جامعة الملك خالد
KING KHALID UNIVERSITY

وزارة الاتصالات وتقنية المعلومات
MINISTRY OF COMMUNICATIONS AND INFORMATION TECHNOLOGY

أكاديمية كاوست
KAUST ACADEMY

KAUST Academy & Tech Camp AI Week

Presented By: Ali Alqutayfi & Hassain Alsayhah

Linear Regression

Logistic Regression

Neural Networks

Deep Learning

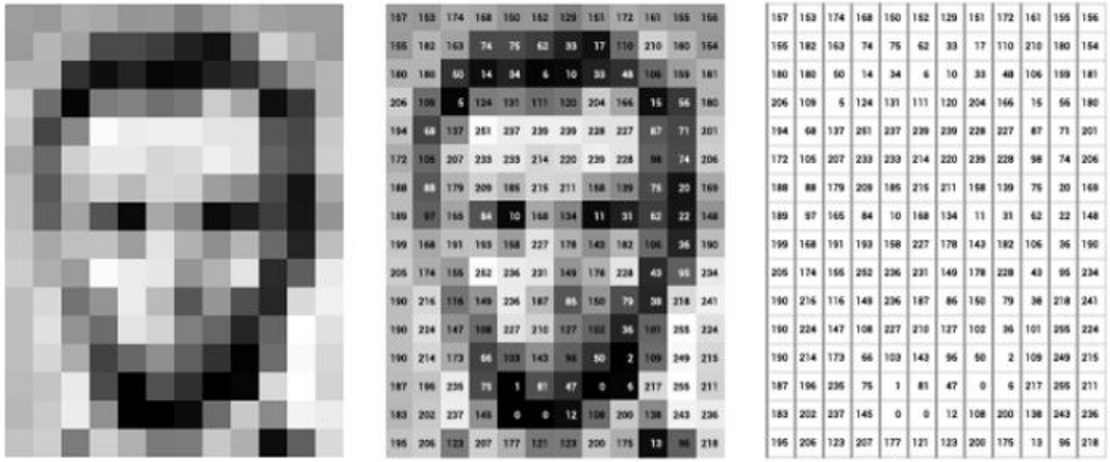
Artificial Intelligence and Machine Learning
Convolutional Neural Networks CNNs

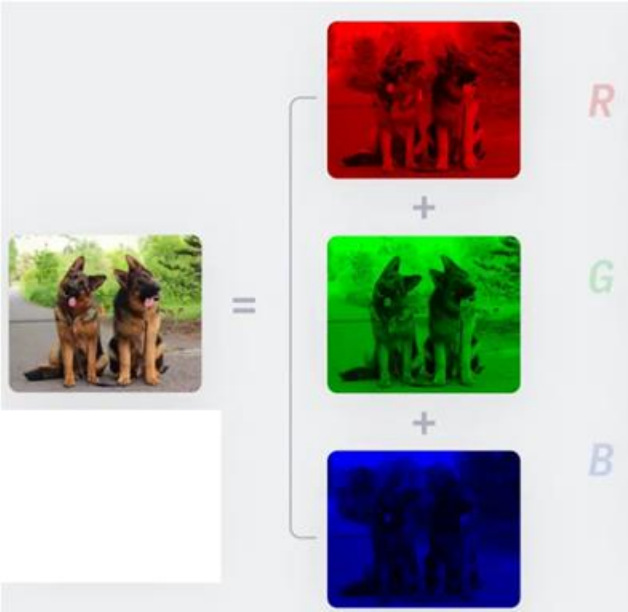
Lecture Outline

- Understand how images are represented and processed in computers
- Explain the fundamental building blocks of Convolutional Neural Networks (CNNs)
- Learn how CNNs solve image classification problems
- Understand key CNN architectures and their innovations

How Computers See Images

- **Grayscale images**: Single channel matrix with values in [0,255]
- **RGB images**: Three channels (Red, Green, Blue) with values in [0,255]
- Each pixel becomes a feature for processing





Why Not Fully Connected Networks?

Problems with Traditional Neural Networks for Images

1. Massive Parameter Count

- For a 32×32 RGB image: 32×32×3 = 3,072 input features
- First hidden layer with 1000 neurons: 3,072,000 parameters
- Becomes computationally expensive quickly

