Example 2-9. Adding tensors together

```
>>> c = tf.ones((2, 2))
>>> d = tf.ones((2, 2))
>>> e = c + d
>>> e.eval()
array([[ 2.,  2.],
       [ 2.,  2.]], dtype=float32)
>>> f = 2 * e
>>> f.eval()
array([[ 4.,  4.],
       [ 4.,  4.]], dtype=float32)
```

Tensors can also be multiplied this way. Note, however, when multiplying two tensors we get elementwise multiplication and not matrix multiplication, which can be seen in Example 2-10.

Example 2-10. Elementwise tensor multiplication

```
>>> c = tf.fill((2,2), 2.)
>>> d = tf.fill((2,2), 7.)
>>> e = c * d
>>> e.eval()
array([[ 14.,  14.],
       [ 14.,  14.]], dtype=float32)
```

Matrix Operations

TensorFlow provides a variety of amenities for working with matrices. (Matrices by far are the most common type of tensor used in practice.) In particular, TensorFlow provides shortcuts to make certain types of commonly used matrices. The most widely used of these is likely the identity matrix. Identity matrices are square matrices that are 0 everywhere except on the diagonal, where they are 1. `tf.eye()` allows for fast construction of identity matrices of desired size (Example 2-11).

Example 2-11. Creating an identity matrix

```
>>> a = tf.eye(4)
>>> a.eval()
array([[ 1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.]], dtype=float32)
```

Diagonal matrices are another common type of matrix. Like identity matrices, diagonal matrices are only nonzero along the diagonal. Unlike identity matrices, they may

take arbitrary values along the diagonal. Let's construct a diagonal matrix with ascending values along the diagonal (Example 2-12). To start, we'll need a method to construct a vector of ascending values in TensorFlow. The easiest way for doing this is invoking `tf.range(start, limit, delta)`. Note that `limit` is excluded from the range and `delta` is the step size for the traversal. The resulting vector can then be fed to `tf.diag(diagonal)`, which will construct a matrix with the specified diagonal.

Example 2-12. Creating diagonal matrices

```
>>> r = tf.range(1, 5, 1)
>>> r.eval()
array([1, 2, 3, 4], dtype=int32)
>>> d = tf.diag(r)
>>> d.eval()
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]], dtype=int32)
```

Now suppose that we have a specified matrix in TensorFlow. How do we compute the matrix transpose? `tf.matrix_transpose()` will do the trick nicely (Example 2-13).

Example 2-13. Taking a matrix transpose

```
>>> a = tf.ones((2, 3))
>>> a.eval()
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]], dtype=float32)
>>> at = tf.matrix_transpose(a)
>>> at.eval()
array([[ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]], dtype=float32)
```

Now, let's suppose we have a pair of matrices we'd like to multiply using matrix multiplication. The easiest way to do so is by invoking `tf.matmul()` (Example 2-14).

Example 2-14. Performing matrix multiplication

```
>>> a = tf.ones((2, 3))
>>> a.eval()
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]], dtype=float32)
>>> b = tf.ones((3, 4))
>>> b.eval()
array([[ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.]], dtype=float32)
```

```
>>> c = tf.matmul(a, b)
>>> c.eval()
array([[ 3.,  3.,  3.,  3.],
       [ 3.,  3.,  3.,  3.]], dtype=float32)
```

You can check that this answer matches the mathematical definition of matrix multiplication we provided earlier.

Tensor Types

You may have noticed the `dtype` notation in the preceding examples. Tensors in TensorFlow come in a variety of types such as `tf.float32`, `tf.float64`, `tf.int32`, `tf.int64`. It's possible to to create tensors of specified types by setting `dtype` in tensor construction functions. Furthermore, given a tensor, it's possible to change its type using casting functions such as `tf.to_double()`, `tf.to_float()`, `tf.to_int32()`, `tf.to_int64()`, and others (Example 2-15).

Example 2-15. Creating tensors of different types

```
>>> a = tf.ones((2,2), dtype=tf.int32)
>>> a.eval()
array([[0, 0],
       [0, 0]], dtype=int32)
>>> b = tf.to_float(a)
>>> b.eval()
array([[ 0.,  0.],
       [ 0.,  0.]], dtype=float32)
```

Tensor Shape Manipulations

Within TensorFlow, tensors are just collections of numbers written in memory. The different shapes are views into the underlying set of numbers that provide different ways of interacting with the set of numbers. At different times, it can be useful to view the same set of numbers as forming tensors with different shapes. `tf.reshape()` allows tensors to be converted into tensors with different shapes (Example 2-16).

Example 2-16. Manipulating tensor shapes

```
>>> a = tf.ones(8)
>>> a.eval()
array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.], dtype=float32)
>>> b = tf.reshape(a, (4, 2))
>>> b.eval()
array([[ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]], dtype=float32)
```