# Introduction to Artificial Intelligence

## Subject 8: Neural Networks and Deep Learning Fundamentals

Dr. Mohammed Shambour

# Learning Objectives

## By the end of this lecture, students will be able to:

Define neural networks and explain their motivation

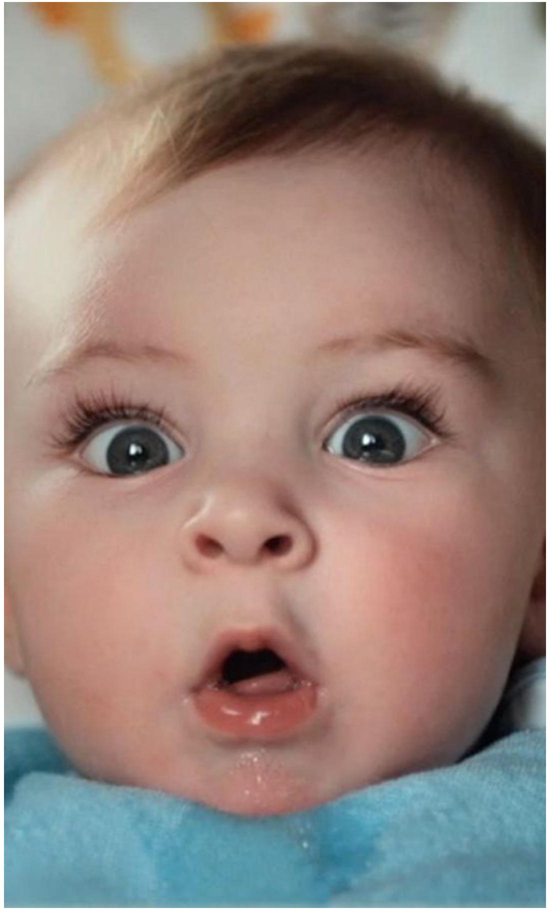
Describe the biological neuron analogy

Explain the perceptron model mathematically

Compare common activation functions

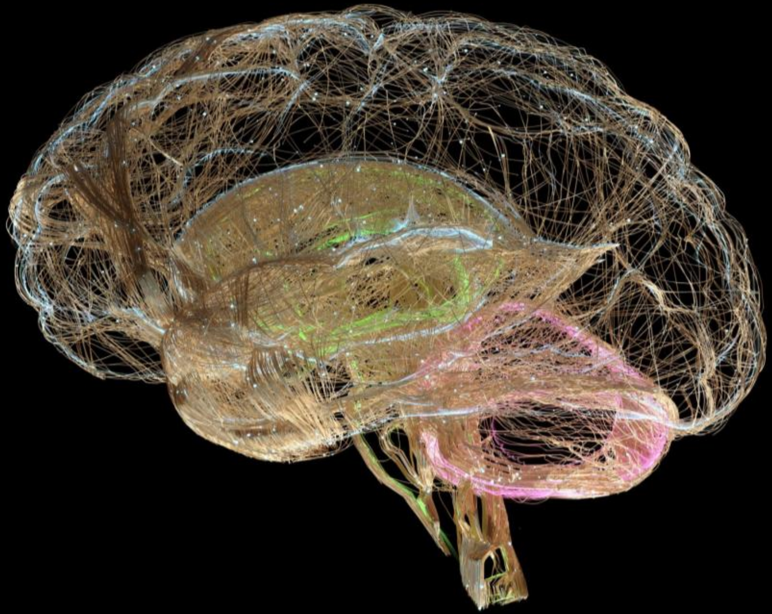
Understand how a single-layer neural network makes decisions

# Reactions



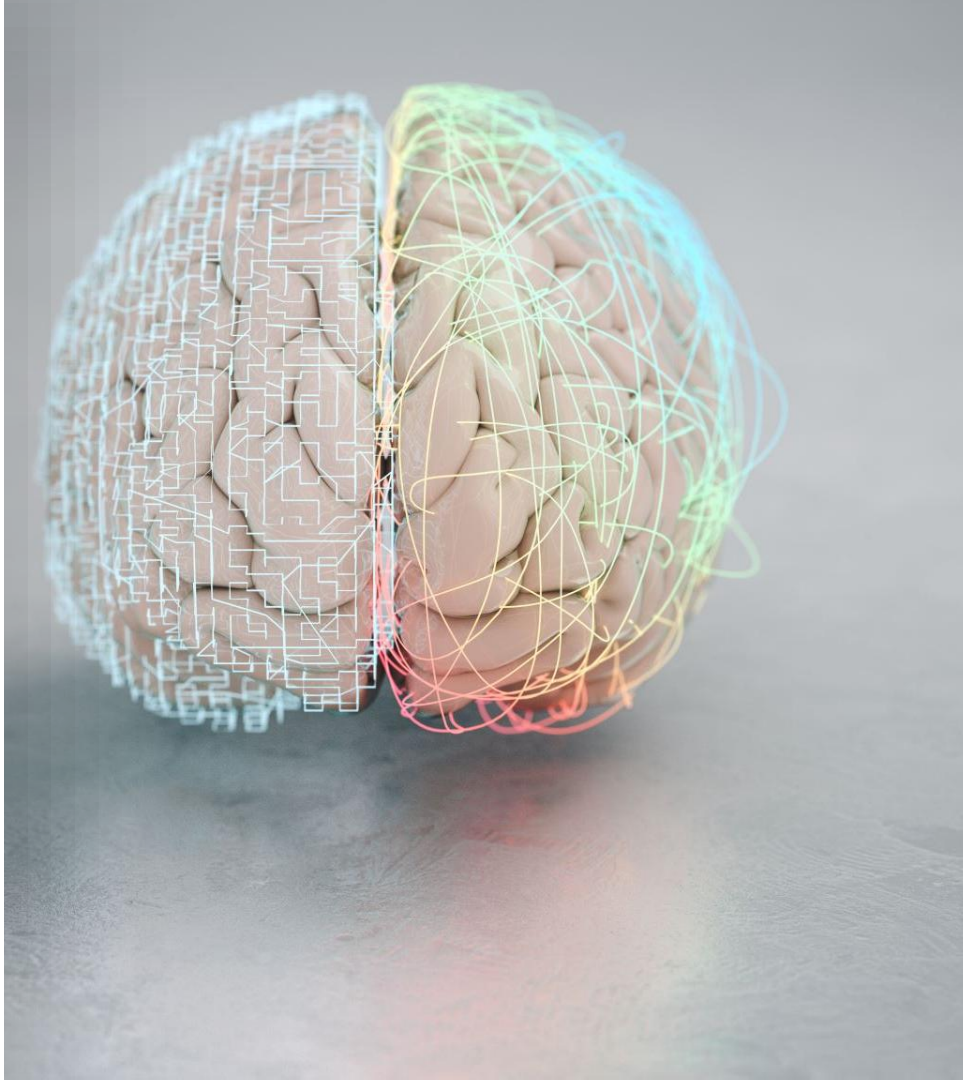
# What is a Neural Network?

