Tensor Creation Techniques

Creating tensors

When working with TensorFlow, there are times when we need to create tensors with specific characteristics. Tensors are the fundamental building blocks of TensorFlow, and having the ability to create them in various forms is essential for effective model development.
For example, we might want to create a tensor where all the elements are 1, or one where the elements form a sequence. Additionally, we may need- to generate a tensor whose elements are random.
In this lecture, we will explore some built-in methods in TensorFlow that make these tasks straightforward and efficient.

Tensor Creation Techniques

Built-in methods for tensor creation

1. All-Zero Tensor

In this example, we first import The TensorFlow library. Next, we create a session which is used to execute operations in the computational graph. Then, we create a tensor A, filled with zeros. The shape of this tensor is [2, 3], meaning it has two rows and three columns.
We can observe the output of this code is a 2×3 matrix where all the elements are zero.

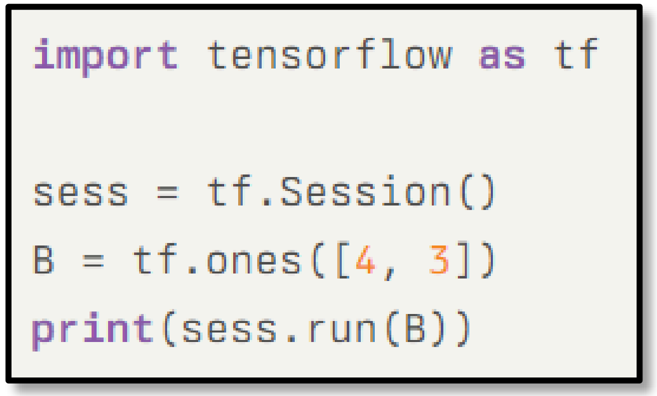
```
import tensorflow as tf

sess = tf.Session()
A = tf.zeros([2, 3])
print(sess.run(A))
```

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

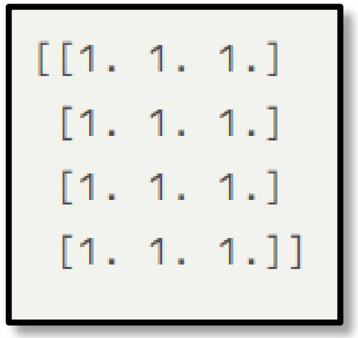
Built-in methods for tensor creation

2. All-One Tensor

In this example, after importing the TensorFlow library, we create a session. Next, we create a tensor B, filled with ones. The shape of this tensor is [4, 3], meaning it has four rows and three columns.
We can observe the output of this code is a 4×3 matrix where all the elements are 1.



```python
import tensorflow as tf

sess = tf.Session()
B = tf.ones([4, 3])
print(sess.run(B))
```



```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

Tensor Creation Techniques

Built-in methods for tensor creation

3. tf.fill

With $\boldsymbol{tf.fill}$, we can create a tensor of a desired shape and set all the elements to a specified value.
Here, we create a tensor C with the shape of [2, 3], and set all the elements to 13.
We can observe the output of the code is a 2×3 matrix where all the elements are 13.

