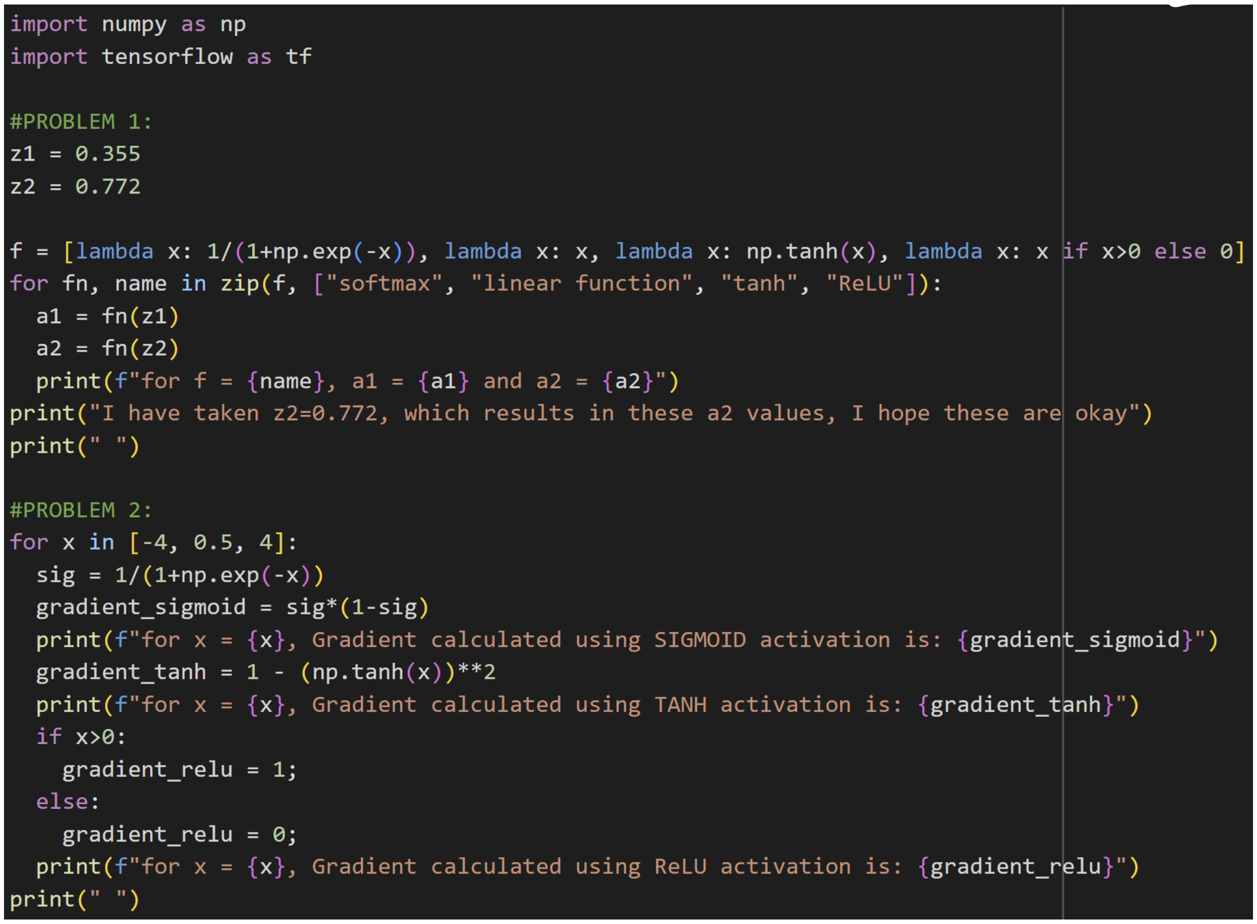
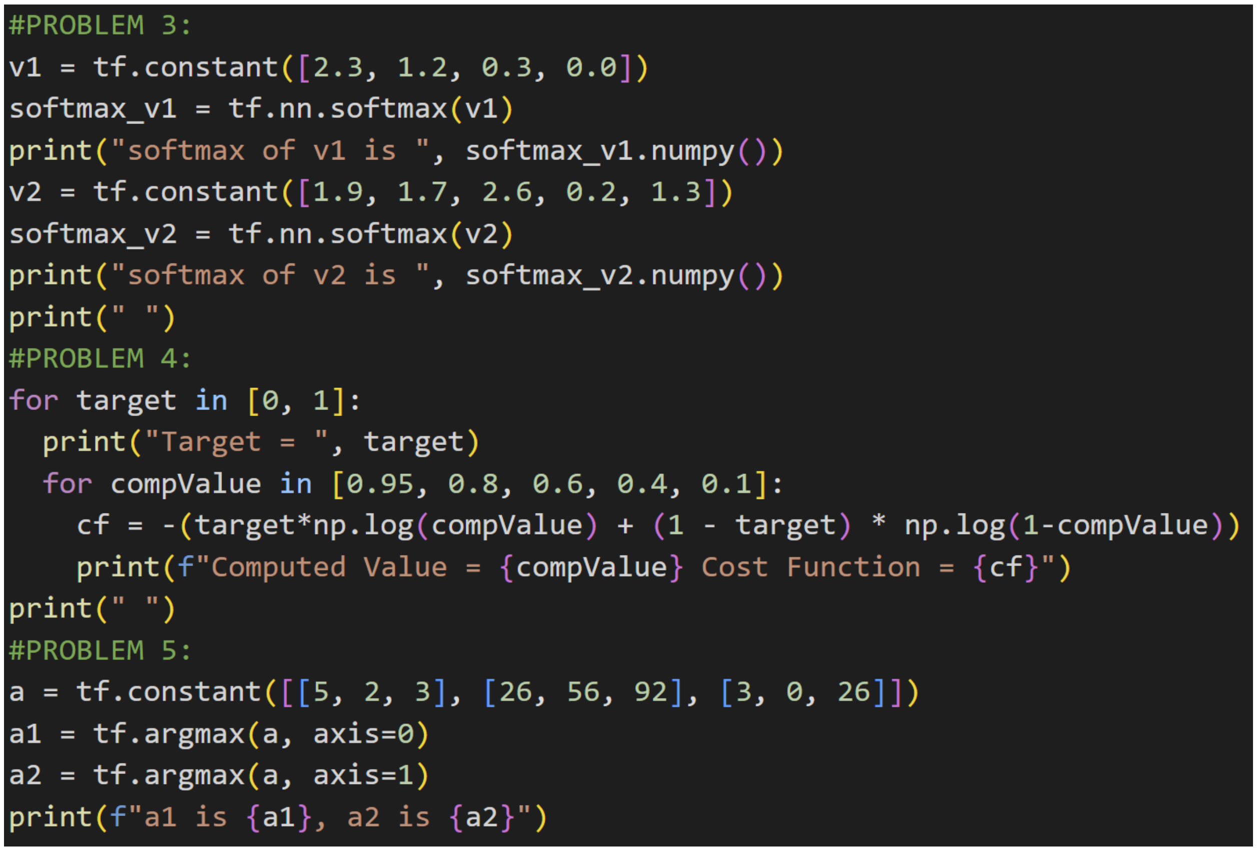
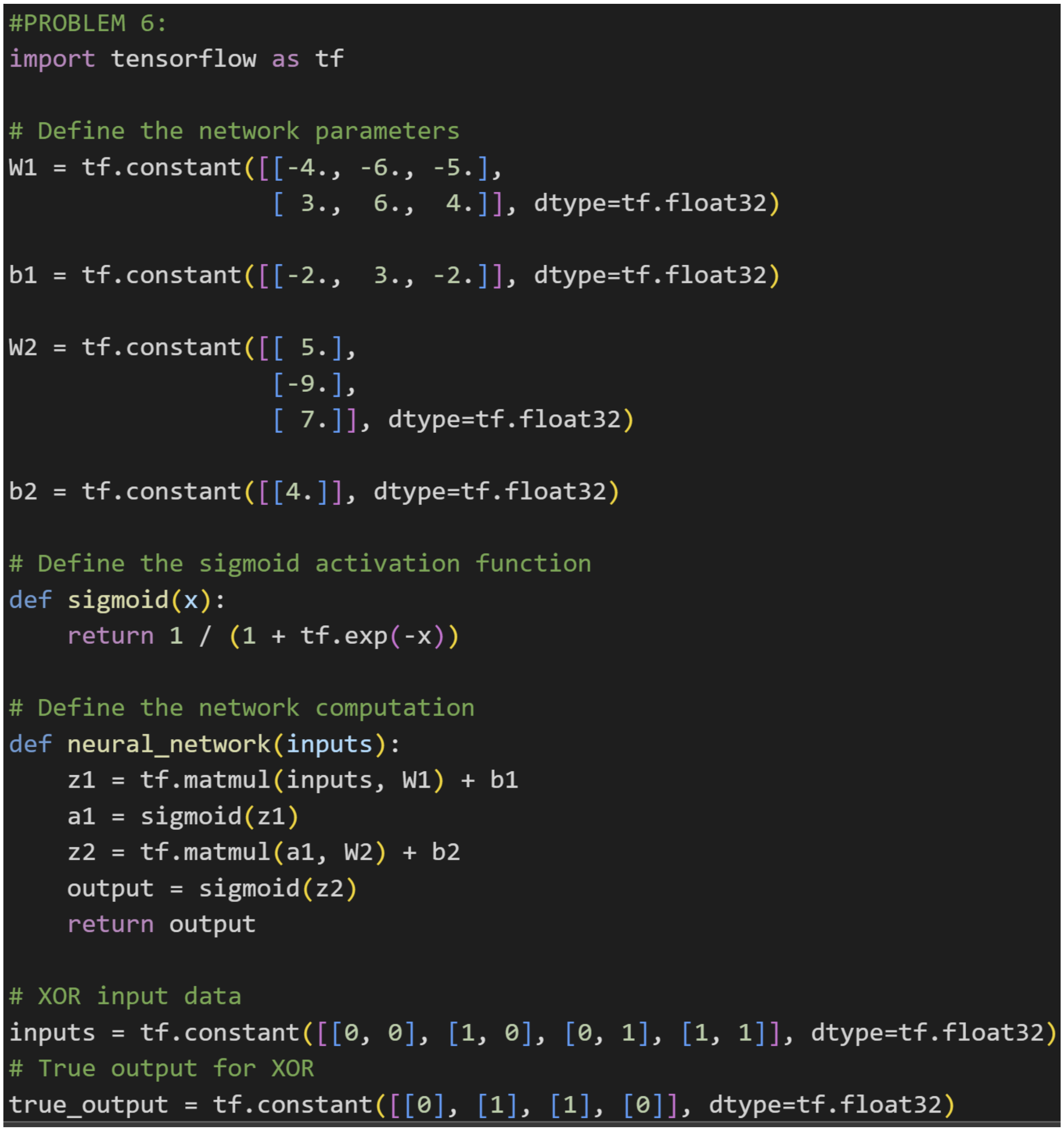
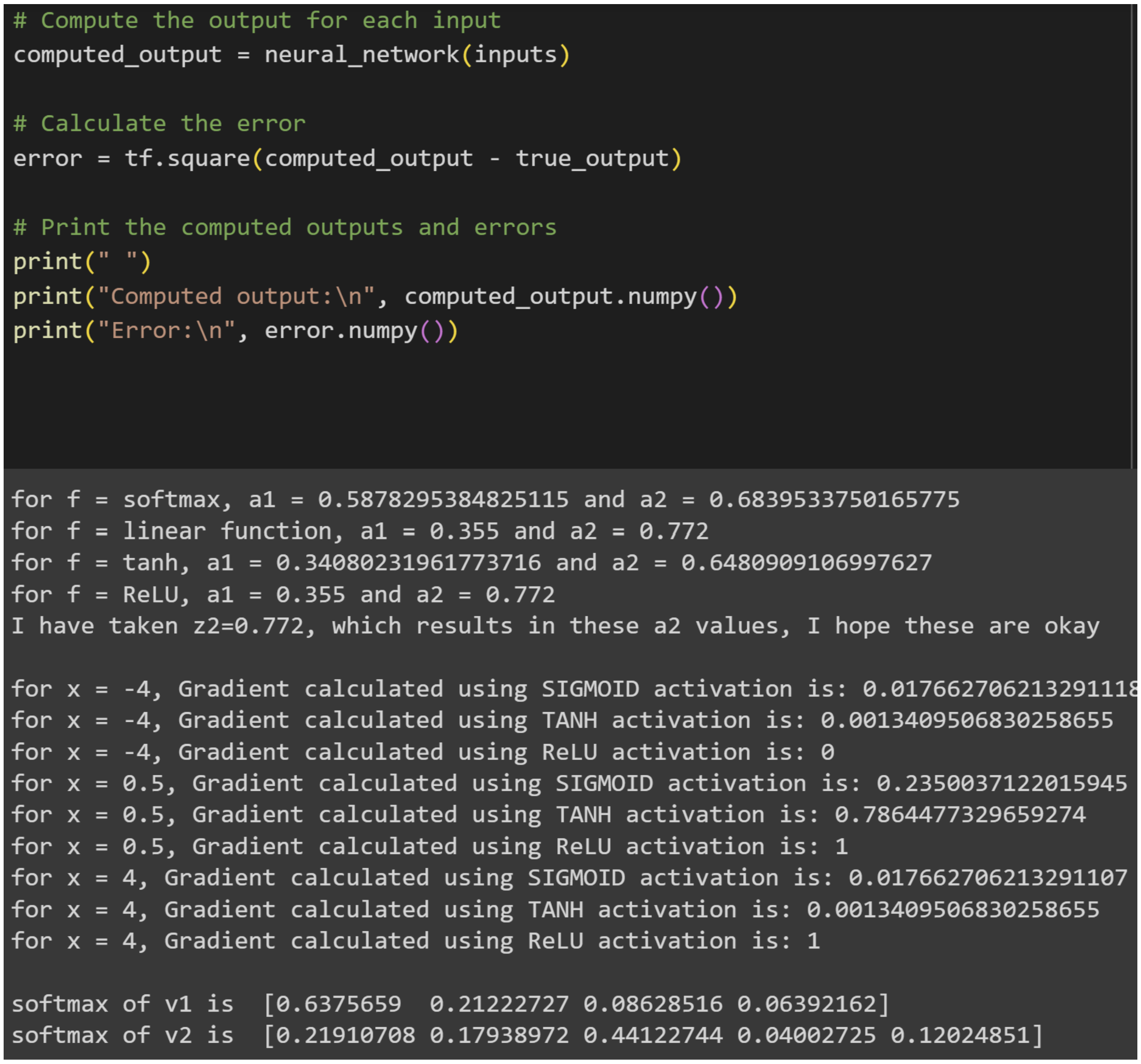


```python
import numpy as np
import tensorflow as tf

#PROBLEM 1:
z1 = 0.355
z2 = 0.772

f = [lambda x: 1/(1+np.exp(-x)), lambda x: x, lambda x: np.tanh(x), lambda x: x if x>0 else 0]
for fn, name in zip(f, ["softmax", "linear function", "tanh", "ReLU"]):
  a1 = fn(z1)
  a2 = fn(z2)
  print(f"for f = {name}, a1 = {a1} and a2 = {a2}")
print("I have taken z2=0.772, which results in these a2 values, I hope these are okay")
print(" ")

#PROBLEM 2:
for x in [-4, 0.5, 4]:
  sig = 1/(1+np.exp(-x))
  gradient_sigmoid = sig*(1-sig)
  print(f"for x = {x}, Gradient calculated using SIGMOID activation is: {gradient_sigmoid}")
