GOVERNMENT OF KERALA

DEPARTMENT OF TECHNICAL EDUCATION

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(GOVT. ENGINEERING COLLEGE)

KOTTAYAM - 686501

RECORD BOOK

GOVERNMENT OF KERALA

DEPARTMENT OF TECHNICAL EDUCATION

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(GOVT. ENGINEERING COLLEGE)

KOTTAYAM - 686501

20MCA241 DATA SCIENCE LAB

Name: Vaishnav V

Branch: Master of Computer Applications

Semester: 3

Roll No: 51

CERTIFIED BONAFIDE RECORD WORK DONE BY

Reg No. ....

STAFF IN CHARGE

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Contents

Assignment 1
Review of python programming

Problem Statement

Write Python code to explore and practice with the basic data types, containers, functions, and classes of Python.

1. Start by creating variables of various numeric data types and assigning them values.
2. Print the data types and values of these variables.
3. Perform mathematical operations on these variables.
4. Update the values of these variables.
5. Create boolean variables with True or False values.
6. Print the data types of these boolean variables.
7. Perform Boolean operations on these boolean variables.
8. Create string variables with text values.
9. Print the contents and lengths of these string variables.
10. Concatenate strings.
11. Format strings with variables.
12. Use string methods to manipulate strings by capitalizing, converting to uppercase, justifying, centering, replacing substrings, and stripping whitespace.
13. Create and use Python lists. Perform tasks like appending elements, indexing, slicing, and iterating through the list.
14. Create and use Python tuples. Perform tasks like indexing, slicing, and concatenation.
15. Create and use Python sets. Perform tasks like accessing, adding, deleting set elements.
16. Create and use Python dictionaries. Perform tasks like adding, updating, and removing key-value pairs, and accessing values.
17. Define simple functions with parameters and return values.
18. Call functions with different arguments and use the returned results.
19. Write functions that accept other functions as arguments.

20. Define and use Python classes. Include tasks like creating a class, defining methods, and creating instances.

21. Implement class inheritance and method overriding.

22. Create a class with class variables and instance variables, and demonstrate their usage.

1.1 Basic data types

1.1.1 Numbers

```
# Your Python code here
print("Hello, world!")
print(x + 1)   # Addition
print(x - 1)   # Subtraction
print(x * 2)   # Multiplication
print(x ** 2)  # Exponentiation
```

```
Hello, world! 7   5   14  49
```

1.1.2 Booleans

```
t, f = True, False
print(type(t))
print(t and f) # Logical AND;
print(t or f)  # Logical OR;
print(not t)   # Logical NOT;
print(t != f)  # Logical XOR;
```

```
<class 'bool'>
False True False True
```

1.1.3 Strings

```
hello = 'hello'
world = "world"
print(hello, len(hello))
hw = hello + ' ' + world  # String concatenation
print(hw)
hw12 = '{} {} {}'.format(hello, world, 12)  # string formatting
print(hw12)
s = "hello"
print(s.capitalize())
print(s.upper())
```

```
hello 5
hello world
hello world 12
Hello
```

```
print(s.rjust(7))
print(s.center(7))
print(s.replace('l', '(ell)'))
print('  world '.strip())
```

```
HELLO
  hello
 hello
he(ell)(ell)o
world
```

1.2 Containers

1.2.1 Lists

```
xs = [3, 1, 2]
print(xs, xs[2])
print(xs[-1])
xs[2] = 'foo'
print(xs)
xs.append('bar')
print(xs)
x = xs.pop()
print(x, xs)
```

```
[3, 1, 2] 2
2
[3, 1, 'foo']
[3, 1, 'foo', 'bar']
foo [3, 1]
```

1.2.2 Slicing

```python
nums = list(range(5))
print(nums)
print(nums[2:4])
print(nums[2:])
print(nums[:2])
print(nums[:])
print(nums[:-1])
nums[2:4] = [8, 9] print(nums)
```

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[0, 1, 8, 9, 4]
```

1.2.3 Loops

```python
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
```

```
cat
dog
monkey
```

1.2.4 List comprehensions

```python
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)
```

```
[0, 1, 4, 9, 16]
```

1.2.5 Dictionaries

```python
d = {'cat': 'cute', 'dog': 'furry'}
print(d['cat'])
print('cat' in d)
d['fish'] = 'wet'
print(d['fish'])
```

```
cute
True
wet
```

1.2.6 Sets

```python
animals = {'cat', 'dog'}
print('cat' in animals)
print('fish' in animals)
animals.add('cat')
print(len(animals))
animals.remove('cat')
print(len(animals))
```

```
True
False
3
2
```

1.2.7 Tuples

```python
d = {(x, x + 1): x for x in range(10)}
t = (5, 6)
print(type(t))
print(d[t])
print(d[(1, 2)])
```

```
<class 'tuple'>
5
1
```

1.3 Functions

```python
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'
for x in [-1, 0, 1]:
    print(sign(x))
```

```
negative
zero
positive
```

1.4 Classes

```python
class Greeter:
    def __init__(self, name):
        self.name = name
    def greet(self, loud=False):
        if loud:
          print('HELLO, {}'.format(self.name.upper()))
        else:
          print('Hello, {}!'.format(self.name))
g = Greeter('Fred')
g.greet()
g.greet(loud=True)
```

```
Hello, Fred!
HELLO, FRED
```

1.5 Inheritance

```python
class Animal:
    def speak(self):
        return "Generic animal sound"

class Dog(Animal):
    def speak(self):
        return "Woof!"

class Cat(Animal):
    def speak(self):
        return "Meow!"

animal = Animal()
dog = Dog()
cat = Cat()

print(animal.speak())
print(dog.speak())
print(cat.speak())
```

```
Generic animal sound
Woof!
Meow!
```

1.6 instance variables

```python
class MyClass:
    class_var = "I am a class variable" # Class variable

    def __init__(self, instance_val):
        self.instance_var = instance_val # Instance variable

    def display_vars(self):
        print(f"Class variable: {MyClass.class_var}")
        print(f"Instance variable: {self.instance_var}")

# Demonstrate usage
obj1 = MyClass("Value for obj1")
obj2 = MyClass("Value for obj2")

obj1.display_vars()
obj2.display_vars()

# Accessing class variable through the class and instances
print(MyClass.class_var)
print(obj1.class_var)
obj2.class_var
```

```
Class variable: I am a class variable
Instance variable: Value for obj1
Class variable: I am a class variable
Instance variable: Value for obj2
I am a class variable
I am a class variable
I am a class variable
```

Assignment 2
Vectorized Computations using Numpy

Problem Statement

Implement the following computations using NumPy:

1. Create a matrix U of shape (m, n) with input values, where m and n are input positive integers.
2. Compute X as the transpose of U.
3. Create a matrix Y of shape $(1, m)$ with random values $\in [0, 1]$.
4. Create a matrix W_1 of shape (p, n) with random values $\in [0, 1]$, where p is an input positive integer.
5. Create a vector B_1 of shape $(p, 1)$ with random values $\in [0, 1]$.
6. Create a vector W_2 of shape $(1, p)$ with all zeros.
7. Create a scalar B2 with a random value $\in [0, 1]$.
8. Perform the following computations iteratively 15 times:
   (a) $Z_1 = W_1 \cdot X + B_1$ (Matrix multiplication)
   (b) $A_1 = f(Z_1)$, where f is a function that returns 0 for negative values and the input value itself otherwise (ReLU)
   (c) $Z_2 = W_2 \cdot A_1 + B_2$
   (d) $A_2 = g(Z_2)$, where $g(x) = \frac{1}{1+e^{-x}}$
   (e) $L = \frac{1}{2}(A_2 - Y)^2$
   (f) $dA_2 = A_2 - Y$
   (g) $dZ_2 = dA_2 \circ g'(Z_2)$, where $g'(x) = g(x) \cdot (1 - g(x))$
   (h) $dA_1 = W_2^\top \cdot dZ_2$
   (i) $dZ_1 = dA_1 \circ f'(Z_1)$, where:

$$f'(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

   (j) $dW_1 = \frac{1}{m} \cdot dZ_1 \cdot X^\top$
   (k) $dB_1 = \frac{1}{m} \sum dZ_1$ (sum along the columns)

(l) $dW_2 = \frac{1}{m} \cdot dZ_2 \cdot A_1^\top$

(m) $dB_2 = \frac{1}{m} \sum dZ_2$ (sum along the columns)

(n) Update and print W_1, B_1, W_2, and B_2 for $\alpha = 0.01$:

i. $W_1 = W_1 - \alpha \cdot dW_1$

ii. $B_1 = B_1 - \alpha \cdot dB_1$

iii. $W_2 = W_2 - \alpha \cdot dW_2$

iv. $B_2 = B_2 - \alpha \cdot dB_2$

Code and Outputs

1. **Program Code :**

