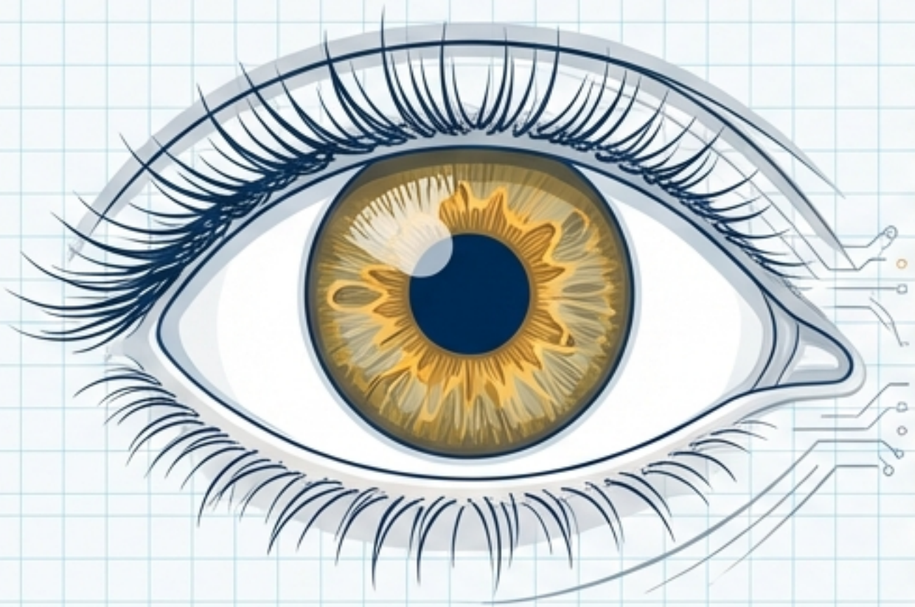
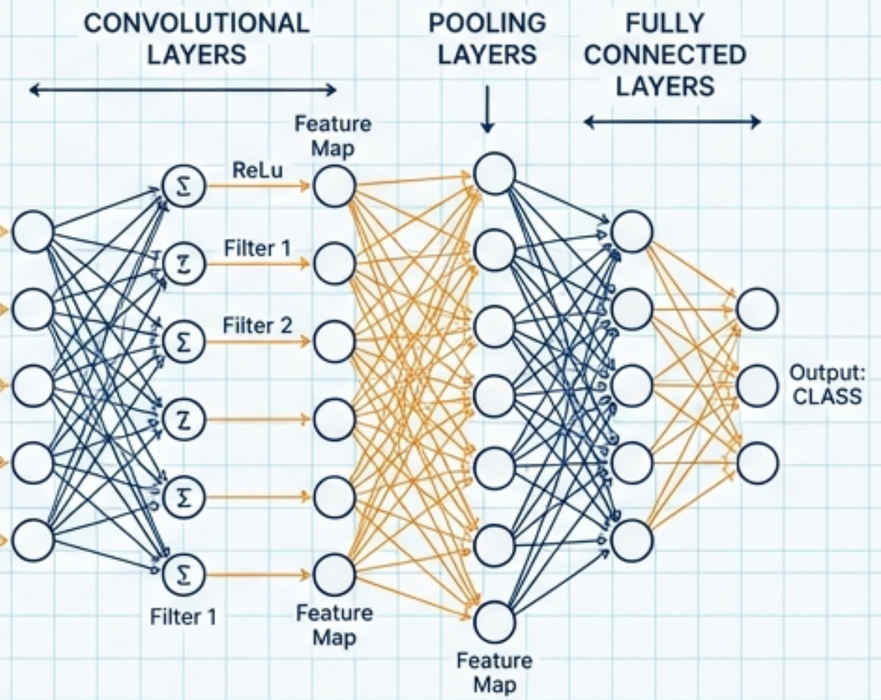
The Digital Eye: Mastering Convolutional Neural Networks (CNNs)

From Pixel Grids to Intelligent Vision – A Comprehensive Guide



120	150	180	140	140	120	100	80	60	40	20	
130	160	190	150	150	130	110	90	70	50	30	
140	170	200	160	140	120	120	100	80	60	40	
150	180	210	170	150	150	130	110	90	70	50	
160	190	220	180	160	140	140	120	100	80	60	
170	200	230	190	170	150	150	130	110	90	70	
180	210	240	200	180	180	160	140	120	100	80	
190	220	240	210	190	160	150	140	120	100	80	
190	220	250	210	190	170	170	150	130	110	90	90
200	230	255	220	200	200	180	160	140	120	120	100
210	240	255	230	210	190	190	170	150	130	110	110



HUMAN VISION
PHOTONS TO SIGNALS
Inter Medium

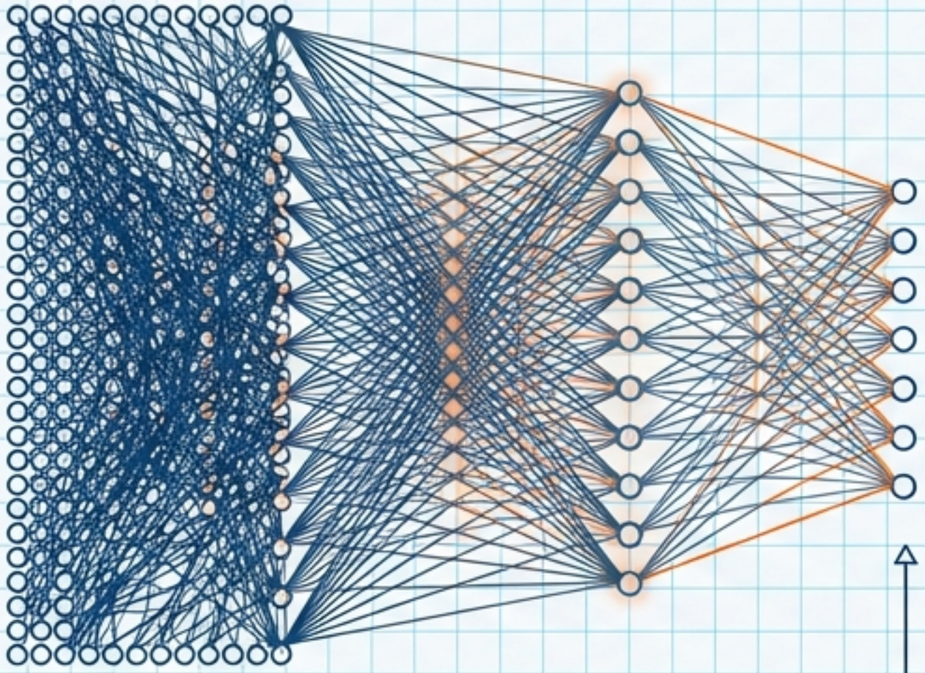
PIXEL MATRIX
(NUMERICAL DATA)
Inter Medium

