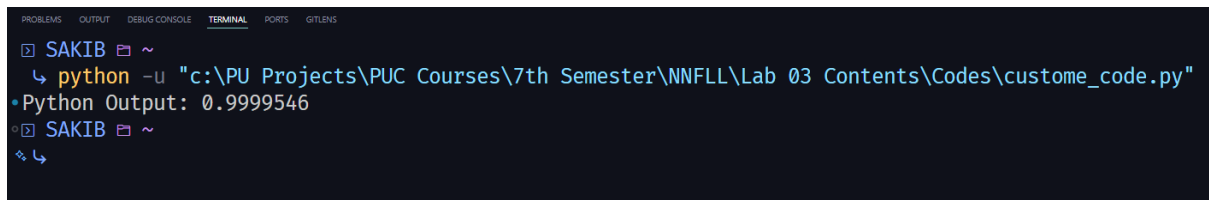


Forward Propagation Custom Code

```
1 import numpy as np
2
3 class NeuralNetwork:
4     def __init__(self):
5         # Layer 1 weights and biases (input: 3 → hidden1: 3)
6         self.W1 = np.array([[2, 4, 6],
7                               [3, 5, 7],
8                               [4, 6, 8]], dtype=np.float32)
9         self.b1 = np.array([[1], [1], [1]], dtype=np.float32)
10
11        # Layer 2 weights and biases (hidden1: 3 → hidden2: 2)
12        self.W2 = np.array([[3, 5],
13                              [4, 6],
14                              [7, 8]], dtype=np.float32)
15        self.b2 = np.array([[2], [2]], dtype=np.float32)
16
17        # Layer 3 weights and biases (hidden2: 2 → hidden3: 2)
18        self.W3 = np.array([[5, 7],
19                              [6, 8]], dtype=np.float32)
20        self.b3 = np.array([[3], [3]], dtype=np.float32)
21
22        # Output layer weights and biases (hidden3: 2 → output: 1)
23        self.W4 = np.array([[4],
24                              [5]], dtype=np.float32)
25        self.b4 = np.array([[1]], dtype=np.float32)
26
27    def relu(self, x):
28        return np.maximum(0, x)
29
30    def sigmoid(self, x):
31        return 1 / (1 + np.exp(-x))
32
33    def forward(self, x):
34        x = x.reshape(-1, 1) # Ensure column vector
35
36        # Layer 1
37        z1 = np.dot(self.W1.T, x) + self.b1
38        a1 = self.relu(z1)
39
40        # Layer 2
41        z2 = np.dot(self.W2.T, a1) + self.b2
42        a2 = self.relu(z2)
43
44        # Layer 3
45        z3 = np.dot(self.W3.T, a2) + self.b3
46        a3 = self.sigmoid(z3)
47
48        # Output Layer
49        z4 = np.dot(self.W4.T, a3) + self.b4
50        output = self.sigmoid(z4)
51
52        return output
53
54    # Instantiate model
55    model = NeuralNetwork()
56
57    # Input vector
58    x = np.array([5, 3, 2], dtype=np.float32)
59
60    # Forward pass
61    output = model.forward(x)
62
63    print("Python Output:", output[0][0])
64
```

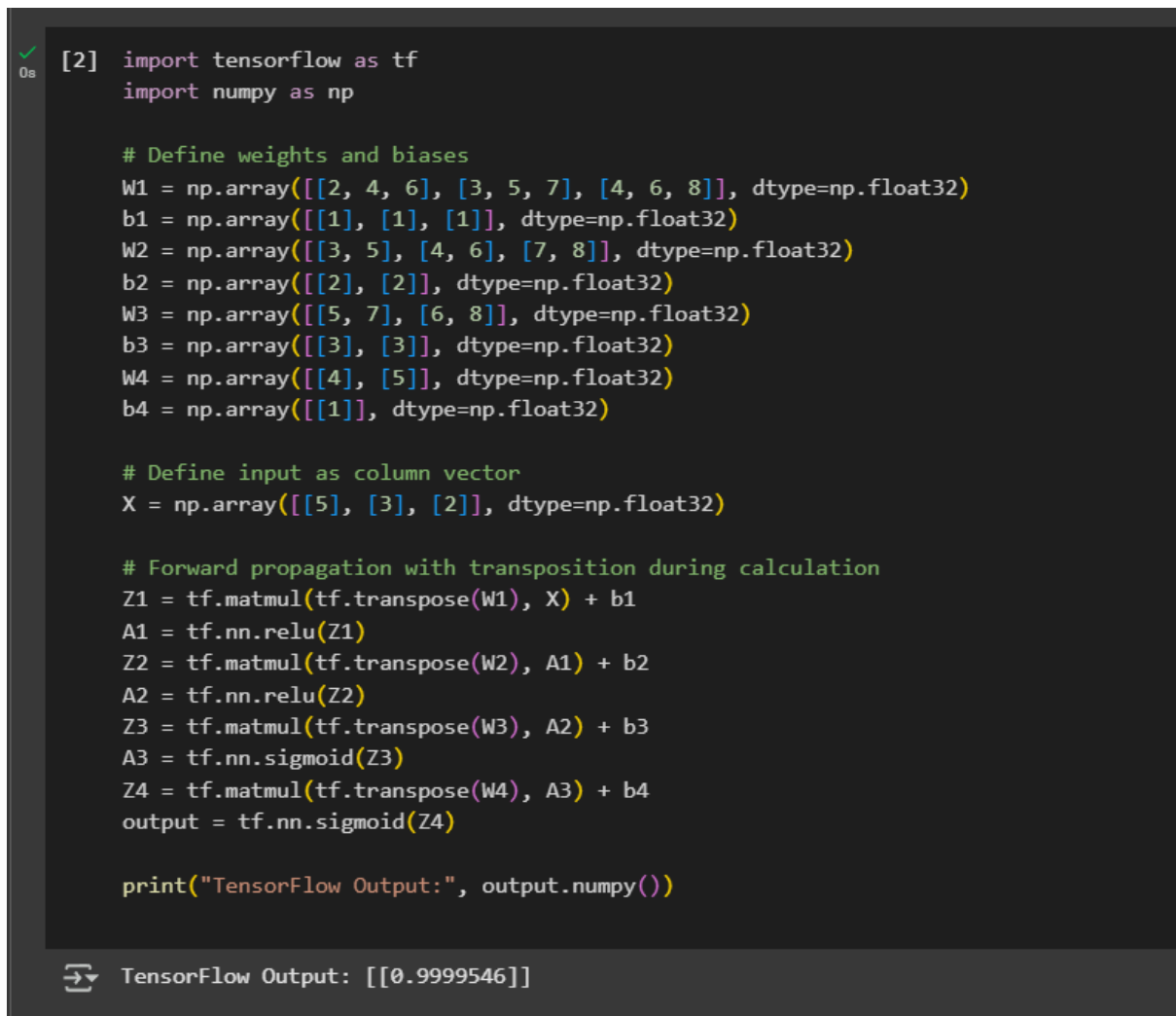
Figure 1: Custom Code for Forward Propagation



