Module: tf.compat.v1.lite & tf.lite

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite#top_of_page)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite#modules)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite#functions)

Public API for tf.lite namespace.

Modules

[constants](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/constants) module: Public API for tf.lite.constants namespace.

[experimental](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental) module: Public API for tf.lite.experimental namespace.

Classes

[class Interpreter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Interpreter): Interpreter interface for TensorFlow Lite Models.

[class OpHint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint): A class that helps build tflite function invocations.

[class OpsSet](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/OpsSet): Enum class defining the sets of ops available to generate TFLite models.

[class Optimize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Optimize): Enum defining the optimizations to apply when generating tflite graphs.

[class RepresentativeDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/RepresentativeDataset): Representative dataset to evaluate optimizations.

[class TFLiteConverter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TFLiteConverter): Convert a TensorFlow model into output\_format.

[class TargetSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/TargetSpec): Specification of target device.

[class TocoConverter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TocoConverter): Convert a TensorFlow model into output\_format using TOCO.

Functions

[toco\_convert(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/toco_convert): Convert a model using TOCO. (deprecated)

# tf.compat.v1.lite.OpHint

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint#top_of_page)
* [Class OpHint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint#class_ophint)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint#__init__)
* [Child Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint#child_classes)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint#methods)

## Class OpHint

A class that helps build tflite function invocations.

Defined in [lite/python/op\_hint.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/lite/python/op_hint.py).

It allows you to take a bunch of TensorFlow ops and annotate the construction such that toco knows how to convert it to tflite. This embeds a pseudo function in a TensorFlow graph. This allows embedding high-level API usage information in a lower level TensorFlow implementation so that an alternative implementation can be substituted later.

Essentially, any "input" into this pseudo op is fed into an identity, and attributes are added to that input before being used by the constituent ops that make up the pseudo op. A similar process is done to any output that is to be exported from the current op.

## \_\_init\_\_

\_\_init\_\_(  
    function\_name,  
    level=1,  
    children\_inputs\_mappings=None,  
    \*\*kwargs  
)

Create a OpHint.

#### Args:

* **function\_name**: Name of the function (the custom op name in tflite)
* **level**: OpHint level.
* **children\_inputs\_mappings**: Children OpHint inputs/outputs mapping. children\_inputs\_mappings should like below: "parent\_first\_child\_input": [{"parent\_input\_index": num, "child\_input\_index": num}, ...] "parent\_last\_child\_output": [{"parent\_output\_index": num, "child\_output\_index": num}, ...] "internal\_children\_input\_output": [{"child\_input\_index": num, "child\_output\_index": num}, ...]
* **\*\*kwargs**: Keyword arguments of any constant attributes for the function.

## Child Classes

[class OpHintArgumentTracker](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint/OpHintArgumentTracker)

## Methods

### add\_input

add\_input(  
    \*args,  
    \*\*kwargs  
)

Add a wrapped input argument to the hint.

#### Args:

* **\*args**: The input tensor.
* **\*\*kwargs**: "name" label "tag" a tag to group multiple arguments that will be aggregated. I.e. a string like 'cool\_input'. Basically multiple inputs can be added to the same hint for parallel operations that will eventually be combined. An example would be static\_rnn which creates multiple copies of state or inputs. "aggregate" aggregation strategy that is valid only for tag non None. Acceptable values are OpHint.AGGREGATE\_FIRST, OpHint.AGGREGATE\_LAST, and OpHint.AGGREGATE\_STACK. "index\_override" The global index to use. This corresponds to the argument order in the final stub that will be generated.

#### Returns:

The wrapped input tensor.

### add\_inputs

add\_inputs(  
    \*args,  
    \*\*kwargs  
)

Add a sequence of inputs to the function invocation.

#### Args:

* **\*args**: List of inputs to be converted (should be Tf.Tensor).
* **\*\*kwargs**: This allows 'names' which should be a list of names.

#### Returns:

Wrapped inputs (identity standins that have additional metadata). These are also are also tf.Tensor's.

### add\_output

add\_output(  
    \*args,  
    \*\*kwargs  
)

Add a wrapped output argument to the hint.

#### Args:

* **\*args**: The output tensor.
* **\*\*kwargs**: "name" label "tag" a tag to group multiple arguments that will be aggregated. I.e. a string like 'cool\_input'. Basically multiple inputs can be added to the same hint for parallel operations that will eventually be combined. An example would be static\_rnn which creates multiple copies of state or inputs. "aggregate" aggregation strategy that is valid only for tag non None. Acceptable values are OpHint.AGGREGATE\_FIRST, OpHint.AGGREGATE\_LAST, and OpHint.AGGREGATE\_STACK. "index\_override" The global index to use. This corresponds to the argument order in the final stub that will be generated.

#### Returns:

The wrapped output tensor.

### add\_outputs

add\_outputs(  
    \*args,  
    \*\*kwargs  
)

Add a sequence of outputs to the function invocation.

#### Args:

* **\*args**: List of outputs to be converted (should be tf.Tensor).
* **\*\*kwargs**: See

#### Returns:

Wrapped outputs (identity standins that have additional metadata). These are also tf.Tensor's.