A Neural Network (NN) is a type of artificial intelligence (AI) model **designed** to recognize patterns and make decisions in a way that mimics the human brain.

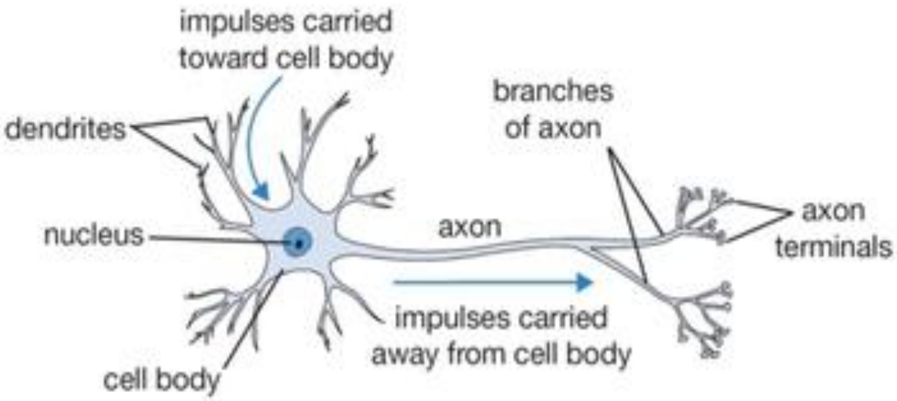


# The Simple Analogy: The Brain

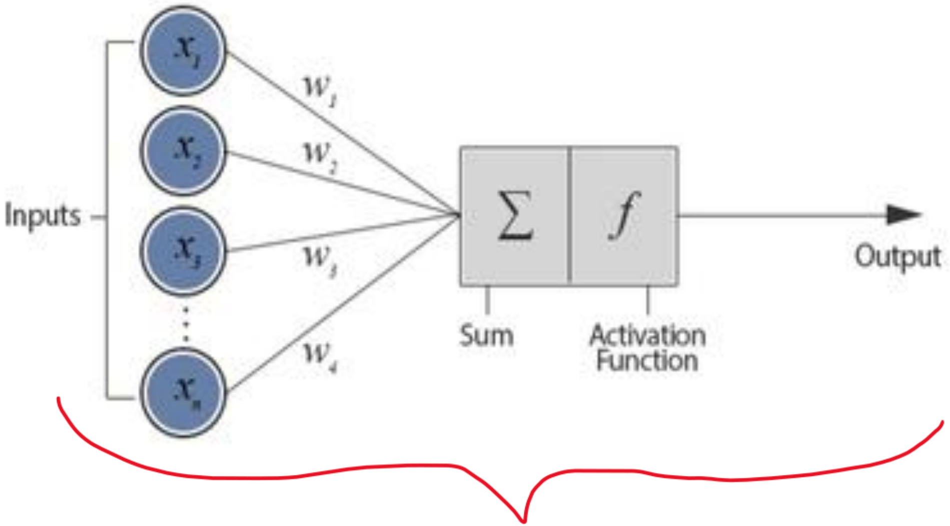
Think of your brain: billions of tiny cells called **neurons** connected to each other. When you learn something new (like riding a bike in a forest), certain connections between neurons strengthen, and a pathway is formed. A neural network mimics this on a very basic level.



# Biological Neuron versus Artificial Neural Network


impulses carried toward cell body
dendrites
nucleus
cell body
branches of axon
axon
impulses carried away from cell body
axon terminals

https://devopedia.org/artificial-neural-network



**A Perceptron**

- **A Perceptron** is the simplest form of a neural network that makes decisions by combining **inputs** ($x_1$,$x_2$,..,$x_n$ ) with **weights** ($w_1$,$w_2$,.., $w_n$) and applying an **activation function** (*f*).
- It is mainly **used** for binary classification problems.
- It forms the **basic building block** of many deep learning models.

# Learning from Experience: Your Brain's Action → Reaction System

Think about learning any new skill - such as avoiding falls while riding a bicycle:

- ❑ Action: You turn the handlebars slightly right
- ❑ Reaction: The bike leans right and you start to fall
- ❑ Adjustment: Your brain notes: “Turning left causes the fall to start !"
- ❑ Next Attempt: You try a smaller left turn → better balance!
- ❑ Repeat: After many wobbles and corrections, you will find the right action to avoid falling..

# Learning from Experience: How This Connects to Neural Networks

## □ Your brain is essentially a biological neural network that:

- Receives **inputs** (sights, sounds, balance signals)
- Takes **actions** (muscle movements)
- Observes **results** (staying up vs. falling)
- **Strengthens connections** for successful movements
- **Weakens connections** for unsuccessful ones

## □ An Artificial Neural Network learns the exact same way:

| Your Brain | AI Neural Network |
|---|---|
| Sees bike wobbling | Receives numerical data (sensor readings) |
| Sends "turn slightly" signal | Computes an output (prediction) |
| Feels "still falling" | Compares to desired outcome (loss calculation) |
| Adjusts muscle memory | Adjusts mathematical weights |
| In the end, it goes smoothly | In the end,  makes accurate predictions |

# Alternative Real-Life Scenarios (Learning from Feedback)

**Learning to cook:**

Too much salt → dish tastes bad → use less next time

**Playing basketball:**

Shoot from far → miss → move closer next time

**Investing:**

Buy stock X → lose money → adjust investment strategy

**Social interactions:**

Say joke Y → people laugh → tell similar jokes again

# Key Insight: Learning is Adjustment

Whether biological or artificial, **learning = gradually adjusting connections based on feedback**. Every **mistake** teaches **what *not* to do**; every **success** reinforces **what *works***.

