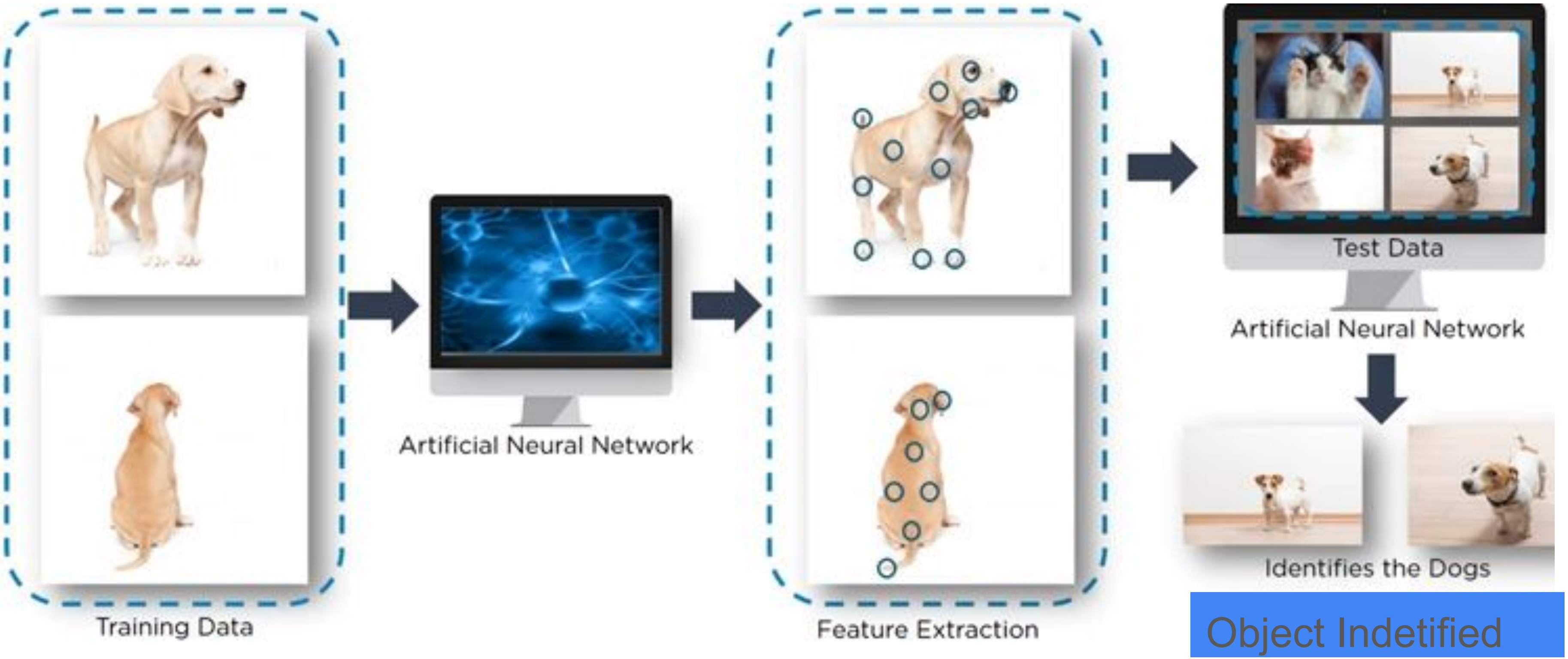
