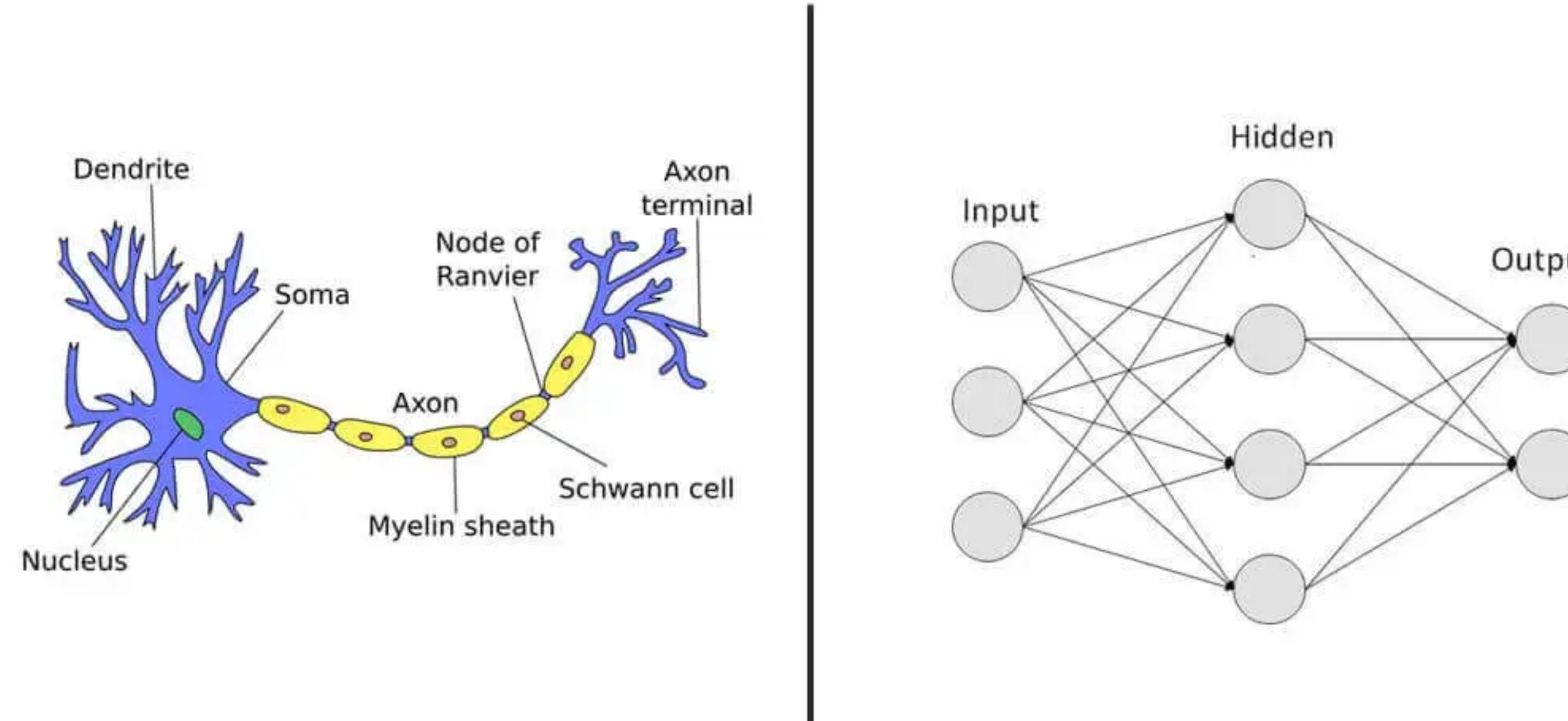


Convolutional Neural Networks (CNN)



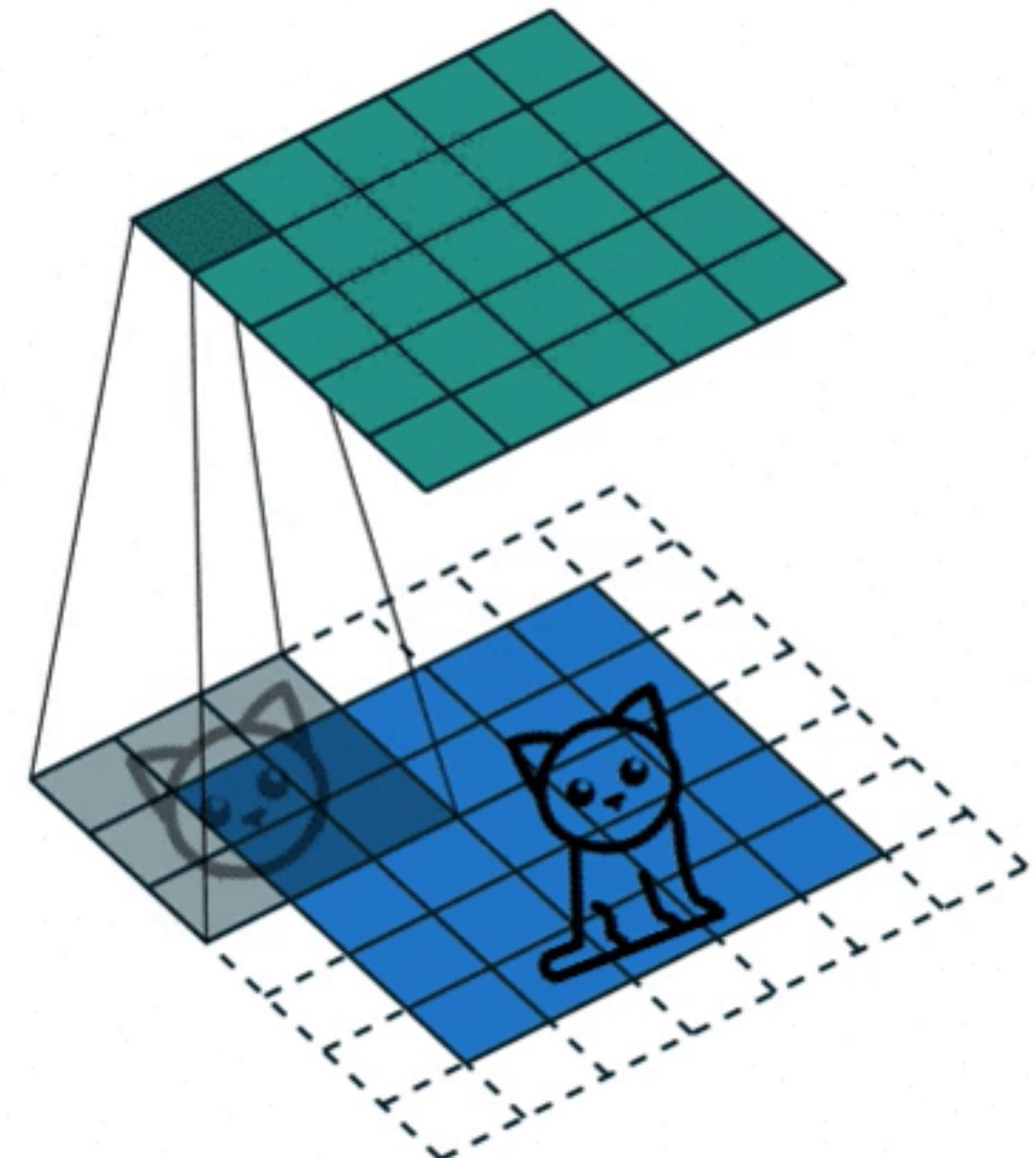
Sharing Session - [@SoramitsuKhmer](#)

