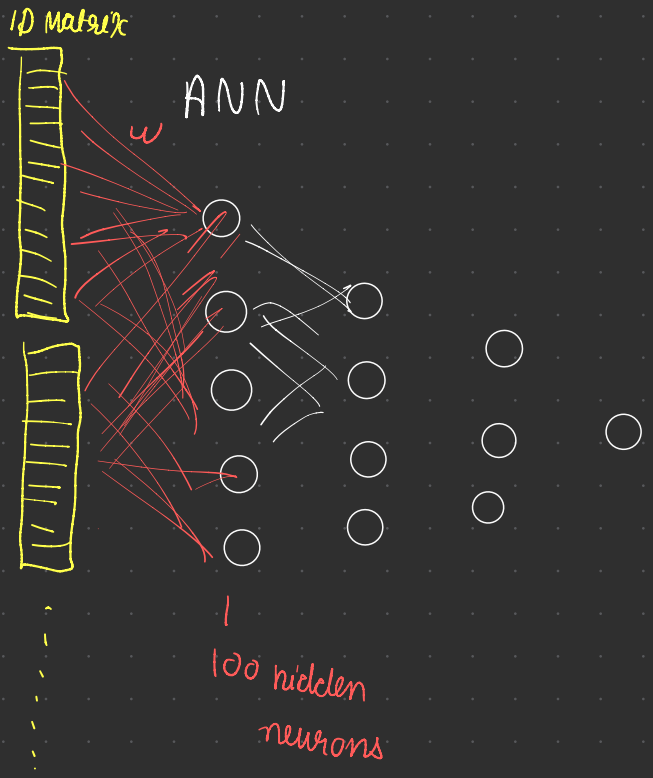
# convolutional neural network (CNN)

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


$32 \times 32 =$ 1024 – inputs

$1024 \times 100$

$= 102400$

## 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.

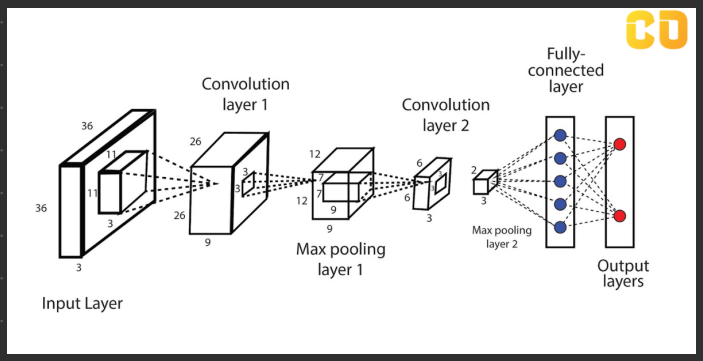
Then you put those pieces together in your mind:

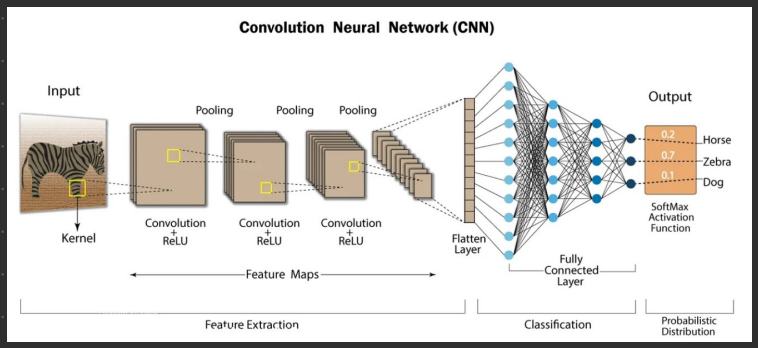
"Oh, I see windows!"

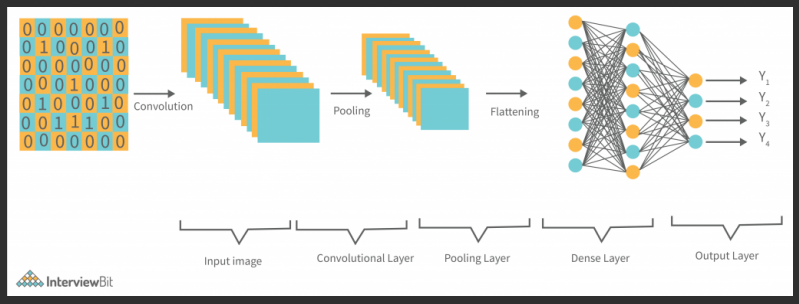
"Oh, that looks like a tower!"