## Class Members

* AGGREGATE\_FIRST = 'first'
* AGGREGATE\_LAST = 'last'
* AGGREGATE\_STACK = 'stack'
* CHILDREN\_INPUTS\_MAPPINGS = '\_tflite\_children\_ophint\_inputs\_mapping'
* FUNCTION\_AGGREGATE\_ATTR = '\_tflite\_function\_aggregate'
* FUNCTION\_INPUT\_INDEX\_ATTR = '\_tflite\_function\_input\_index'
* FUNCTION\_LEVEL\_ATTR = '\_tflite\_ophint\_level'
* FUNCTION\_NAME\_ATTR = '\_tflite\_function\_name'
* FUNCTION\_OUTPUT\_INDEX\_ATTR = '\_tflite\_function\_output\_index'
* FUNCTION\_SORT\_INDEX\_ATTR = '\_tflite\_function\_sort\_index'
* FUNCTION\_UUID\_ATTR = '\_tflite\_function\_uuid'
* TFLITE\_INPUT\_INDICES = '\_tflite\_input\_indices'

# tf.compat.v1.lite.OpHint.OpHintArgumentTracker

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint/OpHintArgumentTracker#top_of_page)
* [Class OpHintArgumentTracker](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint/OpHintArgumentTracker#class_ophintargumenttracker)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint/OpHintArgumentTracker#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint/OpHintArgumentTracker#methods)
  + [add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/OpHint/OpHintArgumentTracker#add)

## Class OpHintArgumentTracker

Conceptually tracks indices of arguments of "OpHint functions".

Defined in [lite/python/op\_hint.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/lite/python/op_hint.py).

The inputs and arguments of these functions both use an instance of the class so they can have independent numbering.

## \_\_init\_\_

\_\_init\_\_(  
    function\_name,  
    unique\_function\_id,  
    node\_name\_prefix,  
    attr\_name,  
    level=1,  
    children\_inputs\_mappings=None  
)

Initialize ophint argument.

#### Args:

* **function\_name**: Name of the function that this tracks arguments for.
* **unique\_function\_id**: UUID of function that this tracks arguments for.
* **node\_name\_prefix**: How identities that are created are named.
* **attr\_name**: Name of attribute to use to store the index for this hint. i.e. FUNCTION\_INPUT\_INDEX or FUNCTION\_OUTPUT\_INDEX
* **level**: Hierarchical level of the Ophint node, a number.
* **children\_inputs\_mappings**: Inputs/Outputs mapping for children hints.

## Methods

### add

add(  
    arg,  
    tag=None,  
    name=None,  
    aggregate=None,  
    index\_override=None  
)

Return a wrapped tensor of an input tensor as an argument.

#### Args:

* **arg**: A TensorFlow tensor that should be considered an argument.
* **tag**: String tag to identify arguments that should be packed.
* **name**: Name of argument. This is included in the Identity hint op names.
* **aggregate**: Strategy to aggregate. Acceptable values are OpHint.AGGREGATE\_FIRST, OpHint.AGGREGATE\_LAST, and OpHint.AGGREGATE\_STACK. Note, aggregate is only valid if tag is specified.
* **index\_override**: Specify what input/output index should this be in the final stub. i.e. add(arg0, index=1); add(arg1, index=0) will make the final stub be as stub\_func(inputs[arg1, arg0], outputs=[]) rather than the default call order based ordering.

#### Returns:

A tensor representing the wrapped argument.

#### Raises:

* **ValueError**: When indices are not consistent.

# tf.compat.v1.lite.TFLiteConverter

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TFLiteConverter#top_of_page)
* [Class TFLiteConverter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TFLiteConverter#class_tfliteconverter)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TFLiteConverter#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TFLiteConverter#methods)
  + [convert](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TFLiteConverter#convert)

## Class TFLiteConverter

Convert a TensorFlow model into output\_format.

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

This is used to convert from a TensorFlow GraphDef, SavedModel or tf.keras model into either a TFLite FlatBuffer or graph visualization.

#### Attributes:

* **inference\_type**: Target data type of real-number arrays in the output file. Must be {tf.float32, tf.uint8}. If optimzations are provided, this parameter is ignored. (default tf.float32)
* **inference\_input\_type**: Target data type of real-number input arrays. Allows for a different type for input arrays. If an integer type is provided and optimizations are not used,quantized\_inputs\_stats must be provided. If inference\_type is tf.uint8, signaling conversion to a fully quantized model from a quantization-aware trained input model, theninference\_input\_type defaults to tf.uint8. In all other cases, inference\_input\_typedefaults to tf.float32. Must be {tf.float32, tf.uint8, tf.int8}
* **inference\_output\_type**: Target data type of real-number output arrays. Allows for a different type for output arrays. If inference\_type is tf.uint8, signaling conversion to a fully quantized model from a quantization-aware trained output model, then inference\_output\_type defaults to tf.uint8. In all other cases, inference\_output\_type must be tf.float32, an error will be thrown otherwise. Must be {tf.float32, tf.uint8, tf.int8}
* **output\_format**: Output file format. Currently must be {TFLITE, GRAPHVIZ\_DOT}. (default TFLITE)
* **quantized\_input\_stats**: Dict of strings representing input tensor names mapped to tuple of floats representing the mean and standard deviation of the training data (e.g., {"foo" : (0., 1.)}). Only need if inference\_input\_type is QUANTIZED\_UINT8. real\_input\_value = (quantized\_input\_value - mean\_value) / std\_dev\_value. (default {})
* **default\_ranges\_stats**: Tuple of integers representing (min, max) range values for all arrays without a specified range. Intended for experimenting with quantization via "dummy quantization". (default None)
* **drop\_control\_dependency**: Boolean indicating whether to drop control dependencies silently. This is due to TFLite not supporting control dependencies. (default True)
* **reorder\_across\_fake\_quant**: Boolean indicating whether to reorder FakeQuant nodes in unexpected locations. Used when the location of the FakeQuant nodes is preventing graph transformations necessary to convert the graph. Results in a graph that differs from the quantized training graph, potentially causing differing arithmetic behavior. (default False)
* **change\_concat\_input\_ranges**: Boolean to change behavior of min/max ranges for inputs and outputs of the concat operator for quantized models. Changes the ranges of concat operator overlap when true. (default False)
* **allow\_custom\_ops**: Boolean indicating whether to allow custom operations. When false any unknown operation is an error. When true, custom ops are created for any op that is unknown. The developer will need to provide these to the TensorFlow Lite runtime with a custom resolver. (default False)
* **post\_training\_quantize**: Deprecated. Please specify [Optimize.DEFAULT] foroptimizations instead. Boolean indicating whether to quantize the weights of the converted float model. Model size will be reduced and there will be latency improvements (at the cost of accuracy). (default False)
* **dump\_graphviz\_dir**: Full filepath of folder to dump the graphs at various stages of processing GraphViz .dot files. Preferred over --output\_format=GRAPHVIZ\_DOT in order to keep the requirements of the output file. (default None)
* **dump\_graphviz\_video**: Boolean indicating whether to dump the graph after every graph transformation. (default False)
* **target\_ops**: Deprecated. Please specify target\_spec.supported\_ops instead. Set of OpsSet options indicating which converter to use. (default set([OpsSet.TFLITE\_BUILTINS]))
* **target\_spec**: Experimental flag, subject to change. Specification of target device.
* **optimizations**: Experimental flag, subject to change. A list of optimizations to apply when converting the model. E.g. [Optimize.DEFAULT]
* **representative\_dataset**: A representative dataset that can be used to generate input and output samples for the model. The converter can use the dataset to evaluate different optimizations.