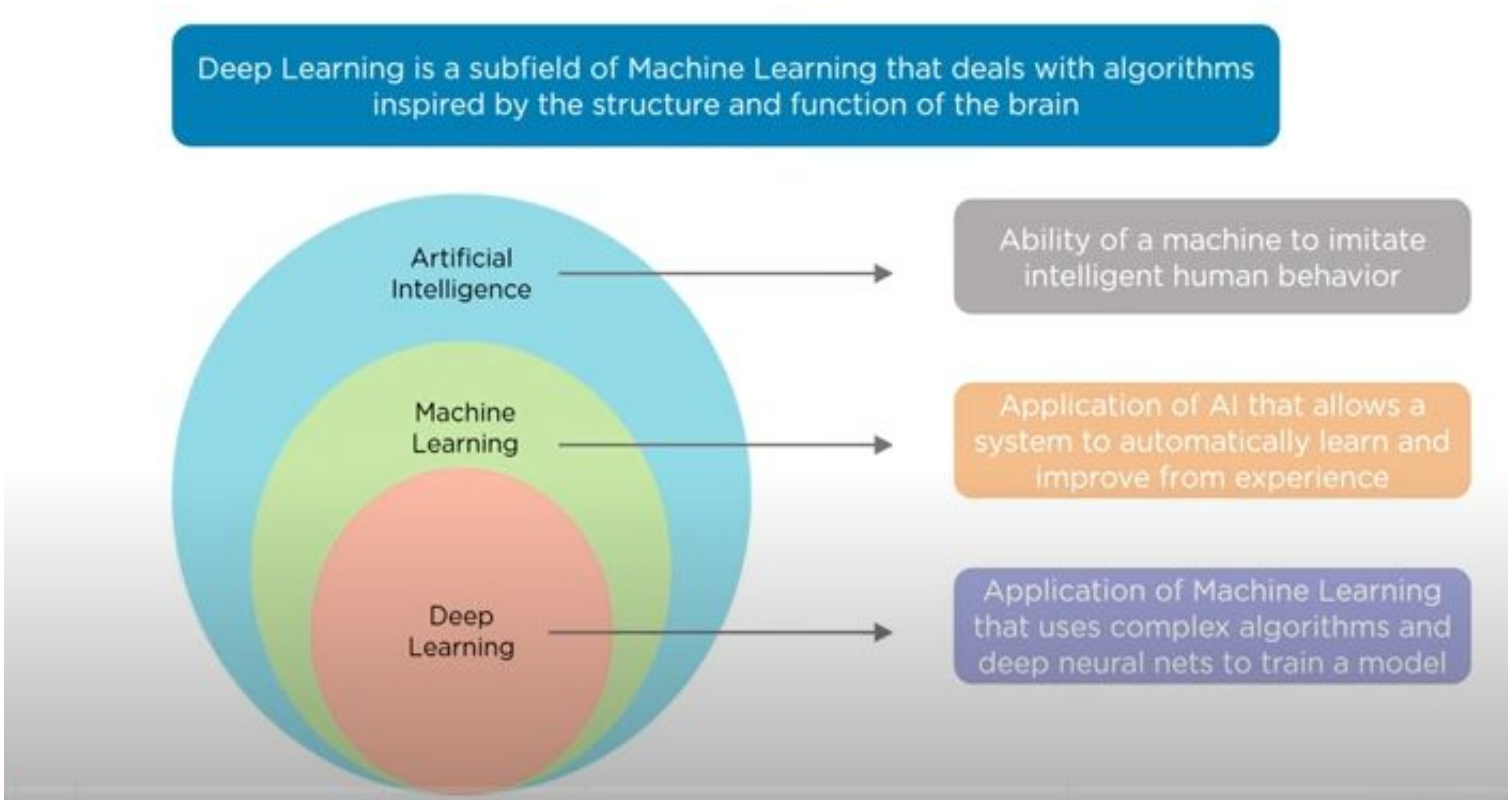
Deep Learning

