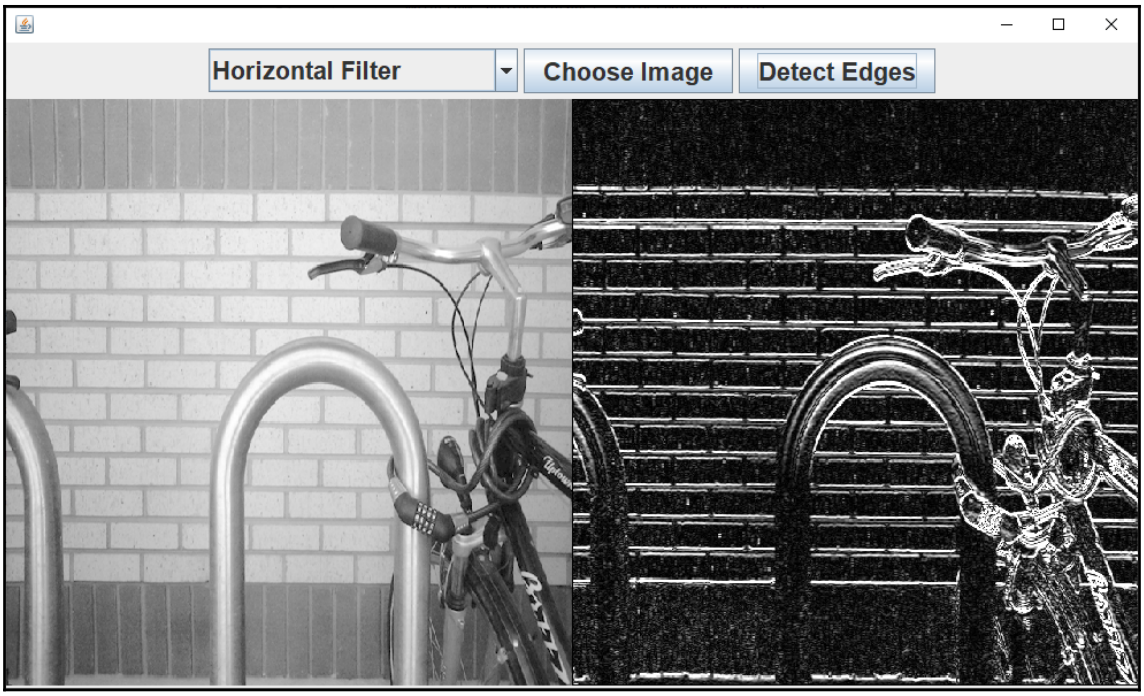
Scharr Horizontal

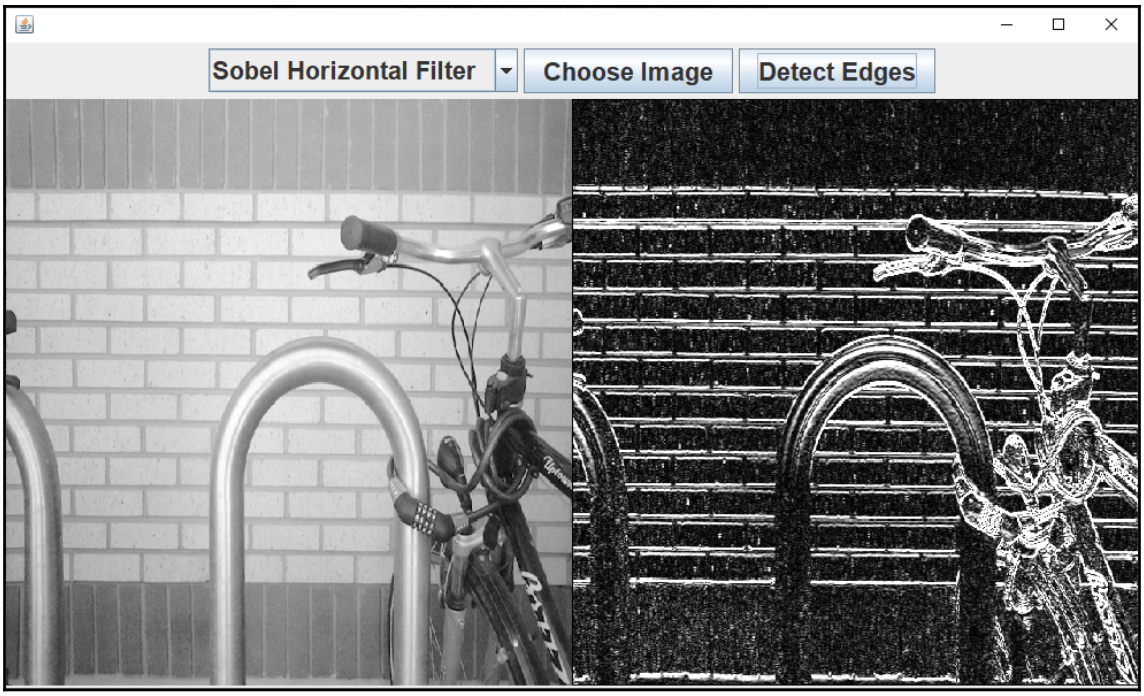


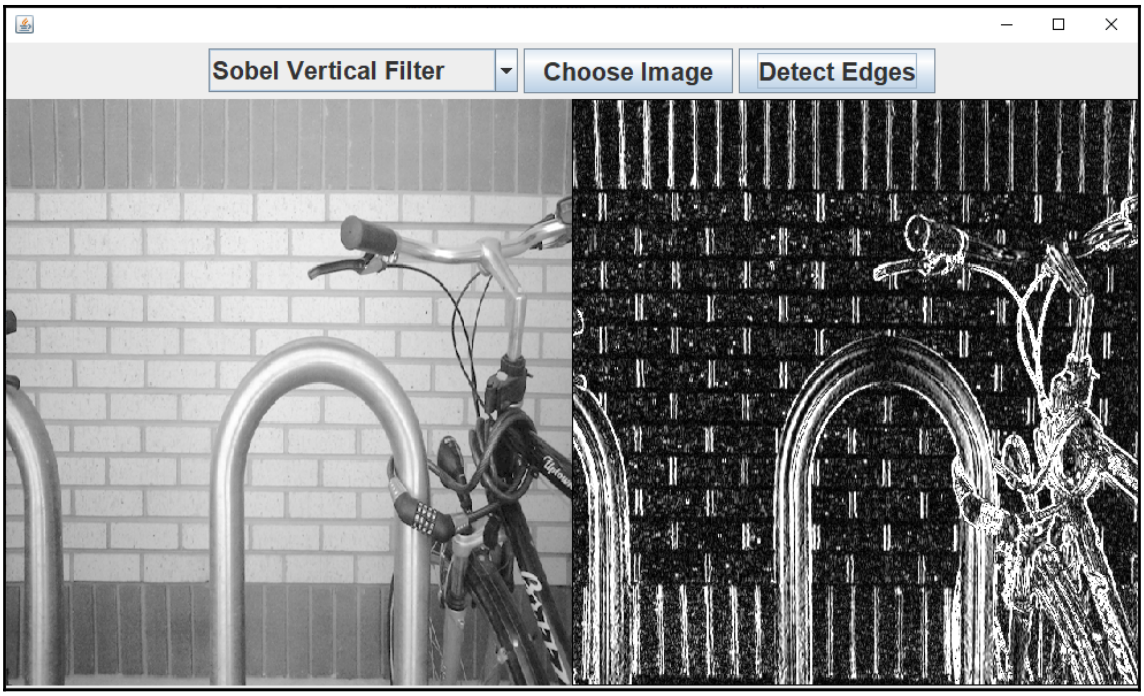

Horizontal Filter
Choose Image
Detect Edges

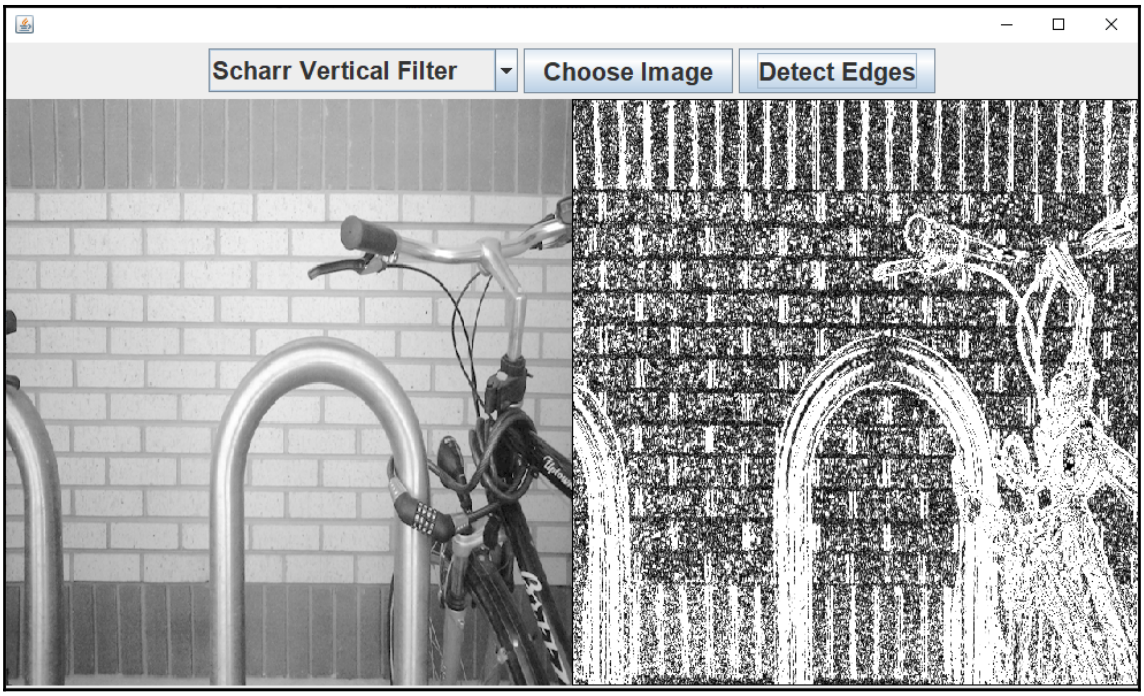

Vertical Filter
Choose Image
Detect Edges

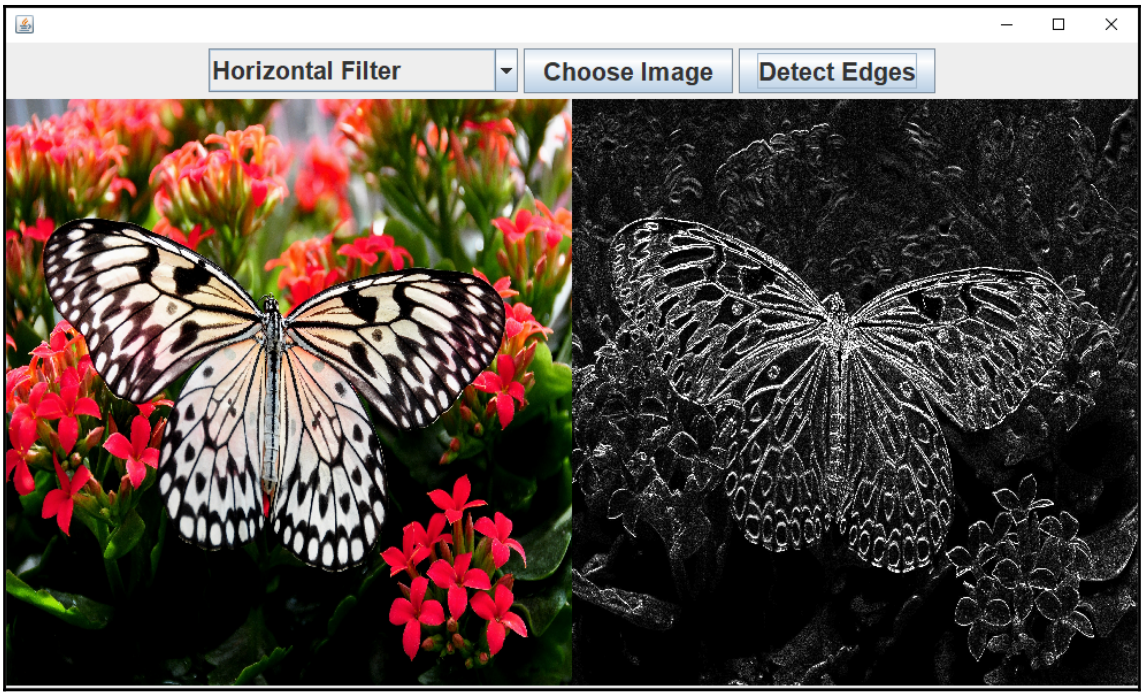

Vertical Filter
Choose Image
Detect Edges

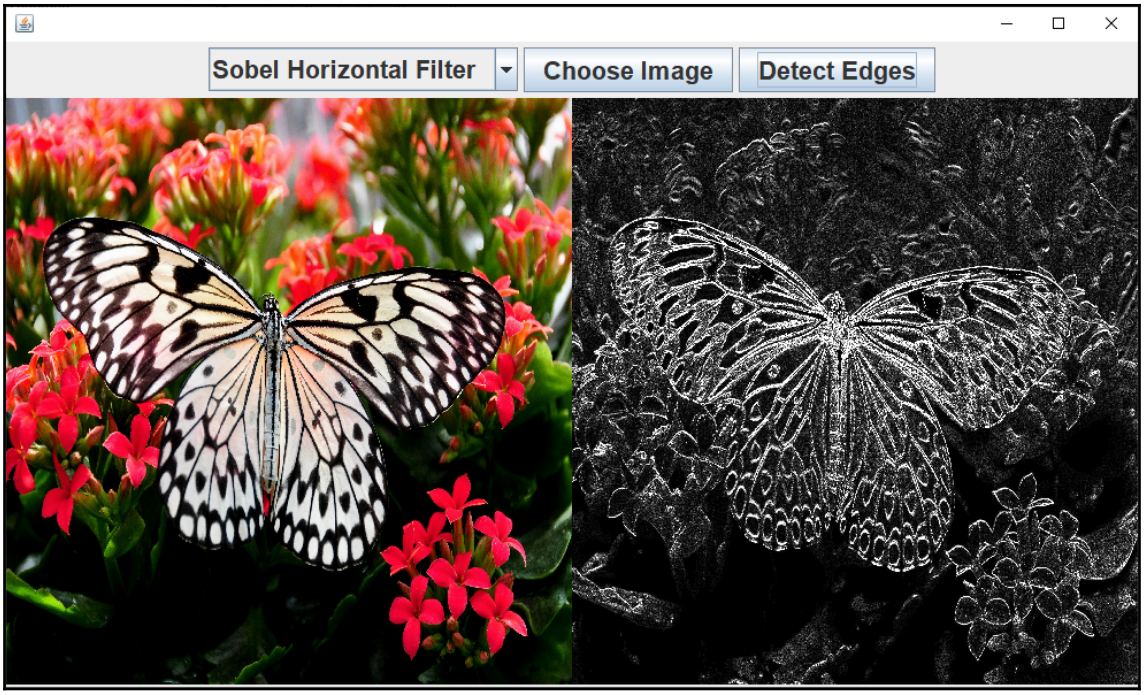

Horizontal Filter
Choose Image
Detect Edges

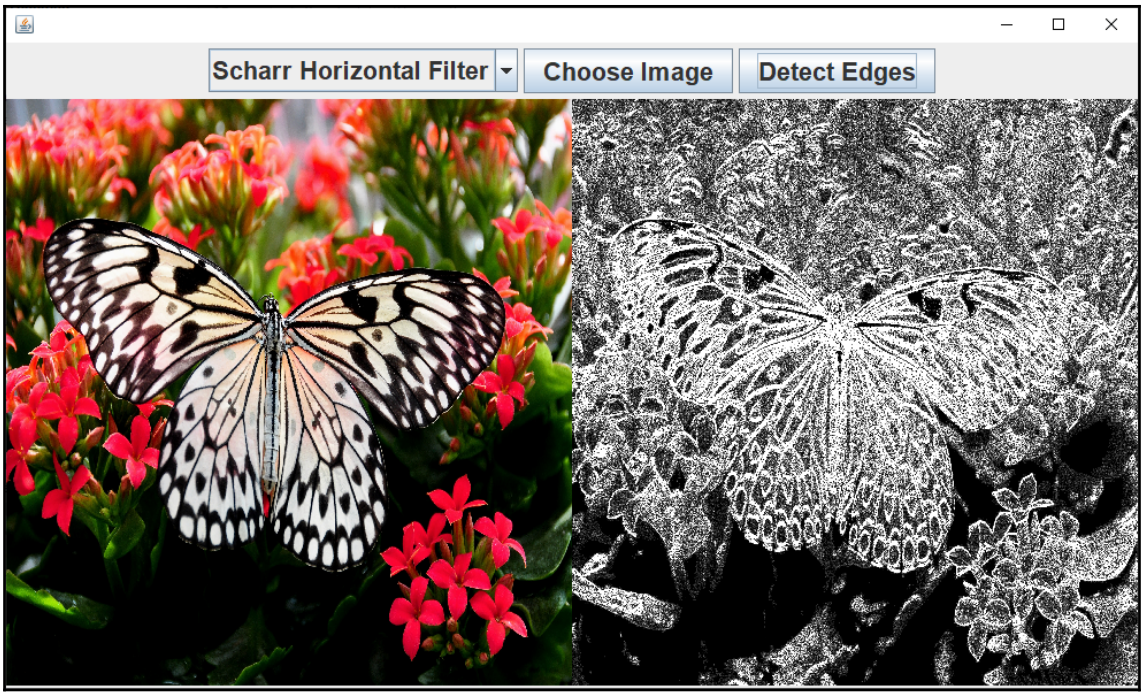

Sobel Horizontal Filter
Choose Image
Detect Edges

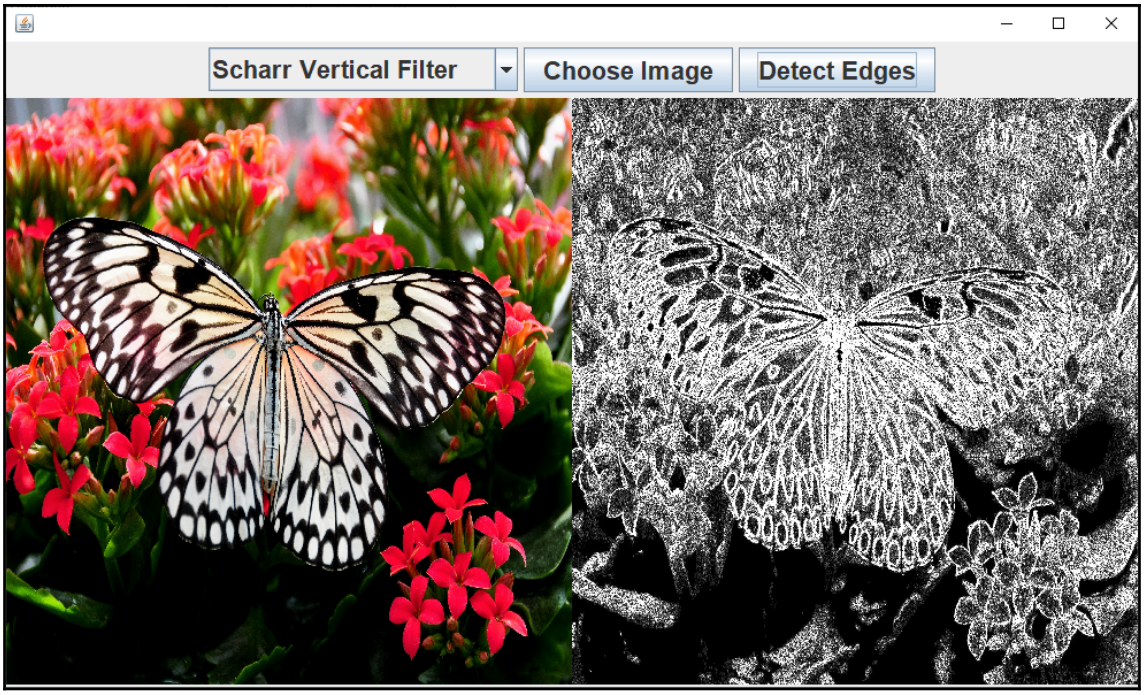

Sobel Vertical Filter
Choose Image
Detect Edges


Scharr Vertical Filter
Choose Image
Detect Edges


Horizontal Filter
Choose Image
Detect Edges


Sobel Horizontal Filter
Choose Image
Detect Edges


Scharr Horizontal Filter
Choose Image
Detect Edges



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

Horizontal

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

Vertical

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

Sobel Vertical

3	0	-3
10	0	-10
3	0	-3

Scharr Vertical

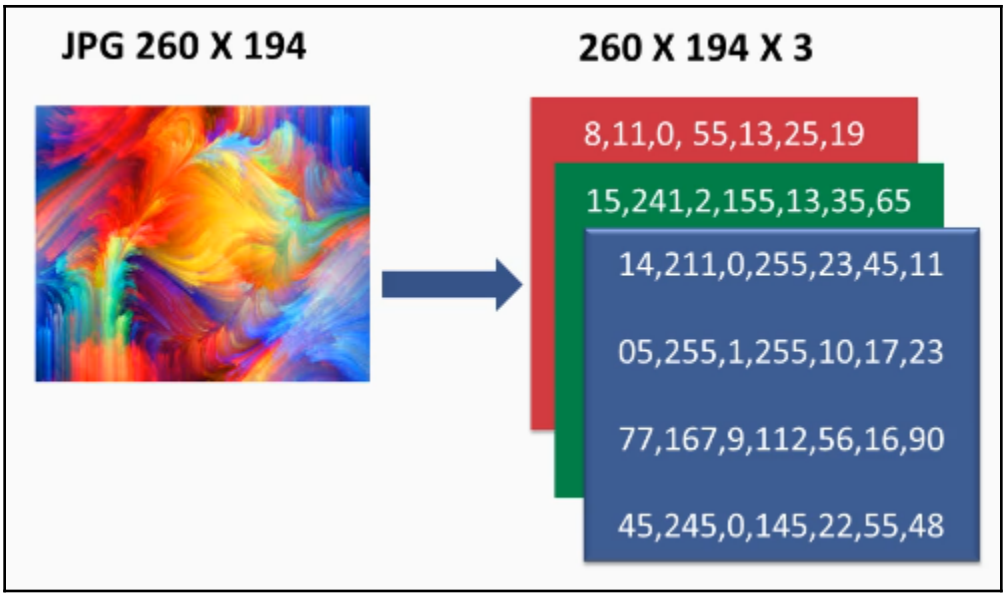
W11	W12	W13
W21	W22	W23
W31	W32	W33

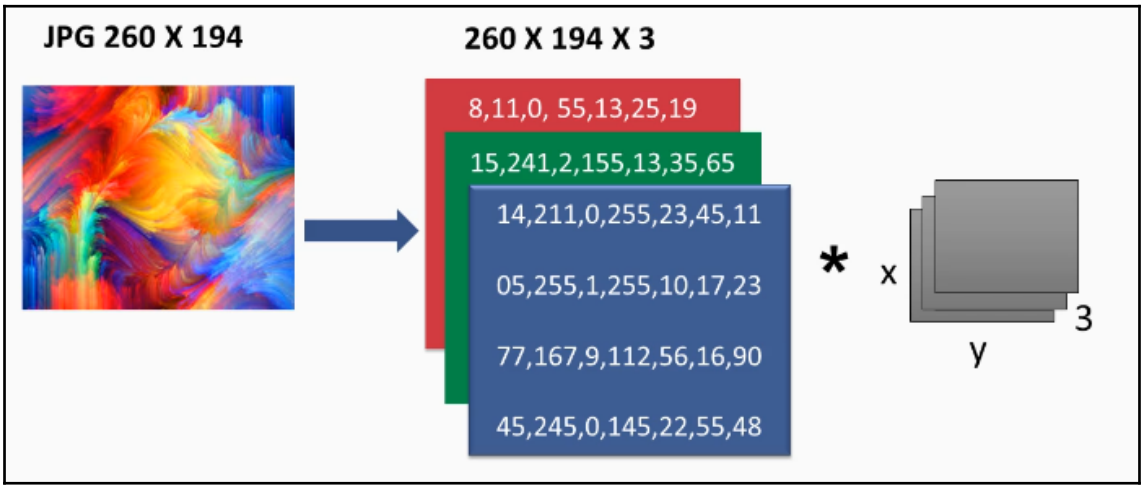
1	2	1
0	0	-
-1	-2	-1

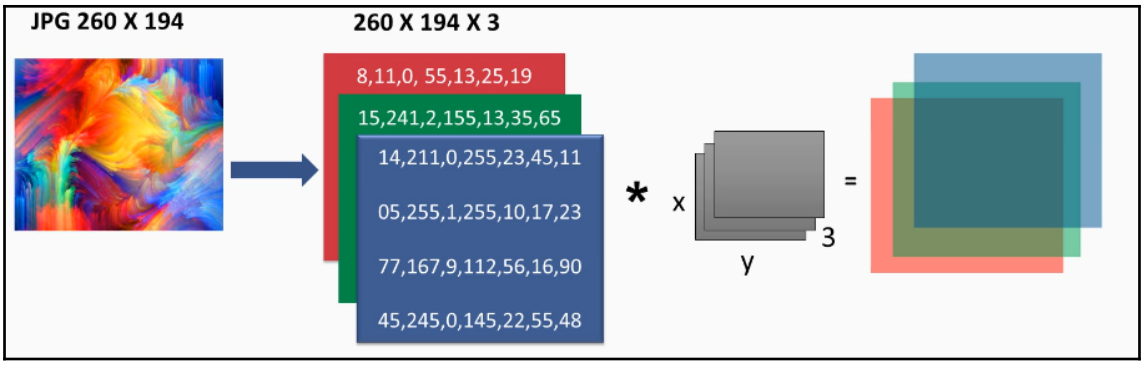
Sobel Horizontal

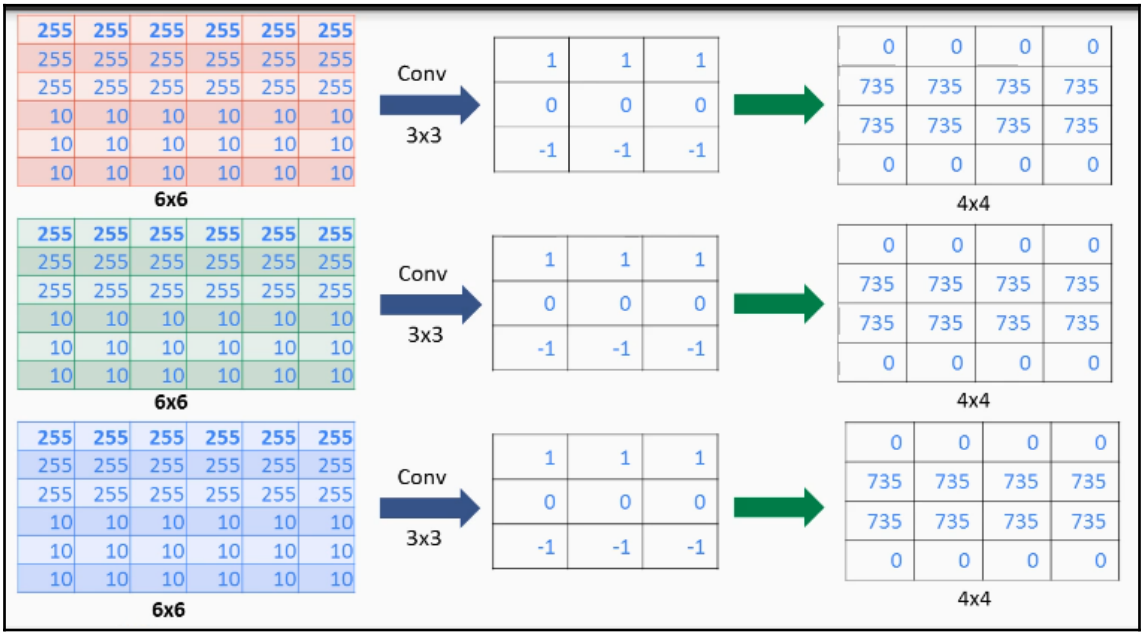
3	0	-3
10	0	-10
3	0	-3

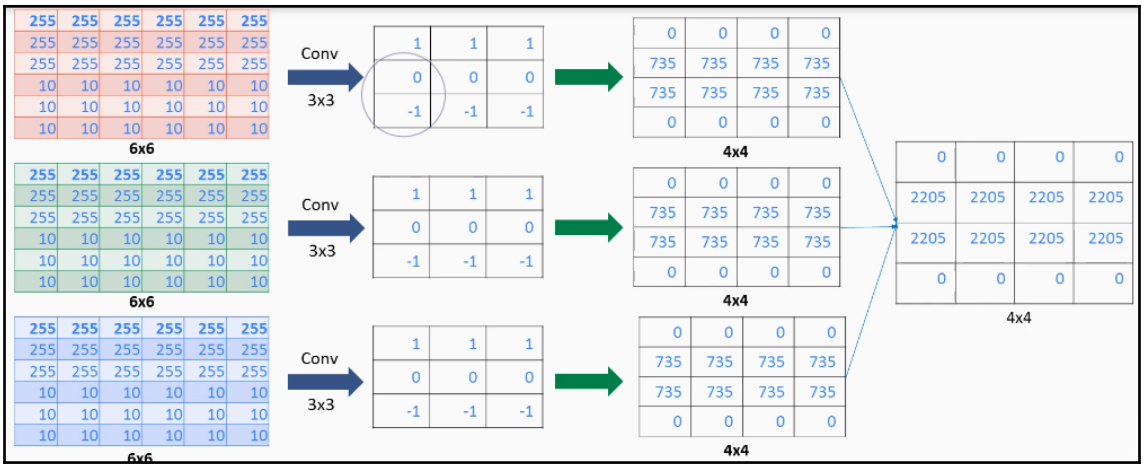
Scharr Horizontal

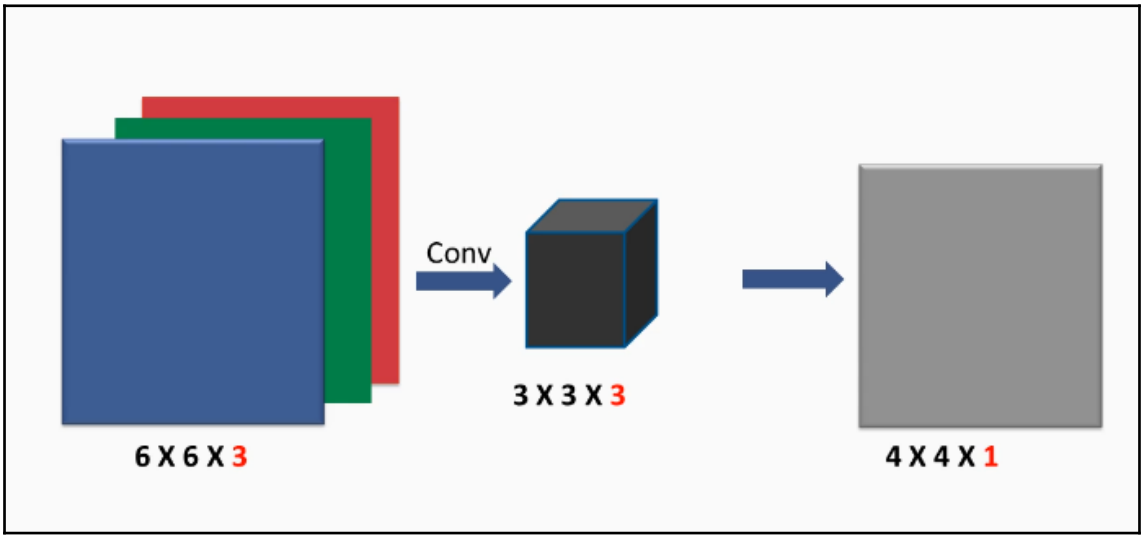

JPG 260 X 194
260 X 194 X 3
8,11,0, 55,13,25,19
15,241,2,155,13,35,65
14,211,0,255,23,45,11
05,255,1,255,10,17,23
77,167,9,112,56,16,90
45,245,0,145,22,55,48

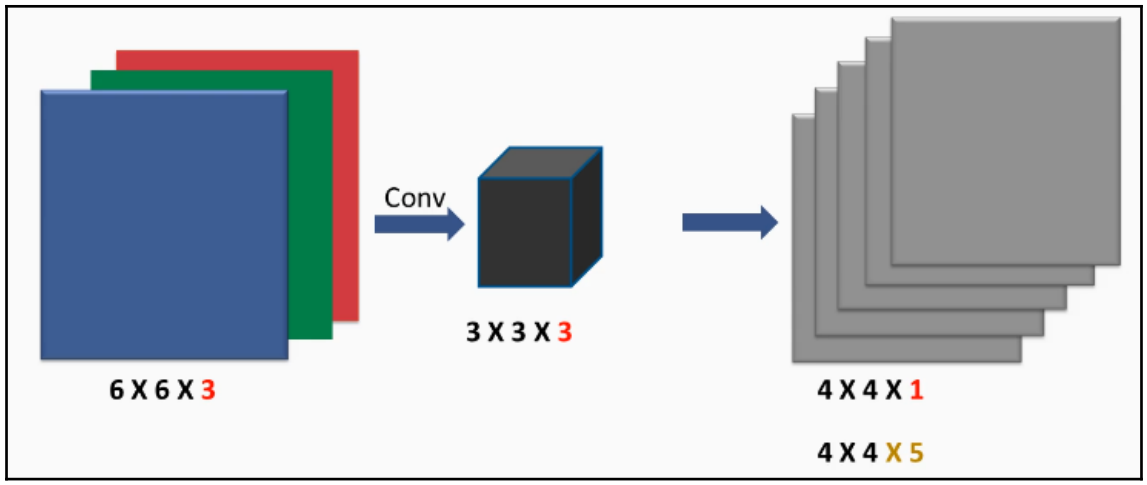

JPG 260 X 194
260 X 194 X 3
8,11,0, 55,13,25,19
15,241,2,155,13,35,65
14,211,0,255,23,45,11
05,255,1,255,10,17,23
77,167,9,112,56,16,90
45,245,0,145,22,55,48
*
x
y
3

