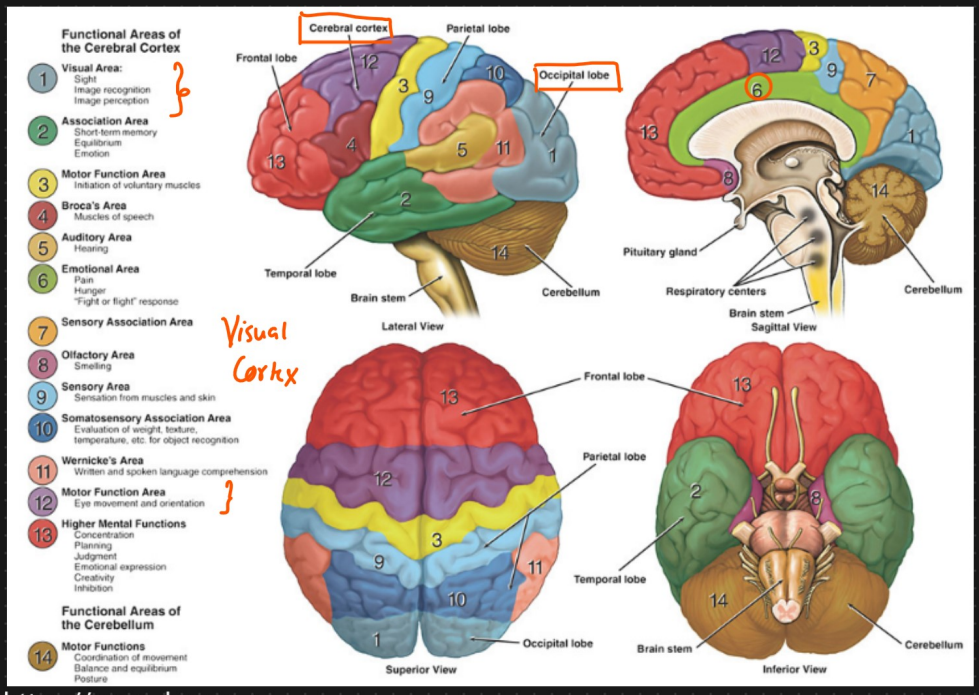
Convolutional Neural N/w



https://www.dana.org

① ANN → Supervised Learning → [Classification, Regression]

Dataset : I/p features O/p

② CNN : I/p => Images Eg: Image classification, Object Detection, Segmentation

② Cerebral Cortex And Visual Cortex

Visual Cortex (V1-V5) [Region of the brain that receives, integrates and processes visual information relayed from the retinas].

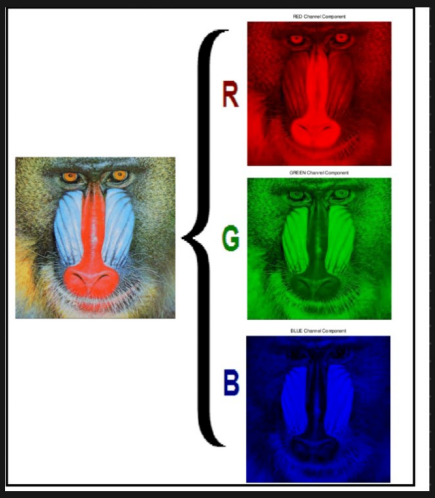
→ V1 → Primary Visual Cortex [Orientation of edges And Lines]

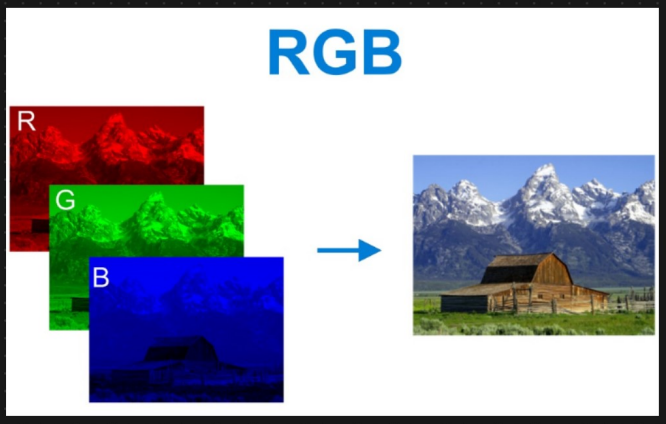
V2 [Differences in color, complex patterns, Object orientation]

V3 V4 V5 [Object Recognition]

