


Convolutional Neural Network




Prof. Somesh Nandi
Dept of AIML
RVCE

Introduction

 A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision.

 **Why CNN?**

Key Limitations of Traditional Neural Networks:

-  **High Computational Cost**
-  **Loss of Spatial Information**
-  **Inefficient for Large Images**

Why CNN?

Example: Handwritten Digit Recognition

🤖 Task: Classify a handwritten digit from a grayscale of 28×28 -pixel image.

Using a Traditional Neural Network:

Input size: $28 \times 28 = 784$ pixels. Each pixel is treated as an independent feature.

Number of parameters (1 hidden layer, 1000 neurons): $784 \times 1000 = 784,000$.

This network ignores spatial patterns (e.g., edges or curves forming digits) and relies on brute-force learning, leading to poor efficiency and generalization.

Using a CNN:

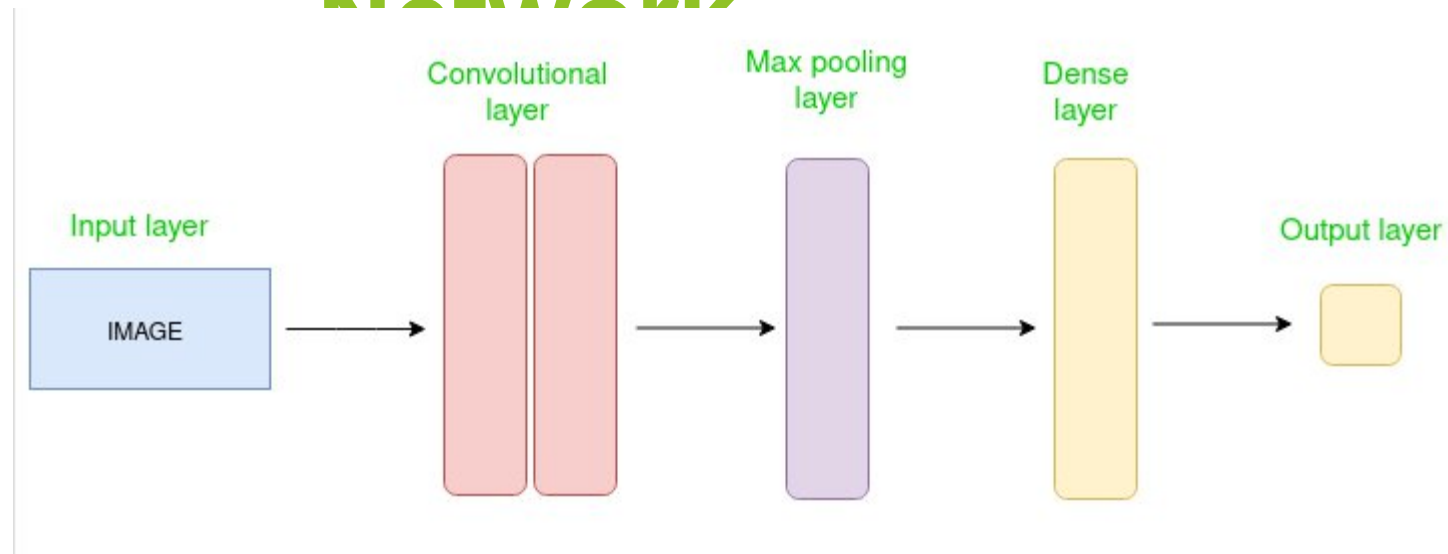
Instead of feeding all pixels at once, a convolutional layer applies filters (e.g., 3×3 to detect edges, curves, and textures).

A pooling layer (e.g., max pooling) reduces dimensions while preserving key features.

The network learns hierarchical patterns: edges \rightarrow shapes \rightarrow digits.

Number of parameters for a filter: $3 \times 3 = 9$. Even with multiple filters, parameter count is far smaller than an FCNN.

Basic Structure of a Convolutional Network



- The Convolutional layer **applies filters** to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction.
- The network learns the optimal filters through backpropagation and gradient descent

Convolutional Layer (Filter)

A **filter** (also called a **kernel**) in a Convolutional Neural Network (CNN) is a small matrix of learnable weights used to detect specific features in input data, such as **edges**, **textures**, or **patterns**.

Filters are at the heart of the convolutional operation, enabling CNNs to learn and extract hierarchical features from input images.