Goals

- Familiar with CNN terminology (know the keywords)
- Learn CNN without diving deep into the math
- Be able to implement basic CNN model

Agenda

1. Background
2. What is Neural Network?
3. What is CNN?
4. CNN Components
5. Coding Session



Background

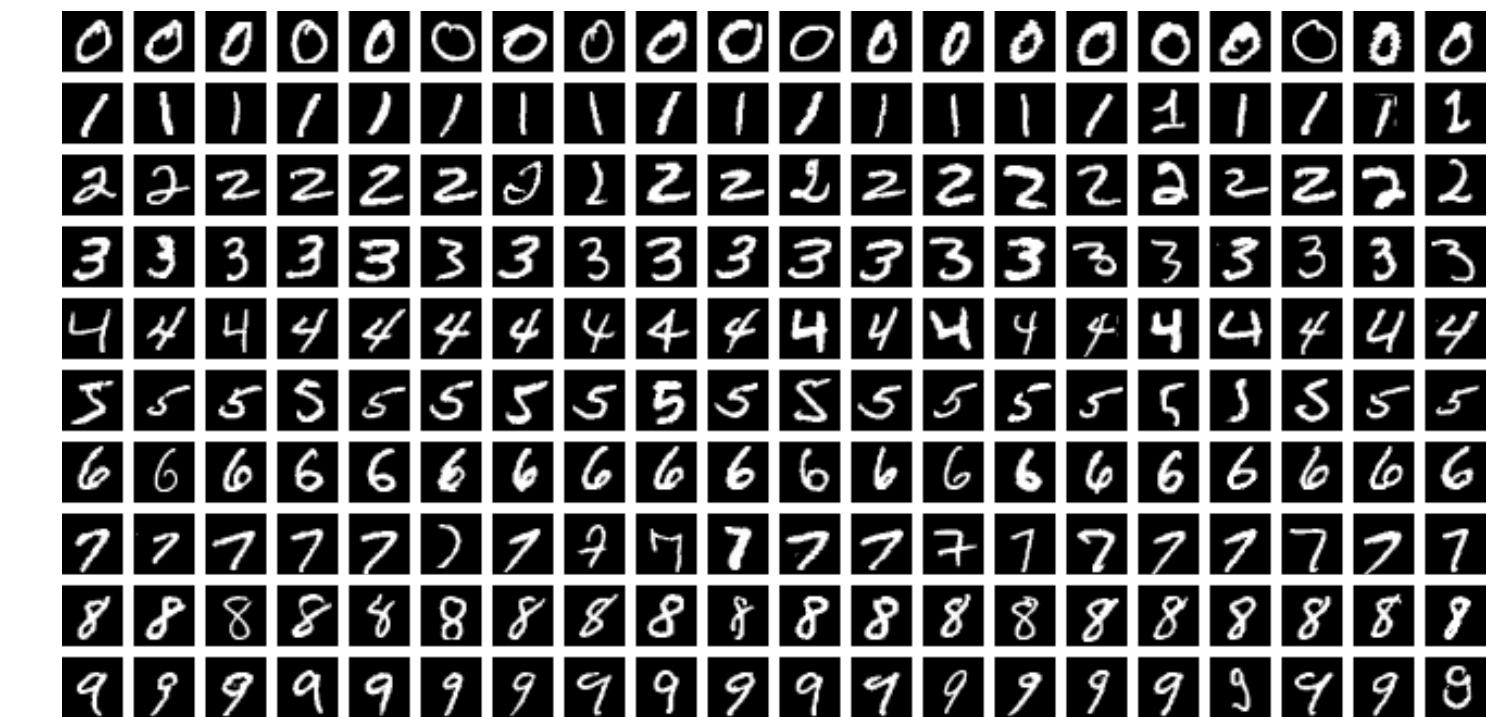
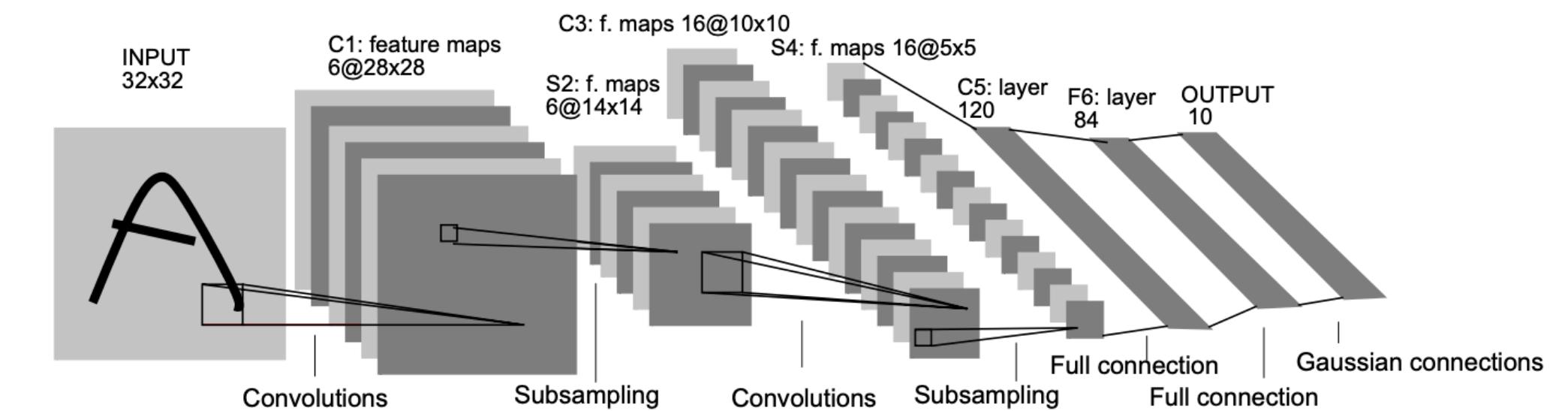
Origins of Neural Networks

- **1958:** The **Perceptron**, developed by **Frank Rosenblatt**, was an early neural network model capable of learning simple binary classifiers. Due to its limited capacity to learn complex patterns and its inability to handle non-linearly separable data, led to the decline of research on perceptrons in the 1960s and 1970s.
- **1986:** **Backpropagation** was popularized by **Rumelhart**, **Hinton**, and **Williams**, enabling multi-layer perceptrons (MLPs) to be trained more effectively. MLPs renewed interest in artificial neural networks and sparked a new era of research and innovation in machine learning.