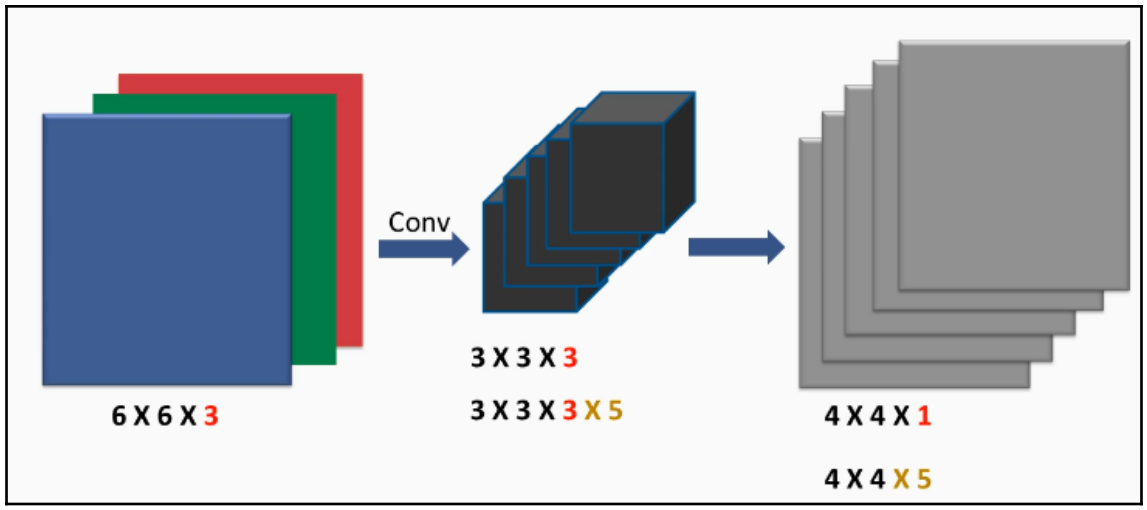

JPG 260 X 194
260 X 194 X 3
8,11,0, 55,13,25,19
15,241,2,155,13,35,65
14,211,0,255,23,45,11
05,255,1,255,10,17,23
77,167,9,112,56,16,90
45,245,0,145,22,55,48
*
x
y
3
=

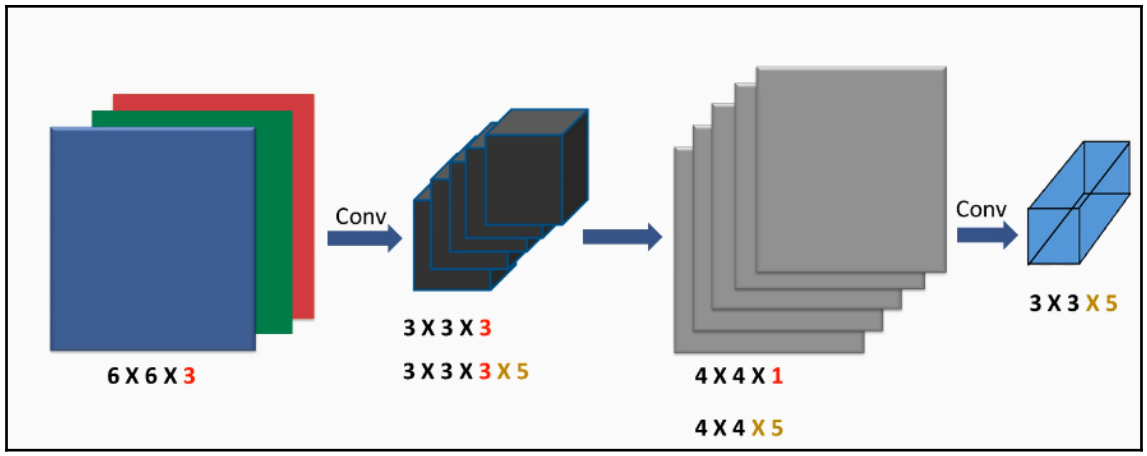

255 255 255 255 255 255
255 255 255 255 255 255
255 255 255 255 255 255
10 10 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
6x6
Conv
3x3
1 1 1
0 0 0
-1 -1 -1
0 0 0 0
735 735 735 735
735 735 735 735
0 0 0 0
4x4
255 255 255 255 255 255
255 255 255 255 255 255
255 255 255 255 255 255
10 10 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
6x6
Conv
3x3
1 1 1
0 0 0
-1 -1 -1
0 0 0 0
735 735 735 735
735 735 735 735
0 0 0 0
4x4
255 255 255 255 255 255
255 255 255 255 255 255
255 255 255 255 255 255
10 10 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
6x6
Conv
3x3
1 1 1
0 0 0
-1 -1 -1
0 0 0 0
735 735 735 735
735 735 735 735
0 0 0 0
4x4

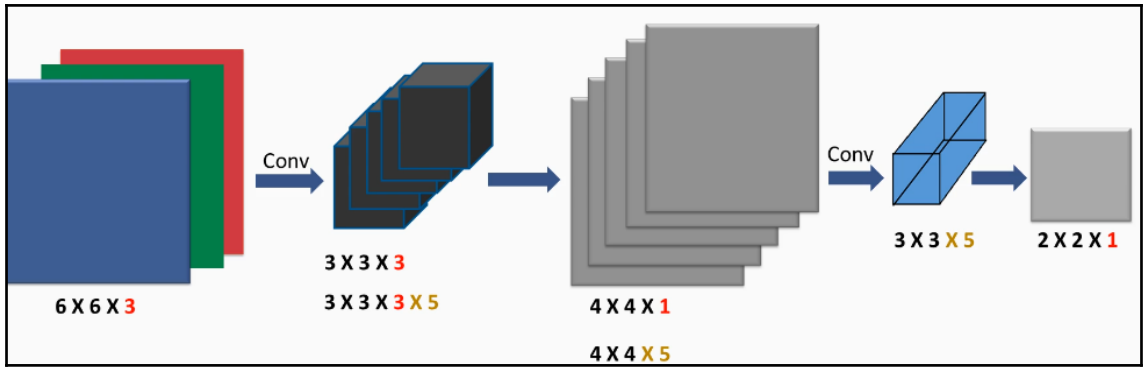

255 255 255 255 255 255
255 255 255 255 255 255
255 255 255 255 255 255
10 10 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
6x6
Conv
3x3
1 1 1
0 0 0
-1 -1 -1
0 0 0 0
735 735 735 735
735 735 735 735
0 0 0 0
4x4
0 0 0 0
2205 2205 2205 2205
2205 2205 2205 2205
0 0 0 0
4x4

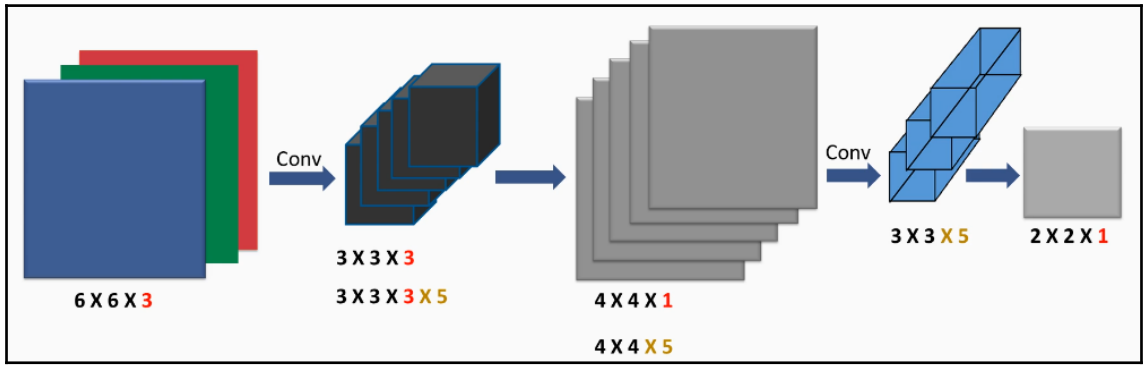

Conv
6 X 6 X 3
3 X 3 X 3
4 X 4 X 1

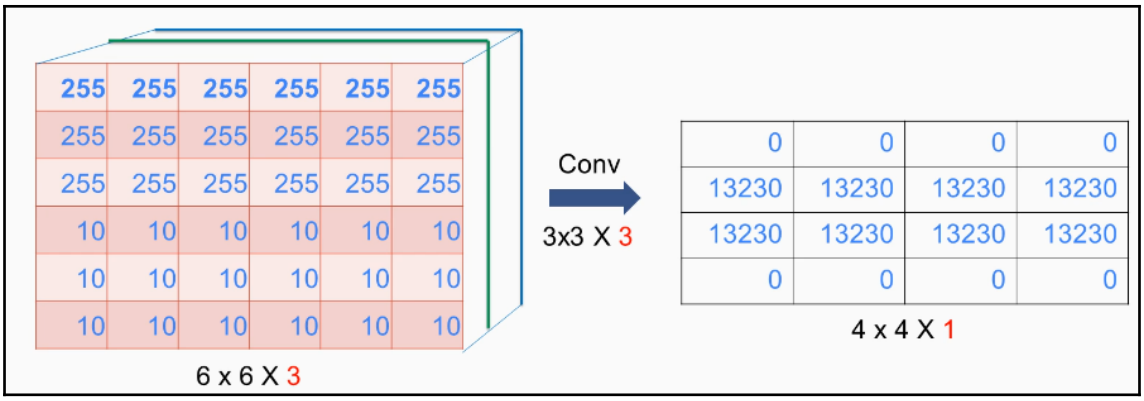

Conv
6 X 6 X 3
3 X 3 X 3
4 X 4 X 1
4 X 4 X 5

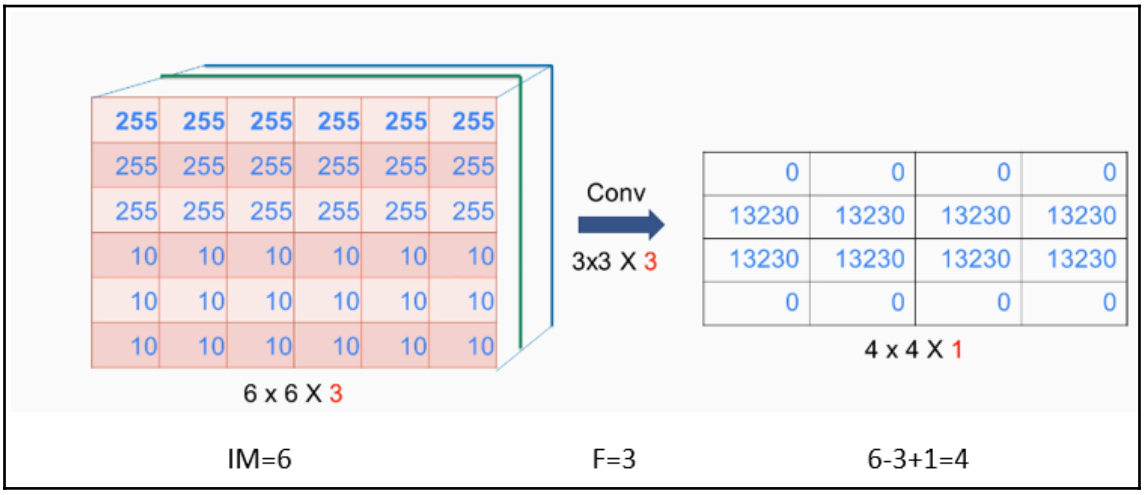

Conv
6 X 6 X 3
3 X 3 X 3
3 X 3 X 3 X 5
4 X 4 X 1
4 X 4 X 5

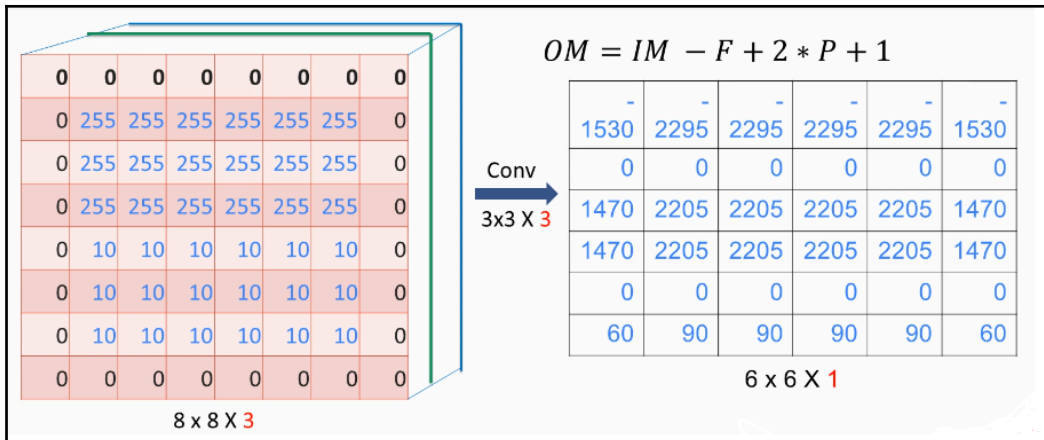

Conv
Conv
6 X 6 X 3
3 X 3 X 3
3 X 3 X 3 X 5
4 X 4 X 1
4 X 4 X 5
3 X 3 X 5

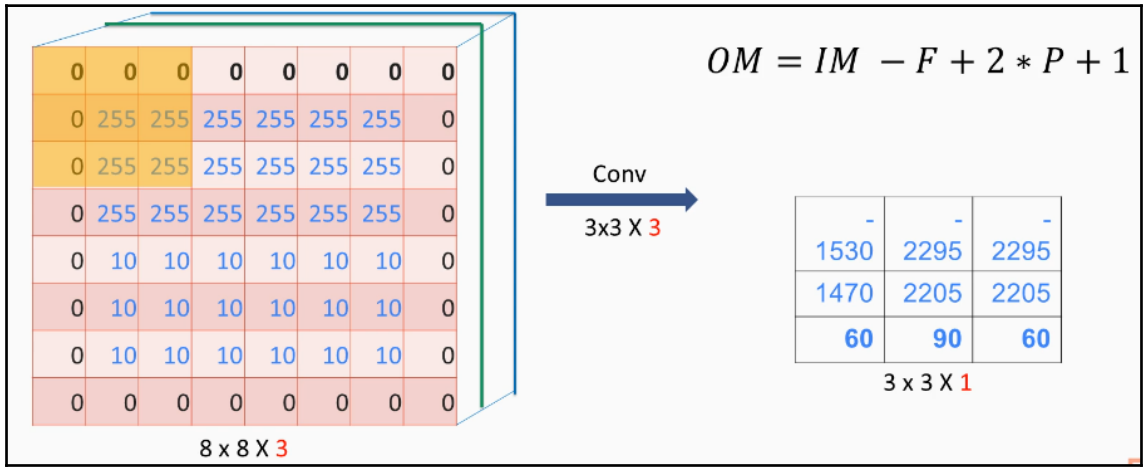

Conv
Conv
6 X 6 X 3
3 X 3 X 3
3 X 3 X 3 X 5
4 X 4 X 1
4 X 4 X 5
3 X 3 X 5
2 X 2 X 1

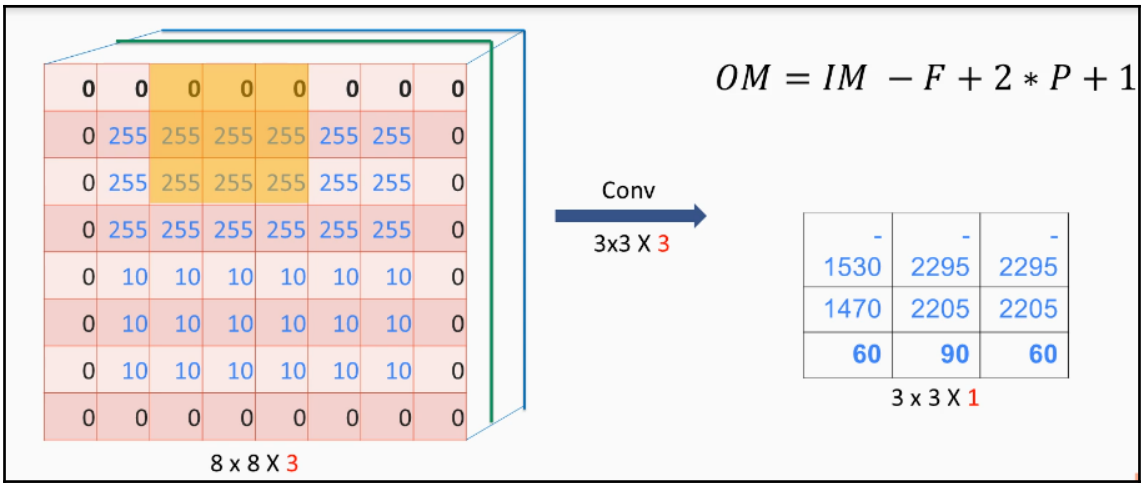

Conv
Conv
6 X 6 X 3
3 X 3 X 3
3 X 3 X 3 X 5
4 X 4 X 1
4 X 4 X 5
3 X 3 X 5
2 X 2 X 1

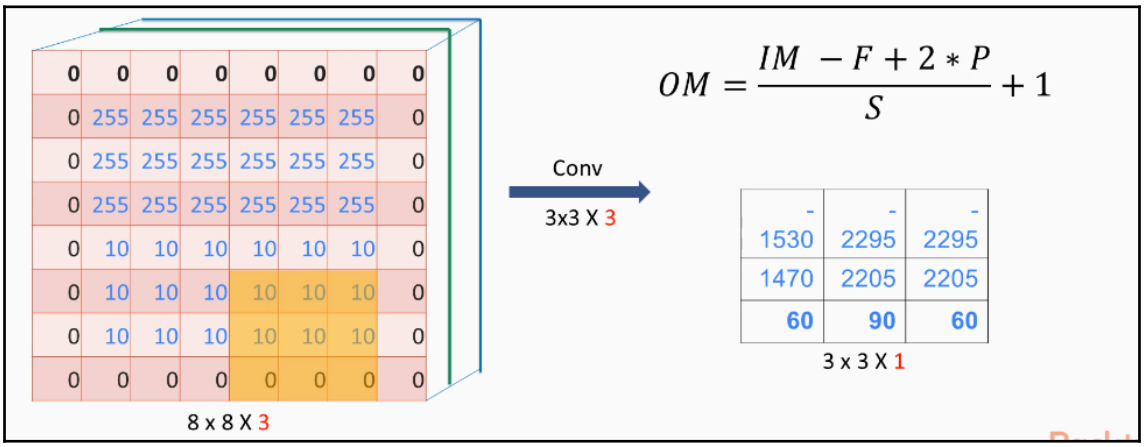

255 255 255 255 255 255
255 255 255 255 255 255
255 255 255 255 255 255
10 10 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
6 x 6 X 3
Conv
3x3 X 3
0 0 0 0
13230 13230 13230 13230
13230 13230 13230 13230
0 0 0 0
4 x 4 X 1

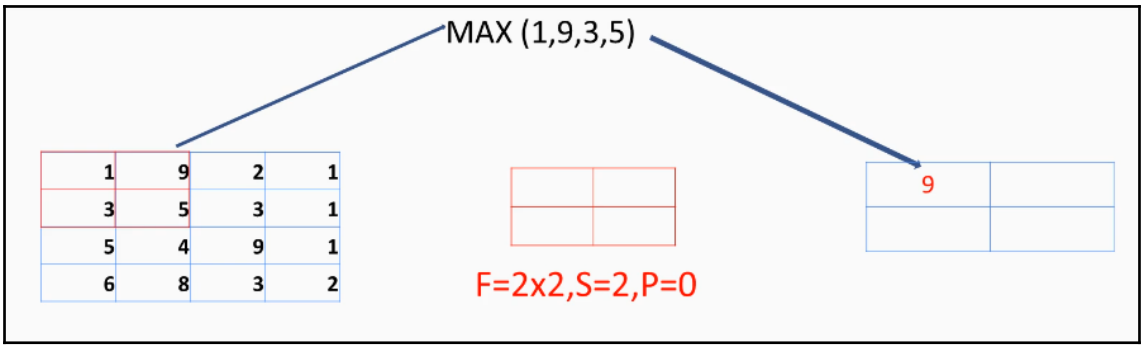

255 255 255 255 255 255
255 255 255 255 255 255
255 255 255 255 255 255
10 10 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
6 x 6 X 3
Conv
3x3 X 3
0 0 0 0
13230 13230 13230 13230
13230 13230 13230 13230
0 0 0 0
4 x 4 X 1
IM=6
F=3
6-3+1=4

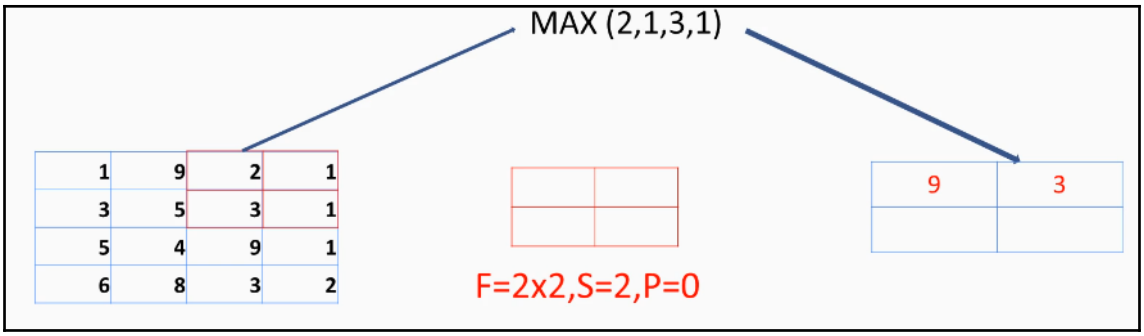

0 0 0 0 0 0 0 0
0 255 255 255 255 255 255 0
0 255 255 255 255 255 255 0
0 255 255 255 255 255 255 0
0 10 10 10 10 10 10 0
0 10 10 10 10 10 10 0
0 10 10 10 10 10 10 0
0 0 0 0 0 0 0 0
8 x 8 X 3
Conv
3x3 X 3
$OM = IM - F + 2 * P + 1$
- - - - - -
1530 2295 2295 2295 2295 1530
0 0 0 0 0 0
1470 2205 2205 2205 2205 1470
1470 2205 2205 2205 2205 1470
0 0 0 0 0 0
60 90 90 90 90 60
6 x 6 X 1

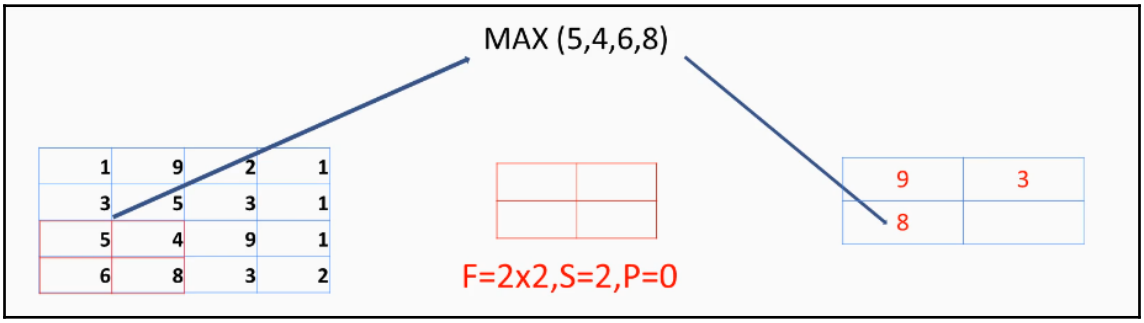

0 0 0 0 0 0 0 0
0 255 255 255 255 255 255 0
0 255 255 255 255 255 255 0
0 255 255 255 255 255 255 0
0 10 10 10 10 10 10 0
0 10 10 10 10 10 10 0
0 10 10 10 10 10 10 0
0 0 0 0 0 0 0 0
8 x 8 X 3
Conv
3x3 X 3
$OM = IM - F + 2 * P + 1$
- - -
1530 2295 2295
1470 2205 2205
60 90 60
3 x 3 X 1

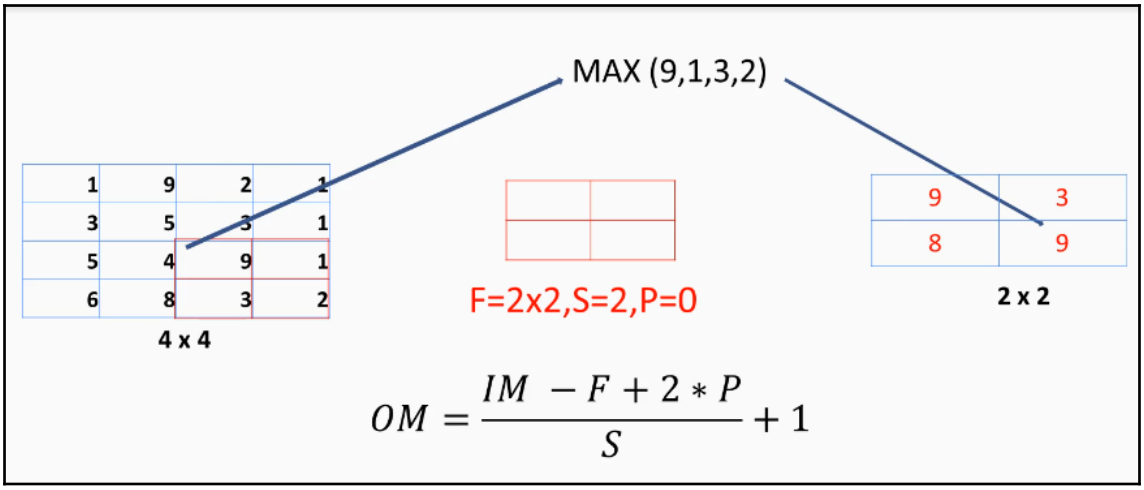

0 0 0 0 0 0 0 0
0 255 255 255 255 255 255 0
0 255 255 255 255 255 255 0
0 255 255 255 255 255 255 0
0 10 10 10 10 10 10 0
0 10 10 10 10 10 10 0
0 10 10 10 10 10 10 0
0 0 0 0 0 0 0 0
8 x 8 X 3
Conv
3x3 X 3
- - -
1530 2295 2295
1470 2205 2205
60 90 60
3 x 3 X 1
$OM = IM - F + 2 * P + 1$


0 0 0 0 0 0 0 0
0 255 255 255 255 255 255 0
0 255 255 255 255 255 255 0
0 255 255 255 255 255 255 0
0 10 10 10 10 10 10 0
0 10 10 10 10 10 10 0
0 10 10 10 10 10 10 0
0 0 0 0 0 0 0 0
8 x 8 X 3
Conv
3x3 X 3
- - -
1530 2295 2295
1470 2205 2205
60 90 60
3 x 3 X 1
$OM = \frac{IM - F + 2 * P}{S} + 1$

