Yog Chaudhary

11727095

ADTA 5550: Deep Learning with Big Data

Professor: Dr. Hamidreza Moradi

University of North Texas

Jan 29, 2024

1. Declare two constant tensors that have the values of 15 and 45. Add these two tensors and print out the results.

In [1]:

```
import tensorflow as tf
```

Declare two constant tensors

In [2]:

```
tensor1 = tf.constant(15)
```

In [3]:

```
tensor2 = tf.constant(45)
```

Add the two tensors

In [4]:

```
result_tensor = tf.add(tensor1, tensor2)
```

In [5]:

```
with tf.compat.v1.Session() as sess:
    result = sess.run(result_tensor)
```

```
User settings:

   KMP_AFFINITY=granularity=fine,verbose,compact,1,0
   KMP_BLOCKTIME=0
   KMP_SETTINGS=1
   OMP_NUM_THREADS=8

Effective settings:

   KMP_ABORT_DELAY=0
   KMP_ADAPTIVE_LOCK_PROPS='1,1024'
   KMP_ALIGN_ALLOC=64
   KMP_ALL_THREADPRIVATE=128
   KMP_ATOMIC_MODE=2
   KMP_BLOCKTIME=0
   KMP_CPUINFO_FILE: value is not defined
   KMP_DETERMINISTIC_REDUCTION=false
   KMP_DEVICE_THREAD_LIMIT=2147483647
   KMP_DISP_HAND_THREAD=false
   KMP_DISP_NUM_BUFFERS=7
   KMP_DUPLICATE_LIB_OK=false
   KMP_FORCE_REDUCTION: value is not defined
   KMP_FOREIGN_THREADS_THREADPRIVATE=true
   KMP_FORKJOIN_BARRIER='2,2'
   KMP_FORKJOIN_BARRIER_PATTERN='hyper,hyper'
   KMP_FORKJOIN_FRAMES=true
   KMP_FORKJOIN_FRAMES_MODE=3
   KMP_GTID_MODE=3
   KMP_HANDLE_SIGNALS=false
   KMP_HOT_TEAMS_MAX_LEVEL=1
   KMP_HOT_TEAMS_MODE=0
   KMP_INIT_AT_FORK=true
   KMP_ITT_PREPARE_DELAY=0
   KMP_LIBRARY=throughput
   KMP_LOCK_KIND=queuing
   KMP_MALLOC_POOL_INCR=1M
   KMP_MWAIT_HINTS=0
   KMP_NUM_LOCKS_IN_BLOCK=1
   KMP_PLAIN_BARRIER='2,2'
   KMP_PLAIN_BARRIER_PATTERN='hyper,hyper'
   KMP_REDUCTION_BARRIER='1,1'
   KMP_REDUCTION_BARRIER_PATTERN='hyper,hyper'
   KMP_SCHEDULE='static,balanced;guided,iterative'
   KMP_SETTINGS=true
   KMP_SPIN_BACKOFF_PARAMS='4096,100'
   KMP_STACKOFFSET=64
   KMP_STACKPAD=0
   KMP_STACKSIZE=8M
   KMP_STORAGE_MAP=false
   KMP_TASKING=2
   KMP_TASKLOOP_MIN_TASKS=0
   KMP_TASK_STEALING_CONSTRAINT=1
   KMP_TEAMS_THREAD_LIMIT=8
   KMP_TOPOLOGY_METHOD=all
   KMP_USER_LEVEL_MWAIT=false
   KMP_USE_YIELD=1
   KMP_VERSION=false
   KMP_WARNINGS=true
   OMP_AFFINITY_FORMAT='OMP: pid %P tid %i thread %n bound to OS proc set {%A}'
   OMP_ALLOCATOR=omp_default_mem_alloc
```

```
    OMP_CANCELLATION=false
    OMP_DEBUG=disabled
    OMP_DEFAULT_DEVICE=0
    OMP_DISPLAY_AFFINITY=false
    OMP_DISPLAY_ENV=false
    OMP_DYNAMIC=false
    OMP_MAX_ACTIVE_LEVELS=2147483647
    OMP_MAX_TASK_PRIORITY=0
    OMP_NESTED=false
    OMP_NUM_THREADS='8'
    OMP_PLACES: value is not defined
    OMP_PROC_BIND='intel'
    OMP_SCHEDULE='static'
    OMP_STACKSIZE=8M
    OMP_TARGET_OFFLOAD=DEFAULT
    OMP_THREAD_LIMIT=2147483647
    OMP_TOOL=enabled
    OMP_TOOL_LIBRARIES: value is not defined
    OMP_WAIT_POLICY=PASSIVE
    KMP_AFFINITY='verbose,warnings,respect,granularity=fine,compact,1,0'

2024-01-28 03:03:42.292449: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94]
CPU Frequency: 2200210000 Hz
2024-01-28 03:03:42.293907: I tensorflow/compiler/xla/service/service.cc:168] XLA ser
vice 0x558080004730 initialized for platform Host (this does not guarantee that XLA w
ill be used). Devices:
2024-01-28 03:03:42.293944: I tensorflow/compiler/xla/service/service.cc:176]   Strea
mExecutor device (0): Host, Default Version
2024-01-28 03:03:42.294560: I tensorflow/core/common_runtime/process_util.cc:136] Cre
ating new thread pool with default inter op setting: 2. Tune using inter_op_paralleli
sm_threads for best performance.
```