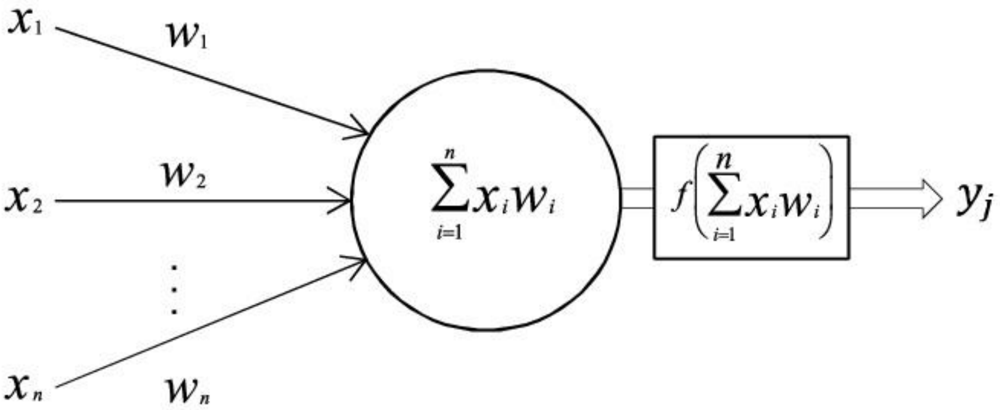
**Why This is important for AI:**

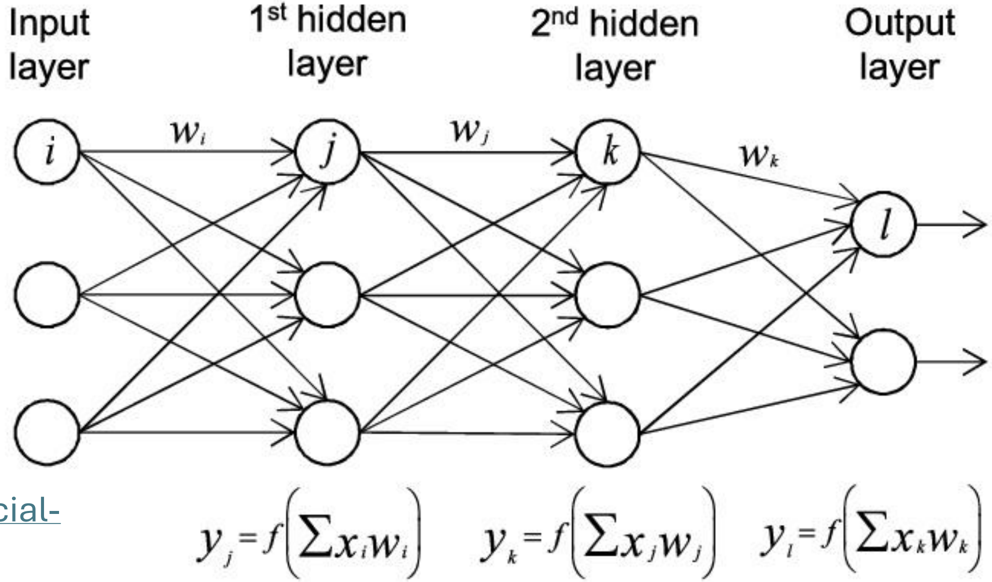
This action-reaction-adjustment loop explains how AI can:

- **Recognize** faces after seeing thousands of examples
- **Translate** languages by comparing millions of sentences
- **Play** games by trying moves and seeing results
- **Without being explicitly programmed for each scenario!**

# Applications of Neural Networks

- Image and video recognition
- Speech and natural language processing
- Medical diagnosis
- Recommendation systems
- Autonomous vehicles




Input layer
1st hidden layer
2nd hidden layer
Output layer
$i$
$w_i$
$j$
$w_j$
$k$
$w_k$
$l$
$y_j = f\left(\sum x_i w_i\right)$
$y_k = f\left(\sum x_j w_j\right)$
$y_l = f\left(\sum x_k w_k\right)$

https://devopedia.org/artificial-neural-network

# Components of Neural networks (NN)

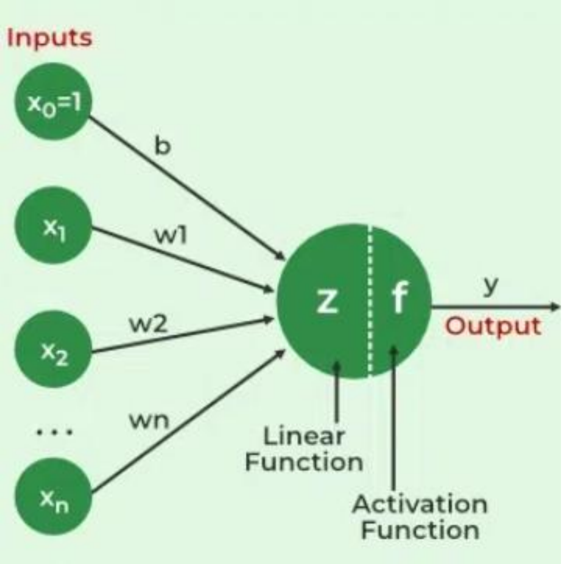
**A typical network is organized in layers:**

**1.Input Layer:** This is where the data enters the network. Each node represents one feature of your data (e.g., one pixel, one word in a sentence).

**2.Hidden Layer(s):** This is where the magic happens. There can be one or many (hence "deep" learning) hidden layers. Each layer detects increasingly complex patterns.

1. The first hidden layer might detect simple edges in an image.
2. The next might detect shapes like circles or squares.
3. A deeper layer might detect complex objects like eyes, wheels, or doors.