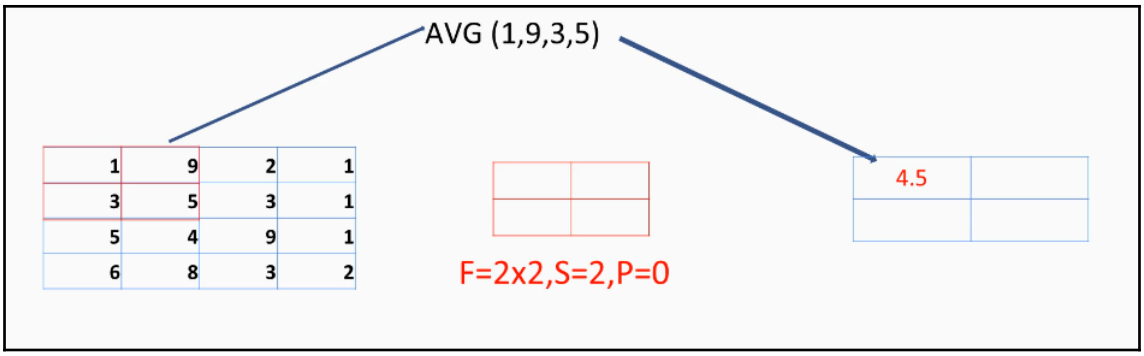

MAX (1,9,3,5)
1 9 2 1
3 5 3 1
5 4 9 1
6 8 3 2
9
F=2x2,S=2,P=0

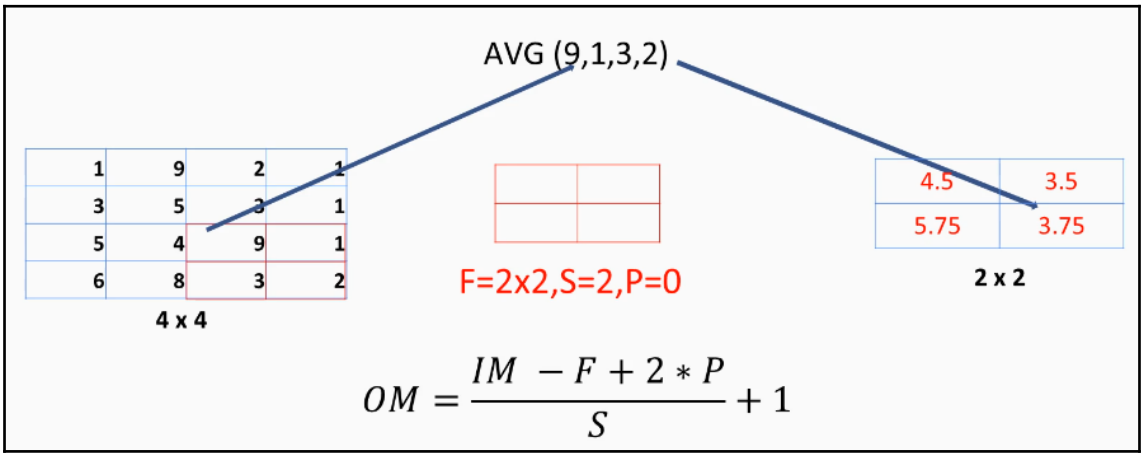




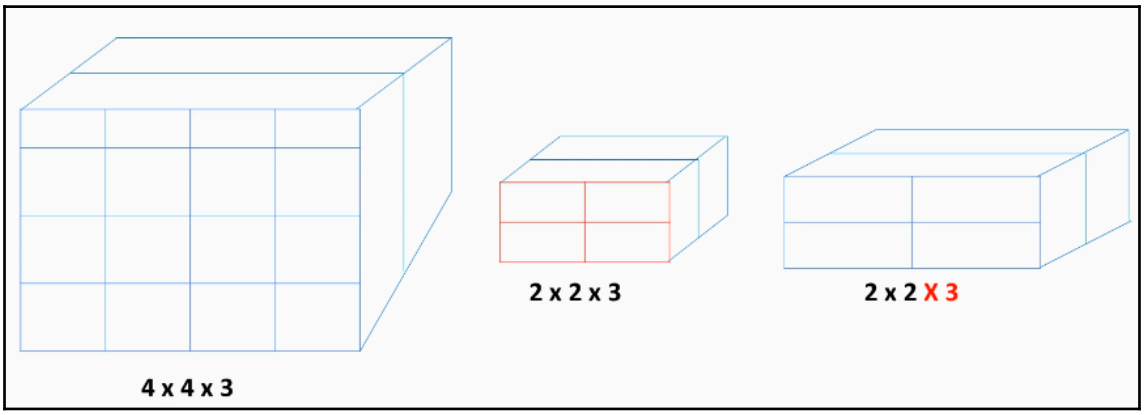


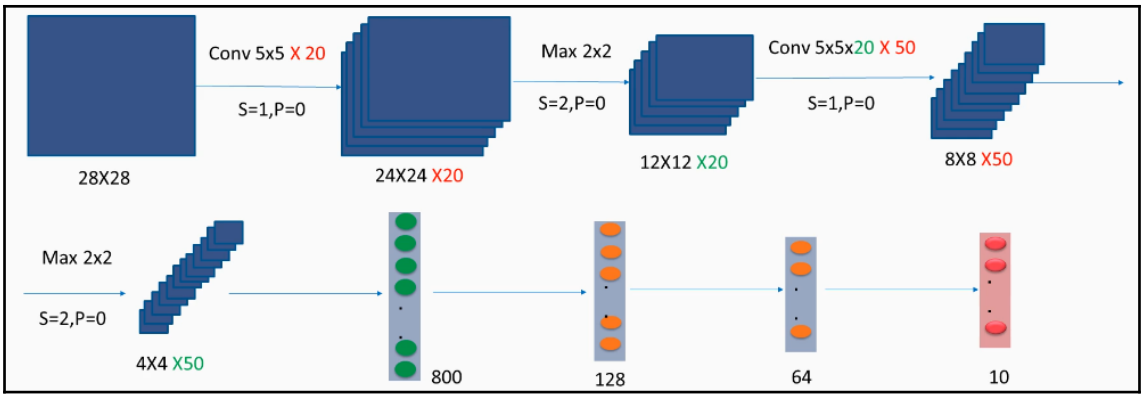
$$OM = \frac{IM - F + 2 * P}{S} + 1$$

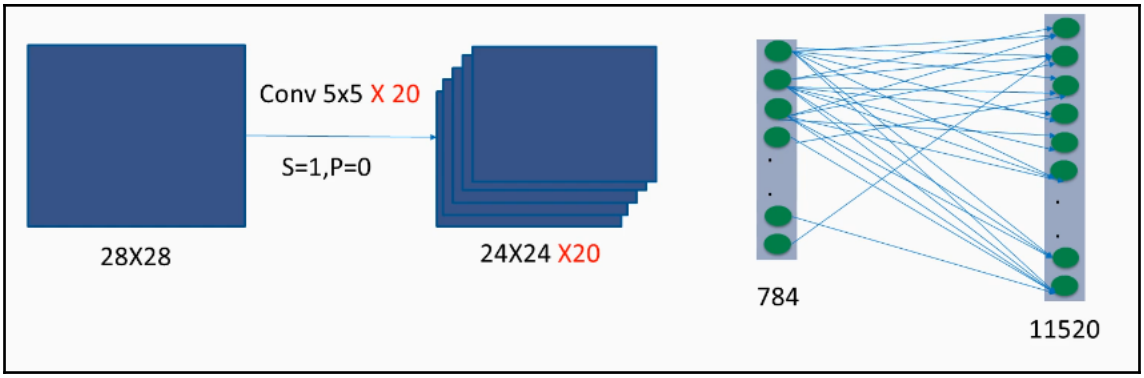


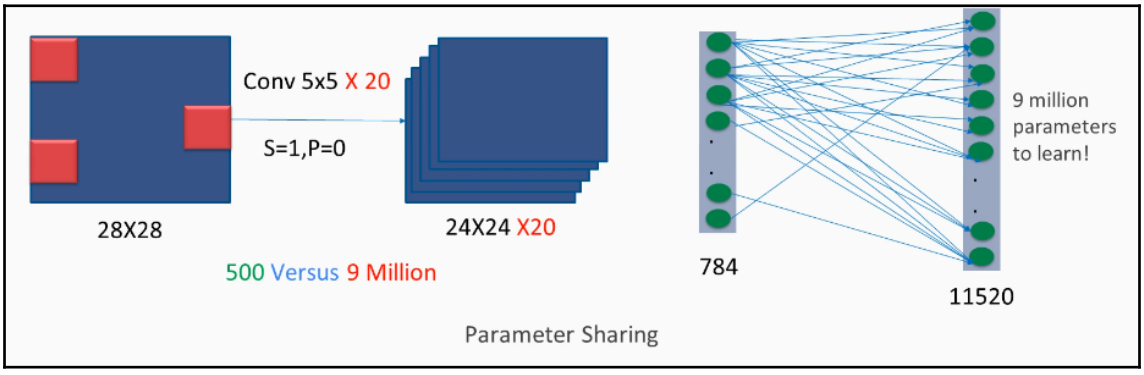


$$OM = \frac{IM - F + 2 * P}{S} + 1$$

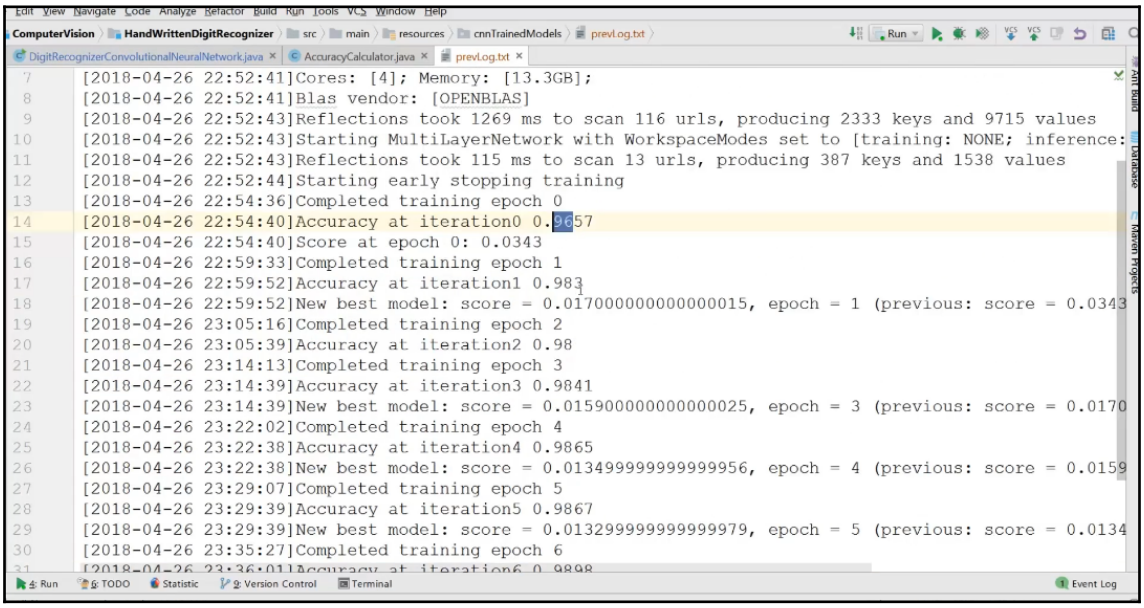


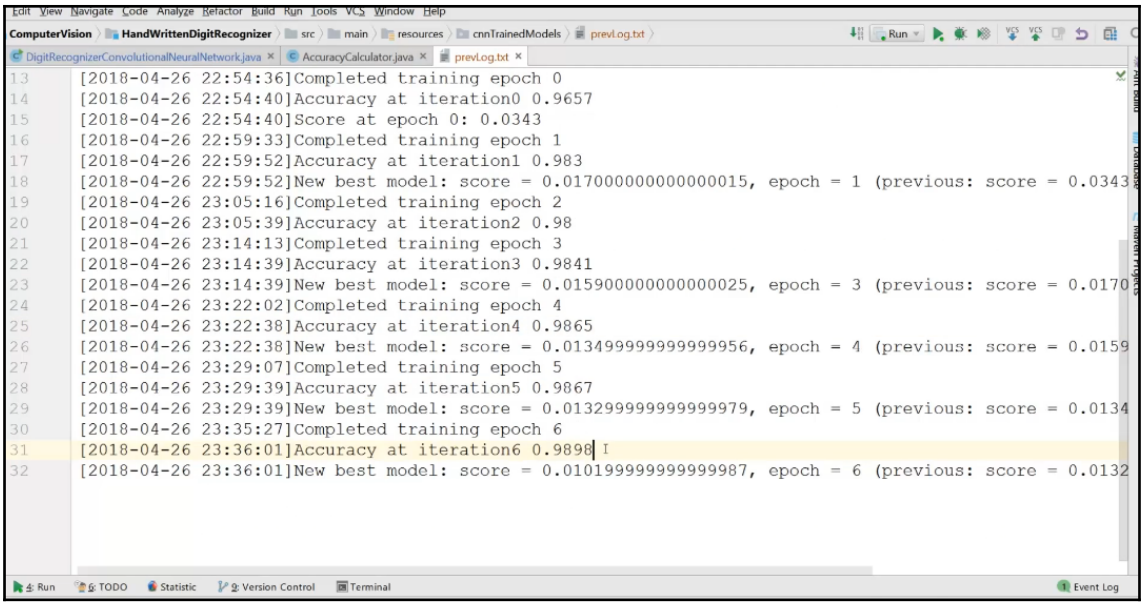

Conv 5x5 X 20
S=1,P=0
Max 2x2
S=2,P=0
Conv 5x5x20 X 50
S=1,P=0
28X28
24X24 X20
12X12 X20
8X8 X50
Max 2x2
S=2,P=0
4X4 X50
800
128
64
10

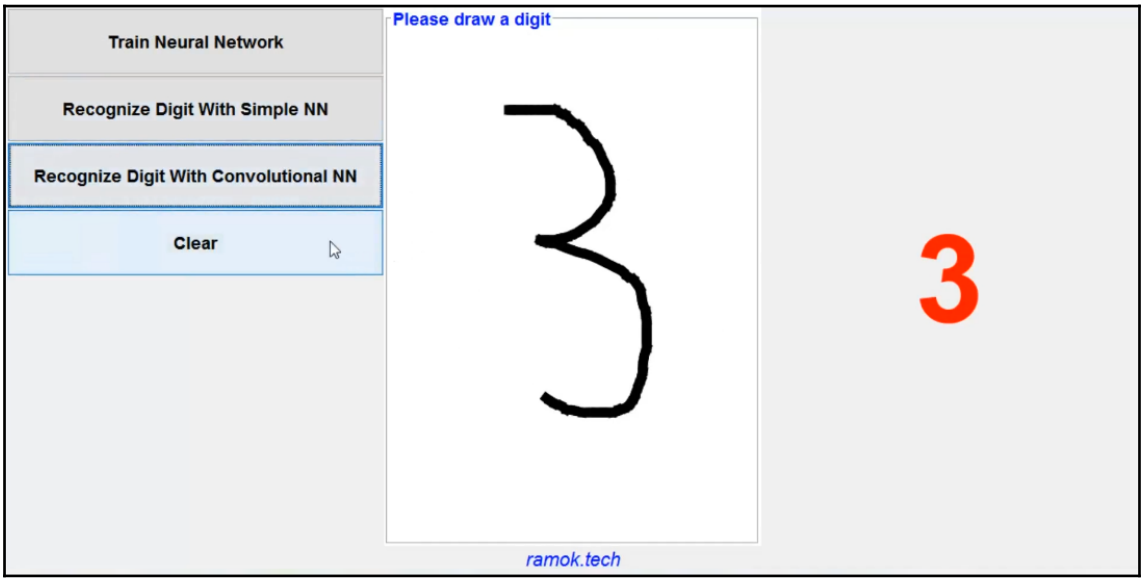

Conv 5x5 X 20
S=1,P=0
28X28
24X24 X20
784
11520

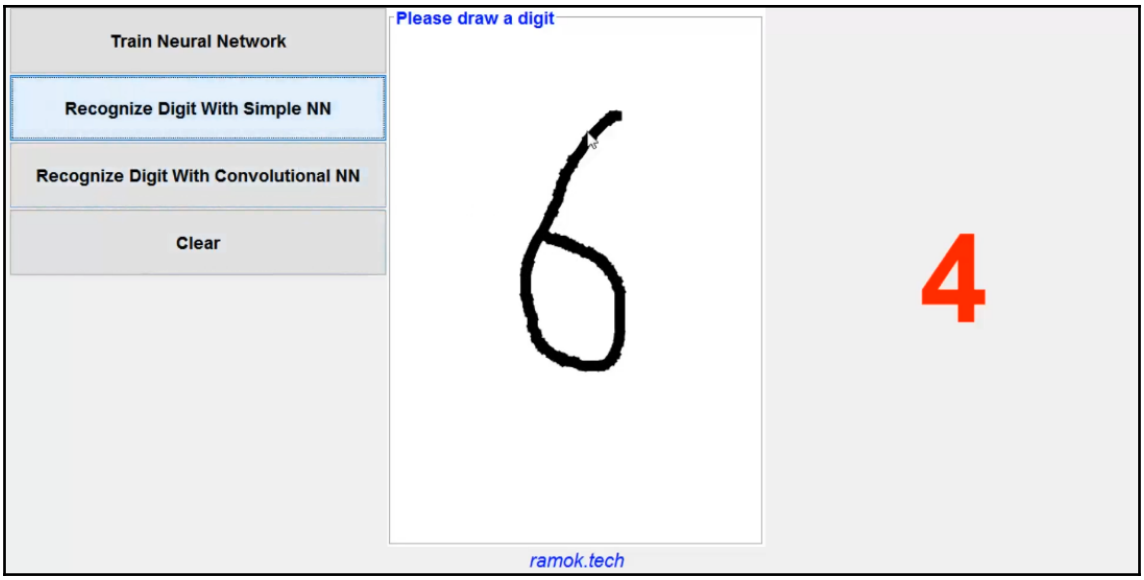

Conv 5x5 X 20
S=1,P=0
28X28
24X24 X20
500 Versus 9 Million
784
11520
9 million parameters to learn!
Parameter Sharing

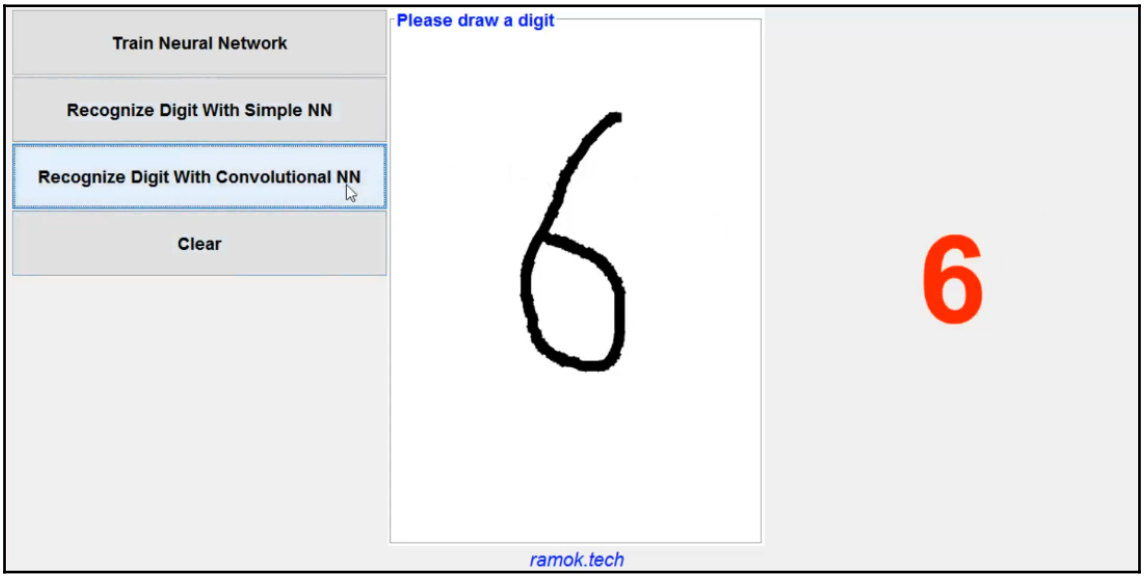
	Activation Matrix (Input)	Activation Size	Parameters Size
Image Matrix	28 X 28 X 1	784	0
Conv 5X5X20,S=1	24 X 24 X 20	11520	500
Max 2 X 2 S=2	12 X 12 X 20	2880	0
Conv 5X5X50,S=1	8 X 8 X 50	3200	1250
Max 2 X 2 S=2	4 X 4 X 50	800	0
FC1	128 X 1	128	102400
FC2	64 X 1	64	8192
OUT	10 X 1	10	640



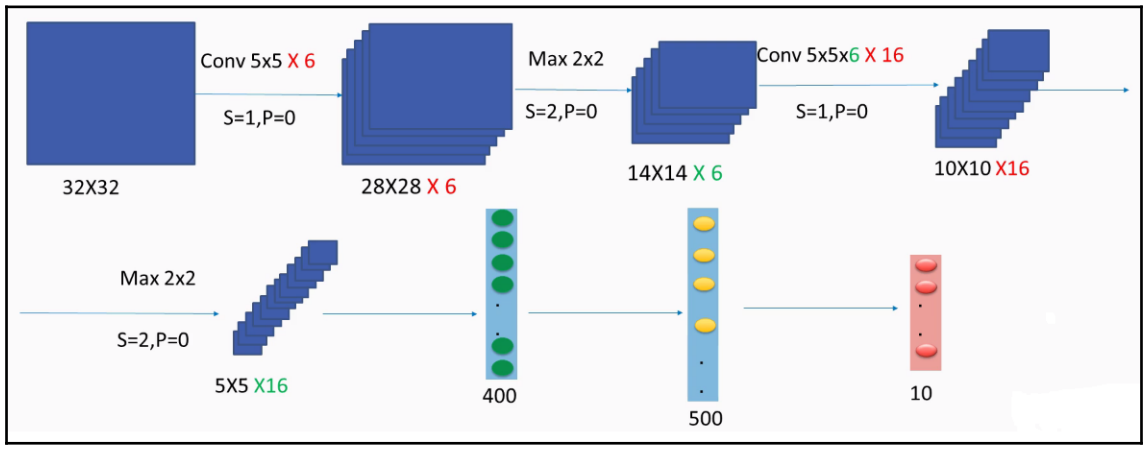

Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ComputerVision HandWrittenDigitRecognizer src main resources cnnTrainedModels prevLog.txt
DigitRecognizerConvolutionalNeuralNetwork.java
AccuracyCalculator.java
prevLog.txt
[2018-04-26 22:54:36]Completed training epoch 0
[2018-04-26 22:54:40]Accuracy at iteration0 0.9657
[2018-04-26 22:54:40]Score at epoch 0: 0.0343
[2018-04-26 22:59:33]Completed training epoch 1
[2018-04-26 22:59:52]Accuracy at iteration1 0.983
[2018-04-26 22:59:52]New best model: score = 0.017000000000000015, epoch = 1 (previous: score = 0.0343
[2018-04-26 23:05:16]Completed training epoch 2
[2018-04-26 23:05:39]Accuracy at iteration2 0.98
[2018-04-26 23:14:13]Completed training epoch 3
[2018-04-26 23:14:39]Accuracy at iteration3 0.9841
[2018-04-26 23:14:39]New best model: score = 0.015900000000000025, epoch = 3 (previous: score = 0.0170
[2018-04-26 23:22:02]Completed training epoch 4
[2018-04-26 23:22:38]Accuracy at iteration4 0.9865
[2018-04-26 23:22:38]New best model: score = 0.013499999999999956, epoch = 4 (previous: score = 0.0159
[2018-04-26 23:29:07]Completed training epoch 5
[2018-04-26 23:29:39]Accuracy at iteration5 0.9867
[2018-04-26 23:29:39]New best model: score = 0.013299999999999979, epoch = 5 (previous: score = 0.0134
[2018-04-26 23:35:27]Completed training epoch 6
[2018-04-26 23:36:01]Accuracy at iteration6 0.9898
[2018-04-26 23:36:01]New best model: score = 0.010199999999999987, epoch = 6 (previous: score = 0.0132
Run
TODO
Statistic
Version Control
Terminal
Event Log

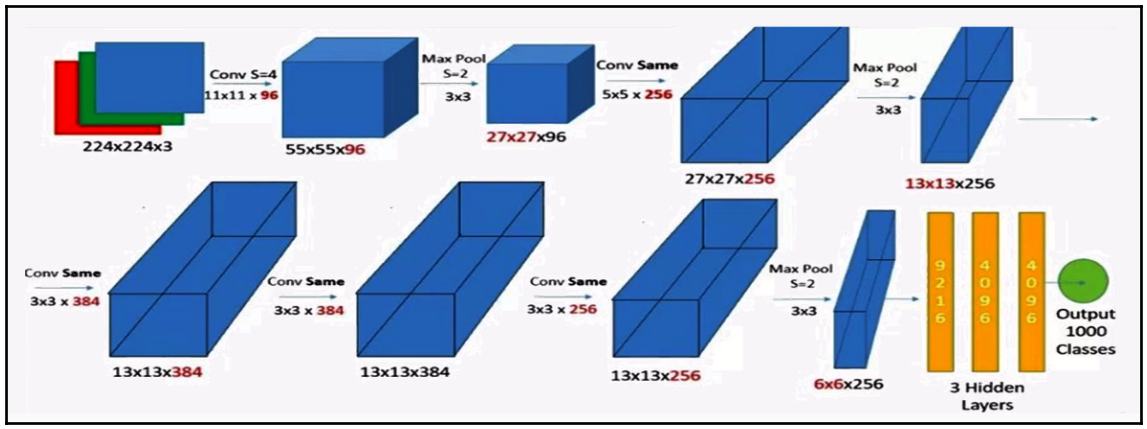

Train Neural Network
Recognize Digit With Simple NN
Recognize Digit With Convolutional NN
Clear
Please draw a digit
3
ramok.tech

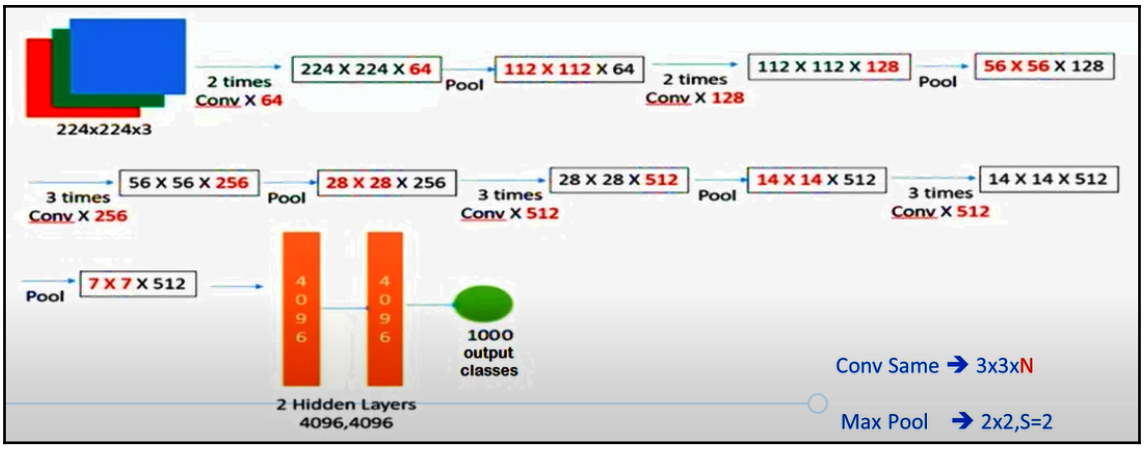

Train Neural Network
Recognize Digit With Simple NN
Recognize Digit With Convolutional NN
Clear
Please draw a digit
4
ramok.tech

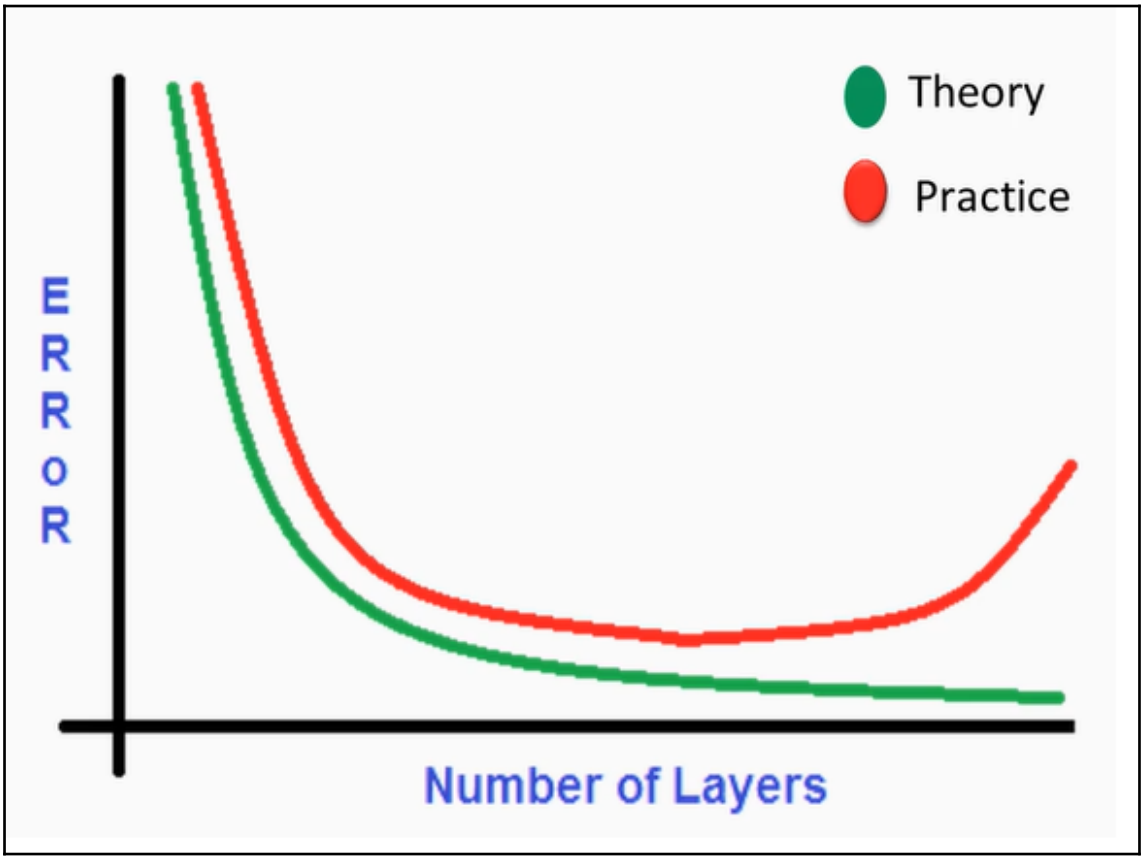

Train Neural Network
Recognize Digit With Simple NN
Recognize Digit With Convolutional NN
Clear
Please draw a digit
6
ramok.tech

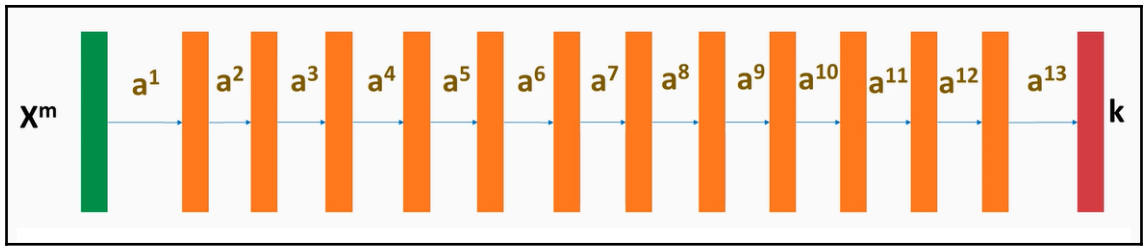
Chapter 3: Transfer Learning and Deep CNN Architectures

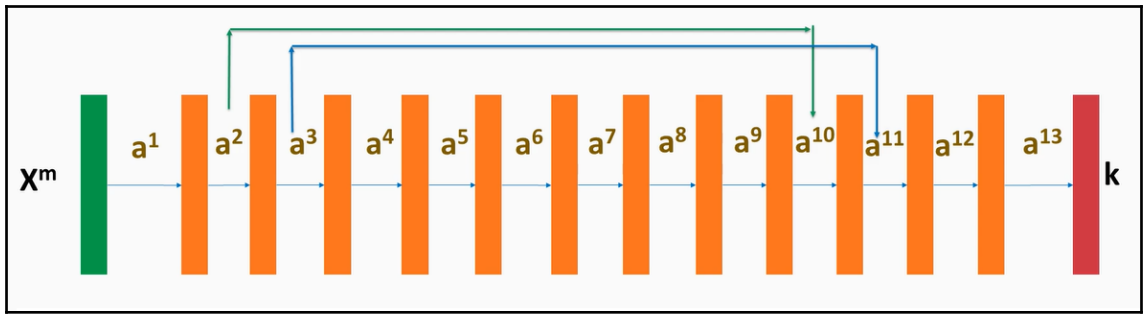


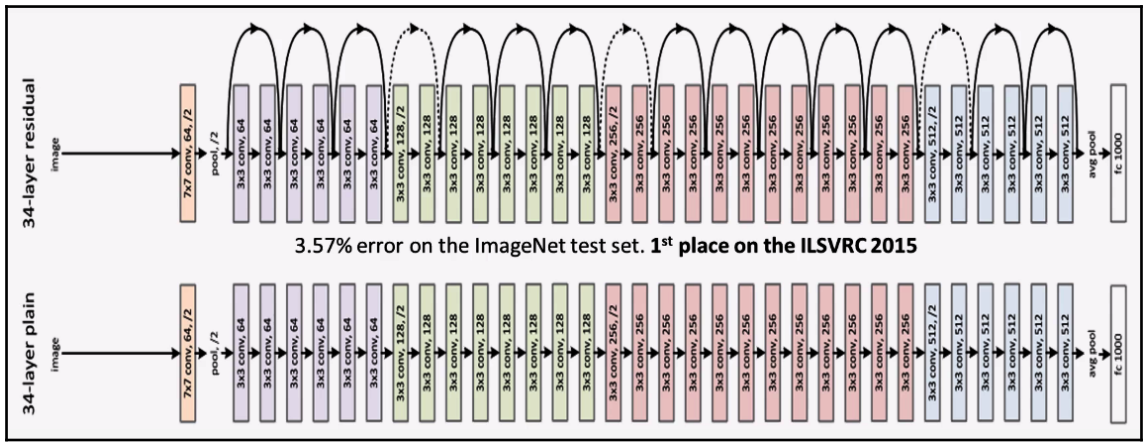



224x224x3
2 times
Conv X 64
224 X 224 X 64
Pool
112 X 112 X 64
2 times
Conv X 128
112 X 112 X 128
Pool
56 X 56 X 128
3 times
Conv X 256
56 X 56 X 256
Pool
28 X 28 X 256
3 times
Conv X 512
28 X 28 X 512
Pool
14 X 14 X 512
3 times
Conv X 512
14 X 14 X 512
Pool
7 X 7 X 512
4096
4096
2 Hidden Layers
4096,4096
1000
output
classes
Conv Same → 3x3xN
Max Pool → 2x2,S=2


Theory
Practice
ERROR
Number of Layers







16	22	18	44
111	34	41	78
1	21	17	76
45	52	78	91

4x4x1

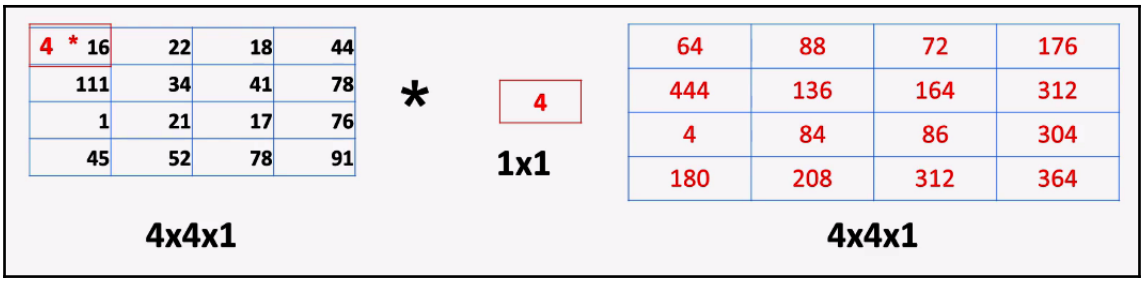
*

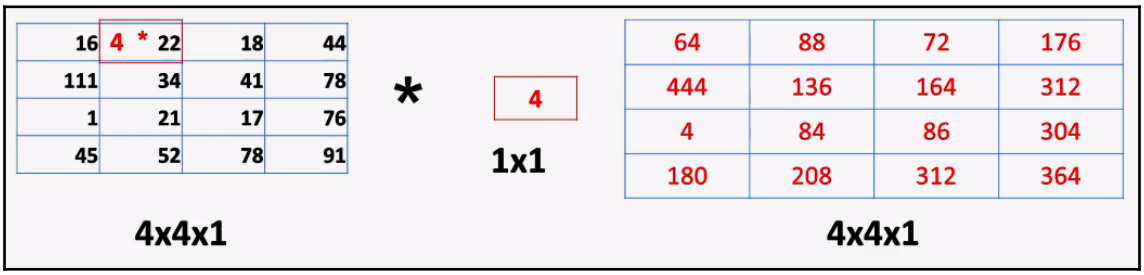
4

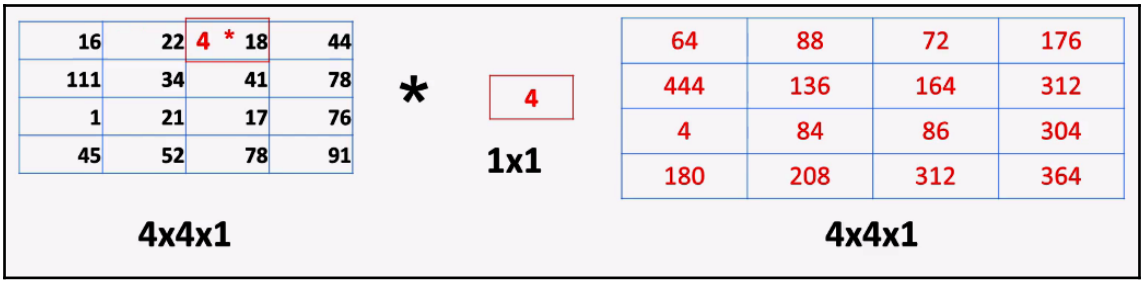
1x1

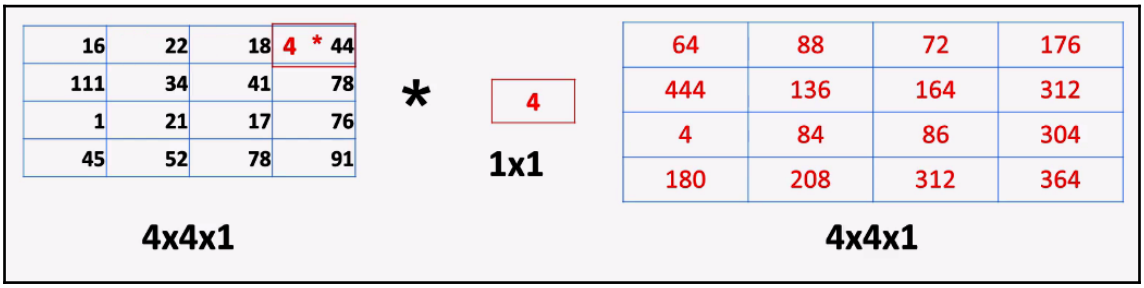
64	88	72	176
444	136	164	312
4	84	86	304
180	208	312	364