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 3×3 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.**

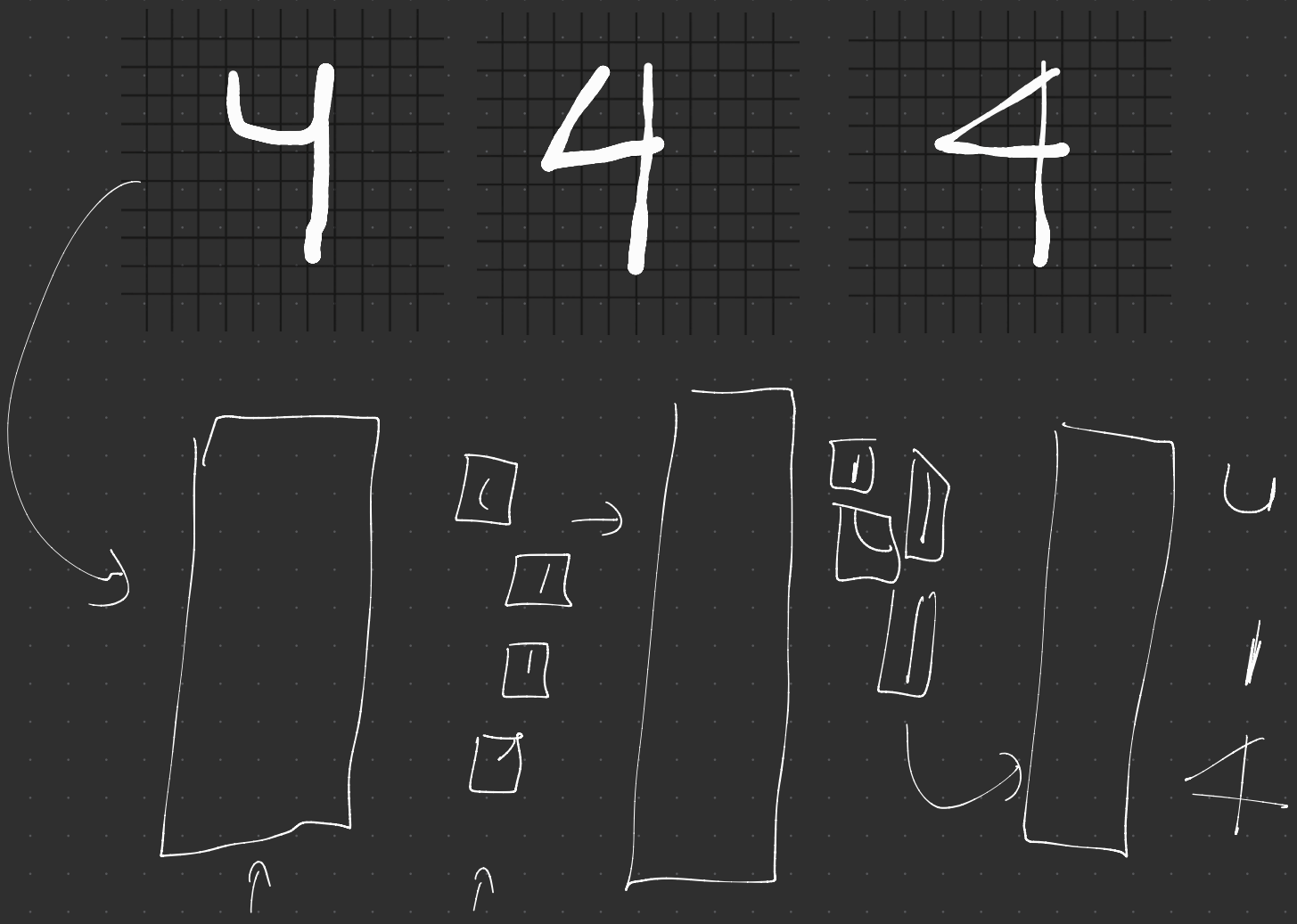
**Here it says:**

**"70% chance it's a cat "**

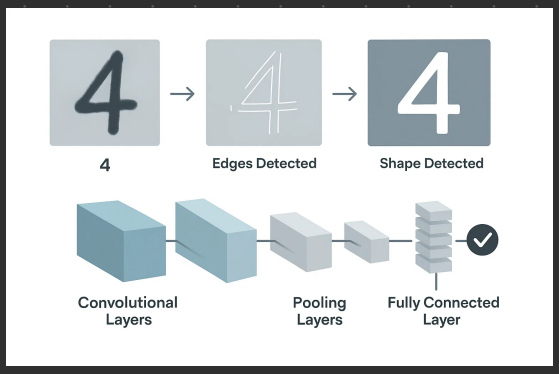