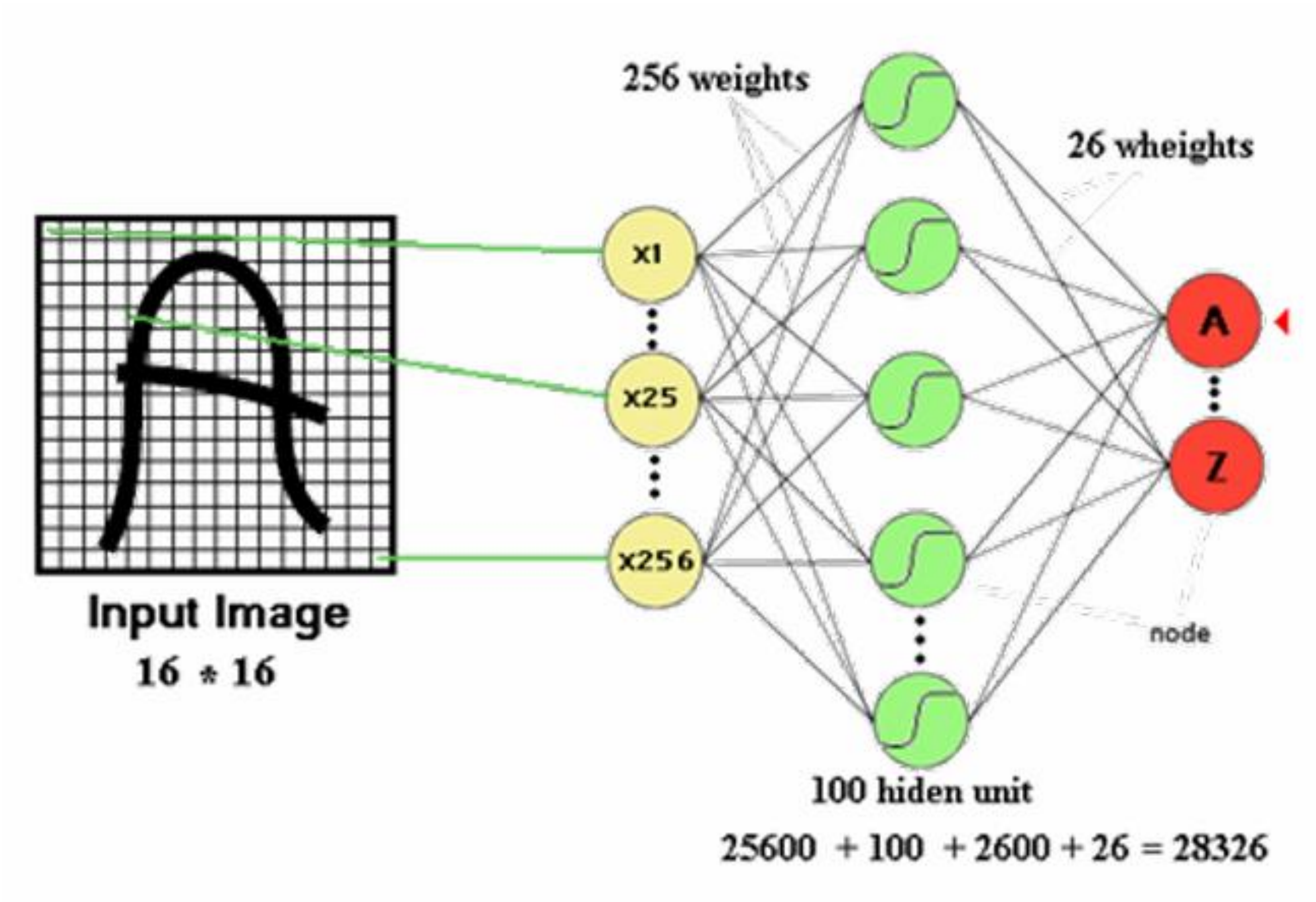
2. No Spatial Awareness

- Treats each pixel independently
- Ignores spatial relationships between nearby pixels
- Cannot recognize patterns like edges or shapes

3. No Translation Invariance

- Cannot recognize the same object if it's moved to a different position
- A shifted image is treated as completely different

Why Not Fully Connected Networks?



Why Not Fully Connected Networks?

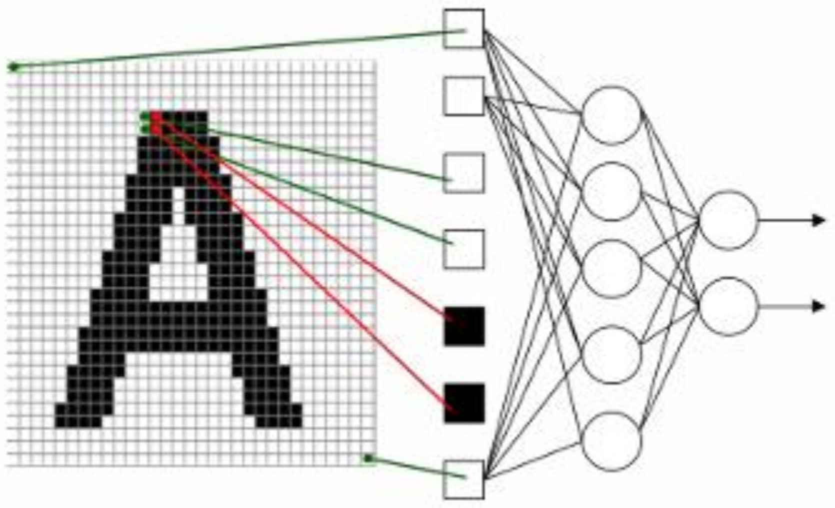


Figure 2: Original "A" character.