COMPUTER VISION
DIGITAL IMAGE = MATRIX OF NUMBERS
Inter Regular

Architecture Series: Deep Learning Fundamentals

The Challenge: Why Not Just Connect Everything?

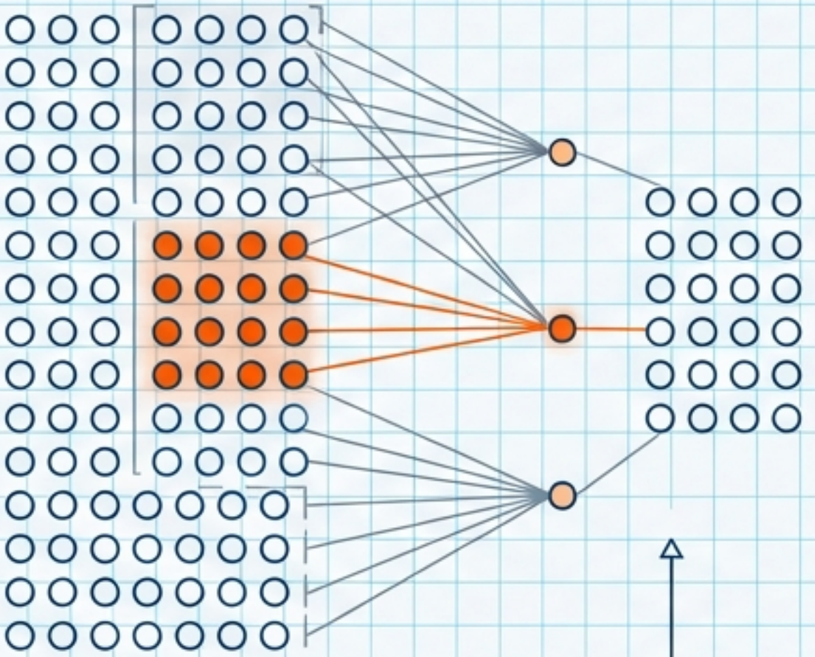
The Old Way: Fully Connected (MLP)

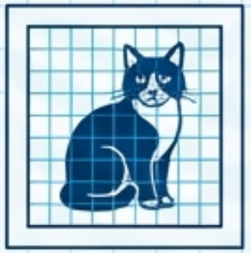


FLATTENED INPUT

Spatial Structure Destroyed

The Solution: Convolutional (CNN)