```
import numpy as np
m = int(input("Enter number of rows (m): "))
n = int(input("Enter number of columns (n): "))
u = []
for i in range(m):
    row = input(f"Enter {n} elements for row {i+1}: ")
    row_elements = list(map(float, row.strip().split()))
    u.append(row_elements)
U = np.array(u)
print(U)
```

Output:

```
Enter number of rows (m):  2
Enter number of columns (n):  2
Enter 2 elements for row 1, separated by spaces:  1 2
Enter 2 elements for row 2, separated by spaces:  3 4
[[1. 2.]
 [3. 4.]]
```

2. **Program Code :**

```
print("transpose of u is:")
X=U.T
print(X)
```

Output:

```
transpose of u is:
[[1. 3.]
```

```
 [2. 4.]]
```

3. **Program Code :**

```python
Y = np.random.rand(1, m)
print(Y)
```

Output:

```
[[0.93419463 0.26522306]]
```

4. **Program Code :**

```python
p=int(input("enter p value:"))
W1=np.random.rand(p,n)
print(W1)
```

Output:

```
enter p value: 2
[[0.1101886  0.01550452]
 [0.87842457 0.99378785]]
```

5. **Program Code :**

```python
B1=np.random.rand(p,1)
print(B1)
```

Output:

```
[[0.32971822]
 [0.44886329]]
```

6. **Program Code :**

```python
W2=np.zeros((1,p))
print(W2)
```

Output:

```
[[0. 0.]]
```

7. **Program Code :**

```
B2=np.random.rand()
print(B2)
```

Output:

```
0.8918397872420092
```

8. **Program Code :**

```
def f(x):  # ReLU
    return np.maximum(0, x)

def fprime(x):
    return (x > 0).astype(float)

def g(x):  # Sigmoid
    return 1 / (1 + np.exp(-x))

def gprime(x):
    s = g(x)
    return s * (1 - s)

# Learning rate
alpha = 0.01
for iteration in range(15):
    # (a) Z1 = W1 * X + B1
    Z1 = W1 @ X + B1  # shape (p, m)

    # (b) A1 = f(Z1)
    A1 = f(Z1)  # ReLU output, shape (p, m)

    # (c) Z2 = W2 * A1 + B2
    Z2 = W2 @ A1 + B2  # shape (1, m)

    # (d) A2 = g(Z2)
    A2 = g(Z2)  # sigmoid output, shape (1, m)

    # (e) L = 1/2 * (A2 - Y)^2 (Loss per sample)
    L = 0.5 * np.square(A2 - Y)  # shape (1, m)

    # (f) dA2 = A2 - Y
```

```
    dA2 = A2 - Y  # shape (1, m)

    \# (g) dZ2 = dA2 * gprime(Z2)
    dZ2 = dA2 * gprime(Z2)  # element-wise multiplication, shape (1, m)

    \# (h) dA1 = W2.T * dZ2
    dA1 = W2.T @ dZ2  # shape (p, m)

    \# (i) dZ1 = dA1 . fprime(Z1)
    dZ1 = dA1 * fprime(Z1)  # shape (p, m)

    # (j) dW1 = 1/m * dZ1 * X.T
    dW1 = (1 / m) * dZ1 @ X.T  # shape (p, n)

    # (k) dB1 = 1/m * sum of dZ1 along columns
    dB1 = (1 / m) * np.sum(dZ1, axis=1, keepdims=True)  # shape (p, 1)

    # (l) dW2 = 1/m * dZ2 * A1.T
    dW2 = (1 / m) * dZ2 @ A1.T  # shape (1, p)

    # (m) dB2 = 1/m * sum of dZ2 along columns
    dB2 = (1 / m) * np.sum(dZ2, axis=1, keepdims=True)  # shape (1, 1), scalar but kept as array

    W1 -= alpha * dW1
    B1 -= alpha * dB1
    W2 -= alpha * dW2
    B2 -= alpha * dB2.item()  # Convert to scalar

    # Print updated parameters
    print(f"Iteration {iteration + 1}:")
    print("W1:\n", W1)
    print("B1:\n", B1)
    print("W2:", W2)
    print("B2:", B2)
    print("Loss:", np.mean(L))
    print("-" * 40)
```

Output:

```
Iteration 1:
W1:
 [[0.22823926 0.34764299]
 [0.0156025  0.52544597]]
B1:
 [[0.32971266]
 [0.44885914]]
```

```
W2: [[0.00441104 0.00326981]]
B2: 0.33647478663392744
Loss: 0.04972936057114144
----------------------------------------
Iteration 2:
W1:
 [[0.22824147 0.34764463]
 [0.01560414 0.52544719]]
B1:
 [[0.3297121 ]
 [0.44885873]]
W2: [[0.00461287 0.0034165 ]]
B2: 0.3363477117272872
Loss: 0.049721486757244575
----------------------------------------
Iteration 3:
W1:
 [[0.22824377 0.34764635]
 [0.01560584 0.52544846]]
B1:
 [[0.32971151]
 [0.44885829]]
W2: [[0.00481399 0.00356239]]
B2: 0.3362203029438675
Loss: 0.04971365639637108
----------------------------------------
Iteration 4:
W1:
 [[0.22824617 0.34764813]
 [0.01560762 0.52544978]]
B1:
 [[0.32971089]
 [0.44885783]]
W2: [[0.00501439 0.00370751]]
B2: 0.33609256228008216
Loss: 0.04970586898133045
----------------------------------------
Iteration 5:
```

```
W1:
 [[0.22824866 0.34764998]
 [0.01560946 0.52545115]]
B1:
 [[0.32971025]
 [0.44885736]]
W2: [[0.0052141  0.00385186]]
B2: 0.33596449171989246
Loss: 0.049698124011006276
----------------------------------------
Iteration 6:
W1:
 [[0.22825125 0.3476519 ]
 [0.01561137 0.52545257]]
B1:
 [[0.32970958]
 [0.44885687]]
W2: [[0.0054131  0.00399543]]
B2: 0.3358360932348894
Loss: 0.049690420990280926
----------------------------------------
Iteration 7:
W1:
 [[0.22825393 0.34765389]
 [0.01561335 0.52545403]]
B1:
 [[0.32970889]
 [0.44885635]]
W2: [[0.00561141 0.00413824]]
B2: 0.33570736878437524
Loss: 0.04968275942996133
----------------------------------------
Iteration 8:
W1:
 [[0.22825671 0.34765594]
 [0.0156154  0.52545555]]
B1:
 [[0.32970816]
```