  gradient_tanh = 1 - (np.tanh(x))**2
  print(f"for x = {x}, Gradient calculated using TANH activation is: {gradient_tanh}")
  if x>0:
    gradient_relu = 1;
  else:
    gradient_relu = 0;
  print(f"for x = {x}, Gradient calculated using ReLU activation is: {gradient_relu}")
print(" ")
```


```python
#PROBLEM 3:
v1 = tf.constant([2.3, 1.2, 0.3, 0.0])
softmax_v1 = tf.nn.softmax(v1)
print("softmax of v1 is ", softmax_v1.numpy())
v2 = tf.constant([1.9, 1.7, 2.6, 0.2, 1.3])
softmax_v2 = tf.nn.softmax(v2)
print("softmax of v2 is ", softmax_v2.numpy())
print(" ")
#PROBLEM 4:
for target in [0, 1]:
  print("Target = ", target)
  for compValue in [0.95, 0.8, 0.6, 0.4, 0.1]:
    cf = -(target*np.log(compValue) + (1 - target) * np.log(1-compValue))
    print(f"Computed Value = {compValue} Cost Function = {cf}")
print(" ")
#PROBLEM 5:
a = tf.constant([[5, 2, 3], [26, 56, 92], [3, 0, 26]])
a1 = tf.argmax(a, axis=0)
a2 = tf.argmax(a, axis=1)
print(f"a1 is {a1}, a2 is {a2}")
```

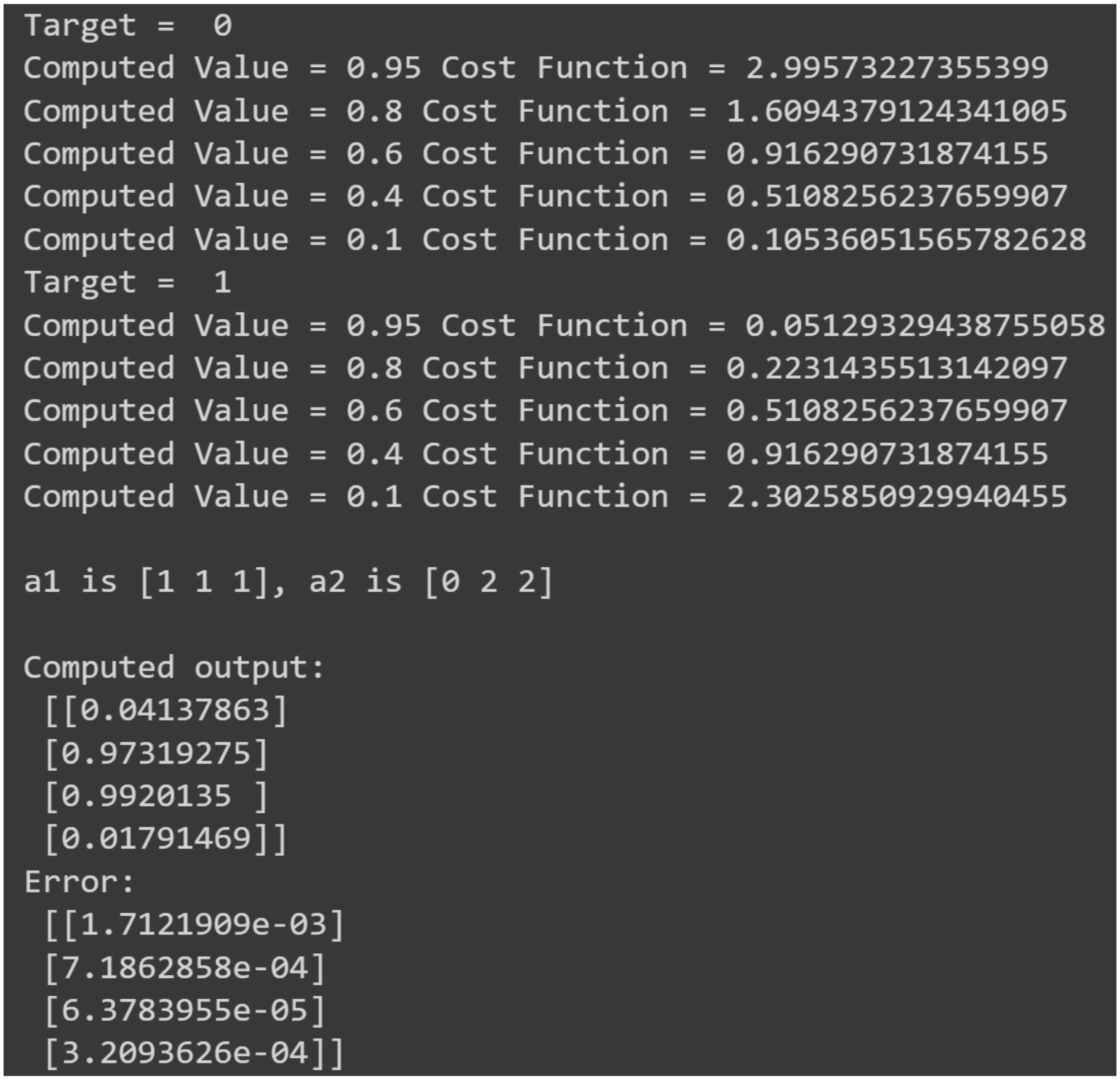


```
#PROBLEM 6:
import tensorflow as tf

# Define the network parameters
W1 = tf.constant([[-4., -6., -5.],
                  [ 3.,  6.,  4.]], dtype=tf.float32)

b1 = tf.constant([[-2.,  3., -2.]], dtype=tf.float32)

W2 = tf.constant([[ 5.],
                  [-9.],
                  [ 7.]], dtype=tf.float32)

b2 = tf.constant([[4.]], dtype=tf.float32)

# Define the sigmoid activation function
def sigmoid(x):
    return 1 / (1 + tf.exp(-x))

# Define the network computation
def neural_network(inputs):
    z1 = tf.matmul(inputs, W1) + b1
    a1 = sigmoid(z1)
    z2 = tf.matmul(a1, W2) + b2