```python
import tensorflow as tf


sess = tf.Session()
C = tf.fill([2, 3], 13)
print(sess.run(C))
```

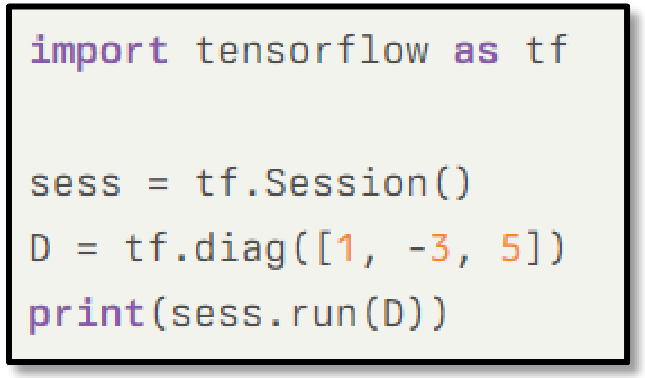
```
[[13 13 13]
 [13 13 13]]
```

Tensor Creation Techniques

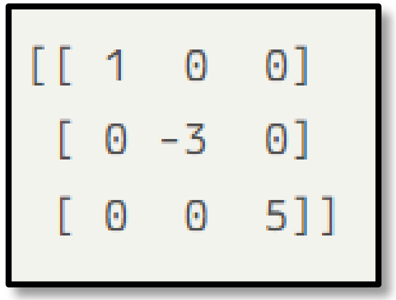
Built-in methods for tensor creation

4. tf.diag

With $\boldsymbol{tf.diag}$, we can create a diagonal matrix of a desired shape and set the elements on the diagonal to specified values.
Here, we create a matrix D, and set the elements of the diagonal to 1, -3 and 5.
We can observe the output of the code is a 3×3 matrix where the elements on the diagonal are the desired values.



```
import tensorflow as tf

sess = tf.Session()
D = tf.diag([1, -3, 5])
print(sess.run(D))
```



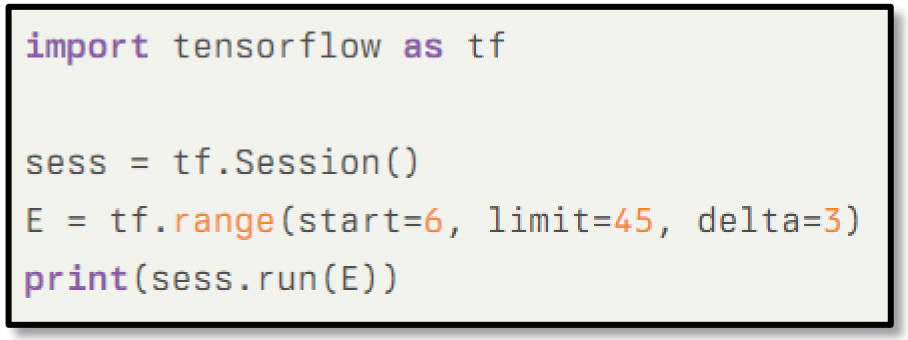
```
[[ 1  0  0]
 [ 0 -3  0]
 [ 0  0  5]]
```

Tensor Creation Techniques

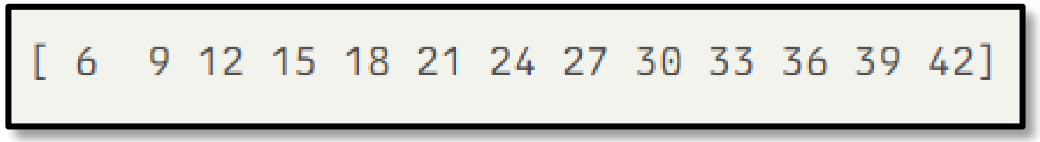
Built-in methods for tensor creation

5. tf.range

With $\boldsymbol{tf.range}$, we can create a tensor where the elements form a sequence of numbers, starting from a specified value, having a limit, and having a specified step.
Here, we create a tensor E, starting from 6, setting the limit to 45 (exclusive), and the step to 3.
We can observe the output of the code is a sequence of numbers from 6 to 45 (exclusive), with a step of 3.