Size:

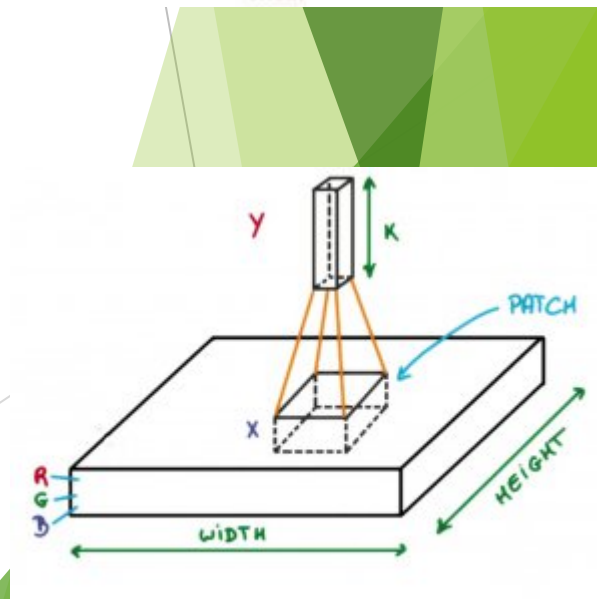
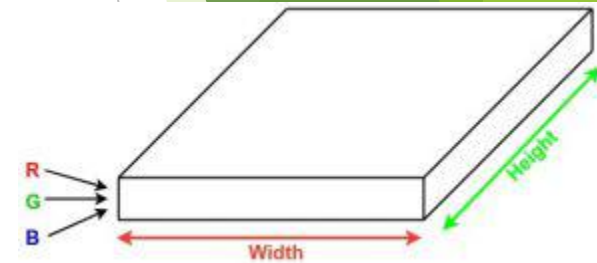
Filters are typically much smaller than the input data. Common sizes include 3×3 , 5×5 , 7×7

- Example: For a 28×28 , 3×3 filter slides across the image to analyze local regions.

Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically.

Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths.

Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution



1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4*	7	2
9	10	12	6	9	8

*

1	0	-1
1	0	-1
1	0	-1

=

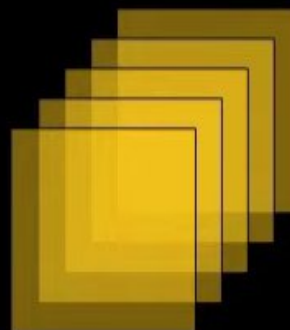
1*1	6 0	9 -1	10	2	8
2*1	5 0	1 -1	8	4	2
3*1	7 0	4 -1	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

-8	-9	-2	14
6	-3	-13	9
4	-4	-8	5
2	2	1	-3

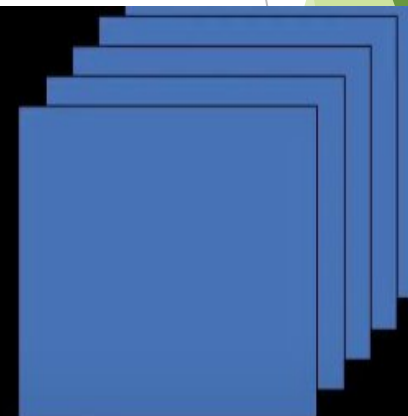
$$(n \times n) * (f \times f) = (n - f + 1) \times (n - f + 1)$$

