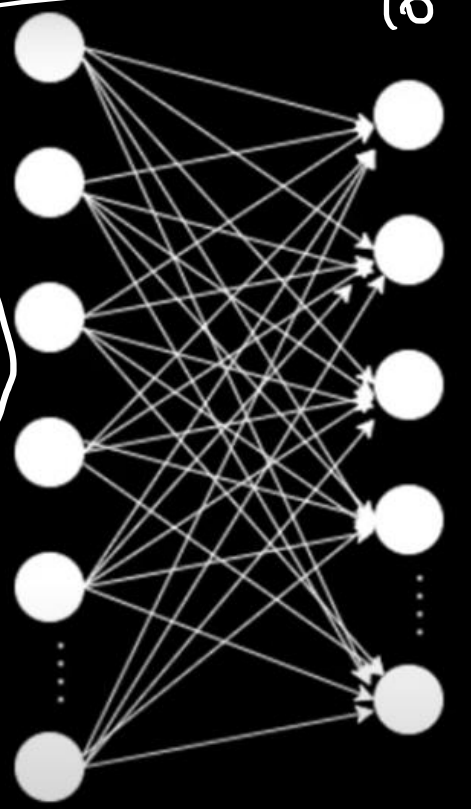
WHY DO WE NEED A CNN FOR IMAGE DATA

$10^3 \times 10^3 \rightarrow 10^6$

ANN is not feasible for image data because, pictures are formed by layers of colours and that creates more neurons, which is comp exp

3×10^6



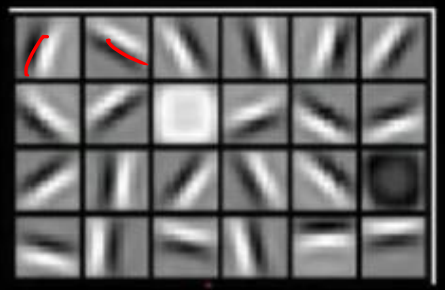
Weights = $3 \times 10^6 \times 10^3 \Rightarrow 3 \times 10^9$

EDGE FILTERS

{ Image made up of Pixel

Vertical Edge Filter

Horizontal Edge Filter



Human

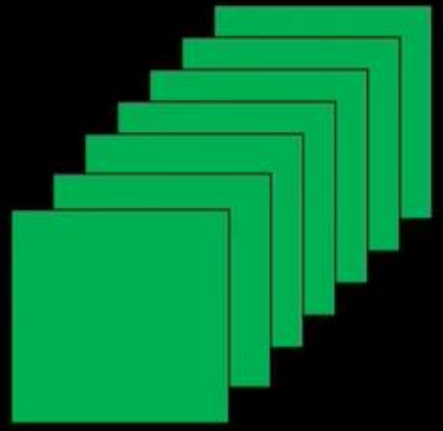
horizontal

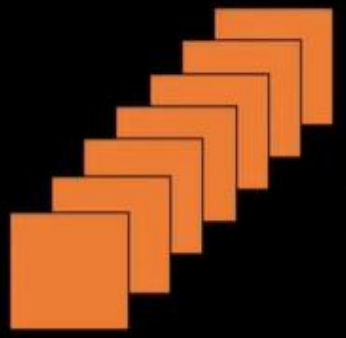
parity

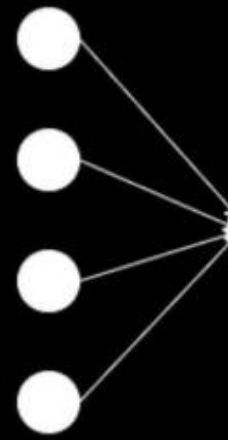
circular

vertical edge filter

Square image → vertical, horizontal Edge → eyes, nose, eyebrow... → human male







Human

Grey Scale Image

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

{6 x 6 pixel}

⇒ {vertical edge filter}

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

{3 x 3 pixel}

* =

-8	-9	-2	14
6	-3	-13	9
4	-4	-8	5
2	2	1	-3

4 x 4

(6 + 0 − 10) + (5 + 0 − 8) + (7 + 0 − 9)

⇒ −4 −3 −2 ⇒ −9

$$\Rightarrow (n \times n) * (f \times f) = (n - f + 1) \times (n - f + 1)$$

$(6 \times 6) * (3 \times 3) \Rightarrow (6 - 3 + 1) \times (6 - 3 + 1) \Rightarrow (4) \times (4)$