```
 [0.44885582]]
W2: [[0.00580902 0.00428029]]
B2: 0.33557832031544466
Loss: 0.0496751388467056
----------------------------------------
Iteration 9:
W1:
 [[0.22825958 0.34765806]
 [0.01561752 0.52545711]]
B1:
 [[0.32970741]
 [0.44885526]]
W2: [[0.00600594 0.00442158]]
B2: 0.33544894976306516
Loss: 0.04966755876295083
----------------------------------------
Iteration 10:
W1:
 [[0.22826254 0.34766024]
 [0.0156197  0.52545872]]
B1:
 [[0.32970663]
 [0.44885469]]
W2: [[0.00620218 0.00456213]]
B2: 0.33531925905015725
Loss: 0.04966001870684142
----------------------------------------
Iteration 11:
W1:
 [[0.2282656  0.34766249]
 [0.01562194 0.52546037]]
B1:
 [[0.32970582]
 [0.4488541 ]]
W2: [[0.00639774 0.00470192]]
B2: 0.33518925008767375
Loss: 0.04965251821215874
----------------------------------------
```

```
Iteration 12:
W1:
 [[0.22826874 0.3476648 ]
 [0.01562426 0.52546207]]
B1:
 [[0.32970499]
 [0.44885348]]
W2: [[0.00659262 0.00484097]]
B2: 0.33505892477467886
Loss: 0.049645056818251515
----------------------------------------
Iteration 13:
W1:
 [[0.22827198 0.34766718]
 [0.01562663 0.52546381]]
B1:
 [[0.32970413]
 [0.44885285]]
W2: [[0.00678683 0.00497928]]
B2: 0.3349282849984264
Loss: 0.04963763406996691
----------------------------------------
Iteration 14:
W1:
 [[0.2282753  0.34766962]
 [0.01562907 0.5254656 ]]
B1:
 [[0.32970324]
 [0.4488522 ]]
W2: [[0.00698038 0.00511686]]
B2: 0.33479733263443795
Loss: 0.04963024951758298
----------------------------------------
Iteration 15:
W1:
 [[0.22827872 0.34767211]
 [0.01563158 0.52546743]]
B1:
```

```
 [[0.32970232]
 [0.44885153]]
W2: [[0.00717325 0.00525371]]
B2: 0.33466606954657985
Loss: 0.04962290271674142
----------------------------------------
```

Assignment 3
Vectorized Computations using TensorFlow

Problem Statement

Implement the following computations using NumPy:

1. Create a matrix U of shape (m, n) with input values, where m and n are input positive integers.
2. Compute X as the transpose of U.
3. Create a matrix Y of shape $(1, m)$ with random values $\in [0, 1]$.
4. Create a matrix W_1 of shape (p, n) with random values $\in [0, 1]$, where p is an input positive integer.
5. Create a vector B_1 of shape $(p, 1)$ with random values $\in [0, 1]$.
6. Create a vector W_2 of shape $(1, p)$ with all zeros.
7. Create a scalar B2 with a random value $\in [0, 1]$.
8. Perform the following computations iteratively 15 times:
   (a) $Z_1 = W_1 \cdot X + B_1$ (Matrix multiplication)
   (b) $A_1 = f(Z_1)$, where f is a function that returns 0 for negative values and the input value itself otherwise (ReLU)
   (c) $Z_2 = W_2 \cdot A_1 + B_2$
   (d) $A_2 = g(Z_2)$, where $g(x) = \frac{1}{1+e^{-x}}$
   (e) $L = \frac{1}{2}(A_2 - Y)^2$
   (f) $dA_2 = A_2 - Y$
   (g) $dZ_2 = dA_2 \circ g'(Z_2)$, where $g'(x) = g(x) \cdot (1 - g(x))$
   (h) $dA_1 = W_2^\top \cdot dZ_2$
   (i) $dZ_1 = dA_1 \circ f'(Z_1)$, where:

$$f'(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

   (j) $dW_1 = \frac{1}{m} \cdot dZ_1 \cdot X^\top$
   (k) $dB_1 = \frac{1}{m} \sum dZ_1$ (sum along the columns)

(l) $dW_2 = \frac{1}{m} \cdot dZ_2 \cdot A_1^\top$

(m) $dB_2 = \frac{1}{m} \sum dZ_2$ (sum along the columns)

(n) Update and print W_1, B_1, W_2, and B_2 for $\alpha = 0.01$:

i. $W_1 = W_1 - \alpha \cdot dW_1$

ii. $B_1 = B_1 - \alpha \cdot dB_1$

iii. $W_2 = W_2 - \alpha \cdot dW_2$

iv. $B_2 = B_2 - \alpha \cdot dB_2$

Code and Outputs

1. **Program Code :**

```python
import tensorflow as tf
m = int(input("enter row:"))
n = int(input("enter column:"))
U=tf.random.uniform([m,n],minval=1,maxval=10)
print(U)
```

Output:

```
enter row:3
enter column:4
tf.Tensor(
[[3.1435356 4.676051  9.81332   1.7113744]
 [4.332196  8.284798  9.341486  3.9649148]
 [8.549854  6.2711954 4.6171894 8.232712 ]], shape=(3, 4), dtype=float32)
```

2. **Program Code :**

```python
X = tf.transpose(U)
print(X)
```

Output:

```
tf.Tensor(
[[3.1435356 4.332196  8.549854 ]
 [4.676051  8.284798  6.2711954]
 [9.81332   9.341486  4.6171894]
 [1.7113744 3.9649148 8.232712 ]], shape=(4, 3), dtype=float32)
```

3. **Program Code :**

```
Y = np.random.rand(1, m)
printY = tf.random.uniform(shape=(1, m), minval=0, maxval=10, dtype=tf.int32)
print(Y)
```

Output:

```
tf.Tensor([[1 5 9]], shape=(1, 3), dtype=int32)
```

4. **Program Code :**

```
W1 = tf.Variable(tf.random.uniform(shape=(p, n), minval=0, maxval=1, dtype=tf.float32))
print(W1)
```

Output:

```
<tf.Variable 'Variable:0' shape=(6, 4) dtype=float32, numpy=
array([[0.5436977 , 0.72874427, 0.68472576, 0.94835377],
       [0.96630347, 0.9926599 , 0.03001082, 0.14628553],
       [0.05960095, 0.17020118, 0.12600875, 0.8665062 ],
       [0.46588528, 0.23078609, 0.10589218, 0.2996155 ],
       [0.31661344, 0.59627604, 0.35098183, 0.4152949 ],
       [0.20630908, 0.14224207, 0.982635  , 0.02675736]], dtype=float32)>
```

5. **Program Code :**

```
B1 = tf.Variable(tf.random.uniform(shape=(p, 1), minval=0, maxval=1, dtype=tf.float32))
print(B1)
```

Output:

```
<tf.Variable 'Variable:0' shape=(6, 1) dtype=float32, numpy=
array([[0.8493881 ],
       [0.08162391],
       [0.31380033],
       [0.9507289 ],
       [0.2724731 ],
```

```
        [0.89534795]], dtype=float32)>
```

6. **Program Code :**

```
W2 = tf.Variable(tf.zeros(shape=(10, p), dtype=tf.float32))
print(W2)
```

Output:

```
<tf.Variable ’Variable:0’ shape=(10, 6) dtype=float32, numpy=
array([[0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0.]], dtype=float32)>
```

7. **Program Code :**

```
B2=np.random.rand()
printB2 = tf.Variable(tf.random.uniform(shape=(), minval=0, maxval=1, ↩
    dtype=tf.float32))
print(B2)
```

Output:

```
<tf.Variable ’Variable:0’ shape=() dtype=float32, numpy=0.9369667768478394>
```

8. **Program Code :**