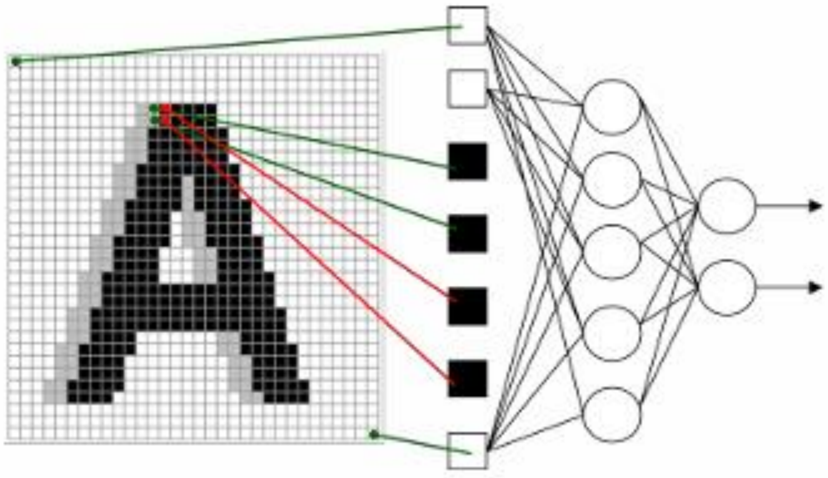


Figure 3: Shifted "A" character.

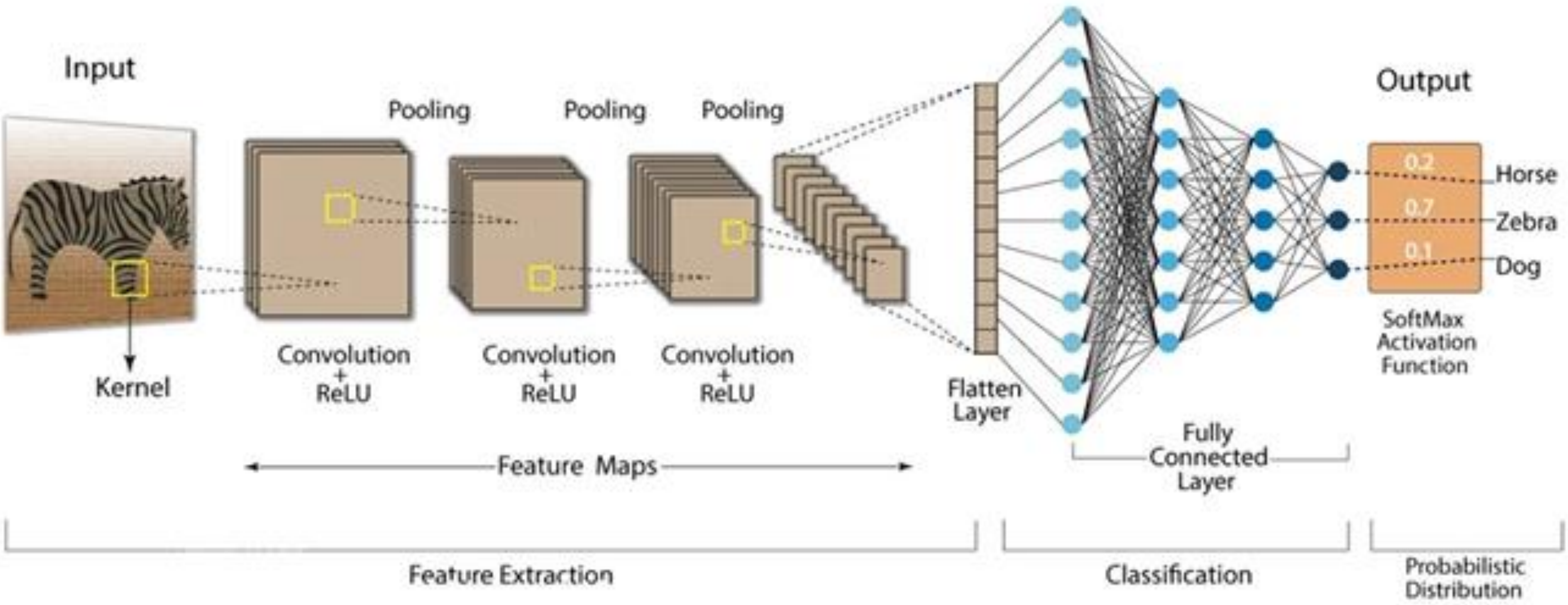
Why Not Fully Connected Networks?