2D IMAGE INPUT
JetBrains Mono

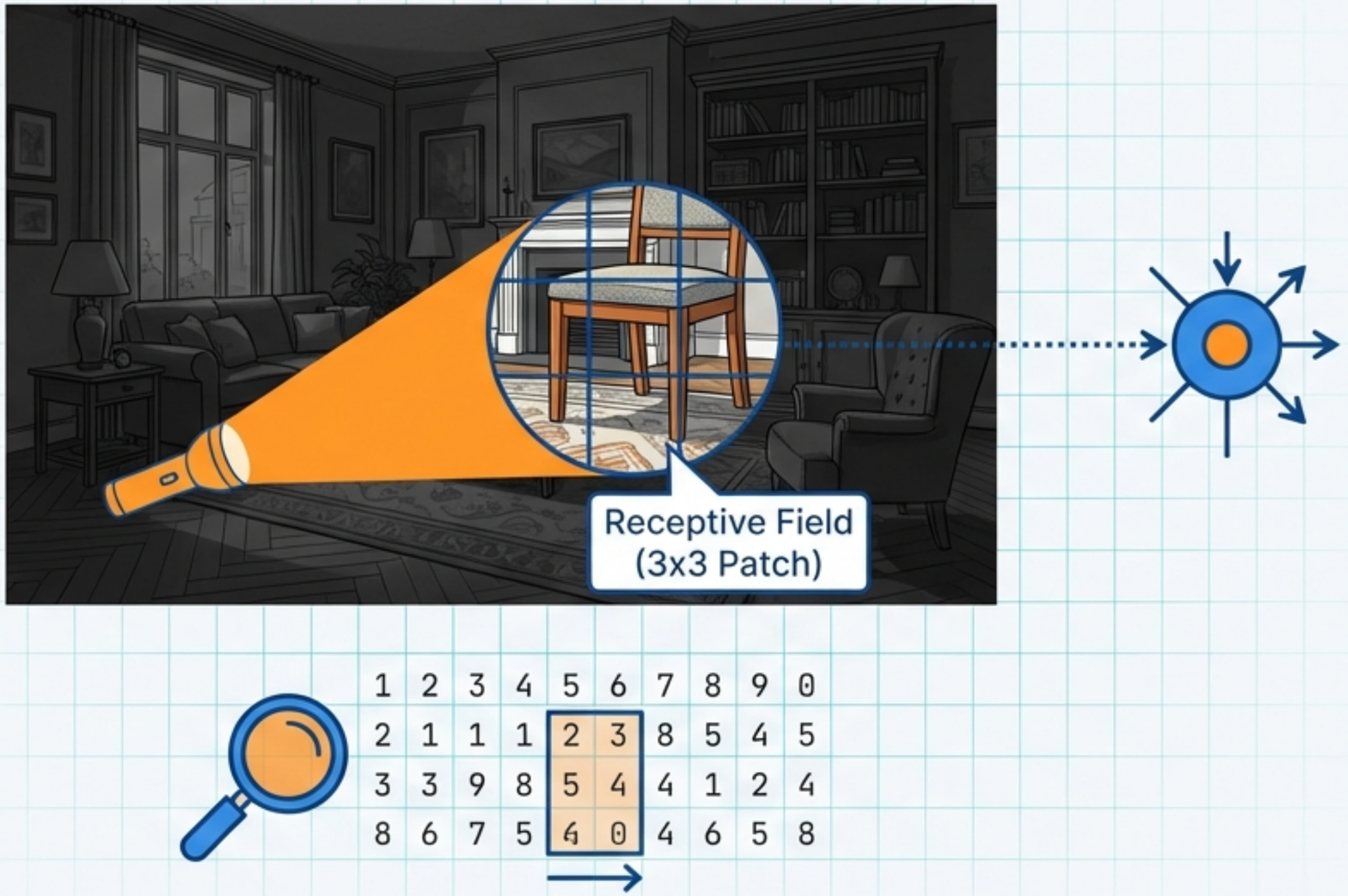
Spatial Structure Preserved

Why FC Networks Fail Images:

1. Loss of Spatial Structure (Flattening).
2. Translation Sensitivity (Moving the object breaks the recognition).
3. Parameter Explosion (Connecting every pixel = **Millions of weights**).

The Flashlight Analogy: Scanning for Features

A CNN does not see the whole image at once. It scans the image patch-by-patch, looking for specific patterns in local regions.

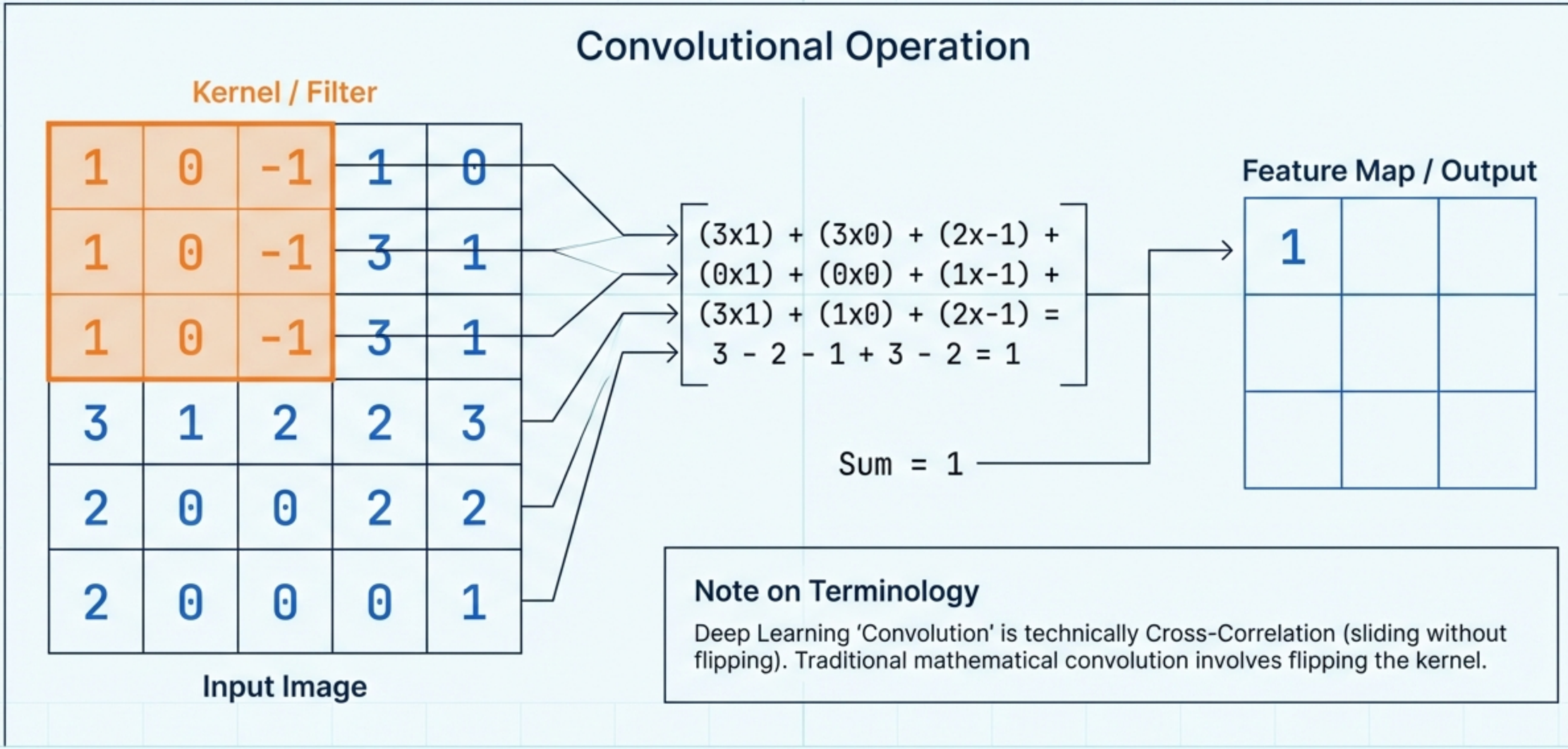


Technical Orange

Key Term: Local Connectivity

Neurons connect only to a small region of the input volume.