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4*	7	2
9	10	12	6	9	8

*



=



$f \times f \times c$

$(n-f+1) \times (n-f+1) \times c$



*

1	0	-1
1	0	-1
1	0	-1

=



Vertical Edges

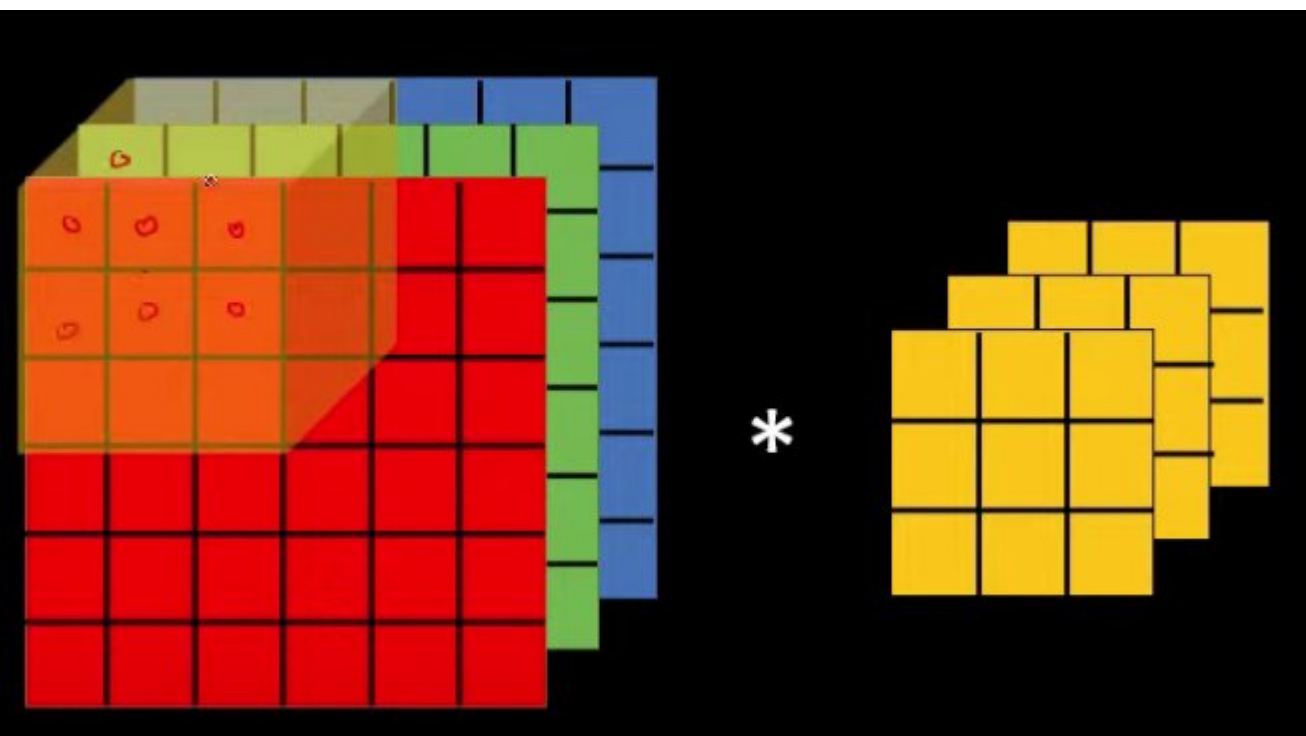


*

1	1	1
0	0	0
-1	-1	-1

=

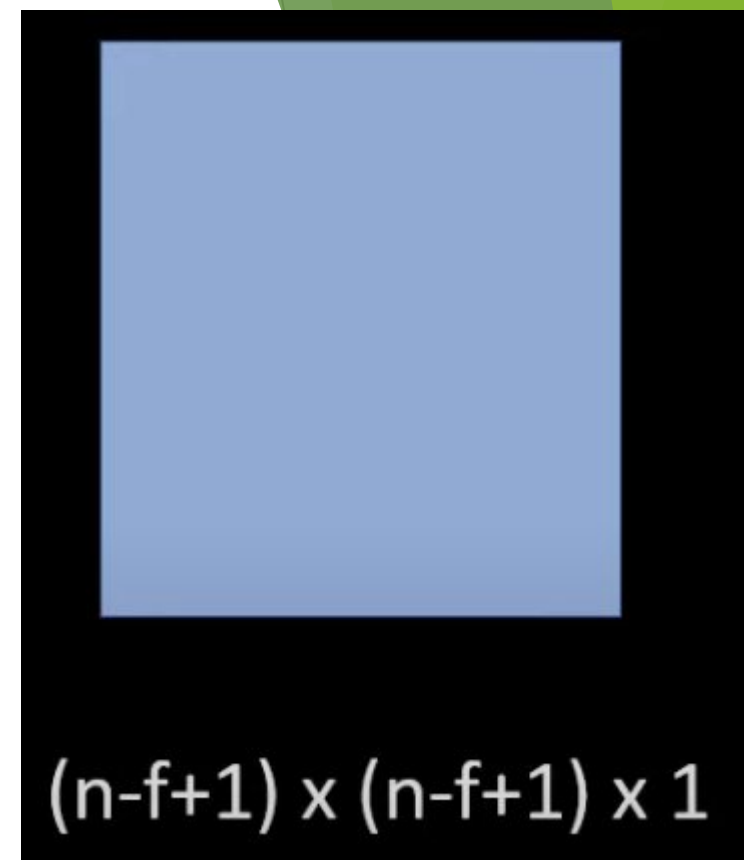




$n \times n \times 3$

$f \times f \times 3$

$=$



$(n-f+1) \times (n-f+1) \times 1$

Stride

 **Stride** in a Convolutional Neural Network (CNN) refers to the step size with which the convolutional filter moves across the input feature map during the convolution operation.

Key Characteristics of Stride

1. Default Stride:

1. A stride of $S=1$ means the filter moves one pixel at a time.
2. This results in maximum overlap between adjacent filter applications and produces the largest output size.

2. Larger Stride:

1. A stride of $S=2$ means the filter moves two pixels at a time.
2. This skips some input data, resulting in a smaller output feature map (down sampling).
3. $S = \lfloor \frac{n-f}{s} \rfloor + 1$

Padding

Padding in Convolutional Neural Networks (CNNs) refers to adding extra layers of pixels around the edges of an input image or feature map. This technique is used to control the spatial dimensions of the output after a convolution operation.

Why Use Padding?

1. Preserve Spatial Dimensions:

- 1. Without padding, applying filters reduces the dimensions of the input feature map, especially near the edges.**
- 2. Padding ensures that the output has the same dimensions as the input (in "same" padding).**

