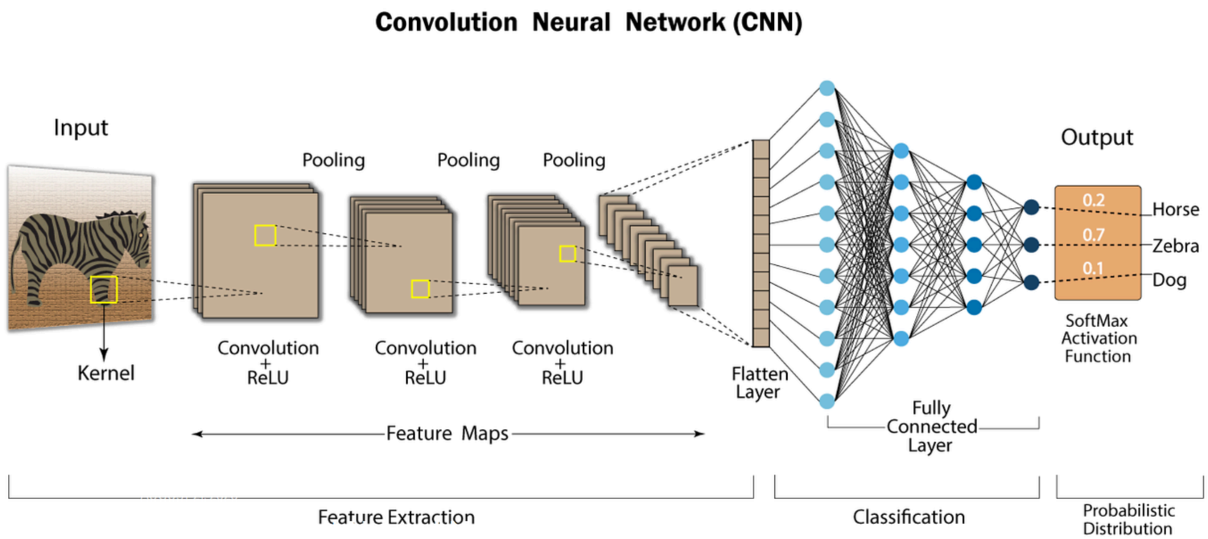
Introduction to Convolutional Neural Network (CNN)



- A **Convolutional Neural Network (CNN)** is a special type of **neural network** mainly used to **analyze visual data like images**.
- It can also work on audio, videos, and text (in some cases).
- It is designed to automatically learn features from the input data **without needing manual feature selection**.

💭 Imagine This:

- Think of an image like a grid made of tiny colored squares — these are **pixels**.
- A CNN looks at these grids in **small parts (patches)** and learns **patterns**, like edges, corners, textures, etc., just like your brain does when you see a picture.

📸 Why CNN Instead of Regular Neural Network for Images?

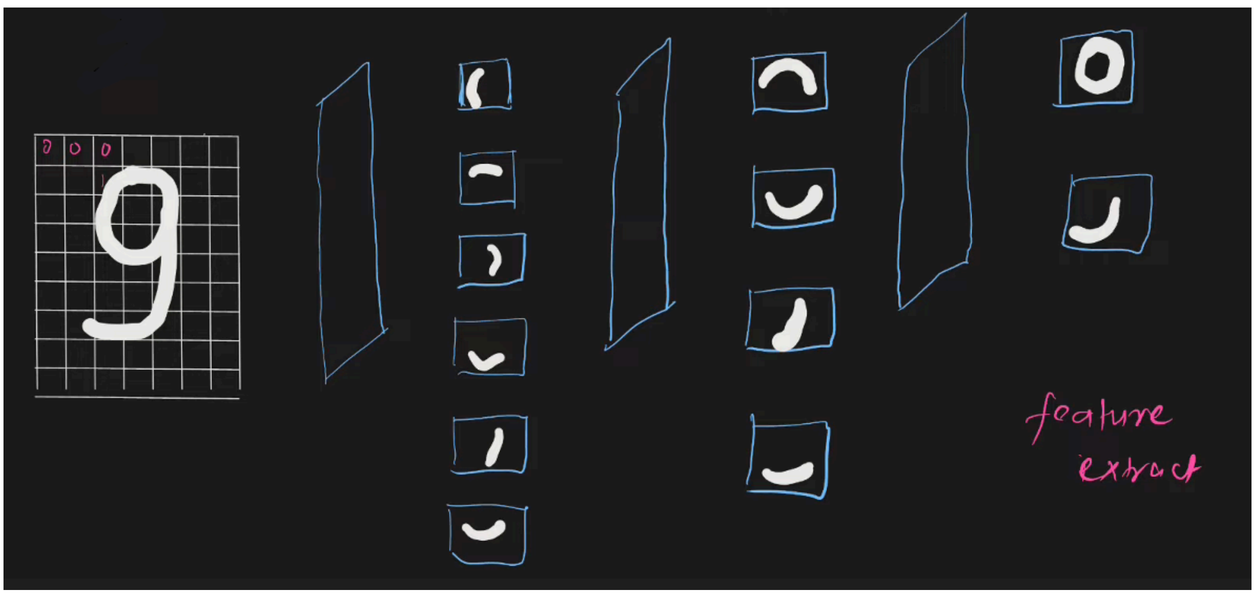
Regular (fully connected) neural networks:

- Don't scale well for images (too many connections!).
- Can't handle position and spatial relationships well.

CNNs fix this by:

- Looking at **local regions** of the image.
- Using fewer parameters.
- Learning **spatial features** like edges, curves, and shapes.

🧩 What is "Convolution"?



- Convolution means **sliding a small filter (or kernel)** over the image and **multiplying** it with the local patch of the image.
- This detects **patterns** like edges, corners, etc.

📐 How does it work?

Imagine this like a stencil or cookie-cutter that goes over each small part of the image and checks:

- ***"Is there an edge here?"***

- ***“Is this area dark or light?”***

⚙️ Parameters:

- `filters`: Number of different pattern detectors (like 32 or 64).
- `kernel_size`: Size of the filter (e.g., 3×3 or 5×5).
- `stride`: How many steps it moves each time (default = 1).
- `padding`: Whether to add extra border to keep size same.
    - `"valid"`: No padding, output is smaller.
    - `"same"`: Adds padding, keeps output size same.

Fully Connected (Dense) Layers:

- Used at the end of the network for classification/regression.
- Flattens feature maps and connects every neuron to output predictions.

Common CNN Architectures:

- **LeNet-5** (Early CNN for digit recognition)
- **AlexNet** (Deep learning breakthrough in 2012)
- **VGGNet** (Uses small 3×3 filters repeatedly)
- **ResNet** (Uses skip connections to train very deep networks)
- **EfficientNet** (Balances accuracy and computational efficiency)

Applications of CNNs:

- **Image Classification** (e.g., identifying objects in photos)
- **Object Detection** (e.g., YOLO, Faster R-CNN)
- **Semantic Segmentation** (e.g., medical imaging)
- **Face Recognition** (e.g., DeepFace)

- **Video Analysis & Autonomous Driving**

Important Components of CNN

Activation Function (Usually ReLU)

After convolution, we apply an activation function like **ReLU** (Rectified Linear Unit):

- ReLU = **max(0, x)**
- It removes negative values → helps the network **focus on important patterns**.

Pooling Layer (Downsampling)

This layer **reduces the size** of the image but **keeps the important parts**.

- Common type: **Max Pooling**
  - It takes a patch (say 2×2) and picks the **maximum value**.
- This helps in:
  - Reducing computation.
  - Preventing overfitting.
  - Making the system **more robust** to small changes.

Flattening Layer

Once the image is small enough, it is converted into a **1D array** (flat vector) to be processed by a **fully connected layer** (like in regular neural networks).

Fully Connected (Dense) Layer

This is the final part of the CNN:

- It uses all the learned patterns to make a decision:
  - "This looks like a cat."
  - "This is 97% a dog."

Output Layer

- Uses **Softmax** or **Sigmoid**:
  - For binary classification (e.g., cat vs not cat): **Sigmoid**.
  - For multi-class (e.g., cat, dog, bird): **Softmax**.

🖼️ Visual Example (Step-by-Step Image Classification)

1. **Input Image** → 28×28 pixels (e.g., handwritten digit)
2. **Conv Layer** → Applies filters (e.g., detects edges)
3. **ReLU** → Removes negative values
4. **Pooling Layer** → Reduces image size
5. **Flatten** → Converts 2D into 1D
6. **Dense Layer** → Learns final patterns
7. **Output Layer** → Predicts digit (0–9)