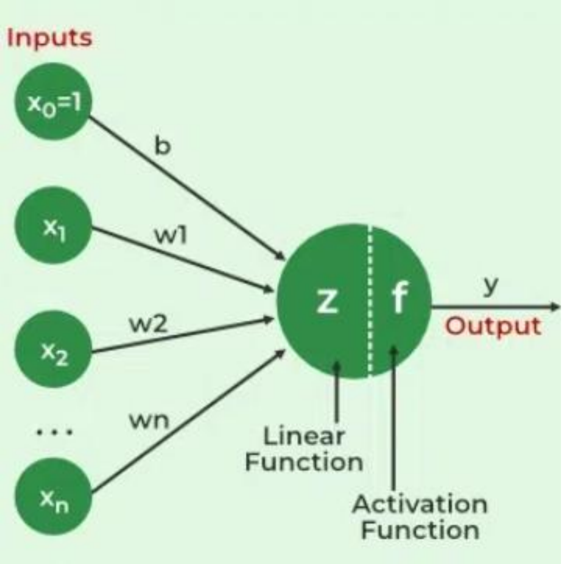
**3.Output Layer:** This produces the final result. For a cat/dog classifier, it might have two nodes: one for "cat probability" and one for "dog probability."



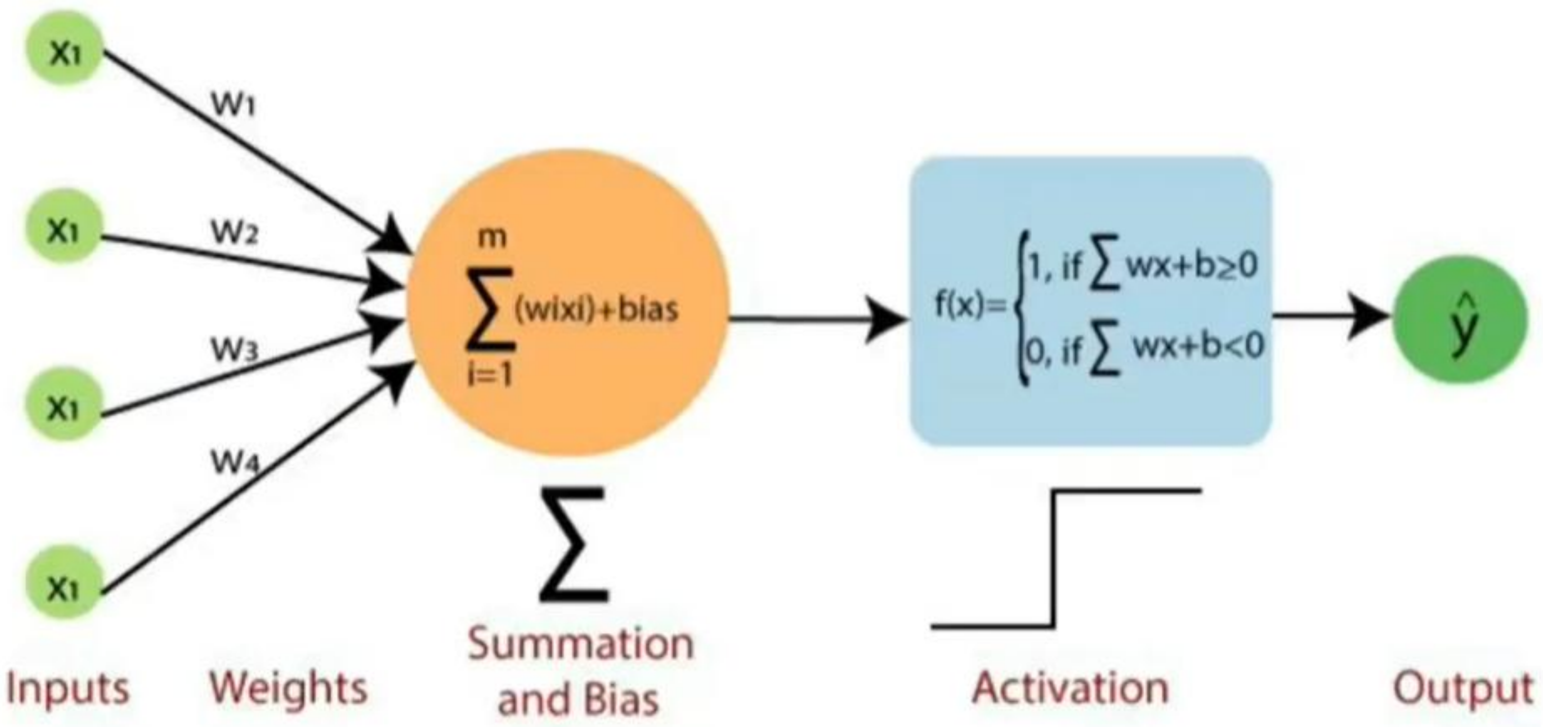
https://www.geeksforgeeks.org/machine-learning/neural-networks-a-beginners-guide/

