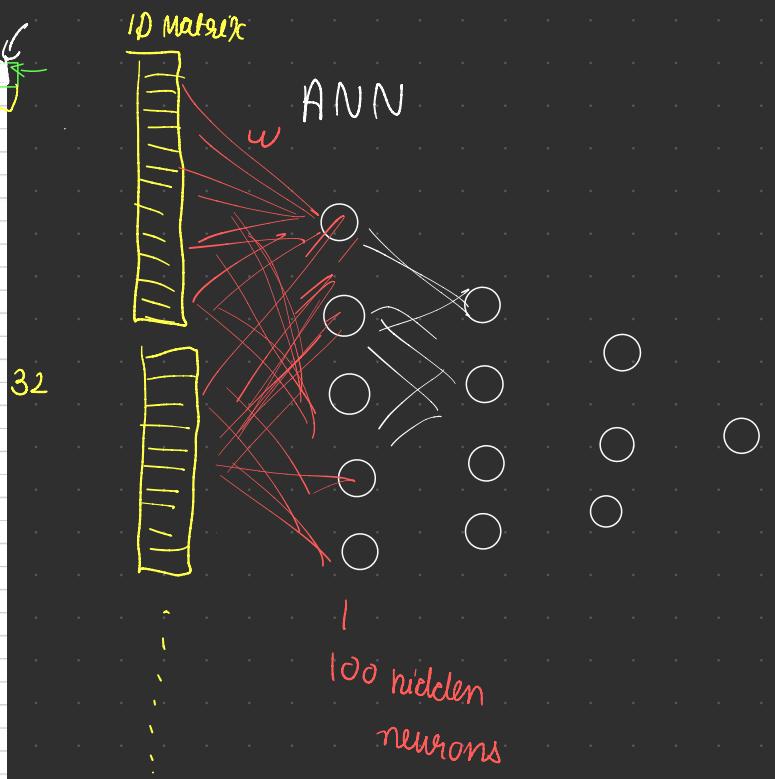


# **convolutional neural network (CNN)**

CNN is used for images and we will first see why we cannot use ANN for images



$$32 \times 32 = \boxed{1024} - \text{inputs}$$

$$\begin{aligned} 1024 \times 100 \\ = 102400 \end{aligned}$$

## Problems

- ① high computational power.
- ② Spatial Arrangement is missing
- ③ Overfitting

## What actually is CNN ?



If I ask you “What’s in the picture?”, you don’t try to look at all the LEGO pieces at once.

Instead, you:

First look at small areas maybe a window, a tower, or a door.

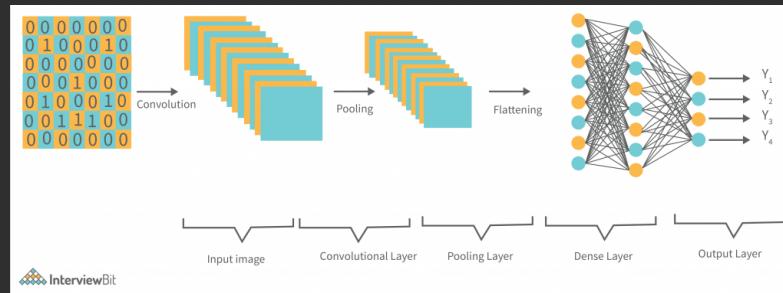
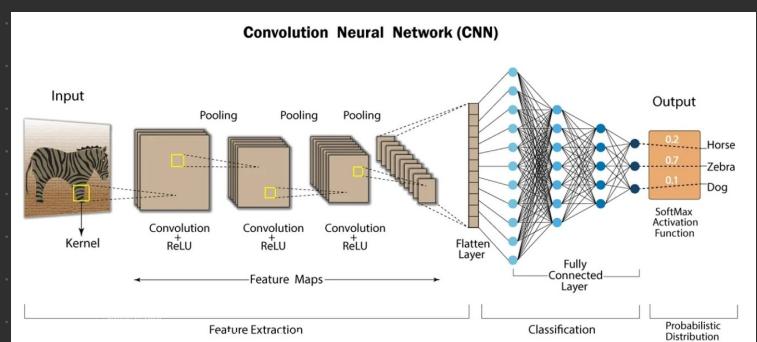
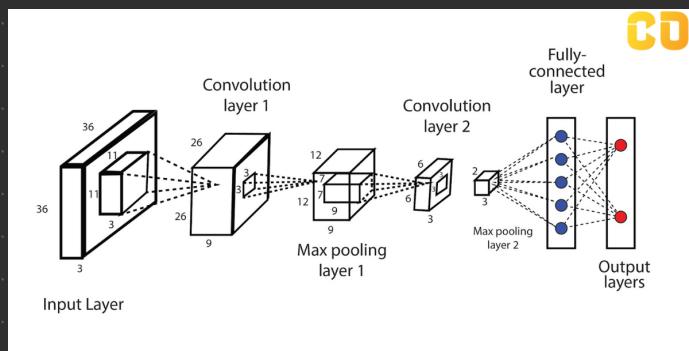
Then you put those pieces together in your mind:

