tf.matmul

```
matmul(
    a,
    b,
    transpose_a=False,
    transpose_b=False,
    adjoint_a=False,
    adjoint_b=False,
    a_is_sparse=False,
    b_is_sparse=False,
    name=None
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Math > Matrix Math Functions

Multiplies matrix `a` by matrix `b`, producing `a` * `b`.

The inputs must, following any transpositions, be tensors of rank >= 2 where the inner 2 dimensions specify valid matrix multiplication arguments, and any further outer dimensions match.

Both matrices must be of the same type. The supported types are: `float16`, `float32`, `float64`, `int32`, `complex64`, `complex128`.

Either matrix can be transposed or adjointed (conjugated and transposed) on the fly by setting one of the corresponding flag to `True`. These are `False` by default.

If one or both of the matrices contain a lot of zeros, a more efficient multiplication algorithm can be used by setting the corresponding `a_is_sparse` or `b_is_sparse` flag to `True`. These are `False` by default. This optimization is only available for plain matrices (rank-2 tensors) with datatypes `bfloat16` or `float32`.

For example:

```
# 2-D tensor `a`
# [[1, 2, 3],
#  [4, 5, 6]]
a = tf.constant([1, 2, 3, 4, 5, 6], shape=[2, 3])

# 2-D tensor `b`
# [[ 7,  8],
#  [ 9, 10],
#  [11, 12]]
b = tf.constant([7, 8, 9, 10, 11, 12], shape=[3, 2])

# `a` * `b`
# [[ 58,  64],
#  [139, 154]]
c = tf.matmul(a, b)

# 3-D tensor `a`
# [[[ 1,  2,  3],
#   [ 4,  5,  6]],
#  [[ 7,  8,  9],
#   [10, 11, 12]]]
a = tf.constant(np.arange(1, 13, dtype=np.int32),
                shape=[2, 2, 3])

# 3-D tensor `b`
# [[[13, 14],
#   [15, 16],
#   [17, 18]],
#  [[19, 20],
#   [21, 22],
#   [23, 24]]]
b = tf.constant(np.arange(13, 25, dtype=np.int32),
                shape=[2, 3, 2])

# `a` * `b`
# [[[ 94, 100],
#   [229, 244]],
#  [[508, 532],
#   [697, 730]]]
c = tf.matmul(a, b)

# Since python >= 3.5 the @ operator is supported (see PEP 465).
# In TensorFlow, it simply calls the `tf.matmul()` function, so the
# following lines are equivalent:
d = a @ b @ [[10.], [11.]]
d = tf.matmul(tf.matmul(a, b), [[10.], [11.]])
```

Args:

- `a`: **`Tensor`** of type **`float16`**, **`float32`**, **`float64`**, **`int32`**, **`complex64`**, **`complex128`** and rank > 1.
- `b`: **`Tensor`** with same type and rank as **`a`**.
- `transpose_a`: If **`True`**, **`a`** is transposed before multiplication.
- `transpose_b`: If **`True`**, **`b`** is transposed before multiplication.
- `adjoint_a`: If **`True`**, **`a`** is conjugated and transposed before multiplication.
- `adjoint_b`: If **`True`**, **`b`** is conjugated and transposed before multiplication.
- `a_is_sparse`: If **`True`**, **`a`** is treated as a sparse matrix.
- `b_is_sparse`: If **`True`**, **`b`** is treated as a sparse matrix.
- `name`: Name for the operation (optional).

Returns:

A `Tensor` of the same type as `a` and `b` where each inner-most matrix is the product of the corresponding matrices in `a` and `b`, e.g. if all transpose or adjoint attributes are `False`:

`output`[..., i, j] = sum_k (`a`[..., i, k] * `b`[..., k, j]), for all indices i, j.

- `Note`: This is matrix product, not element-wise product.

Raises:

- `ValueError`: If transpose_a and adjoint_a, or transpose_b and adjoint_b are both set to True.

Last updated November 2, 2017.