flow chart of working principle of Neural networks

Start
↓
🟢 Input Layer (X1, X2, ..., Xn)
↓
🔵 Apply Weights (W1, W2, ..., Wn)
↓
🟠 Summation Function: Z = Σ(Wi * Xi) + B
↓
🔴 Activation Function (ReLU / Sigmoid / Tanh)
↓
🟡 Hidden Layer 1 (Transformed Features)
↓
🔴 Activation Function (ReLU / Sigmoid / Tanh)
↓
🟡 Hidden Layer 2 (Further Feature Learning)
↓
🔴 Activation Function (ReLU / Sigmoid / Tanh)
↓
🔵 Output Layer (Final Prediction)
↓
🟠 Loss Calculation (Compare Predicted vs. Actual)
↓
🔄 Backpropagation (Update Weights)
↓
🔁 Repeat Until Convergence
↓
End

Explanation of the Flowchart

1️⃣ **Input Layer** → Takes input features (X1, X2, Xn).

2️⃣ **Weighted Summation** → Computes `Z = Σ(Wi * Xi) + B`.

3️⃣ **Activation Function** → Introduces non-linearity (ReLU, Sigmoid, etc.).

4️⃣ **Hidden Layers** → Learn feature representations using activations.

5️⃣ **Output Layer** → Produces final prediction/classification.

6️⃣ **Loss Calculation** → Computes the error between predicted and actual values.

7️⃣ **Backpropagation** → Adjusts weights to minimize loss.

8️⃣ **Repeat Training** → Until the model learns patterns correctly.

🔷 Step-by-Step Working of ANN

1️⃣ Input Layer: Receiving Data

- The **input layer neurons** take the raw input features (`X1, X2, ..., Xn`).
- Each feature is assigned an initial **weight (`W1, W2, ..., Wn`)**.
- Example: For an image, input neurons represent **pixel values**.

💡 **Example (House Price Prediction)**
Features (X):

- `X1 = Square Footage`
- `X2 = Number of Bedrooms`
- `X3 = Age of House`

2️⃣ Hidden Layers: Feature Transformation

- Each neuron computes a **weighted sum** using the formula:Z=∑(Wi×Xi)+BZ = \sum (W_i \times X_i) + BZ=∑(Wi×Xi)+B
- Example Calculation:Z=(W1×X1)+(W2×X2)+(W3×X3)+BZ = (W1 \times X1) + (W2 \times X2) + (W3 \times X3) + BZ=(W1×X1)+(W2×X2)+(W3×X3)+B
- The result **Z** is passed through an **activation function** to introduce non-linearity.

Common Activation Functions:

- **ReLU (Rectified Linear Unit)** → Best for deep networks (sets negative values to 0)

- **Sigmoid** → Used for probability outputs (binary classification)
- **Tanh** → Used when values should be centered around zero

Example Activation (ReLU):

$A = \max(0, Z)$

3️⃣ Output Layer: Generating Predictions

- The final layer applies an **activation function**:
  - **Sigmoid (0 to 1) → Binary Classification**
  - **Softmax (Multi-class) → Probability Distribution**
  - **Linear (Regression) → Continuous Output**
- Produces the final **predicted value or class**.

💡 **Example:**
If predicting house price, the output layer will provide a **single continuous value**.
If classifying images (e.g., dog vs cat), the output layer provides a **probability score for each class**.

🔄 Training Process (Weight Optimization)

The ANN **learns** by adjusting the **weights** and **biases** through **backpropagation**:

1. **Forward Propagation**:
   a. Inputs are multiplied by weights, summed, and activated to pass forward.
2. **Loss Calculation**:
   a. The model compares the **predicted output** vs. the **actual output**.
   b. Uses **Loss Functions**:
      i. MSE (Regression)
      ii. Cross-Entropy (Classification)
3. **Backpropagation & Optimization**:
   a. **Gradient Descent** updates weights using derivatives of the loss function.
   b. Adjusts weights to minimize the error.
4. **Repeat Until Convergence**:
   a. The model continues training until **error is minimized**.

🎯 Summary

✅ **Input Layer** → Receives raw data (features)

✅ **Summation Function** → Computes weighted sums (`Σ Wi * Xi + B`)

✅ **Hidden Layers** → Apply activation functions for feature transformation

✅ **Output Layer** → Produces final prediction/classification

✅ **Training Process** → Optimizes weights using backpropagation

Activation Function	Where to Use?	When to Use?	Why Use It?	Example Use Cases
tanh (Hyperbolic Tangent)	Hidden layers of Neural Networks	When inputs need to be centered around zero (good for handling both positive and negative values).	Outputs range from -1 to 1, making gradient updates smoother.	- Sentiment Analysis (captures both +ve & -ve sentiments) - Stock Price Prediction (positive & negative trends) - Medical Image Processing (handling pixel values centered at zero)
Sigmoid	Binary classification problems,	When probabilities need to be between 0	Outputs a smooth probability,	- Spam Detection (Email is spam or not) - Medical Diagnosis

	logistic regression, output layers	and 1 (useful for probability-based outputs).	easy to interpret.	(Disease prediction: yes/no) - Fraud Detection (Classifying transactions as fraud/not)
Step Function	Perceptrons, Binary classification	When a hard decision is needed (ON/OFF).	Simple but not used in deep learning due to non-differentiability.	- Threshold-based Systems (Light ON/OFF based on sensor values) - Basic Perceptron Networks (Early AI models)
ReLU (Rectified Linear Unit)	Deep Neural Networks (CNNs, RNNs, MLPs)	When training deep networks to avoid vanishing gradients.	Keeps positive values, sets negative values to zero, making training faster.	- Image Classification (CNNs) (e.g., Face Recognition) - Speech Recognition (Deep Speech models) - Object Detection (YOLO, Faster R-CNN)
Softmax	Last layer of multi-class classification models	When multiple classes exist, and probabilities are needed.	Converts logits into a probability distribution (sum = 1).	- Handwritten Digit Recognition (MNIST, multi-class) - Language Translation Models (Choosing the best word) - Object Detection Models (Identifying objects in images)

please see the table difference of activation function

- Use **tanh** if you need both **positive and negative** values.
- Use **sigmoid** for **binary classification** or probability-based outputs.
- Use **step function** for **hard binary decisions** (not used in deep learning).
- Use **ReLU** for deep networks to improve training efficiency.
- Use **softmax** for **multi-class classification**.