Narayan

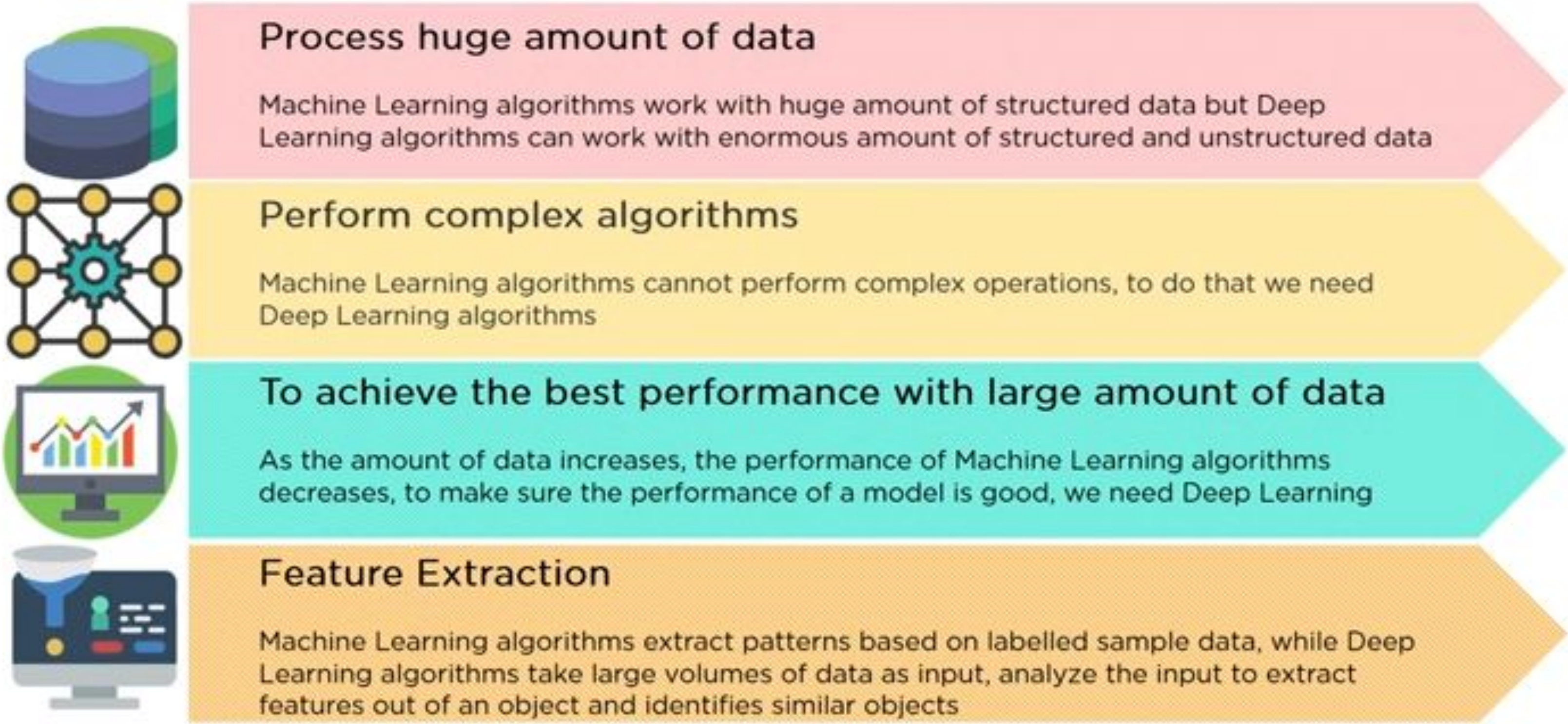
Image Recognition



Deep Learning



Need of Deep Learning



Applications

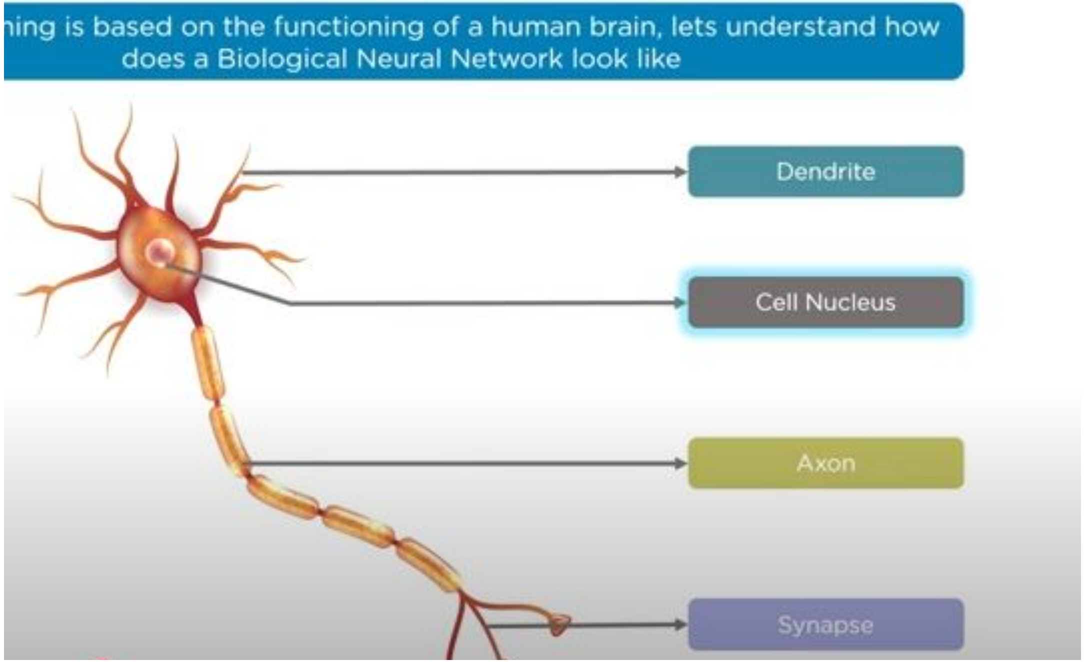
Robot Navigation