The Engine Room: The Convolution Kernel



Note on Terminology

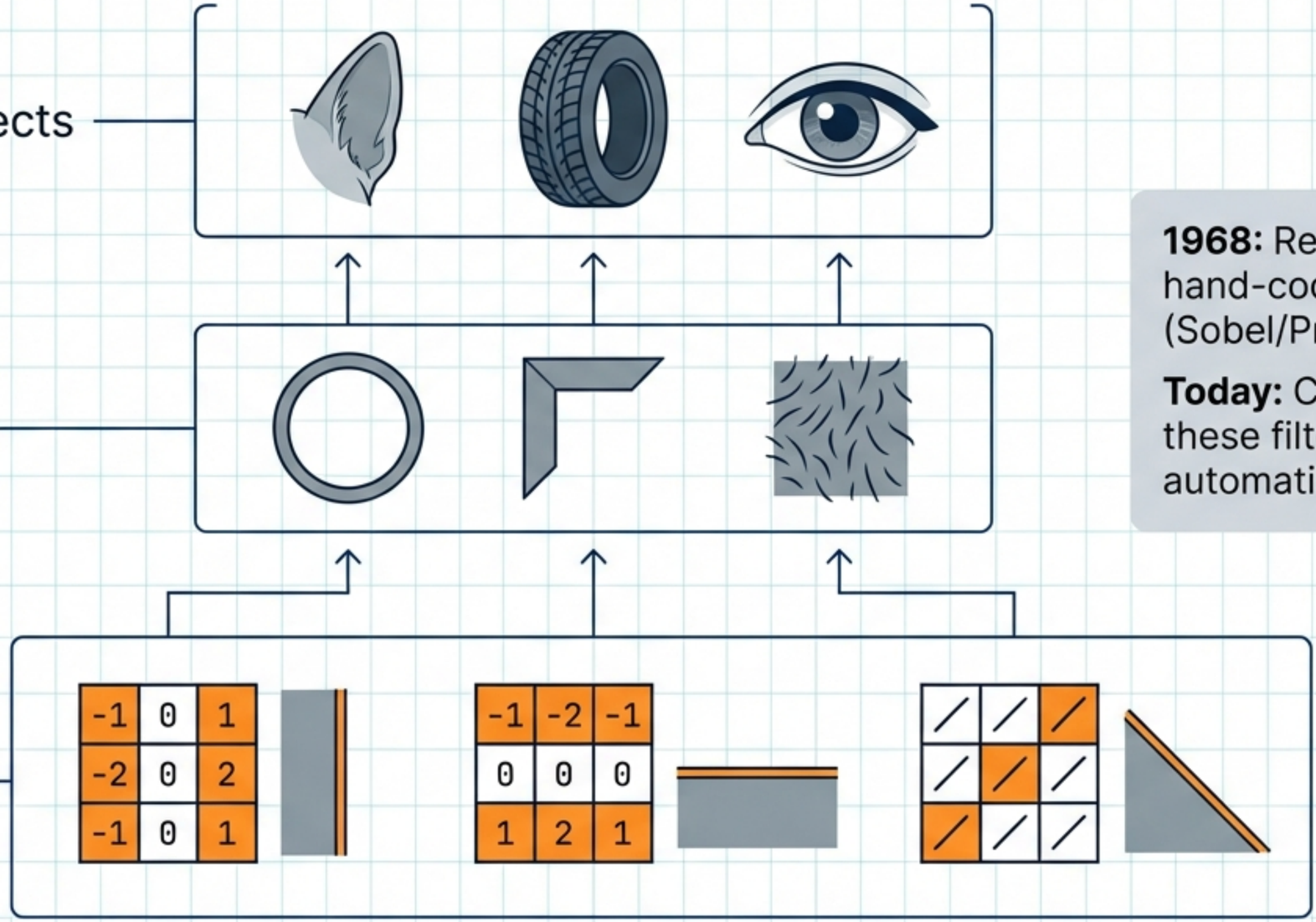
Deep Learning 'Convolution' is technically Cross-Correlation (sliding without flipping). Traditional mathematical convolution involves flipping the kernel.

The Hierarchy of Abstraction

High-Level: Objects

Mid-Level: Shapes & Parts

Low-Level: Edges & Gradients



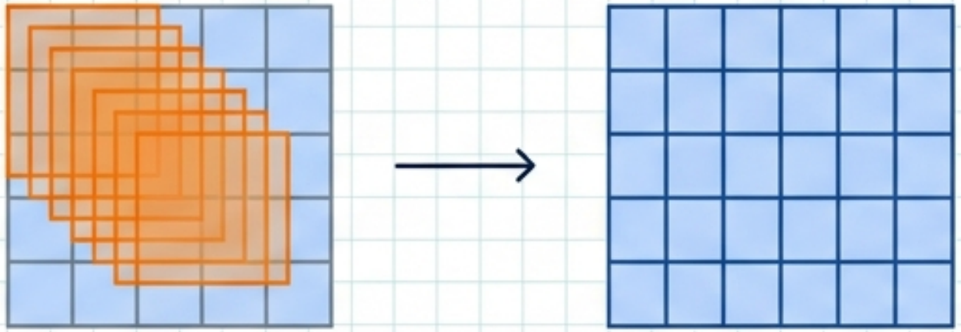
1968: Researchers hand-coded filters (Sobel/Prewitt).

Today: CNNs learn these filters automatically.

Controlling the Scan: Stride and Padding

Stride

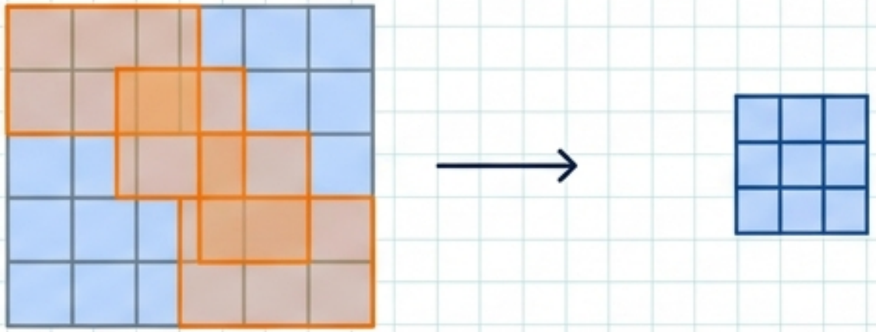
Stride = 1 (Heavy Overlap)



Large Output

Kernel moves 1 pixel/step.

