□ Day 18 – The Role of Weights and Biases in Neural Networks

🔍 Introduction

In a neural network, **weights** and **biases** are the two primary building blocks that learn from data. Every prediction the model makes relies on how these parameters are adjusted. Mastering their concept is key to understanding the internal workings of deep learning.

⚙ What Are Weights?

- Weights define the **strength of the connection** between neurons.
- When an input is passed through the network, it is **multiplied by a weight**—this helps decide how much influence that input should have.
- At the start, weights are **randomly initialized**, but during training, they are **fine-tuned using backpropagation** to reduce error.

📌 **Example:**
If the input = 4 and the weight = 2,
→ Weighted input = 4 × 2 = **8**

🎯 What Is Bias?

- Bias lets the activation function **shift up or down**, rather than always going through the origin.
- This shift gives the model **more flexibility** to fit complex patterns in data.
- Even when all inputs are 0, bias ensures that neurons can still fire and produce a useful output.

📌 **Example:**
If weighted input = 8 and bias = 3,
→ Final output before activation = 8 + 3 = **11**

□ Neuron's Mathematical Expression

The output from a neuron (before applying activation function) can be written as:

z=(w1·x1+w2·x2+...+wn·xn)+bz = (w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n) + bz=(w1·x1+w2·x2+...+wn·xn)+b

Where:

- w1,w2,...,wnw_1, w_2, ..., w_nw1,w2,...,wn = weights
- x1,x2,...,xnx_1, x_2, ..., x_nx1,x2,...,xn = inputs
- bbb = bias

🌈 Visual Intuition

- **Weight** = how steep the line is (slope of output curve)
- **Bias** = how much the curve is shifted up/down (vertical translation)

Together, they define the **shape and position** of the model's response.

📊 Comparison Table

Feature	Weights	Bias
Role	Adjusts importance of input	Allows output shift
Applied As	Multiplies input	Added after weighted sum
Function	Helps learn relationships	Adds model flexibility
Trainable	✅ Yes	✅ Yes
Quantity	One per input	One per neuron

🔄 How Are They Trained?

Both weights and bias are **updated during training** using algorithms like **Gradient Descent** and **Backpropagation**.

📌 Update Formula:

w=w−learning_rate⋅∂Loss∂ww = w - \text{learning_rate} \cdot \frac{\partial \text{Loss}}{\partial w}w=w−learning_rate⋅∂w∂Loss b=b−learning_rate⋅∂Loss∂bb = b - \text{learning_rate} \cdot \frac{\partial \text{Loss}}{\partial b}b=b−learning_rate⋅∂b∂Loss

This ensures the model improves by minimizing prediction errors over time.

🧑‍💻 Code: Visualizing Effect of Weight & Bias

```
import numpy as np

import matplotlib.pyplot as plt

def neuron_output(x, weight, bias):

    return weight * x + bias

x = np.linspace(-5, 5, 100)

y1 = neuron_output(x, weight=1, bias=0)

y2 = neuron_output(x, weight=2, bias=0)

y3 = neuron_output(x, weight=2, bias=3)

plt.plot(x, y1, label='w=1, b=0')

plt.plot(x, y2, label='w=2, b=0')

plt.plot(x, y3, label='w=2, b=3')

plt.title("Effect of Weights and Bias")

plt.xlabel("Input")

plt.ylabel("Output")

plt.grid(True)
```

plt.legend()

plt.show()

📌 **What this shows:**

- **Increasing weight** makes the slope steeper
- **Adding bias** shifts the line vertically

✅ **Conclusion**

- Weights and biases are at the **core of a neural network's learning**.
- Weights decide **how strongly inputs affect the output**.
- Bias ensures that the model can **adjust independently** of input values.
- Both are continuously updated to make the model smarter with each epoch.