#### Example usage:

# Converting a GraphDef from session.  
converter = lite.TFLiteConverter.from\_session(sess, in\_tensors, out\_tensors)  
tflite\_model = converter.convert()  
open("converted\_model.tflite", "wb").write(tflite\_model)  
  
# Converting a GraphDef from file.  
converter = lite.TFLiteConverter.from\_frozen\_graph(  
  graph\_def\_file, input\_arrays, output\_arrays)  
tflite\_model = converter.convert()  
open("converted\_model.tflite", "wb").write(tflite\_model)  
  
# Converting a SavedModel.  
converter = lite.TFLiteConverter.from\_saved\_model(saved\_model\_dir)  
tflite\_model = converter.convert()  
open("converted\_model.tflite", "wb").write(tflite\_model)  
  
# Converting a tf.keras model.  
converter = lite.TFLiteConverter.from\_keras\_model\_file(keras\_model)  
tflite\_model = converter.convert()  
open("converted\_model.tflite", "wb").write(tflite\_model)

## \_\_init\_\_

\_\_init\_\_(  
    graph\_def,  
    input\_tensors,  
    output\_tensors,  
    input\_arrays\_with\_shape=None,  
    output\_arrays=None  
)

Constructor for TFLiteConverter.

#### Args:

* **graph\_def**: Frozen TensorFlow GraphDef.
* **input\_tensors**: List of input tensors. Type and shape are computed using foo.shape and foo.dtype.
* **output\_tensors**: List of output tensors (only .name is used from this).
* **input\_arrays\_with\_shape**: Tuple of strings representing input tensor names and list of integers representing input shapes (e.g., [("foo" : [1, 16, 16, 3])]). Use only when graph cannot be loaded into TensorFlow and when input\_tensors and output\_tensors are None. (default None)
* **output\_arrays**: List of output tensors to freeze graph with. Use only when graph cannot be loaded into TensorFlow and when input\_tensors and output\_tensors are None. (default None)

#### Raises:

* **ValueError**: Invalid arguments.

## Methods

### convert

convert()

Converts a TensorFlow GraphDef based on instance variables.

#### Returns:

The converted data in serialized format. Either a TFLite Flatbuffer or a Graphviz graph depending on value in output\_format.

#### Raises:

* **ValueError**: Input shape is not specified. None value for dimension in input\_tensor.

### from\_frozen\_graph

@classmethod  
from\_frozen\_graph(  
    cls,  
    graph\_def\_file,  
    input\_arrays,  
    output\_arrays,  
    input\_shapes=None  
)

Creates a TFLiteConverter class from a file containing a frozen GraphDef.

#### Args:

* **graph\_def\_file**: Full filepath of file containing frozen GraphDef.
* **input\_arrays**: List of input tensors to freeze graph with.
* **output\_arrays**: List of output tensors to freeze graph with.
* **input\_shapes**: Dict of strings representing input tensor names to list of integers representing input shapes (e.g., {"foo" : [1, 16, 16, 3]}). Automatically determined when input shapes is None (e.g., {"foo" : None}). (default None)

#### Returns:

TFLiteConverter class.

#### Raises:

* **IOError**: File not found. Unable to parse input file.
* **ValueError**: The graph is not frozen. input\_arrays or output\_arrays contains an invalid tensor name. input\_shapes is not correctly defined when required

### from\_keras\_model\_file

@classmethod  
from\_keras\_model\_file(  
    cls,  
    model\_file,  
    input\_arrays=None,  
    input\_shapes=None,  
    output\_arrays=None,  
    custom\_objects=None  
)

Creates a TFLiteConverter class from a tf.keras model file.

#### Args:

* **model\_file**: Full filepath of HDF5 file containing the tf.keras model.
* **input\_arrays**: List of input tensors to freeze graph with. Uses input arrays from SignatureDef when none are provided. (default None)
* **input\_shapes**: Dict of strings representing input tensor names to list of integers representing input shapes (e.g., {"foo" : [1, 16, 16, 3]}). Automatically determined when input shapes is None (e.g., {"foo" : None}). (default None)
* **output\_arrays**: List of output tensors to freeze graph with. Uses output arrays from SignatureDef when none are provided. (default None)
* **custom\_objects**: Dict mapping names (strings) to custom classes or functions to be considered during model deserialization. (default None)

#### Returns:

TFLiteConverter class.