**"20% chance it's a dog "**

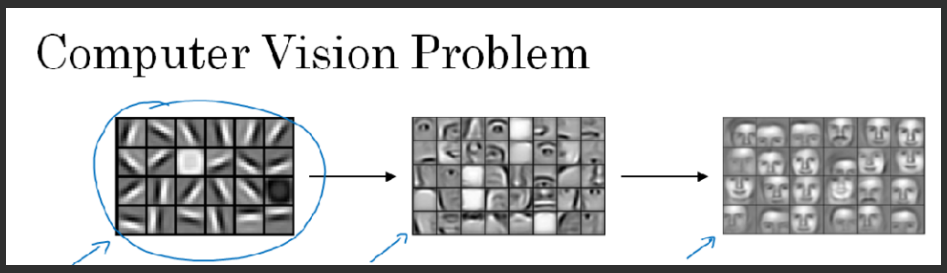
**"10% chance it's a rabbit "**

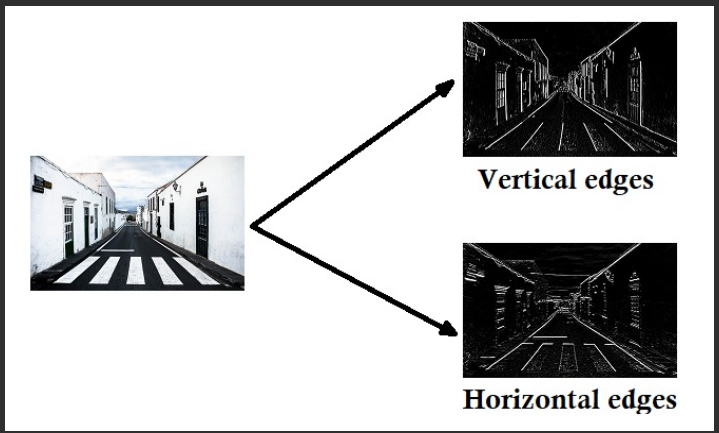
**Whichever is highest, that's the answer!**