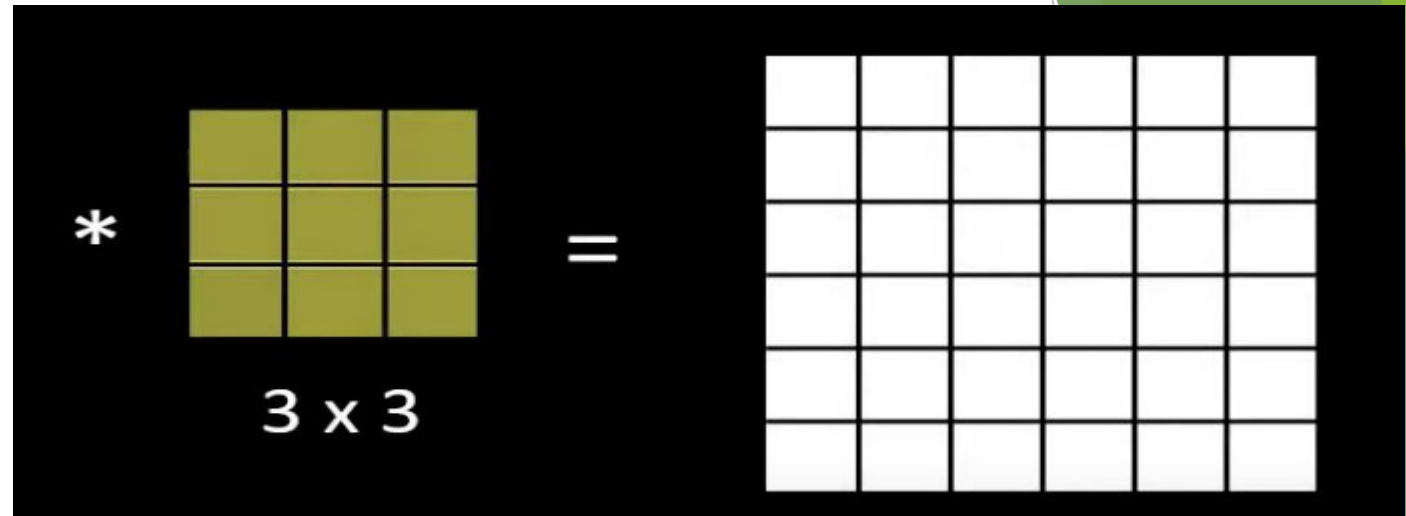
2. Prevent Information Loss:

- 1. Filters inherently struggle to process the edges of an image because there are fewer surrounding pixels.**
- 2. Padding provides artificial pixels (usually zeroes) so that edge features are captured effectively.**

3. Allow Deeper Networks:

- 1. Maintaining dimensions across layers (using padding) helps design deeper networks without quickly shrinking the feature maps.**

0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0



Types of Padding

1.Zero Padding:

1. Adds zero-value pixels around the edges.
2. Most common type of padding used in CNNs.

2.Valid Padding:

1. No padding is added. Convolutions are applied only where the filter completely overlaps the input.
2. Results in smaller output dimensions.

3.Same Padding:

1. Adds enough padding so the output dimensions are the same as the input dimensions.
2. Ensures consistent size across layers.

Pooling

- 🚗 **Pooling in Convolutional Neural Networks (CNNs)** is a technique used to reduce the spatial dimensions of the input feature maps while retaining essential features.
- 🚗 It helps in reducing the computational complexity and controlling overfitting by summarizing the features in a region of the image.
- 🚗 Pooling is typically applied after the convolutional layers to reduce the size of the feature maps and make the network more computationally efficient.
- 🚗 There are two type of pooling : Max Pooling and Average Pooling



Max Pooling

0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5

0.1	0.3	0.5
0.1	0.3	0.5
0.1	0.3	0.5

Average Pooling

8	1	3	6		
3	2	2	1		
5	0	7	1		
2	4	9	7		
				3.5	3
				2.8	6

Typical Settings

- **Filter Size (Kernel Size):**
 - Commonly 3x3 or 5x5 for convolution operations.
 - Determines the receptive field to extract features like edges and textures.
- **Stride:**
 - Default is 1 for overlapping coverage.
 - Larger strides (e.g., 2) reduce dimensionality but risk losing fine-grained details.
- **Padding:**
 - Same Padding: Maintains input dimensions by padding with zeros.
 - Valid Padding: No padding, reduces spatial dimensions.
- **Pooling Layer:**
 - Typically uses max pooling or average pooling with a size of 2x2 and a stride of 2.
 - Reduces spatial dimensions and retains dominant features.