Background

Birth of CNNs: LeNet-5

- In the late 1980s and 1990s, **Yann LeCun** introduced **LeNet-5**, one of the first convolutional neural networks.
- Used by the US Postal Service to read handwritten ZIP codes.
- Introduced weight sharing and local receptive fields, drastically reducing model complexity for image inputs.



Background

Key Milestone: AlexNet (2012)

- **AlexNet** created by **Alex Krizhevsky**, with **Ilya Sutskever** and **Geoffrey Hinton** at the University of Toronto.
- Trained on **ImageNet** — a massive dataset with 1.2 million labeled images across 1000 categories.
- Utilized GPUs (**NVIDIA GTX 580**) to massively accelerate training.
- Introduced **ReLU**, **Dropout**, and **Data Augmentation** in a deep CNN architecture.
- Published as “**ImageNet Classification with Deep Convolutional Neural Networks**”.
- **AlexNet** wasn't the first CNN, but it was the first to show deep CNNs can scale to real-world, large datasets and crush traditional methods. It marked the start of the **deep learning** era we're in today.

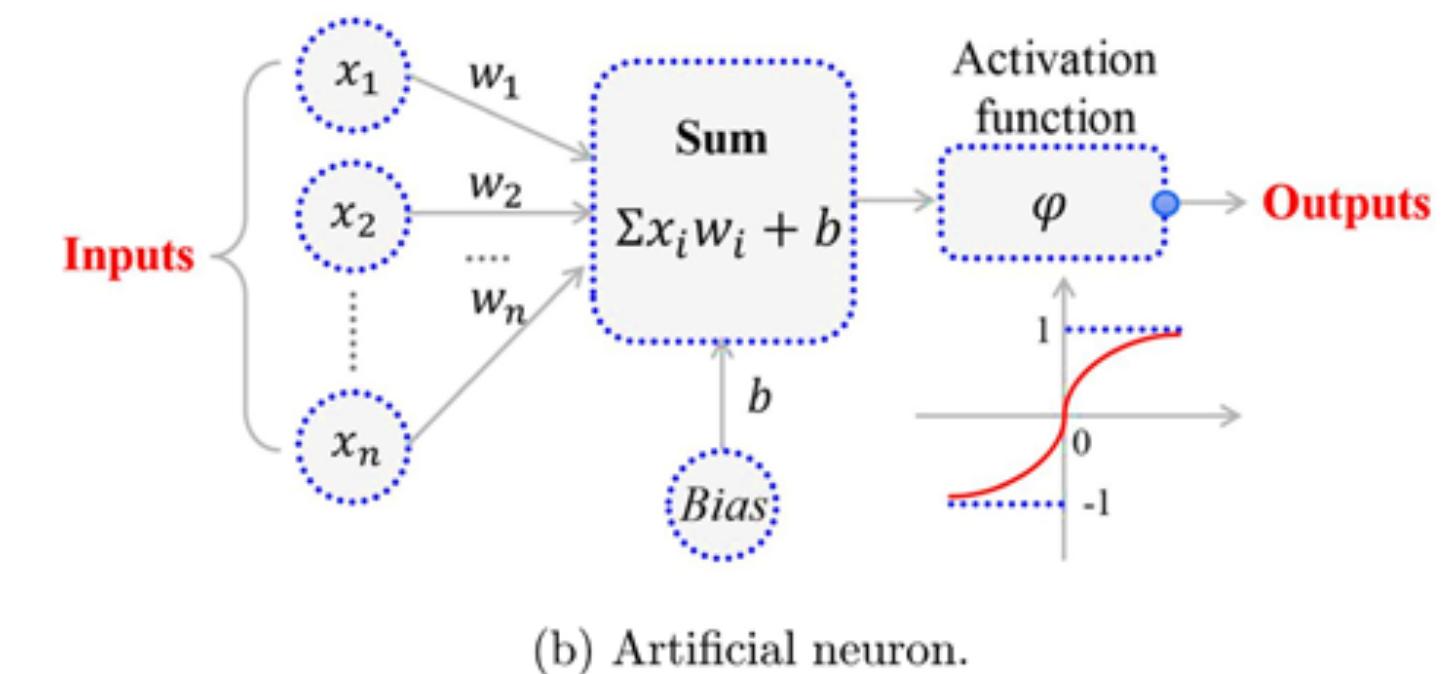
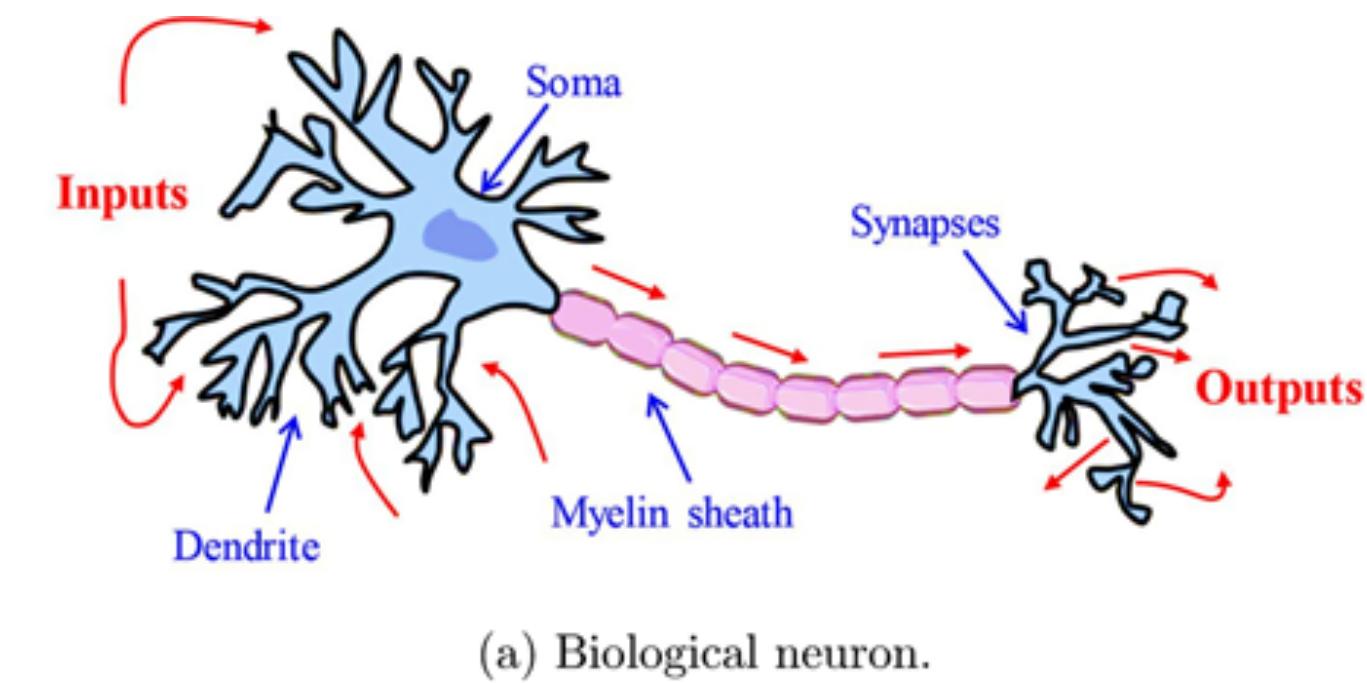
Background