4x4x1


4 * 16 22 18 44
111 34 41 78
1 21 17 76
45 52 78 91
4x4x1
*
4
1x1
64 88 72 176
444 136 164 312
4 84 86 304
180 208 312 364
4x4x1


16 4 * 22 18 44
111 34 41 78
1 21 17 76
45 52 78 91
4x4x1
*
4
1x1
64 88 72 176
444 136 164 312
4 84 86 304
180 208 312 364
4x4x1


16 22 4 * 18 44
111 34 41 78
1 21 17 76
45 52 78 91
4x4x1
*
4
1x1
64 88 72 176
444 136 164 312
4 84 86 304
180 208 312 364
4x4x1


16 22 18 4 * 44
111 34 41 78
1 21 17 76
45 52 78 91
4x4x1
*
4
1x1
64 88 72 176
444 136 164 312
4 84 86 304
180 208 312 364
4x4x1

16	22	18	44
4 * 111	34	41	78
1	21	17	76
45	52	78	91

4x4x1

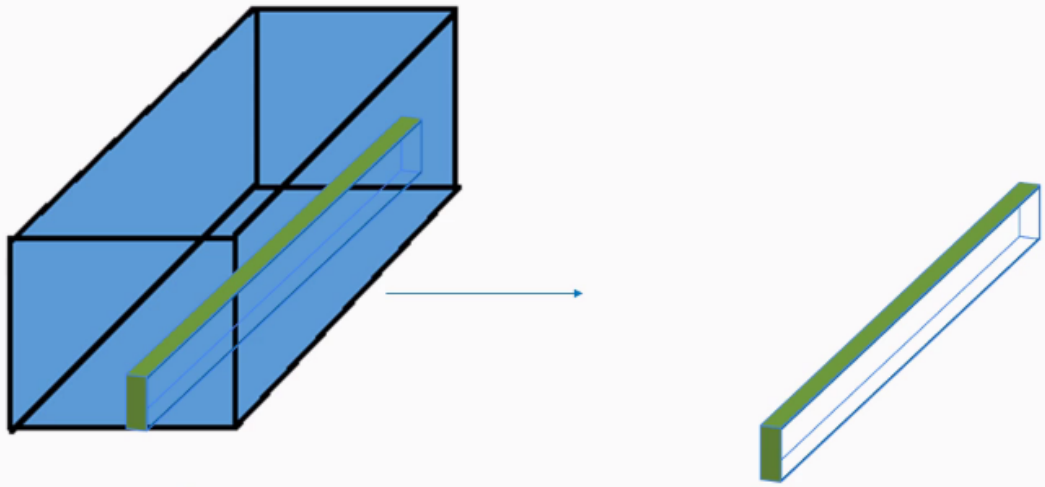
*

4

1x1

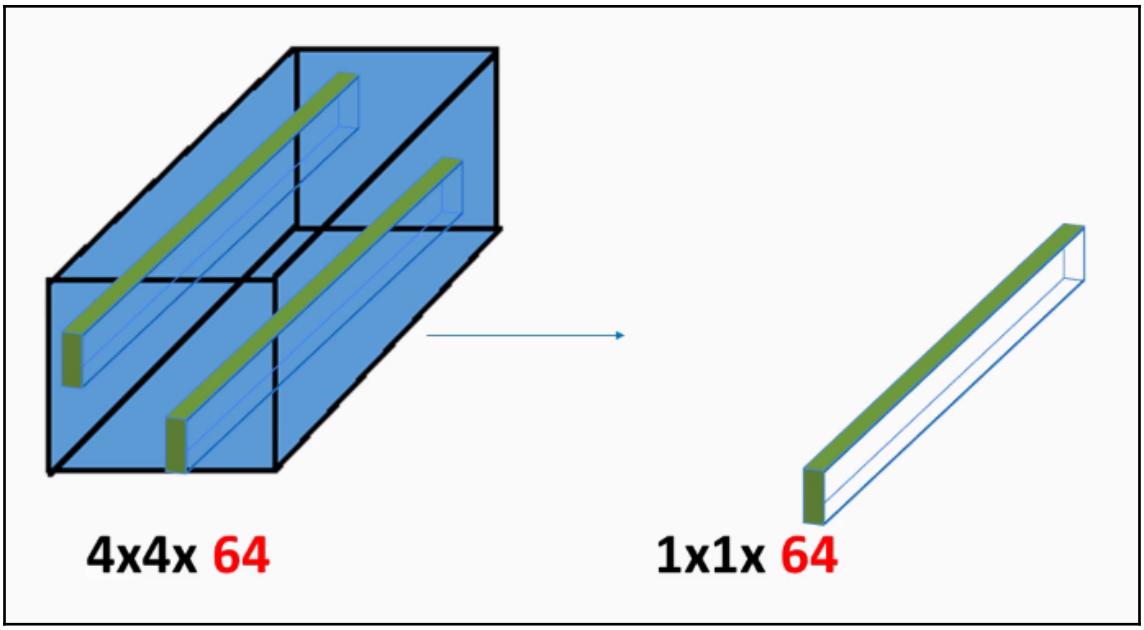
64	88	72	176
444	136	164	312
4	84	86	304
180	208	312	364

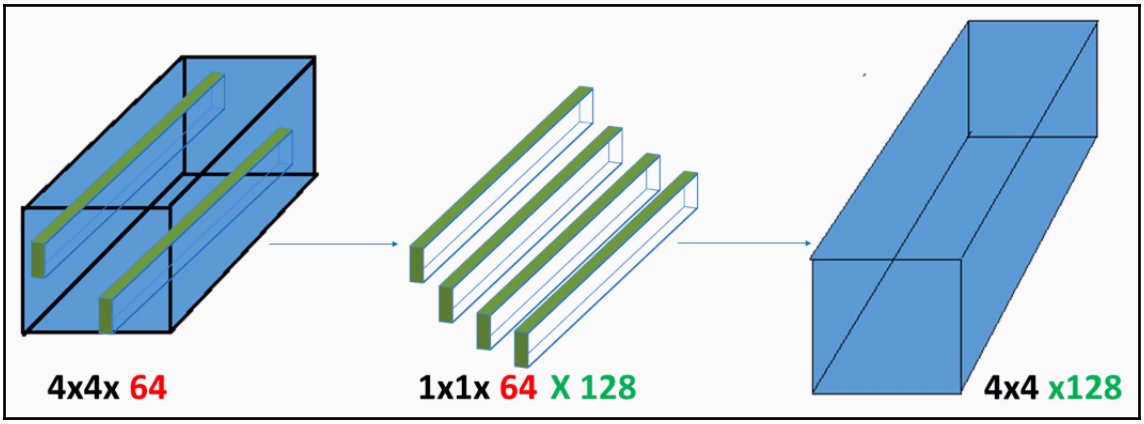
4x4x1

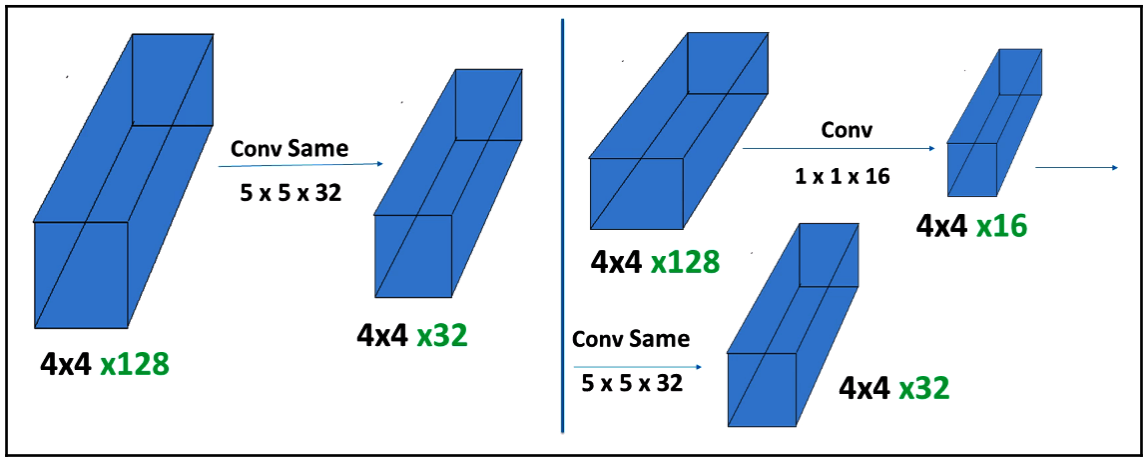


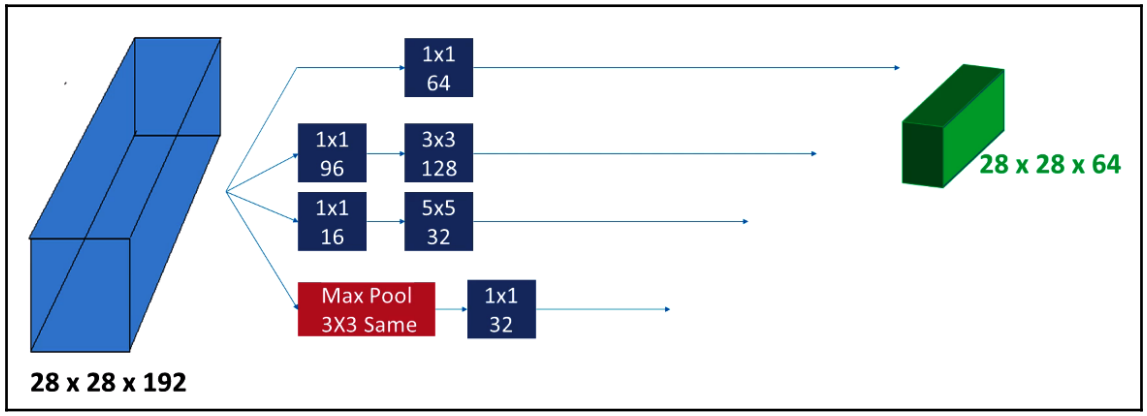
4x4x 64

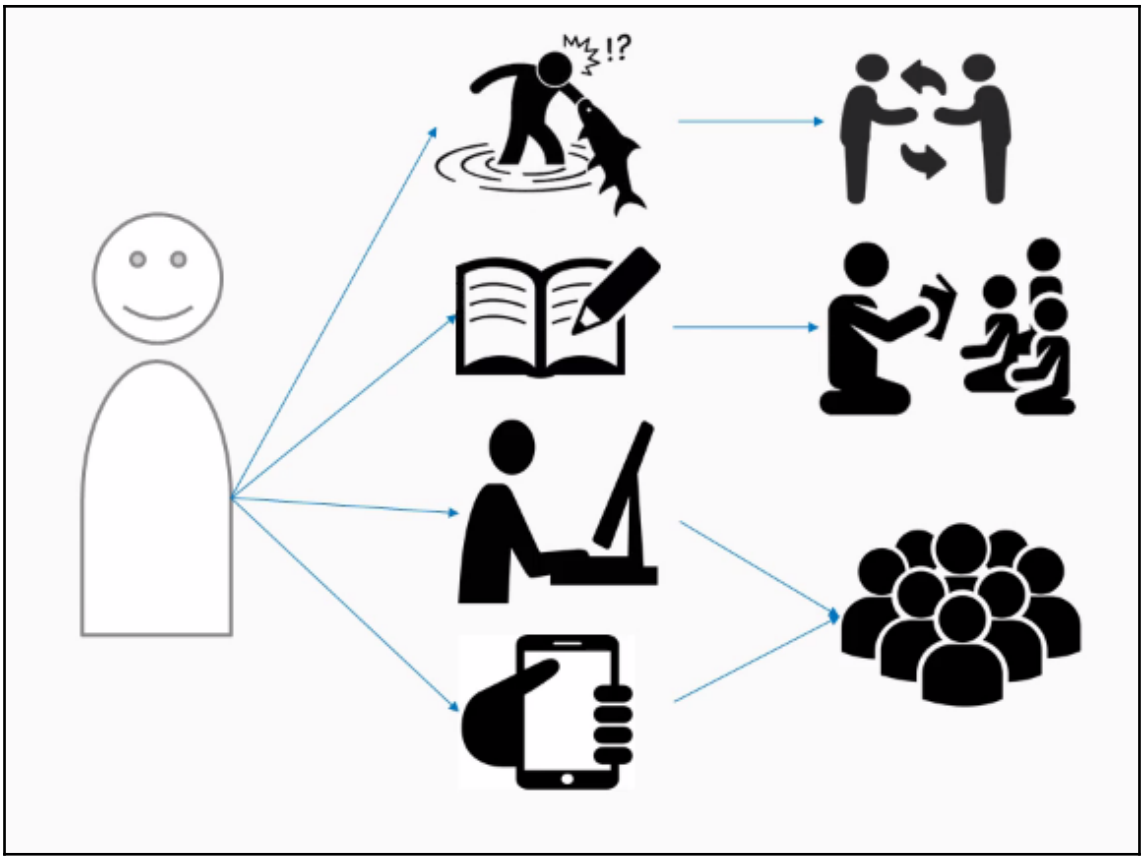
1x1x 64

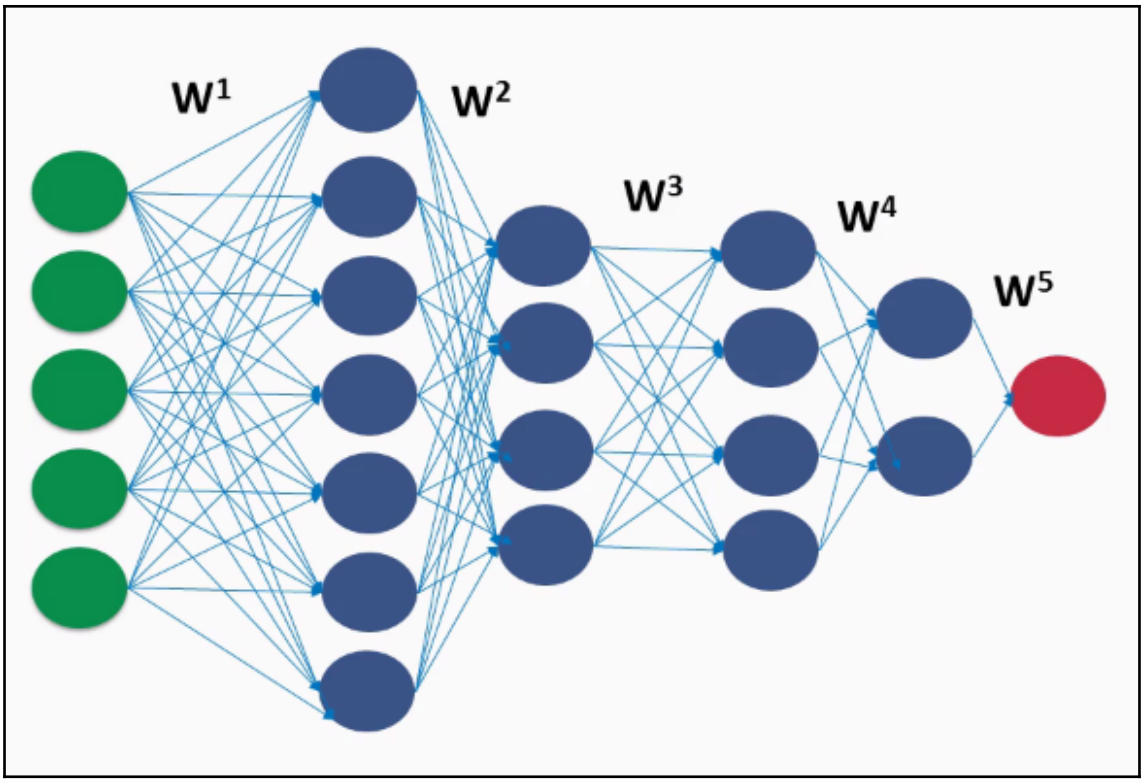

4x4x 64
1x1x 64

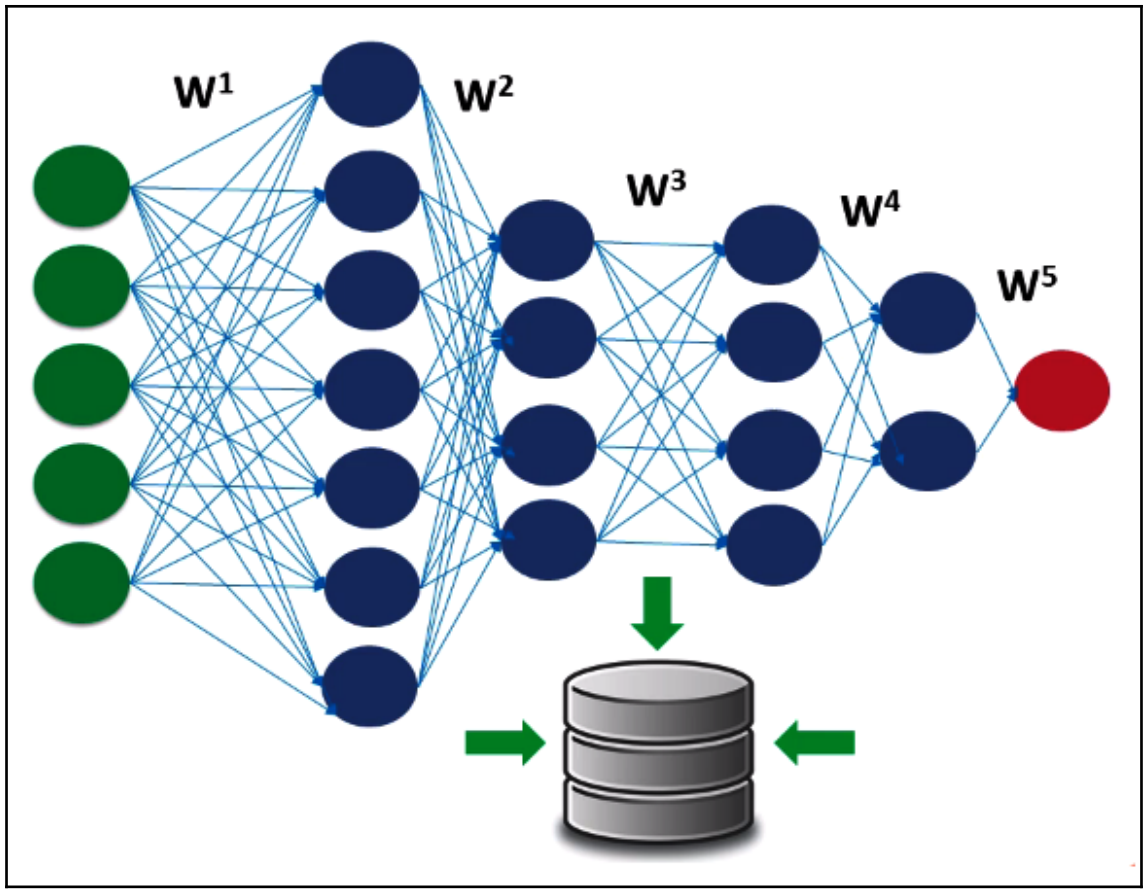

4x4x 64
1x1x 64 X 128
4x4 x128

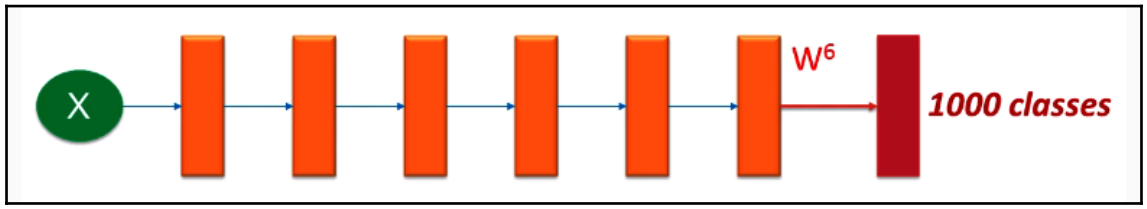

Conv Same
5 x 5 x 32
4x4 x128
4x4 x32
Conv
1 x 1 x 16
4x4 x16
4x4 x128
Conv Same
5 x 5 x 32
4x4 x32

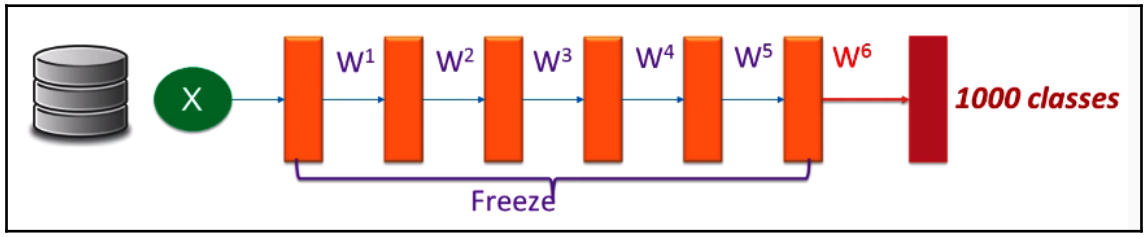

1x1
64
1x1
96
3x3
128
1x1
16
5x5
32
Max Pool
3X3 Same
1x1
32
28 x 28 x 64
28 x 28 x 192

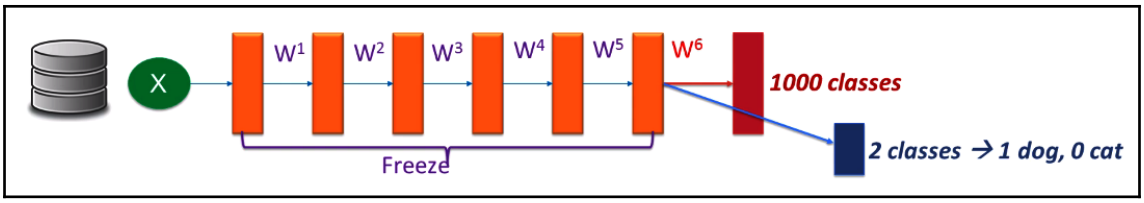


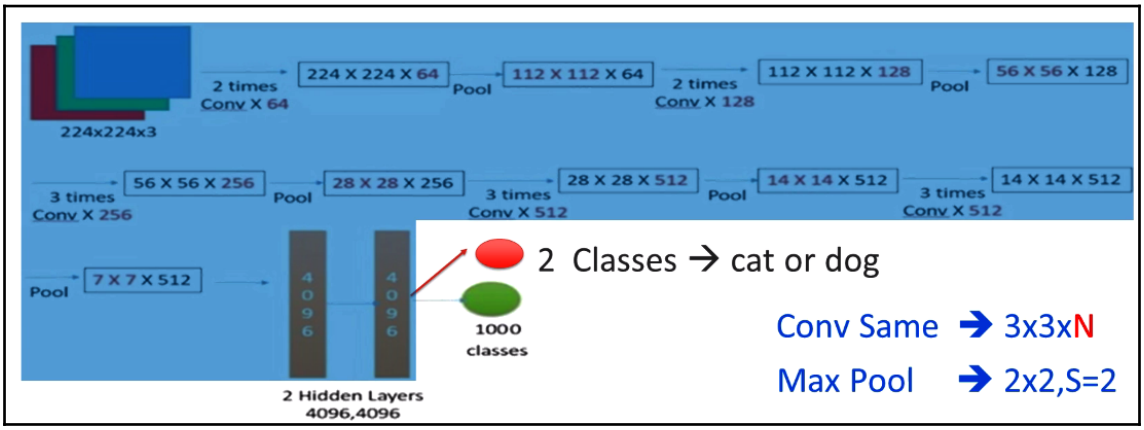

W1
W2
W3
W4
W5


W1
W2
W3
W4
W5


X
W6
1000 classes


X
W1
W2
W3
W4
W5
W6
1000 classes
Freeze



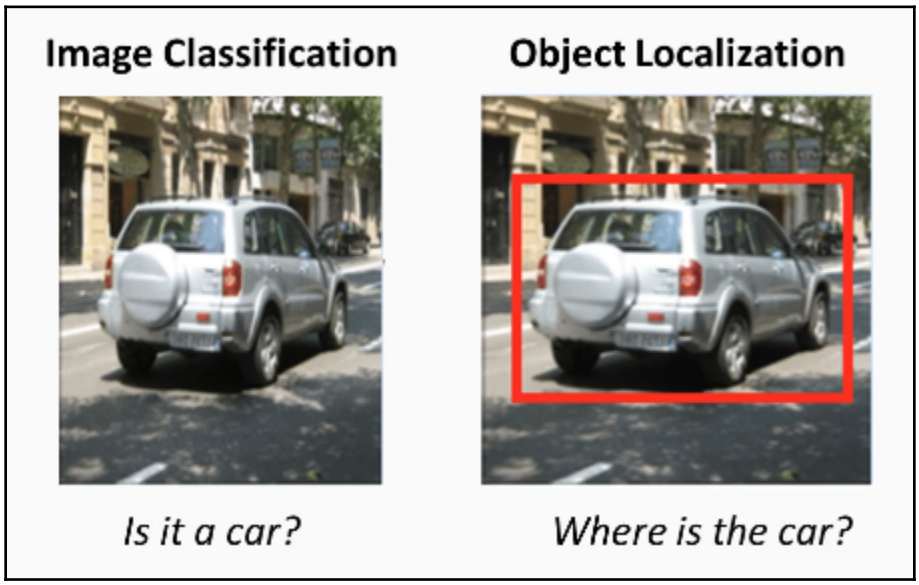


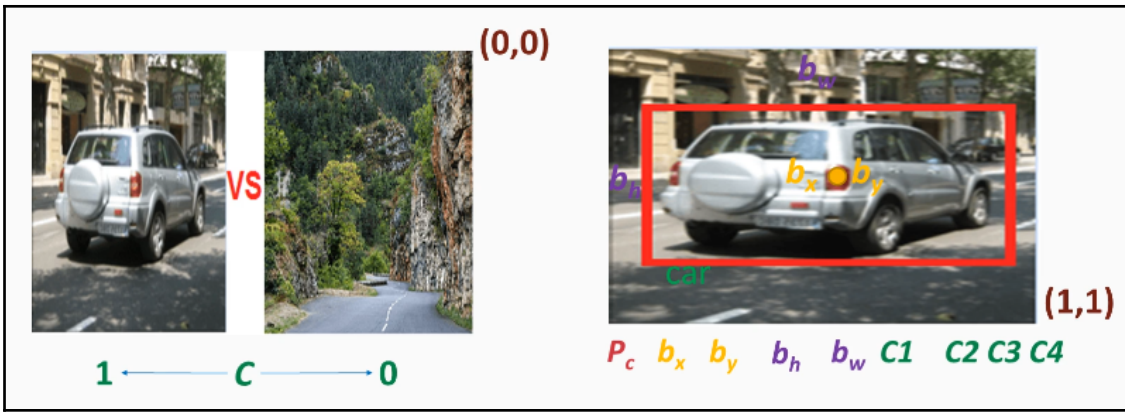
VertexName (VertexType)	nIn,nOut	TotalParams	ParamsShape	Vertex Inputs
input_1 (InputVertex)	-,-	-	-	-
block1_conv1 (ConvolutionLayer)	3,64	1792	b:{1,64}, W:{64,3,3,3}	[input_1]
block1_conv2 (ConvolutionLayer)	64,64	36928	b:{1,64}, W:{64,64,3,3}	[block1_conv1]
block1_pool (SubsamplingLayer)	-,-	0	-	[block1_conv2]
block2_conv1 (ConvolutionLayer)	64,128	73856	b:{1,128}, W:{128,64,3,3}	[block1_pool]
block2_conv2 (ConvolutionLayer)	128,128	147584	b:{1,128}, W:{128,128,3,3}	[block2_conv1]
block2_pool (SubsamplingLayer)	-,-	0	-	[block2_conv2]
block3_conv1 (ConvolutionLayer)	128,256	295168	b:{1,256}, W:{256,128,3,3}	[block2_pool]
block3_conv2 (ConvolutionLayer)	256,256	590080	b:{1,256}, W:{256,256,3,3}	[block3_conv1]
block3_conv3 (ConvolutionLayer)	256,256	590080	b:{1,256}, W:{256,256,3,3}	[block3_conv2]
block3_pool (SubsamplingLayer)	-,-	0	-	[block3_conv3]
block4_conv1 (ConvolutionLayer)	256,512	1180160	b:{1,512}, W:{512,256,3,3}	[block3_pool]
block4_conv2 (ConvolutionLayer)	512,512	2359808	b:{1,512}, W:{512,512,3,3}	[block4_conv1]
block4_conv3 (ConvolutionLayer)	512,512	2359808	b:{1,512}, W:{512,512,3,3}	[block4_conv2]
block4_pool (SubsamplingLayer)	-,-	0	-	[block4_conv3]
block5_conv1 (ConvolutionLayer)	512,512	2359808	b:{1,512}, W:{512,512,3,3}	[block4_pool]
block5_conv2 (ConvolutionLayer)	512,512	2359808	b:{1,512}, W:{512,512,3,3}	[block5_conv1]
block5_conv3 (ConvolutionLayer)	512,512	2359808	b:{1,512}, W:{512,512,3,3}	[block5_conv2]
block5_pool (SubsamplingLayer)	-,-	0	-	[block5_conv3]
flatten (PreprocessorVertex)	-,-	-	-	[block5_pool]
fc1 (DenseLayer)	25088,4096	102764544	b:{1,4096}, W:{25088,4096}	[flatten]
fc2 (DenseLayer)	4096,4096	16781312	b:{1,4096}, W:{4096,4096}	[fc1]
predictions (DenseLayer)	4096,1000	4097000	b:{1,1000}, W:{4096,1000}	[fc2]

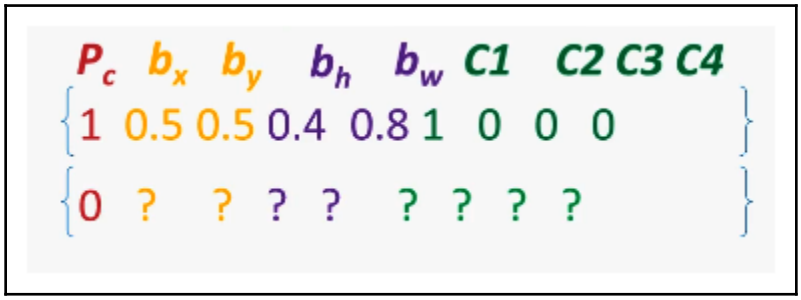
VertexName (VertexType)	nIn,nOut	TotalParams	ParamsShape	Vertex Inputs
input_1 (InputVertex)	-,-	-	-	-
block1_conv1 (Frozen ConvolutionLayer)	3,64	1792	b:{1,64}, W:{64,3,3,3}	[input_1]
block1_conv2 (Frozen ConvolutionLayer)	64,64	36928	b:{1,64}, W:{64,64,3,3}	[block1_conv1]
block1_pool (Frozen SubsamplingLayer)	-,-	0	-	[block1_conv2]
block2_conv1 (Frozen ConvolutionLayer)	64,128	73856	b:{1,128}, W:{128,64,3,3}	[block1_pool]
block2_conv2 (Frozen ConvolutionLayer)	128,128	147584	b:{1,128}, W:{128,128,3,3}	[block2_conv1]
block2_pool (Frozen SubsamplingLayer)	-,-	0	-	[block2_conv2]
block3_conv1 (Frozen ConvolutionLayer)	128,256	295168	b:{1,256}, W:{256,128,3,3}	[block2_pool]
block3_conv2 (Frozen ConvolutionLayer)	256,256	590080	b:{1,256}, W:{256,256,3,3}	[block3_conv1]
block3_conv3 (Frozen ConvolutionLayer)	256,256	590080	b:{1,256}, W:{256,256,3,3}	[block3_conv2]
block3_pool (Frozen SubsamplingLayer)	-,-	0	-	[block3_conv3]