```
import tensorflow as tf

sess = tf.Session()
E = tf.range(start=6, limit=45, delta=3)
print(sess.run(E))
```



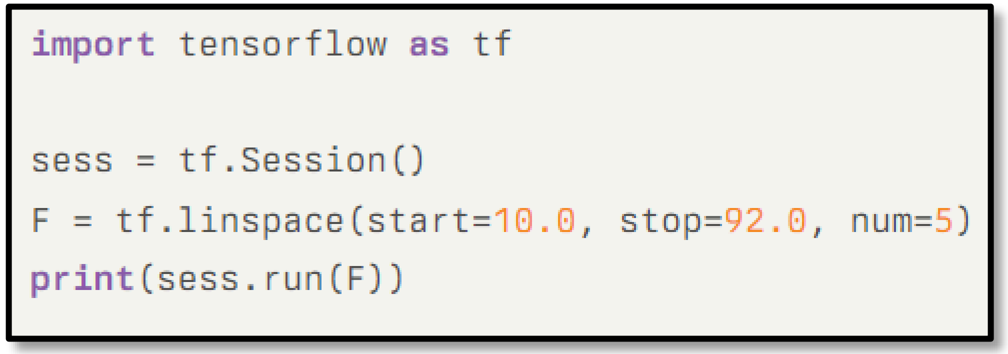
```
[ 6  9 12 15 18 21 24 27 30 33 36 39 42]
```

Tensor Creation Techniques

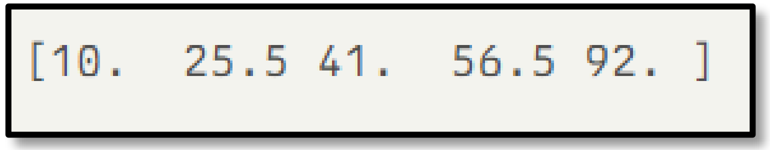
Built-in methods for tensor creation

6. tf.linspace

With $\boldsymbol{tf.linspace}$, we can create a tensor where the elements form a sequence of numbers, with evenly spaced values.
Here, we create a tensor F, starting from 10 and ending at 92 (inclusive), setting the number of elements to 5.
The output is a sequence consisting of 5 numbers from 10 to 92, and each pair of consecutive elements have the same difference.



```
import tensorflow as tf

sess = tf.Session()
F = tf.linspace(start=10.0, stop=92.0, num=5)
print(sess.run(F))
```



```
[10.   25.5 41.   56.5 92. ]
```

Built-in methods for tensor creation

7. tf.random_uniform

With $\boldsymbol{tf.random\_uniform}$, we can create a tensor with random values drawn from a uniform distribution within a specified range.
Here, we create a tensor G with the shape of [2, 3], setting the minimum value to 0, and the maximum value to 4.
The output is a 2×3 matrix where the elements are random values from a uniform distribution within the specified range.

```
import tensorflow as tf

sess = tf.Session()
G = tf.random.uniform(shape=[2, 3], minval=0, maxval=4)
print(sess.run(G))
```

```
[[2.3456743 1.2345674 0.7654321]
 [3.4567893 0.1234567 2.8765432]]
```

Built-in methods for tensor creation

8. tf.random_normal

With $\boldsymbol{tf.random\_normal}$, we can create a tensor with random values drawn from a normal distribution with a specified mean and standard deviation.
We create a tensor H with the shape of [2, 3], setting the mean to 5 and the standard deviation to 4.
The output is a 2×3 matrix where the elements are random values from a normal distribution with the specified mean and standard deviation.

```python
import tensorflow as tf

sess = tf.Session()
H = tf.random.normal(shape=[2, 3], mean=5.0, stddev=4.0)
print(sess.run(H))
```

```
[[ 6.234   2.456   3.789]
 [ 8.123   4.567   9.876]]
```

Built-in methods for tensor creation

8. tf.eye

With $\boldsymbol{tf.eye}$, we can create an identity matrix, which is a square matrix with ones on the diagonal and zeros elsewhere.
We create a tensor I with the shape of [2, 2].
The output is a 2 ×2 matrix where the elements on the diagonal are 1, while all other elements are zero.