Stride = 2 (Skipping)

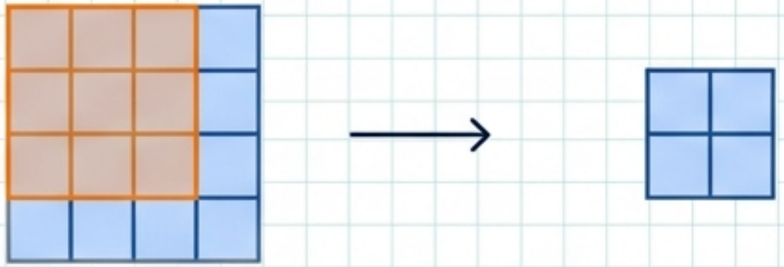


Smaller Output

Kernel moves 2 pixels/step.

Padding

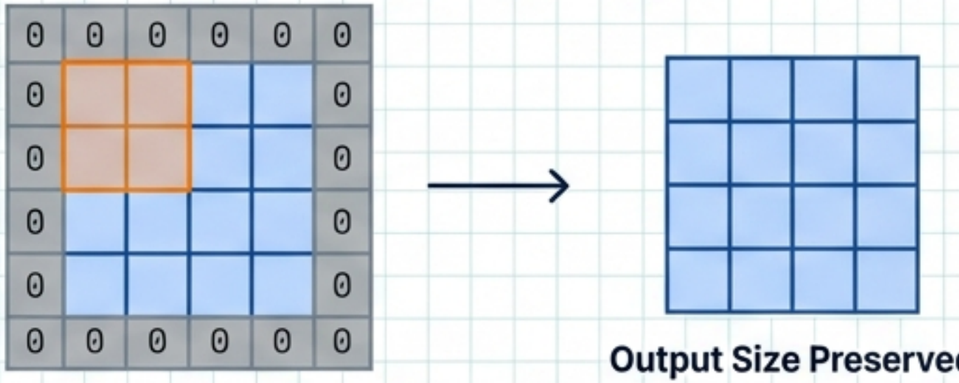
Valid (No Padding)



Output Smaller than Input

Kernel cannot reach corners.

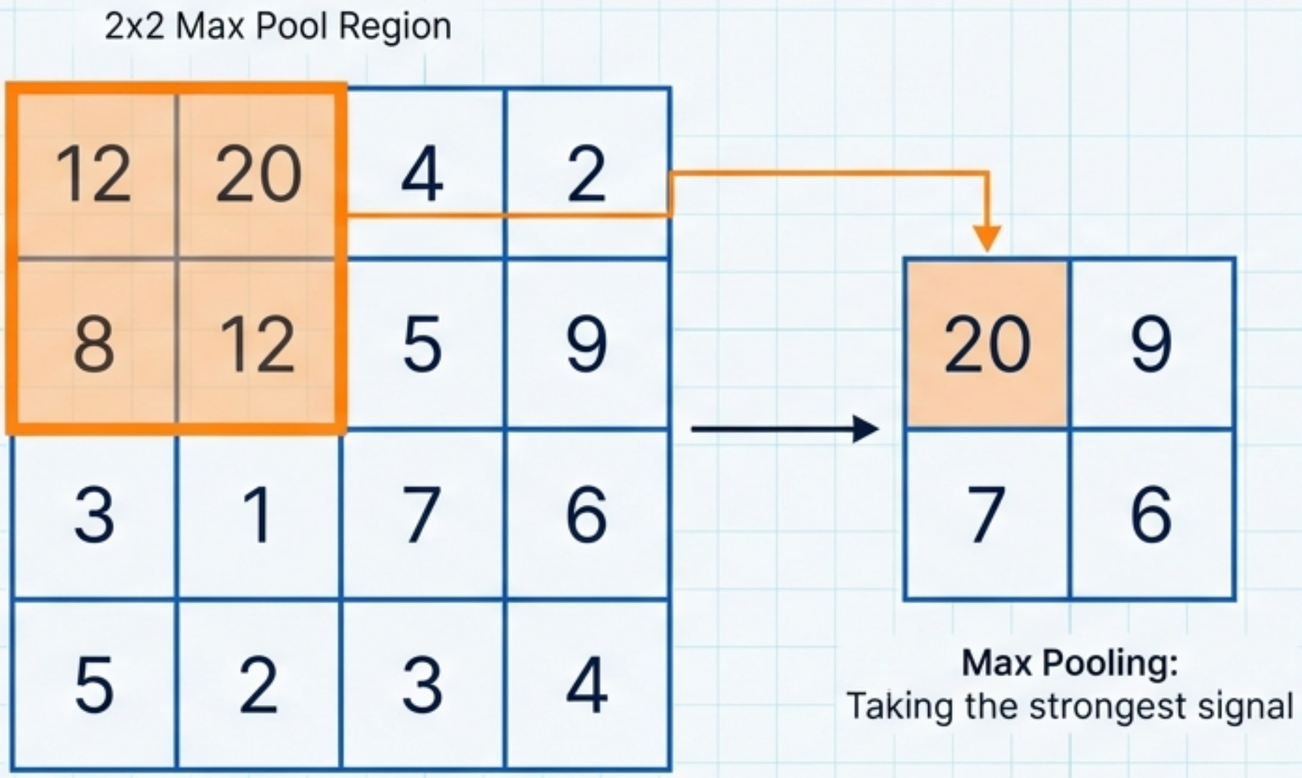
Same (Zero Padding)



Output Size Preserved

Kernel can center on corner pixel.