Input data

{ Grey scale image } × { multiple filter (horizontal, vertical edge filter) } ⇒ { output image with layers & filter }

{ Grey scale image } → $(n \times n) \times 1$ → layer

{ multiple filter } → $(f \times f) \times c$ → filter

{ output image with layers & filter } → $(n-f+1) \times (n-f+1) \times c$ → # filters

$\times$ { Grey scale image — 1 layer
Colour image — 3 layer (RGB) }

Multiple output layer

1 time

6 times

{6x6 px}

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

*

{Vertical edge filter}

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

{3x3 px}

=

0	3	3	0
0	3	3	0
0	3	3	0
0	3	3	0

{4x4 px}

Grey scaled image

*

{ vertical edge }

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

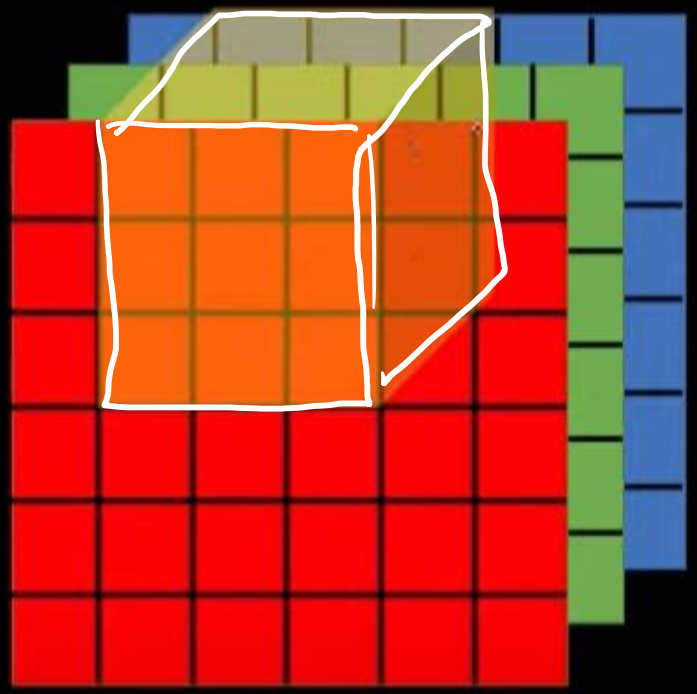
=

*

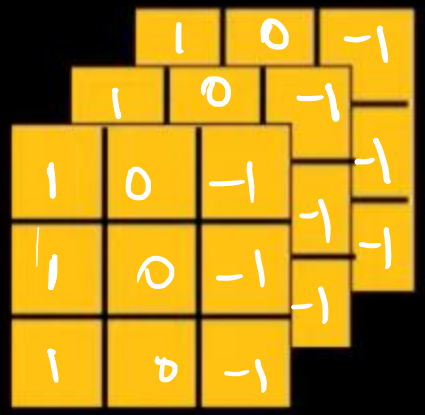
{ horizontal edge }

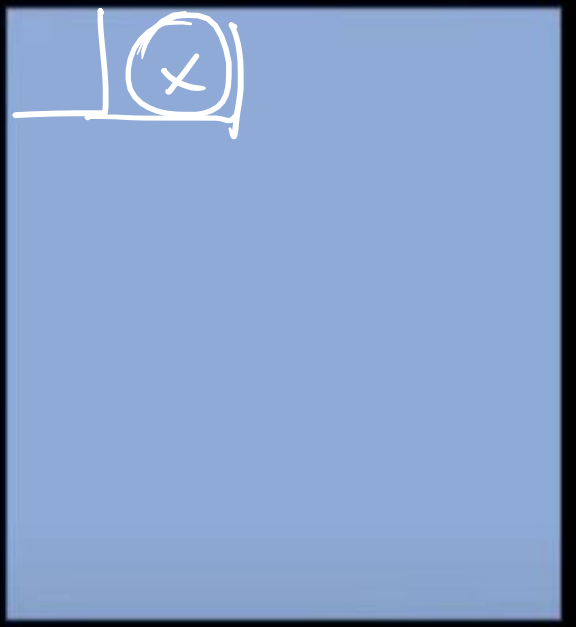
1	1	1
0	0	0
-1	-1	-1

=



(layer1) + (layer2) + (layer3)