Enter Convolutional Neural Networks (CNNs)

What CNNs Solve

- **Find Patterns**: Detect local features like edges, textures, shapes
- **Work Efficiently**: Share parameters across spatial locations
- **Handle Variations**: Recognize objects regardless of position

Enter Convolutional Neural Networks (CNNs)



1. Convolution Layer

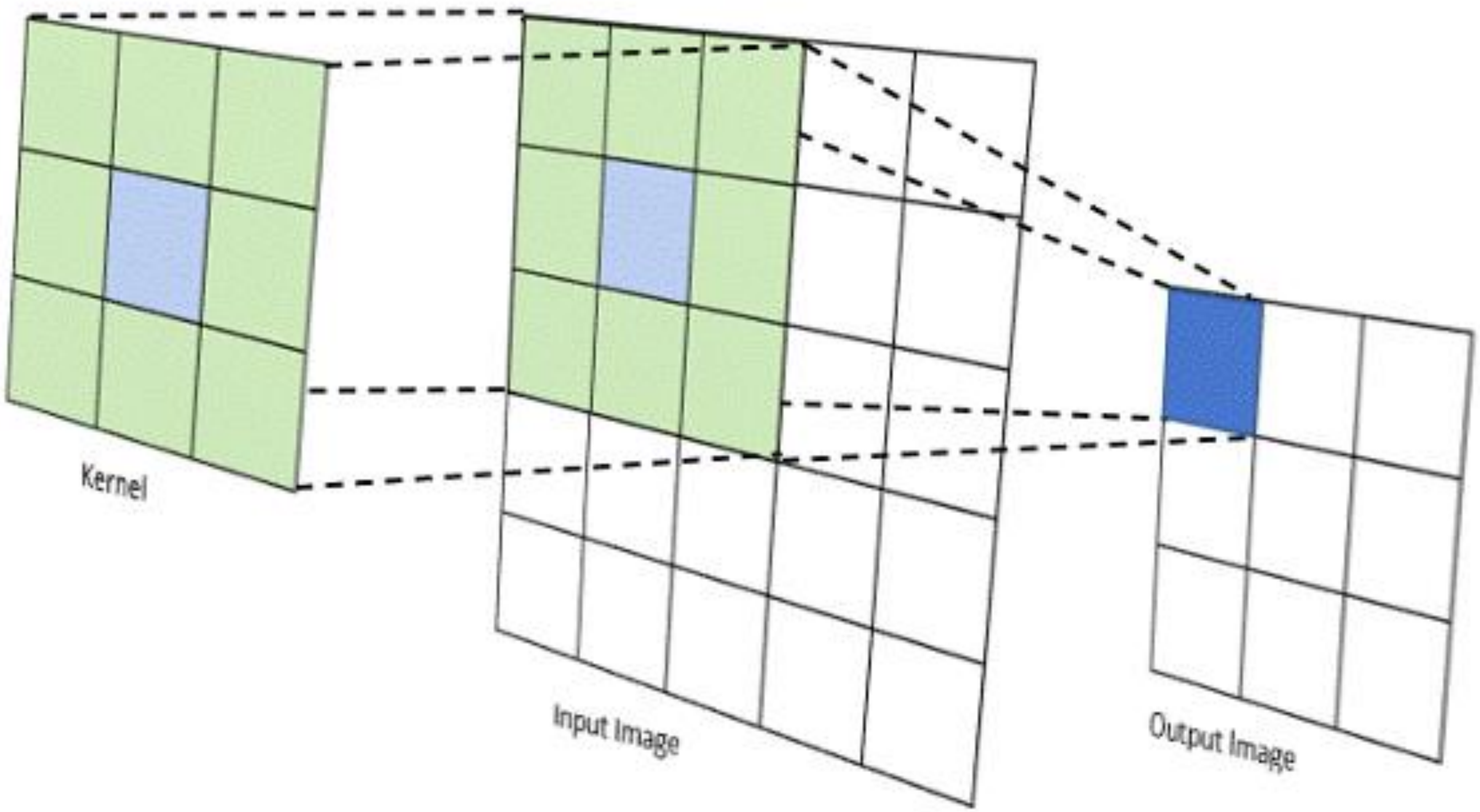
How Convolution Works:

- Small filter (kernel) slides across the image
- Computes dot product at each position
- Creates feature map showing where patterns are detected

Mathematical Formula:

$z = W * x = \Sigma\, \Sigma\, W[a,b] \times X[i+a, j+b]$

1. Convolution Layer



2. Controlling Convolution

Padding: Add zeros around image borders

- **Same Padding**: Output size = Input size
- **Valid Padding**: No padding, output size shrinks