```
alpha = 0.01  # Learning rate
epochs = 15   # Iterations
for i in range(epochs):
    # a. Z1 = W1     X + B1
    Z1 = tf.matmul(W1, X) + B1

    # b. A1 = ReLU(Z1)
```

```
    A1 = tf.nn.relu(Z1)

    # c. Z2 = W2    A1 + B2
    Z2 = tf.matmul(W2, A1) + B2

    # d. A2 = softmax(Z2) along axis=0
    A2 = tf.nn.softmax(Z2, axis=0)

    # e. One-hot encode Y (shape (10, m))
    Y_one_hot = tf.one_hot(tf.squeeze(Y), depth=10, axis=0)

    # f. dZ2 = A2 - one_hot(Y)
    dZ2 = A2 - Y_one_hot

    # g. dA2 = W2^T    dZ2
    dA2 = tf.matmul(tf.transpose(W2), dZ2)

    # h. dW2 = 1/m * dZ2    A1^T
    dW2 = (1/m) * tf.matmul(dZ2, tf.transpose(A1))

    # i. dB2 = 1/m * sum(dZ2)
    dB2 = (1/m) * tf.reduce_sum(dZ2)

    # j. ReLU derivative
    relu_derivative = tf.cast(Z1 > 0, tf.float32)
    dZ1 = dA2 * relu_derivative

    # k. dA1 = W1^T    dZ1
    dA1 = tf.matmul(tf.transpose(W1), dZ1)

    # l. dB1 = 1/m * sum over columns
    dB1 = (1/m) * tf.reduce_sum(dZ1, axis=1, keepdims=True)

    # m. dW1 = 1/m * dZ1    X^T
    dW1 = (1/m) * tf.matmul(dZ1, tf.transpose(X))

    # Gradient descent update
    W1.assign_sub(alpha * dW1)
    B1.assign_sub(alpha * dB1)
    W2.assign_sub(alpha * dW2)
    B2.assign_sub(alpha * dB2)

    # Print updates
    print(f"\nIteration {i + 1}")
    print("W1:\n", W1.numpy())
    print("B1:\n", B1.numpy())
    print("W2:\n", W2.numpy())
    print("B2:\n", B2.numpy())
```

Output:

```
Iteration 1
W1:
 [[0.5436977  0.72874427 0.68472576 0.94835377]
 [0.96630347 0.9926599  0.03001082 0.14628553]
 [0.05960095 0.17020118 0.12600875 0.8665062 ]
 [0.46588528 0.23078609 0.10589218 0.2996155 ]
 [0.31661344 0.59627604 0.35098183 0.4152949 ]
 [0.20630908 0.14224207 0.982635   0.02675736]]
B1:
 [[0.8493881 ]
 [0.08162391]
 [0.31380033]
 [0.9507289 ]
 [0.2724731 ]
 [0.89534795]]
W2:
 [[-0.01824814 -0.01252315 -0.00673919 -0.00714713 -0.01049294 -0.01081975]
 [ 0.02944714  0.01516291  0.00664917  0.00967398  0.01687705  0.02883933]
 [-0.01824814 -0.01252315 -0.00673919 -0.00714713 -0.01049294 -0.01081975]
 [-0.01824814 -0.01252315 -0.00673919 -0.00714713 -0.01049294 -0.01081975]
 [-0.01824814 -0.01252315 -0.00673919 -0.00714713 -0.01049294 -0.01081975]
 [ 0.04641449  0.03198412  0.01524353  0.01638018  0.02787183  0.03002335]
 [-0.01824814 -0.01252315 -0.00673919 -0.00714713 -0.01049294 -0.01081975]
 [-0.01824814 -0.01252315 -0.00673919 -0.00714713 -0.01049294 -0.01081975]
 [-0.01824814 -0.01252315 -0.00673919 -0.00714713 -0.01049294 -0.01081975]
 [ 0.05187537  0.04051497  0.02528163  0.02397573  0.02870169  0.01687554]]
B2:
 0.9369668

Iteration 2
W1:
 [[0.5440847  0.7291323  0.68503964 0.9487317 ]
 [0.96658623 0.99287343 0.0300517  0.14658767]
 [0.05977688 0.170277   0.12593713 0.86670125]
 [0.46606034 0.23089719 0.10591274 0.29979736]
 [0.3168252  0.59651226 0.35118473 0.41550112]
```

```
 [0.20647387 0.1426236  0.9832724  0.02685763]]
B1:
 [[0.8494403 ]
 [0.08164761]
 [0.313808  ]
 [0.95074356]
 [0.27250382]
 [0.8954082 ]]
W2:
 [[-0.02150786 -0.01467208 -0.00786347 -0.00838606 -0.01236975 -0.01295926]
 [ 0.04989494  0.02434149  0.01017301  0.01592855  0.02855928  0.05185796]
 [-0.02150786 -0.01467208 -0.00786347 -0.00838606 -0.01236975 -0.01295926]
 [-0.02150786 -0.01467208 -0.00786347 -0.00838606 -0.01236975 -0.01295926]
 [-0.02150786 -0.01467208 -0.00786347 -0.00838606 -0.01236975 -0.01295926]
 [ 0.05202485  0.03580823  0.01530089  0.01673463  0.0322702   0.03618712]
 [-0.02150786 -0.01467208 -0.00786347 -0.00838606 -0.01236975 -0.01295926]
 [-0.02150786 -0.01467208 -0.00786347 -0.00838606 -0.01236975 -0.01295926]
 [-0.02150786 -0.01467208 -0.00786347 -0.00838606 -0.01236975 -0.01295926]
 [ 0.04863521  0.04255482  0.02957039  0.02603926  0.02575875  0.00266969]]
B2:
 0.9369668

Iteration 3
W1:
 [[0.54436487 0.7295368  0.6856369  0.94895   ]
 [0.9669562  0.99322605 0.03033764 0.14694698]
 [0.06009676 0.17048019 0.12601657 0.8670231 ]
 [0.46629772 0.23107494 0.10606664 0.30002058]
 [0.31694847 0.596751   0.35154873 0.41559023]
 [0.20614487 0.14270681 0.983958   0.02642284]]
B1:
 [[0.8495066 ]
 [0.08169664]
 [0.3138382 ]
 [0.95077336]
 [0.27254036]
 [0.89543575]]
W2:
```

```
 [[-0.02378823 -0.01616677 -0.00864297 -0.0092498  -0.01368246 -0.01447512]
 [ 0.0420111   0.01402709  0.00322986  0.01110312  0.02392759  0.05813773]
 [-0.02378823 -0.01616677 -0.00864297 -0.0092498  -0.01368246 -0.01447512]
 [-0.02378823 -0.01616677 -0.00864297 -0.0092498  -0.01368246 -0.01447512]
 [-0.02378823 -0.01616677 -0.00864297 -0.0092498  -0.01368246 -0.01447512]
 [ 0.05502487  0.03782367  0.0143731   0.01605421  0.0351713   0.04085137]
 [-0.02378823 -0.01616677 -0.00864297 -0.0092498  -0.01368246 -0.01447512]
 [-0.02378823 -0.01616677 -0.00864297 -0.0092498  -0.01368246 -0.01447512]
 [-0.02378823 -0.01616677 -0.00864297 -0.0092498  -0.01368246 -0.01447512]
 [ 0.0694816   0.06131657  0.04289781  0.03759124  0.03667834  0.00233668]]
B2:
 0.9369668