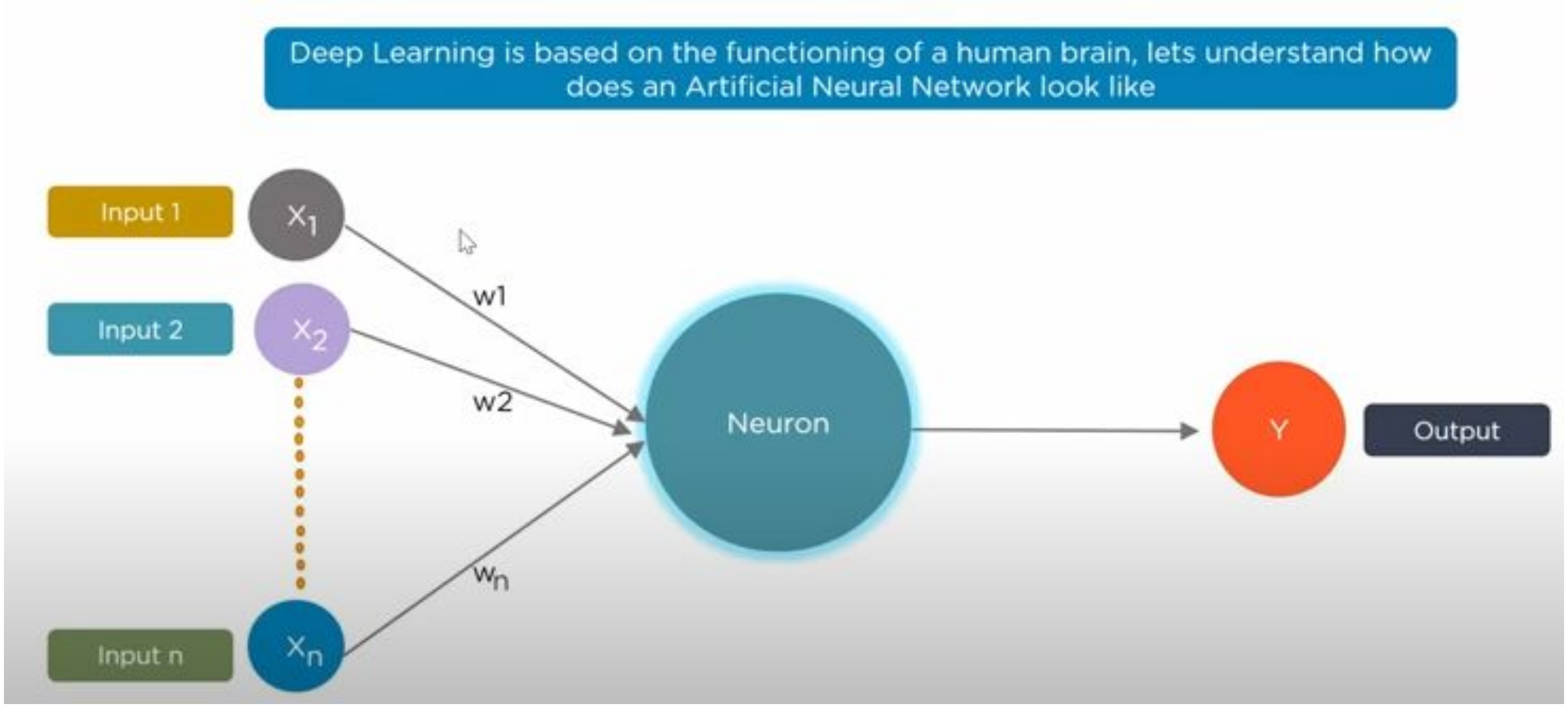

machine
translation
language
communication
internet
vocabulary
technology
display
automation
speech
software
Machine Translation

