Convolutional Neural Networks (CNN)

1)What is a CNN?

A **Convolutional Neural Network (CNN)** is a special type of **neural network designed for images**.

It automatically learns **important features** (like edges, corners, and shapes) from images — without us needing to program them manually.

Think of it like this:

A CNN looks at an image *in small parts* → learns what's important → combines all these patterns → predicts what the image shows

Example

Let's say we want to teach a computer to

recognize a **cat**.

A CNN learns:

- In early layers → **edges, lines**
- In middle layers → **ears, eyes, fur texture**
- In deeper layers → **whole face of a cat**

CNN Architecture (Simple View)

Here's what a CNN looks like:
Input Image → Convolution → ReLU → Pooling → Convolution → ReLU → Pooling → Flatten → Fully Connected (Dense) → Output
Let's break it step-by-step

1. Convolution Layer

Idea:

This layer uses **filters (kernels)** to scan the

image and find patterns (like edges).
A filter is a small grid (like 3×3 or 5×5 numbers).
It **slides** across the image and multiplies its values with the pixels — this operation is called **convolution**.

Terms:

Filter / Kernel

A small matrix (e.g., 3×3) that detects patterns.
Example:
Filter: [[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]]
This detects **vertical edges**.

Stride

How many pixels the filter moves each time.

- **Stride = 1** → moves 1 pixel at a time (output is large)
- **Stride = 2** → moves 2 pixels at a time (output is smaller)

Padding

What to do with image borders.

- **Valid Padding** → No padding (image shrinks after convolution)
- **Same Padding** → Add zeros around the image so output size = input size

Example:

If you have a 28×28 image and use **"same" padding**, your output stays 28×28.

If you use **"valid"**, the output becomes smaller (like 26×26).

Output Size Formula:

For a convolution:

$$O = \frac{(W - K + 2P)}{S} + 1$$

Where:

- = Output size
- = Input size
- = Kernel size
- = Padding
- = Stride

2. Activation Function ReLU

After convolution, we apply **ReLU (Rectified Linear Unit)** to add non-linearity.
Formula:
$f(x) = \max(0, x)$

3. Pooling Layer

This reduces the image size → less computation and prevents overfitting.

Types:

- **Max Pooling:** Takes the **maximum value** in a region
- **Average Pooling:** Takes the **average**

Example (2×2 Max Pooling):
Input: 1 3 2 4 Output: 4 ← (max of 1,2,3,4)

4. Flatten Layer

After convolution + pooling, we have 2D

data. We convert it to **1D** so we can feed it into a fully connected network.
Example:
[[1,2], [3,4]] → [1,2,3,4]

5. Fully Connected (Dense) Layer

This layer connects every neuron to every other neuron.
It takes all learned features and makes the **final decision** (like “cat” or “dog”).
Usually, the last layer uses **Softmax** → gives probabilities for each class.
Example output:
Cat: 0.90 Dog: 0.05 Bird: 0.05