# Forward propagation

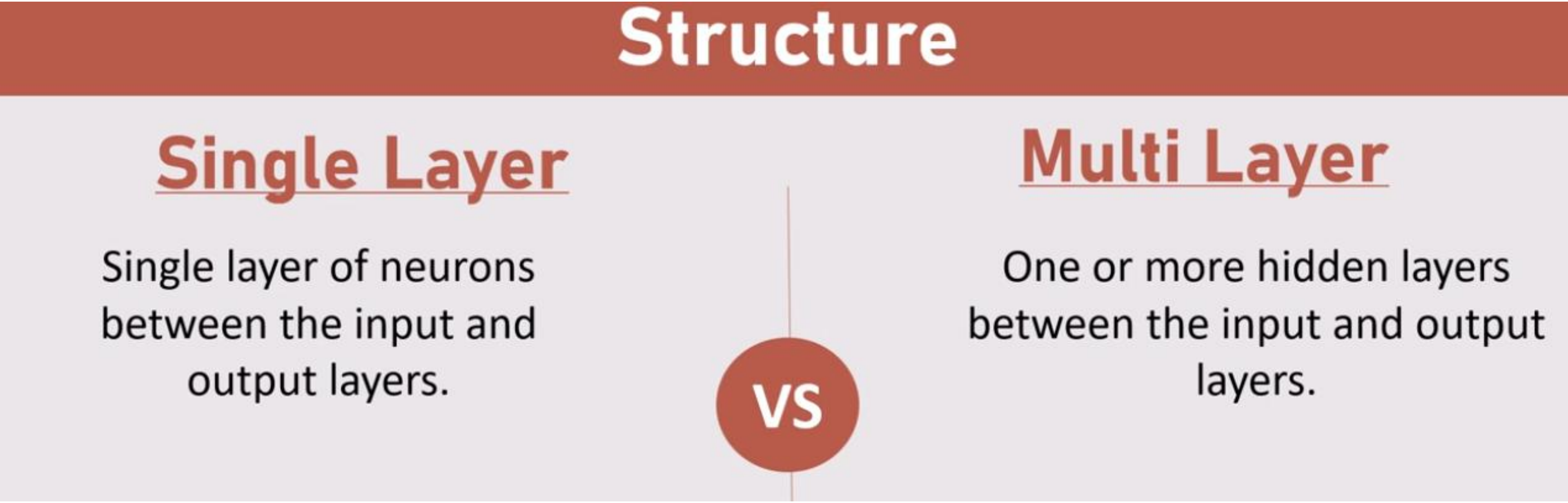
Information flows **forward** from the input layer, through the hidden layers, to the output layer. This is called **forward propagation.**



# Example..



X1
X1
X1
X1
W1
W2
W3
W4
$\sum_{i=1}^{m}$ (wixi)+bias
$\sum$
f(x)= {1, if $\sum$ wx+b≥0; 0, if $\sum$ wx+b<0}
$\hat{y}$
Inputs
Weights
Summation and Bias
Activation
Output



## Structure

### Single Layer

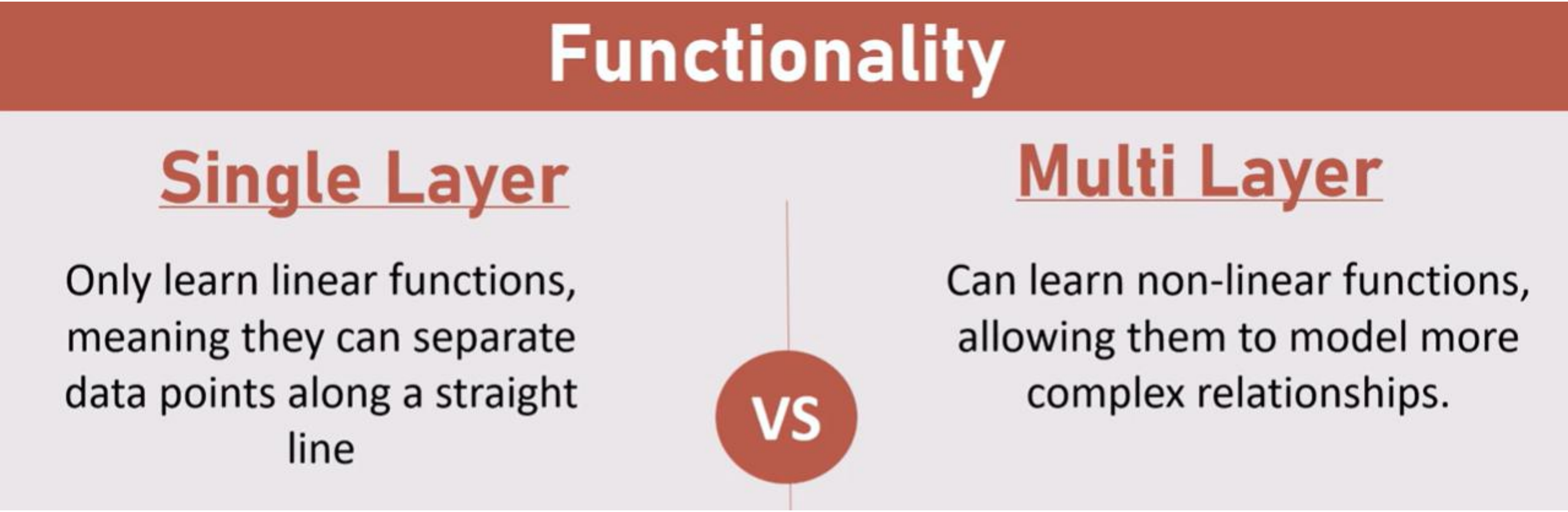
Single layer of neurons between the input and output layers.

VS

### Multi Layer

One or more hidden layers between the input and output layers.