Iteration 4
W1:
 [[0.54456615 0.7296166  0.6855571  0.9491705 ]
 [0.9670649  0.9930576  0.02972147 0.1471456 ]
 [0.06015258 0.17020564 0.12535027 0.8671558 ]
 [0.46637133 0.23091623 0.1056752  0.30013505]
 [0.3170603  0.5968633  0.35159475 0.4157107 ]
 [0.20632197 0.14342077 0.9854092  0.02643069]]
B1:
 [[0.8495157 ]
 [0.08165764]
 [0.31379238]
 [0.9507499 ]
 [0.27255175]
 [0.8955589 ]]
W2:
 [[-0.02536496 -0.01718486 -0.00916773 -0.00984013 -0.01459073 -0.01555978]
 [ 0.06067901  0.02228434  0.00641039  0.01682881  0.03455279  0.07933678]
 [-0.02536496 -0.01718486 -0.00916773 -0.00984013 -0.01459073 -0.01555978]
 [-0.02536496 -0.01718486 -0.00916773 -0.00984013 -0.01459073 -0.01555978]
 [-0.02536496 -0.01718486 -0.00916773 -0.00984013 -0.01459073 -0.01555978]
 [ 0.06793095  0.047042    0.01752629  0.01947974  0.04372374  0.05035991]
 [-0.02536496 -0.01718486 -0.00916773 -0.00984013 -0.01459073 -0.01555978]
 [-0.02536496 -0.01718486 -0.00916773 -0.00984013 -0.01459073 -0.01555978]
 [-0.02536496 -0.01718486 -0.00916773 -0.00984013 -0.01459073 -0.01555978]
```

```
 [ 0.04894472  0.05096768  0.04023744  0.03257233  0.0238586  -0.02077828]]
B2:
 0.9369668

Iteration 5
W1:
 [[0.54446983 0.7297462  0.68589073 0.94904184]
 [0.9674453  0.99343157 0.029893   0.14754917]
 [0.06071552 0.17058392 0.12548393 0.8677301 ]
 [0.46670476 0.23115642 0.10583855 0.30045897]
 [0.3168777  0.59688705 0.35175148 0.41551864]
 [0.20494483 0.14272706 0.98565483 0.02495259]]
B1:
 [[0.84953296]
 [0.0816977 ]
 [0.31384468]
 [0.95078754]
 [0.27254885]
 [0.89547443]]
W2:
 [[-0.02663331 -0.01800724 -0.00959386 -0.01031731 -0.01532073 -0.01642304]
 [ 0.04898913  0.00964089 -0.00164447  0.01066934  0.02770217  0.08264242]
 [-0.02663331 -0.01800724 -0.00959386 -0.01031731 -0.01532073 -0.01642304]
 [-0.02663331 -0.01800724 -0.00959386 -0.01031731 -0.01532073 -0.01642304]
 [-0.02663331 -0.01800724 -0.00959386 -0.01031731 -0.01532073 -0.01642304]
 [ 0.0490137   0.03380131  0.0082994   0.01011308  0.03403436  0.04254838]
 [-0.02663331 -0.01800724 -0.00959386 -0.01031731 -0.01532073 -0.01642304]
 [-0.02663331 -0.01800724 -0.00959386 -0.01031731 -0.01532073 -0.01642304]
 [-0.02663331 -0.01800724 -0.00959386 -0.01031731 -0.01532073 -0.01642304]
 [ 0.08843033  0.08260843  0.06050207  0.05143874  0.04550854 -0.01022954]]
B2:
 0.9369668

Iteration 6
W1:
 [[0.54421353 0.7290388  0.6848433  0.9488654 ]
 [0.96697074 0.9923117  0.02798015 0.14727342]
 [0.06020464 0.16939694 0.1235977  0.8673955 ]
```

```
 [0.46634153 0.23028271 0.10452887 0.3002042 ]
 [0.31682476 0.5967054  0.3514299  0.4154978 ]
 [0.20569806 0.14428818 0.9884229  0.02539183]]
B1:
 [[0.8494365 ]
 [0.08151804]
 [0.3136653 ]
 [0.9506631 ]
 [0.2725208 ]
 [0.89574105]]
W2:
 [[-0.02758269 -0.01860692 -0.00989785 -0.0106671  -0.01586794 -0.01710733]
 [ 0.07026246  0.02002005  0.00282962  0.01758609  0.03979877  0.10456118]
 [-0.02758269 -0.01860692 -0.00989785 -0.0106671  -0.01586794 -0.01710733]
 [-0.02758269 -0.01860692 -0.00989785 -0.0106671  -0.01586794 -0.01710733]
 [-0.02758269 -0.01860692 -0.00989785 -0.0106671  -0.01586794 -0.01710733]
 [ 0.08831793  0.06144477  0.02147724  0.02400715  0.05775957  0.06698877]
 [-0.02758269 -0.01860692 -0.00989785 -0.0106671  -0.01586794 -0.01710733]
 [-0.02758269 -0.01860692 -0.00989785 -0.0106671  -0.01586794 -0.01710733]
 [-0.02758269 -0.01860692 -0.00989785 -0.0106671  -0.01586794 -0.01710733]
 [ 0.03449841  0.04878358  0.04497809  0.03307643  0.01351726 -0.05179865]]
B2:
 0.9369668

Iteration 7
W1:
 [[0.54302347 0.72829604 0.684333   0.9477258 ]
 [0.96689343 0.9923197  0.02767759 0.14728399]
 [0.06092195 0.16992201 0.12380151 0.86813194]
 [0.4666424  0.23051909 0.10467075 0.30050385]
 [0.3158323  0.59606874 0.35092968 0.41456085]
 [0.20257539 0.14217022 0.98752165 0.02224048]]
B1:
 [[0.8493061 ]
 [0.08148628]
 [0.313734  ]
 [0.9506963 ]
 [0.27240613]
```

```
 [0.89543796]]
W2:
 [[-0.02824281 -0.01902939 -0.010115   -0.01091341 -0.01624781 -0.01756909]
 [ 0.06654714  0.01310556 -0.00205909  0.01465033  0.03753265  0.11200816]
 [-0.02824281 -0.01902939 -0.010115   -0.01091341 -0.01624781 -0.01756909]
 [-0.02824281 -0.01902939 -0.010115   -0.01091341 -0.01624781 -0.01756909]
 [-0.02824281 -0.01902939 -0.010115   -0.01091341 -0.01624781 -0.01756909]
 [ 0.03471732  0.02397028 -0.0009962   0.00082714  0.02816684  0.03969237]
 [-0.02824281 -0.01902939 -0.010115   -0.01091341 -0.01624781 -0.01756909]
 [-0.02824281 -0.01902939 -0.010115   -0.01091341 -0.01624781 -0.01756909]
 [-0.02824281 -0.01902939 -0.010115   -0.01091341 -0.01624781 -0.01756909]
 [ 0.09643514  0.09612986  0.07386026  0.06091638  0.04803514 -0.02871693]]
B2:
 0.9369668

Iteration 8
W1:
 [[0.54246366 0.72703856 0.6828151  0.94723463]
 [0.9662355  0.99081403 0.02535841 0.14683256]
 [0.06019149 0.16829821 0.12139438 0.8676032 ]
 [0.46605137 0.22921829 0.10289096 0.3000379 ]
 [0.31568065 0.5957041  0.35049978 0.41442525]
 [0.2032713  0.14374001 0.9905854  0.02255804]]
B1:
 [[0.8491543 ]
 [0.08126398]
 [0.3135003 ]
 [0.9505209 ]
 [0.27236387]
 [0.8957241 ]]
W2:
 [[-0.02892415 -0.01945556 -0.0103295  -0.01116276 -0.01664057 -0.01806988]
 [ 0.07020423  0.01201735 -0.00340854  0.01504506  0.03941161  0.12185399]
 [-0.02892415 -0.01945556 -0.0103295  -0.01116276 -0.01664057 -0.01806988]
 [-0.02892415 -0.01945556 -0.0103295  -0.01116276 -0.01664057 -0.01806988]
 [-0.02892415 -0.01945556 -0.0103295  -0.01116276 -0.01664057 -0.01806988]
 [ 0.08880774  0.06156588  0.01743004  0.02039561  0.06042088  0.07340805]
 [-0.02892415 -0.01945556 -0.0103295  -0.01116276 -0.01664057 -0.01806988]
```