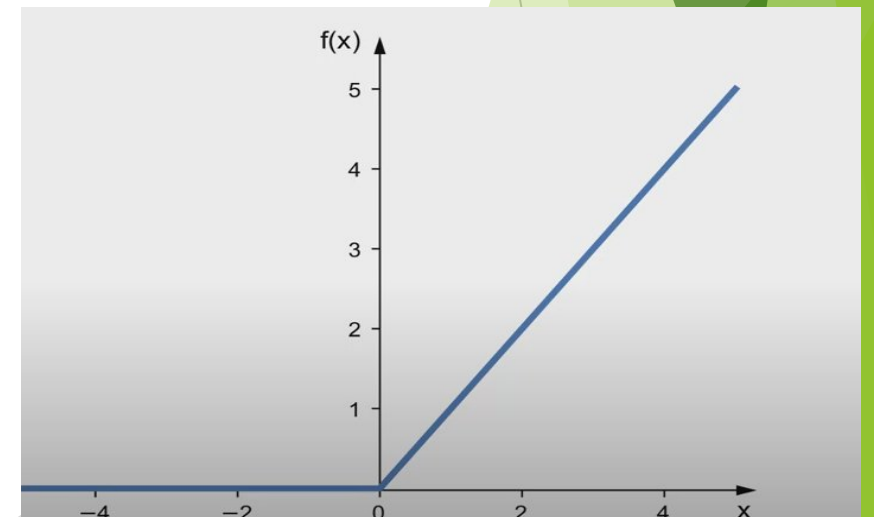
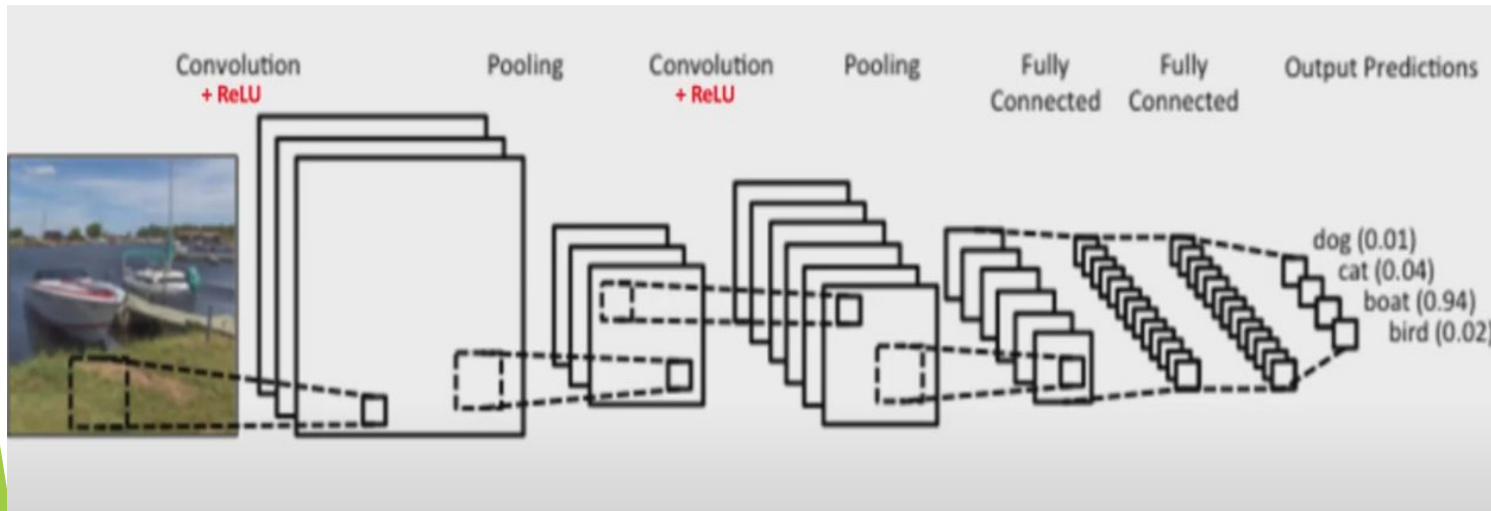
Typical Settings

- **Activation Functions:**
 - ReLU (Rectified Linear Unit) is widely used for introducing non-linearity.
 - Alternatives include Leaky ReLU, Tanh, or Sigmoid depending on the task.
- **Number of Filters (Channels):**
 - Starts with fewer filters (e.g., 32 or 64) in early layers.
 - Increases in deeper layers (e.g., 128, 256, 512) to capture complex patterns.
- **Fully Connected Layers:**
 - Typically 1–2 layers at the end, used for classification or regression tasks.

The ReLU Layer

 The ReLU (Rectified Linear Unit) layer is a key component in convolutional neural networks (CNNs) that helps introduce non-linearity to the model.

The ReLU activation function is defined as $f(x) = \max(0, x)$. This means that if the input is positive, it passes through unchanged, but if the input is negative, it is set to zero. This introduces non-linearity into the network, allowing it to learn complex patterns.