### from\_saved\_model

@classmethod  
from\_saved\_model(  
    cls,  
    saved\_model\_dir,  
    input\_arrays=None,  
    input\_shapes=None,  
    output\_arrays=None,  
    tag\_set=None,  
    signature\_key=None  
)

Creates a TFLiteConverter class from a SavedModel.

#### Args:

* **saved\_model\_dir**: SavedModel directory to convert.
* **input\_arrays**: List of input tensors to freeze graph with. Uses input arrays from SignatureDef when none are provided. (default None)
* **input\_shapes**: Dict of strings representing input tensor names to list of integers representing input shapes (e.g., {"foo" : [1, 16, 16, 3]}). Automatically determined when input shapes is None (e.g., {"foo" : None}). (default None)
* **output\_arrays**: List of output tensors to freeze graph with. Uses output arrays from SignatureDef when none are provided. (default None)
* **tag\_set**: Set of tags identifying the MetaGraphDef within the SavedModel to analyze. All tags in the tag set must be present. (default set("serve"))
* **signature\_key**: Key identifying SignatureDef containing inputs and outputs. (default DEFAULT\_SERVING\_SIGNATURE\_DEF\_KEY)

#### Returns:

TFLiteConverter class.

### from\_session

@classmethod  
from\_session(  
    cls,  
    sess,  
    input\_tensors,  
    output\_tensors  
)

Creates a TFLiteConverter class from a TensorFlow Session.

#### Args:

* **sess**: TensorFlow Session.
* **input\_tensors**: List of input tensors. Type and shape are computed using foo.shape and foo.dtype.
* **output\_tensors**: List of output tensors (only .name is used from this).

#### Returns:

TFLiteConverter class.

### get\_input\_arrays

get\_input\_arrays()

Returns a list of the names of the input tensors.

#### Returns:

List of strings.

# tf.compat.v1.lite.TocoConverter

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TocoConverter#top_of_page)
* [Class TocoConverter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TocoConverter#class_tococonverter)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TocoConverter#methods)
  + [from\_frozen\_graph](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TocoConverter#from_frozen_graph)
  + [from\_keras\_model\_file](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TocoConverter#from_keras_model_file)
  + [from\_saved\_model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TocoConverter#from_saved_model)
  + [from\_session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/TocoConverter#from_session)

## Class TocoConverter

Convert a TensorFlow model into output\_format using TOCO.

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

This class has been deprecated. Please use lite.TFLiteConverter instead.

## Methods

### from\_frozen\_graph

@classmethod  
from\_frozen\_graph(  
    cls,  
    graph\_def\_file,  
    input\_arrays,  
    output\_arrays,  
    input\_shapes=None  
)

Creates a TocoConverter class from a file containing a frozen graph. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **lite.TFLiteConverter.from\_frozen\_graph** instead.

### from\_keras\_model\_file

@classmethod  
from\_keras\_model\_file(  
    cls,  
    model\_file,  
    input\_arrays=None,  
    input\_shapes=None,  
    output\_arrays=None  
)

Creates a TocoConverter class from a tf.keras model file. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **lite.TFLiteConverter.from\_keras\_model\_file** instead.

### from\_saved\_model

@classmethod  
from\_saved\_model(  
    cls,  
    saved\_model\_dir,  
    input\_arrays=None,  
    input\_shapes=None,  
    output\_arrays=None,  
    tag\_set=None,  
    signature\_key=None  
)

Creates a TocoConverter class from a SavedModel. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **lite.TFLiteConverter.from\_saved\_model** instead.

### from\_session

@classmethod  
from\_session(  
    cls,  
    sess,  
    input\_tensors,  
    output\_tensors  
)

Creates a TocoConverter class from a TensorFlow Session. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **lite.TFLiteConverter.from\_session** instead.

# tf.compat.v1.lite.toco\_convert

Convert a model using TOCO. (deprecated)

tf.compat.v1.lite.toco\_convert(  
    input\_data,  
    input\_tensors,  
    output\_tensors,  
    \*args,  
    \*\*kwargs  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **lite.TFLiteConverter** instead.

Typically this function is used to convert from TensorFlow GraphDef to TFLite. Conversion can be customized by providing arguments that are forwarded to build\_toco\_convert\_protos (see documentation for details). This function has been deprecated. Please use lite.TFLiteConverterinstead.

#### Args:

* **input\_data**: Input data (i.e. often sess.graph\_def),
* **input\_tensors**: List of input tensors. Type and shape are computed using foo.shape and foo.dtype.
* **output\_tensors**: List of output tensors (only .name is used from this).
* **\*args**: See build\_toco\_convert\_protos,
* **\*\*kwargs**: See build\_toco\_convert\_protos.

#### Returns:

The converted data. For example if TFLite was the destination, then this will be a tflite flatbuffer in a bytes array.

#### Raises:

Defined in build\_toco\_convert\_protos.

Module: tf.compat.v1.lite.experimental

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental#top_of_page)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental#modules)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental#functions)

Public API for tf.lite.experimental namespace.

Modules

[nn](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn) module: Public API for tf.lite.experimental.nn namespace.

Functions

[convert\_op\_hints\_to\_stubs(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/convert_op_hints_to_stubs): Converts a graphdef with LiteOp hints into stub operations.

[get\_potentially\_supported\_ops(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/get_potentially_supported_ops): Returns operations potentially supported by TensorFlow Lite.

# tf.compat.v1.lite.experimental.convert\_op\_hints\_to\_stubs

Converts a graphdef with LiteOp hints into stub operations.

tf.compat.v1.lite.experimental.convert\_op\_hints\_to\_stubs(  
    session=None,  
    graph\_def=None,  
    write\_callback=(lambda graph\_def, comments: None)  
)

Defined in [lite/python/op\_hint.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/lite/python/op_hint.py).