    output = sigmoid(z2)
    return output

# XOR input data
inputs = tf.constant([[0, 0], [1, 0], [0, 1], [1, 1]], dtype=tf.float32)
# True output for XOR
true_output = tf.constant([[0], [1], [1], [0]], dtype=tf.float32)
```



```
# Compute the output for each input
computed_output = neural_network(inputs)

# Calculate the error
error = tf.square(computed_output - true_output)

# Print the computed outputs and errors
print(" ")
print("Computed output:\n", computed_output.numpy())
print("Error:\n", error.numpy())
```

```
for f = softmax, a1 = 0.5878295384825115 and a2 = 0.6839533750165775
for f = linear function, a1 = 0.355 and a2 = 0.772
for f = tanh, a1 = 0.34080231961773716 and a2 = 0.6480909106997627
for f = ReLU, a1 = 0.355 and a2 = 0.772
I have taken z2=0.772, which results in these a2 values, I hope these are okay

for x = -4, Gradient calculated using SIGMOID activation is: 0.01766270621329111
for x = -4, Gradient calculated using TANH activation is: 0.0013409506830258655
for x = -4, Gradient calculated using ReLU activation is: 0
for x = 0.5, Gradient calculated using SIGMOID activation is: 0.2350037122015945
for x = 0.5, Gradient calculated using TANH activation is: 0.7864477329659274
for x = 0.5, Gradient calculated using ReLU activation is: 1
for x = 4, Gradient calculated using SIGMOID activation is: 0.017662706213291107
for x = 4, Gradient calculated using TANH activation is: 0.0013409506830258655
for x = 4, Gradient calculated using ReLU activation is: 1

softmax of v1 is  [0.6375659  0.21222727 0.08628516 0.06392162]
softmax of v2 is  [0.21910708 0.17938972 0.44122744 0.04002725 0.12024851]
```



```
Target =  0
Computed Value = 0.95 Cost Function = 2.995732273553991
Computed Value = 0.8 Cost Function = 1.6094379124341005
Computed Value = 0.6 Cost Function = 0.916290731874155
Computed Value = 0.4 Cost Function = 0.5108256237659907
Computed Value = 0.1 Cost Function = 0.10536051565782628
Target =  1
Computed Value = 0.95 Cost Function = 0.05129329438755058
Computed Value = 0.8 Cost Function = 0.2231435513142097
Computed Value = 0.6 Cost Function = 0.5108256237659907
Computed Value = 0.4 Cost Function = 0.916290731874155
Computed Value = 0.1 Cost Function = 2.3025850929940455

a1 is [1 1 1], a2 is [0 2 2]

Computed output:
 [[0.04137863]
 [0.97319275]
 [0.9920135 ]
 [0.01791469]]
Error:
 [[1.7121909e-03]
 [7.1862858e-04]
 [6.3783955e-05]
 [3.2093626e-04]]
```