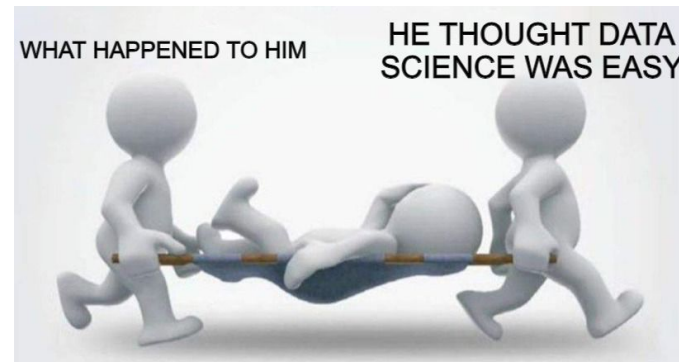


# Deep Convolution Network

M.Tech. Data Science, Second Year, NMIMS

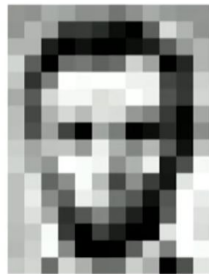
By,

Bilal Hungund, Data Scientist, Halliburton



What Computer see?

→ Images are numbers



187	182	174	168	160	152	126	181	172	163	165	166
188	182	182	74	74	62	93	17	176	270	180	164
188	180	88	14	84	6	10	83	48	156	158	181
204	178	7	114	137	111	140	204	196	16	60	180
194	68	137	261	287	288	288	288	287	67	71	281
172	188	287	288	288	214	220	288	288	74	288	
188	68	178	288	188	218	211	188	188	75	88	188
188	7	188	84	74	188	138	11	81	62	83	188
188	148	181	188	188	287	178	188	188	75	88	188
205	174	188	288	288	287	188	178	288	63	88	284
188	278	138	148	288	287	88	188	75	88	278	281
188	288	147	188	287	278	127	113	88	188	288	211
188	274	178	88	138	88	88	3	287	288	211	
187	188	288	74	7	81	67	6	817	288	211	
188	288	287	188	8	13	188	288	188	288	288	
188	288	188	287	177	188	188	288	188	188	218	

Types of images

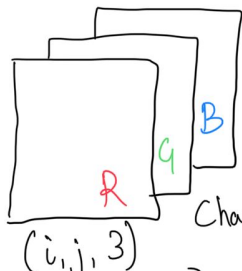
Grayscale

RGB

[Black and White pixels]

#channels

$(i, j, 1)$



Channels

$(i, j, 3)$

Values between (0, 255)

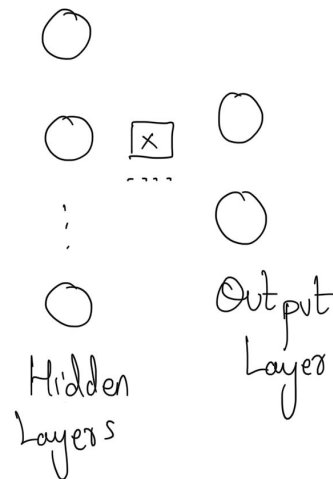
$a_{11}$	$a_{12}$	...	...	$a_{1j}$
$a_{i1}$				$a_{ij}$

$i \times j$

Flatten

$\begin{bmatrix} a_{11} \\ a_{12} \\ \vdots \\ a_{ij} \\ \vdots \\ a_{ij} \end{bmatrix}$

Input Layer  
( $i \times j$  neurons)



Traditional DNN

# Convolution & operation

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

6x6 image

Filter (Weights)

-1	1	-1
2	3	2
1	1	1

3x3 Filter

$$n - f + 1$$

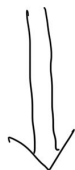
$$= 6 - 3 + 1$$

$$= 4$$

Output Size

4	4	4	4
4	..	..	..
:	:	:	:
:	:	:	:

4x4



Output without padding

Stride = 1  
(step size)

With Padding

$$n - 2p - f + 1$$

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	1	1	1

4x4 → 6x6  
padding

Padding

Filter

1	-1
1	-1




4x4

# Pooling

25	48	11	58
192	10	20	110
38	0	9	31
50	8	23	47

Stride = 2  
(Recommended  
for Pooling)