This is used to prepare for toco conversion of complex intrinsic usages. Note: only one of session or graph\_def should be used, not both.

#### Args:

* **session**: A TensorFlow session that contains the graph to convert.
* **graph\_def**: A graph def that we should convert.
* **write\_callback**: A function pointer that can be used to write intermediate steps of graph transformation (optional).

#### Returns:

A new graphdef with all ops contained in OpHints being replaced by a single op call with the right parameters.

#### Raises:

* **ValueError**: If both session and graph\_def are provided.

# tf.compat.v1.lite.experimental.get\_potentially\_supported\_ops

Returns operations potentially supported by TensorFlow Lite.

tf.compat.v1.lite.experimental.get\_potentially\_supported\_ops()

Defined in [lite/experimental/tensorboard/ops\_util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/lite/experimental/tensorboard/ops_util.py).

The potentially support list contains a list of ops that are partially or fully supported, which is derived by simply scanning op names to check whether they can be handled without real conversion and specific parameters.

Given that some ops may be partially supported, the optimal way to determine if a model's operations are supported is by converting using the TensorFlow Lite converter.

#### Returns:

A list of SupportedOp.

Module: tf.compat.v1.lite.experimental.nn

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn#top_of_page)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn#functions)

Public API for tf.lite.experimental.nn namespace.

Classes

[class TFLiteLSTMCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TFLiteLSTMCell): Long short-term memory unit (LSTM) recurrent network cell.

[class TfLiteRNNCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TfLiteRNNCell): The most basic RNN cell.

Functions

[dynamic\_rnn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/dynamic_rnn): Creates a recurrent neural network specified by RNNCell cell.

# tf.compat.v1.lite.experimental.nn.dynamic\_rnn

Creates a recurrent neural network specified by RNNCell cell.

tf.compat.v1.lite.experimental.nn.dynamic\_rnn(  
    cell,  
    inputs,  
    sequence\_length=None,  
    initial\_state=None,  
    dtype=None,  
    parallel\_iterations=None,  
    swap\_memory=False,  
    time\_major=True,  
    scope=None  
)

Defined in [lite/experimental/examples/lstm/rnn.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/lite/experimental/examples/lstm/rnn.py).

Performs fully dynamic unrolling of inputs.

#### Example:

# create a BasicRNNCell  
rnn\_cell = tf.compat.v1.nn.rnn\_cell.BasicRNNCell(hidden\_size)  
  
# 'outputs' is a tensor of shape [batch\_size, max\_time, cell\_state\_size]  
  
# defining initial state  
initial\_state = rnn\_cell.zero\_state(batch\_size, dtype=tf.float32)  
  
# 'state' is a tensor of shape [batch\_size, cell\_state\_size]  
outputs, state = tf.compat.v1.nn.dynamic\_rnn(rnn\_cell, input\_data,  
                                   initial\_state=initial\_state,  
                                   dtype=tf.float32)

# create 2 LSTMCells  
rnn\_layers = [tf.compat.v1.nn.rnn\_cell.LSTMCell(size) for size in [128, 256]]  
  
# create a RNN cell composed sequentially of a number of RNNCells  
multi\_rnn\_cell = tf.compat.v1.nn.rnn\_cell.MultiRNNCell(rnn\_layers)  
  
# 'outputs' is a tensor of shape [batch\_size, max\_time, 256]  
# 'state' is a N-tuple where N is the number of LSTMCells containing a  
# tf.nn.rnn\_cell.LSTMStateTuple for each cell  
outputs, state = tf.compat.v1.nn.dynamic\_rnn(cell=multi\_rnn\_cell,  
                                   inputs=data,  
                                   dtype=tf.float32)

#### Args:

* **cell**: An instance of RNNCell.
* **inputs**: The RNN inputs. If time\_major == False (default), this must be a Tensor of shape:[batch\_size, max\_time, ...], or a nested tuple of such elements. If time\_major == True, this must be a Tensor of shape: [max\_time, batch\_size, ...], or a nested tuple of such elements. This may also be a (possibly nested) tuple of Tensors satisfying this property. The first two dimensions must match across all the inputs, but otherwise the ranks and other shape components may differ. In this case, input to cell at each time-step will replicate the structure of these tuples, except for the time dimension (from which the time is taken). The input to cellat each time step will be a Tensor or (possibly nested) tuple of Tensors each with dimensions [batch\_size, ...].
* **sequence\_length**: (optional) An int32/int64 vector sized [batch\_size]. Used to copy-through state and zero-out outputs when past a batch element's sequence length. So it's more for performance than correctness.
* **initial\_state**: (optional) An initial state for the RNN. If cell.state\_size is an integer, this must be a Tensor of appropriate type and shape [batch\_size, cell.state\_size]. If cell.state\_size is a tuple, this should be a tuple of tensors having shapes [batch\_size, s] for s in cell.state\_size.
* **dtype**: (optional) The data type for the initial state and expected output. Required if initial\_state is not provided or RNN state has a heterogeneous dtype.
* **parallel\_iterations**: (Default: 32). The number of iterations to run in parallel. Those operations which do not have any temporal dependency and can be run in parallel, will be. This parameter trades off time for space. Values >> 1 use more memory but take less time, while smaller values use less memory but computations take longer.
* **swap\_memory**: Transparently swap the tensors produced in forward inference but needed for back prop from GPU to CPU. This allows training RNNs which would typically not fit on a single GPU, with very minimal (or no) performance penalty.
* **time\_major**: The shape format of the inputs and outputs Tensors. If true, these Tensorsmust be shaped [max\_time, batch\_size, depth]. If false, these Tensors must be shaped [batch\_size, max\_time, depth]. Using time\_major = True is a bit more efficient because it avoids transposes at the beginning and end of the RNN calculation. However, most TensorFlow data is batch-major, so by default this function accepts input and emits output in batch-major form.
* **scope**: VariableScope for the created subgraph; defaults to "rnn".