The ReLU Layer



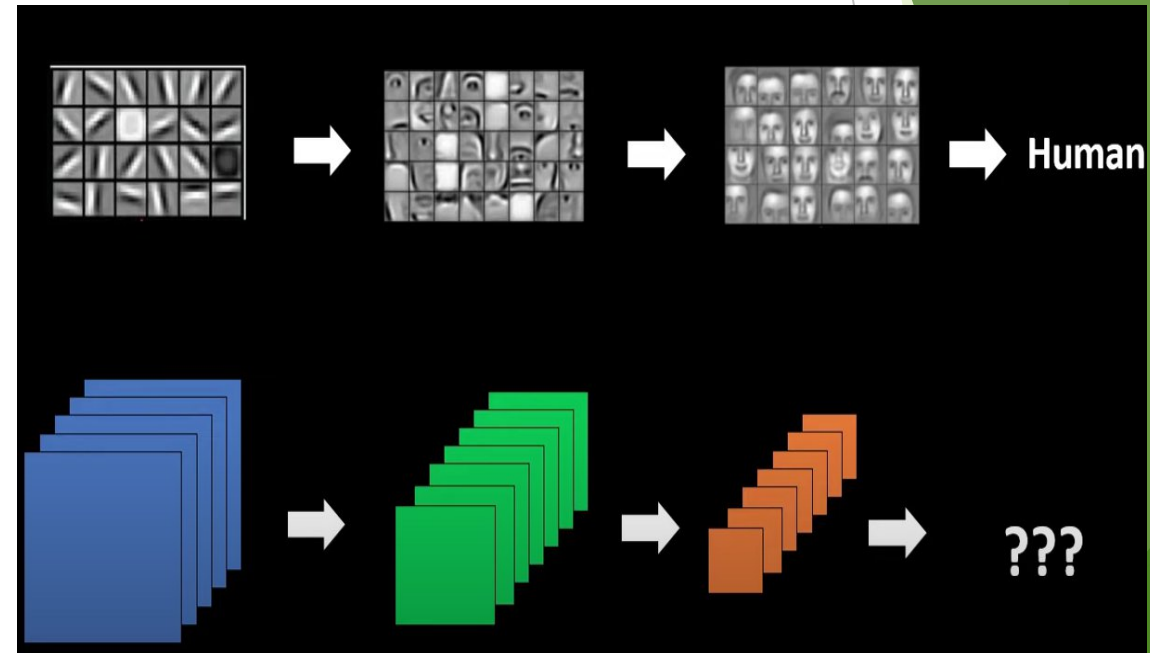
ReLU
→



Fully Connected Layer

🚗 After extracting all the features, what next?

🚗 How do I classify or label?