Summary Timeline

Year	Milestone	Contribution
1958	Perceptron	First artificial neuron model
1986	Backpropagation	Multi-layer training becomes feasible
1998	LeNet-5	First CNN for image classification
2012	AlexNet	CNN breakthrough on large-scale datasets

What is Neural Network?

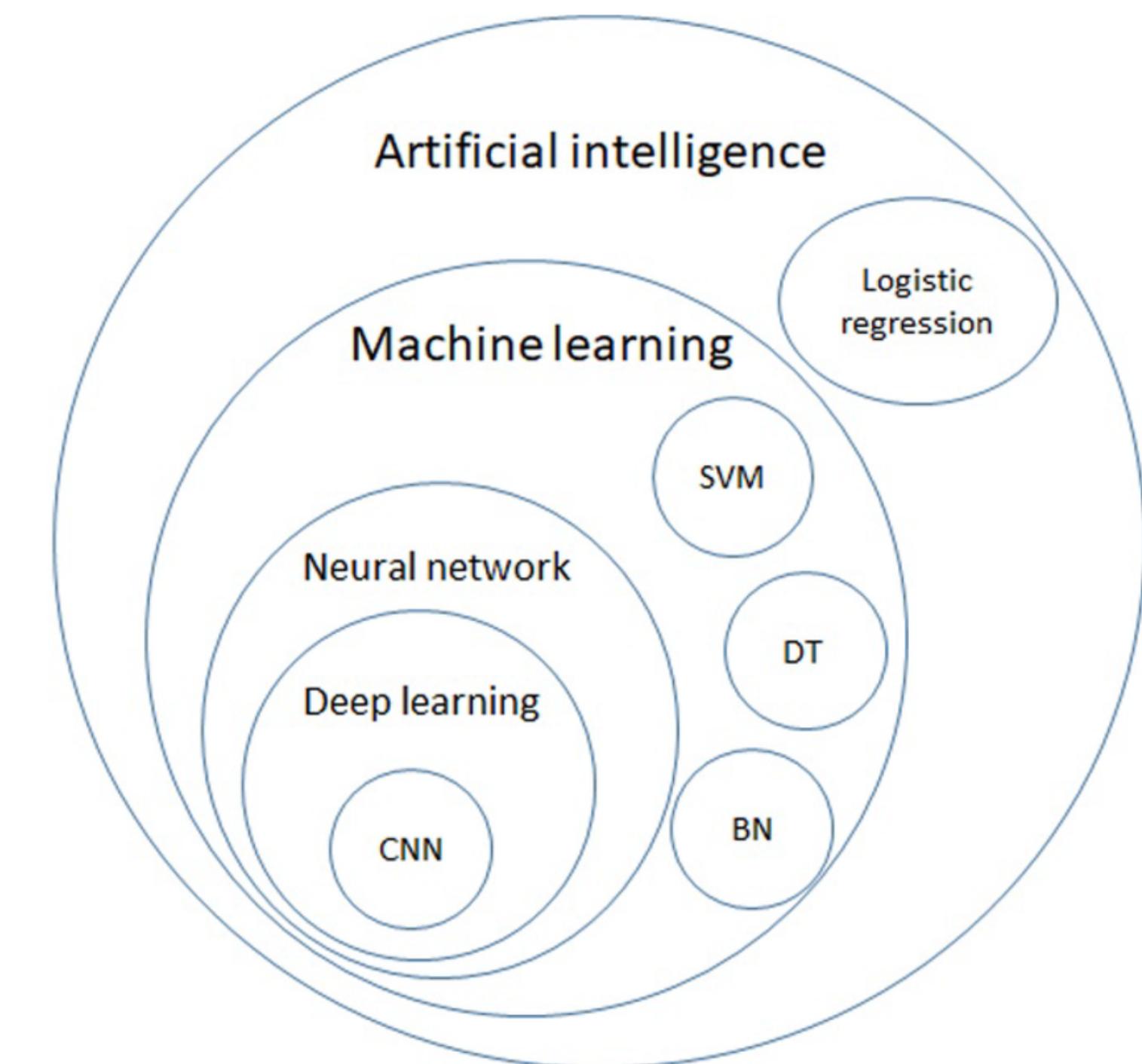
- A neural network is a type of machine learning model inspired by the way the human brain works.
- **Neurons:** These are like tiny processors that receive input, do some math (like summing and applying an activation function), and pass the result to the next layer.
- **Input Layer:** Takes in the raw data (like pixels in an image).
- **Hidden Layers:** Do the heavy lifting—extracting patterns, features, and relationships.
- **Output Layer:** Produces the final result (like predicting a number or a category).
- **Weights & Biases:** Each connection between neurons has a weight (importance), and each neuron has a bias (a tweak factor). These are adjusted during training to improve accuracy.



- Neural networks learn by adjusting weights and biases based on how wrong they are, using a process called **backpropagation** with **gradient descent**.

What is CNN?