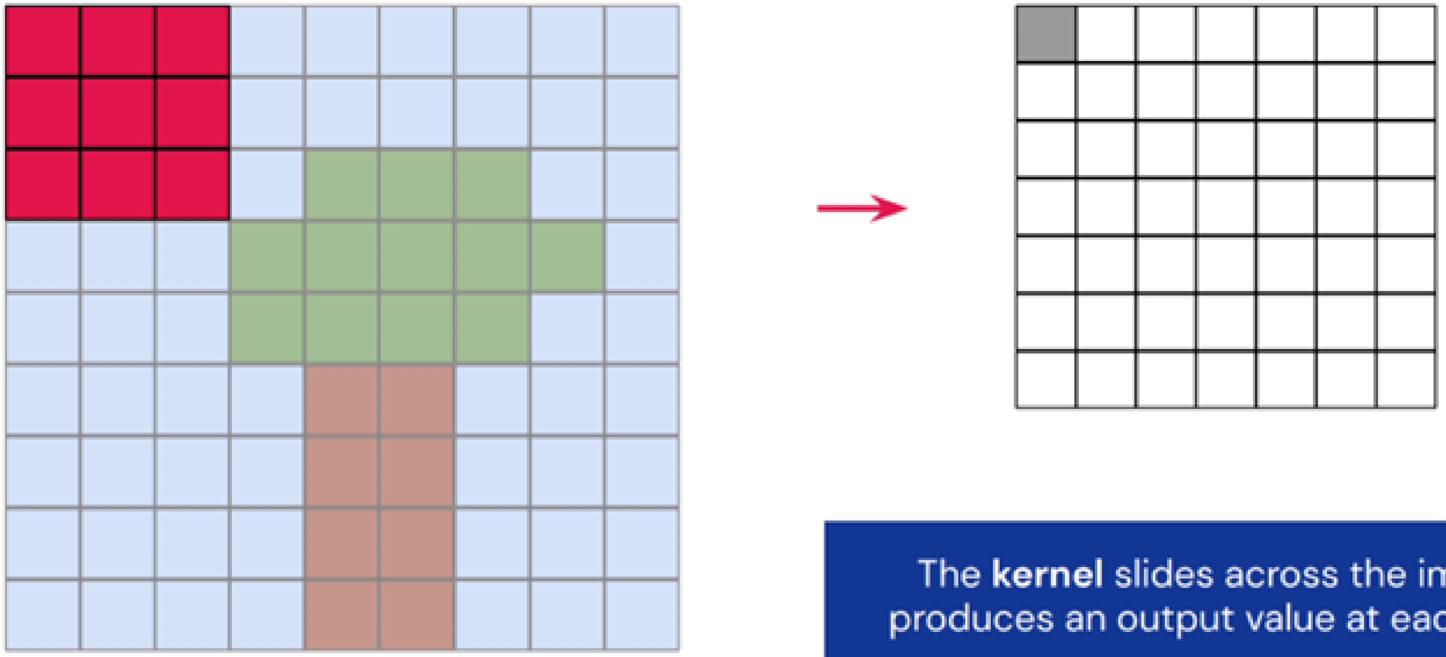
Stride: How many pixels the filter moves each step

- Stride = 1: Move one pixel at a time
- Stride > 1: Skip pixels, reduces output size

Output Size Formula:

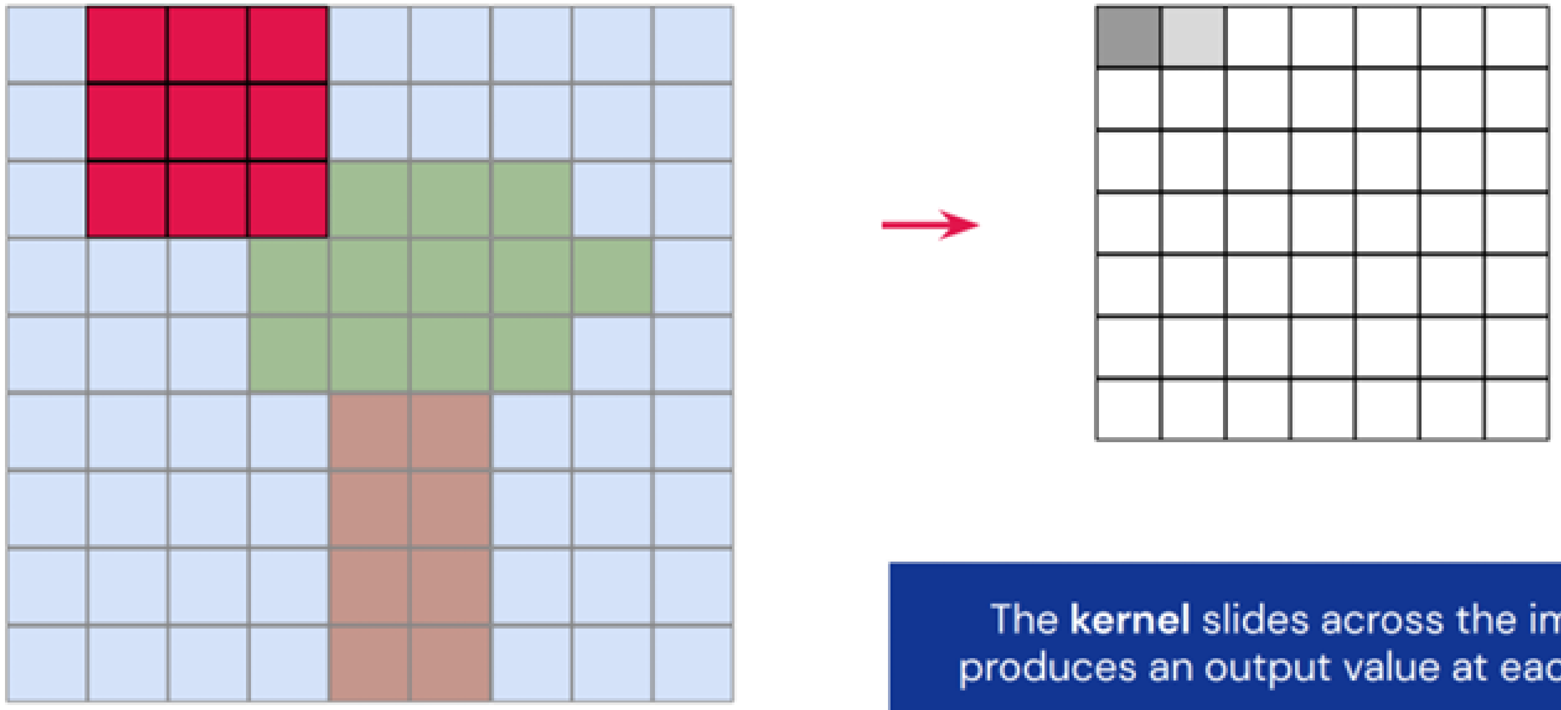
Output Size = floor(Input Size + 2×Padding - Kernel Size) / Stride + 1