#### Returns:

A pair (outputs, state) where:

* **outputs**: The RNN output Tensor.

If time\_major == False (default), this will be a Tensor shaped: [batch\_size, max\_time, cell.output\_size].

If time\_major == True, this will be a Tensor shaped: [max\_time, batch\_size, cell.output\_size].

Note, if cell.output\_size is a (possibly nested) tuple of integers or TensorShape objects, then outputs will be a tuple having the same structure as cell.output\_size, containing Tensors having shapes corresponding to the shape data in cell.output\_size.

* **state**: The final state. If cell.state\_size is an int, this will be shaped [batch\_size, cell.state\_size]. If it is a TensorShape, this will be shaped [batch\_size] + cell.state\_size. If it is a (possibly nested) tuple of ints or TensorShape, this will be a tuple having the corresponding shapes. If cells are LSTMCells state will be a tuple containing a LSTMStateTuple for each cell.

#### Raises:

* **TypeError**: If cell is not an instance of RNNCell.
* **ValueError**: If inputs is None or an empty list.
* **RuntimeError**: If not using control flow v2.

# tf.compat.v1.lite.experimental.nn.TFLiteLSTMCell

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TFLiteLSTMCell#top_of_page)
* [Class TFLiteLSTMCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TFLiteLSTMCell#class_tflitelstmcell)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TFLiteLSTMCell#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TFLiteLSTMCell#properties)
  + [graph](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TFLiteLSTMCell#graph)

## Class TFLiteLSTMCell

Long short-term memory unit (LSTM) recurrent network cell.

Defined in [lite/experimental/examples/lstm/rnn\_cell.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/lite/experimental/examples/lstm/rnn_cell.py).

This is used only for TfLite, it provides hints and it also makes the variables in the desired for the tflite ops (transposed and seaparated).

The default non-peephole implementation is based on:

https://pdfs.semanticscholar.org/1154/0131eae85b2e11d53df7f1360eeb6476e7f4.pdf

Felix Gers, Jurgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." IET, 850-855, 1999.

The peephole implementation is based on:

https://research.google.com/pubs/archive/43905.pdf

Hasim Sak, Andrew Senior, and Francoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." INTERSPEECH, 2014.

The class uses optional peep-hole connections, optional cell clipping, and an optional projection layer.

Note that this cell is not optimized for performance. Please use tf.contrib.cudnn\_rnn.CudnnLSTMfor better performance on GPU, or tf.contrib.rnn.LSTMBlockCell and tf.contrib.rnn.LSTMBlockFusedCell for better performance on CPU.

## \_\_init\_\_

\_\_init\_\_(  
    num\_units,  
    use\_peepholes=False,  
    cell\_clip=None,  
    initializer=None,  
    num\_proj=None,  
    proj\_clip=None,  
    num\_unit\_shards=None,  
    num\_proj\_shards=None,  
    forget\_bias=1.0,  
    state\_is\_tuple=True,  
    activation=None,  
    reuse=None,  
    name=None,  
    dtype=None  
)

Initialize the parameters for an LSTM cell.

#### Args:

* **num\_units**: int, The number of units in the LSTM cell.
* **use\_peepholes**: bool, set True to enable diagonal/peephole connections.
* **cell\_clip**: (optional) A float value, if provided the cell state is clipped by this value prior to the cell output activation.
* **initializer**: (optional) The initializer to use for the weight and projection matrices.
* **num\_proj**: (optional) int, The output dimensionality for the projection matrices. If None, no projection is performed.
* **proj\_clip**: (optional) A float value. If num\_proj > 0 and proj\_clip is provided, then the projected values are clipped elementwise to within [-proj\_clip, proj\_clip].
* **num\_unit\_shards**: Deprecated, will be removed by Jan. 2017. Use a variable\_scope partitioner instead.
* **num\_proj\_shards**: Deprecated, will be removed by Jan. 2017. Use a variable\_scope partitioner instead.
* **forget\_bias**: Biases of the forget gate are initialized by default to 1 in order to reduce the scale of forgetting at the beginning of the training. Must set it manually to 0.0 when restoring from CudnnLSTM trained checkpoints.
* **state\_is\_tuple**: If True, accepted and returned states are 2-tuples of the c\_state and m\_state. If False, they are concatenated along the column axis. This latter behavior will soon be deprecated.
* **activation**: Activation function of the inner states. Default: tanh.
* **reuse**: (optional) Python boolean describing whether to reuse variables in an existing scope. If not True, and the existing scope already has the given variables, an error is raised.
* **name**: String, the name of the layer. Layers with the same name will share weights, but to avoid mistakes we require reuse=True in such cases.
* **dtype**: Default dtype of the layer (default of None means use the type of the first input). Required when build is called before call. When restoring from CudnnLSTM-trained checkpoints, use CudnnCompatibleLSTMCell instead.

## Properties

### graph

DEPRECATED FUNCTION

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Stop using this property because tf.layers layers no longer track their graph.

### output\_size

### scope\_name

### state\_size

## Methods

### get\_initial\_state

get\_initial\_state(  
    inputs=None,  
    batch\_size=None,  
    dtype=None  
)

### zero\_state

zero\_state(  
    batch\_size,  
    dtype  
)

Return zero-filled state tensor(s).

#### Args:

* **batch\_size**: int, float, or unit Tensor representing the batch size.
* **dtype**: the data type to use for the state.