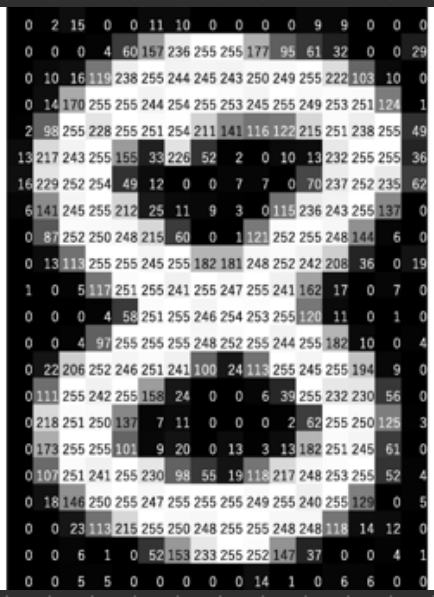
# Layer 1 Convolutional Layer

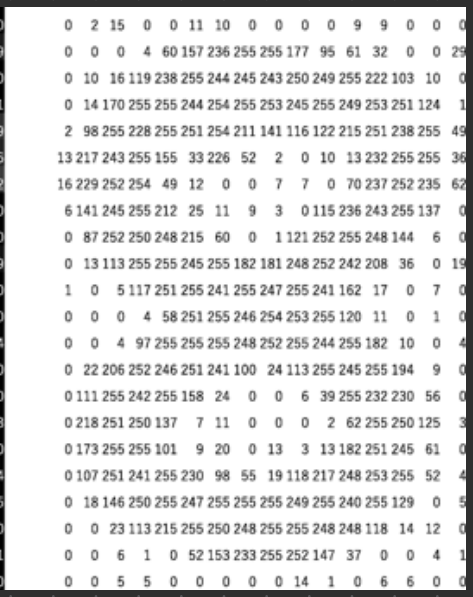


# Finding the edge with maths



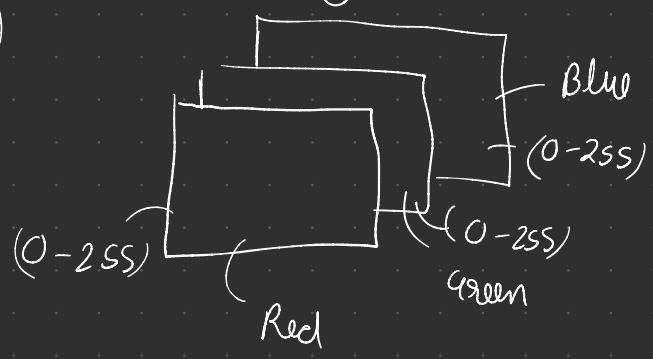


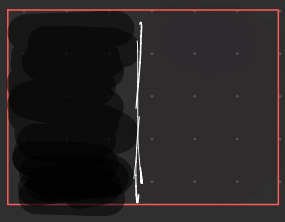
| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |

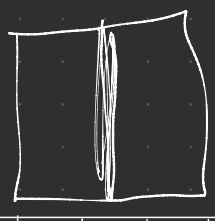
Grayscale

(0 – 255)

RGB image





6

| | | | | | |
|---|---|---|---|---|---|
| 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 |

6 × 6

horizontal

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

filter

3 × 3

Vertical

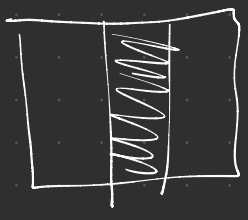
| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

filter

4 × 4

4

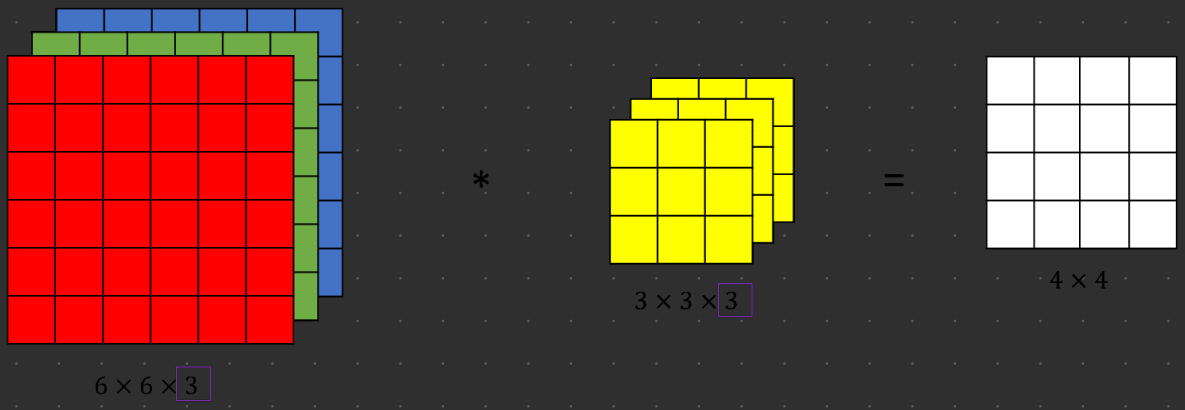
| | | | |
|---|---|---|---|
| 0 | 255 | 255 | 0 |
| 0 | 255 | 255 | 0 |
| 0 | 255 | 255 | 0 |
| 0 | 255 | 255 | 0 |



(-1x0 + 0x0 + 1x0 +

-1x0 - - - - - )

28 × 28 × 3 × 3 = (26 × 26)