Music Composition

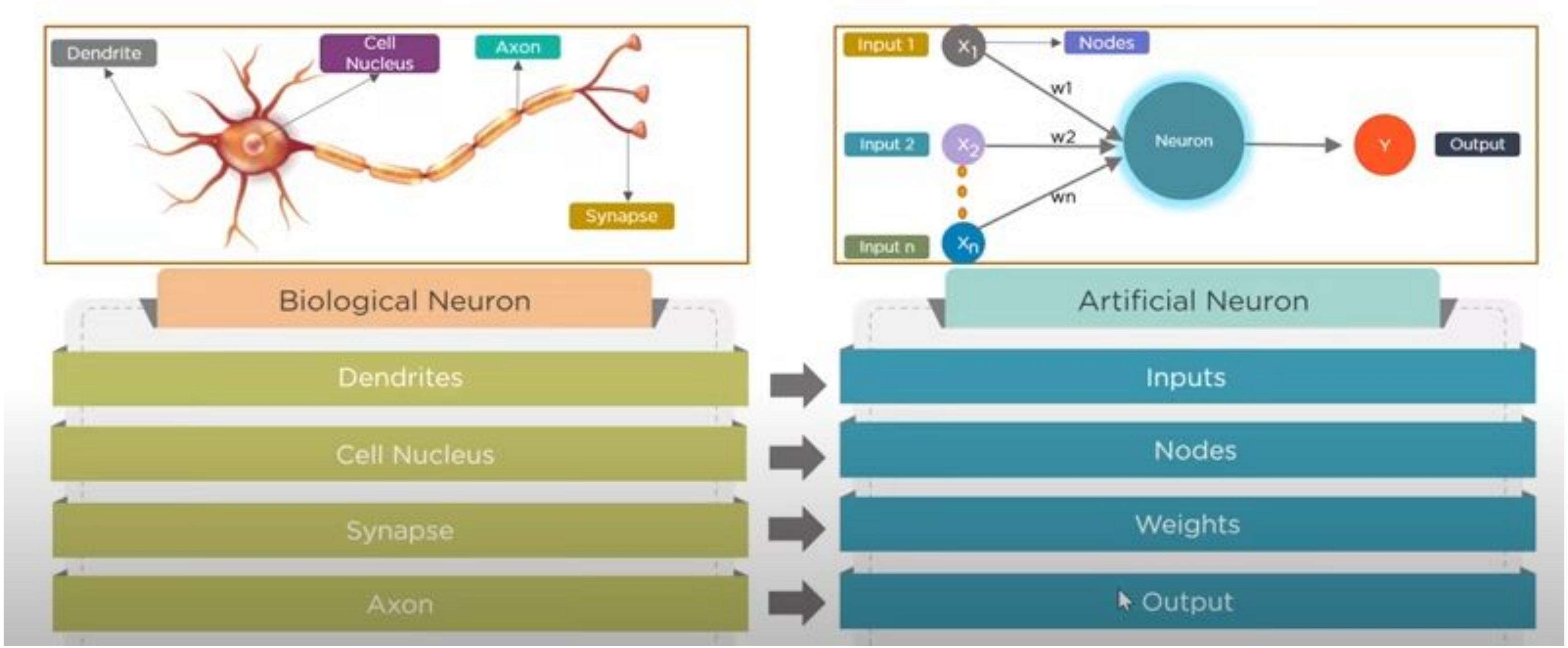
Colorization of images

Neural Network

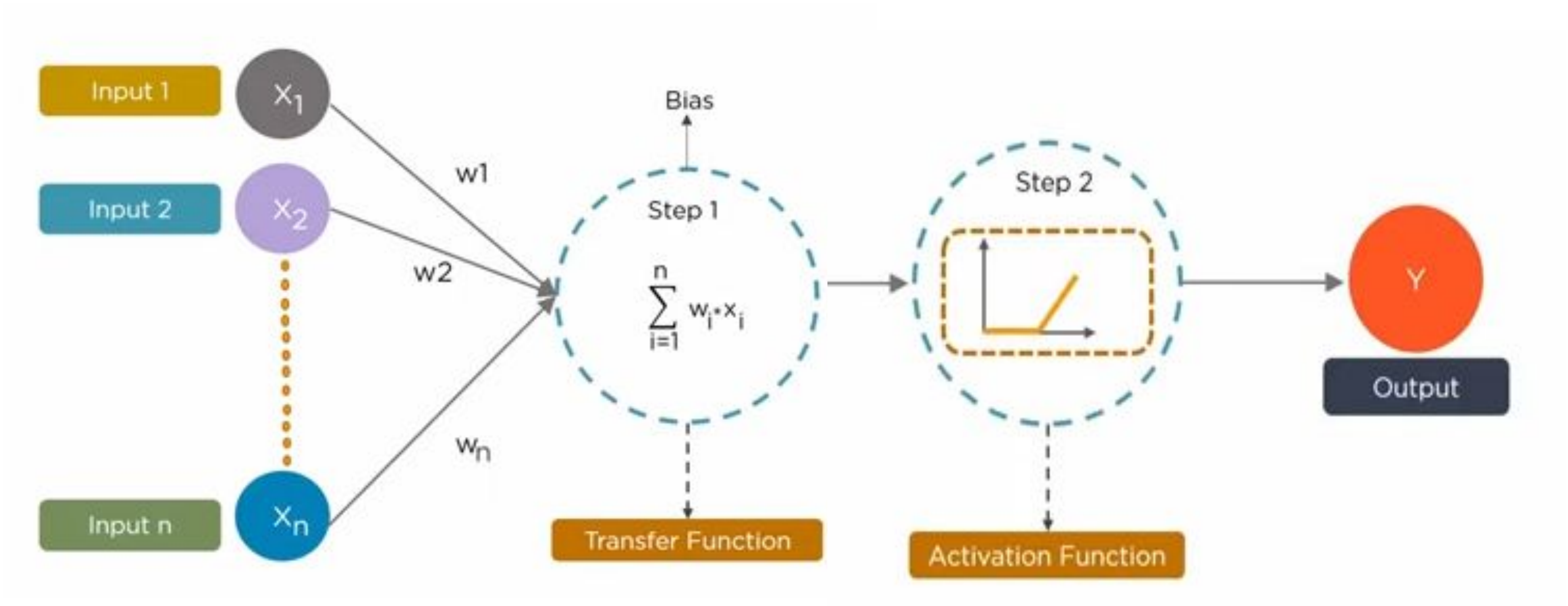



Deep Learning is based on the functioning of a human brain, lets understand how does an Artificial Neural Network look like
Input 1
Input 2
Input n
X1
X2
Xn
w1
w2
wn
Neuron
Y
Output

