Practical 1

Aim:Performing matrix multiplication and finding eigen vectors and eigen values using TensorFlow

```
# create tensor
'''The example below defines a 3x3x3 tensor as a NumPy ndarray.
Here, we first define rows, then a list of rows stacked as columns,
then a list of columns stacked as levels in a cubefrom numpy import array'''
from numpy import array
T = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
print("Tensor Shape : ",T.shape)
print("~~~~~~ Tensor ~~~~~~")
print(T)
```

```
Tensor Shape :  (3, 3, 3)
~~~~~~ Tensor ~~~~~~
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]]

 [[11 12 13]
  [14 15 16]
  [17 18 19]]

 [[21 22 23]
  [24 25 26]
  [27 28 29]]]
```

```
# tensor addition
'''
The element-wise addition of two tensors with the same dimensions results
in a new tensor with the same dimensions where each scalar value
is the element-wise addition of the scalars in the parent tensors.
'''
from numpy import array
A = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
B = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
C = A + B
print("~~~~~~ Tensor Addition ~~~~~~")
print(C)
```

rocktheit.com

```
     ~~~~~~ Tensor Addition ~~~~~~
     [[[ 2  4  6]
       [ 8 10 12]
       [14 16 18]]

      [[22 24 26]
       [28 30 32]
       [34 36 38]]

      [[42 44 46]
       [48 50 52]
       [54 56 58]]]
```

```
# tensor subtraction
'''
The element-wise subtraction of one tensor from another tensor with the same
dimensions results in a new tensor with the same dimensions where each
scalar value is the element-wise subtraction of the scalars in the parent tensors
'''
from numpy import array
A = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
B = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
C = A - B

print("~~~~~~ Tensor Subtraction ~~~~~~")
print(C)
```

```
     ~~~~~~ Tensor Subtraction ~~~~~~
     [[[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]]]
```

```
# tensor Hadamard product
'''
The element-wise multiplication of one tensor from another tensor with the same c
results in a new tensor with the same dimensions where each scalar
value is the element-wise multiplication of the scalars in the parent tensors.

As with matrices, the operation is referred to as the Hadamard Product
```

rocktneit.com

```
to differentiate it from tensor multiplication.
'''
from numpy import array
A = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
B = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
C = A * B
print("~~~~~~ Tensor Hadamard Product ~~~~~~")
print(C)
```

```
     ~~~~~~ Tensor Hadamard Product ~~~~~~
     [[[  1   4   9]
       [ 16  25  36]
       [ 49  64  81]]

      [[121 144 169]
       [196 225 256]
       [289 324 361]]

      [[441 484 529]
       [576 625 676]
       [729 784 841]]]
```

```
# tensor division
'''
The element-wise division of one tensor from another tensor with the same
dimensions results in a new tensor with the same dimensions where each
scalar value is the element-wise division of the scalars in the parent tensors.
'''
from numpy import array
A = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
B = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
C = A / B
print("~~~~~~ Tensor Division ~~~~~~")
print(C)
```

```
     ~~~~~~ Tensor Division ~~~~~~
     [[[1. 1. 1.]
       [1. 1. 1.]
       [1. 1. 1.]]
```

rocktheit.com

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

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

```
# tensor product
'''
Given a tensor A with q dimensions and tensor B with r dimensions,
the product of these tensors will be a new tensor with the
order of q + r or, said another way, q + r dimensions.

'''
from numpy import array
from numpy import tensordot
#The function takes as arguments the two tensors to be multiplied and
#the axis on which to sum the products over, called the sum reduction.
#To calculate the tensor product,
#also called the tensor dot product in NumPy, the axis must be set to 0.
A = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
B = array([
  [[1,2,3],    [4,5,6],    [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]],
  ])
C = tensordot(A, B, axes=0)
print(C)
```

```
         [[294 308 322]
          [336 350 364]
          [378 392 406]]]


        [[[ 15  30  45]
          [ 60  75  90]
          [105 120 135]]

         [[165 180 195]
          [210 225 240]
          [255 270 285]]

         [[315 330 345]
          [360 375 390]
          [405 420 435]]]


        [[[ 16  32  48]
          [ 64  80  96]
          [112 128 144]]
```

rockheit.com

```
         [[176 192 208]
          [224 240 256]
          [272 288 304]]

         [[336 352 368]
          [384 400 416]
          [432 448 464]]]]



       [[[[ 17  34  51]
          [ 68  85 102]
          [119 136 153]]

         [[187 204 221]
          [238 255 272]
          [289 306 323]]

         [[357 374 391]
          [408 425 442]
          [459 476 493]]]


        [[[ 18  36  54]
          [ 72  90 108]
          [126 144 162]]

         [[198 216 234]
          [252 270 288]
          [306 324 342]]

         [[378 396 414]
          [432 450 468]
          [486 504 522]]]
```

rocktheit.com

```
# Linear Equation Plot
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-10, 10)
y = 2*x + 1
y1 = 6*x - 2

plt.figure()
plt.plot(x, y)
plt.plot(x, y1)
plt.xlim(-2, 10)
plt.ylim(-2, 10)
# draw axes
plt.axvline(x=0, color='#A9A9A9')
plt.axhline(y=0, color='#A9A9A9')
plt.show()
plt.close()
```

```
# A python program to illustrate orthogonal vector

# Import numpy module
import numpy

# Taking two vectors
v1 = [[1, -2, 4]]
v2 = [[2, 5, 2]]

# Transpose of v1
transposeOfV1 = numpy.transpose(v1)

# Matrix multiplication of both vectors
result = numpy.dot(v2, transposeOfV1)
print("Result = ", result)
```

```
     Result =  [[0]]
```

rocktheit.com