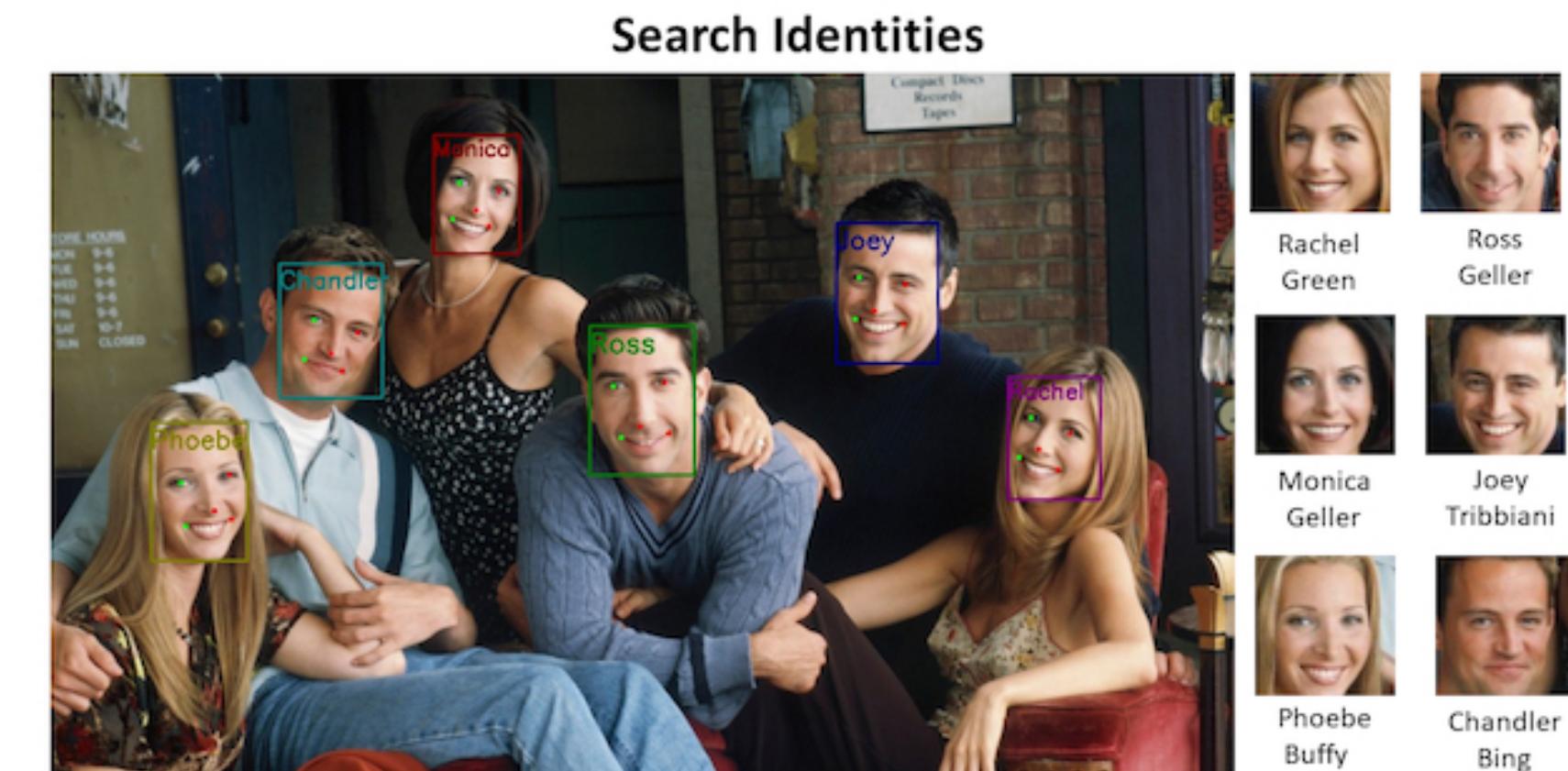
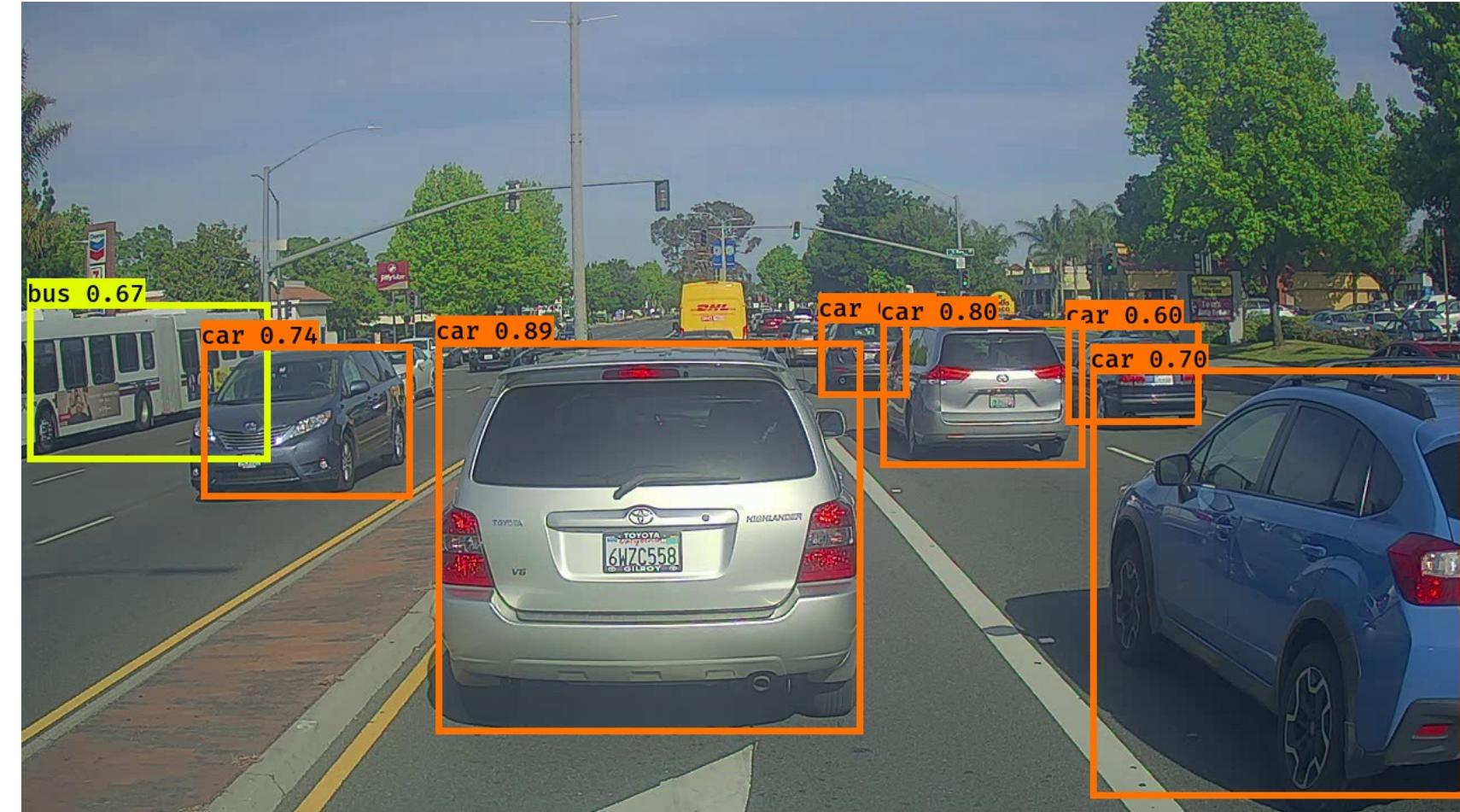
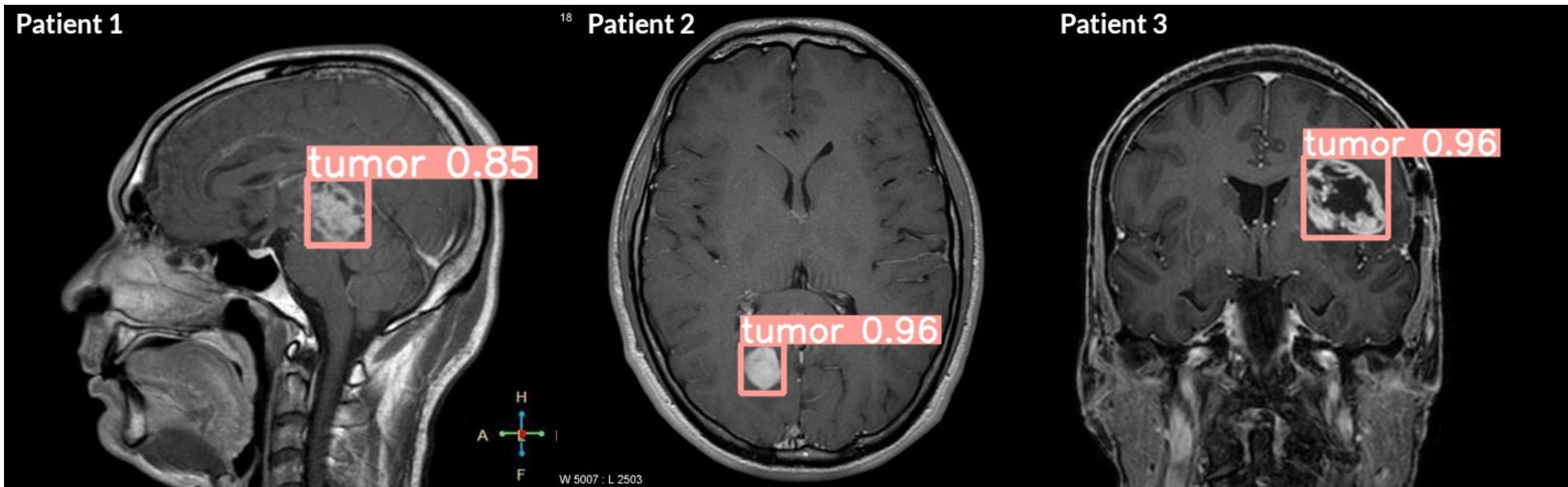
- A **CNN** is a type of deep learning algorithm used primarily for image recognition, video analysis, natural language processing, and more.
- It's designed to automatically and adaptively learn patterns in data using layers like:
 - **Convolutional layers** – for detecting features like edges, textures, shapes.
 - **Pooling layers** – for reducing dimensionality and keeping important information.
 - **Fully connected layers** – for making final predictions or classifications.