```
 [-0.02892415 -0.01945556 -0.0103295  -0.01116276 -0.01664057 -0.01806988]
 [-0.02892415 -0.01945556 -0.0103295  -0.01116276 -0.01664057 -0.01806988]
 [ 0.04345703  0.06260569  0.05828495  0.04269861  0.0166515  -0.06877295]]
B2:
 0.9369668

Iteration 9
W1:
 [[0.5415247  0.7264723  0.68241304 0.9463413 ]
 [0.9665     0.99105006 0.02511411 0.14719701]
 [0.06129592 0.1690763  0.12172379 0.86872387]
 [0.46665335 0.22965361 0.10316045 0.3006296 ]
 [0.31474468 0.59511226 0.3500033  0.41355047]
 [0.19970281 0.14134848 0.9896464  0.01894275]]
B1:
 [[0.8490511 ]
 [0.0812621 ]
 [0.31360787]
 [0.9505868 ]
 [0.27225354]
 [0.8953848 ]]
W2:
 [[-0.029511   -0.01983119 -0.01052256 -0.01138173 -0.01697828 -0.01848031]
 [ 0.06822014  0.00642733 -0.00751743  0.01286176  0.03812703  0.1299833 ]
 [-0.029511   -0.01983119 -0.01052256 -0.01138173 -0.01697828 -0.01848031]
 [-0.029511   -0.01983119 -0.01052256 -0.01138173 -0.01697828 -0.01848031]
 [-0.029511   -0.01983119 -0.01052256 -0.01138173 -0.01697828 -0.01848031]
 [ 0.03646701  0.02498666 -0.00453142 -0.0022603   0.03154057  0.04676023]
 [-0.029511   -0.01983119 -0.01052256 -0.01138173 -0.01697828 -0.01848031]
 [-0.029511   -0.01983119 -0.01052256 -0.01138173 -0.01697828 -0.01848031]
 [-0.029511   -0.01983119 -0.01052256 -0.01138173 -0.01697828 -0.01848031]
 [ 0.10188979  0.1074043   0.08570674  0.06907065  0.04918036 -0.04738141]]
B2:
 0.9369668

Iteration 10
W1:
 [[0.5408759  0.72508544 0.6807365  0.94577277]
```

```
 [0.96565    0.98924756 0.0223654  0.14659499]
 [0.06032266 0.16704345 0.11877286 0.8679947 ]
 [0.46588698 0.22805344 0.10100104 0.30001444]
 [0.31459942 0.594781   0.3496062  0.41342235]
 [0.2007812  0.14357159 0.9936414  0.01955996]]
B1:
 [[0.8488808 ]
 [0.080993  ]
 [0.31331548]
 [0.9503702 ]
 [0.27221397]
 [0.89576936]]
W2:
 [[-0.03008321 -0.02018627 -0.01070006 -0.01158985 -0.01730827 -0.01890765]
 [ 0.07295451  0.00633978 -0.00821404  0.01380793  0.04060707  0.1398281 ]
 [-0.03008321 -0.02018627 -0.01070006 -0.01158985 -0.01730827 -0.01890765]
 [-0.03008321 -0.02018627 -0.01070006 -0.01158985 -0.01730827 -0.01890765]
 [-0.03008321 -0.02018627 -0.01070006 -0.01158985 -0.01730827 -0.01890765]
 [ 0.09076723  0.06275165  0.01399934  0.01740499  0.06391215  0.080535  ]
 [-0.03008321 -0.02018627 -0.01070006 -0.01158985 -0.01730827 -0.01890765]
 [-0.03008321 -0.02018627 -0.01070006 -0.01158985 -0.01730827 -0.01890765]
 [-0.03008321 -0.02018627 -0.01070006 -0.01158985 -0.01730827 -0.01890765]
 [ 0.0468607   0.07221244  0.0691151   0.04991601  0.01663865 -0.0880096 ]]
B2:
 0.9369668

Iteration 11
W1:
 [[0.5399688  0.72453207 0.68033326 0.94491154]
 [0.9661409  0.9896521  0.02218279 0.14719337]
 [0.06176664 0.1680632  0.11925951 0.8694467 ]
 [0.46672073 0.2286476  0.10139    0.30082735]
 [0.31359196 0.5941376  0.34904775 0.41248482]
 [0.19655208 0.14068383 0.9924138  0.01528791]]
B1:
 [[0.84878   ]
 [0.08101254]
 [0.31346074]
```

```
 [0.95046294]
 [0.2720933 ]
 [0.8953586 ]]
W2:
 [[-3.0577136e-02 -2.0502195e-02 -1.0862343e-02 -1.1774052e-02
  -1.7592518e-02 -1.9253636e-02]
 [ 6.9718137e-02  1.8911436e-05 -1.2650739e-02  1.1206380e-02
   3.8587943e-02  1.4688736e-01]
 [-3.0577136e-02 -2.0502195e-02 -1.0862343e-02 -1.1774052e-02
  -1.7592518e-02 -1.9253636e-02]
 [-3.0577136e-02 -2.0502195e-02 -1.0862343e-02 -1.1774052e-02
  -1.7592518e-02 -1.9253636e-02]
 [-3.0577136e-02 -2.0502195e-02 -1.0862343e-02 -1.1774052e-02
  -1.7592518e-02 -1.9253636e-02]
 [ 3.9565910e-02  2.6947603e-02 -7.5424276e-03 -4.8029479e-03
   3.5684243e-02  5.4571651e-02]
 [-3.0577136e-02 -2.0502195e-02 -1.0862343e-02 -1.1774052e-02
  -1.7592518e-02 -1.9253636e-02]
 [-3.0577136e-02 -2.0502195e-02 -1.0862343e-02 -1.1774052e-02
  -1.7592518e-02 -1.9253636e-02]
 [-3.0577136e-02 -2.0502195e-02 -1.0862343e-02 -1.1774052e-02
  -1.7592518e-02 -1.9253636e-02]
 [ 1.0475588e-01  1.1654881e-01  9.6229553e-02  7.6014899e-02
   4.8875418e-02 -6.6683590e-02]]
B2:
 0.9369668

Iteration 12
W1:
 [[0.5393303  0.7231805  0.67871255 0.9443487 ]
 [0.9652373  0.98777163 0.01930565 0.14655292]
 [0.06067017 0.16580652 0.11602376 0.8686132 ]
 [0.46586597 0.22688438 0.09903852 0.3001335 ]
 [0.31349677 0.5939145  0.3487785  0.41240132]
 [0.19787967 0.14336914 0.9970492  0.01610302]]
B1:
 [[0.8486146 ]
 [0.08072957]
```