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
SAKIB ~
python -U "c:\PU Projects\PUC Courses\7th Semester\NNFLL\Lab 03 Contents\Codes\custome_code.py"
Python Output: 0.9999546
SAKIB ~
```

Figure 2: Output of Custom Forward Propagation Code in PyTorch

Forward Propagation in TensorFlow



```
[2] import tensorflow as tf
import numpy as np

# Define weights and biases
W1 = np.array([[2, 4, 6], [3, 5, 7], [4, 6, 8]], dtype=np.float32)
b1 = np.array([[1], [1], [1]], dtype=np.float32)
W2 = np.array([[3, 5], [4, 6], [7, 8]], dtype=np.float32)
b2 = np.array([[2], [2]], dtype=np.float32)
W3 = np.array([[5, 7], [6, 8]], dtype=np.float32)
b3 = np.array([[3], [3]], dtype=np.float32)
W4 = np.array([[4], [5]], dtype=np.float32)
b4 = np.array([[1]], dtype=np.float32)

# Define input as column vector
X = np.array([[5], [3], [2]], dtype=np.float32)


# Forward propagation with transposition during calculation
Z1 = tf.matmul(tf.transpose(W1), X) + b1
A1 = tf.nn.relu(Z1)
Z2 = tf.matmul(tf.transpose(W2), A1) + b2
A2 = tf.nn.relu(Z2)
Z3 = tf.matmul(tf.transpose(W3), A2) + b3
A3 = tf.nn.sigmoid(Z3)
Z4 = tf.matmul(tf.transpose(W4), A3) + b4
output = tf.nn.sigmoid(Z4)

print("TensorFlow Output:", output.numpy())

TensorFlow Output: [[0.9999546]]
```

Figure 3: Forward Propagation Code using TensorFlow

Forward Propagation in PyTorch

```
✓ 8s  import torch
import torch.nn as nn

# Define weights and biases
W1 = torch.tensor([[2, 4, 6], [3, 5, 7], [4, 6, 8]], dtype=torch.float32)
b1 = torch.tensor([[1], [1], [1]], dtype=torch.float32)
W2 = torch.tensor([[3, 5], [4, 6], [7, 8]], dtype=torch.float32)
b2 = torch.tensor([[2], [2]], dtype=torch.float32)
W3 = torch.tensor([[5, 7], [6, 8]], dtype=torch.float32)
b3 = torch.tensor([[3], [3]], dtype=torch.float32)
W4 = torch.tensor([[4], [5]], dtype=torch.float32)
b4 = torch.tensor([[1]], dtype=torch.float32)

# Define input as column vector
X = torch.tensor([[5], [3], [2]], dtype=torch.float32)

# Forward propagation with transposition during calculation
Z1 = torch.matmul(torch.transpose(W1, 0, 1), X) + b1
A1 = torch.relu(Z1)
Z2 = torch.matmul(torch.transpose(W2, 0, 1), A1) + b2
A2 = torch.relu(Z2)
Z3 = torch.matmul(torch.transpose(W3, 0, 1), A2) + b3
A3 = torch.sigmoid(Z3)
Z4 = torch.matmul(torch.transpose(W4, 0, 1), A3) + b4
output = torch.sigmoid(Z4)

print("PyTorch Output:", output.numpy())
```


 PyTorch Output: [[0.9999546]]

Figure 4: Forward Propagation Code using PyTorch (Official Implementation)