What is CNN?

CNNs in the Real World

- Medical imaging (tumor detection)
- Autonomous driving (object/lane recognition)
- Face recognition
- Image search and recommendation
- Social media (auto-tagging, moderation)



Why Use CNNs for Images?

- **Regular Neural Networks (NNs):**
 - Work fine for small data
 - Struggle with large images (millions of pixels)
 - Treat every pixel separately => Too many parameters
- **CNNs (Convolutional Neural Networks):**
 - Look at small areas of the image (like scanning)
 - Use same filter across the image (Weight sharing)
 - Understand spatial patterns (e.g., edges, shapes)
 - Need far fewer parameters than NNs

CNNs vs Regular Neural Networks (NNs)

 Feature	 Regular NNs	 CNNs (Convolutional NNs)
How they see images	Looks at all pixels separately	Looks at small parts at a time
Use of image structure	Doesn't know nearby pixels are related	Understands nearby pixels go together
Number of connections	A lot (millions!)	Much fewer (shared weights)
Learning speed	Slow	Faster
Best for	Simple data (numbers, text)	Images, faces, objects, etc.

CNN Components

1. Convolution Layer

- Applies filters to detect patterns (edges, textures)

2. ReLU Activation

- Introduces non-linearity: $\text{ReLU}(x) = \max(0, x)$

3. Pooling Layer

- Downsamples input (typically with Max Pooling)
- Reduces size, keeps essential features