```
 [0.31313813]
 [0.9502257 ]
 [0.2720667 ]
 [0.89580953]]
W2:
 [[-0.03108736 -0.02081736 -0.01101914 -0.01195885 -0.01788689 -0.0196383 ]
 [ 0.07292172 -0.00087409 -0.01366197  0.01168513  0.0401637   0.15520842]
 [-0.03108736 -0.02081736 -0.01101914 -0.01195885 -0.01788689 -0.0196383 ]
 [-0.03108736 -0.02081736 -0.01101914 -0.01195885 -0.01788689 -0.0196383 ]
 [-0.03108736 -0.02081736 -0.01101914 -0.01195885 -0.01788689 -0.0196383 ]
 [ 0.09279409  0.06401701  0.01063557  0.01446605  0.06743419  0.08760698]
 [-0.03108736 -0.02081736 -0.01101914 -0.01195885 -0.01788689 -0.0196383 ]
 [-0.03108736 -0.02081736 -0.01101914 -0.01195885 -0.01788689 -0.0196383 ]
 [-0.03108736 -0.02081736 -0.01101914 -0.01195885 -0.01788689 -0.0196383 ]
 [ 0.05189565  0.08257857  0.08016034  0.05756071  0.01761032 -0.10534731]]
B2:
 0.9369668

Iteration 13
W1:
 [[0.53850067 0.72266775 0.67829555 0.9435711 ]
 [0.9658835  0.988254   0.01905915 0.14732693]
 [0.06238842 0.16700055 0.11658952 0.8703384 ]
 [0.46691504 0.22762088 0.09952141 0.3011546 ]
 [0.31244072 0.59322596 0.34813514 0.41142884]
 [0.19324295 0.14022903 0.99575526 0.01141409]]
B1:
 [[0.8485185 ]
 [0.08075564]
 [0.3133102 ]
 [0.95034206]
 [0.27193567]
 [0.8953631 ]]
W2:
 [[-0.03151779 -0.02109226 -0.01116013 -0.01211914 -0.01813465 -0.01994086]
 [ 0.07308393 -0.00476204 -0.01675039  0.01045995  0.04009466  0.1640435 ]
 [-0.03151779 -0.02109226 -0.01116013 -0.01211914 -0.01813465 -0.01994086]
 [-0.03151779 -0.02109226 -0.01116013 -0.01211914 -0.01813465 -0.01994086]
```

```
 [-0.03151779 -0.02109226 -0.01116013 -0.01211914 -0.01813465 -0.01994086]
 [ 0.03972107  0.02697465 -0.01152234 -0.00842446  0.03810911  0.06037893]
 [-0.03151779 -0.02109226 -0.01116013 -0.01211914 -0.01813465 -0.01994086]
 [-0.03151779 -0.02109226 -0.01116013 -0.01211914 -0.01813465 -0.01994086]
 [-0.03151779 -0.02109226 -0.01116013 -0.01211914 -0.01813465 -0.01994086]
 [ 0.1078195   0.12543316  0.10639361  0.08279843  0.04873872 -0.08483642]]
B2:
 0.9369668

Iteration 14
W1:
 [[0.5378311  0.72126913 0.6766497  0.9429732 ]
 [0.9649589  0.98634005 0.01615    0.146666  ]
 [0.06120392 0.16457646 0.1131702  0.869423  ]
 [0.46597752 0.2257017  0.09700566 0.30038217]
 [0.31237292 0.59306353 0.34794652 0.4113675 ]
 [0.19465855 0.14309706 1.0006347  0.0123005 ]]
B1:
 [[0.8483492 ]
 [0.08046882]
 [0.31296778]
 [0.95008683]
 [0.27191707]
 [0.89583856]]
W2:
 [[-0.0319625  -0.02136588 -0.01129568 -0.01227961 -0.01839134 -0.02027887]
 [ 0.0722108  -0.00814562 -0.01894066  0.00951892  0.03928887  0.16915812]
 [-0.0319625  -0.02136588 -0.01129568 -0.01227961 -0.01839134 -0.02027887]
 [-0.0319625  -0.02136588 -0.01129568 -0.01227961 -0.01839134 -0.02027887]
 [-0.0319625  -0.02136588 -0.01129568 -0.01227961 -0.01839134 -0.02027887]
 [ 0.09346053  0.06436759  0.00681808  0.01103149  0.07015482  0.09378393]
 [-0.0319625  -0.02136588 -0.01129568 -0.01227961 -0.01839134 -0.02027887]
 [-0.0319625  -0.02136588 -0.01129568 -0.01227961 -0.01839134 -0.02027887]
 [-0.0319625  -0.02136588 -0.01129568 -0.01227961 -0.01839134 -0.02027887]
 [ 0.05806613  0.09333915  0.09119229  0.06540683  0.01929564 -0.12098998]]
B2:
 0.9369668
```