Summarizing the View: The Pooling Layer



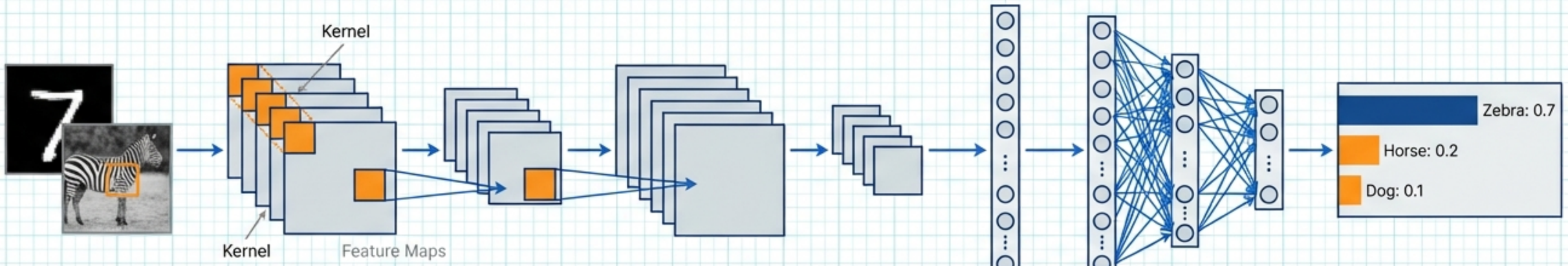
Max Pooling Operation
(stride=2, kernel=2)

Benefits of Pooling

1. **Dimensionality Reduction**
   (Shrinks the image)
2. **Parameter Efficiency**
   (Less to calculate)
3. **Translation Invariance**
   (Features detected regardless of slight shifts)

```
nn.MaxPool2d(kernel_size=2, stride=2)
```

The Assembly Line: A Complete CNN Architecture



Input Image (e.g., Zebra or Handwriting)
Inter Tight Bold

C1: Convolution + ReLU (Feature Extraction)
JetBrains Mono

S2: Pooling (Downsampling)
JetBrains Mono

C3: Convolution + ReLU
JetBrains Mono

S4: Pooling
JetBrains Mono

Flatten Layer (Converting 2D to 1D)
JetBrains Mono

Fully Connected Layers (Classification)

Output (Softmax)
JetBrains Mono

The Chef (Feature Extraction)
In Inter Tight Bold

The Critic (Classification)
In Inter Tight Bold

The Mathematics of Dimensions

Output Size Calculation Formula

$$N_{out} = \left\lfloor \frac{N_{in} + 2P - K}{S} \right\rfloor + 1$$

N_{in} = Input Size
P = Padding
K = Kernel Size
S = Stride

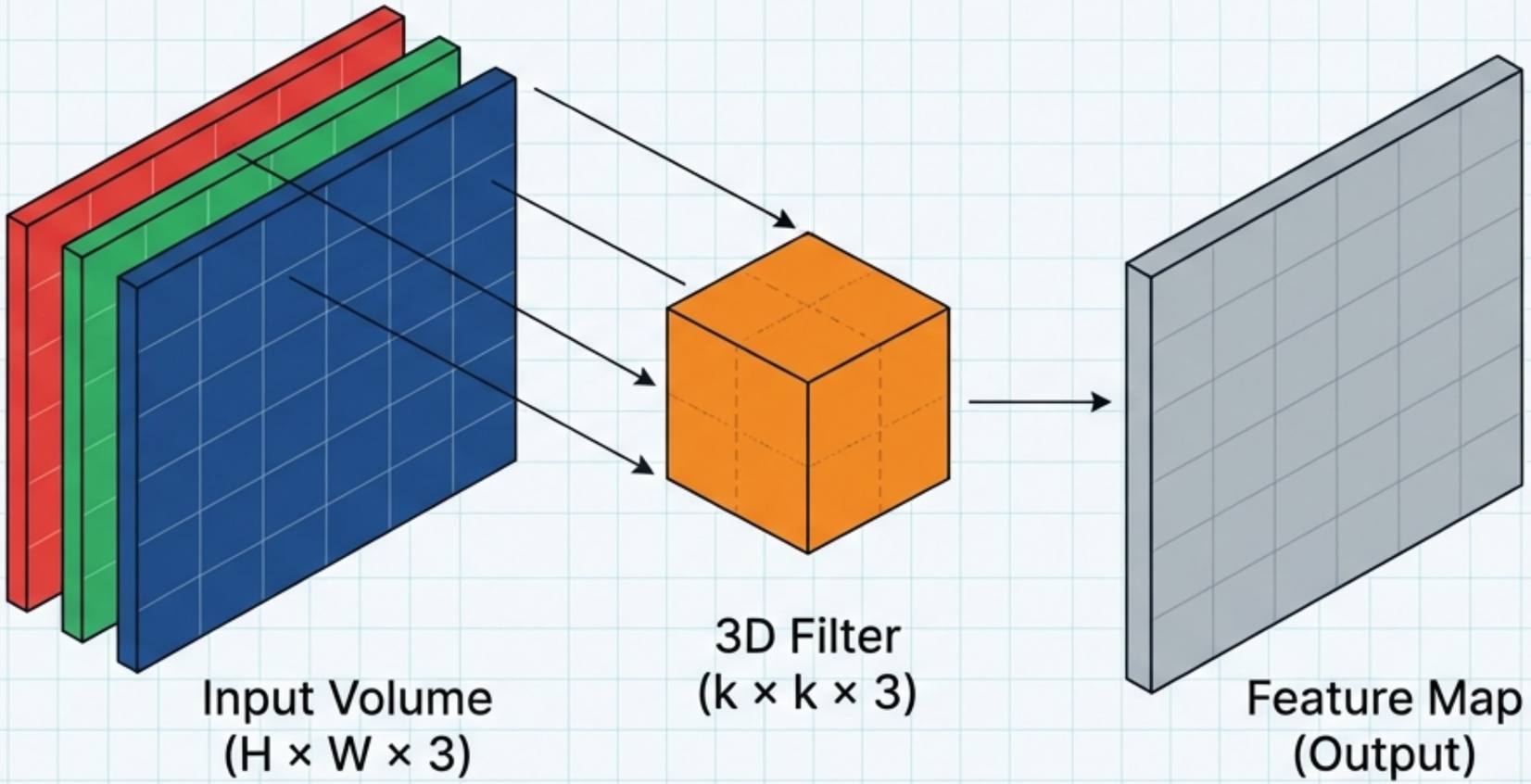
Worked Example

Input: 32x32 image
Kernel: 5x5
Padding: 0
Stride: 1

Calculation:
`(32 + 0 - 5) / 1 + 1 = 28`

Result: Output Feature Map is 28x28

Seeing in Color: Multi-Channel Inputs



The kernel processes all color channels simultaneously to find features.