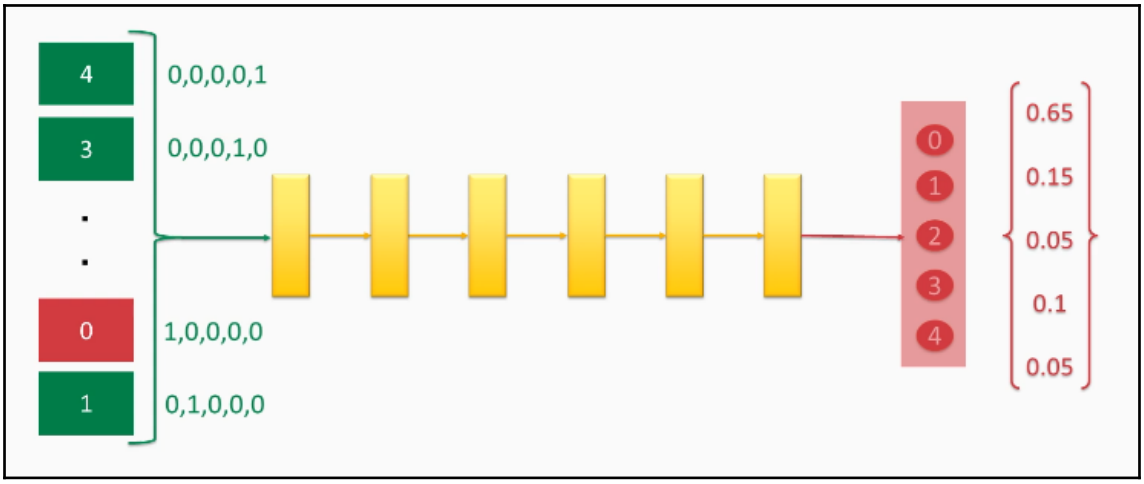
```
Total Parameters:  138357544
Trainable Parameters:  138357544
Frozen Parameters:  0
```

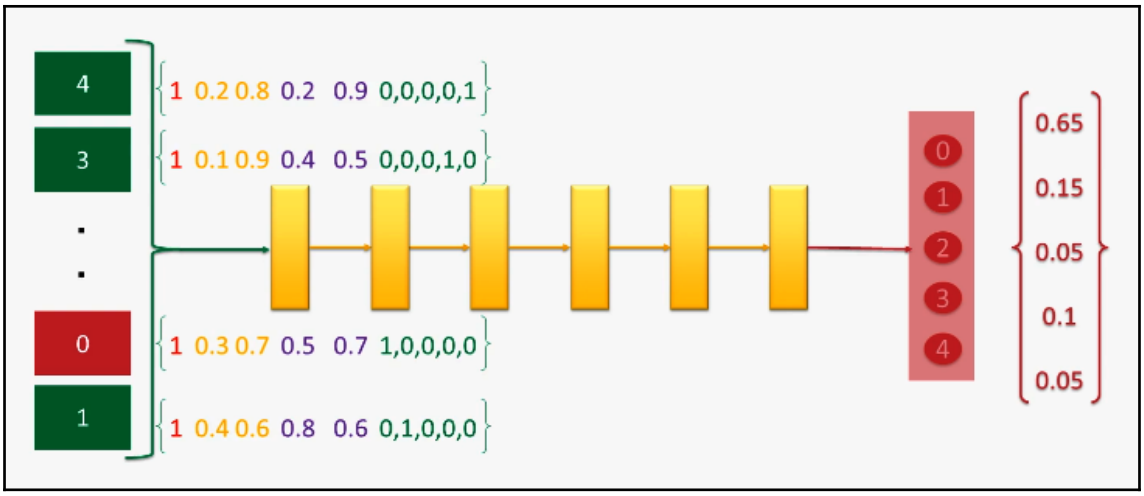
Chapter 4: Real-Time Object Detection

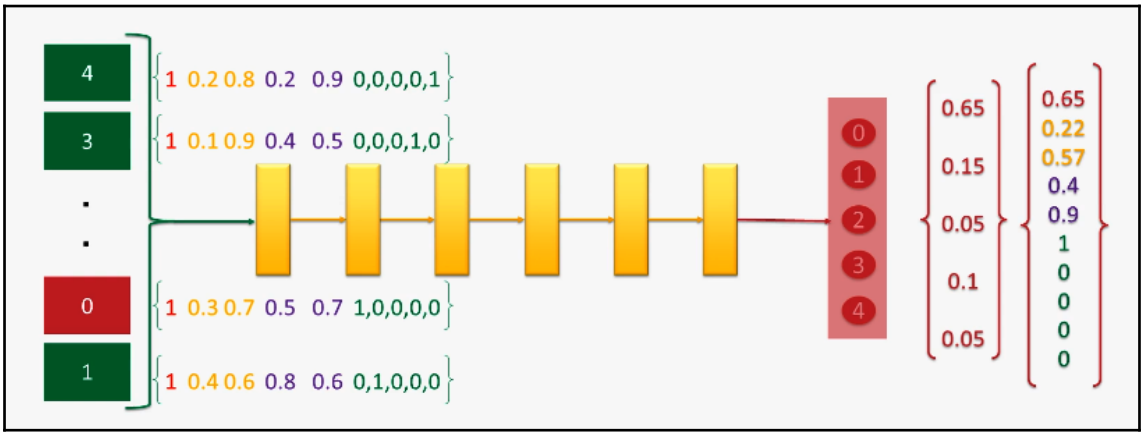


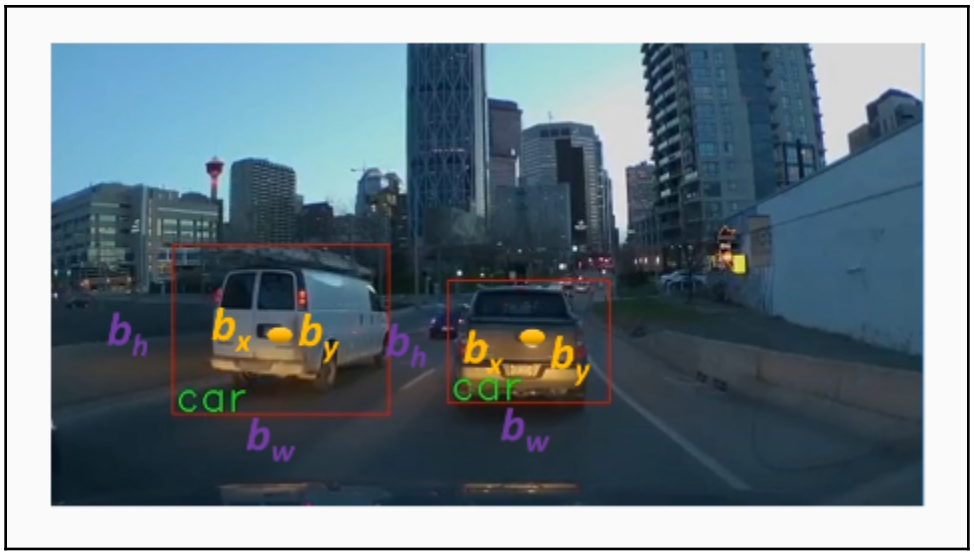

(0,0)
VS
1 ← C → 0
b_w
b_h
b_x
b_y
car
(1,1)
P_c b_x b_y b_h b_w C1 C2 C3 C4

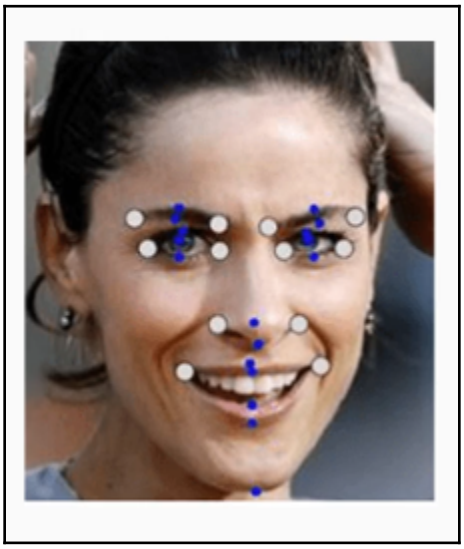

P_c b_x b_y b_h b_w C1 C2 C3 C4
1 0.5 0.5 0.4 0.8 1 0 0 0
0 ? ? ? ? ? ? ? ?

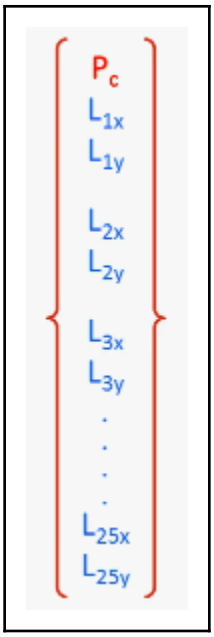

4
3
0
1
0,0,0,0,1
0,0,0,1,0
1,0,0,0,0
0,1,0,0,0
0
1
2
3
4
0.65
0.15
0.05
0.1
0.05

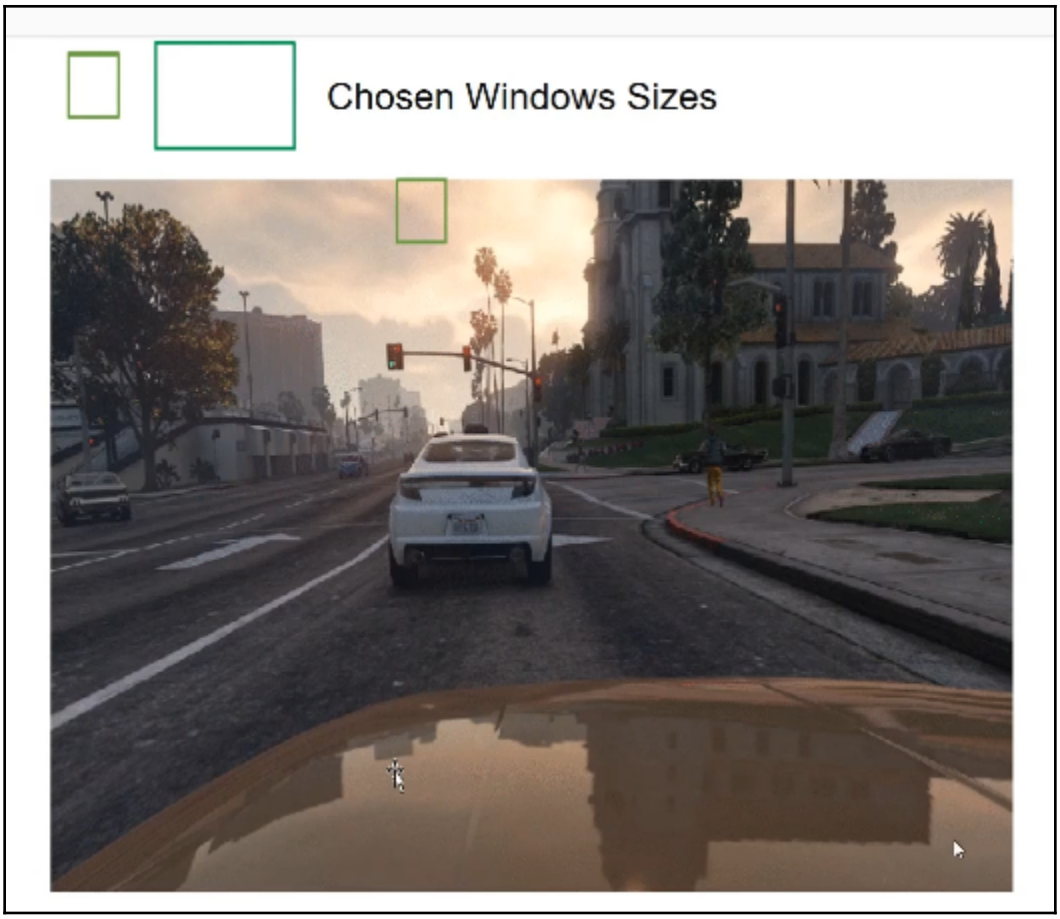

4
3
0
1
1 0.2 0.8 0.2 0.9 0,0,0,0,1
1 0.1 0.9 0.4 0.5 0,0,0,1,0
1 0.3 0.7 0.5 0.7 1,0,0,0,0
1 0.4 0.6 0.8 0.6 0,1,0,0,0
0
1
2
3
4
0.65
0.15
0.05
0.1
0.05

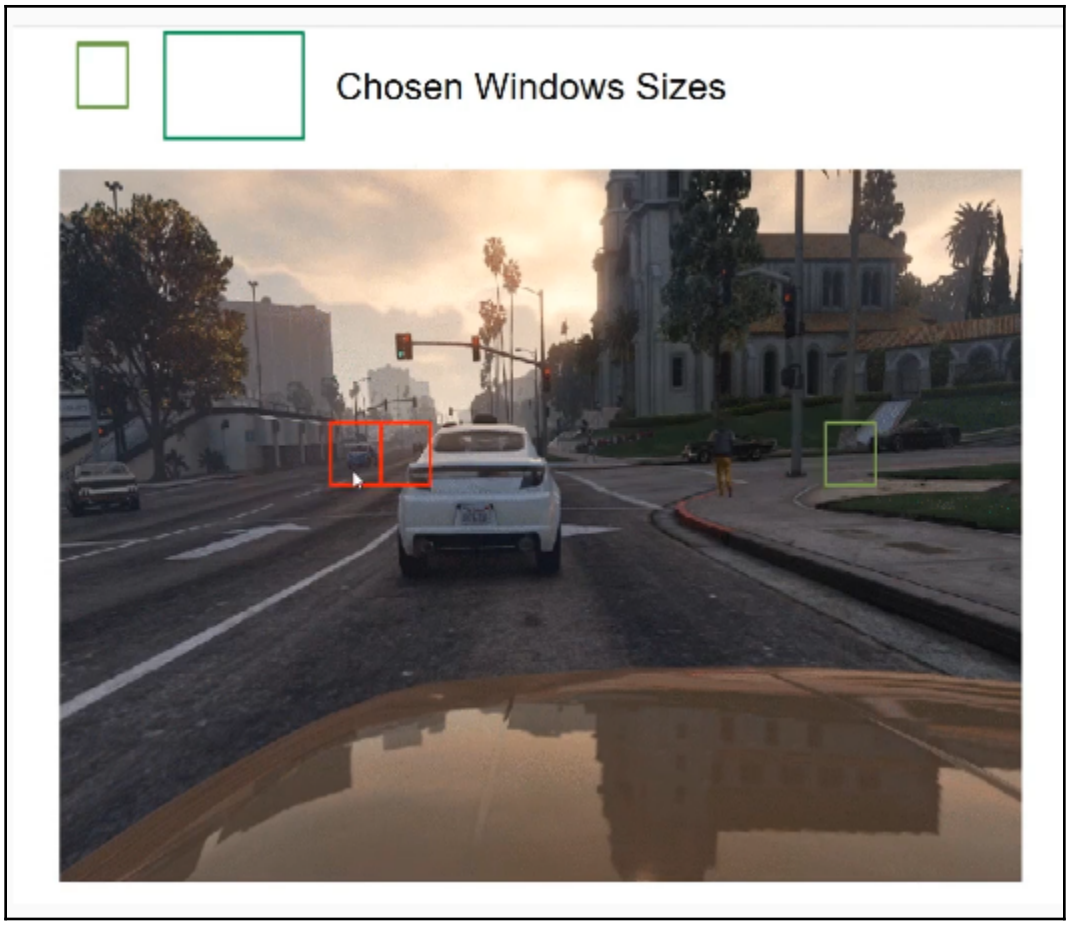

4
3
0
1
1 0.2 0.8 0.2 0.9 0,0,0,0,1
1 0.1 0.9 0.4 0.5 0,0,0,1,0
1 0.3 0.7 0.5 0.7 1,0,0,0,0
1 0.4 0.6 0.8 0.6 0,1,0,0,0
0
1
2
3
4
0.65
0.15
0.05
0.1
0.05
0.65
0.22
0.57
0.4
0.9
1
0
0
0
0

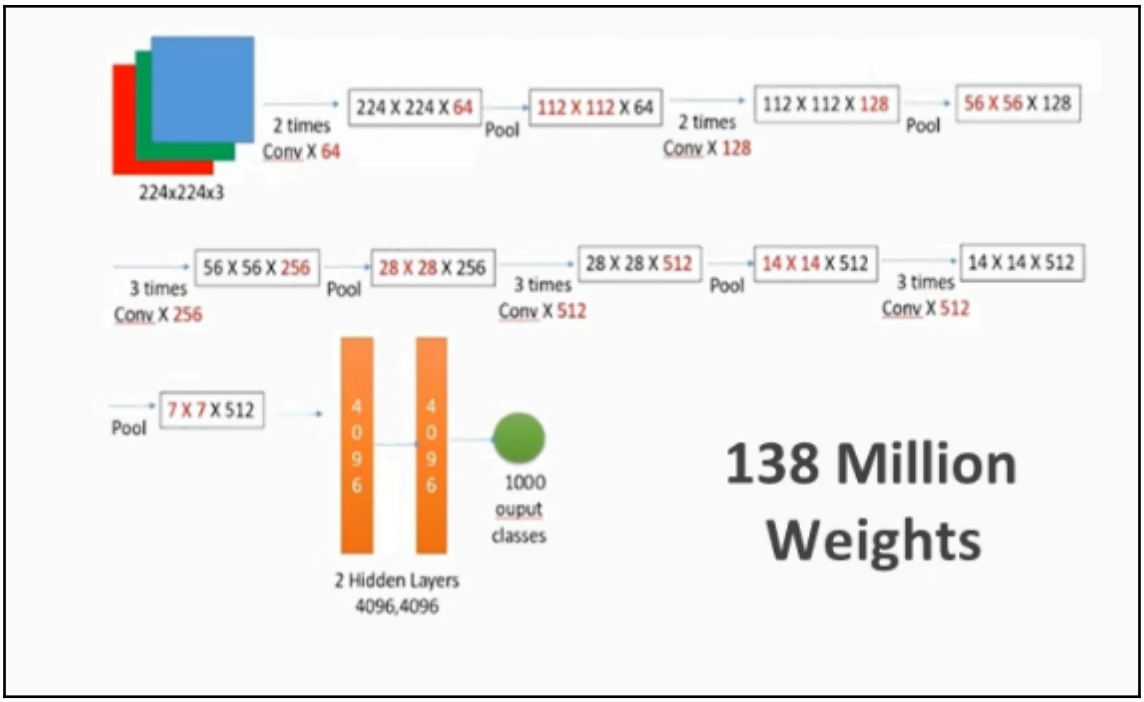

b_h
b_x
b_y
car
b_w
b_h
b_x
b_y
car
b_w

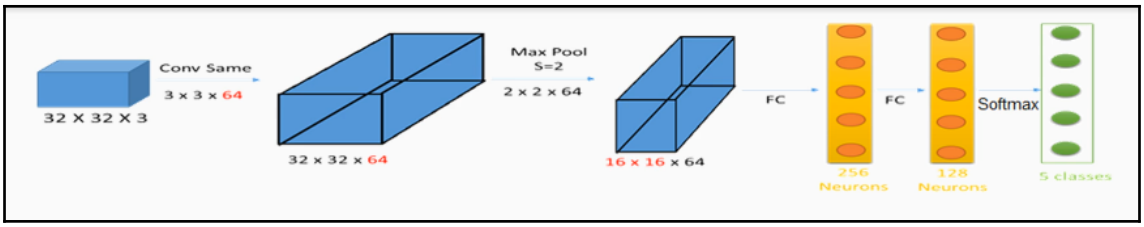


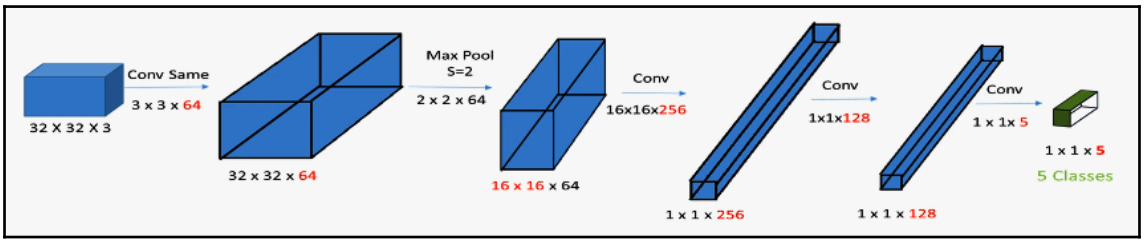

P_c
L_{1x}
L_{1y}
L_{2x}
L_{2y}
L_{3x}
L_{3y}
L_{25x}
L_{25y}

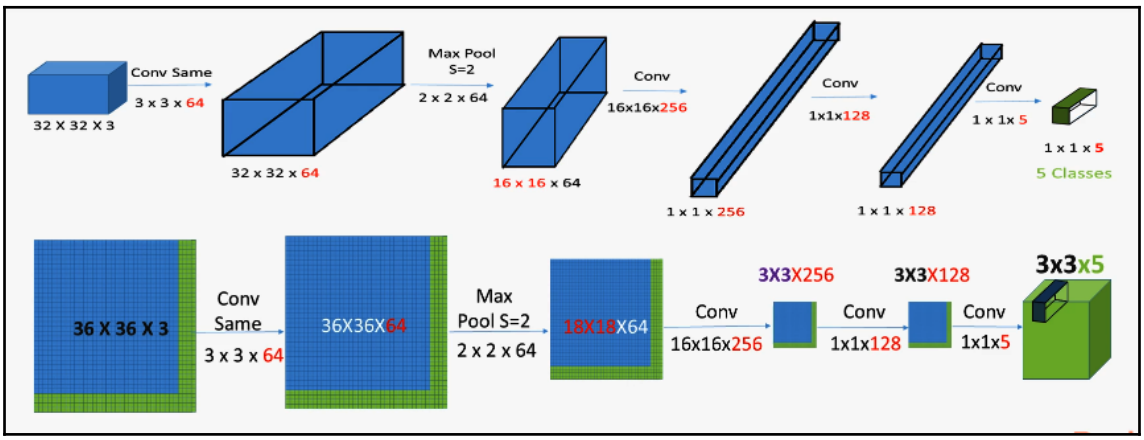

Chosen Windows Sizes

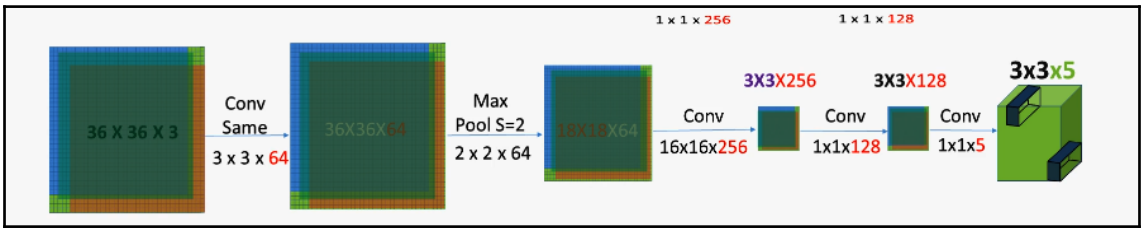

Chosen Windows Sizes

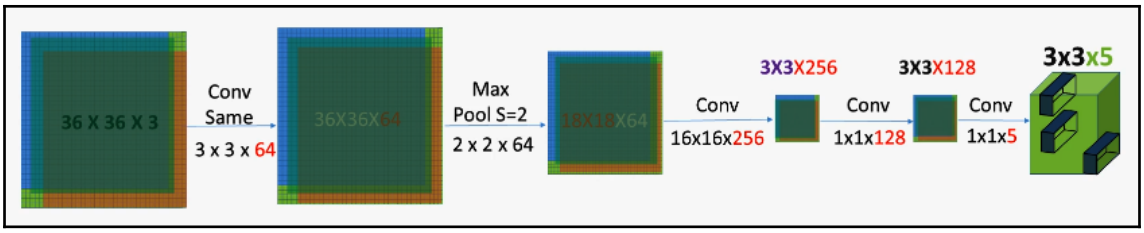

224x224x3
2 times
Conv X 64
224 X 224 X 64
Pool
112 X 112 X 64
2 times
Conv X 128
112 X 112 X 128
Pool
56 X 56 X 128
3 times
Conv X 256
56 X 56 X 256
Pool
28 X 28 X 256
3 times
Conv X 512
28 X 28 X 512
Pool
14 X 14 X 512
3 times
Conv X 512
14 X 14 X 512
Pool
7 X 7 X 512
4096
4096
1000
ouput
classes
2 Hidden Layers
4096,4096
138 Million
Weights


32 X 32 X 3
Conv Same
3 x 3 x 64
32 x 32 x 64
Max Pool
S=2
2 x 2 x 64
16 x 16 x 64
FC
256
Neurons
FC
128
Neurons
Softmax
5 classes

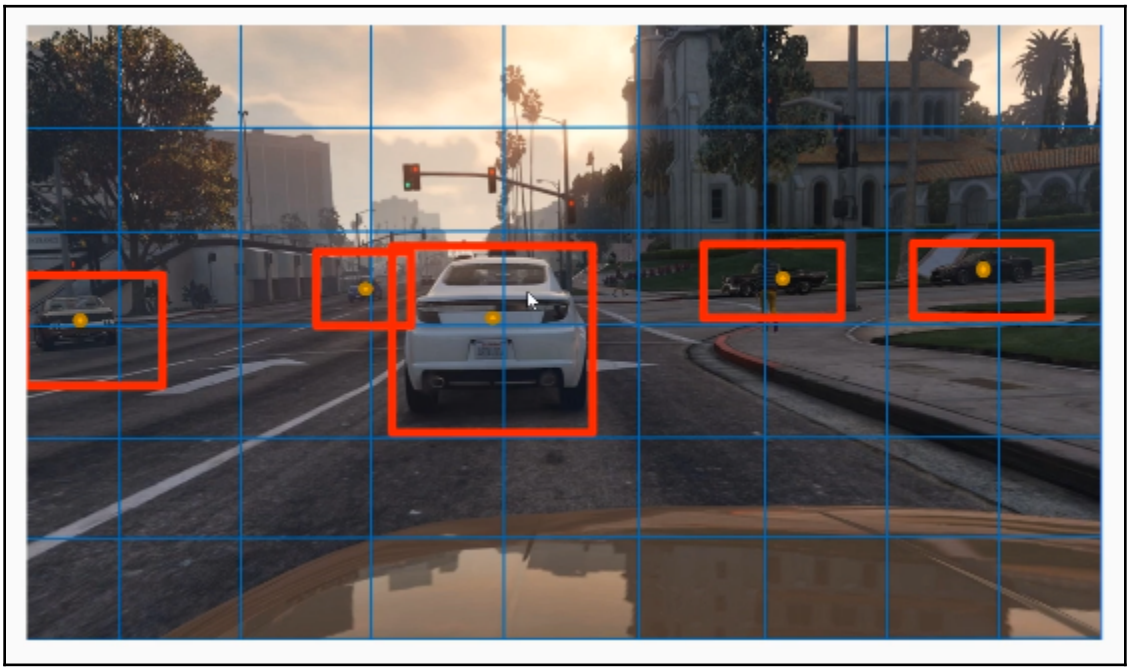

32 X 32 X 3
Conv Same
3 x 3 x 64
32 x 32 x 64
Max Pool
S=2
2 x 2 x 64
16 x 16 x 64
Conv
16x16x256
1 x 1 x 256
Conv
1x1x128
1 x 1 x 128
Conv
1 x 1x 5
1 x 1 x 5
5 Classes

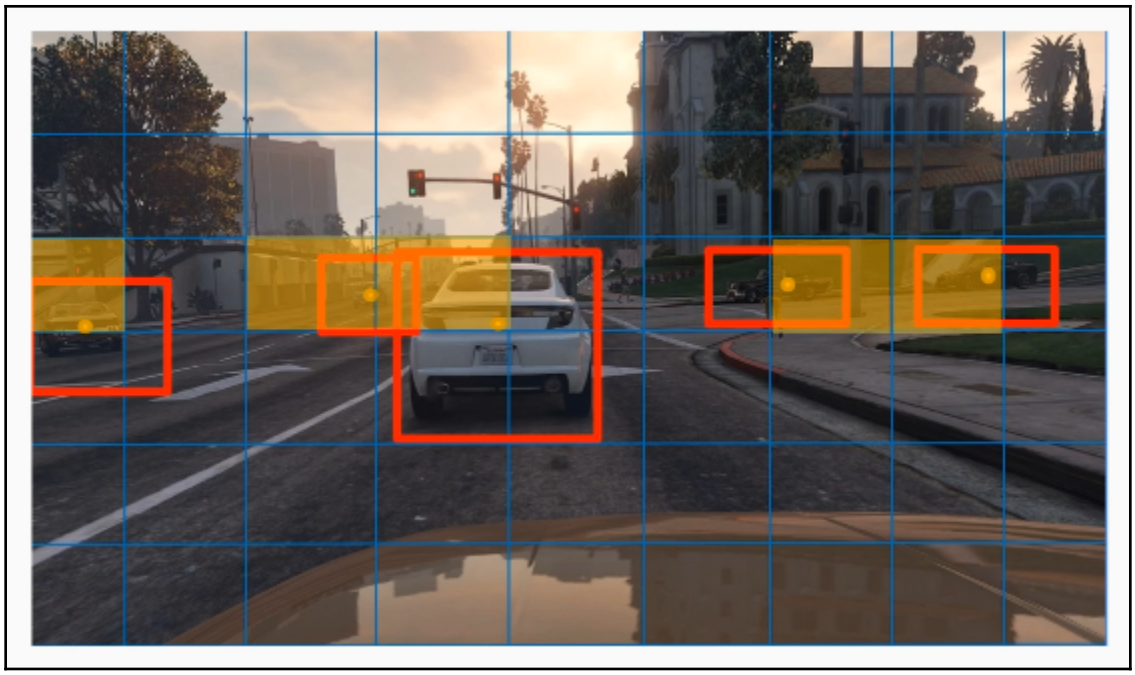

32 X 32 X 3
Conv Same
3 x 3 x 64
32 x 32 x 64
Max Pool
S=2
2 x 2 x 64
16 x 16 x 64
Conv
16x16x256
1 x 1 x 256
Conv
1x1x128
1 x 1 x 128
Conv
1 x 1x 5
1 x 1 x 5
5 Classes
36 X 36 X 3
Conv
Same
3 x 3 x 64
36X36X64
Max
Pool S=2
2 x 2 x 64
18X18X64
Conv
16x16x256
3X3X256
Conv
1x1x128
3X3X128
Conv
1x1x5
3x3x5


1 x 1 x 256
1 x 1 x 128
36 X 36 X 3
Conv
Same
3 x 3 x 64
36X36X64
Max
Pool S=2
2 x 2 x 64
18X18X64
Conv
16x16x256
3X3X256
Conv
1x1x128
3X3X128
Conv
1x1x5
3x3x5

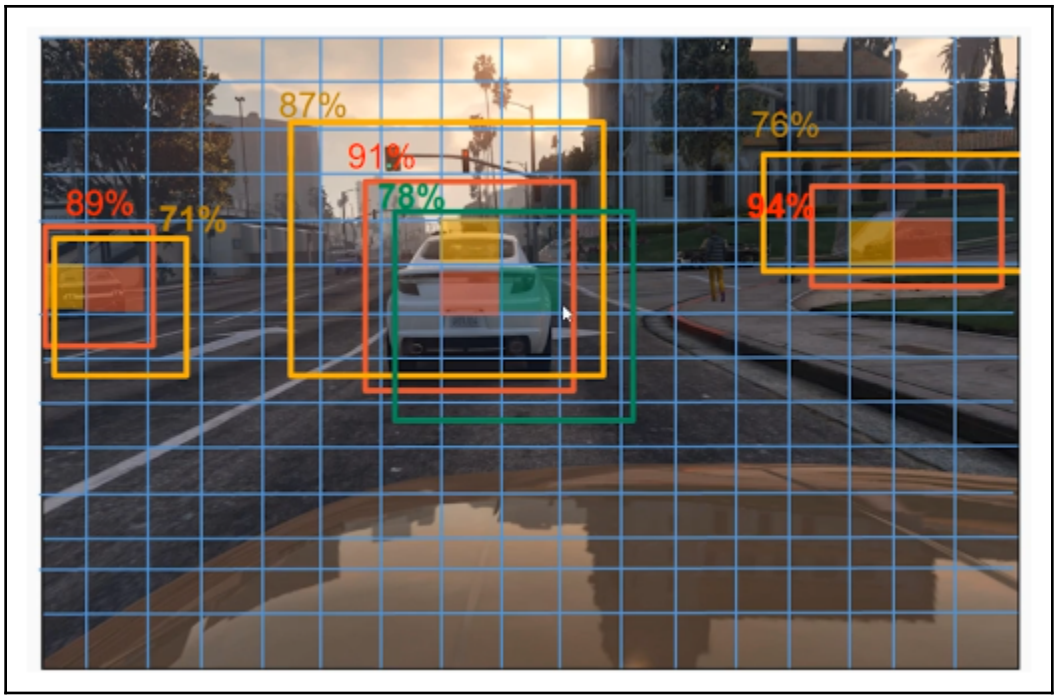

36 X 36 X 3
Conv
Same
3 x 3 x 64
36X36X64
Max
Pool S=2
2 x 2 x 64
18X18X64
Conv
16x16x256
3X3X256
Conv
1x1x128
3X3X128
Conv
1x1x5
3x3x5