https://www.youtube.com/watch?v=WeGAcvuVKpg&t=119s



# Functionality

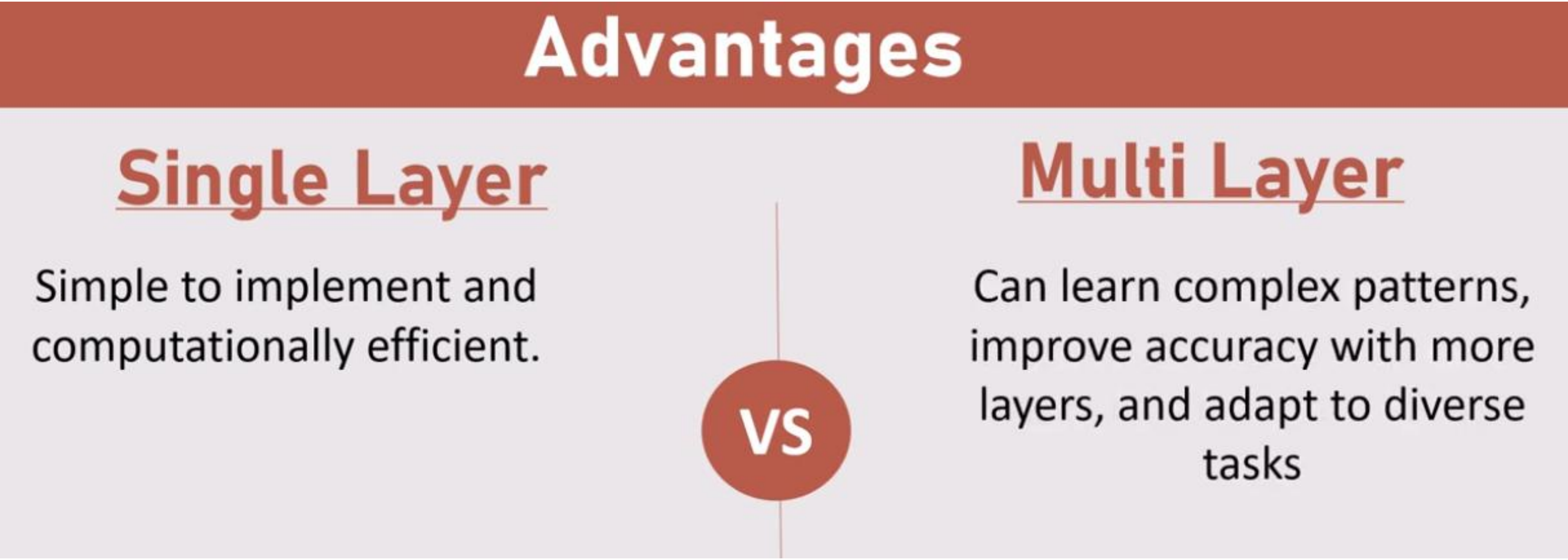
## Single Layer

Only learn linear functions, meaning they can separate data points along a straight line

VS

## Multi Layer

Can learn non-linear functions, allowing them to model more complex relationships.



# Advantages

## Single Layer

Simple to implement and computationally efficient.

VS

## Multi Layer

Can learn complex patterns, improve accuracy with more layers, and adapt to diverse tasks

# Limitations

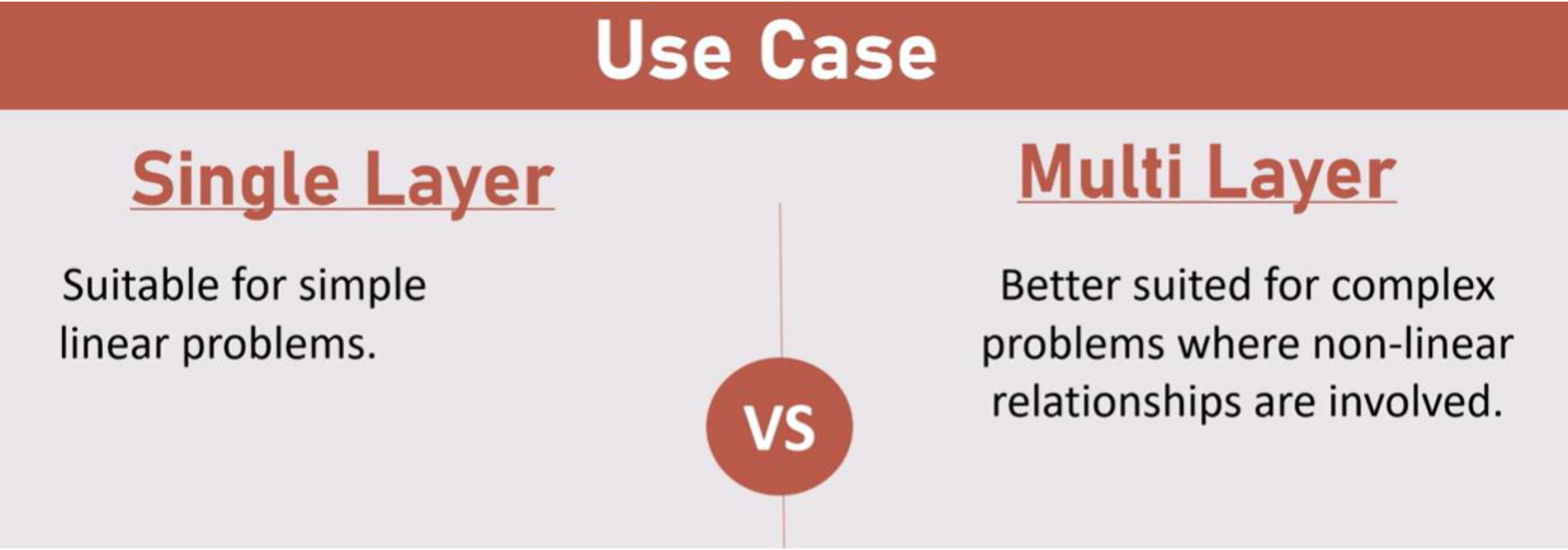
## Single Layer

Limited in their ability to model complex, non-linear relationships.

VS

## Multi Layer

Require more training time and computational resources.



