LAB ASSIGNMENT - 1

Mrinal Bhan
DSAI : *211020428*

Q1. Explore the CPU & GPU-based installation method of the TensorFlow library.

```
In [27]: import tensorflow as tf
         print(tf.__version__)
```

```
2.13.0
```

Q2. Create 1-D, 2-D,3-D & an n-dimensional tensor using different method(constant(), zeros(),ones(),fill()) & display the result.

```
import tensorflow as tf

tensor_1d = tf.constant([1, 2, 3, 4, 5])
tensor_2d = tf.zeros(shape=(2, 2))
tensor_3d = tf.ones(shape=(2, 3, 4))
n = 5  # Specify the desired dimension
tensor_nd = tf.fill(dims=[2] * n, value=9)

print("1D Tensor:")
print(tensor_1d)
print("2D Tensor:")
print(tensor_2d)
print("3D Tensor:")
print(tensor_3d)
print(f"{n}-D Tensor:")
print(tensor_nd)
```

```
1D Tensor:
tf.Tensor([1 2 3 4 5], shape=(5,), dtype=int32)
2D Tensor:
tf.Tensor(
[[0. 0.]
 [0. 0.]], shape=(2, 2), dtype=float32)
3D Tensor:
tf.Tensor(
[[[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]

 [[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]], shape=(2, 3, 4), dtype=float32)
5-D Tensor:
tf.Tensor(
[[[[[9 9]
    [9 9]]

   [[9 9]
    [9 9]]]


  [[[9 9]
    [9 9]]

   [[9 9]
    [9 9]]]], shape=(2, 2, 2, 2, 2), dtype=int32)
```

LAB ASSIGNMENT - 1

```
[[[[9 9]
   [9 9]]

  [[9 9]
   [9 9]]]


 [[[9 9]
   [9 9]]

  [[9 9]
   [9 9]]]], shape=(2, 2, 2, 2, 2), dtype=int32)
```

Q3. Find the rank/degree of the tensors.

In [6]:
```
rank_1d = tf.rank(tensor_1d)
rank_2d = tf.rank(tensor_2d)
rank_3d = tf.rank(tensor_3d)
rank_nd = tf.rank(tensor_nd)

print("Rank of 1D Tensor:", rank_1d.numpy())
print("Rank of 2D Tensor:", rank_2d.numpy())
print("Rank of 3D Tensor:", rank_3d.numpy())
print("Rank of {n}-D Tensor:", rank_nd.numpy())
```

```
Rank of 1D Tensor: 1
Rank of 2D Tensor: 2
Rank of 3D Tensor: 3
Rank of {n}-D Tensor: 5
```

Q4. Find out the shape of the tensor using the TensorFlow library.

In [8]:
```
shape_1d = tf.shape(tensor_1d)
shape_2d = tf.shape(tensor_2d)
shape_3d = tf.shape(tensor_3d)
shape_nd = tf.shape(tensor_nd)

print("Shape of 1D Tensor:", shape_1d.numpy())
print("Shape of 2D Tensor:", shape_2d.numpy())
print("Shape of 3D Tensor:", shape_3d.numpy())
print("Shape of {n}-D Tensor:", shape_nd.numpy())
```

```
Shape of 1D Tensor: [5]
Shape of 2D Tensor: [2 2]
Shape of 3D Tensor: [2 3 4]
Shape of {n}-D Tensor: [2 2 2 2 2]
```

Q5. Reshape the tensor using the TensorFlow library.

```
reshaped_2d = tf.reshape(tensor_2d, shape=(1, 4))
print("Reshaped 2D Tensor:")
print(reshaped_2d)
```

```
Reshaped 2D Tensor:
tf.Tensor([[0. 0. 0. 0.]], shape=(1, 4), dtype=float32)
```

LAB ASSIGNMENT - 1

Q6. Perform Slicing on the Tensors.

In [11]:
```
sliced_1d = tensor_1d[1:4]
print("Sliced 1D Tensor:")
print(sliced_1d)
```

```
Sliced 1D Tensor:
tf.Tensor([2 3 4], shape=(3,), dtype=int32)
```

Q7. Perform various mathematical operations on the tensor (matmul(). add(), subtract(), transpose() etc)

In [12]:
```
matmul_result = tf.matmul(tensor_2d, tensor_2d)
add_result = tf.add(tensor_1d, tensor_1d)
subtract_result = tf.subtract(tensor_1d, tensor_1d)
transpose_result = tf.transpose(tensor_2d)

print("Matrix Multiplication Result:")
print(matmul_result)
print("Addition Result:")
print(add_result)
print("Subtraction Result:")
print(subtract_result)
print("Transposed 2D Tensor:")
print(transpose_result)
```

```
Matrix Multiplication Result:
tf.Tensor(
[[0. 0.]
 [0. 0.]], shape=(2, 2), dtype=float32)
Addition Result:
tf.Tensor([ 2  4  6  8 10], shape=(5,), dtype=int32)
Subtraction Result:
tf.Tensor([0 0 0 0 0], shape=(5,), dtype=int32)
Transposed 2D Tensor:
tf.Tensor(
[[0. 0.]
 [0. 0.]], shape=(2, 2), dtype=float32)
```

LAB ASSIGNMENT - 1

Q8 and 9. Perform mathematical computation on tensor through constant, variable & placeholder methods.Perform computation on the tensor through building a graph, run the session & then
evaluate results.