4. Fully Connected Layer

- Final classification layer using dense neurons

A Convolution Operation

Source layer

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

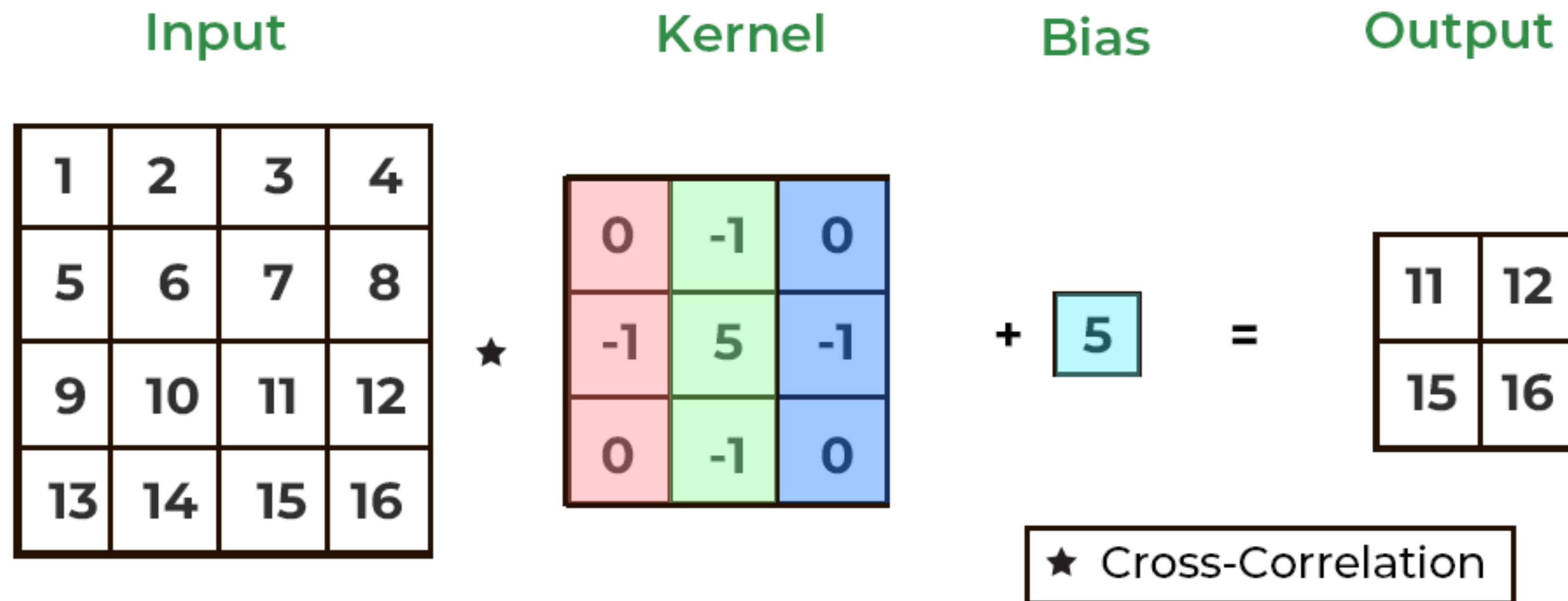
Convolutional kernel

-1	0	1
2	1	2
1	-2	0

Destination layer

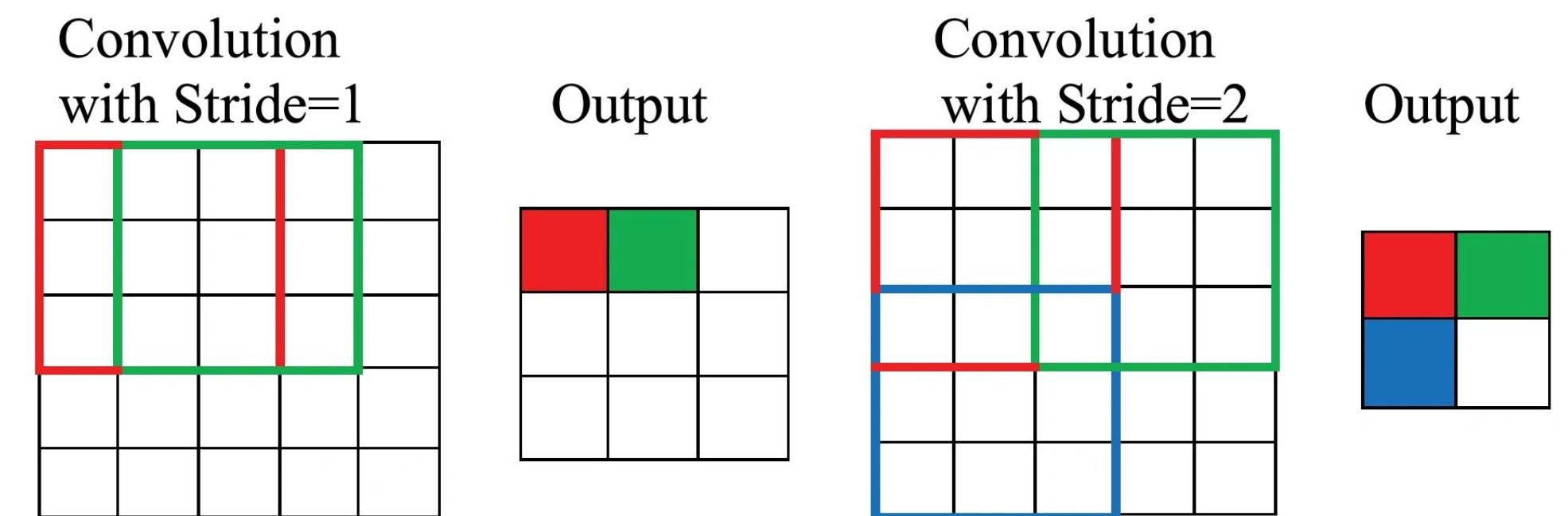
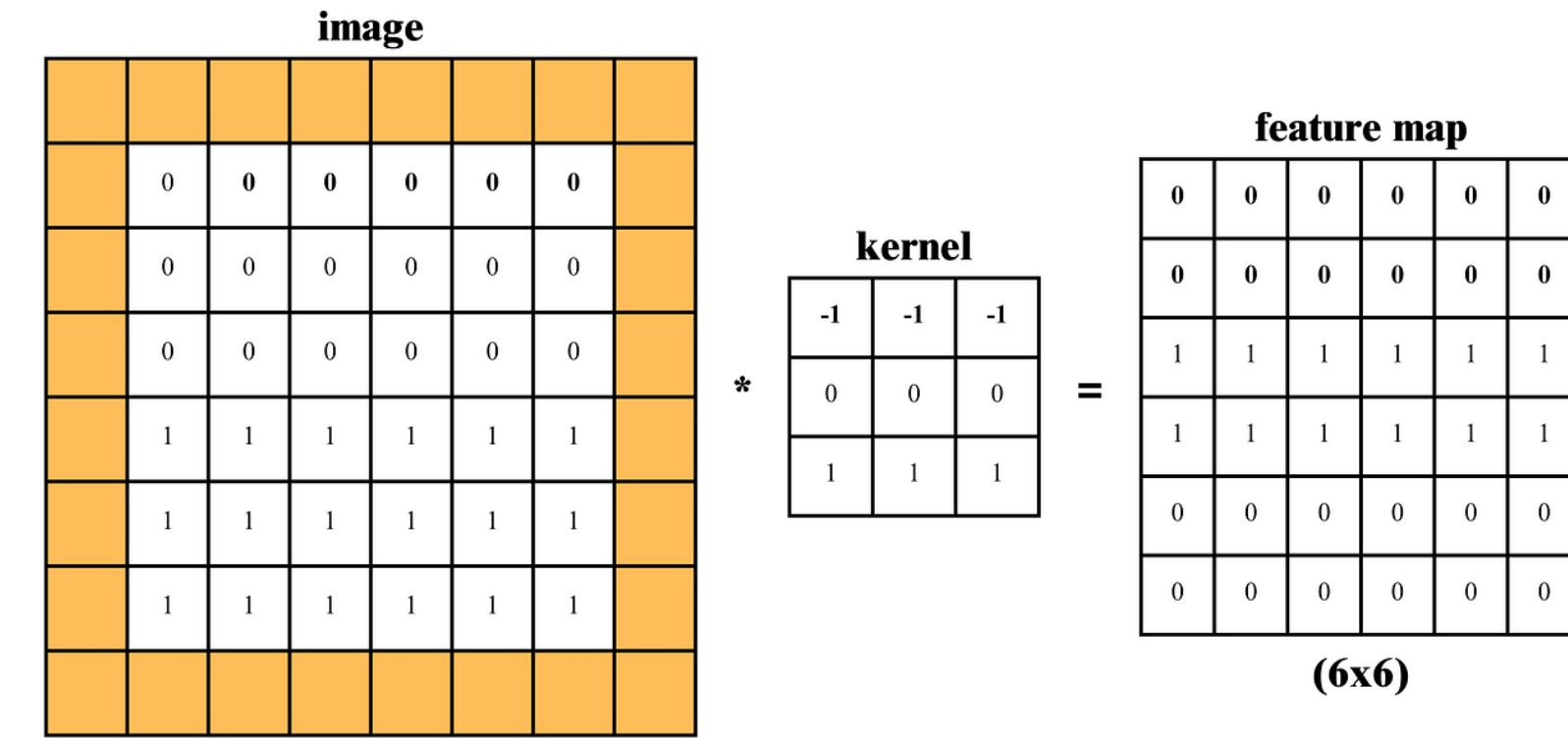
$$(-1 \times 5) + (0 \times 2) + (1 \times 6) + \\ (2 \times 4) + (1 \times 3) + (2 \times 4) + \\ (1 \times 3) + (-2 \times 9) + (0 \times 2) = 5$$