Packt

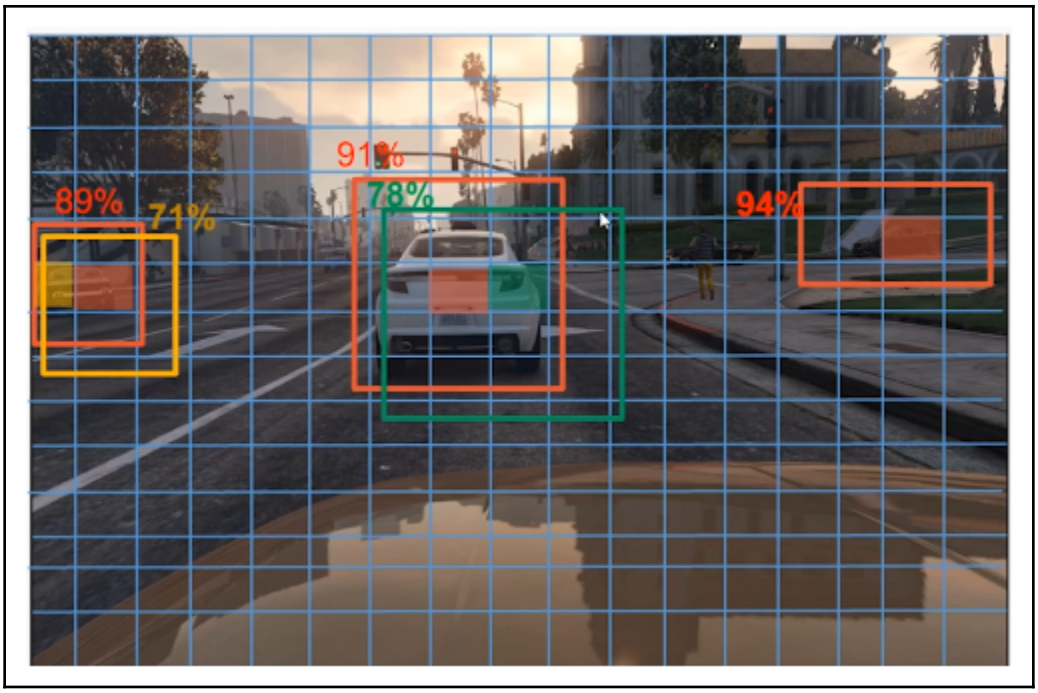


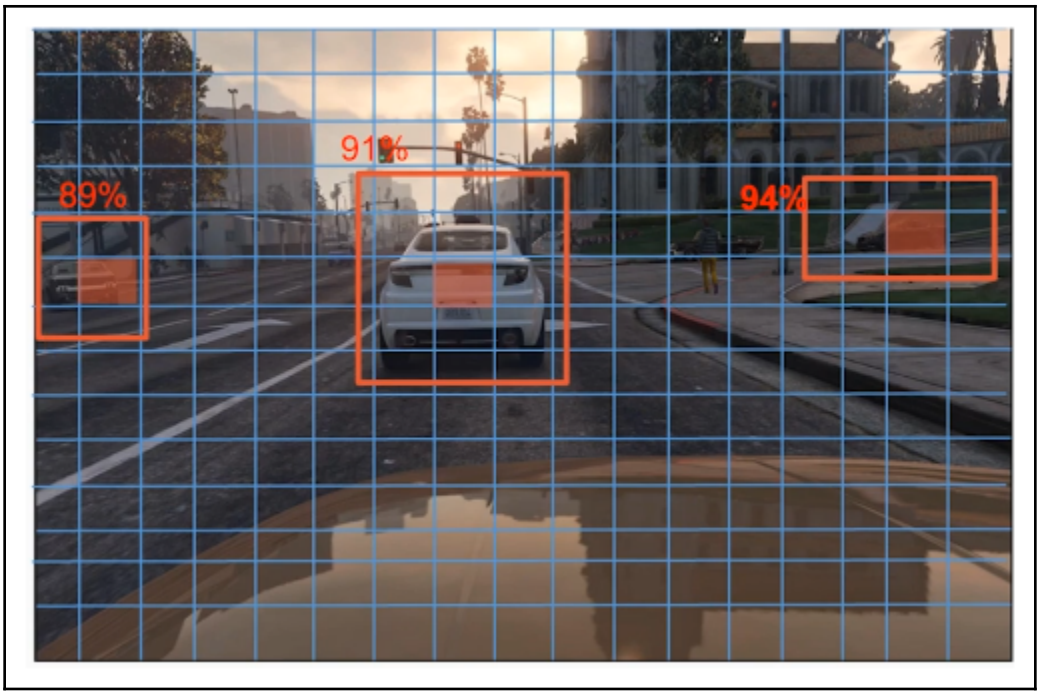


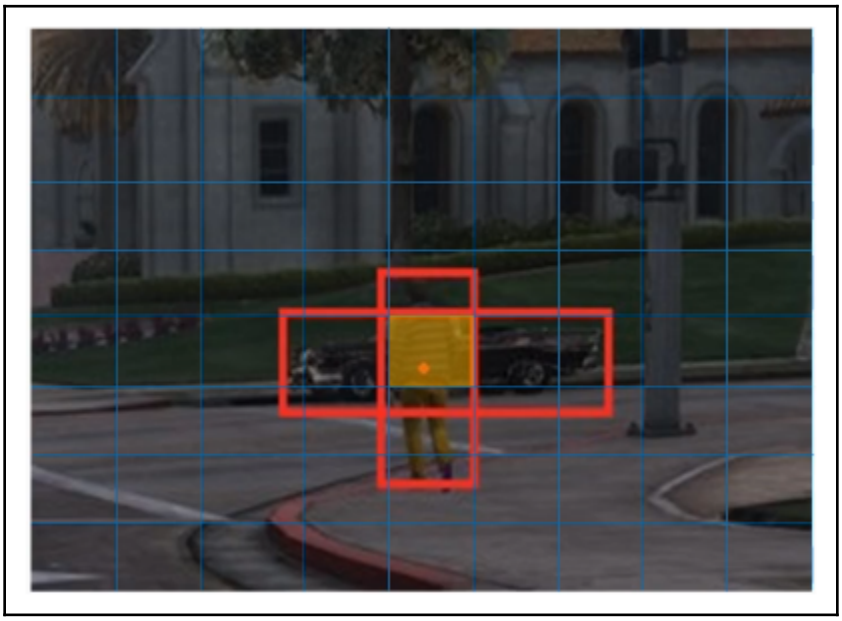
$$\begin{Bmatrix} 0.6 \\ \\ b_x \\ b_y \\ b_h \\ b_w \\ \\ 1 \\ 0 \\ 0 \end{Bmatrix}$$

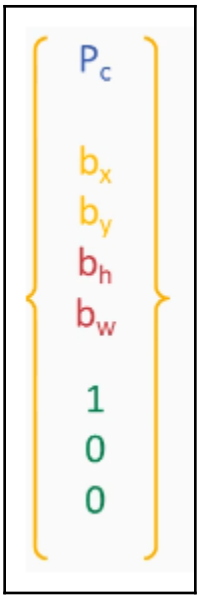

87%
91%
78%
89%
71%
76%
94%

```
list = all bounding boxes

repeat until no boxes left in list
   maxBox=choose box with max P_c
   list.remove(maxBox)
      for box in list
         if(IoU(box, maxBox)>0.5)
            list.remove(box)
   showBox(maxBox);
```

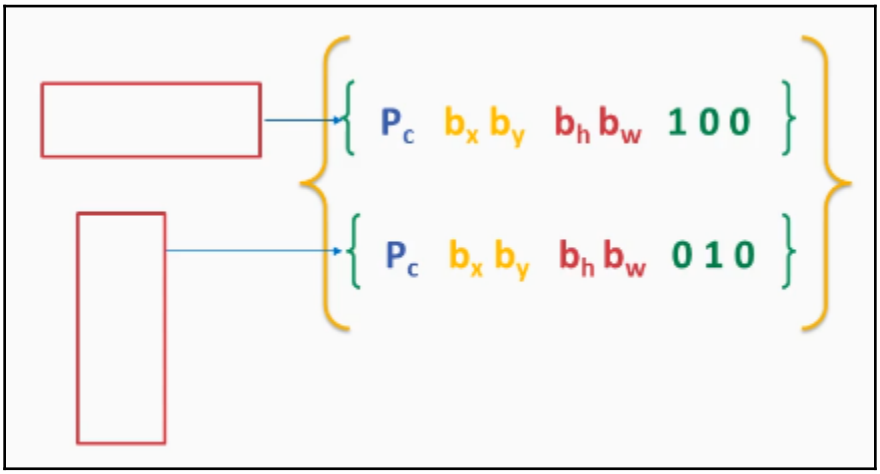

91%
78%
89%
71%
94%


89%
91%
94%



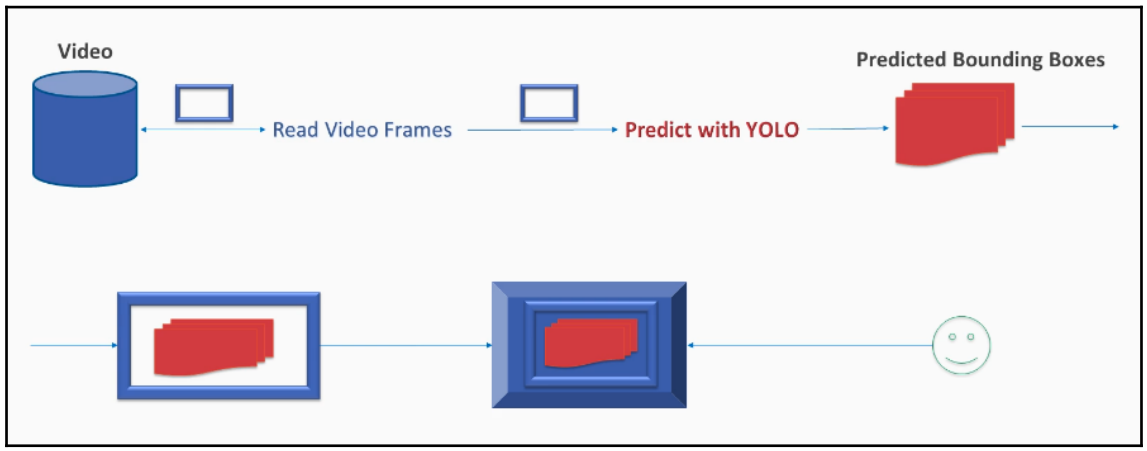


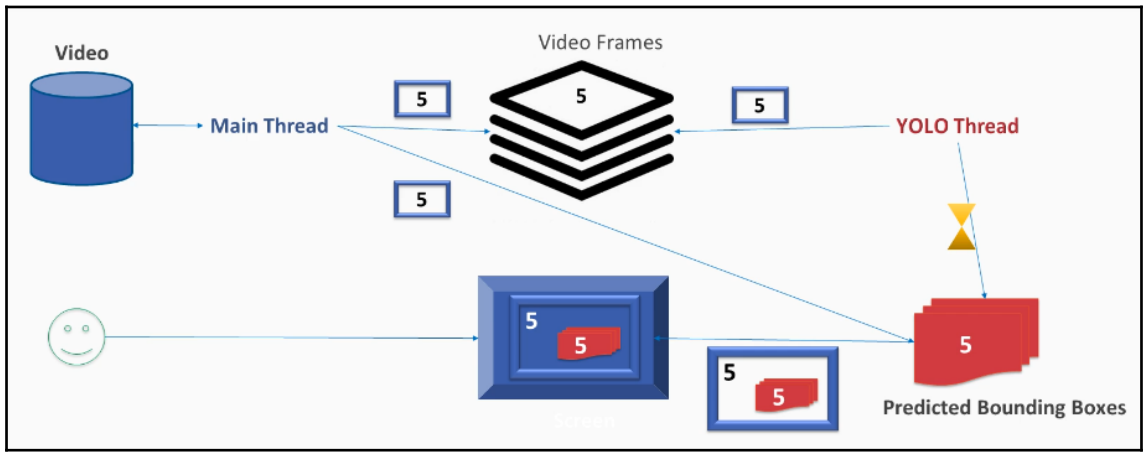
8 x 9 x (5+C)
8 x 9 x 8

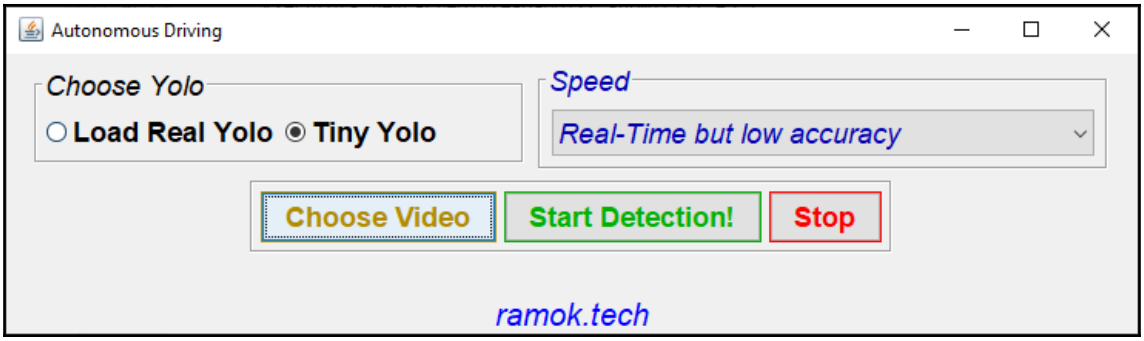


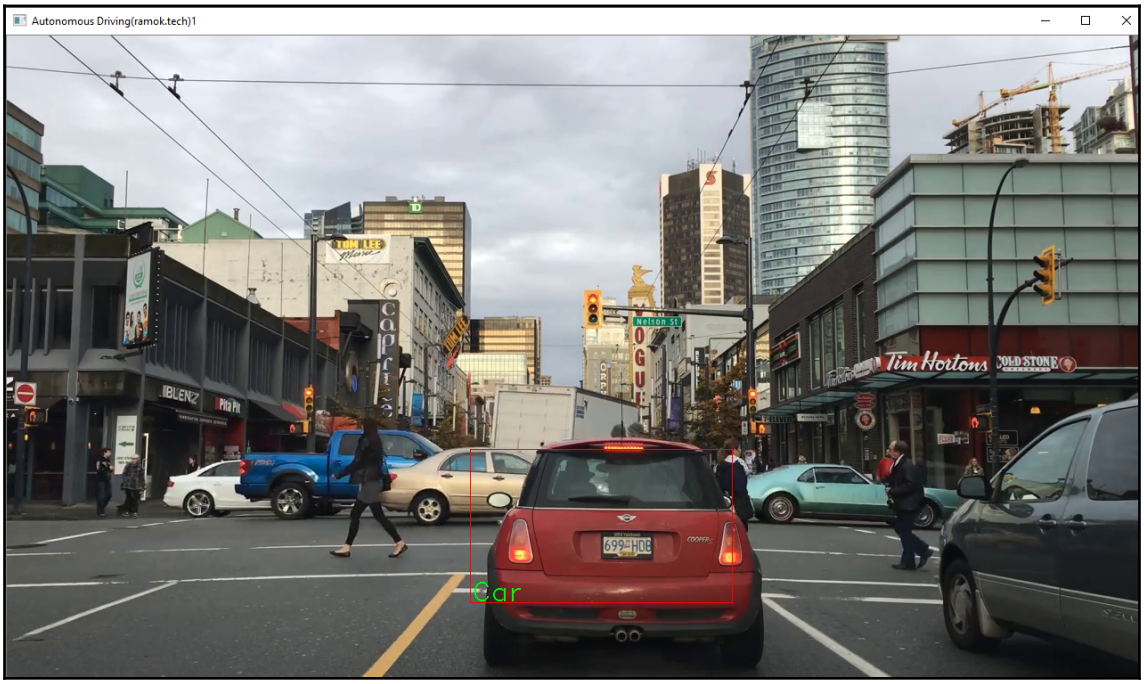
$$8 \times 9 \times 2 \times (5+C)$$
$$8 \times 9 \times 16$$

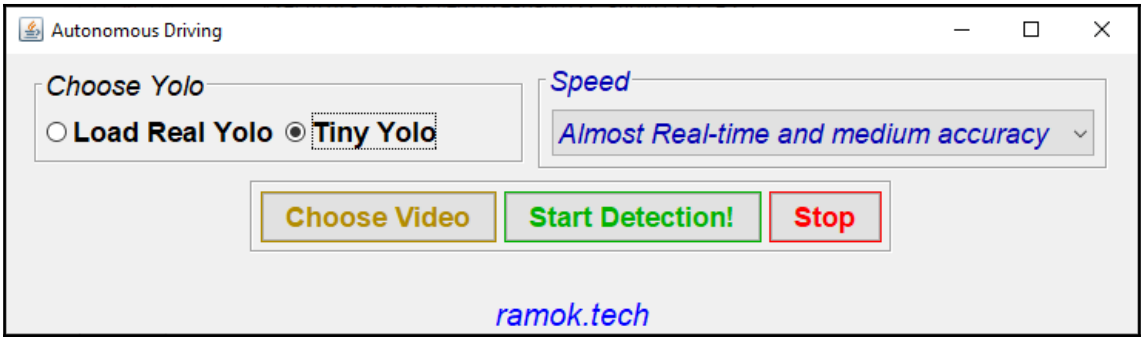
$$8 \times 9 \times 2 \times (5+C)$$
$$8 \times 9 \times 16$$

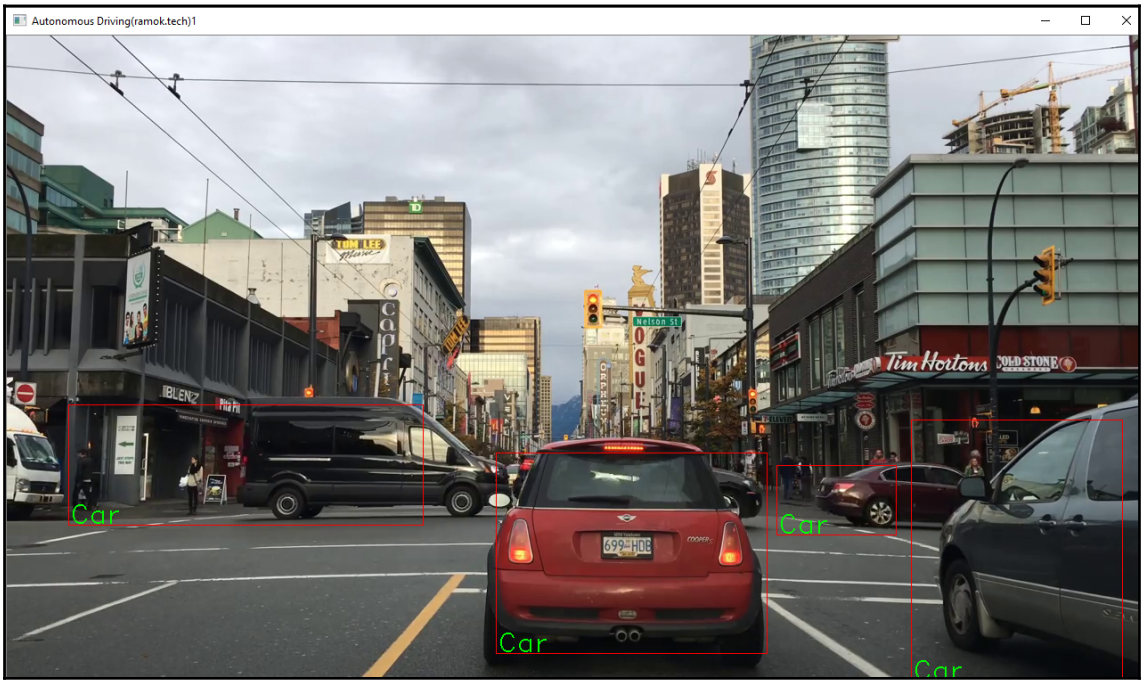




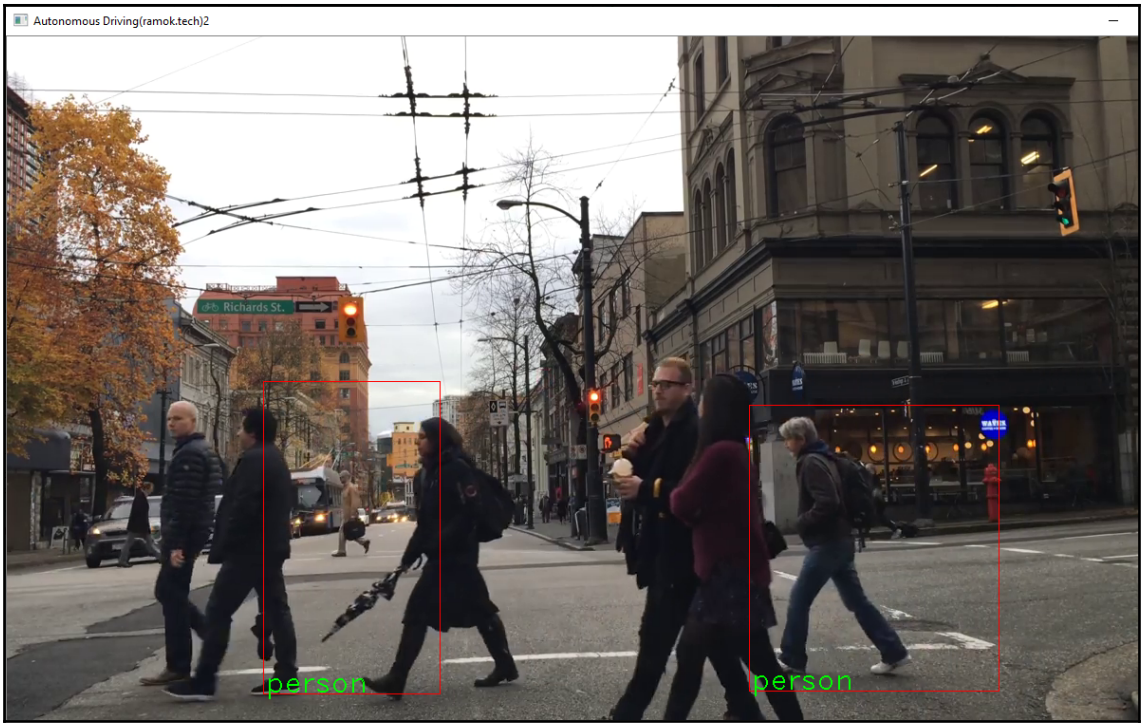

Autonomous Driving
Choose Yolo
Load Real Yolo
Tiny Yolo
Speed
Real-Time but low accuracy
Choose Video
Start Detection!
Stop
ramok.tech

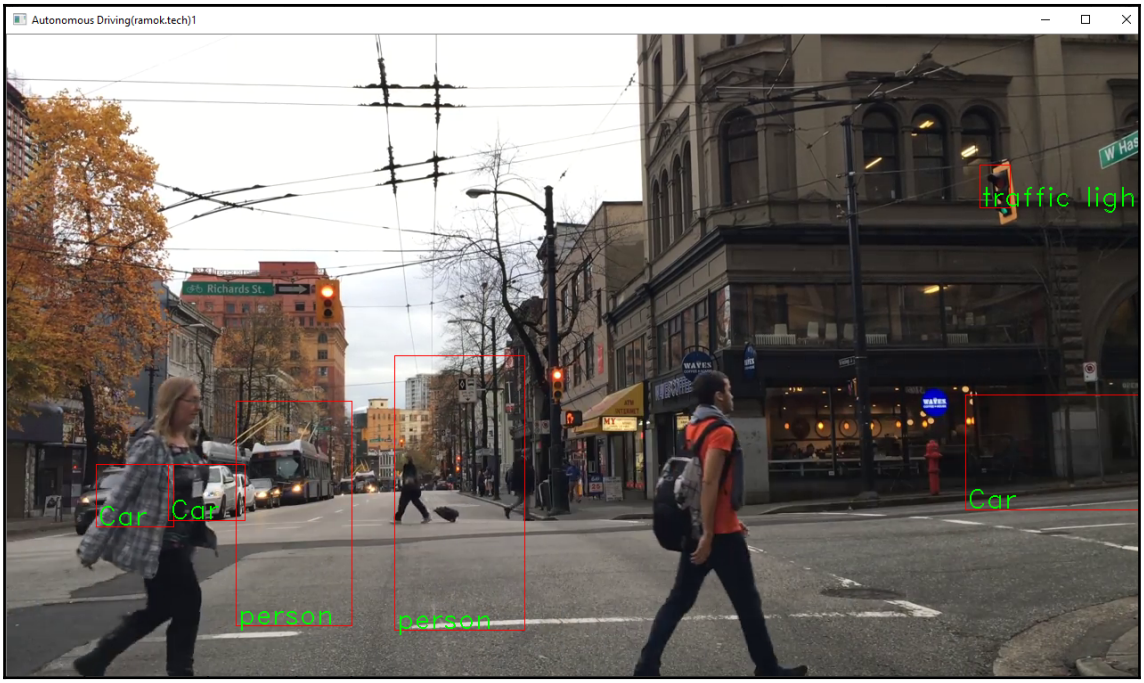

Autonomous Driving(ramok.tech)1
Car


Autonomous Driving
Choose Yolo
Load Real Yolo
Tiny Yolo
Speed
Almost Real-time and medium accuracy
Choose Video
Start Detection!
Stop
ramok.tech

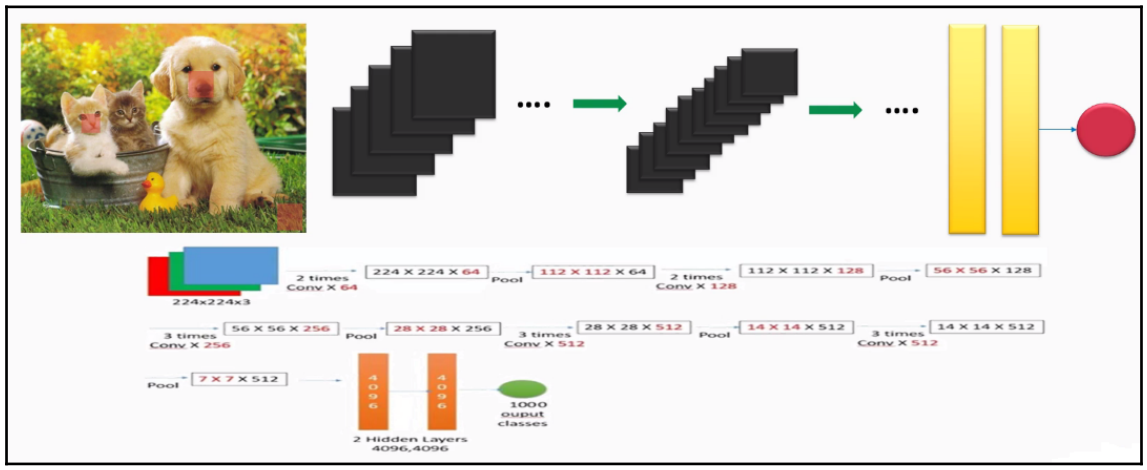

Autonomous Driving(ramok.tech)1
Car
Car
Car
Car

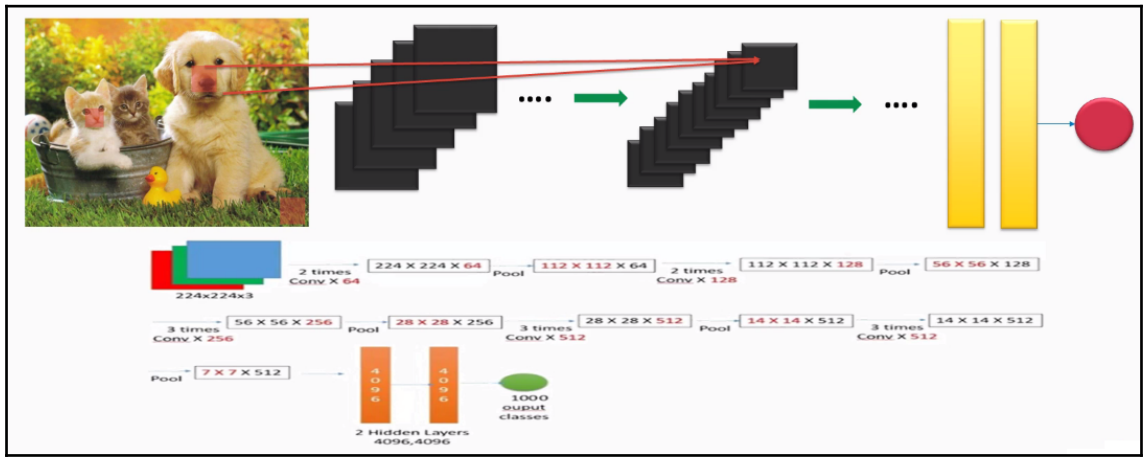

Autonomous Driving(ramok.tech)1
TOM LEE
Caprice
Pita Pit
Nelson St
traffic light
Car
Car
Car
Car

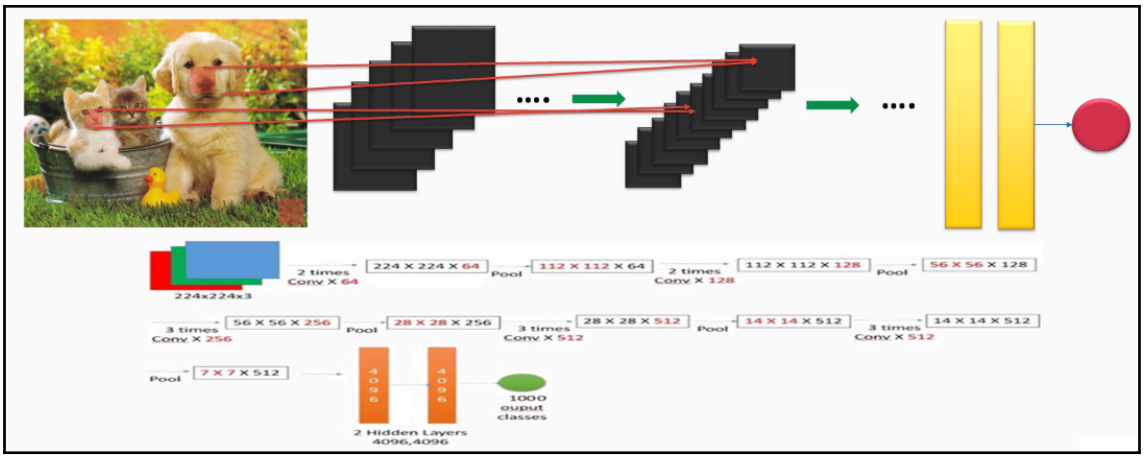

Autonomous Driving(ramok.tech)2
Richards St.
person
person

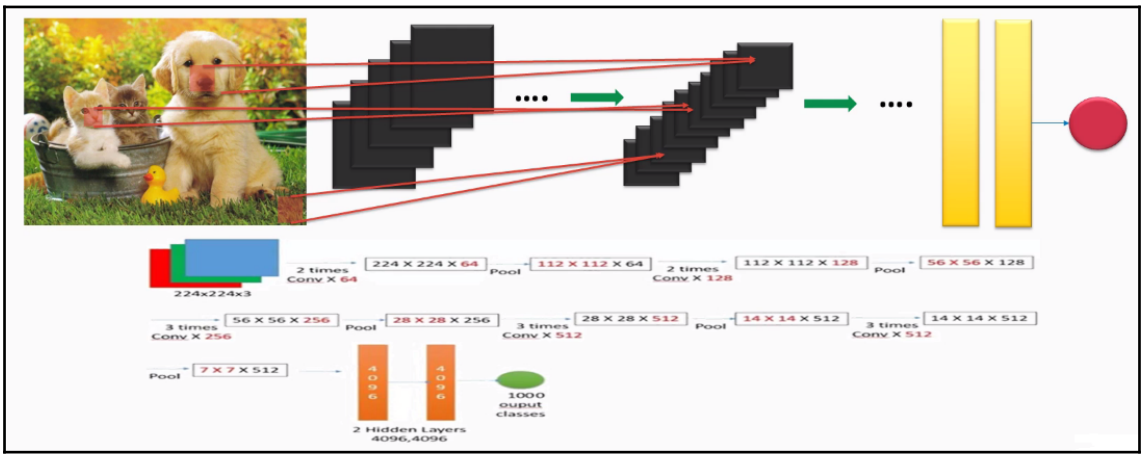

Autonomous Driving(ramok.tech)1
Richards St.
Car
Car
person
person
Car
Traffic ligh

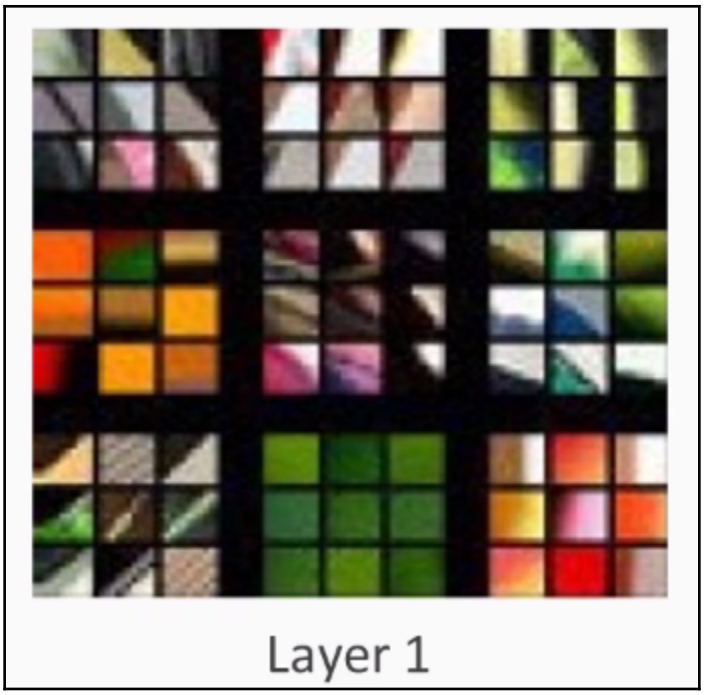
Chapter 5: Creating Art with Neural Style Transfer