2. Controlling Convolution



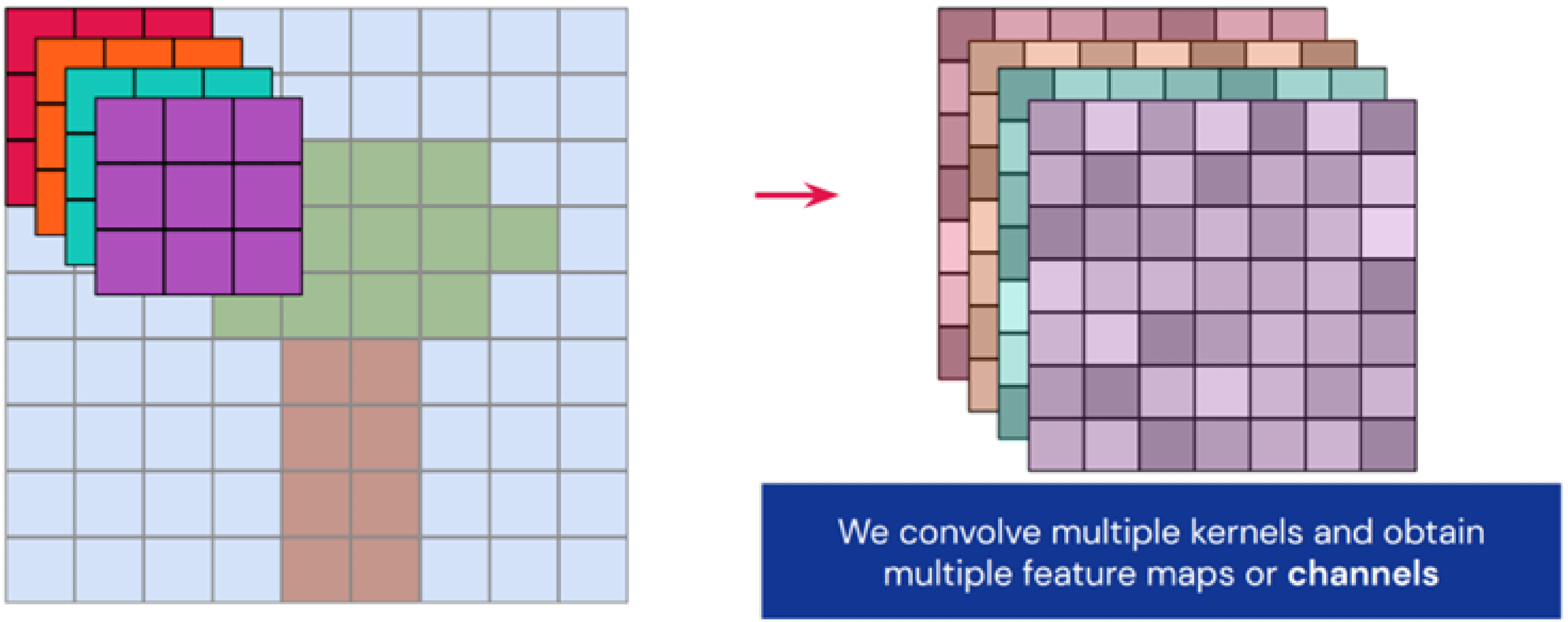
The **kernel** slides across the image and produces an output value at each position

2. Controlling Convolution



The **kernel** slides across the image and produces an output value at each position

2. Controlling Convolution



How Convolution Works?