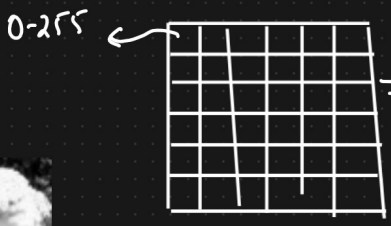
Visualize the Image

③ RGB Images And Gray Scale Images

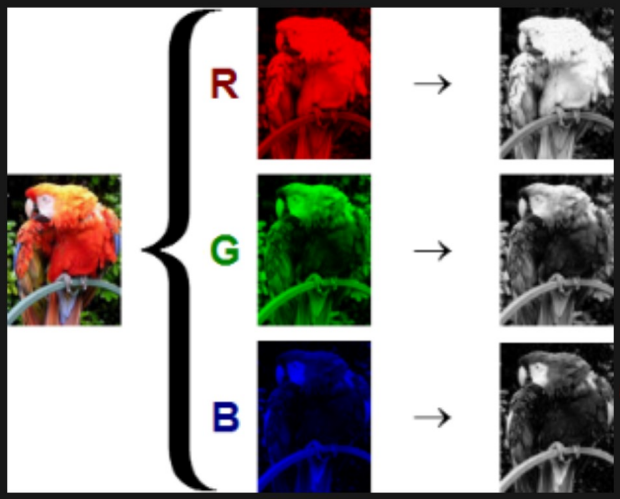




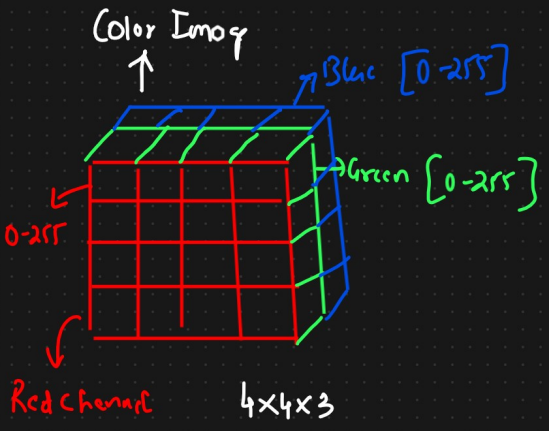
https://www.researchgate.net/



6x6x1



https://commons.wikimedia.org/



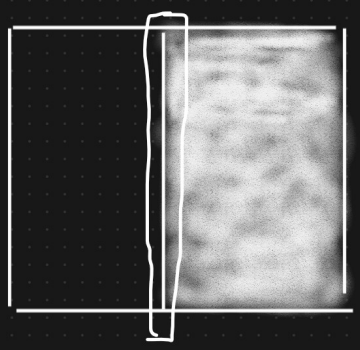
4x4x3

④ Convolution Operation In CNN

→ (0,1)

0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

6x6x1



Step 1

(1) Normalize

Divide by 255

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

Convolution operation

Stride=1 $n=6$ $f=3$ $o/p=4$

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	1	1	1

$6 \times 6 \times 1$

$*$

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

3×3 filters

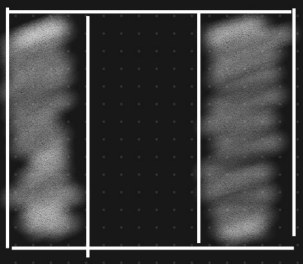
Vertical edge filters

$=$

0	-4	-4	0
0	-4	-4	0
0	-4	-4	0
0	-4	-4	0

4×4

$n - f + 1 = 6 - 3 + 1 = 4$



$\Leftarrow$

255	0	0	255
255	0	0	255
255	0	0	255
255	0	0	255

⑤ Padding In CNN

Apply zero in empty padded sections

① Zero padding

② Neighbour padding ⇒ assign neighbours from original section.

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

6x6 $n=6$ 8x8 padding layer

$*$

$f=3$

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

→ original

$n-f+2p+1$

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

$3+2p+1=6$

$2p=6-4$

$p=\frac{2}{2}=1$

$=$

	0	-4	-4	0
	0	-4	-4	0
	0	-4	-4	0
	0	-4	-4	0

6x6

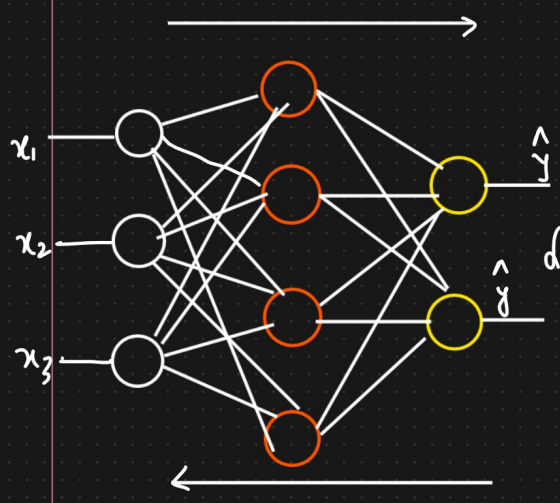
7x7 3x3 7x7

How much padding you need to apply?

