Module: tf.xla.experimental

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental#top_of_page)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental#functions)

Public API for tf.xla.experimental namespace.

Functions

[compile(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental/compile): Builds an operator that compiles and runs computation with XLA.

[jit\_scope(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental/jit_scope): Enable or disable JIT compilation of operators within the scope.

# tf.xla.experimental.compile

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental/compile#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental/compile#aliases)

Builds an operator that compiles and runs computation with XLA.

### Aliases:

* tf.compat.v1.xla.experimental.compile
* tf.compat.v2.xla.experimental.compile
* tf.xla.experimental.compile

tf.xla.experimental.compile(  
    computation,  
    inputs=None  
)

Defined in [python/compiler/xla/xla.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/compiler/xla/xla.py).

NOTE: In eager mode, computation will have @tf.function semantics.

#### Args:

* **computation**: A Python function that builds a computation to apply to the input. If the function takes n inputs, 'inputs' should be a list of n tensors.

computation may return a list of operations and tensors. Tensors must come before operations in the returned list. The return value of compile is a list of tensors corresponding to the tensors from the output of computation.

All Operations returned from computation will be executed when evaluating any of the returned output tensors.

* **inputs**: A list of inputs or None (equivalent to an empty list). Each input can be a nested structure containing values that are convertible to tensors. Note that passing an N-dimension list of compatible values will result in a N-dimension list of scalar tensors rather than a single Rank-N tensors. If you need different behavior, convert part of inputs to tensors with [tf.convert\_to\_tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/convert_to_tensor).

#### Returns:

Same data structure as if computation(\*inputs) is called directly with some exceptions for correctness. Exceptions include: 1) None output: a NoOp would be returned which control-depends on computation. 2) Single value output: A tuple containing the value would be returned. 3) Operation-only outputs: a NoOp would be returned which control-depends on computation. TODO(b/121383831): Investigate into removing these special cases.

#### Raises:

* **RuntimeError**: if called when eager execution is enabled.

# tf.xla.experimental.jit\_scope

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental/jit_scope#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental/jit_scope#aliases)

Enable or disable JIT compilation of operators within the scope.

### Aliases:

* tf.compat.v1.xla.experimental.jit\_scope
* tf.compat.v2.xla.experimental.jit\_scope
* tf.xla.experimental.jit\_scope

tf.xla.experimental.jit\_scope(  
    \*args,  
    \*\*kwds  
)

NOTE: This is an experimental feature.

The compilation is a hint and only supported on a best-effort basis.

#### Example usage:

with tf.xla.experimental.jit\_scope(): c = tf.matmul(a, b) # compiled with tf.xla.experimental.jit\_scope(compile\_ops=False): d = tf.matmul(a, c) # not compiled with tf.xla.experimental.jit\_scope( compile\_ops=lambda node\_def: 'matmul' in node\_def.op.lower()): e = tf.matmul(a, b) + d # matmul is compiled, the addition is not.

Example of separate\_compiled\_gradients: # In the example below, the computations for f, g and h will all be compiled # in separate scopes. with tf.xla.experimental.jit\_scope( separate\_compiled\_gradients=True): f = tf.matmul(a, b) g = tf.gradients([f], [a, b], name='mygrads1') h = tf.gradients([f], [a, b], name='mygrads2')

#### Args:

* **compile\_ops**: Whether to enable or disable compilation in the scope. Either a Python bool, or a callable that accepts the parameter node\_def and returns a python bool.
* **separate\_compiled\_gradients**: If true put each gradient subgraph into a separate compilation scope. This gives fine-grained control over which portions of the graph will be compiled as a single unit. Compiling gradients separately may yield better performance for some graphs. The scope is named based on the scope of the forward computation as well as the name of the gradients. As a result, the gradients will be compiled in a scope that is separate from both the forward computation, and from other gradients.

#### Raises:

* **RuntimeError**: if called when eager execution is enabled.

#### Yields:

The current scope, enabling or disabling compilation.

# tf.compat.v1.user\_ops.my\_fact

Example of overriding the generated code for an Op.

tf.compat.v1.user\_ops.my\_fact()

Defined in [python/user\_ops/user\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/user_ops/user_ops.py).