25	48
192	10

11	58
20	110

38	0
50	8

9	31
23	47

Pooling  
⇒




Max Pooling

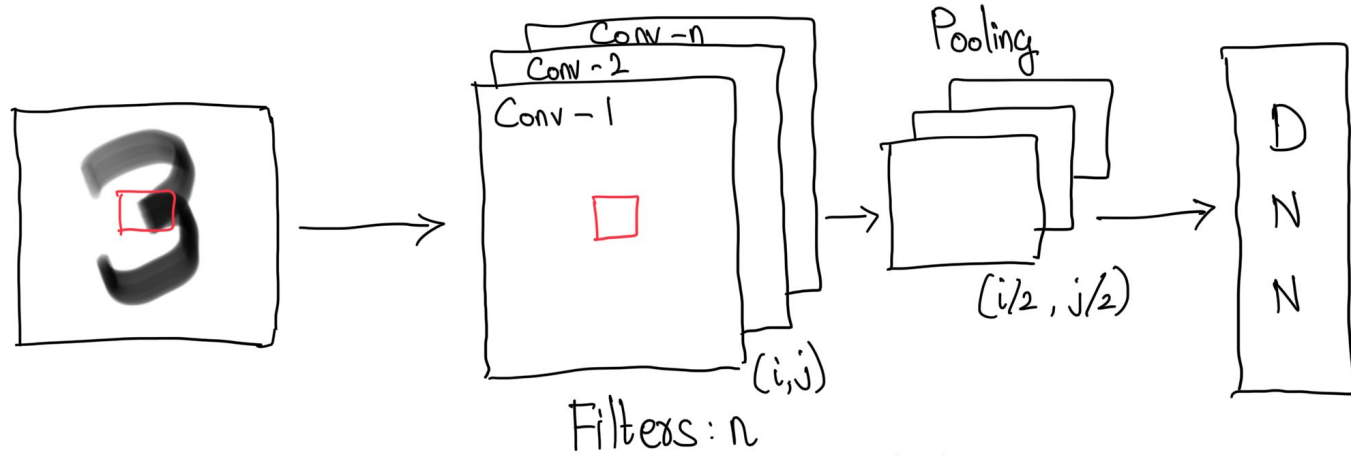
192	110
50	47

69	50
22	28

Average Pooling

# Convolution Neural Network (CNN) for Classification

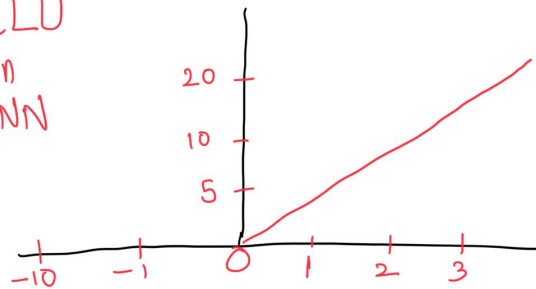
 `tf.keras.layers.Conv2D`  
 `tf.keras.activations.*`  
 `tf.keras.layers.MaxPool2D`



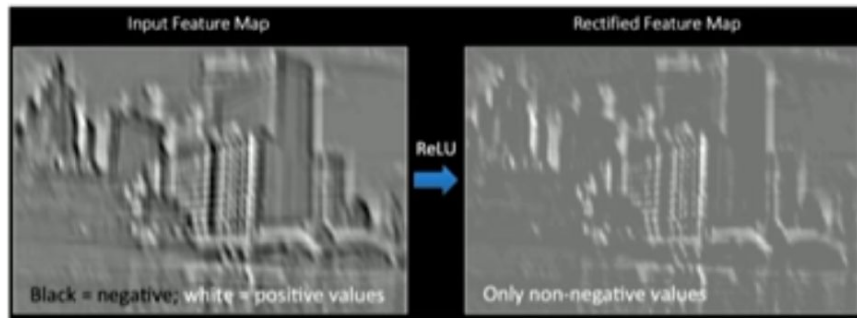
- 1) Convolution: Filters to generate feature maps
- 2) Non-linearity: often relu
- 3) Backpropagation
- 4) Pooling: Downsampling feature maps

# Non-Linearity

ReLU  
in  
CNN



$$g(z) = \max(0, z)$$



# Application of CNN

- Classification
- Object Detection
- Segmentation