```
Iteration 15
W1:
 [[0.53716016 0.72086126 0.67627966 0.9423546 ]
 [0.96575385 0.9868942  0.01584487 0.14760675]
 [0.06312533 0.16588475 0.11377048 0.8713521 ]
 [0.4672057  0.22654887 0.0975593  0.301576  ]
 [0.31134313 0.5923907  0.34727433 0.41043004]
 [0.18987103 0.13991404 0.9994435  0.00744072]]
B1:
 [[0.8482687 ]
 [0.08050141]
 [0.31315798]
 [0.9502223 ]
 [0.2717856 ]
 [0.8953886 ]]
W2:
 [[-0.03235386 -0.02161551 -0.01142349 -0.01242512 -0.01861667 -0.02055485]
 [ 0.07504897 -0.0101071  -0.02095571  0.00938393  0.04075382  0.17936324]
 [-0.03235386 -0.02161551 -0.01142349 -0.01242512 -0.01861667 -0.02055485]
 [-0.03235386 -0.02161551 -0.01142349 -0.01242512 -0.01861667 -0.02055485]
 [-0.03235386 -0.02161551 -0.01142349 -0.01242512 -0.01861667 -0.02055485]
 [ 0.04136546  0.02813231 -0.01479971 -0.011366    0.04135526  0.0667384 ]
 [-0.03235386 -0.02161551 -0.01142349 -0.01242512 -0.01861667 -0.02055485]
 [-0.03235386 -0.02161551 -0.01142349 -0.01242512 -0.01861667 -0.02055485]
 [-0.03235386 -0.02161551 -0.01142349 -0.01242512 -0.01861667 -0.02055485]
 [ 0.1100626   0.13328335  0.1157198   0.08895791  0.0482076  -0.10221774]]
B2:
 0.9369668
```

Implementing a Fully Connected Neural Network (FCNN) from Scratch Using TensorFlow

Problem Statement

Implement the following computations using TensorFlow on the MNIST dataset:

1. Load the MNIST dataset, consisting of 60,000 training and 10,000 test grayscale images of size 28×28 pixels and labels from 0 to 9.
2. Flatten each image into a 784-dimensional vector and form matrix U of shape (m, n) with $m = 60000$ and $n = 784$.
3. Compute $X = U^T$ and normalize pixel values to $[0, 1]$.
4. Form label matrix Y of shape $(1, m)$ from training labels.
5. Similarly, prepare X_{test} and Y_{test} from the test dataset.
6. Display one training image and its label.
7. Define hyperparameters: hidden layer size $p = 10$, output size $q = 10$, learning rate $\alpha = 0.01$, and epochs = 1000.
8. Initialize weights and biases:
   - $W_1 \in \mathbb{R}^{p\times n} \sim \mathcal{N}(0,1) \times \frac{\sqrt{q}}{n}$
   - $B_1 \in \mathbb{R}^{p\times 1} = 0$
   - $W_2 \in \mathbb{R}^{q\times p} \sim \mathcal{N}(0,1) \times \frac{\sqrt{q}}{p}$
   - $B_2 \in \mathbb{R}^{q\times 1} = 0$
9. Perform forward and backward propagation with ReLU and softmax activations for 1000 epochs:
   (a) $Z_1 = W_1X + B_1$
   (b) $A_1 = \mathrm{ReLU}(Z_1)$
   (c) $Z_2 = W_2A_1 + B_2$
   (d) $A_2 = \mathrm{softmax}(Z_2)$
   (e) Compute cross-entropy loss and accuracy every 100 epochs.
   (f) Backpropagate errors to compute gradients dW_1, dB_1, dW_2, dB_2.
   (g) Update parameters using learning rate α.

10. Repeat training using TensorFlow's `GradientTape` for automatic differentiation.

11. Test the trained model on one test image and display the prediction.

12. Evaluate overall accuracy on the entire test dataset.

4.1 Implementation and Computations

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

# Load MNIST data
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Prepare training data
m_train = x_train.shape[0]
n = 28 * 28
U = tf.reshape(x_train, shape=(m_train, n))
X = tf.transpose(U)
X = tf.cast(X, tf.float32) / 255.0
Y = tf.reshape(y_train, shape=(1, m_train))

# Prepare test data
m_test = x_test.shape[0]
V = tf.reshape(x_test, shape=(m_test, n))
Xtest = tf.transpose(V)
Xtest = tf.cast(Xtest, tf.float32) / 255.0
Ytest = tf.reshape(y_test, shape=(1, m_test))

# Display one training image and label
idx = 0
image = tf.reshape(X[:, idx], shape=(28, 28))
plt.imshow(image, cmap='gray')
plt.title(f"Label: {Y[0, idx].numpy()}")
plt.show()

# Hyperparameters
p = 10   # hidden neurons
q = 10   # output neurons (digits 0-9)
alpha = 0.01
epochs = 1000

# Initialize weights and biases
W1 = tf.Variable(tf.random.normal(shape=(p, n), mean=0.0, stddev=1.0) * (q ** 0.5 / n))
B1 = tf.Variable(tf.zeros(shape=(p, 1)))
W2 = tf.Variable(tf.random.normal(shape=(q, p), mean=0.0, stddev=1.0) * (q ** 0.5 / p))
```

```
B2 = tf.Variable(tf.zeros(shape=(q, 1)))

# Activation functions
def relu(Z):
    return tf.maximum(Z, 0)

def relu_derivative(Z):
    return tf.cast(Z > 0, tf.float32)

def softmax(Z):
    exp_Z = tf.exp(Z - tf.reduce_max(Z, axis=0, keepdims=True))
    return exp_Z / tf.reduce_sum(exp_Z, axis=0, keepdims=True)

def one_hot_T(Y, num_classes=10):
    Y_flat = tf.reshape(Y, [-1])
    oh = tf.one_hot(Y_flat, depth=num_classes)
    return tf.transpose(oh)

m = m_train

# Training loop (manual gradients)
for epoch in range(1, epochs + 1):
    with tf.GradientTape() as tape:
        Z1 = tf.matmul(W1, X) + B1
        A1 = relu(Z1)
        Z2 = tf.matmul(W2, A1) + B2
        A2 = softmax(Z2)
        Y_onehot = one_hot_T(Y, q)
        loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=↩
            tf.transpose(Y_onehot), logits=tf.transpose(Z2)))

    # Backprop manually
    dZ2 = A2 - Y_onehot
    dW2 = (1/m) * tf.matmul(dZ2, A1, transpose_b=True)
    dB2 = (1/m) * tf.reduce_sum(dZ2, axis=1, keepdims=True)
    dA1 = tf.matmul(W2, dZ2, transpose_a=True)
    dZ1 = dA1 * relu_derivative(Z1)
    dW1 = (1/m) * tf.matmul(dZ1, X, transpose_b=True)
    dB1 = (1/m) * tf.reduce_sum(dZ1, axis=1, keepdims=True)

    # Update weights
    W1.assign_sub(alpha * dW1)
    B1.assign_sub(alpha * dB1)
    W2.assign_sub(alpha * dW2)
    B2.assign_sub(alpha * dB2)

    if epoch % 100 == 0:
        preds = tf.argmax(A2, axis=0, output_type=tf.int32)
        correct = tf.equal(preds, tf.cast(Y[0], tf.int32))
```

```
        accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))
        print(f"Epoch {epoch}, Loss: {loss.numpy():.4f}, Accuracy: {accuracy.numpy()*100:.2f}%")

print("\nTraining with GradientTape automatic differentiation...")

# Re-initialize weights for GradientTape training
W1.assign(tf.random.normal(shape=(p, n), mean=0.0, stddev=1.0) * (q ** 0.5 / n))
B1.assign(tf.zeros(shape=(p, 1)))
W2.assign(tf.random.normal(shape=(q, p), mean=0.0, stddev=1.0) * (q ** 0.5 / p))
B2.assign(tf.zeros(shape=(q, 1)))

for epoch in range(1, epochs + 1):
    with tf.GradientTape() as tape:
        Z1 = tf.matmul(W1, X) + B1
        A1 = relu(Z1)
        Z2 = tf.matmul(W2, A1) + B2
        A2 = softmax(Z2)
        Y_onehot = one_hot_T(Y, q)
        loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=tf.transpose(Y_onehot), logits=tf.transpose(Z2)))

    grads = tape.gradient(loss, [W1, B1, W2, B2])
    W1.assign_sub(alpha * grads[0])
    B1.assign_sub(alpha * grads[1])
    W2.assign_sub(alpha * grads[2])
    B2.assign_sub(alpha * grads[3])

    if epoch % 100 == 0:
        preds = tf.argmax(A2, axis=0, output_type=tf.int32)
        correct = tf.equal(preds, tf.cast(Y[0], tf.int32))
        accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))
        print(f"Epoch {epoch}, Loss: {loss.numpy():.4f}, Accuracy: {accuracy.numpy()*100:.2f}%")

# Test on one test image
test_idx = 0
test_image = tf.reshape(Xtest[:, test_idx], (n,1))
plt.imshow(tf.reshape(test_image, (28,28)), cmap='gray')
plt.title(f"True Label: {Ytest[0, test_idx].numpy()}")
plt.show()

Z1_test = tf.matmul(W1, test_image) + B1
A1_test = relu(Z1_test)
Z2_test = tf.matmul(W2, A1_test) + B2
A2_test = softmax(Z2_test)
pred_label = tf.argmax(A2_test, axis=0).numpy()
```

```python
print(f"Predicted Label: {pred_label}")
print("Correct prediction!" if pred_label == Ytest[0, test_idx].numpy() else "Incorrect prediction.")

# Test accuracy on full test set
Z1_test_all = tf.matmul(W1, Xtest) + B1
A1_test_all = relu(Z1_test_all)
Z2_test_all = tf.matmul(W2, A1_test_all) + B2
A2_test_all = softmax(Z2_test_all)
preds_test = tf.argmax(A2_test_all, axis=0, output_type=tf.int32)
correct_test = tf.equal(preds_test, tf.cast(Ytest[0], tf.int32))
accuracy_test = tf.reduce_mean(tf.cast(correct_test, tf.float32))
print(f"Test Set Accuracy: {accuracy_test.numpy()*100:.2f}%")
```

```
Epoch 100, Loss: 1.7792, Accuracy: 53.91%
Epoch 200, Loss: 1.4631, Accuracy: 65.27%
Epoch 300, Loss: 1.2204, Accuracy: 71.58%
Epoch 400, Loss: 1.0432, Accuracy: 75.02%
Epoch 500, Loss: 0.9166, Accuracy: 77.23%
Epoch 600, Loss: 0.8239, Accuracy: 79.22%
Epoch 700, Loss: 0.7536, Accuracy: 80.71%
Epoch 800, Loss: 0.6985, Accuracy: 81.88%
Epoch 900, Loss: 0.6540, Accuracy: 82.86%
Epoch 1000, Loss: 0.6172, Accuracy: 83.88%

Predicted Label: 7
Correct prediction!
Test Set Accuracy: 84.88%
```