# Use Case

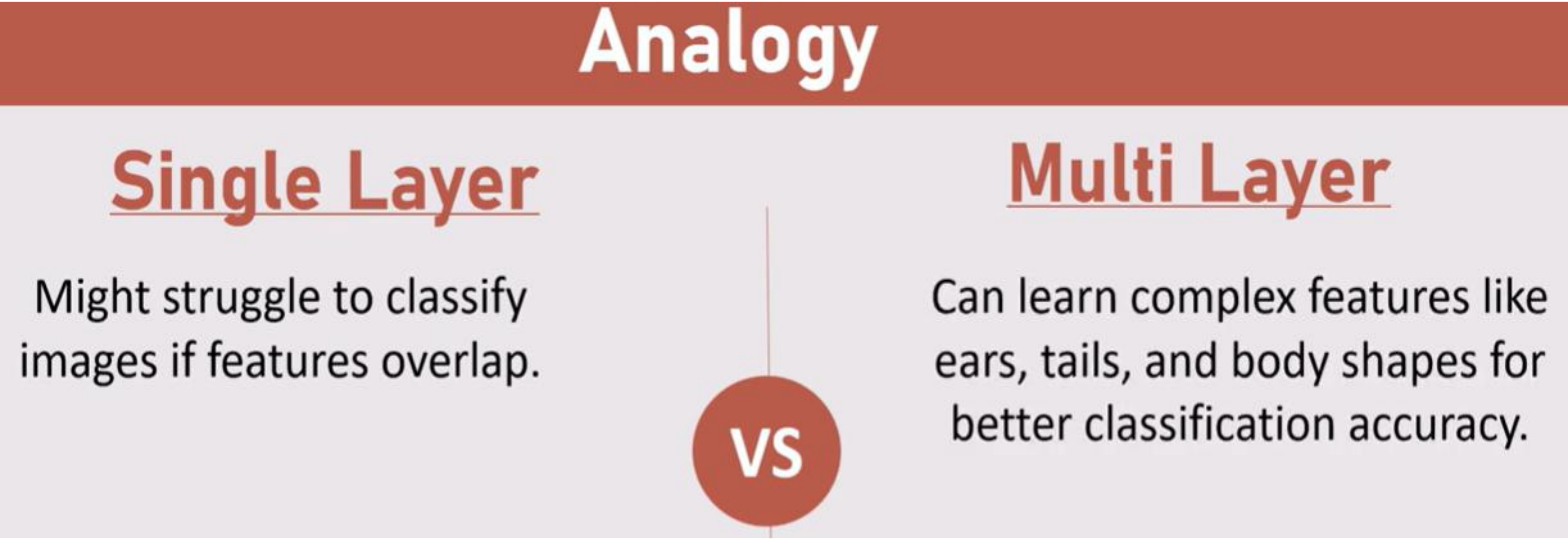
## Single Layer

Suitable for simple linear problems.

VS

## Multi Layer

Better suited for complex problems where non-linear relationships are involved.



# Analogy

## Single Layer

Might struggle to classify images if features overlap.

VS

## Multi Layer

Can learn complex features like ears, tails, and body shapes for better classification accuracy.

# Shallow and Deep Neural Networks

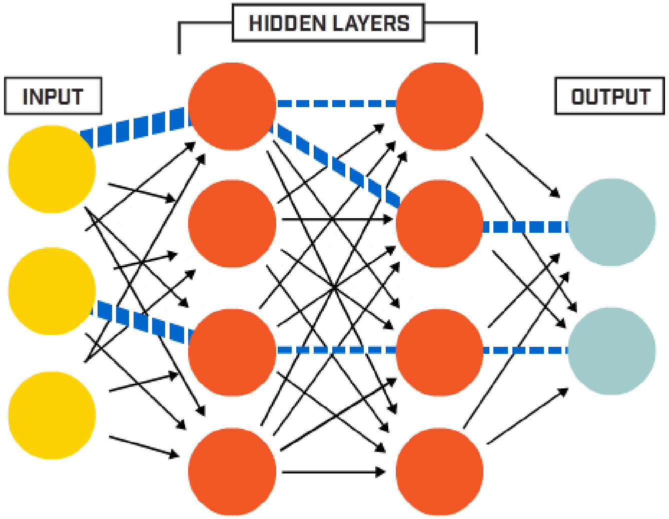
| Aspect | Shallow Neural Networks | Deep Neural Networks |
|---|---|---|
| **Definition** | Neural networks with few layers (typically one to three hidden layers) | Neural networks with many layers (multiple hidden layers) |
| **Model Depth** | Shallow architecture | Deep architecture |
| **Computational Complexity** | Low | High |
| **Learning Capacity** | Limited | High |
| **Data Requirement** | Requires less training data | Requires large amounts of data for effective training |
| **Number of Parameters** | Fewer parameters | Significantly more parameters |
| **Computational Resources** | Requires minimal computational resources | Requires much computational resources (e.g., GPUs, TPUs) |
| **Interpretability** | Easier to interpret and analyze | More difficult to interpret (black-box nature) |
| **Training Time** | Short | Long |
| **Typical Performance** | Suitable for simple problems | Superior performance on complex problems |
| **Example Models** | Single-layer Perceptron, Logistic Regression | CNNs, RNNs, Transformers |

# Types of Neural Networks (Common Architectures)

**1. Feedforward Neural Networks (FFNNs):** The simplest type, where connections don't form cycles.

### Core Architecture

- Strictly unidirectional flow - Information moves only forward: Input → Hidden Layers → Output
- Fully connected - Each neuron in one layer connects to every neuron in the next layer (dense connections)
- No feedback loops - No connections going backward or forming cycles
- Typically **shallow** - Usually 1-3 hidden layers (though can be deeper)



### Example Applications

- Customer churn prediction
- Real estate price prediction
- Simple classification problems

https://medium.com/@b.terryjack/introduction-to-deep-learning-feed-forward-neural-networks-ffnns-a-k-a-c688d83a309d

# Types of Neural Networks (Common Architectures)

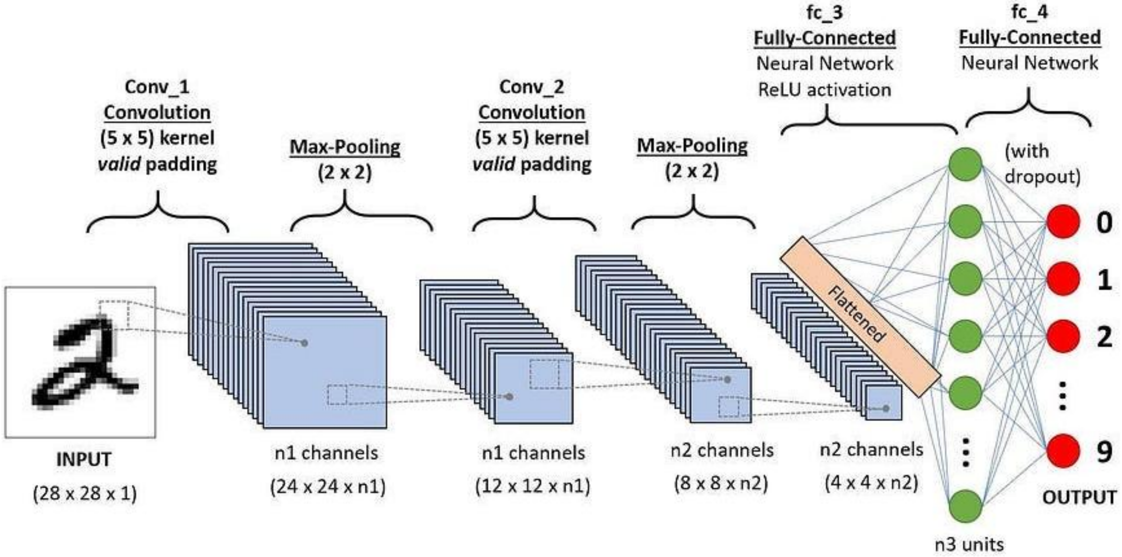
**2. Convolutional Neural Networks (CNNs):** The gold standard for image/video recognition. They use special "filters" to scan for spatial patterns (like edges and textures).