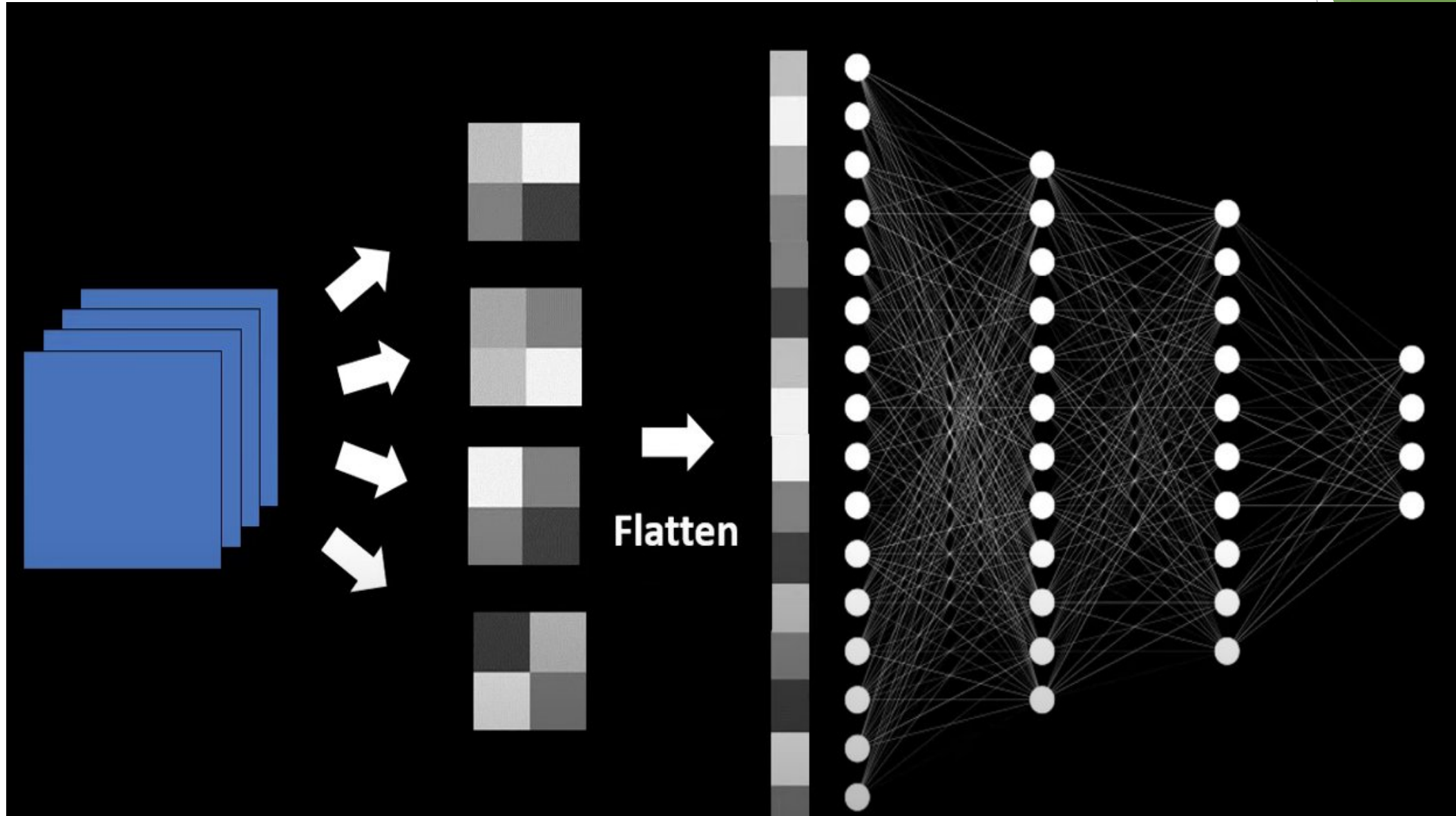


[Input Image] → [Convolutional Layers] → [Pooling Layers]
→ **[Fully Connected Layers]** → [Output]

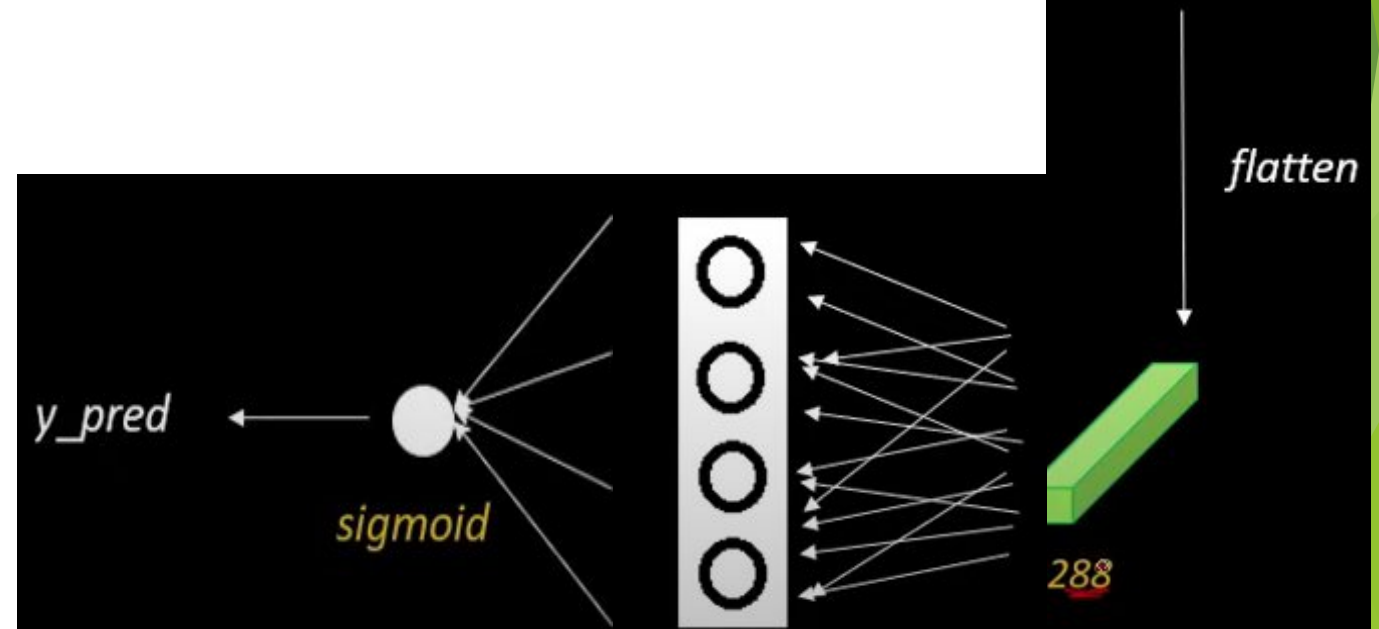
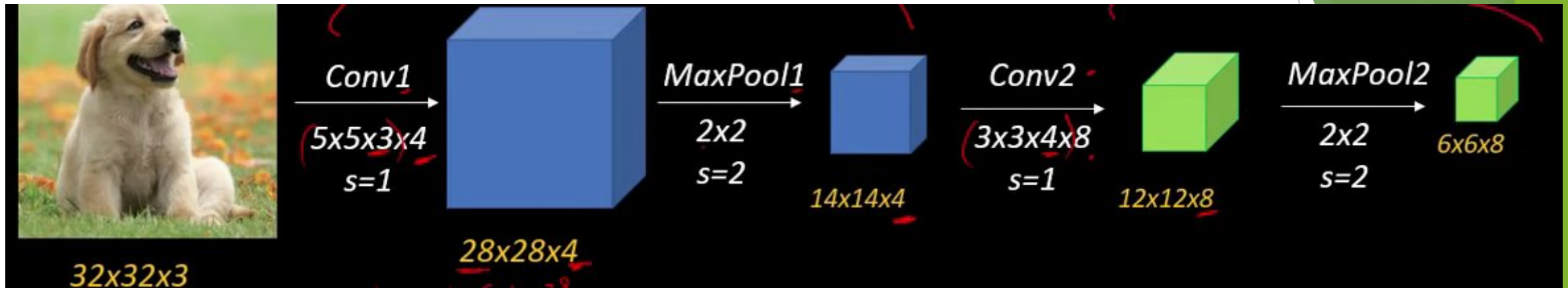
Fully Connected Layer