```
import tensorflow as tf

sess = tf.Session()
I = tf.eye(2)
print(sess.run(I))
```

```
[[1. 0.]
 [0. 1.]]
```

Working on matrices

At this point, we are familiar with the various methods for creating tensors with desired characteristics.
Now, we will shift our focus to working with matrices in TensorFlow. Matrices are two-dimensional tensors, and they play a crucial role in many mathematical computations.
we will explore different operations that can be applied to matrices, namely addition, subtraction, multiplication and transposition.

We will provide code examples for each operation to illustrate their practical applications.

Tensor Creation Techniques

Working on matrices

Before we delve into the various matrix operations, it is essential to first create some matrices that we will use for our examples.
We will create the following matrices:

- Matrix $\boldsymbol{A}$: a 2×3 matrix representing sample data
- Matrix $\boldsymbol{B}$: a 2×3 matrix representing sample data
- Matrix $\boldsymbol{C}$: a 3×2 matrix to demonstrate matrix multiplication
- Matrix $\boldsymbol{D}$: a 2×2 identity matrix

```
import tensorflow as tf

sess = tf.Session()

# Creating Matrix A (2x3)
A = tf.constant([[1, 2, 3],
                 [4, 5, 6]])

# Creating Matrix B (2x3)
B = tf.constant([[2, 5, -3],
                 [4, -1, 3]])

# Creating Matrix C (3x2)
C = tf.constant([[-1, 6],
                 [4, 1],
                 [3, 2]])

# Creating Matrix D (2x2)
D = tf.eye(2)
```

```
print(sess.run(A))
```

```
[[1 2 3]
 [4 5 6]]
```

```
print(sess.run(B))
```

```
[[ 2  5 -3]
 [ 4 -1  3]]
```

```
print(sess.run(C))
```

```
[[-1  6]
 [ 4  1]
 [ 3  2]]
```

```
print(sess.run(D))
```

```
[[1. 0.]
 [0. 1.]]
```

Matrix Operations

1. Addition

In this example, we will add matrices A and B to demonstrate this operation.

```
print(sess.run(A+B))
```

```
[[ 3  7  0]
 [ 8  4  9]]
```

2. Subtraction

In this example, we will subtract matrices A and B to demonstrate this operation.

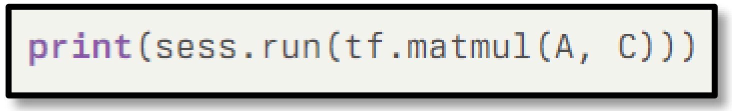
```
print(sess.run(A - B))
```

```
[[-1 -3  6]
 [ 0  6  3]]
```

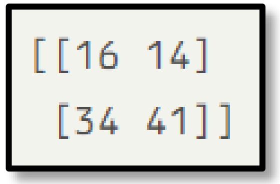
Matrix Operations

3. Multiplication

In this example, we will multiply matrices A and B to demonstrate this operation.

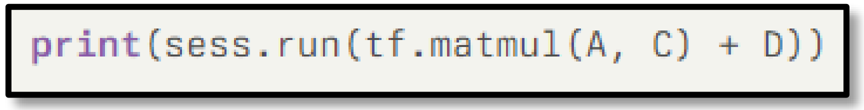


```
print(sess.run(tf.matmul(A, C)))
```

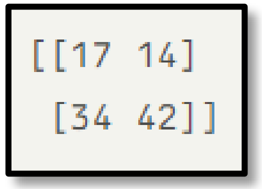


```
[[16 14]
 [34 41]]
```

Now, let us explore how to combine addition and multiplication operations on matrices.



```
print(sess.run(tf.matmul(A, C) + D))
```



```
[[17 14]
 [34 42]]
```

Matrix Operations

4. Transposition

In this example, we will calculate the transpose of matrix A to demonstrate this operation.

```
print(sess.run(tf.transpose(A)))
```

```
[[1 4]
 [2 5]
 [3 6]]
```

We will also calculate the transpose of matrix C to further clarify this operation.

```
print(sess.run(tf.transpose(C)))
```

```
[[-1  4  3]
 [ 6  1  2]]
```