A Convolution Operation



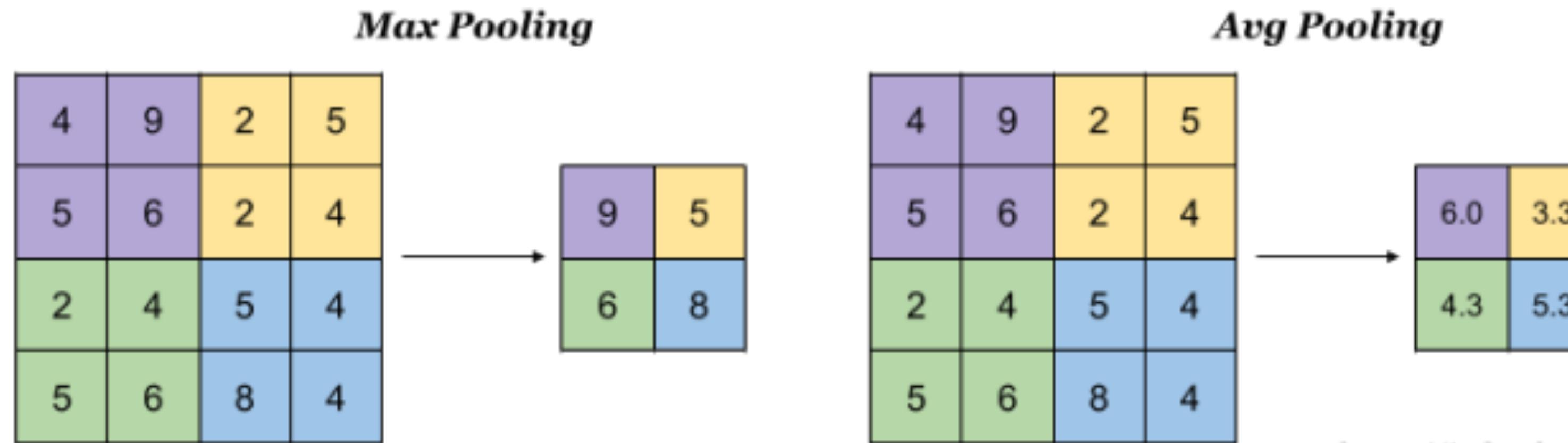
Padding and Strides in CNN

- In Convolutional Neural Networks (CNNs), **padding** and **strides** are important concepts that determine how the convolution operation is applied to an input, affecting the output size and how features are extracted.
- Padding** refers to the process of adding extra pixels (usually zeros) around the border of the input image.
- Strides** refer to the number of pixels by which the filter (or kernel) moves or slides across the input image.



Pooling Layer

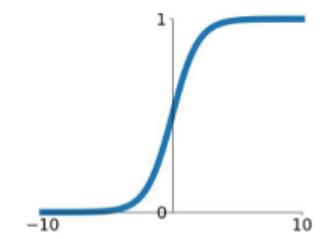
- **Pooling layer** is used to reduce the size of the representations and to speed up calculations, as well as to make some of the features it detects a bit more robust.



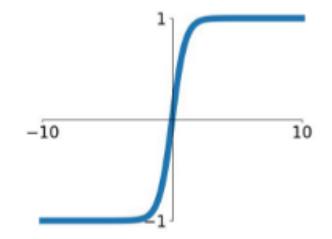
Activation Functions

- An activation function is a mathematical function applied to the output of a neuron. It introduces non-linearity into the model.
- Without activation functions, CNNs would be like a bunch of multiplication steps with no real "decision-making" ability.
- Activation functions help CNNs:
 - Detect non-linear patterns
 - Separate different classes better
 - Learn hierarchical features (edges, textures, shapes, objects)

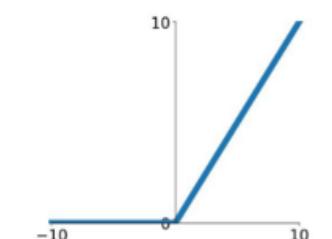
Sigmoid
 $\sigma(x) = \frac{1}{1+e^{-x}}$



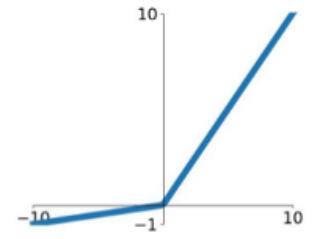
tanh
 $\tanh(x)$



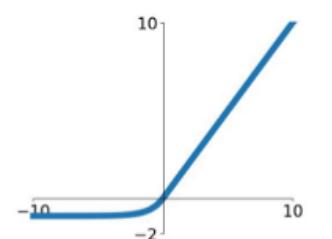
ReLU
 $\max(0, x)$



Leaky ReLU
 $\max(0.1x, x)$



Maxout
 $\max(w_1^T x + b_1, w_2^T x + b_2)$



ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

Cost Function

- **Cost function** (also called a **loss function**) is a mathematical function that measures how well the CNN's predictions match the actual target values (i.e., the ground truth).
- It is used during training to guide the optimization process by providing feedback on how to adjust the model's parameters (weights and biases) to minimize errors.
- Purpose of the Cost Function
 - **Quantifies Error:** Tells us how far off the predictions are from the actual values.
 - **Optimization Target:** The goal of training is to minimize this cost.
 - **Backpropagation:** It's used to calculate gradients, which help update weights during backpropagation.

CNN as a Student

Activation Function = How the Student Thinks

- Think of the **activation function** as how the student **processes information** from a lesson.
 - A **linear** student just memorizes facts.
 - An **activated** student (using ReLU, sigmoid, etc.) can think critically, understand patterns, and make connections.

Why it's important:

- Without an activation function, the student can only learn in a straight line — simple input => simple output.
- With it, they can handle complexity, like recognizing faces or reading emotions in photos.

CNN as a Student

Loss Function = The Student's Grade

- The **loss function** is like the **score** the student gets on a test.
 - High loss?  They made lots of mistakes.
 - Low loss?  They're understanding the material.



Why it's important:

- The student needs feedback to know how well they're doing. The loss function tells them exactly where and how much they went wrong.

CNN as a Student

Optimization = The Study Plan

- **Optimizer** is the **study strategy** or **tutor** that helps the student improve based on their grades (**loss**).
 - If they keep getting a question wrong, the tutor adjusts how they study it.
 - Strategies like SGD or Adam are different tutoring methods — some learn steadily, some adapt fast.

Why it's important:

- Without optimization, the student would just keep taking tests and failing without improving.
Optimization makes learning happen.

Coding Session

<https://github.com/ishinvin/cnn-demo>

Thanks!

Q & !A