Biological versus Artificial Neuron

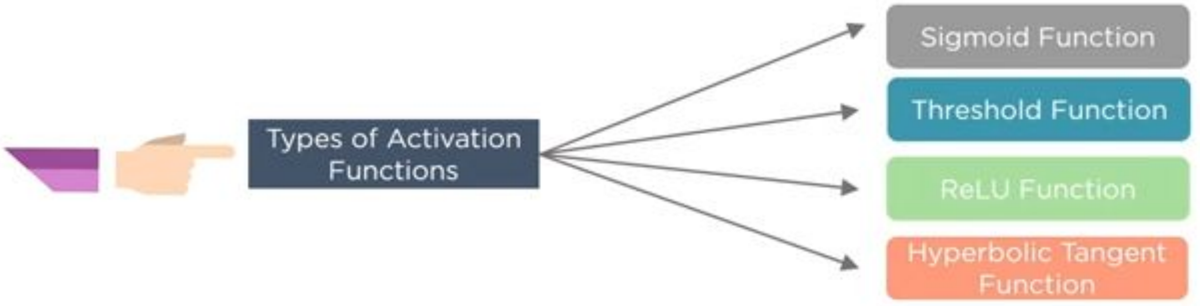


Neural Network



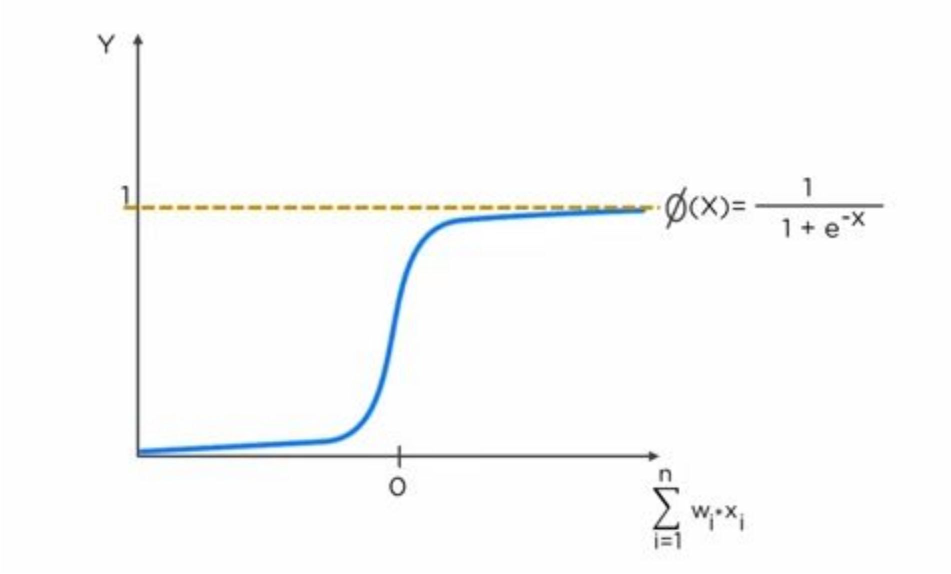
Activation Function

To Bring in nonlinearity



Sigmod

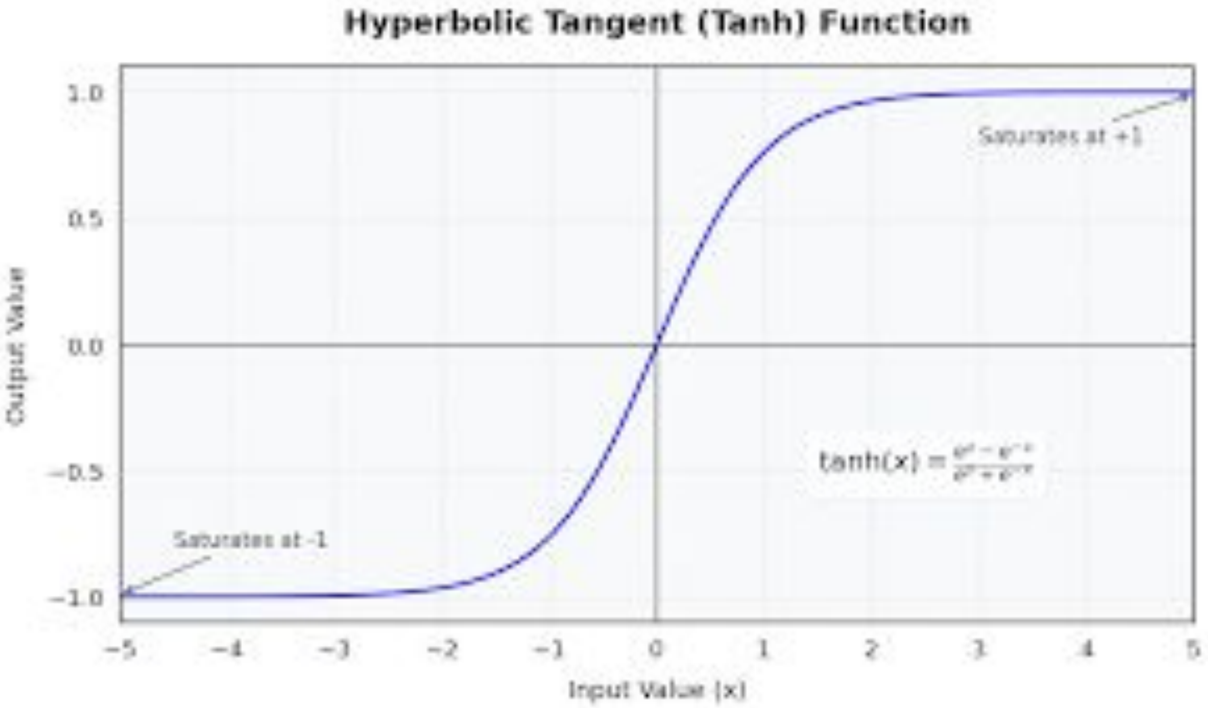
Between 0 and 1



Hyperbolic Tangent (tanh)

✅ Pros: Zero-centered output → Better convergence.

⚠️ Cons: Still suffers from vanishing gradient for large inputs.