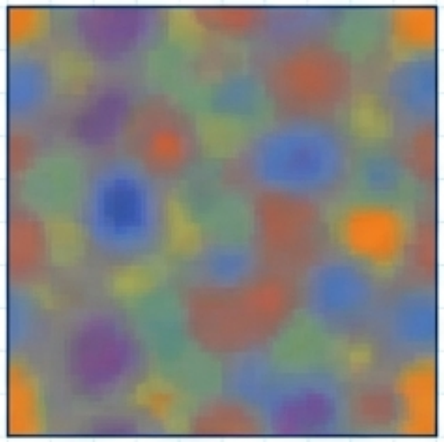
From Random Noise to Structured Signal

Learning

Epoch 0 (Initialization)

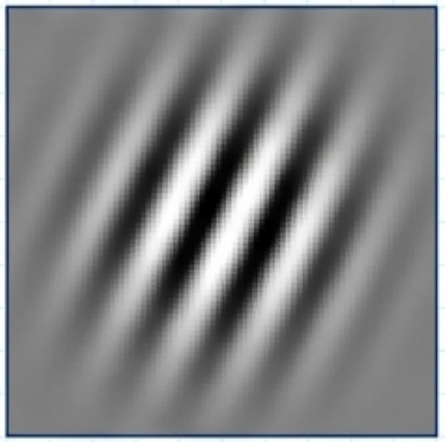
NT = P(Random value)

Epoch 10



JetBrains Mono

Epoch 100 (Trained)



data 0.005

Forward Pass (Guess) → **Loss Function** (Error) → **Backpropagation** (Update Weights) → **Repeat**

We don’t design the kernels. The network learns the best patterns to minimize error.

Building it in Code (PyTorch)

```
class CNN_Model(nn.Module):
    def __init__(self):
        super(CNN_Model, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=6, kernel_size=5)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

Model = CNN_Model().to(device)
```

The Feature Extractor

The Downsampler

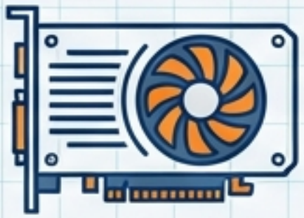
The Classifier

Critical Analysis: Strengths & Limitations

The Good

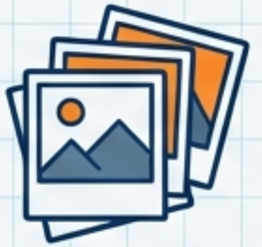
Spatial Invariance
(Recognizes objects anywhere)

Parameter Efficiency
(Shared weights)



GPU Acceleration
(Matrix Math Friendly)

The Bad



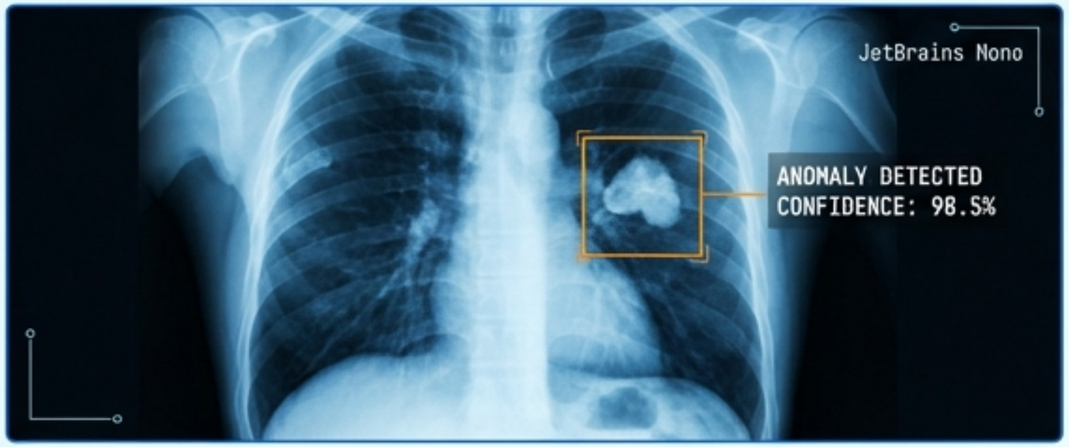
Data Hungry
(Needs massive datasets)

Computationally Expensive

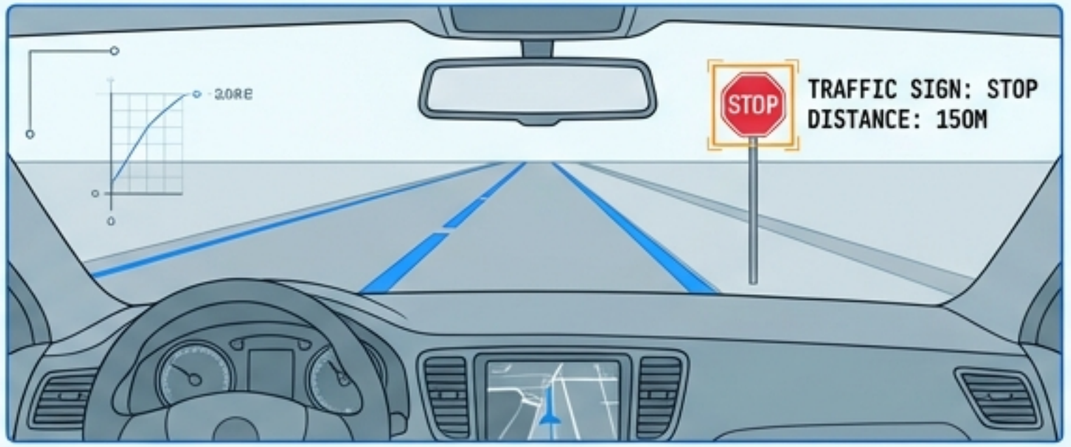
Context Blindness
(Struggles with long text sequences/NLP)