$n \times n \times 3$

↓ layer (RGB)

$f \times f \times 3$

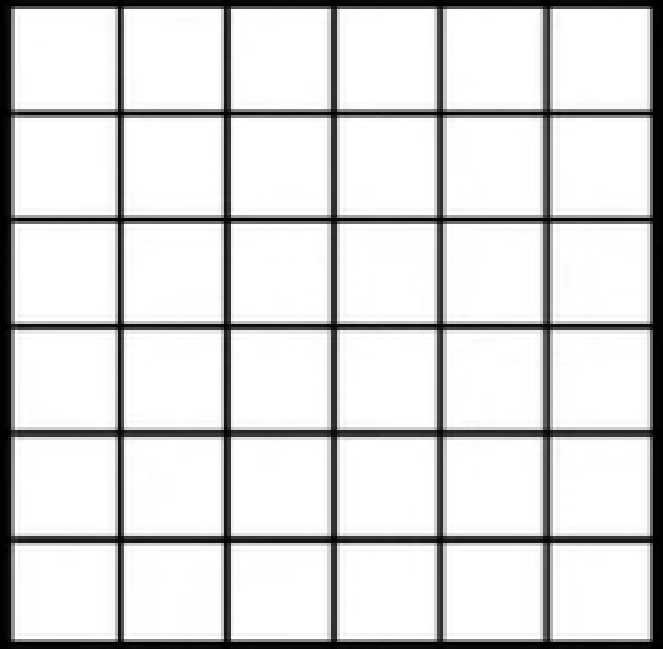
↓ filter (vertical)

$(n-f+1) \times (n-f+1) \times 1$


(nxn x3)
*
vertical edge filter
fxfx3
horizontal edge filter
fxfx3
=
vertical edge filter output
elliptical edge filter
fxfx3

PADDING

{ padding help to retain the output pixel size w.r.t input pixel } ⇒ { also it helps to analyse edge pixels more # of times }



⟹ 1-px

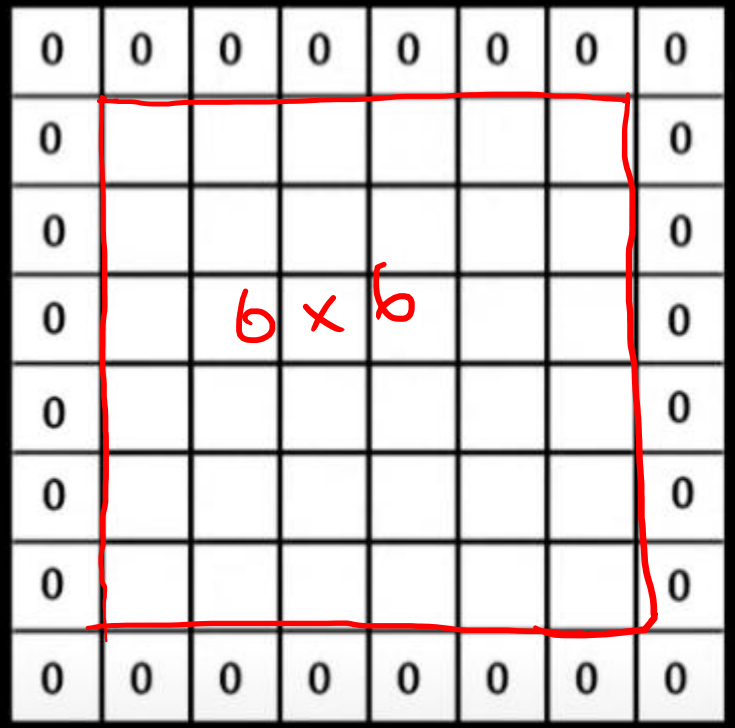
0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