“Oh, I see windows!”

“Oh, that looks like a tower!”

Finally, you say: “Yes, this is a castle!”

That’s exactly how Convolutional Neural Networks (CNNs) work. They don’t try to understand the whole image at once they look at small patches and slowly build up the full picture.



### 1. Convolution Layer The Detective’s Magnifying Glass

This layer slides a small filter (like a 3x3 window) across the picture.

The filter might highlight edges, colors, or textures.

Example: One filter learns to find “straight lines,” another learns “round shapes.”

It’s like a detective scanning for clues in tiny parts of the picture.

### 2. Pooling Layer The Shortcut

Once the detective finds details, we don’t need to keep every tiny pixel.

Pooling just keeps the most important information.

Think of it like zooming out a little you lose small details, but still know the big shape.

### 3. More Convolution + Pooling Layers Building Understanding

As the network goes deeper:

First layers find edges (lines, curves).

Middle layers find shapes (eyes, noses, wheels).

Deeper layers find objects (a cat, a car, a castle).

### 4. Fully Connected Layer – The Decision Maker

At the end, the CNN flattens everything it learned and connects it like a normal ANN.

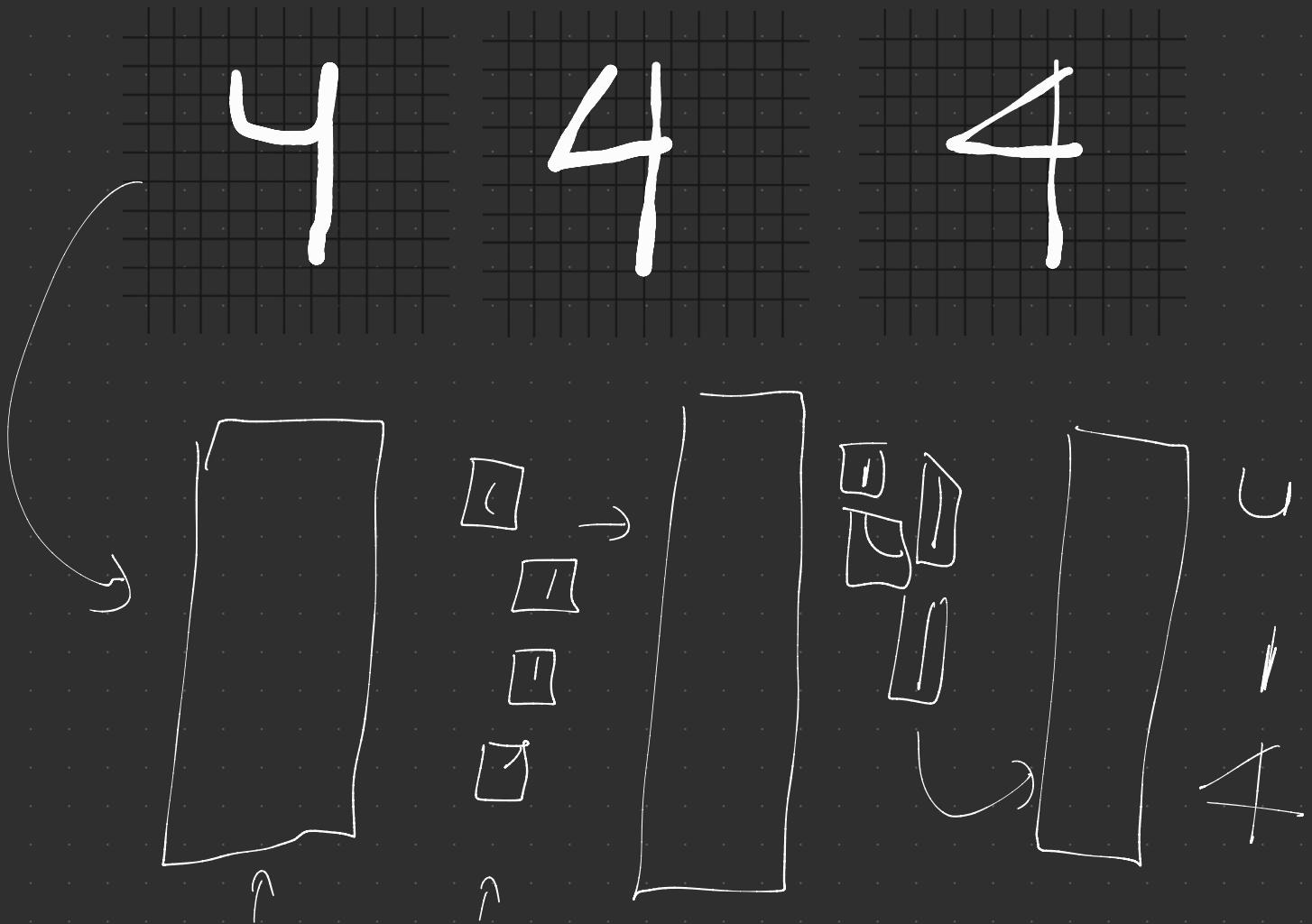
Here it says:

“70% chance it’s a cat”

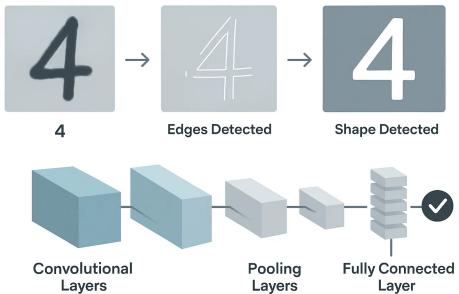
“20% chance it’s a dog”

“10% chance it’s a rabbit”

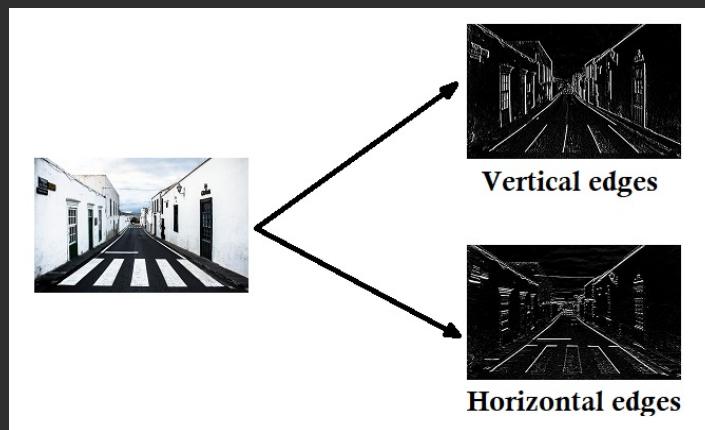
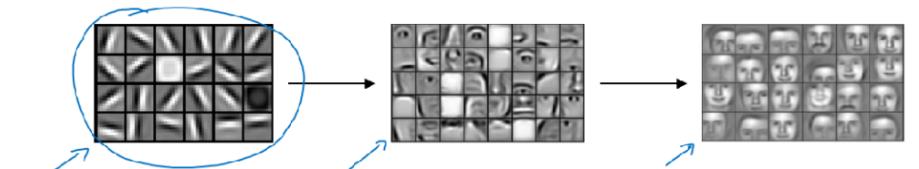
Whichever is highest, that’s the answer!