$n=7$ $f=3$ $n-f+2p+1=7$ → find size of o/p

$7-3+2p+1=7 \Rightarrow \boxed{p=1}$

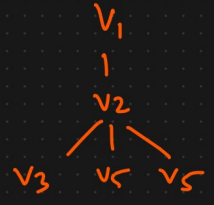
⑥ Operation of CNN Vs ANN



$$z = w^T x_i + b$$

$$relu(z)$$

Visual Cortex



Image

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	1	1	1

*

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

f_1

=

0	-4	-4	0
0	-4	-4	0
0	-4	-4	0
0	-4	-4	0

→ Relu operation $max(0,x)$ ← on each and every cell content

f_2
f_3
f_n

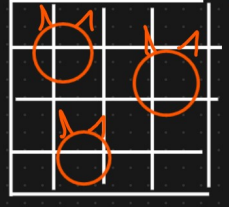
Multiple o/p for multiple filters

no. of o/p at each state = no. of filters applied in prev. state

1 Convolution Layer

ReLU because the operations are easy.

⑦ Max Pooling, Min Pooling, Mean Pooling



⊛

2x2 filter

P=1

S=1

=

Relu → stride jump of 2

1	2	3
4	3	6
2	8	4

3x3

Stride = 2

Max Pooling =

2x2

↓ maximum value from each stride

4	6
8	4

f_2
f_3
f_n

Convolution Layer

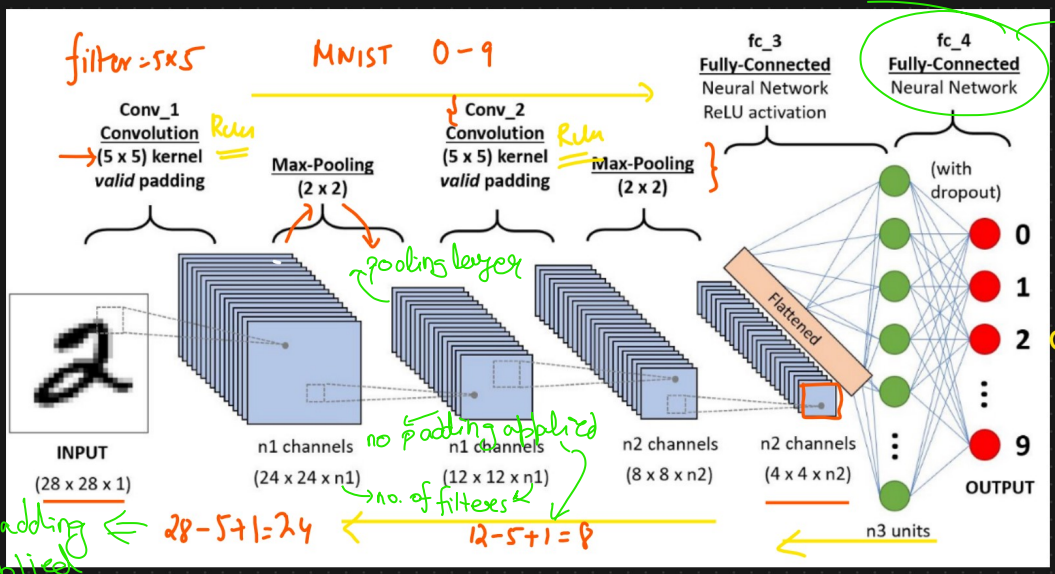
⑦ Location Invariant

Pooling extract required features irrespective of their location in images.

Max pooling is used to collect info about most highlighted features from image

Similar working is of Min & mean pooling but for min & mean. pooling depends upon task.