{6x6}

{8x8}

PADDING

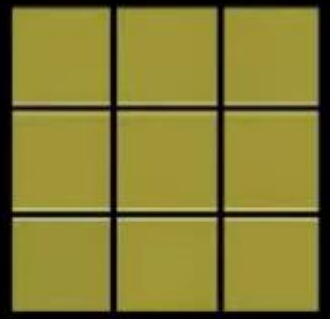
padded image



{8 x 8}

*

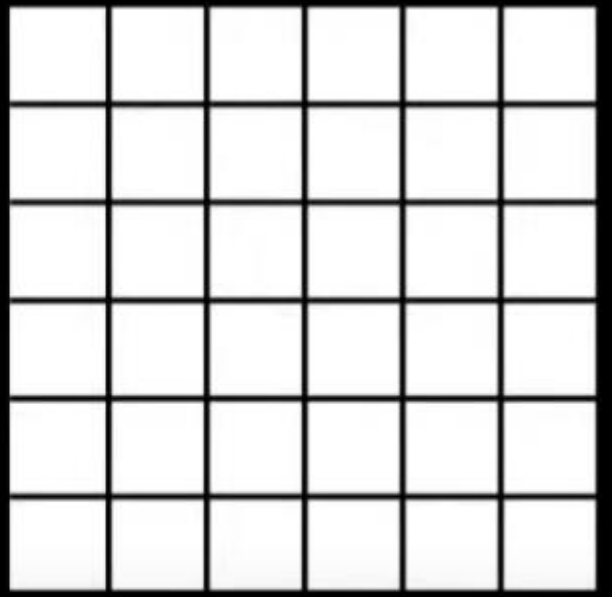
filter



{3 x 3}

=

output image



{6 x 6}

$(n \times n) \times (f \times f) = \{n-f+1\} \times \{n-f+1\}$

$(8 \times 8) \times (3 \times 3) = \{8-3+1\} \times \{8-3+1\}$

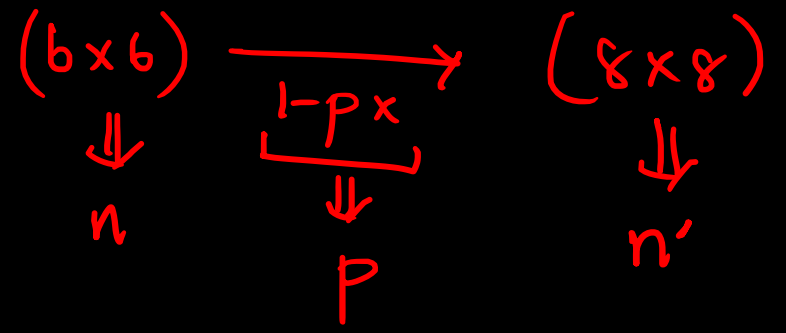
$\Rightarrow 6 \times 6 \Rightarrow$ original image size

PADDING

Types:

1. Valid convolution $\Rightarrow$ No padding.

2. Same Convolution $\Rightarrow$ $n' = n + 2p$



$\Rightarrow (n' - f + 1) = n$

$\Rightarrow (n + 2p - f + 1) = n$

$\Rightarrow 2p = n - (n - f + 1)$

$\Rightarrow 2p = n - n + f - 1$

$\Rightarrow 2p = f - 1 \Rightarrow p = \{(f-1)/2\} \Rightarrow$ Padding size formula

(2) $(n' \times n') \times (f \times f) \Rightarrow \{n' - f + 1\} \times \{n' - f + 1\}$

PADDING

$\{6 \times 6\} \times \{3 \times 3\} \Rightarrow \{n \times n\} \times \{f \times f\}$

$\Rightarrow \{n' - f + 1\} = n$

$\Rightarrow \{n + 2p - f + 1\} = n \Rightarrow \{6 + 2p - 3 + 1\} = 6$

$\Rightarrow 2p = 6 - (6 - 3 + 1)$

$\Rightarrow 2p = 6 - 6 + 3 - 1$

$\Rightarrow 2p = 2$

$\Rightarrow p = 2/2 \Rightarrow p = 1$

$n' = n + 2p$

$n' = 6 + 2(1)$

$n' \Rightarrow 8$

$(n' \times n')$

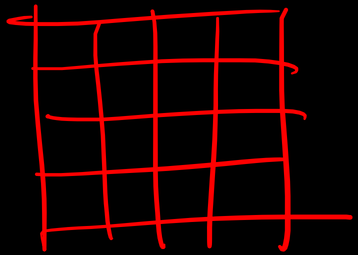
Python:

```
tf.nn.conv2d(x, W1,
             stride=[1,1,1,1],
             padding='VALID')
             (or)
             'SAME'
```