## Layer 1 Convolutional Layer



## Computer Vision Problem



## Finding the edge with maths



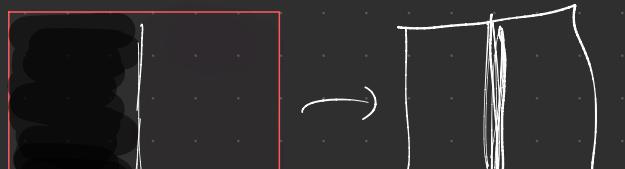
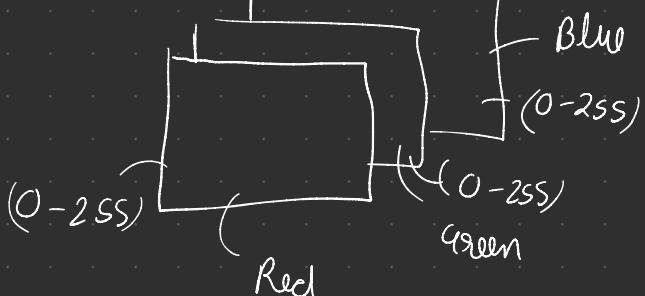
0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	21	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0
0	0	5	5	0	0	0	0	0	0	14	1	0	6	6	0

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	21	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0
0	0	5	5	0	0	0	0	0	0	14	1	0	6	6	0

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

Grayscale  
↓  
(0 - 255)

RGB image  
↓



horizontal

-1	-1	-1
0	0	0
1	1	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

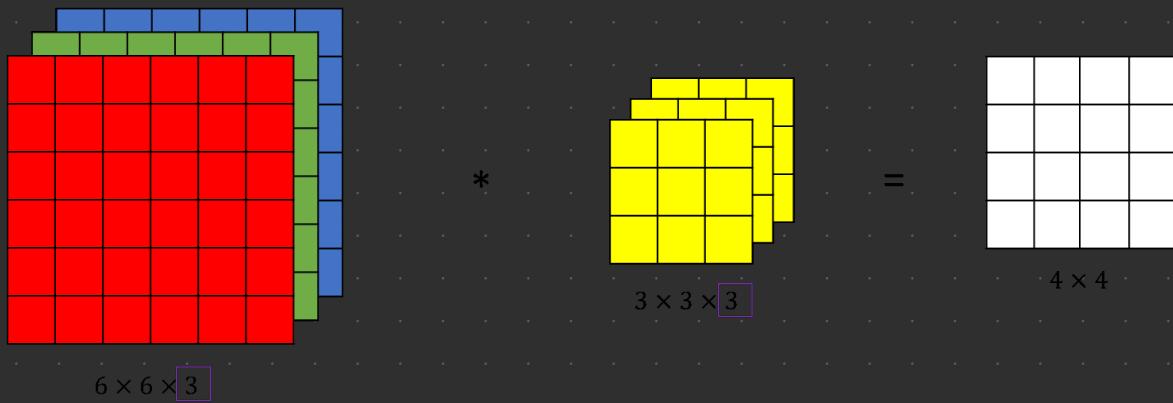
filter

$$(-1 \times 0 + 0 \times 0 + 1 \times 0 +$$

$$-1 \times 0 - - - - )$$

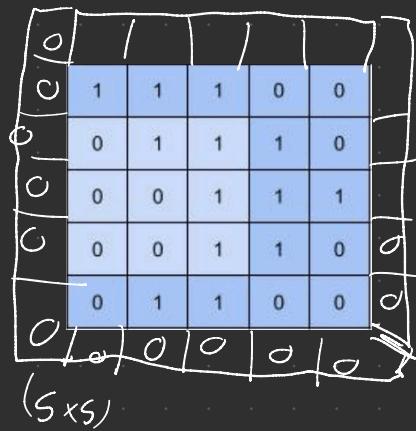
filter

$$28 \times 28 \times 3 \times 3 = (28 \times 28) \times (3 \times 3)$$

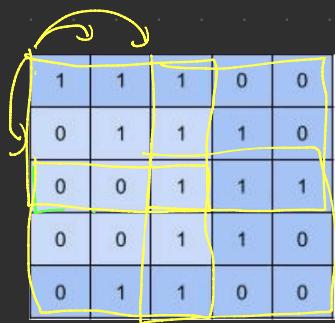


### Padding and Strides

$$\begin{array}{c}
 \boxed{(5 \times 5)} \\
 n \times n \\
 \downarrow \\
 (7 \times 7) * (3 \times 3) = (5 \times 5)
 \end{array}
 \quad
 \begin{array}{l}
 * (3 \times 3) = (3 \times 3) \\
 m \times m \quad (n-m+1) \times (n-m+1) \\
 (n-m+1) = n \\
 n - 3 + 1 = 5 \\
 n = 7
 \end{array}$$



