

# CNN-based Image Features

Convolutional Neural Networks learn hierarchical visual features

## Feature Hierarchy Across Layers

### Early Layers

Edges  
Textures  
Colors



### Middle Layers

Object Parts  
Patterns



### Deep Layers

High-level Semantic  
Concepts & Objects



Convolutional filters automatically learned from data



Pooling operations provide translation invariance



## Pre-trained CNNs as Feature Extractors

VGG, ResNet, and more serve as powerful feature extractors

# Historical Context of CNNs

## The evolution of Convolutional Neural Networks in computer vision

1989

### LeNet (Yann LeCun)

First successful CNN, used for handwritten digit recognition. Applied to automatic postal code recognition systems

2012

### AlexNet (ImageNet Challenge)

Marked the resurgence of deep learning. Achieved overwhelming performance through large-scale GPU training

2014

### VGGNet & GoogLeNet

Explored deeper network architectures. VGG featured simple, uniform structure; GoogLeNet introduced Inception modules

2015

### ResNet (Residual Networks)

Enabled training of very deep networks (152 layers) with skip connections. Achieved human-level image recognition performance

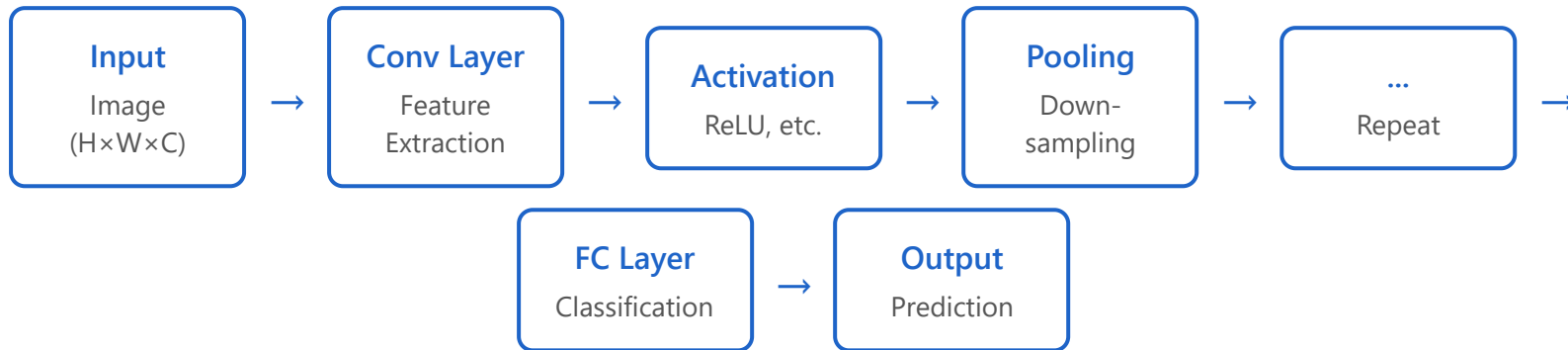
2017+

### Modern Architectures

Emergence of modern architectures like EfficientNet and Vision Transformer (ViT) pursuing both efficiency and performance

# CNN Layer Architecture

## Typical structure of a Convolutional Neural Network



**Convolutional Layers:** Extract local features from images



**Activation Functions:** Add non-linearity (mainly ReLU)



**Pooling Layers:** Reduce spatial dimensions and achieve invariance



**Fully Connected:** High-level reasoning and classification

**Key Principle:** Learn hierarchical features by repeating Conv-Activation-Pooling blocks multiple times

# Convolution & Pooling Operations



## Convolution Operation

### How it Works

A small filter (kernel) slides over the image, performing element-wise multiplication and summation

### Key Parameters

- **Filter size:**  $3 \times 3$ ,  $5 \times 5$ , etc.
- **Stride:** Step size
- **Padding:** Border handling

### Features

- Detects local patterns
- Parameter efficiency through weight sharing
- Preserves spatial relationships

### Learning

Filter weights are automatically learned from data through backpropagation



## Pooling Operation

### Purpose

Reduces spatial dimensions of feature maps, decreases computation, and achieves invariance

### Max Pooling

Selects maximum value within window. Preserves strongest features. Most widely used

### Average Pooling

Computes average value within window. Extracts smoother features

### Advantages

- Invariance to small translations/deformations
- Reduces overfitting
- Improves computational efficiency
- Expands receptive field



**Together:** Convolution extracts features, and Pooling compresses and makes them more robust

# Key Concepts in CNN Operations

## Receptive Field

- The region of input image that a single neuron "sees"
- Deeper layers have wider receptive fields
- Enables capturing larger contextual information

## Output Size Calculation

Output size =  $\lfloor (\text{Input size} - \text{Filter size} + 2 \times \text{Padding}) / \text{Stride} \rfloor + 1$

Example: Input 32×32, Filter 5×5, stride 1, padding 2

→ Output:  $\lfloor (32 - 5 + 4) / 1 \rfloor + 1 = 32$  (size preserved)



## Why CNNs Work Well

### 1. Translation Invariance:

Can recognize objects even when their position changes

### 2. Hierarchical Learning:

Simple features → Complex features

### 3. Parameter Sharing:

Same filter applied across entire image

### 4. Sparse Connectivity:

Efficient learning through local connections



## Feature Visualization

### Early layers:

Basic elements like edges, corners, color blobs

### Middle layers:

Textures, repetitive patterns, object parts

### Deep layers:

Complete objects, scenes, semantic concepts

Filters in each layer learn increasingly complex and abstract features

## Interactive Learning Resource

Experience how CNNs work and visually understand the impact of hyperparameters



### CNN Explainer

Interactive visualization of Convolutional Neural Networks



[Visit CNN Explainer](#)



**Interactive Demo:** Visualize what happens in each CNN layer in real-time



**Hyperparameters:** Directly see the effects of kernel size, stride, padding, etc.



**Feature Maps:** Observe outputs and activations of each layer in real-time



**Educational:** The best tool to intuitively understand how CNNs work

#### Learning Tips

- Start with the "Understanding Hyperparameters" section to learn basic concepts
- Change various hyperparameter values and observe how the output changes

- Follow step-by-step how CNNs respond to real images