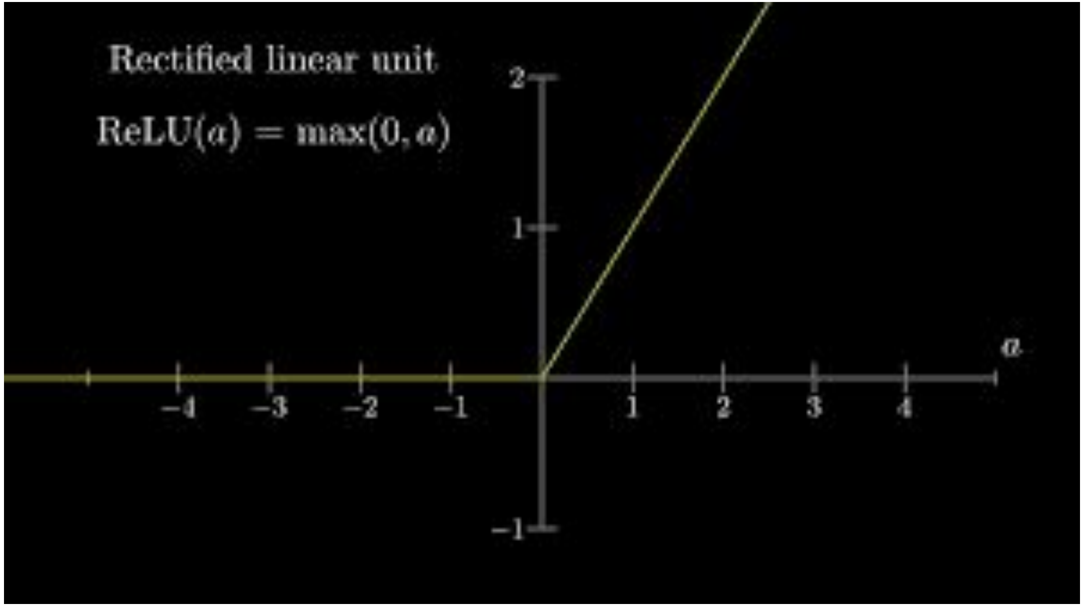
ReLU (Rectified Linear Unit)

Pros:

- Computationally efficient.
- Solves vanishing gradient problem for positive inputs.

Cons:

- "Dying ReLU" problem → Some neurons may never activate (output always 0).



Leaky ReLU

Pros: Prevents dying neurons problem.

Formula:

$$\text{Leaky ReLU}(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$$

(where α is a small constant like 0.01)

Softmax (for Multi-Class Classification)

- Formula:

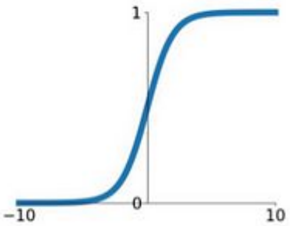
$$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

✅ Converts outputs into probabilities (sum to 1).

Activation Functions

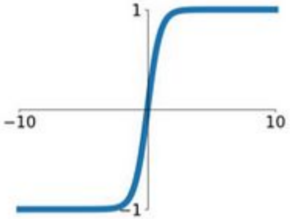
Sigmoid

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



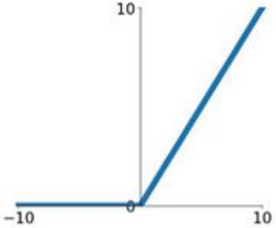
tanh

$\tanh(x)$



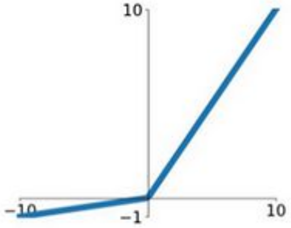
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

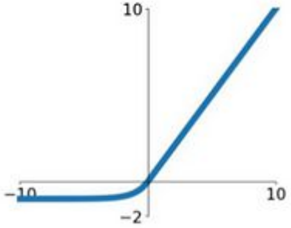


Maxout

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

ELU

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



What is a Deep Feedforward Neural Network?

A Deep Feedforward Neural Network (also called a Multilayer Perceptron, MLP) is the most basic type of deep neural network.

- It consists of:
    1. An Input Layer
    2. One or more Hidden Layers
    3. An Output Layer
- Data flows in one direction— Input → Hidden Layers → Output
- ➔ There is no feedback or cycle (unlike recurrent networks).

Architecture Overview

- **Input Layer**: Takes the input features (e.g., pixel values for images).
- **Hidden Layers:**
  - Each layer contains several **neurons (units)**.
  - Each neuron computes:

    $$z = w \cdot x + b$$

    Then applies an **activation function** (ReLU, sigmoid, etc.).
- **Output Layer**:
  - Produces the final prediction/output.
  - Activation depends on the task:
    - Regression → Linear activation
    - Binary classification → Sigmoid activation
    - Multi-class classification → Softmax activation

Mathematical Expression

For a simple 3-layer deep network:

- Layer 1:
  $$a^{(1)} = \sigma(W^{(1)}x + b^{(1)})$$
- Layer 2:
  $$a^{(2)} = \sigma(W^{(2)}a^{(1)} + b^{(2)})$$
- Output Layer:
  $$\hat{y} = f(W^{(3)}a^{(2)} + b^{(3)})$$

Where:

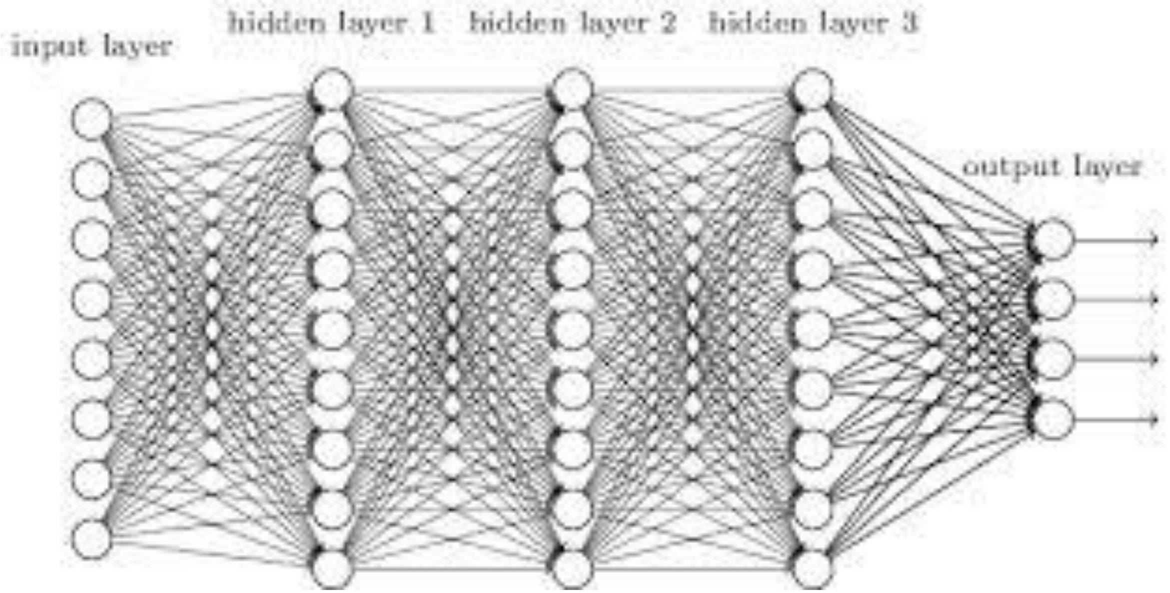
- W = Weight matrices
- b = Bias vectors
- σ = Activation function (ReLU, tanh, etc.)
- f = Output activation function

Why "Deep"?

The term "Deep" refers to having multiple hidden layers.

More hidden layers → Can learn more complex patterns.

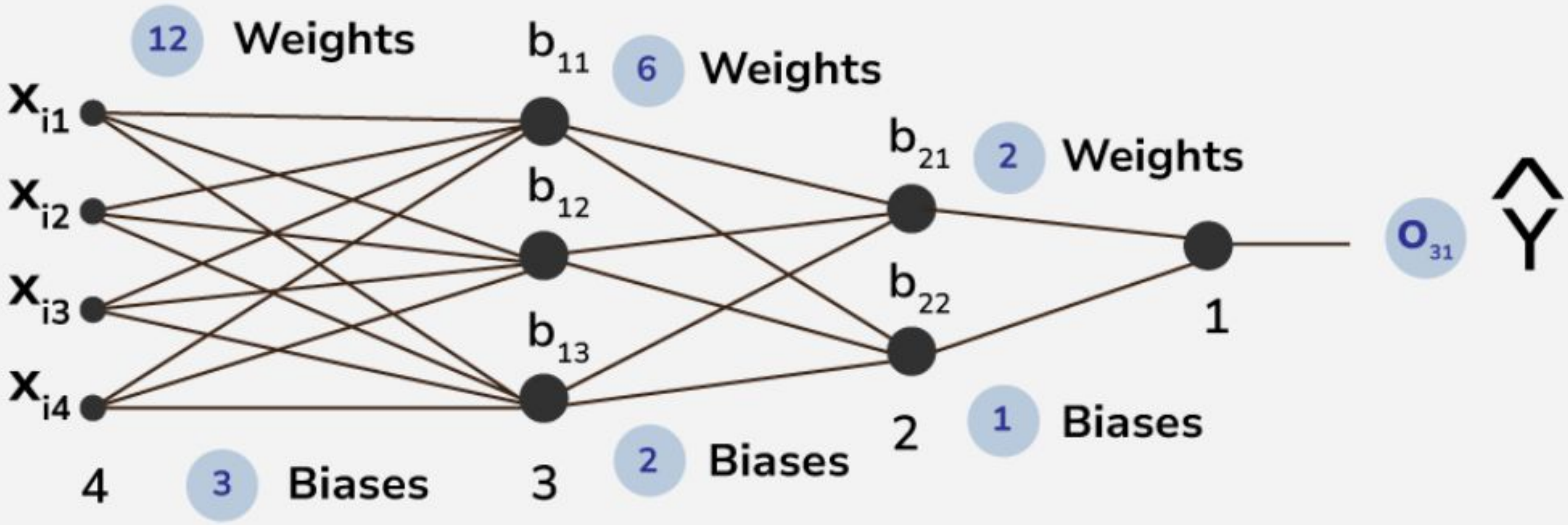
But too many layers → Risk of overfitting and vanishing gradient problem (solved by ReLU and other techniques).



Forward Propagation Process

1. Input data x enters the network.
2. For each layer:
   - Compute linear combination:
     $z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$
   - Apply activation:
     $a^{(l)} = \sigma(z^{(l)})$
3. Final output $\hat{y}$ is computed in the output layer.

Forward Propagation Process



Applications

- **Image Classification (MNIST handwritten digits).**
- **Spam detection in emails.**
- **Predicting house prices.**

Summary

No cycles (unlike recurrent networks).

✔️ Deep networks are more powerful but require:

- Proper regularization (Dropout, L2 regularization).

- Proper activation functions (ReLU to avoid vanishing gradient).

✔️ Training via Backpropagation + Gradient Descent.

Thanks

"The depth of a neural network is not just in its layers, but in its ability to learn complex patterns that we could never explicitly program."