Beyond Cats and Dogs: Real-World Use Cases

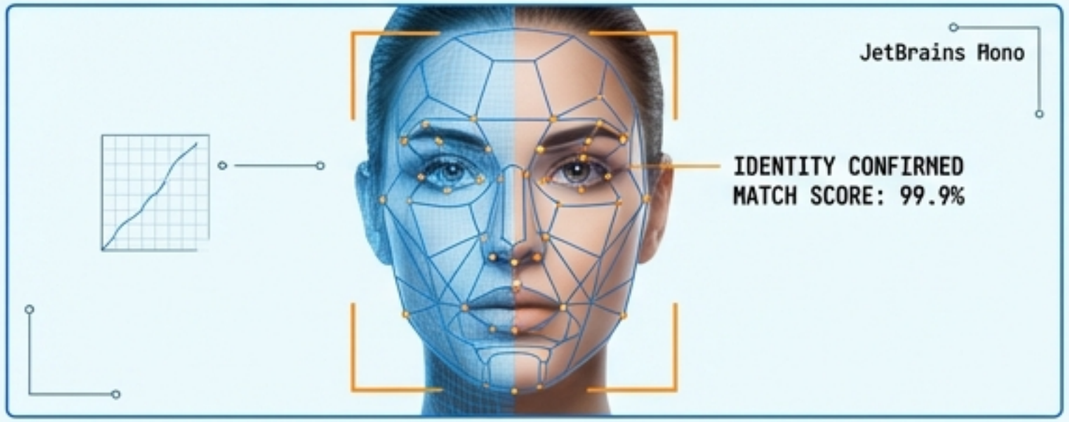
Tumor Detection



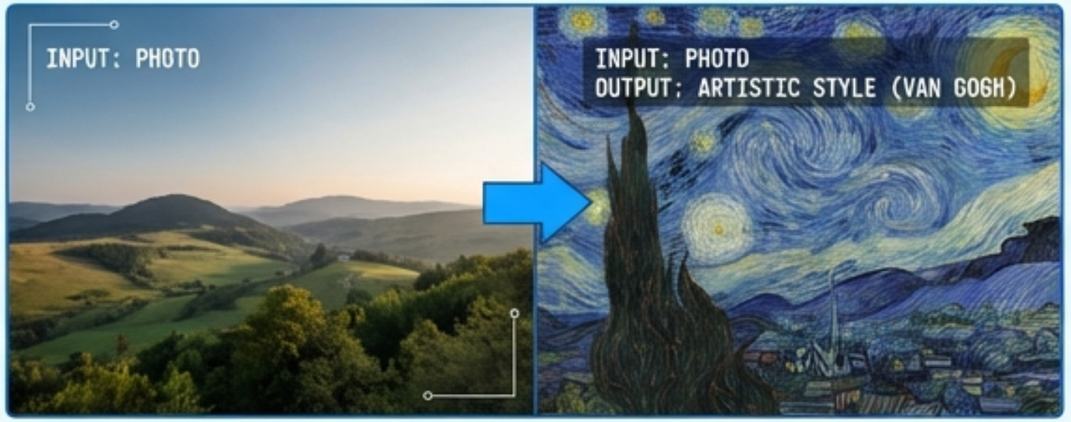
Autonomous Driving



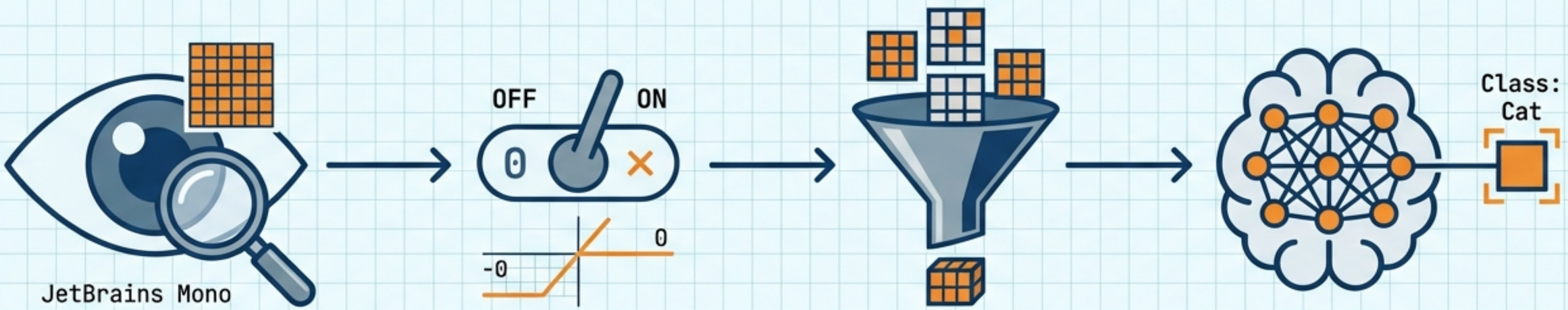
Facial Recognition



Style Transfer & Generative Art



The Takeaway: The Digital Eye



Convolution: Scanning

ReLU: Deciding

Pooling: Summarizing

Fully Connected: Classifying

"CNNs revolutionized AI by accepting the structure of the world as it is—grids of pixels—rather than forcing it into lines of numbers."