### Core Architecture

- Convolutional Layers
- Pooling Layers
- Fully connected NN

### Example Applications

- Image classification
- Audio processing
- Face recognition
- Object detection (e.g., YOLO)
- Video analysis



# Types of Neural Networks (Common Architectures)

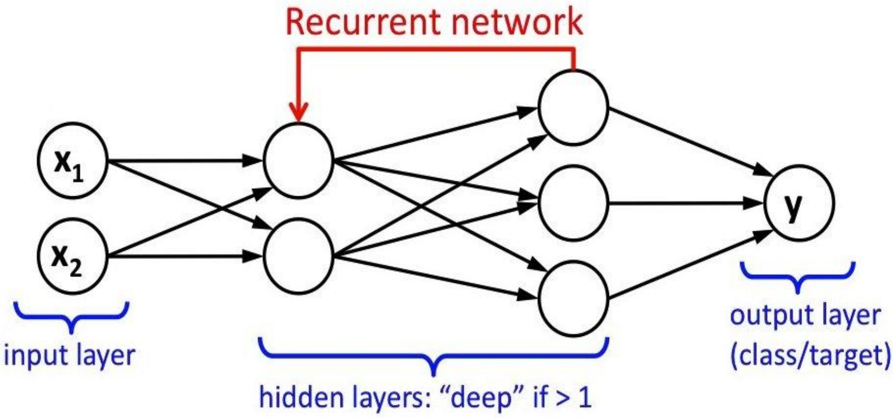
**3. Recurrent Neural Networks (RNNs):** RNNs are neural networks designed for **sequential** data processing. Unlike feedforward networks, RNNs have **internal memory**, which makes the output at time step *t* dependent on the input from previous time steps ($t_1$, $t_2$, …).

### Core Architecture

- Sequential processing – Designed to handle ordered data (time steps)
- Memory of past inputs – Current output depends on current input and previous states
- Parameters shared across time steps– patterns in sequences are often time-invariant (e.g. Analogy: Word meaning doesn't change based on its position in a sentence )

### Example Applications

- Time series forecasting: Stock prices, weather, energy demand
- Natural Language Processing
- Speech recognition: Audio signal processing
- Video captioning: Understanding temporal sequences



# Types of Neural Networks (Common Architectures)

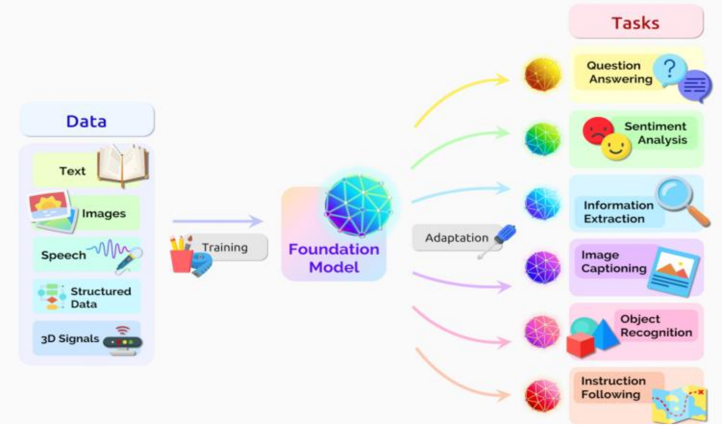
**4. Transformers:** The architecture behind modern large language models (LLM: e.g., GPT, BERT). They use a mechanism called "attention" to weigh the importance of different parts of the input sequence (e.g., all words in a sentence) simultaneously, making them incredibly powerful for language tasks.

## Core Architecture

- Self-attention mechanism – Models relationships between all input elements simultaneously to handle ordered data (time steps)
- Parallel processing – No recurrence; entire sequence processed at once
- Encoder–decoder structure – Encoders capture représentations (context and relationships), decoders generate outputs

## Example Applications

- Machine translation (e.g., Google Translate)
- Text summarization
- Question answering
- Code generation



https://blogs.nvidia.com/blog/what-is-a-transformer-model/

|  | Traditional ML | Neural Networks |
|---|---|---|
| **Feature Engineering** | Manual feature extraction required | Automatically learns features |
| **Model Complexity** | Generally simple to moderately complex | Highly flexible and complex |
| **Data Requirement** | Performs well on small to medium datasets | Typically requires large datasets |
| **Interpretability** | High (models are explainable) | Low to moderate (black box) |
| **Training Time** | Computationally inexpensive training | Computationally high expensive training |
| **Scalability** | Limited scalability with large data | Scales well with GPUs and big data |
| **Performance on Complex Tasks** | Limited for high-dimensional data | Excellent for images, speech, and text |
| **Hardware Dependency** | Runs efficiently on CPUs | Often requires GPUs/TPUs |
| **Typical Use Cases** | Structured/tabular data | Images, text, speech, video |
| **Examples** | K-Nearest Neighbors (k-NN) and Support Vector Machines (SVM) , Decision Trees | Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Feed Forward Networks(FFNN) |

# URL References

- https://www.geeksforgeeks.org/deep-learning/difference-between-shallow-and-deep-neural-networks/
- https://medium.com/@prajun_t/recurrent-neural-networks-rnns-f06affba0c5b
- https://medium.com/@amanatulla1606/transformer-architecture-explained-2c49e2257b4c

That’s all for Today