STRIDE

Stride = 1

(6x6) (3x3)
⇒ (4x4)



1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

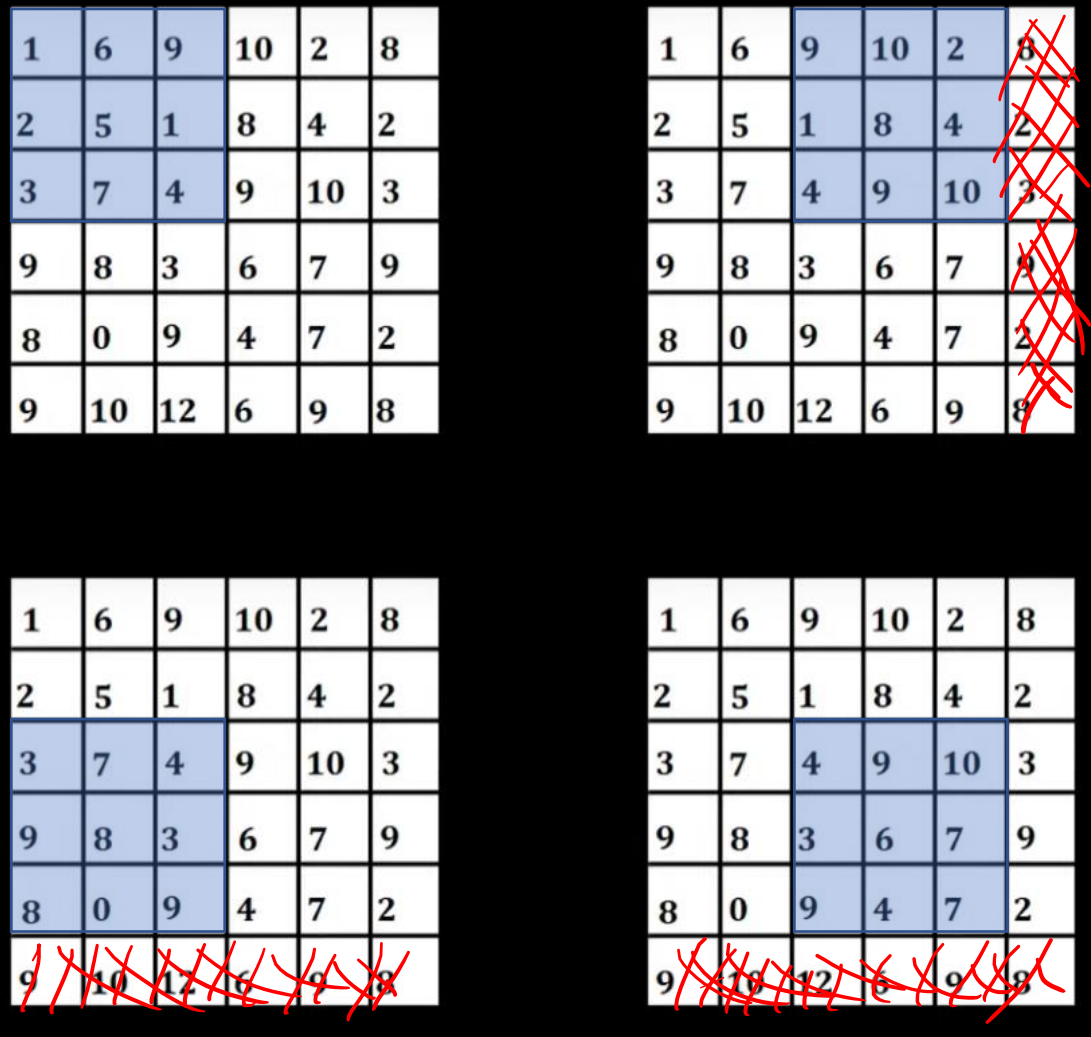
1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

STRIDE

Stride = 2



$$\text{floor}\left[\frac{n-f}{s}+1\right]$$

$s \rightarrow$ stride

$$\left[\frac{6-3}{2}+1\right] \times \left[\frac{7-3}{2}+1\right]$$

$$\Rightarrow [2.5] \times [3]$$

$$\Rightarrow 2 \times 3 \Rightarrow 6$$

This type of convolution is called strided convolution