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

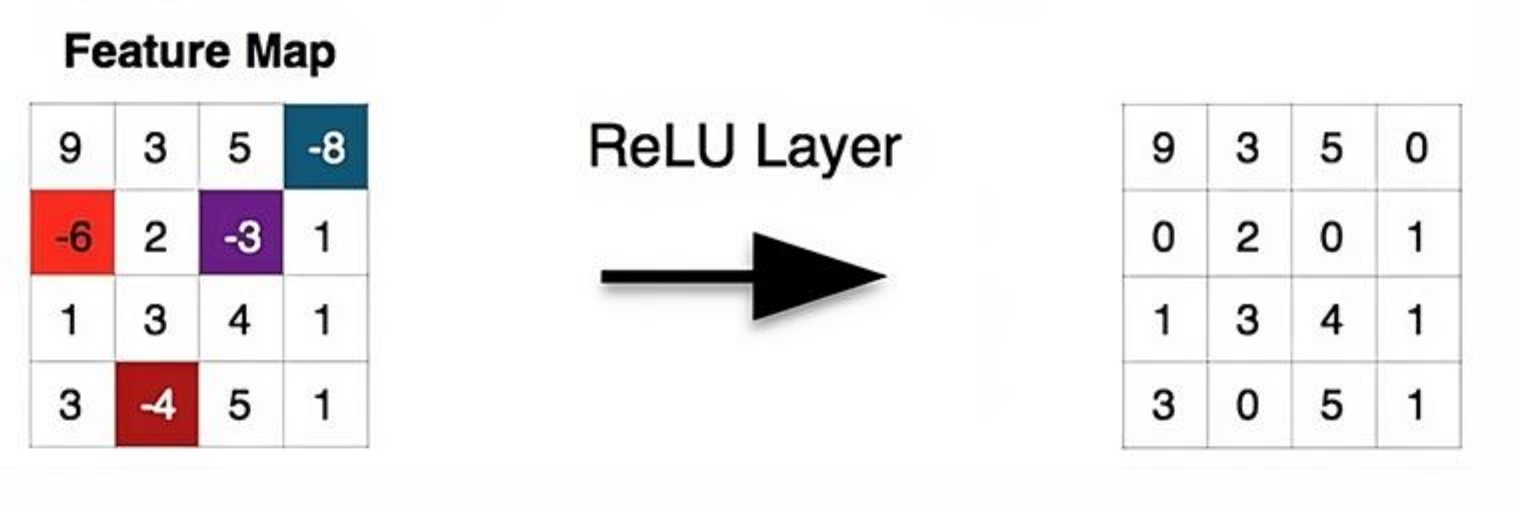
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3. Activation Functions

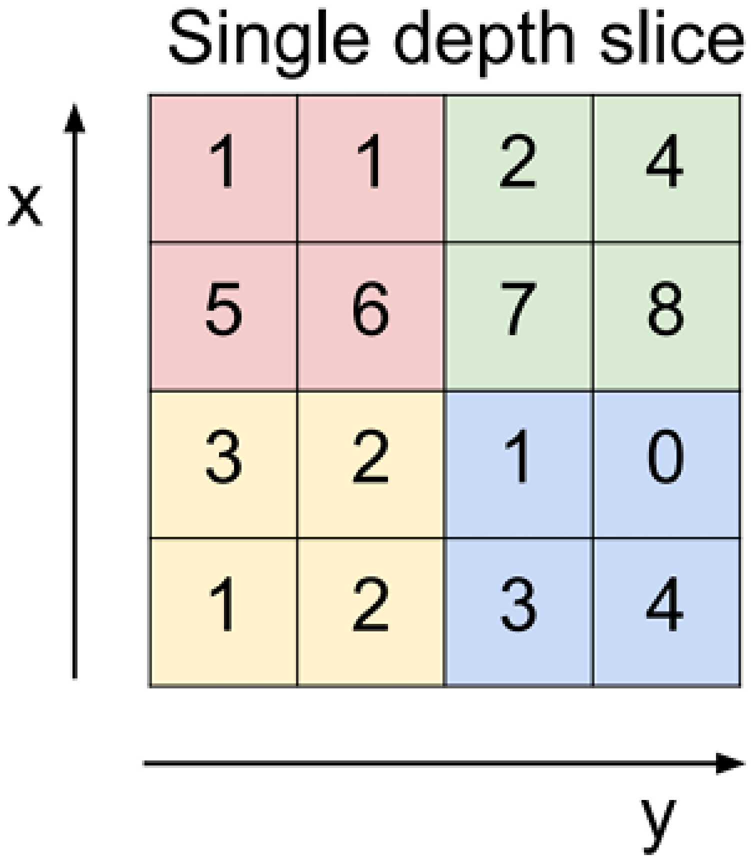
- **ReLU**: σ(x) = max(0,x)
- Adds non-linearity to enable learning complex patterns
- Applied after each convolution