In [7]:

```
print("The result of adding the two tensors is:", result)
```

```
The result of adding the two tensors is: 60
```

2. Declare two variable tensors, a and b, that are initialized with scalar values of 2.75 and 8.5. Find their product and print out the result.

In [8]:

```
a = tf.constant(2.75)
```

In [9]:

```
b = tf.constant(8.5)
```

In [10]:

```
product = tf.multiply(a, b)
```

In [11]:

```
with tf.compat.v1.Session() as sess:
    result = sess.run(product)
```

In [12]:

```
print(product)
```

```
Tensor("Mul:0", shape=(), dtype=float32)
```

In [13]:
```
a = tf.constant(2.75)
```

In [14]:
```
b = tf.constant(8.5)
```

In [15]:
```
product = tf.multiply(a, b)
```

In [16]:
```
with tf.compat.v1.Session() as sess:
    result = sess.run(product)
```

In [17]:
```
print("The product of a and b is:", result)
```

```
The product of a and b is: 23.375
```

3. Create two placeholders: x and y - that are both scalars of 32-bit floats. Assign 5.25 to x and 12.6 to y, multiply them together, and print out the results.

In [18]:
```
import tensorflow as tf
```

In [19]:
```
tf.compat.v1.disable_eager_execution()
```

In [20]:
```
a = tf.constant(5.25, dtype = tf.float32)
b = tf.constant(12.6, dtype = tf.float32)
c = a * b
hello = tf.constant('Hello, TensorFlow!')

sess = tf.compat.v1.Session()

print(sess.run(c))
```

```
66.15
```

4. Create one placeholder: z - that is an N-Dimensional array (N can be >= 1) that can have any shape (shape = None). Feed this vector [1, 3, 5, 7, 9] into z and multiply it by 3. Display the results.

In [21]:
```
import numpy as np
```

In [22]:
```
# create a placeholder z with shape None
z = np.ndarray(shape=None)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: DeprecationWarning: P
assing None into shape arguments as an alias for () is deprecated.
```

In [23]:

```
# feed the vector [1, 3, 5, 7, 9] into z
z = np.array([1, 3, 5, 7, 9])
```

In [24]:

```
# multiply z by 3
z *= 3
```

In [25]:

```
# display the result
print(z)
```

```
[ 3  9 15 21 27]
```

5. Create a constant tensor that is a matrix of the shape (8, 8). The matrix is initialized with all ones (1). Create a variable tensor that is also a matrix of the shape (8, 8) and initialized with random integer values between 0 and 99. Add these two tensors and display the results.

In [50]:

```
import numpy as np
```

In [51]:

```
x = np.ones([8, 8])
x = tf.constant(x)
print(x)
y = np.zeros([8,8])
```

```
Tensor("Const_12:0", shape=(8, 8), dtype=float64)
```

In [59]:

```
import tensorflow as tf
import numpy as np

# Create a constant tensor 'x' filled with ones
x = tf.constant(np.ones([8, 8]))

# Create a numpy array 'y' filled with zeros
y = np.zeros([8, 8])

# Fill 'y' with random integers between 1 and 97
for i in range(8):
    for j in range(8):
        y[i][j] = np.random.randint(1, 98)

# Convert 'y' to a TensorFlow variable
y = tf.Variable(y)

# Add 'x' and 'y' tensors
addition = x + y

# Print the result
print('Addition = ', addition)
```

```
Addition =  Tensor("add_5:0", shape=(8, 8), dtype=float64)
```

In [ ]:

```
# Assignment 2 Deep Learning ADTA 5550
```