DAY 18 – Understanding Weights and Bias in Neural Networks

🧠 Introduction

In any **neural network**, the two most important trainable components are **weights** and **biases**. These parameters are adjusted during training to help the model learn patterns from the data. Understanding how **weights and biases** work is fundamental to understanding how deep learning models function internally.

□ What Are Weights?

- **Weights** are the **core strength** of the connection between two neurons.
- They **multiply** with the input features to determine how much influence that input has on the output.
- Initially, weights are randomly assigned, but during training, they are adjusted using **backpropagation** to minimize the error.

📌 **Example**:
If input = 2 and weight = 3,
Then: `weighted input = 2 × 3 = 6`

⚖□ What Is a Bias?

- **Bias** allows the activation function to be **shifted left or right**, which is crucial for learning complex patterns.
- Without bias, all activation functions will always pass through the origin (0,0), limiting the model's learning.
- Bias helps the model make better predictions even when input is zero.

📌 **Example**:
If weighted input = 6 and bias = 2,
Then final output before activation = 6 + 2 = **8**

□ Equation of a Neuron

The output of a neuron before activation can be given by:

$z = (w1 * x1 + w2 * x2 + ... + wn * xn) + b$
$z=(w1∗x1+w2∗x2+...+wn∗xn)+b$

Where:

- `w1, w2, ..., wn` are weights
- `x1, x2, ..., xn` are input features
- `b` is the bias term

⚡ Visual Analogy

Think of weights and bias like this:

- **Weight** is the **steepness or slope** of the curve (how much an input affects the output)
- **Bias** is the **offset** (how much the output is shifted vertically)

📊 Summary Table

Feature	Weights	Bias
Role	Controls the impact of inputs	Allows shifting of activation function
Operation	Multiplies input features	Added after weight × input
Importance	Helps learn patterns	Helps model flexibility
Trainable	✅ Yes	✅ Yes
Value	Usually multiple (one per input)	Usually single per neuron

---

🔄 How Are They Updated?

Weights and bias are updated during training using **gradient descent** and **backpropagation**.

📌 **Update Rule**:

```
w = w - learning_rate * ∂Loss/∂w
b = b - learning_rate * ∂Loss/∂b
```

🧪 **Code Demonstration: Weights & Bias Effect**

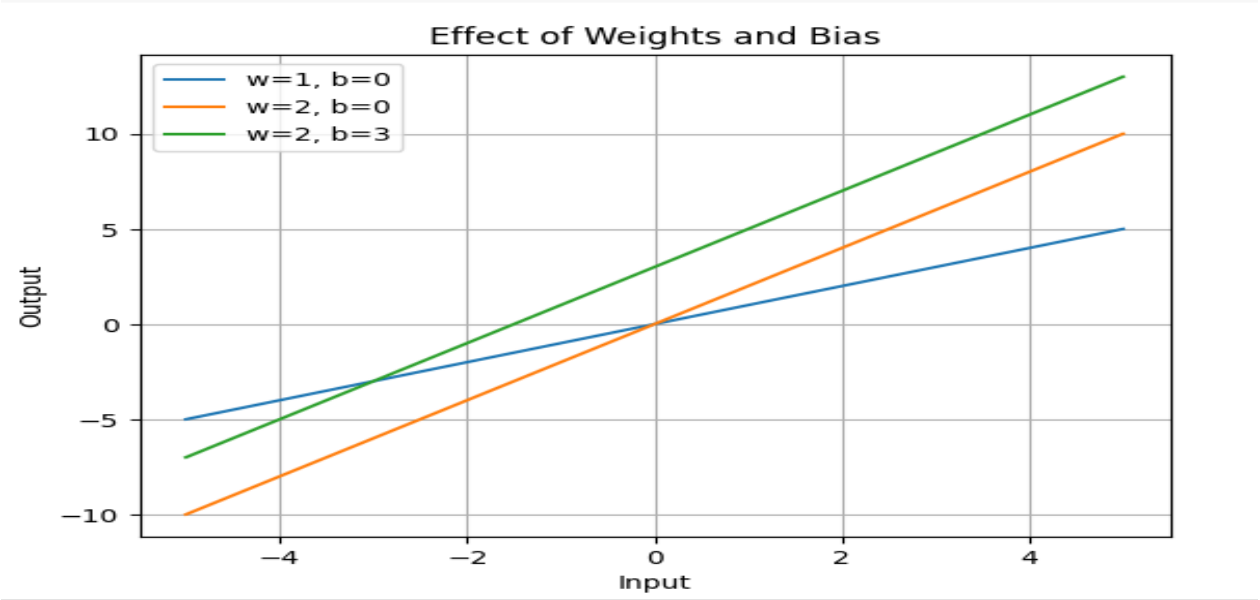
```
import numpy as np
import matplotlib.pyplot as plt

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()
```



📌 This graph shows how:

- Increasing weights steepens the curve
- Adding bias shifts the curve up/down

✅ Conclusion

- **Weights** and **biases** are the foundation of a neural network's learning.
- Weights determine how much importance to give to inputs.
- Bias ensures that the model does not always have to pass through the origin.
- Both are **trainable parameters** and essential for deep learning models to learn complex, non-linear relationships.