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

# Create a constant tensor
const_tensor = tf.constant([[1, 2], [3, 4]])

# Create a variable tensor
var_tensor = tf.Variable([[5, 6], [7, 8]])

# Create a placeholder (input) tensor
placeholder_tensor = tf.placeholder(tf.int32, shape=(None, 2))

# Define a simple computation graph
result_tensor = const_tensor + var_tensor * placeholder_tensor

# Start a TensorFlow session
with tf.Session() as sess:
    # Initialize variables
    sess.run(tf.global_variables_initializer())

    # Define placeholder values
    input_data = [[0.1, 0.2], [0.3, 0.4]]

    # Evaluate the result tensor
    result = sess.run(result_tensor, feed_dict={placeholder_tensor: input_data})

    # Print the results
    print("Constant Tensor:\n", sess.run(const_tensor))
    print("Variable Tensor:\n", sess.run(var_tensor))
    print("Input Placeholder Data:\n", input_data)
    print("Result Tensor:\n", result)
```

```
Constant Tensor:
 [[1 2]
 [3 4]]
Variable Tensor:
 [[5 6]
 [7 8]]
Input Placeholder Data:
 [[0.1, 0.2], [0.3, 0.4]]
Result Tensor:
 [[1 2]
 [3 4]]
```

LAB ASSIGNMENT - 1

Q10. Prepare your own dataset from your input data using the TensorFlow library.

```
import tensorflow as tf
import numpy as np

# Input data
features = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
labels = np.array([0, 1, 0, 1])

# Create a TensorFlow dataset
dataset = tf.data.Dataset.from_tensor_slices((features, labels))

# Print dataset elements
for element in dataset:
    features, label = element
    print("Features:", features.numpy(), "Label:", label.numpy())
```

```
Features: [1 2 3] Label: 0
Features: [4 5 6] Label: 1
Features: [7 8 9] Label: 0
```

LAB ASSIGNMENT - 1

Q11.Apply different transformations techniques(as_numpy_iterator, apply, batch, cardinality, etc.) to a prepared dataset.

```
batch_size = 2

# 1. as_numpy_iterator: Convert dataset to a NumPy iterator
numpy_iterator = dataset.as_numpy_iterator()
for element in numpy_iterator:
    features, label = element
    print("as_numpy_iterator - Features:", features, "Label:", label)

# 2. apply: Apply a map transformation to square the values
def square_values(features, label):
    return features ** 2, label

applied_dataset = dataset.apply(tf.data.experimental.map_and_batch(map_func=square_values, batch_size=batch_size))

# 3. batch: Batch the dataset
batched_dataset = dataset.batch(batch_size)

# 4. cardinality: Get the cardinality of the dataset
num_samples = dataset.cardinality()
print("Cardinality of the dataset:", num_samples.numpy())

# 5. shuffle: Shuffle the dataset
shuffled_dataset = dataset.shuffle(buffer_size=len(dataset))

# 6. repeat: Repeat the dataset for multiple epochs
num_epochs=3
repeated_dataset = dataset.repeat(num_epochs)

# Print transformed dataset elements
for epoch in range(num_epochs):
    print(f"Epoch {epoch+1}:")
    for batch_features, batch_labels in repeated_dataset:
        print("Features:", batch_features.numpy())
        print("Labels:", batch_labels.numpy())
```

```
as_numpy_iterator - Features: [1 2 3] Label: 0
as_numpy_iterator - Features: [4 5 6] Label: 1
as_numpy_iterator - Features: [7 8 9] Label: 0
Cardinality of the dataset: 3
Epoch 1:
Features: [1 2 3]
Labels: 0
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
Features: [1 2 3]
Labels: 0
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
Features: [1 2 3]
Labels: 0
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
Epoch 2:
Features: [1 2 3]
Labels: 0
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
Features: [1 2 3]
Labels: 0
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
Features: [1 2 3]
Labels: 0
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
```

LAB ASSIGNMENT - 1

```
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
Features: [1 2 3]
Labels: 0
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
Features: [1 2 3]
Labels: 0
Features: [4 5 6]
Labels: 1
Features: [7 8 9]
Labels: 0
```