#### Returns:

If state\_size is an int or TensorShape, then the return value is a N-D tensor of shape [batch\_size, state\_size] filled with zeros.

If state\_size is a nested list or tuple, then the return value is a nested list or tuple (of the same structure) of 2-D tensors with the shapes [batch\_size, s] for each s in state\_size.

# tf.compat.v1.lite.experimental.nn.TfLiteRNNCell

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TfLiteRNNCell#top_of_page)
* [Class TfLiteRNNCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TfLiteRNNCell#class_tfliternncell)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TfLiteRNNCell#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TfLiteRNNCell#properties)
  + [graph](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/lite/experimental/nn/TfLiteRNNCell#graph)

## Class TfLiteRNNCell

The most basic RNN cell.

Defined in [lite/experimental/examples/lstm/rnn\_cell.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/lite/experimental/examples/lstm/rnn_cell.py).

This is used only for TfLite, it provides hints and it also makes the variables in the desired for the tflite ops.

## \_\_init\_\_

\_\_init\_\_(  
    num\_units,  
    activation=None,  
    reuse=None,  
    name=None,  
    dtype=None,  
    \*\*kwargs  
)

Initializes the parameters for an RNN cell.

#### Args:

* **num\_units**: int, The number of units in the RNN cell.
* **activation**: Nonlinearity to use. Default: tanh. It could also be string that is within Keras activation function names.
* **reuse**: (optional) Python boolean describing whether to reuse variables in an existing scope. Raises an error if not True and the existing scope already has the given variables.
* **name**: String, the name of the layer. Layers with the same name will share weights, but to avoid mistakes we require reuse=True in such cases.
* **dtype**: Default dtype of the layer (default of None means use the type of the first input). Required when build is called before call.
* **\*\*kwargs**: Dict, keyword named properties for common layer attributes, like trainable etc when constructing the cell from configs of get\_config().

#### Raises:

* **ValueError**: If the existing scope already has the given variables.

## Properties

### graph

DEPRECATED FUNCTION

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Stop using this property because tf.layers layers no longer track their graph.

### output\_size

### scope\_name

### state\_size

## Methods

### get\_initial\_state

get\_initial\_state(  
    inputs=None,  
    batch\_size=None,  
    dtype=None  
)

### zero\_state

zero\_state(  
    batch\_size,  
    dtype  
)

Return zero-filled state tensor(s).

#### Args:

* **batch\_size**: int, float, or unit Tensor representing the batch size.
* **dtype**: the data type to use for the state.

#### Returns:

If state\_size is an int or TensorShape, then the return value is a N-D tensor of shape [batch\_size, state\_size] filled with zeros.

If state\_size is a nested list or tuple, then the return value is a nested list or tuple (of the same structure) of 2-D tensors with the shapes [batch\_size, s] for each s in state\_size.

Module: tf.lite

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite#top_of_page)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite#classes)

Public API for tf.lite namespace.

Classes

[class Interpreter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Interpreter): Interpreter interface for TensorFlow Lite Models.

[class OpsSet](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/OpsSet): Enum class defining the sets of ops available to generate TFLite models.

[class Optimize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Optimize): Enum defining the optimizations to apply when generating tflite graphs.

[class RepresentativeDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/RepresentativeDataset): Representative dataset to evaluate optimizations.

[class TFLiteConverter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/TFLiteConverter): Converts a TensorFlow model into TensorFlow Lite model.

[class TargetSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/TargetSpec): Specification of target device.

# tf.lite.Interpreter

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Interpreter#top_of_page)
* [Class Interpreter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Interpreter#class_interpreter)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Interpreter#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Interpreter#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Interpreter#methods)

## Class Interpreter

Interpreter interface for TensorFlow Lite Models.

### Aliases:

* Class tf.compat.v1.lite.Interpreter
* Class tf.compat.v2.lite.Interpreter
* Class tf.lite.Interpreter

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

This makes the TensorFlow Lite interpreter accessible in Python. It is possible to use this interpreter in a multithreaded Python environment, but you must be sure to call functions of a particular instance from only one thread at a time. So if you want to have 4 threads running different inferences simultaneously, create an interpreter for each one as thread-local data. Similarly, if you are calling invoke() in one thread on a single interpreter but you want to use tensor() on another thread once it is done, you must use a synchronization primitive between the threads to ensure invoke has returned before calling tensor().

## \_\_init\_\_

\_\_init\_\_(  
    model\_path=None,  
    model\_content=None  
)

Constructor.

#### Args:

* **model\_path**: Path to TF-Lite Flatbuffer file.
* **model\_content**: Content of model.

#### Raises:

* **ValueError**: If the interpreter was unable to create.

## Methods

### allocate\_tensors

allocate\_tensors()

### get\_input\_details

get\_input\_details()

Gets model input details.

#### Returns:

A list of input details.

### get\_output\_details

get\_output\_details()

Gets model output details.

#### Returns:

A list of output details.

### get\_tensor

get\_tensor(tensor\_index)

Gets the value of the input tensor (get a copy).

If you wish to avoid the copy, use tensor(). This function cannot be used to read intermediate results.

#### Args:

* **tensor\_index**: Tensor index of tensor to get. This value can be gotten from the 'index' field in get\_output\_details.

#### Returns:

a numpy array.

### get\_tensor\_details

get\_tensor\_details()

Gets tensor details for every tensor with valid tensor details.

Tensors where required information about the tensor is not found are not added to the list. This includes temporary tensors without a name.

#### Returns:

A list of dictionaries containing tensor information.

### invoke

invoke()

Invoke the interpreter.

Be sure to set the input sizes, allocate tensors and fill values before calling this. Also, note that this function releases the GIL so heavy computation can be done in the background while the Python interpreter continues. No other function on this object should be called while the invoke() call has not finished.