$\rightarrow$  Strides <



$$\begin{array}{c}
 * \\
 \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array}
 \end{array}$$

(5x5)

$$(3 \times 3) = (3 \times 3)$$

(1,1)

(2,2)

$$\begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}$$

(2x2)

1) Major computational issues (1)

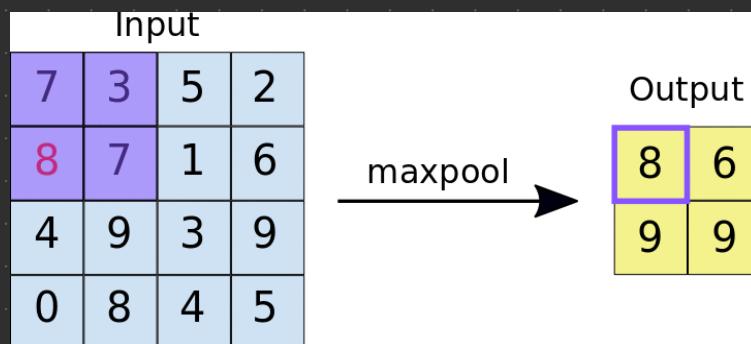
2) low level features

## Pooling

1) Max Pooling

2) Avg Pooling

3) min Pooling



Size -  $(2,2)$  , Stride - 2 ) type  $\Rightarrow$  Max Pooling

- 1) Dominant features are captured ( Max Pooling )
- 2) Size issue ✓
- 3) location Problem ,

### Why Pooling is Used

Reduce dimensions  $\rightarrow$  smaller feature maps, faster computation.

Extract dominant features  $\rightarrow$  keep most important info, ignore noise.

Make model translation-invariant  $\rightarrow$  position of object doesn't matter much.

## Types of Pooling and Where to Use Use Pooling

### Max Pooling

Takes the maximum value from the region.

Use when: detecting strong/dominant features like edges, textures, object parts.

Most commonly used in CNNs.

### Average Pooling

Takes the average of values in the region.

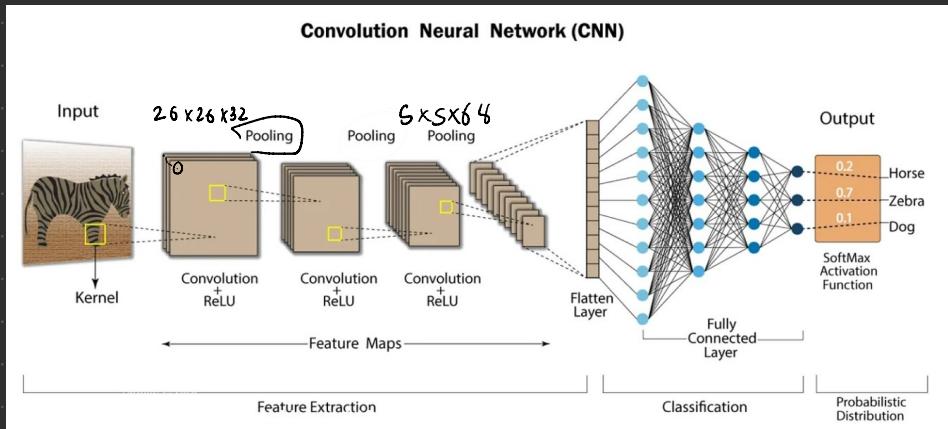
Use when: you want smoother, less noisy representations, e.g., in early layers or for reducing sensitivity.

Less common today, but good for preserving background info.

### Min Pooling

Takes the minimum value from the region.

Use when: highlighting dark features, anomalies, or background suppression (rare in practice).



Input  $\rightarrow 28 \times 28 \times 1$  — step - 1

Step - 2  $\rightarrow (3 \times 3)$   $\rightarrow$  32 filter

$26 \times 26 \times 32$

Step - 3 - Pooling  $-(2 \times 2)$

$13 \times 13 \times 32$

Step - 4 - 2nd conv layer - 64 filters  $-(3 \times 3)$

$11 \times 11 \times 64$  - features

Step 5  $\rightarrow$  Pooling  $(2 \times 2)$

$5 \times 5 \times 64$

Step - 6  $\rightarrow$  flattening  $\rightarrow$  10 vector

$5 \times 5 \times 64 = 1600$  input