- Fully connected layers are dense network of neurons.
- Applied after convolutional and max pooling layers.
- Classifies the output
- Associate features to a particular label

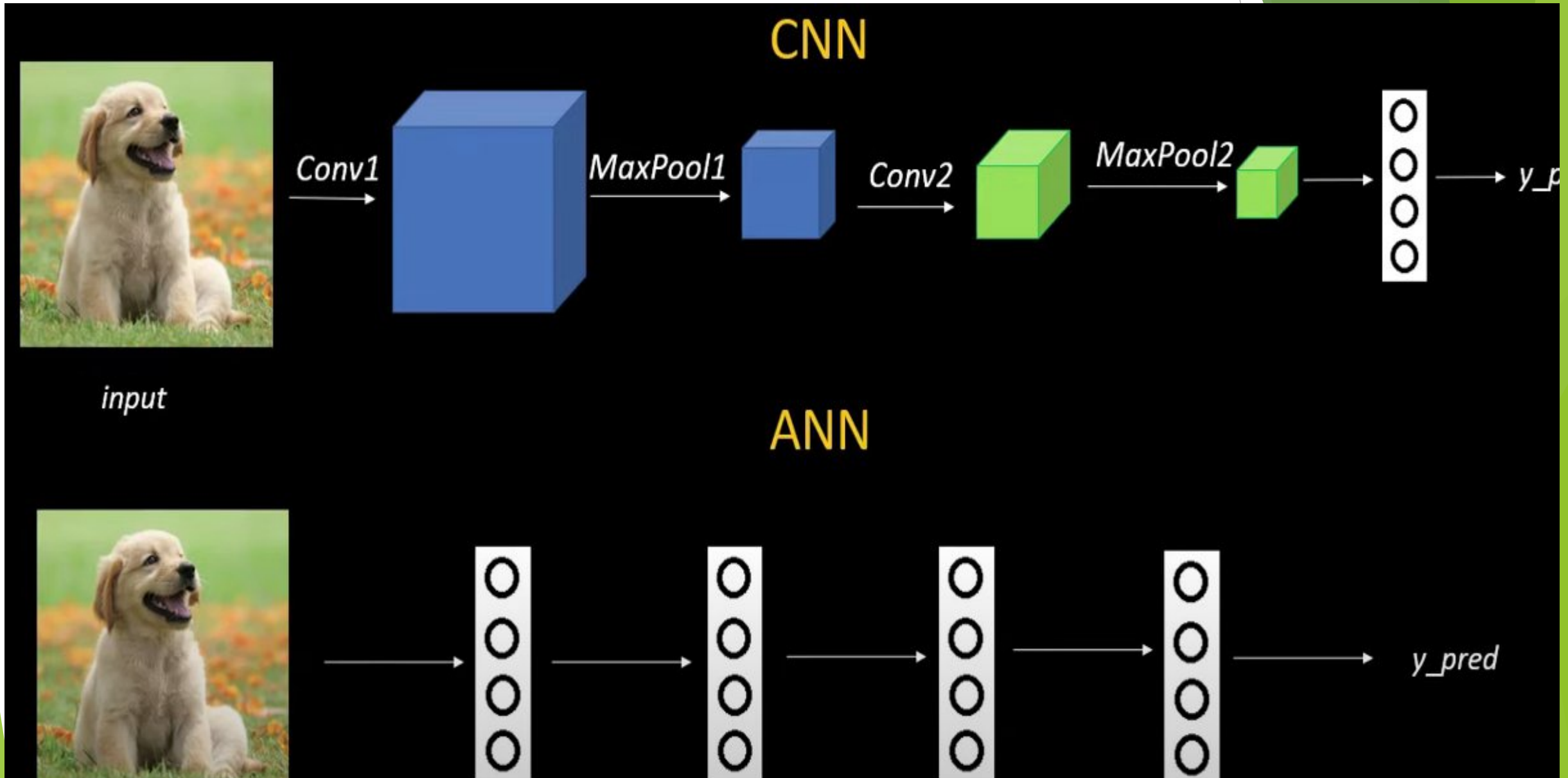
Fully Connected Layer



Summary



Comparison between CNN and ANN



Data Augmentation

- **What is Data Augmentation?**

- It's a technique to reduce overfitting by generating new training data from existing data using transformations.

- **Why is it Useful?**

- Increases the generalization ability of the model.
- Helps the model recognize objects in different orientations, lighting, and conditions.

- **Common Transformations in Image Processing:**

- **Translation:** Shift the image slightly in any direction.
- **Rotation:** Rotate the image by a certain angle.
- **Reflection (Mirroring):** Flip the image horizontally or vertically.
- **Patch Extraction:** Use smaller patches or crops from the image.
- **Color Intensity Adjustment:** Vary the brightness or color levels.