#### Raises:

* **ValueError**: When the underlying interpreter fails raise ValueError.

### reset\_all\_variables

reset\_all\_variables()

### resize\_tensor\_input

resize\_tensor\_input(  
    input\_index,  
    tensor\_size  
)

Resizes an input tensor.

#### Args:

* **input\_index**: Tensor index of input to set. This value can be gotten from the 'index' field in get\_input\_details.
* **tensor\_size**: The tensor\_shape to resize the input to.

#### Raises:

* **ValueError**: If the interpreter could not resize the input tensor.

### set\_tensor

set\_tensor(  
    tensor\_index,  
    value  
)

Sets the value of the input tensor. Note this copies data in value.

If you want to avoid copying, you can use the tensor() function to get a numpy buffer pointing to the input buffer in the tflite interpreter.

#### Args:

* **tensor\_index**: Tensor index of tensor to set. This value can be gotten from the 'index' field in get\_input\_details.
* **value**: Value of tensor to set.

#### Raises:

* **ValueError**: If the interpreter could not set the tensor.

### tensor

tensor(tensor\_index)

Returns function that gives a numpy view of the current tensor buffer.

This allows reading and writing to this tensors w/o copies. This more closely mirrors the C++ Interpreter class interface's tensor() member, hence the name. Be careful to not hold these output references through calls to allocate\_tensors() and invoke(). This function cannot be used to read intermediate results.

#### Usage:

interpreter.allocate\_tensors()  
input = interpreter.tensor(interpreter.get\_input\_details()[0]["index"])  
output = interpreter.tensor(interpreter.get\_output\_details()[0]["index"])  
for i in range(10):  
  input().fill(3.)  
  interpreter.invoke()  
  print("inference %s" % output())

Notice how this function avoids making a numpy array directly. This is because it is important to not hold actual numpy views to the data longer than necessary. If you do, then the interpreter can no longer be invoked, because it is possible the interpreter would resize and invalidate the referenced tensors. The NumPy API doesn't allow any mutability of the the underlying buffers.

#### WRONG:

input = interpreter.tensor(interpreter.get\_input\_details()[0]["index"])()  
output = interpreter.tensor(interpreter.get\_output\_details()[0]["index"])()  
interpreter.allocate\_tensors()  # This will throw RuntimeError  
for i in range(10):  
  input.fill(3.)  
  interpreter.invoke()  # this will throw RuntimeError since input,output

#### Args:

* **tensor\_index**: Tensor index of tensor to get. This value can be gotten from the 'index' field in get\_output\_details.

#### Returns:

A function that can return a new numpy array pointing to the internal TFLite tensor state at any point. It is safe to hold the function forever, but it is not safe to hold the numpy array forever.

# tf.lite.OpsSet

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/OpsSet#top_of_page)
* [Class OpsSet](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/OpsSet#class_opsset)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/OpsSet#aliases)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/OpsSet#class_members)

## Class OpsSet

Enum class defining the sets of ops available to generate TFLite models.

### Aliases:

* Class tf.compat.v1.lite.OpsSet
* Class tf.compat.v2.lite.OpsSet
* Class tf.lite.OpsSet

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

WARNING: Experimental interface, subject to change.

## Class Members

* SELECT\_TF\_OPS
* TFLITE\_BUILTINS
* TFLITE\_BUILTINS\_INT8

# tf.lite.Optimize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Optimize#top_of_page)
* [Class Optimize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Optimize#class_optimize)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Optimize#aliases)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/Optimize#class_members)

## Class Optimize

Enum defining the optimizations to apply when generating tflite graphs.

### Aliases:

* Class tf.compat.v1.lite.Optimize
* Class tf.compat.v2.lite.Optimize
* Class tf.lite.Optimize

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

Some optimizations may come at the cost of accuracy.

## Class Members

* DEFAULT
* OPTIMIZE\_FOR\_LATENCY
* OPTIMIZE\_FOR\_SIZE

# tf.lite.RepresentativeDataset

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/RepresentativeDataset#top_of_page)
* [Class RepresentativeDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/RepresentativeDataset#class_representativedataset)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/RepresentativeDataset#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/RepresentativeDataset#__init__)

## Class RepresentativeDataset

Representative dataset to evaluate optimizations.

### Aliases:

* Class tf.compat.v1.lite.RepresentativeDataset
* Class tf.compat.v2.lite.RepresentativeDataset
* Class tf.lite.RepresentativeDataset

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

A representative dataset that can be used to evaluate optimizations by the converter. E.g. converter can use these examples to estimate (min, max) ranges by calibrating the model on inputs. This can allow converter to quantize a converted floating point model.

## \_\_init\_\_

\_\_init\_\_(input\_gen)

Creates a representative dataset.

#### Args:

* **input\_gen**: an input generator that can be used to generate input samples for the model. This must be a callable object that returns an object that supports the iter() protocol (e.g. a generator function). The elements generated must have same type and shape as inputs to the model.

# tf.lite.TargetSpec

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/TargetSpec#top_of_page)
* [Class TargetSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/TargetSpec#class_targetspec)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/TargetSpec#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lite/TargetSpec#__init__)

## Class TargetSpec

Specification of target device.

### Aliases:

* Class tf.compat.v1.lite.TargetSpec
* Class tf.compat.v2.lite.TargetSpec
* Class tf.lite.TargetSpec

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

Details about target device. Converter optimizes the generated model for specific device.

#### Attributes:

* **supported\_ops**: Experimental flag, subject to change. Set of OpsSet options supported by the device. (default set([OpsSet.TFLITE\_BUILTINS]))

## \_\_init\_\_

\_\_init\_\_(supported\_ops=None)