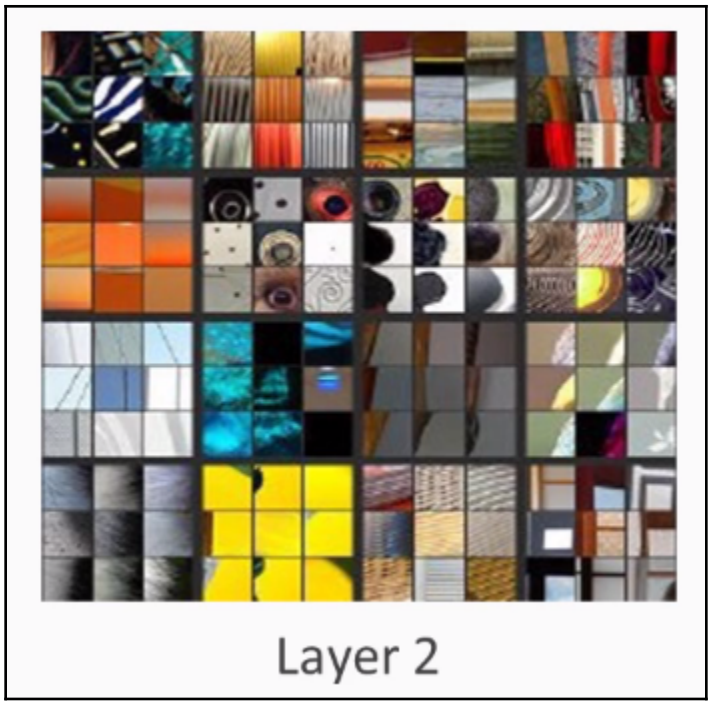


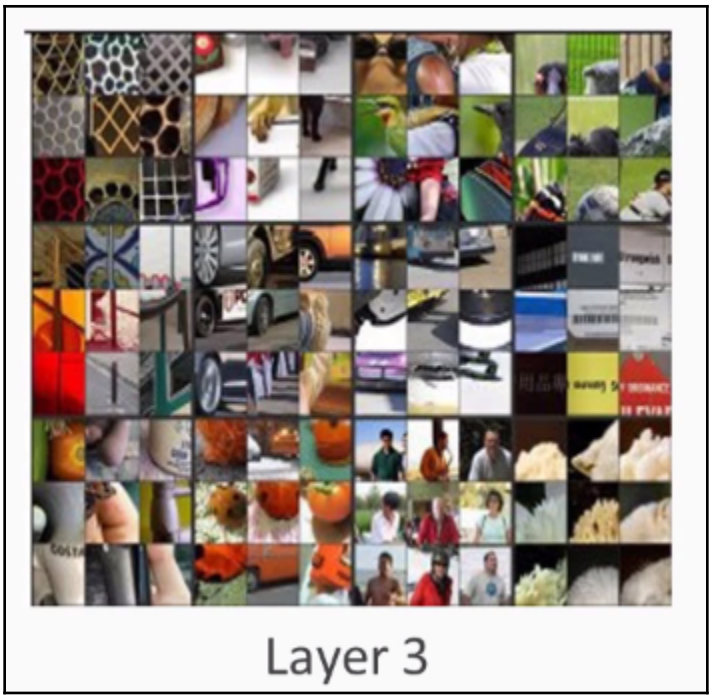



224x224x3
2 times Conv X 64
224 X 224 X 64
Pool
112 X 112 X 64
2 times Conv X 128
112 X 112 X 128
Pool
56 X 56 X 128
3 times Conv X 256
56 X 56 X 256
Pool
28 X 28 X 256
3 times Conv X 512
28 X 28 X 512
Pool
14 X 14 X 512
3 times Conv X 512
14 X 14 X 512
Pool
7 X 7 X 512
4096
4096
1000 ouput classes
2 Hidden Layers 4096,4096


224x224x3
2 times Conv X 64
224 X 224 X 64
Pool
112 X 112 X 64
2 times Conv X 128
112 X 112 X 128
Pool
56 X 56 X 128
3 times Conv X 256
56 X 56 X 256
Pool
28 X 28 X 256
3 times Conv X 512
28 X 28 X 512
Pool
14 X 14 X 512
3 times Conv X 512
14 X 14 X 512
Pool
7 X 7 X 512
4096
4096
1000 ouput classes
2 Hidden Layers 4096,4096

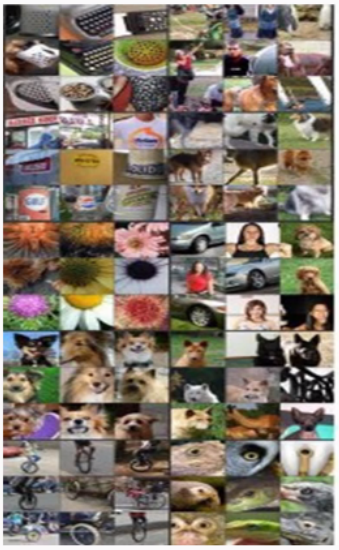

Layer 1


Layer 2

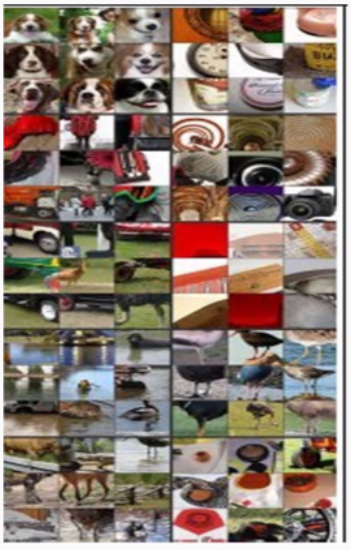


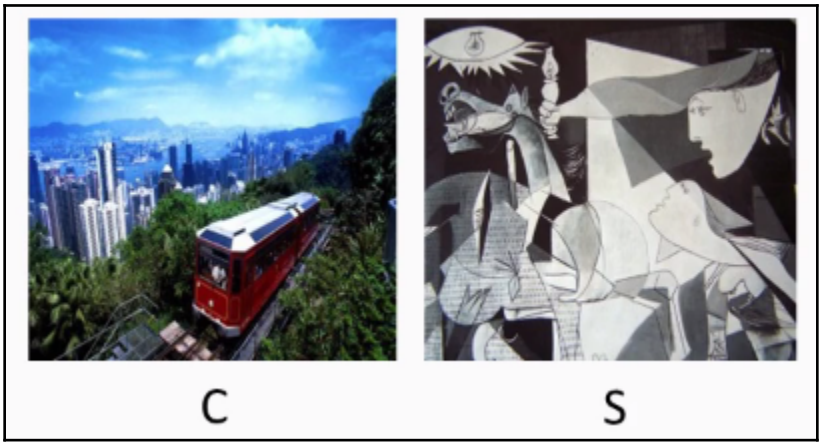
Layer 3

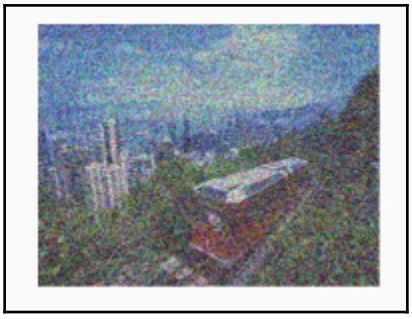
Layer 4

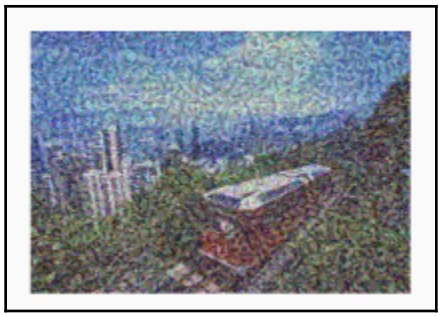


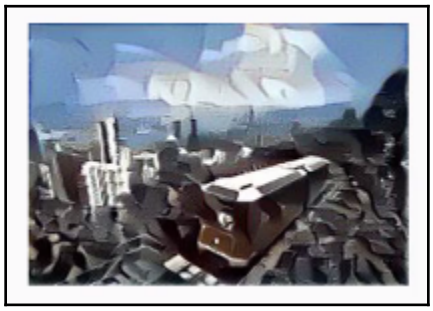
Layer 5

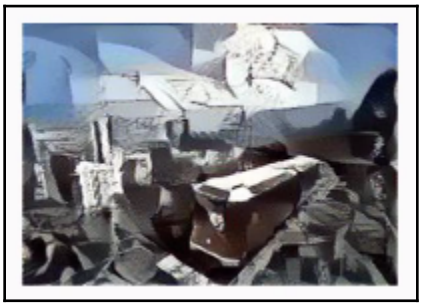


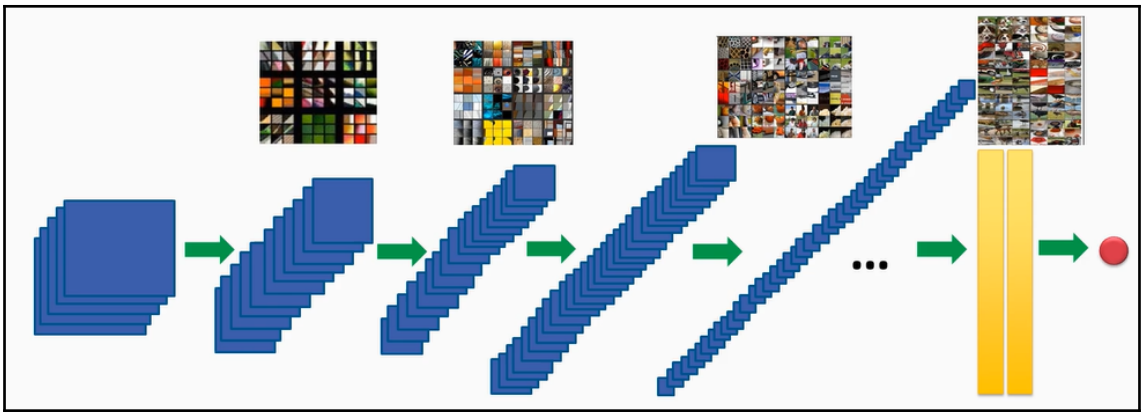

C
S

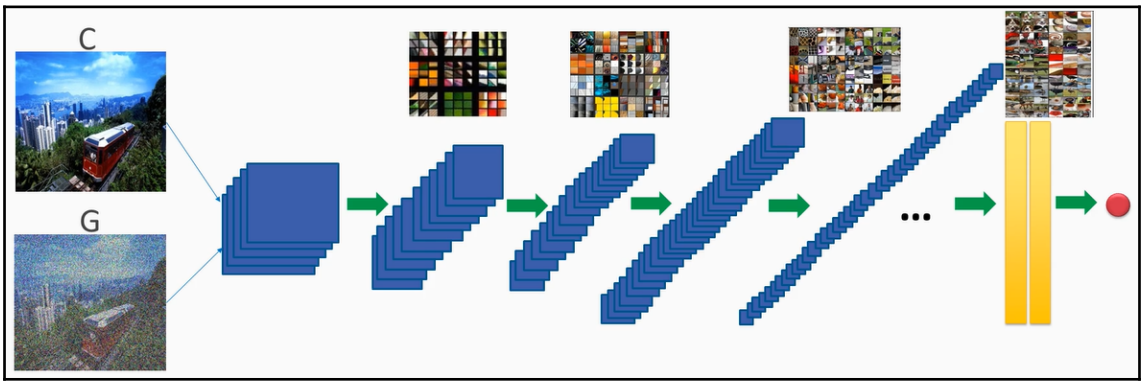


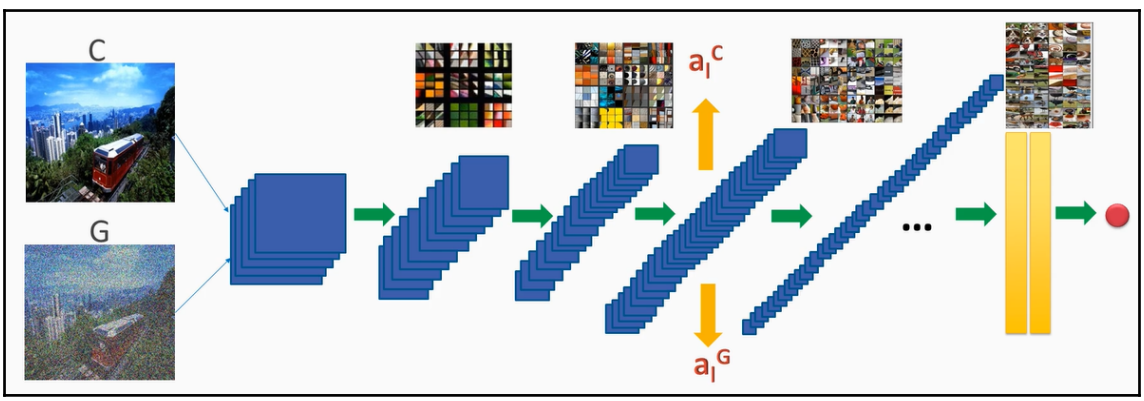


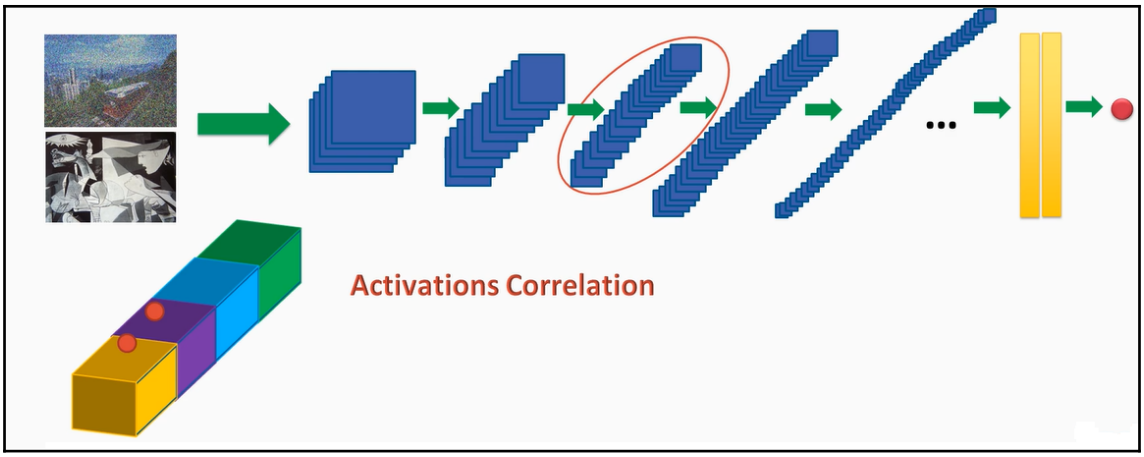


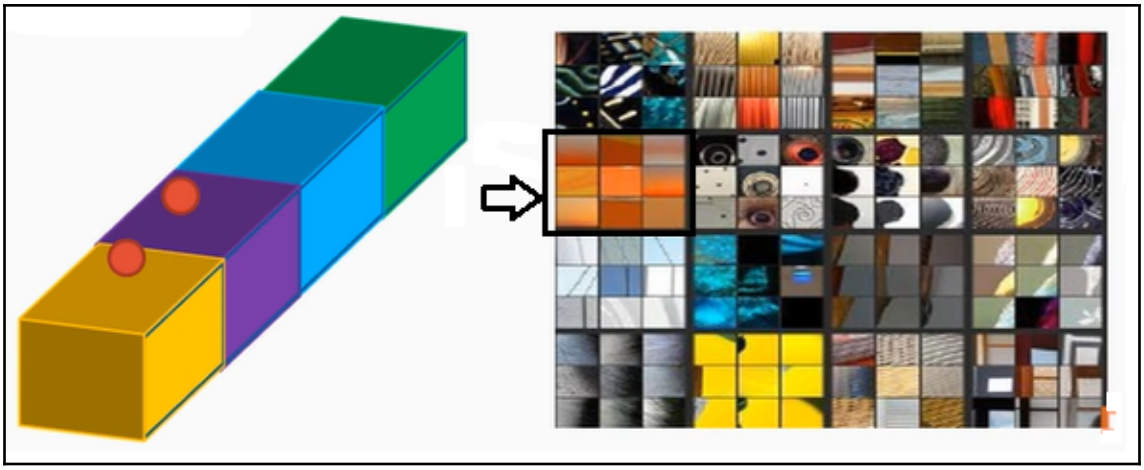


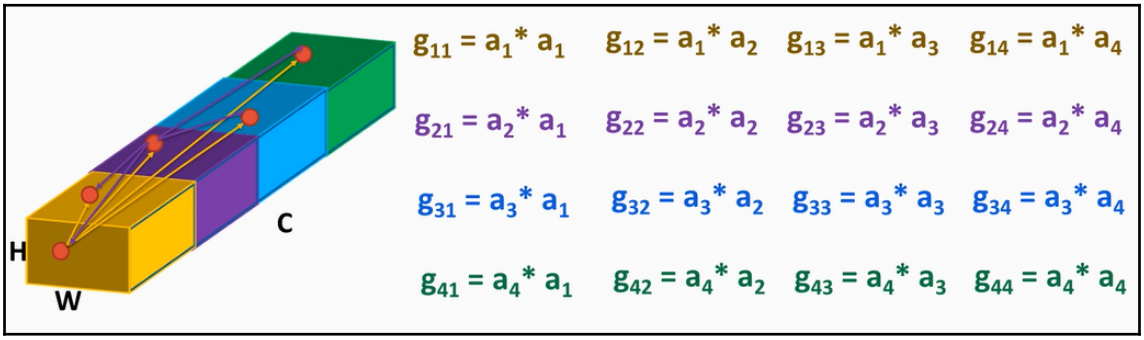


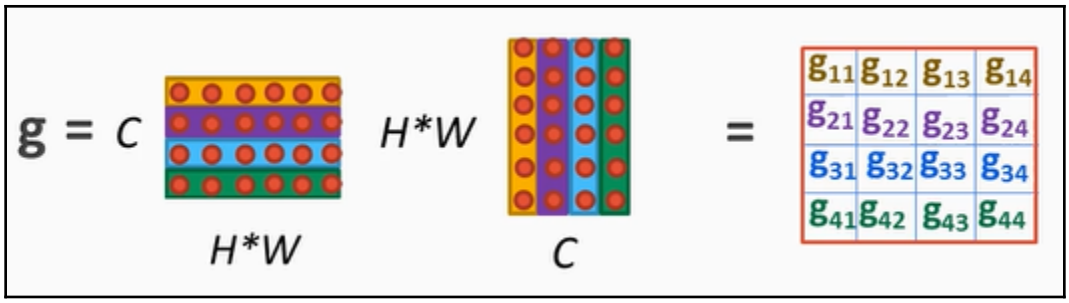

C
G

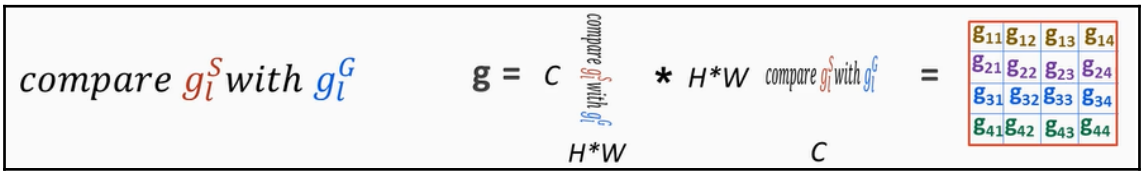

C
G
a_l^C
a_l^G


Activations Correlation




H
W
C
$g_{11} = a_1 * a_1$ $g_{12} = a_1 * a_2$ $g_{13} = a_1 * a_3$ $g_{14} = a_1 * a_4$
$g_{21} = a_2 * a_1$ $g_{22} = a_2 * a_2$ $g_{23} = a_2 * a_3$ $g_{24} = a_2 * a_4$
$g_{31} = a_3 * a_1$ $g_{32} = a_3 * a_2$ $g_{33} = a_3 * a_3$ $g_{34} = a_3 * a_4$
$g_{41} = a_4 * a_1$ $g_{42} = a_4 * a_2$ $g_{43} = a_4 * a_3$ $g_{44} = a_4 * a_4$



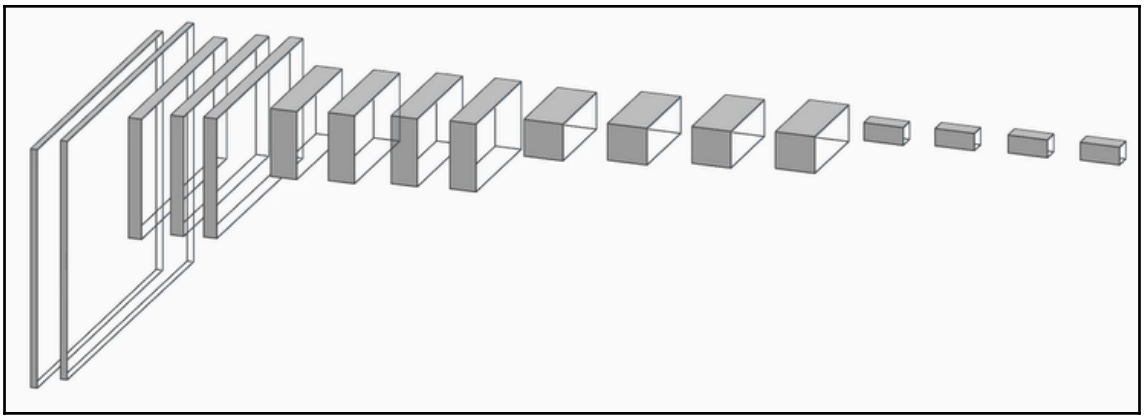


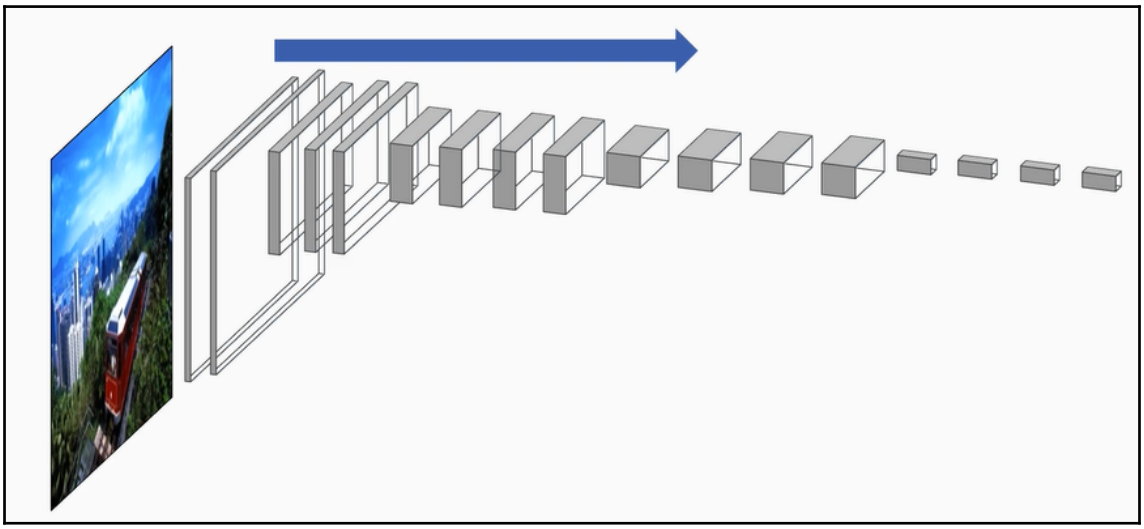
$$J(G) = \alpha * J(C,G) + \beta * J(S,G)$$

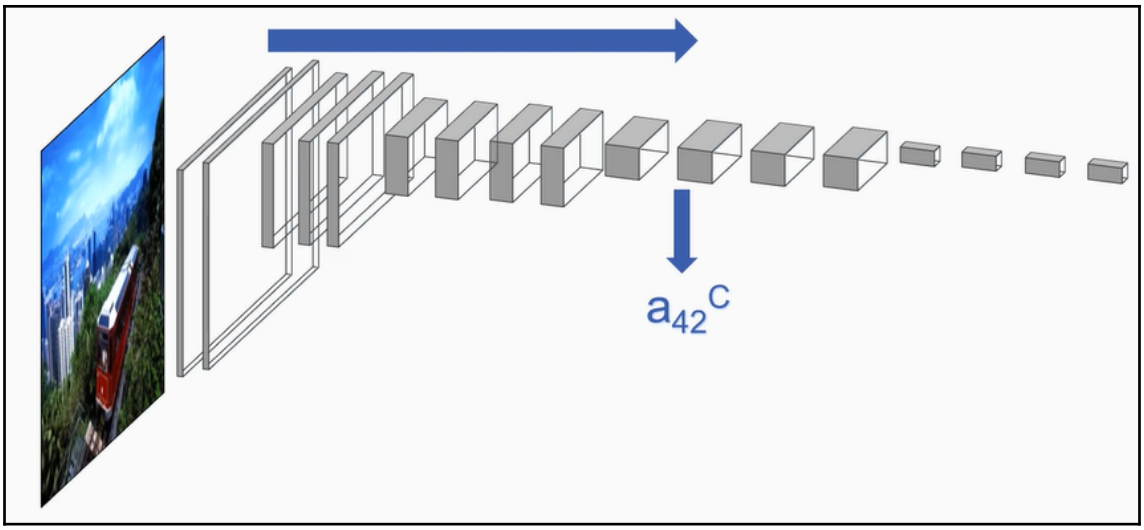
$$J(S,G) = \frac{1}{4 * (W * H * C)^2} * \left\| g_l^S - g_l^G \right\|^2$$

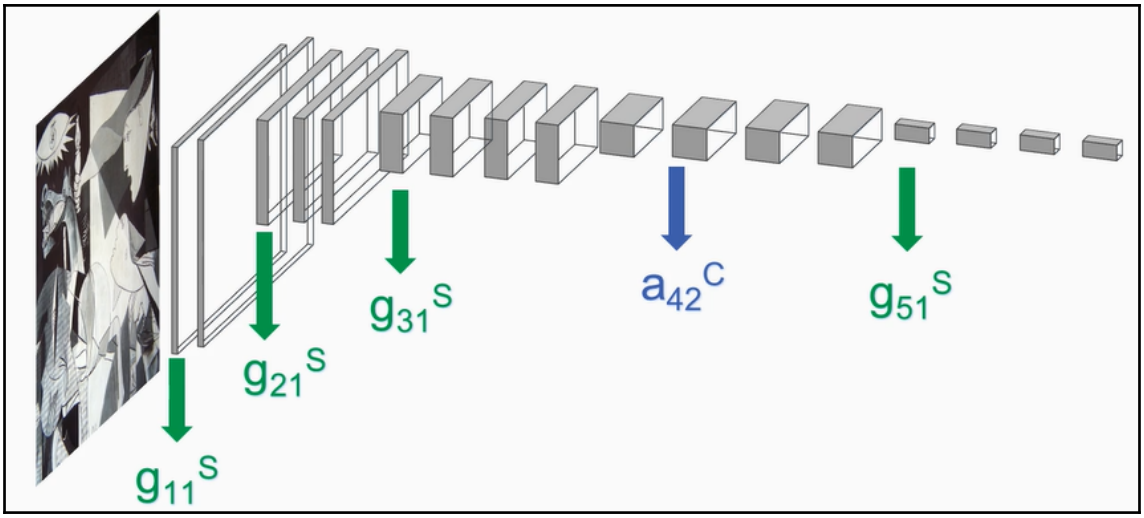
$$g = a_l * a_l^T ; dg = 2 * a_l^T$$

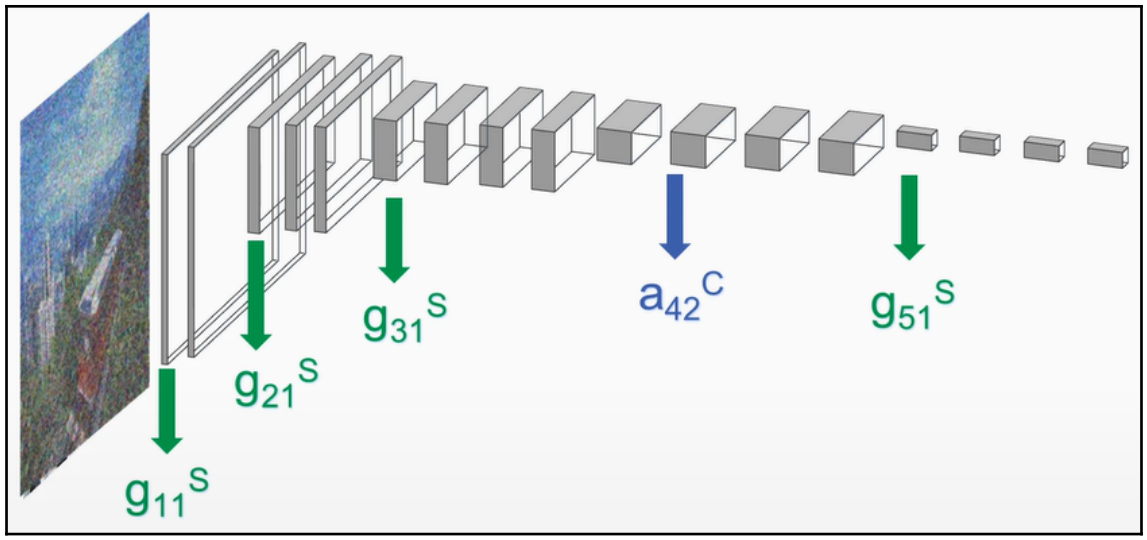
$$\frac{dJ(S,G)}{dG} = \frac{1}{(W * H * C)^2} * \left\| g_l^S - g_l^G \right\| * (a_l^G)^T$$