## Padding and Strides

$(5 \times 5)$ $*$ $(3 \times 3)$ $= (3 \times 3)$

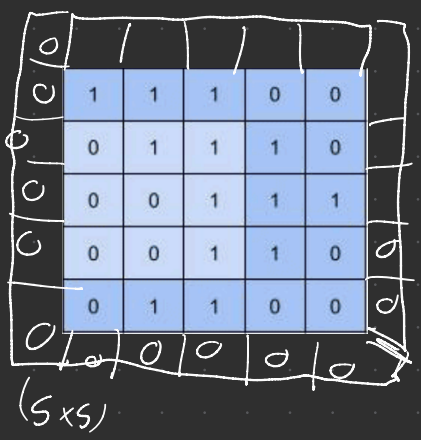
$n \times n$ $\quad m \times m \quad (n-m+1) \times (n-m+1)$

$(n - m + 1) = n$

$n - 3 + 1 = 5$

$n = 7$

$(7 \times 7) * (3 \times 3) = (5 \times 5)$




→ Strides ←


$(5 \times 5)$

$*$

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

$(1,1)$

$(2,2)$ $\Rightarrow$

| | |
|---|---|
| 1 | 0 |
| 0 | 1 |

$(2 \times 2)$

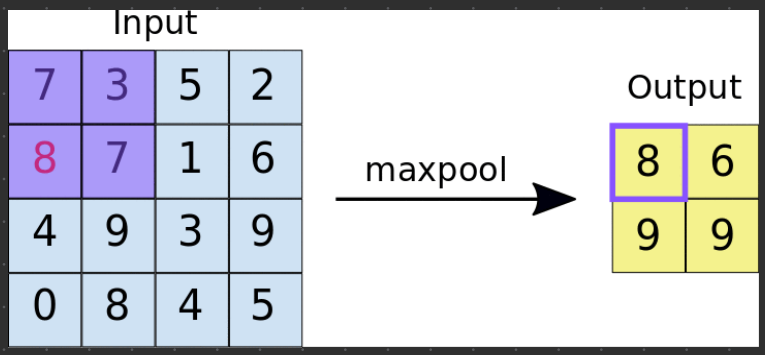
$(3 \times 3) = (3 \times 3)$

1) Major computational issus (1)

2) low level features

# Pooling

1) Max Pooling  2) Avg Pooling  3) min Pooling




Size - (2,2) , Stride - 2 , type ⇒ Max Pooling

1) Dominant features are captured (Max Pooling)

2) Size issue

3) location Problem.

**Why Pooling is Used**

**Reduce dimensions → smaller feature maps, faster computation.**

**Extract dominant features → keep most important info, ignore noise.**

**Make model translation-invariant → 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.**

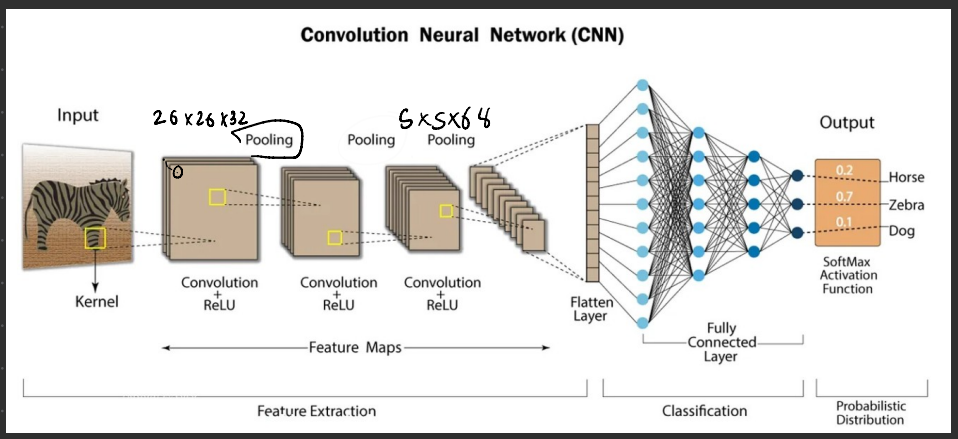
**Use when: you want smoother, less noisy representations, e.g., in early layers or for reducing sensitivity.**

**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 → 28 x 28 x 1 — step - 1

Step - 2 → (3 x 3) → 32 filter

26 x 26 x 32

Step - 3 - Pooling - (2 x 2)

13 x 13 x 32

Step - 4 - 2nd conv layer — 64 filters — (3 x 3)

11 x 11 x 64 - features

Step 5 → Pooling (2 x 2)

5 x 5 x 64

Step - 6 → flattining → 1D vector

5 x 5 x 64 = 1600 input