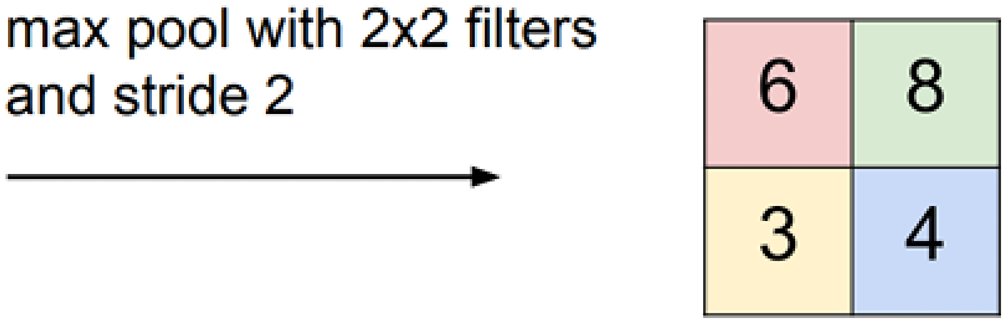


4. Pooling Layers

- **Max Pooling**: Take maximum value in each window
- **Average Pooling**: Take average value in each window
- **Purpose**: Reduce spatial dimensions, increase robustness

4. Pooling Layers





- No learnable parameters
- Introduces spatial invariance

CNN Output Size

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in} : number of input features

n_{out} : number of output features

k : convolution kernel size

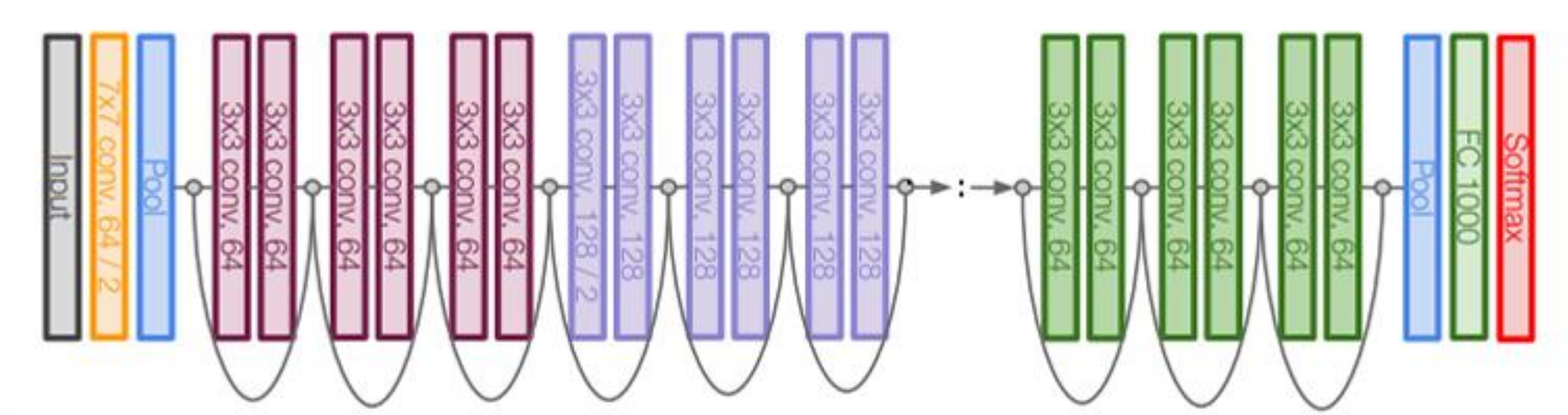
p : convolution padding size

s : convolution stride size

Most Notable CNN-based Architectures

- **AlexNet [Krizhevsky et al. 2012]**: The first CNN to achieve breakthrough performance on image classification.
- **VGGNet [Simonyan and Zisserman, 2014]**: Used very deep networks (up to 19 layers).
- **InceptionNet (GoogLeNet) [Szegedy et al., 2014]**: Used multiple filter sizes per layer (Inception modules).
- **ResNet [He et al., 2015]**: Introduced skip connections for training very deep networks.
- **EfficientNet [Tan and Le, 2019]**: Found a scaling method that simultaneously scales a CNN's depth, width, and resolution optimally using a single scaling coefficient.

ResNet



Input
7x7 conv, 64 / 2
Pool
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128 / 2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
FC 1000
Softmax

Let’s Code

- Understand Pytorch Basics
- Implement CNNs in Pytorch
- Implement Transfer Learning

Homework

- Understand Pytorch Basics
- Implement CNNs in Pytorch for CIFAR10 Dataset
- Implement Transfer Learning for CIFAR10 Dataset