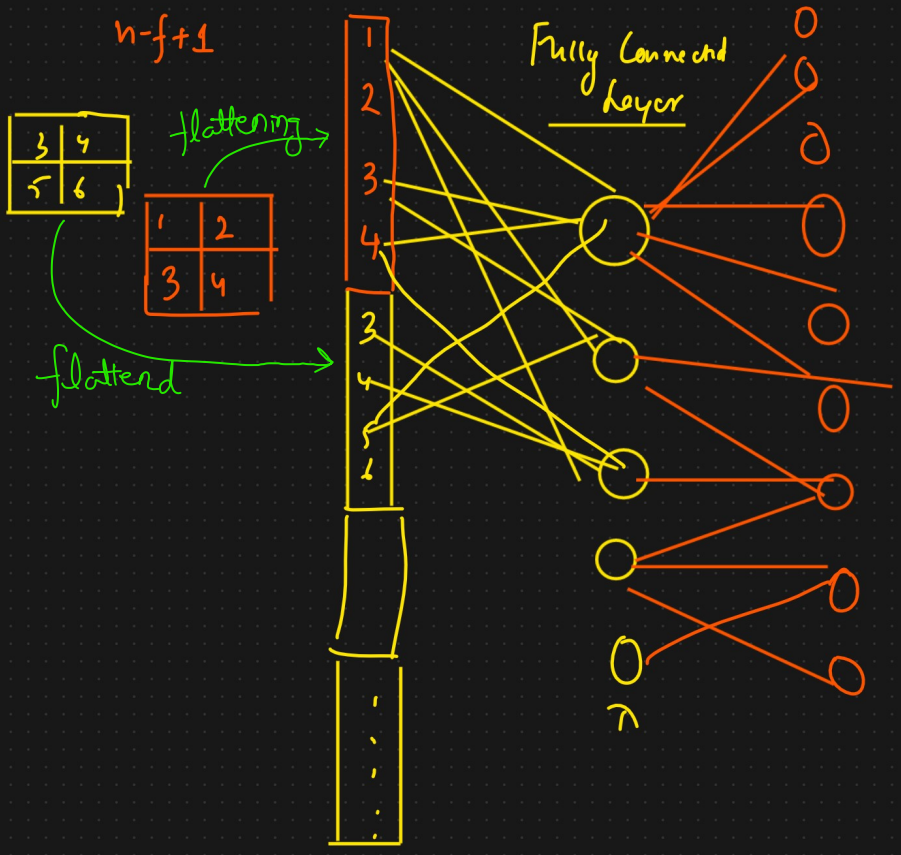
⑧ Fully Connected Layer In CNN [Flattened Layer]



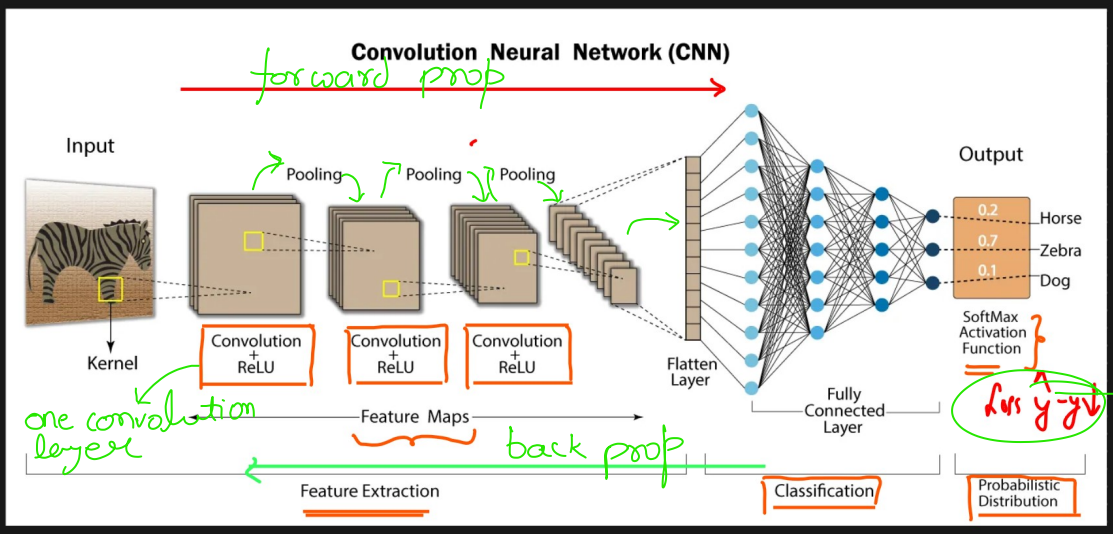
ANN

Loss function ↓↓

Optimizer



⑨ CNN Complete Example



https://developersbreach.com/convolution-neural-network-deep-learning/

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

*

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

0	-4	-4	0
0	-4	-4	0
0	-4	-4	0
0	-4	-4	0