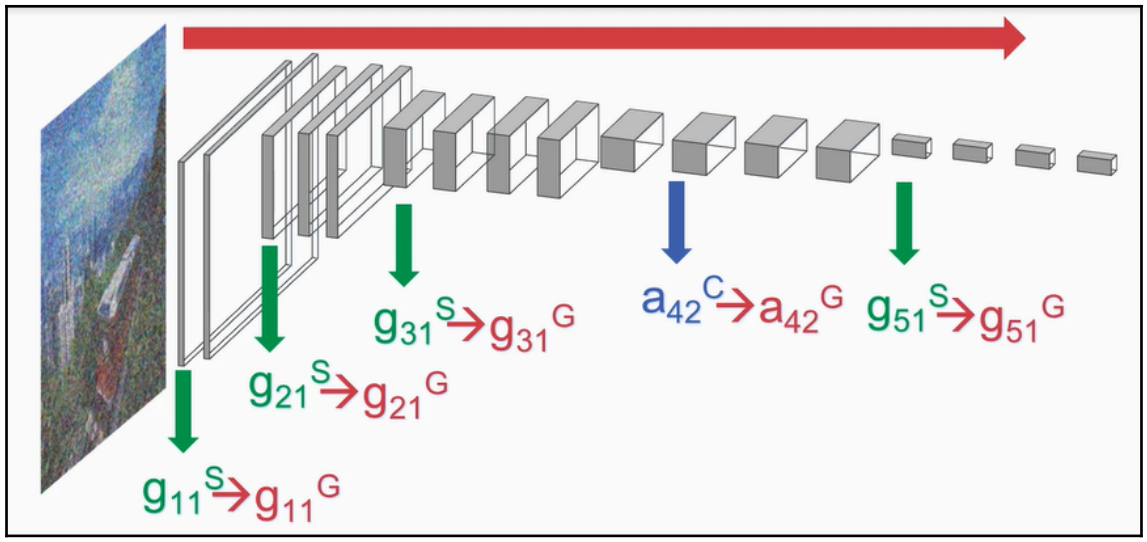





a_{42}^{C}


g_{11}^{S}
g_{21}^{S}
g_{31}^{S}
a_{42}^{C}
g_{51}^{S}



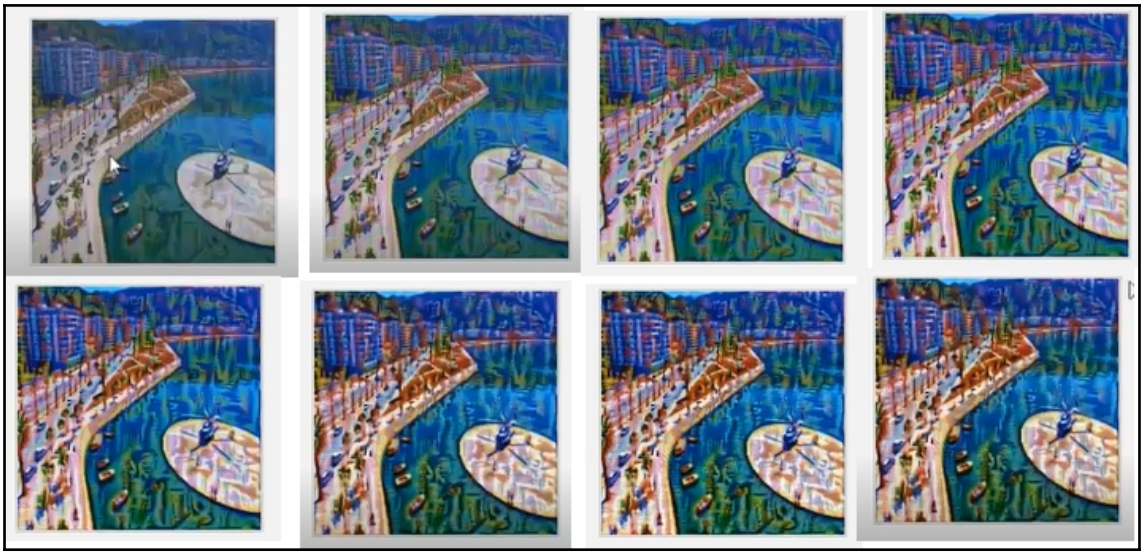


$$J(G) = \alpha * \boldsymbol{J}(C, G) + \beta * (\boldsymbol{k1} * J(S, G)^{l1} + \boldsymbol{k2} * J(S, G)^{l2} + \boldsymbol{k3} * J(S, G)^{l3} + \boldsymbol{k4} * J(S, G)^{l4})$$

$$J(C,G) = \frac{1}{4 * W * H * C} * \left\| a_l^C - a_l^G \right\|^2$$

$$J(S,G) = \frac{1}{4 * (W * H * C)^2} * \left\| g_l^S - g_l^G \right\|^2$$

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" ...
[2019-02-06 17:51:30]Loaded [CpuBackend] backend
[2019-02-06 17:51:30]given scan urls are empty. set urls in the configuration
[2019-02-06 17:51:30]Number of threads used for NativeOps: 4
[2019-02-06 17:51:30]Reflections took 47 ms to scan 13 urls, producing 31 keys and 227 values
[2019-02-06 17:51:30]Number of threads used for BLAS: 4
[2019-02-06 17:51:30]Backend used: [CPU]; OS: [Windows 10]
[2019-02-06 17:51:30]Cores: [8]; Memory: [4.0GB];
[2019-02-06 17:51:30]Blas vendor: [OPENBLAS]
[2019-02-06 17:51:31]Downloading model to C:\Users\Admin\.deeplearning4j\vgg16_dl4j_inference.zip
```



Chapter 6: Face Recognition


Face Verification
Is Person with ID X?


Face Recognition
Is any of known Persons in group?

98%

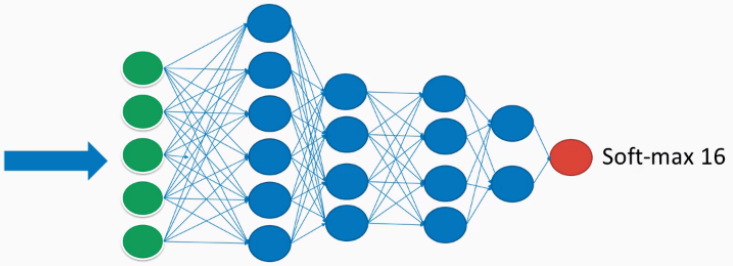
Face Verification

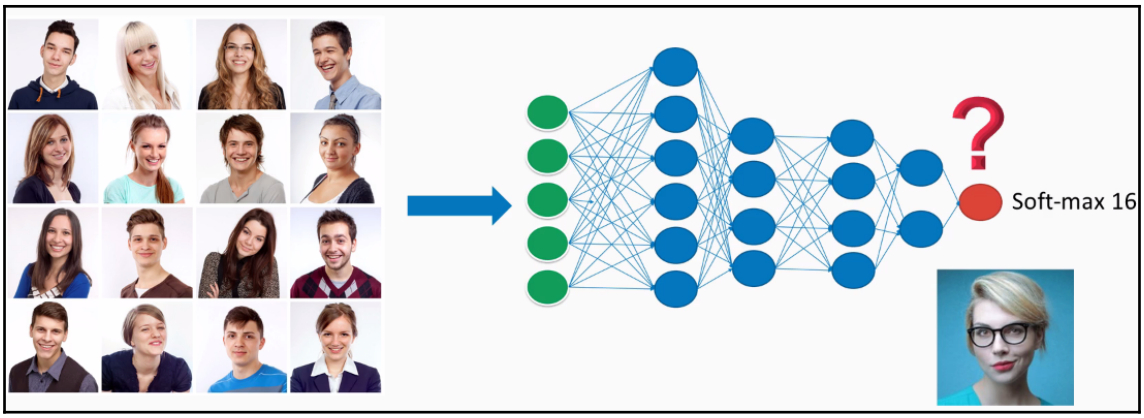
Is Person with ID X?

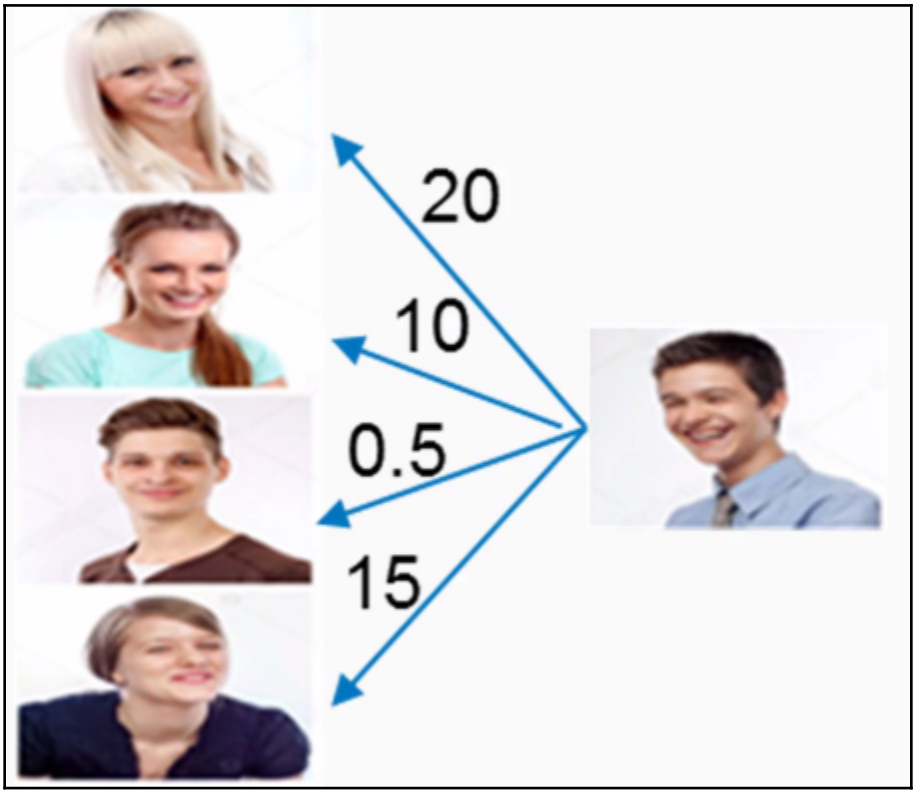
16*2%=32%

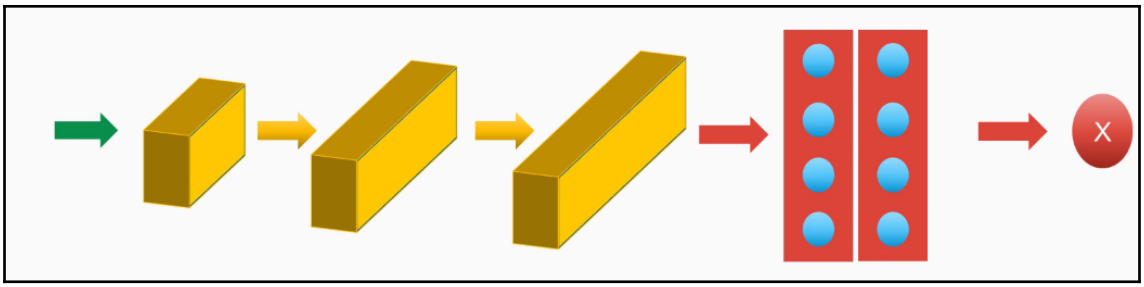
Face Recognition

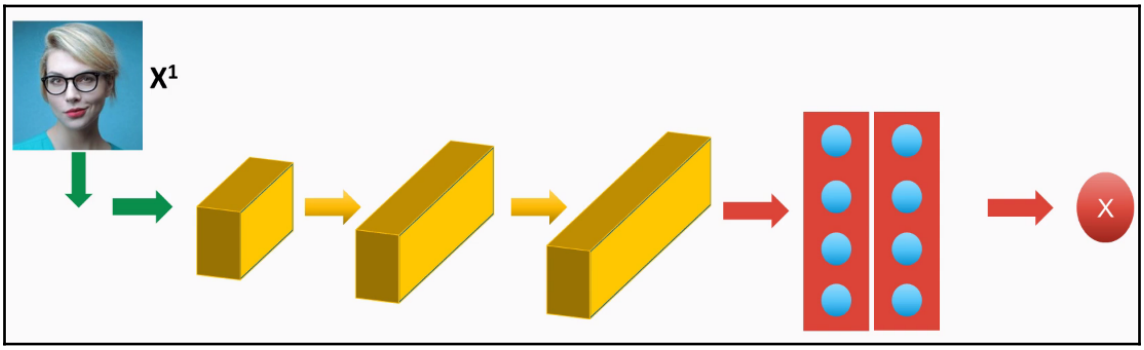
Is any of known Persons in group?

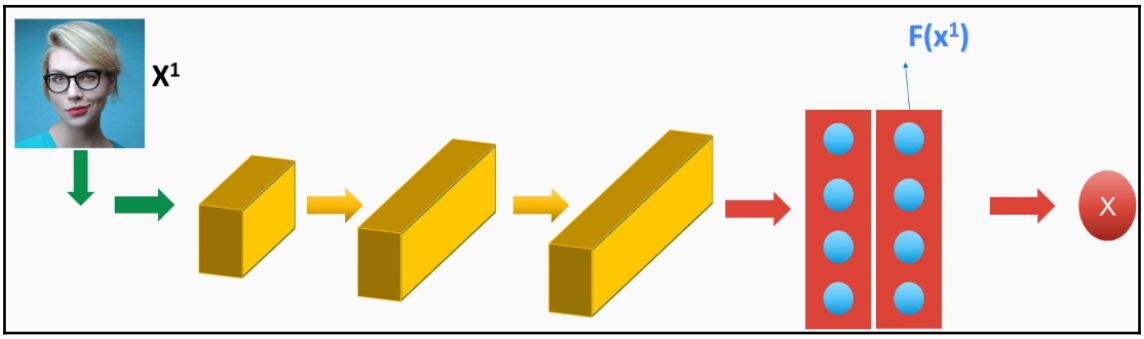


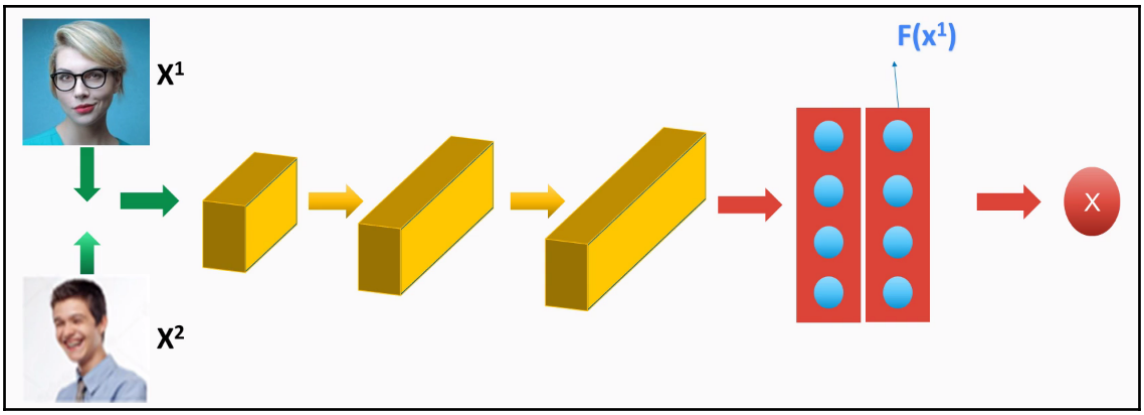

?
Soft-max 16

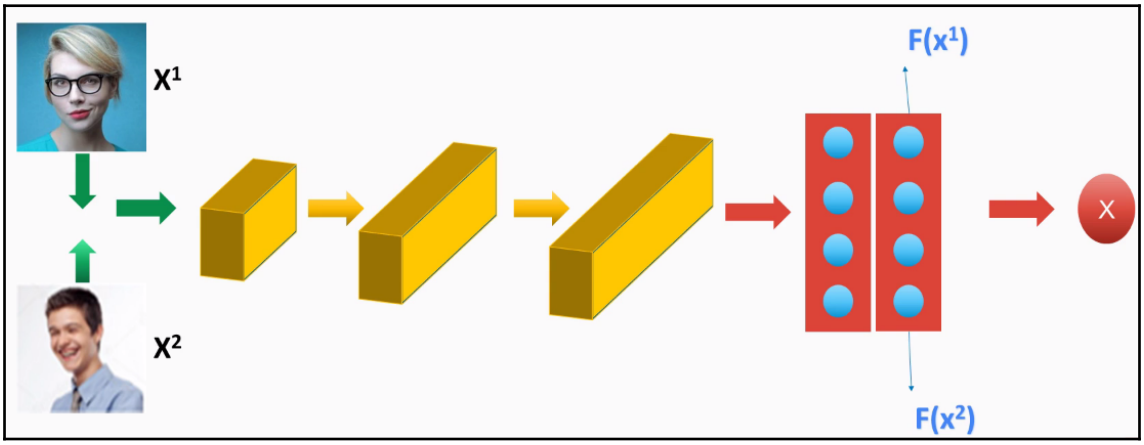

20
10
0.5
15

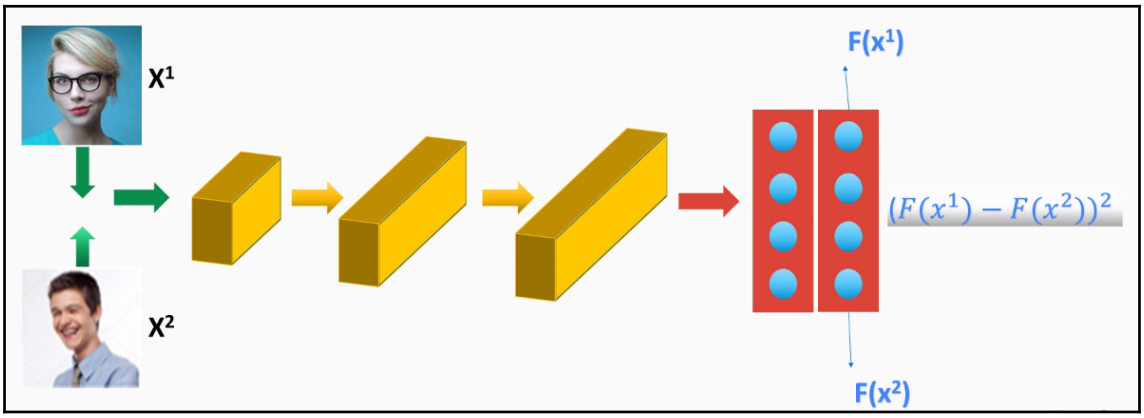

X

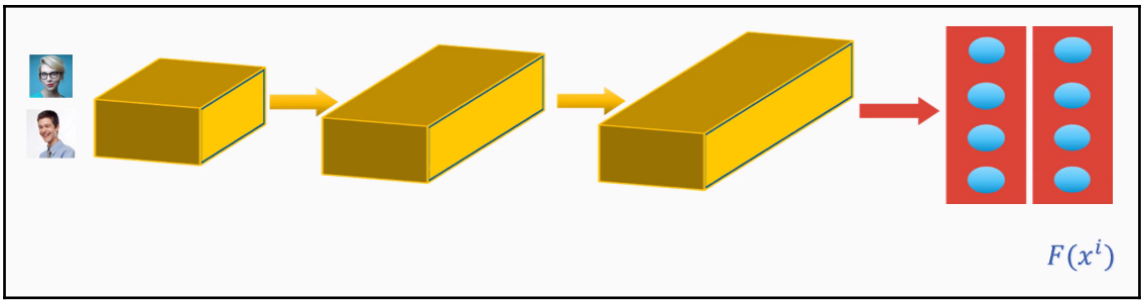

X1
X

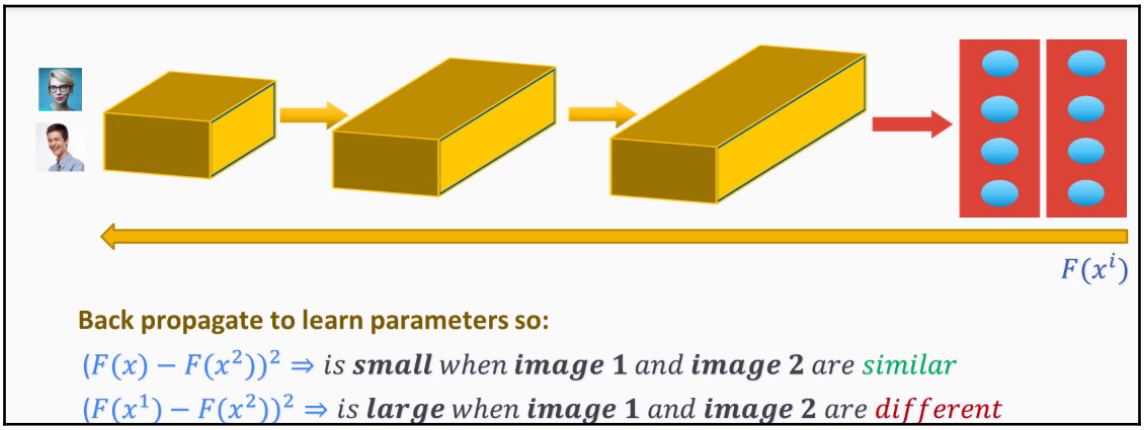

X1
F(x1)
X


X¹
X²
F(x¹)
X


X¹
X²
F(x¹)
F(x²)
X


X¹
X²
F(x¹)
F(x²)
$(F(x^1) - F(x^2))^2$


$F(x^i)$


$F(x^i)$
Back propagate to learn parameters so:
$(F(x) - F(x^2))^2 \Rightarrow$ is **small** when **image 1** and **image 2** are similar
$(F(x^1) - F(x^2))^2 \Rightarrow$ is **large** when **image 1** and **image 2** are different


Anchor

Positive

$$d(A,P)=(F(A)-F(P))^2 \approx 0$$

Negative

$$d(A,N)=(F(A)-F(N))^2 > 0$$

$$(F(A)-F(P))^2 \approx 0 \;;\; (F(A)-F(N))^2 > 0$$

$$(F(A)-F(P))^2 \leq (F(A)-F(N))^2$$

$$(F(A)-F(P))^2 - (F(A)-F(N))^2 \leq 0$$

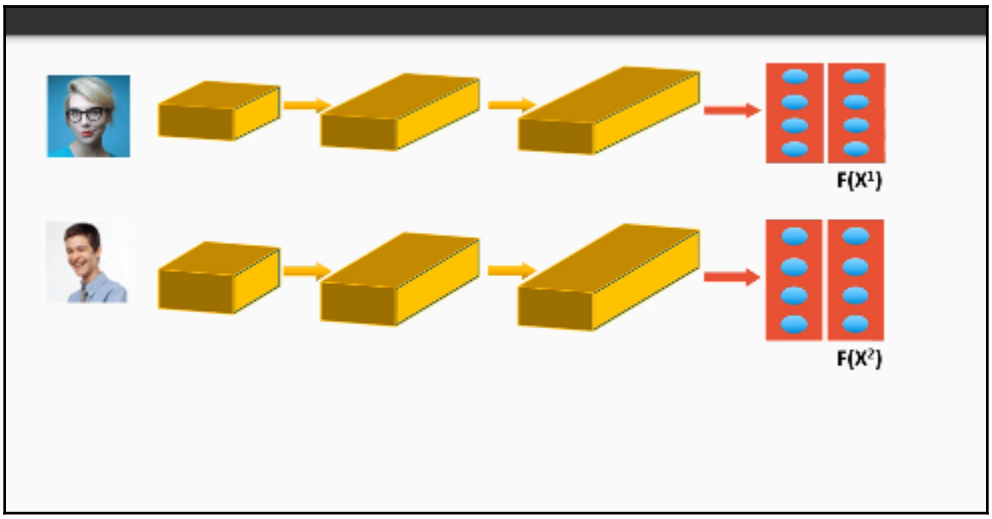
$$(F(A) - F(P))^2 - (F(A) - F(N))^2 \leq -\varepsilon$$

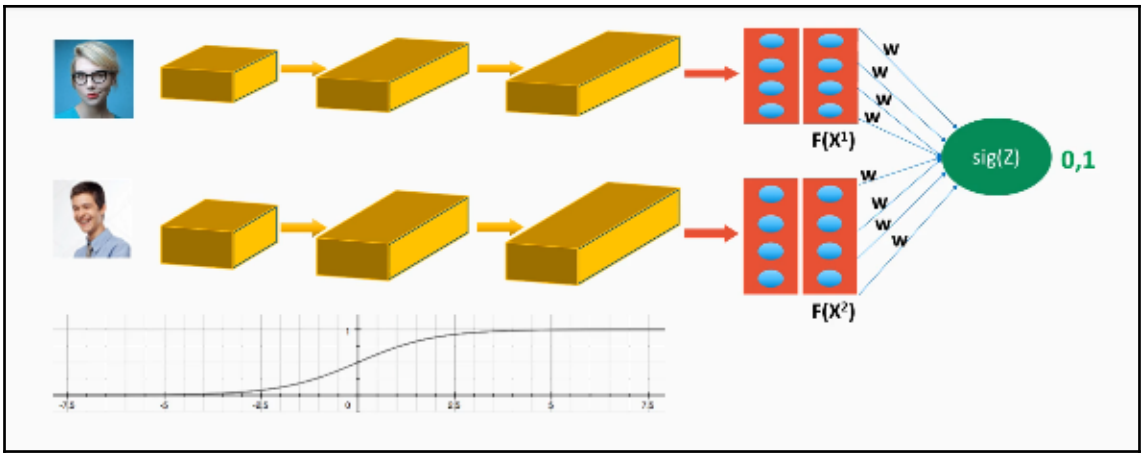
$$(F(A) - F(P))^2 - (F(A) - F(N))^2 + \varepsilon \leq 0$$

$$0.5 - (F(A) - F(N))^2 + 0.2 \leq 0$$

When negative examples N are chosen randomly then the condition is easily satisfied: d(A,P)+"ε"≤"d(A,N)"; since d(A,N) will be always big enough

Choose triplets that are hard to train d(A,P)+"ε"≤"d(A,N)"; so network has to push up hard d(A,N) and down d(A,P) to satisfy the margin "ε"

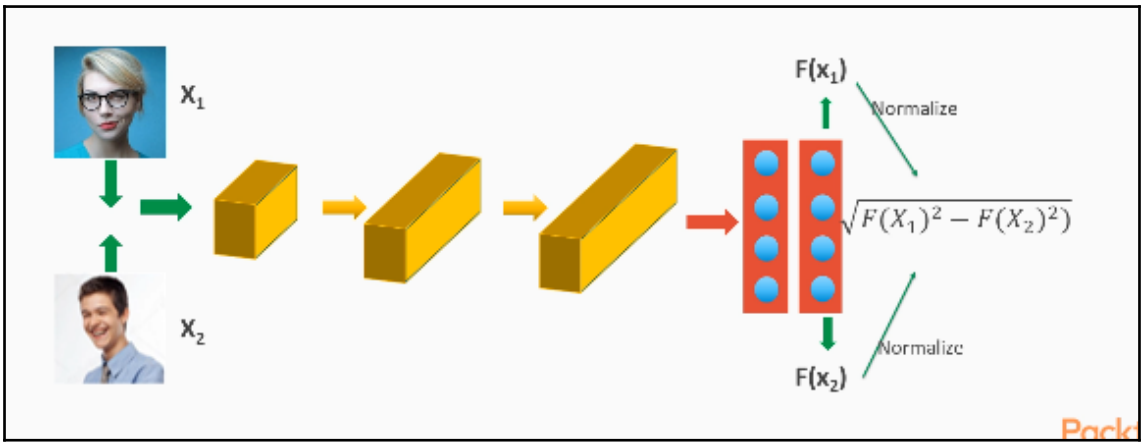


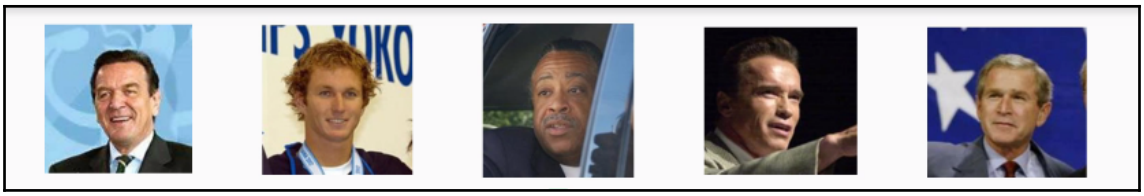


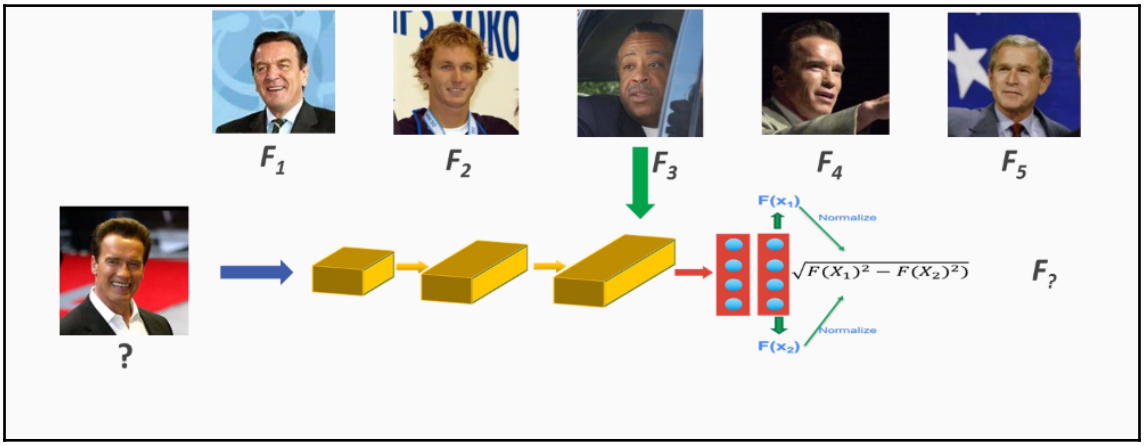
$$Z = sum(W_k * \left(f\left(X_i^1\right) - f(X_i^2) \right) + b)$$

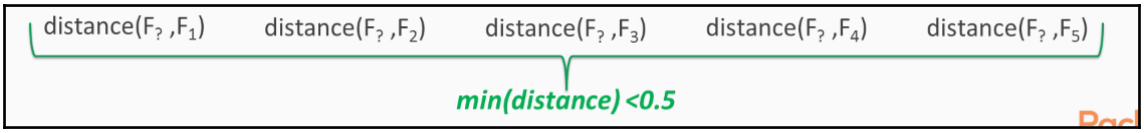
$$Y = sig(Z); 0|1$$

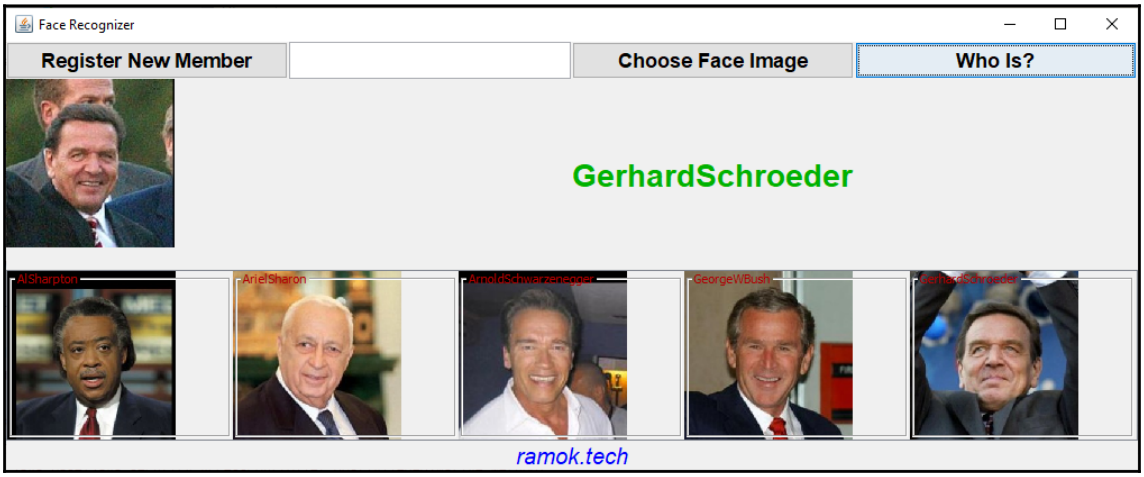
$$Z = sum(W_k * \frac{\left(f(X_i^1) - f(X_i^2) \right)^2}{f(X_i^1) + f(X_i^2)} + b)$$

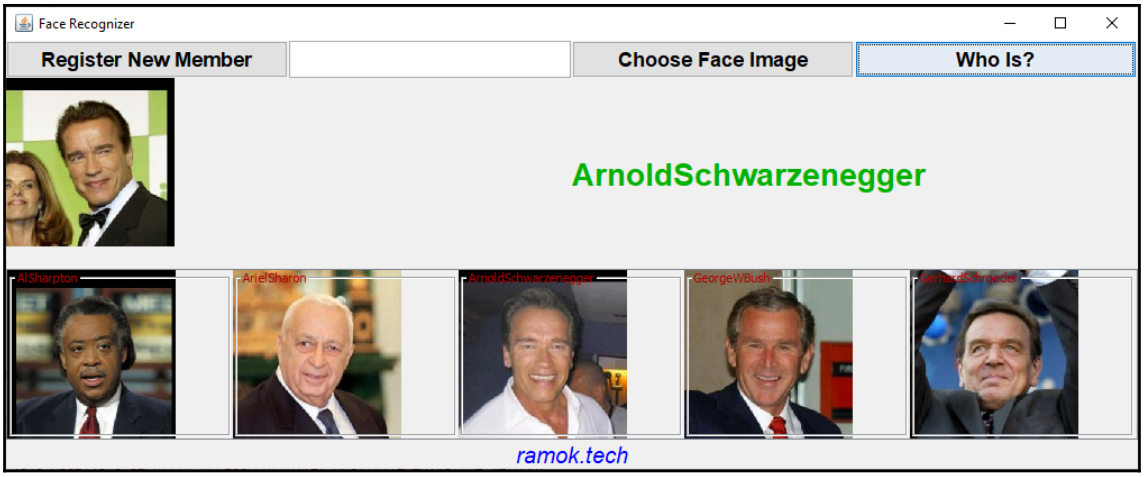


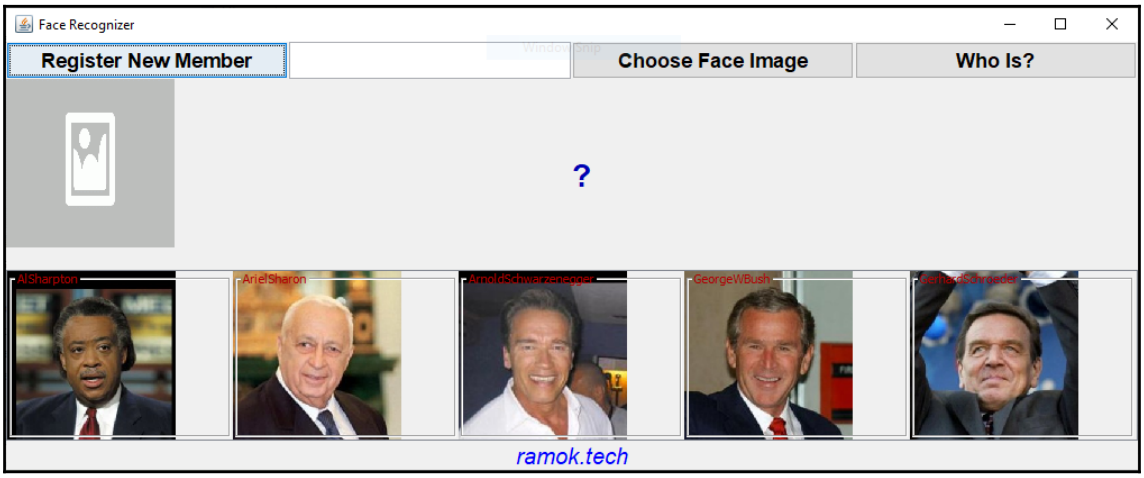



F1
F2
F3
F4
F5
?
F(x1)
Normalize
√F(X1)² − F(X2)²)
F?
F(x2)
Normalize


distance(F? ,F1)
distance(F? ,F2)
distance(F? ,F3)
distance(F? ,F4)
distance(F? ,F5)
min(distance) <0.5


Face Recognizer
Register New Member
Choose Face Image
Who Is?
GerhardSchroeder
AlSharpton
ArielSharon
ArnoldSchwarzenegger
GeorgeWBush
GerhardSchroeder
ramok.tech


Face Recognizer
Register New Member
Choose Face Image
Who Is?
ArnoldSchwarzenegger
AlSharpton
ArielSharon
ArnoldSchwarzenegger
GeorgeWBush
GerhardSchroeder
ramok.tech


Face Recognizer
Register New Member
Choose Face Image
Who Is?
?
AlSharpton
ArielSharon
ArnoldSchwarzenegger
GeorgeWBush
GerhardSchroeder
ramok.tech